Simulacija primjene digitalizatora u korekciji putanje alata prema odstupanju položaja i orijentacije kod stezanja pripremka

Repalust, Luka

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Luka Repalust

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor: Student:

Izv. prof. dr. sc. Tomislav Staroveški, dipl. ing. Luka Repalust

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 stručnim savjetima, pomoći i strpljenju tijekom izrade diplomskog rada.

Također, zahvaljujem se asistentima Luki Drobilu mag. ing. mech. i Dori Bagarić mag. ing. mech. na podijeljenom znanju, stručnim savjetima i pomoći, te kolegi Mihovilu Leginu na pomoći i suradnji tijekom izrade diplomskog rada.

Posebnu zahvalu upućujem svojoj obitelji i prijateljima na razumijevanju, bezuvjetnoj podršci i poticanju tijekom cijelog studija.

Luka Repalust

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: **Luka Repalust** JMBAG: 0035220017

Naslov rada na hrvatskom jeziku: **Simulacija primjene digitalizatora u korekciji putanje alata prema odstupanju položaja i orijentacije kod stezanja pripremka**

Naslov rada na engleskom jeziku: **Simulation of 3D scanner-based tool path correction for position and orientation of the workpiece clamping**

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 većih dimenzija. U sklopu projektnih aktivnosti planira se istražiti utjecaj odstupanja položaja i orijentacije kod stezanja predmetnih ispitnih uzoraka na kvalitetu obrađene površine.
U radu je potrebno:

1. Na osnovi 3D modela (sklopa) zadanog ispitnog uzorka i stezne naprave izraditi 3D modele sklopova u nekoliko varijanti u kojima su ispitni uzorci zamaknuti, odnosno zakrenuti u odnosu na ishodište koordinatnog sustava predviđenog za obradu. 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 sklop ispitnog uzorka koristiti kao idealni (CAD) model, a pripremljeni (zamaknuti) ispitni uzorci 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 zamaknutim površinama uzorka.
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

POPIS SLIKA

POPIS TABLICA

SAŽETAK

Tema ovog diplomskog rada je simulacija primjene digitalizatora za korekciju putanje alata prema odstupanju položaja i orijentacije pripremka prilikom stezanja. U radu je izrađen programski kod koji na temelju rezultata analize odstupanja korigira položaj alata robota prilikom brušenja ispitnih uzoraka. Mjerni postav za analizu rezultata digitalizacije formiran je u programskom paketu GOM Inspect Pro, gdje je zadani sklop stezne naprave i ispitnog uzorka korišten kao idealni CAD model, a sklop sa zamaknutim ispitnim uzorcima kao simulirani rezultati digitalizacije. Rezultati analize odstupanja koordinata točaka i vektora normala na površinu zamaknutih ispitnih uzoraka korišteni su kao ulazni podaci na temelju kojih se ažurira RAPID kod. Na kraju su izvorni program i program s korigiranim položajima alata robota simulirani u programskom paketu ABB RobotStudio te je ustanovljeno da programski kod ispravno korigira putanju alata.

Ključne riječi: industrijski robot, korekcija putanje alata, GOM Inspect

SUMMARY

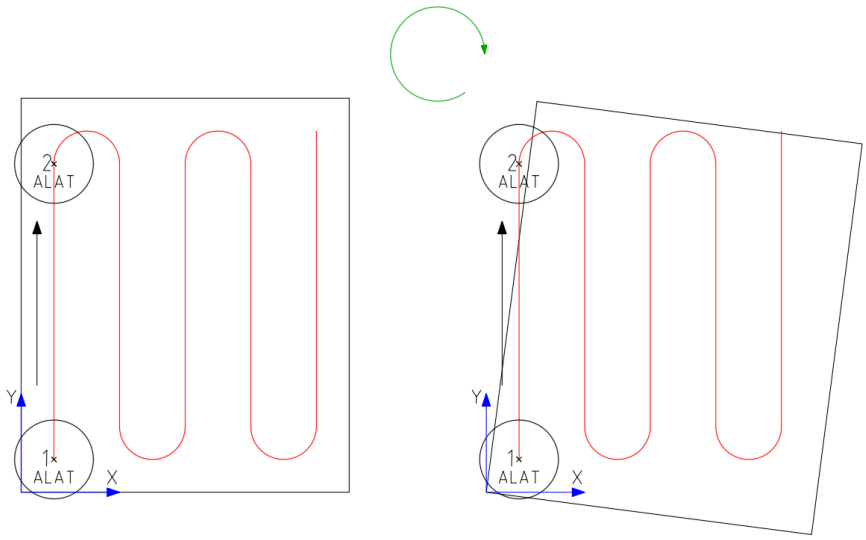
The subject of this Master's thesis is the simulation of a 3D scanner-based tool path correction method required due to possible position and orientation deviations of the workpiece during clamping. In this thesis, a program code was developed that, based on the results of the deviation analysis, corrects the position of the robot's tool for the sanding operations performed on the workpiece. A measuring setup for analyzing 3D scanning results was formed in a software GOM Inspect Pro, where the specified assembly of the workpiece and the clamping device defined the ideal CAD model and 3D models with displaced workpieces were used to simulate the scanning results. The results of the coordinates and vector normals deviation analysis were used as input data to update the RAPID code. Finally, both the initial and corrected programs were simulated in the ABB RobotStudio software package, and it was determined that the program code adjusts the tool path correctly.

Key words: industrial robot, tool path correction, GOM Inspect

1. UVOD

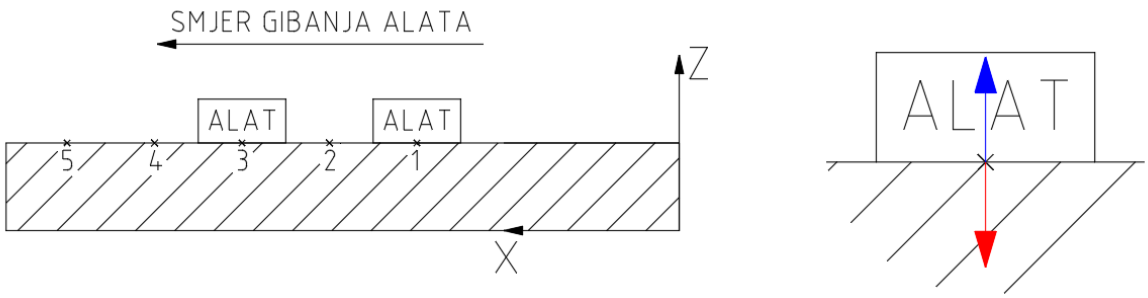
U suvremenim obradnim sustavima sve je češća primjena digitalizatora, odnosno uređaja za beskontaktno 3D skeniranje. Primjenom digitalizatora omogućeno je automatizirano prikupljanje i analiza podataka, a u kombinaciji sa softverima za analizu rezultata digitalizacije moguće je precizno i brzo analizirati složene geometrijske značajke proizvoda. Zbog navedenih karakteristika, digitalizatori se nerijetko primjenjuju u području kontrole kvalitete. Iako se kontrola kvalitete najčešće odnosi na sam kraj proizvodnog procesa, digitalizatore je moguće implementirati i u ranijim fazama izrade proizvoda. Na taj način se prije same obrade mogu odrediti moguće devijacije orijentacije i položaja neispravno stegnutih pripremaka, te detektirati mogući defekti na istima. Pravovremenom identifikacijom navedenih problema te korekcijom putanje alata, moguće je unatoč odstupanju pripremka postići traženu kvalitetu obrađene površine te smanjiti škart.

U ovom radu biti će opisana primjena digitalizatora za korekcije putanja alata prilikom brušenja ispitnih uzoraka. Ukoliko je pripremak prilikom stezanja zamaknut, zakrenut ili deformiran, površina na kojoj će se vršiti obrada odstupa u odnosu na ishodište koordinatnog sustava predviđenog za obradu. Na slici 1 prikazan je slučaj u kojem je obradak zakrenut u odnosu na ishodište koordinatnog sustava obrade. Zbog zakreta obratka u XY ravnini, putanja alata označena crvenom bojom se ne nalazi na površini predviđenoj za obradu. Uslijed navedene situacije, prilikom kretanja alata od točke 1 do točke 2, površina alata nije u potpunosti u kontaktu s obratkom te se ne obrađuju dijelovi površine koji su predviđeni za obradu.



Slika 1. Zakret obratka u odnosu na ishodište koordinatnog sustava predviđenog za obradu

Na slici 2 prikazan je primjer gibanja alata po površini pravilno pozicioniranog i orijentiranog obratka u XZ ravnini koordinatnog sustava obrade. Brojevima od 1 do 5 označene su koordinate točaka putanje alata, crvenom strelicom je označen vektor normale na površinu alata, a plavom strelicom je označen vektor normale na površinu ispitnog uzorka. U navedenom primjeru su vektori normala istog smjera, ali suprotne orijentacije te je površina alata u potpunosti u kontaktu s površinom ispitnog uzorka.

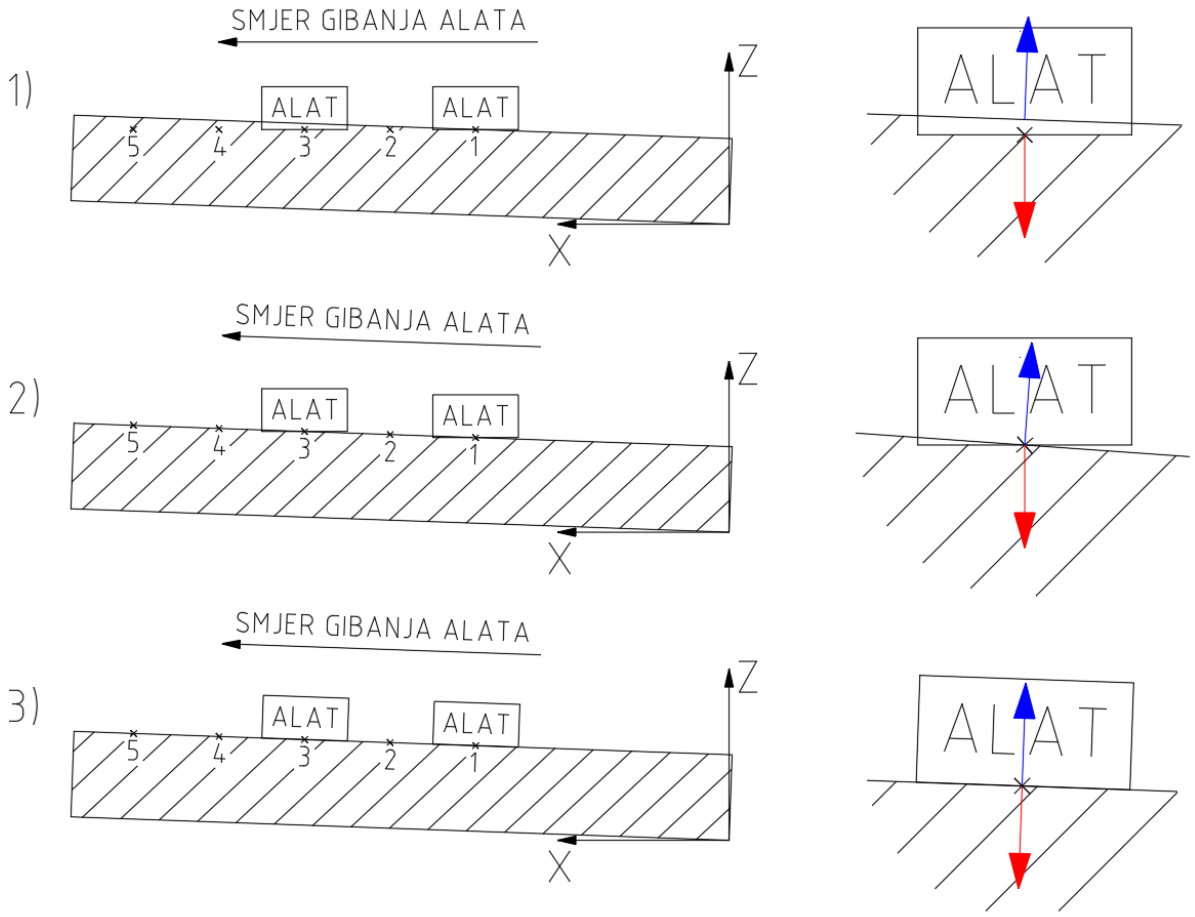


Slika 2. Brušenje pravilno pozicioniranog i orijentiranog obratka

Prilikom zakreta obratka u XZ ili YZ ravnini moguće je da se ne postigne dovoljna sila ili da se djeluje prevelikom silom na obradak, što može imati utjecaj na kvalitetu obrađene površine te vijek trajanja alata.

Na slici 3 je u primjeru 1 prikazan obradak zakrenut u XZ ravnini koordinatnog sustava obrade. Zbog odstupanja obratka, koordinate točka putanje alata ne nalaze se na površini predviđenoj za obradu. U tom slučaju, ovisno o veličini odstupanja, prilikom brušenja može doći do neujednačene kvalitete obrađene površine, oštećenja obratka ili alata te zastoja procesa zbog prekomjerne sile. Primjer 2 prikazuje gibanje alata po korigiranim koordinatama točaka putanje. Iako se točke putanje alata nalaze na površini ispitnog uzorka, vektori normala na površinu alata i vektori normala na površinu ispitnog uzorka različitog su smjera. Površina alata nije u cijelosti u kontaktu s površinom ispitnog uzorka te se ovisno o veličini odstupanja neće postići kvaliteta obrađene površine kao u primjeru na slici 2.

Da bi se postigla tražena kvaliteta obrađene površine, potrebno je osim koordinata točaka putanje korigirati i nagib alata. U tom slučaju (primjer 3), vektor normale na površinu alata istog je smjera, ali suprotne orijentacije u odnosu na vektor normale na površinu ispitnog uzorka. Tada je površina alata u cijelosti u zahvatu s površinom ispitnog uzorka te se može očekivati kvaliteta obrađene površine kao i u primjeru na slici 2.



Slika 3. Korekcija putanje alata

Cilj ovog diplomskog rada je izraditi programski kod koji će na temelju analize rezultata digitalizacije korigirati putanju alata prema odstupanju položaja i orijentacije pripremka prilikom stezanja. U programskom paketu za analizu rezultata digitalizacije GOM Inspect Pro automatiziran je postupak analize odstupanja kontrolnih točaka i pripadajućih vektora normala na površinu zamaknutog ispitnog uzorka, a dobiveni rezultati korišteni su kao ulazni podaci u programskom kodu za korekciju putanje alata. Korekcijom koordinata točaka i orijentacije vrha alata robota izbjegle bi se prethodno opisane greške koje nastaju prilikom neispravnog stezanja pripremka te bi se osigurala ujednačenost kvalitete obrađene površine. Navedenim rješenjem bi se detektirane greške uslijed stezanja mogle namjerno zanemariti, a zbog toga bi se također mogle primijeniti jeftinije i jednostavnije stezne naprave.

2. DIGITALIZATOR

Digitalizatori su uređaji koji služe za pretvaranje podataka o geometriji stvarnog objekta u digitalni oblik. Postupkom digitalizacije prikupljaju se podaci o položaju točaka na površini, a pomoću dobivenih podataka generira se oblak točaka koje softver povezuje u mrežu s ciljem stvaranja 3D modela skeniranog objekta. Metode digitalizacije dijele se na kontaktne i beskontaktne. Virtualni prikaz se kontaktnom metodom ostvaruje prolaskom ticala po površini objekta, dok se beskontaktnom metodom ostvaruje skeniranjem površine stvarnog objekta [1]. Unatoč različitim metodama digitalizacije, u opsegu ovog diplomskog rada će se pod pojmom digitalizator smatrati beskontaktni optički mjerni uređaj, odnosno 3D skener.

2.1. Značajke digitalizatora

Zavisno o korištenoj metodi, digitalizacija omogućava brzo i precizno prikupljanje informacija o milijunima točaka u vrlo kratkom vremenskom periodu. Postupkom digitalizacije generira se oblak točaka, a primjenom softvera za analizu 3D podataka moguće je provjeriti tolerancije oblika i položaja razmatranih značajki digitaliziranog objekta. Zbog navedenog, primjenom digitalizatora te prethodno postavljenih mjernih izvještaja, znatno se ubrzava postupak mjerenja većeg broja objekata te objekata složenih geometrijskih značajki u odnosu na konvencionalne mjerne metode pri kojima se najčešće svaka od dimenzija mjeri zasebno. S obzirom da je digitalizacija beskontaktna mjerna metoda, pomoću nje je omogućena analiza krhkih objekata te objekata koji se lako deformiraju. Ograničenja digitalizatora pojavljuju se prilikom skeniranja vrlo tamnih, transparentnih ili visoko reflektirajućih površina, uslijed čega postoji problem premalog kontrasta, refleksije i apsorpcije svjetlosnih zraka. Predmetni efekti mogu znatno smanjiti kvalitetu rezultata skeniranja ili čak u potpunosti onemogućiti postupak digitalizacije. U takvim slučajevima, razmatrani uzorak se mora pripremiti nanošenjem tankog sloja boje ili praha. Automatizacijom postupka digitalizacije, moguće je vršiti kontrolu i mjerenje objekata u atmosferi opasnoj za zdravlje ljudi, a uz to, smanjuje se i potreban broj djelatnika [2].

2.2. Primjena digitalizatora

Industrije u kojima se digitalizatori najčešće upotrebljavaju su strojarstvo, arhitektura, medicina, zrakoplovna te automobilska industrija [2]. U strojarstvu, postupak digitalizacije je

značajan za kontrolu kvalitete, provjeru složenih geometrijskih značajki razmatranog objekta, izradu CAD modela, povratno inženjerstvo, kontrolu trošenja alata te brzu izradu prototipa [2].

Jedna od najčešćih primjena digitalizatora je u području kontrole kvalitete. Glavni razlog tome su sve veći zahtjevi tržišta za izradom velike količine proizvoda visoke i ujednačene kvalitete. Implementacijom digitalizatora omogućava se automatizacija postupka kontrole kvalitete, čime se značajno ubrzava proces provjere točnosti dimenzija te se reducira mogućnost ljudske greške. Primjenom odgovarajućeg softvera, te usporedbom skeniranih objekata s referentnim CAD modelom, moguće je vrlo brzo provjeriti nalaze li se dijelovi proizvoda u propisanim tolerancijama. Primjenom digitalizatora omogućeno je prikupljanje velikog broja informacija o dimenzijama objekta istovremeno, čime se uvelike povećava produktivnost postupka kontrole kvalitete.

S obzirom na mogućnost brzog i preciznog prikupljanja podataka o površini objekta, primjenom digitalizatora znatno se olakšava izrada CAD modela složene geometrije. Zbog navedenog, primjena digitalizatora je naročito pogodna u području povratnog inženjerstva. Pretvorbom skeniranog objekta u odgovarajući format zapisa, u nekom od CAD softvera moguće je na digitalnom prikazu objekta obavljati potrebne modifikacije. U slučaju da je površina proizvoda na kojem se vrše modifikacije složene geometrije, digitalizacijom se znatno štedi vrijeme u odnosu na 3D modeliranje. Dobiveni se CAD modeli mogu koristiti za izradu dijelova na CNC strojevima ili 3D printerima, a ponovnom primjenom digitalizatora moguće je brzo provjeriti točnost dimenzija dobivenih prototipa [2].

2.3. Beskontaktne metode digitalizacije

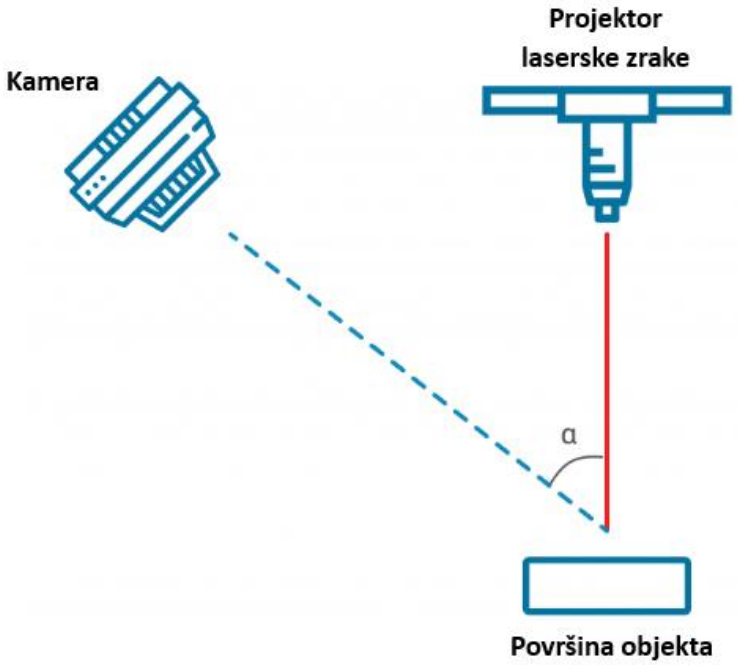
Mnogobrojne prednosti beskontaktnih digitalizatora razlog su sve češće implementacije ove tehnologije u proizvodne sustave. Zbog navedenog, ovisno o potrebi, došlo je do razvoja velikog broja različitih digitalizatora koji se razlikuju u metodama njihovog rada. Beskontaktne metode digitalizacije dijele se na lasersko 3D skeniranje, 3D skeniranje pulsirajućim laserskim zrakama, 3D skeniranje strukturiranim svjetlom te fotogrametriju [3].

2.3.1. Lasersko 3D skeniranje

Metoda laserskog 3D skeniranja zasniva se na principu triangulacije. Na površinu objekta projicira se laserska zraka koja se snima kamerom. Metoda se naziva triangulacija jer kamera, projektor laserske zrake i laserska točka projicirana na površinu objekta čine trokut (slika 4).

Budući da su poznate vrijednosti udaljenosti između kamere i projektora te kut pod kojim je projektor montiran, mjerenjem položaja laserske zrake u vidnom polju kamere dolazi se do udaljenosti između projektora laserske zrake te površine ispitnog uzorka [4].

Zbog jednostavne izvedbe i mobilnosti, laserski 3D skeneri nerijetko se primjenjuju za potrebe kontrole kvalitete [2].



Slika 4. Lasersko 3D skeniranje bazirano na principu triangulacije [5]

2.3.2. 3D skeniranje pulsirajućim laserskim zrakama

3D skeniranje pulsirajućim laserskim zrakama (eng. *Time-of-flight*) bazira se na mjerenju vremena potrebnog da se emitirana laserska zraka odbije od površine ispitnog uzorka i vrati do detektora. S obzirom da je brzina svjetlosti *c* poznata veličina, njenim množenjem s izmjerenim vremenom dobiva se podatak o udaljenosti lasera i površine ispitnog uzorka. Zbog navedenog, preciznost ove metode uvelike ovisi o točnosti izmjerenog vremena. S ciljem skeniranja cijele površine ispitnog uzorka, ovaj tip 3D skenera opremljen je zrcalima koja služe za preusmjeravanje laserskih zraka. *Time-of-flight* 3D skeneri imaju mogućnost prikupljanja informacija od 10000 do 100000 točaka svake sekunde [4].

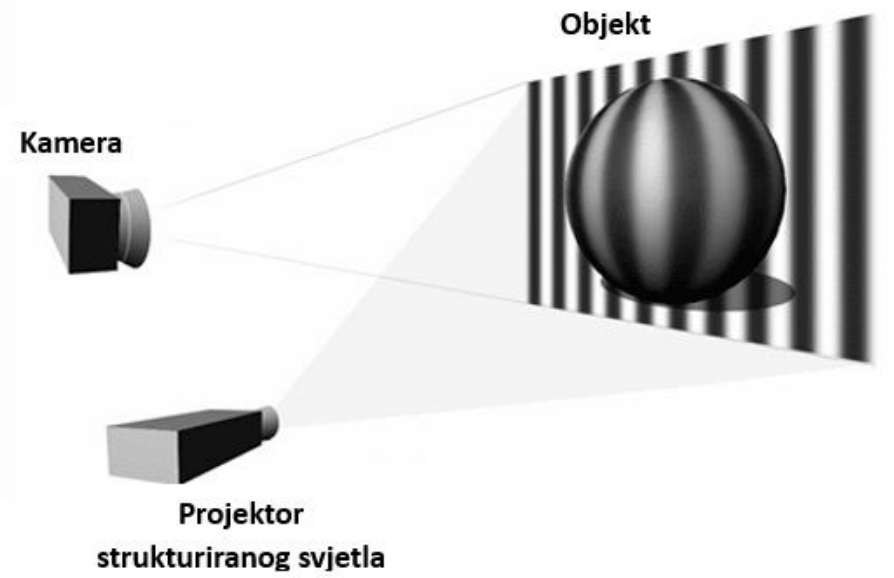
Metodom 3D skeniranja pulsirajućim laserskim zrakama moguće je skenirati objekte na znatno većim udaljenostima u odnosu na lasersko 3D skeniranje baziranom na triangulaciji. Unatoč tome, na većim udaljenostima su manje precizni pa se zbog toga većinom primjenjuju za 3D skeniranje u građevinarstvu, gdje su zahtjevi za točnošću obično manji nego u području

strojarstva. Tijekom skeniranja objekata velikih dimenzija, na površinu je često potrebno projicirati nekoliko milijuna točaka te je zbog toga ova metoda sporija u odnosu na ostale beskontaktne metode 3D skeniranja [4].

2.3.3. 3D skeniranje strukturiranim svjetlom

Uređaji za 3D skeniranje strukturiranim svjetlom sastoje se od projektora te jedne ili najčešće dvije kamere. Projektor na površinu objekta projicira uzorke paralelnih linija, a projicirani uzorci se snimaju kamerom (slika 5). Postupak 3D skeniranja strukturiranim svjetlom bazira se na metodi triangulacije, a udaljenost od površine razmatranog objekta određuje se analizom značajki vidljivih na jednoj ili obje kamere koje su nastale osvjetljavanjem površine razmatranog uzorka strukturiranim izvorom svjetlosti.

S obzirom da ova metoda 3D skeniranja omogućuje prikupljanje informacija o većem broju točaka istovremeno, njena najznačajnija prednost je brzina [6].



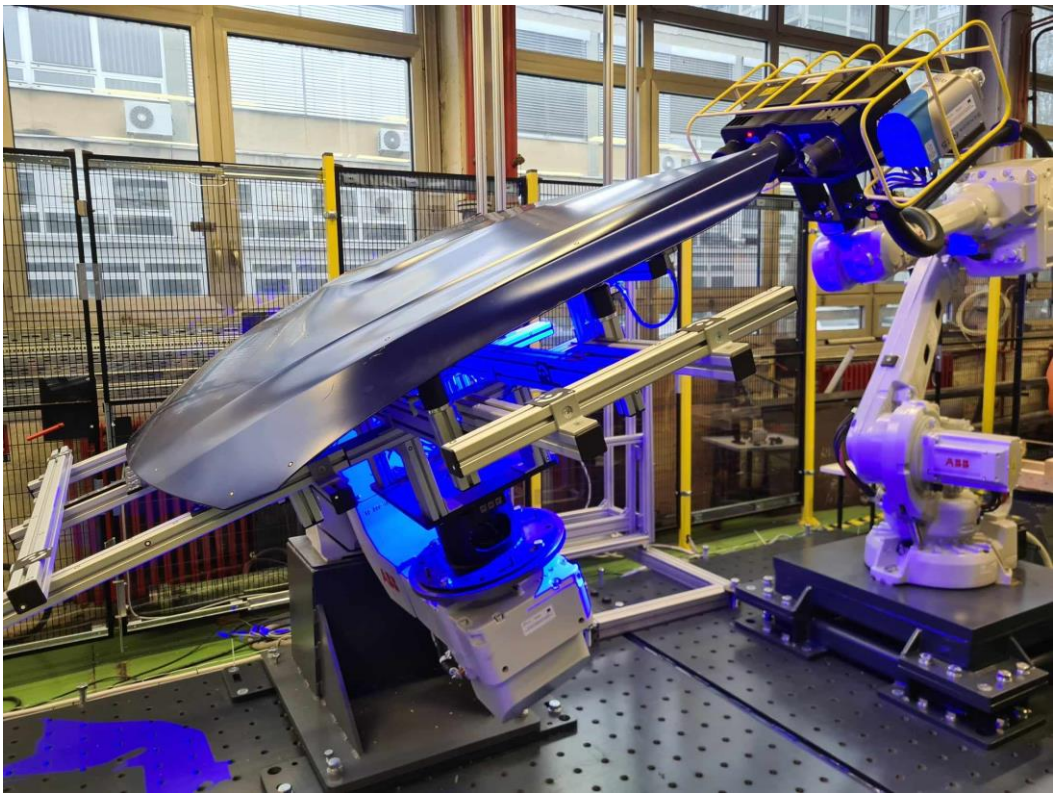
Slika 5. Metoda 3D skeniranja strukturiranim svjetlom [7]

2.3.4. Fotogrametrija

Metodom fotogrametrije digitalni prikaz nastaje fotografiranjem objekta iz različitih pozicija, pri čemu softver preklapa fotografije na način da identificira iste značajke na površini realnog objekta snimljene iz različitih položaja. Kvaliteta dobivenih rezultata uvelike ovisi o rezoluciji kamere te kvaliteti i optičkoj rezoluciji objektiva [8].

3. ROBOTSKI OBRADNI SUSTAV

U sklopu projekta ARCOPS na Katedri za alatne strojeve Fakulteta strojarstva i brodogradnje u Zagrebu razvijen je robotski obradni sustav za brušenje tankostjenih pozicija većih dimenzija. Obrada zadanog ispitnog uzorka (ploča 500x600x5 mm) provoditi će se na robotu ugrađenim u ćeliju, tipa ABB IRB 6660-205/1.9. Za provjeru odstupanja položaja i orijentacije prilikom stezanja ispitnih uzoraka, robotska ćelija je opremljena optičkim mjernim uređajem ATOS 5X montiranim na industrijski robot ABB IRB 4600-40/2.55 (slika 6). Između navedenih industrijskih robota nalazi se okretno-nagibni stol ABB IRBP na koji je pričvršćena stezna naprava za ispitni uzorak.



Slika 6. Optički mjerni uređaj ATOS 5X montiran na industrijskom robotu ABB IRB 4600-40/2.55

3.1. ABB IRB 6660-205/1.9

ABB IRB 6660-205/1.9 (slika 7) je industrijski robot švicarske tvrtke ABB predviđen za određene operacije glodanja i brušenja, odnosno za operacije obrade odvajanjem čestica [9].

Slika 7. ABB IRB 6660-205/1.9 [10]

Robot ABB IRB 6660-205/1.9 ima šest stupnjeva slobode gibanja, doseg od 1,93 m te nosivost od 205 kg. Tehničke karakteristike robota ABB IRB 6660-205/1.9 prikazane su u tablici 1.

Tablica 1. Tehničke karakteristike robota ABB IRB 6660-205/1.9 [9]

Doseg	1,93 m
Nosivost	205 kg
Broj stupnjeva slobode gibanja	6
Ponovljivost	0,07 mm
Masa robota	1730 kg
Stupanj IP zaštite	67

Robot se programira izravno putem operatorskog panela povezanim na upravljački sustav robota IRC5 (slika 8) ili *off-line* u programskom paketu ABB RobotStudio.

Slika 8. Upravljački sustav IRC5 [11]

3.2. ABB IRB 4600-40/2.55

ABB IRB 4600-40/2.55 (slika 9) je industrijski robot namijenjen za potrebe zavarivanja, pakiranja, manipulacije materijalima te laserskog rezanja. Navedeni robot ima šest stupnjeva slobode gibanja, nosivost od 40 kg te doseg od 2550 mm. Za upravljanje robotom koristi se upravljački sustav IRC5 [12].

Slika 9. ABB IRB 4600-40/2.55 [13]

U tablici 2 prikazane su tehničke karakteristike robota ABB IRB 4600-40/2.55.

Tablica 2. Tehničke karakteristike robota ABB IRB 4600-40/2.55 [12]

Doseg	2,55 m
Nosivost	40 kg
Broj stupnjeva slobode gibanja	6
Ponovljivost	0,06 mm
Masa robota	465 kg
Stupanj IP zaštite	67

3.2.1. ATOS 5X

ATOS 5X je optički mjerni uređaj njemačke tvrtke GOM *metrology GmbH* namijenjen za brzo skeniranje u ručnim ili automatiziranim aplikacijama. Glava mjernog uređaja ATOS 5X (slika 10) sastoji se od dvije kamere, te laserskog izvora koji služi za projiciranje strukturiranog svjetlosnog uzorka. Mjerni volumen uređaja se može mijenjati izmjenom odgovarajućih parova objektiva, a raspoloživi mjerni volumeni su 320x240x240 mm^3, 700x530x520 mm^3 te 1000x750x750 mm^3 [14].

Laserski izvor generira monokromatsku plavu svjetlost valne duljine u rasponu od 440 nm do 470 nm koja nije česta u prirodi, a kamere imaju ugrađene filtre koji propuštaju samo te valne duljine. Na taj način se ostvaruje manja osjetljivost uređaja na ambijentalnu svjetlost u prostoru gdje se koristi.

Slika 10. Optički mjerni uređaj ATOS 5X [15]

3.3. ABB IRBP A500

ABB IRBP A500 (slika 11) je okretno-nagibni stol koji u robotskoj ćeliji ARCOPS služi za prihvat stezne naprave s obratkom, te kao dodatni rotacijski posmični prigon. Ima nosivost do 500 kg te je namijenjen za procese kod kojih se obrađuju obratci velikih dimenzija. Dvije rotacijske osi omogućuju željenu orijentaciju alata naspram obratka u slučajevima kada obradak ima veće gabaritne dimenzije ili složeniju geometriju [16]. Za upravljanje okretno-nagibnim stolom koristi se, kao i za robote, upravljački sustav IRC5.

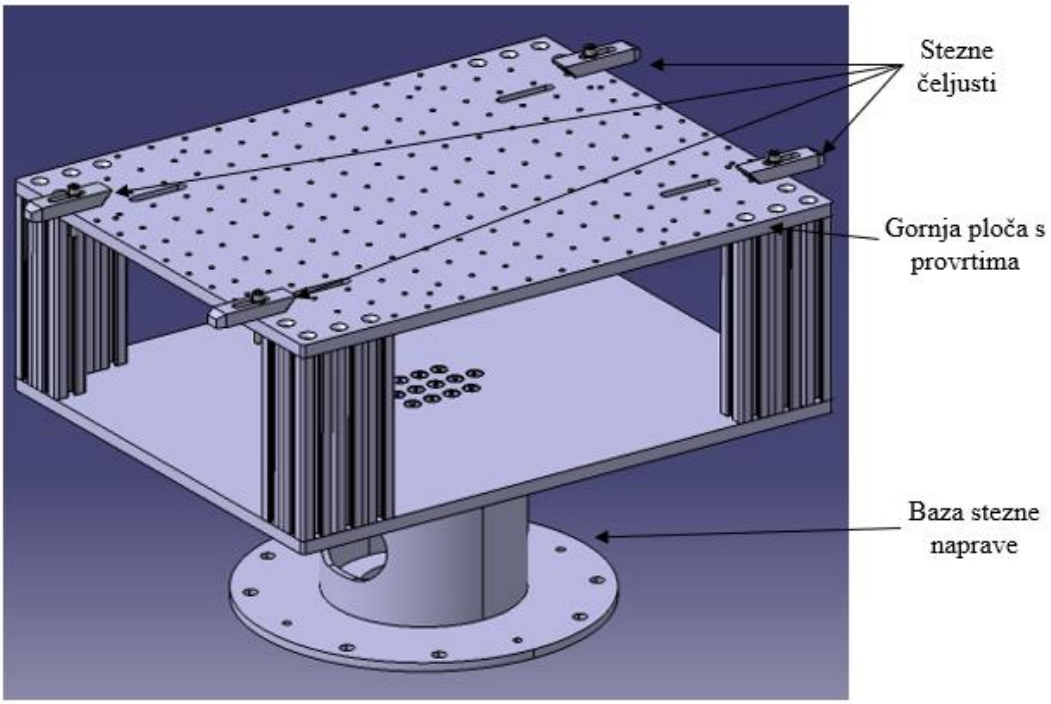
Slika 11. Okretno-nagibni stol ABB IRBP A500 [16]

4. GENERIRANJE PUTANJE ALATA

U svrhu generiranja putanje alata za brušenje ispitnog uzorka na industrijskom robotu ABB IRB 6660-205/1.9 korišteni su CAD/CAM sustav CATIA V5R21 te programski paketi RoboDK i ABB RobotStudio. U nastavku poglavlja opisan je 3D model sklopa stezne naprave i ispitnog uzorka te postupak generiranja NC programa za brušenje ispitnih uzoraka.

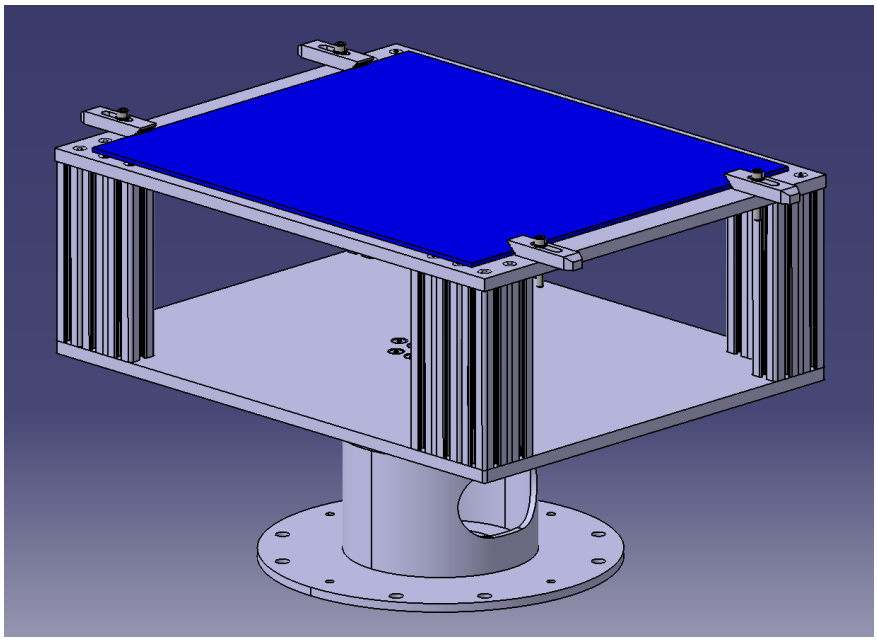
4.1. Ispitni uzorak

3D modeli korištenih ispitnih uzoraka izrađeni su u CAD/CAM sustavu CATIA V5R21. Za stezanje ispitnih uzoraka primijenjena je stezna naprava čija baza omogućava montažu na okretno-nagibni stol (slika 12). Na gornjoj ploči stezne naprave nalaze se provrti s urezanim navojima pomoću kojih je omogućeno fiksiranje ispitnih uzoraka steznim čeljustima i vijcima.



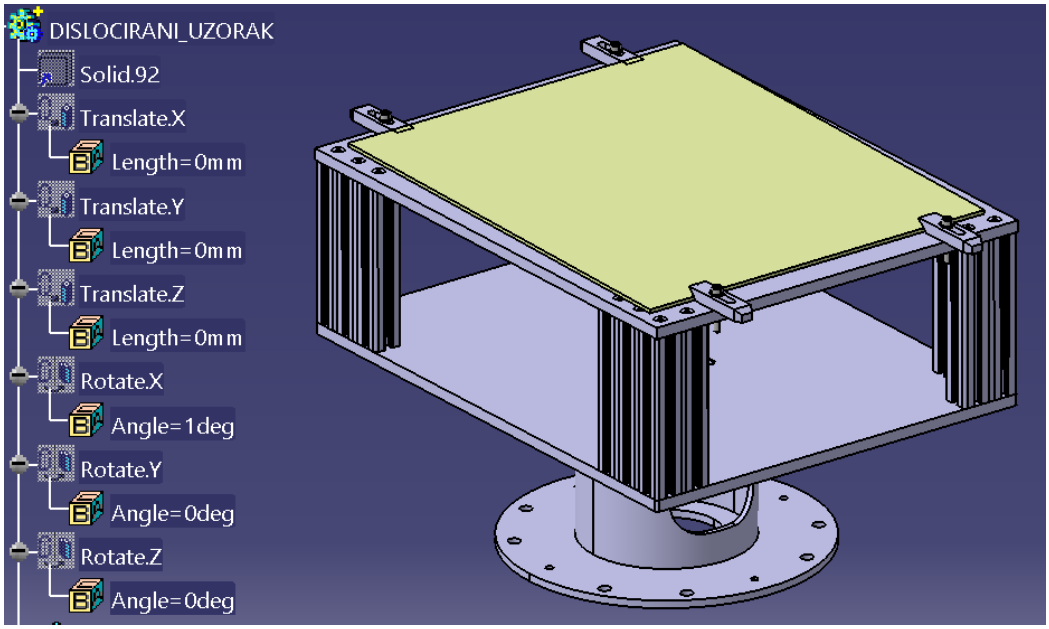
Slika 12. 3D model stezne naprave

Kao ispitni uzorak izrađen je 3D model ploče dimenzija 500x600x5 mm. Slika 13 prikazuje 3D model pravilno pozicioniranog uzorka na steznoj napravi koji će se koristiti kao referentni CAD model, u odnosu na koji će se analizirati odstupanja pogrešno stegnutih ispitnih uzoraka.



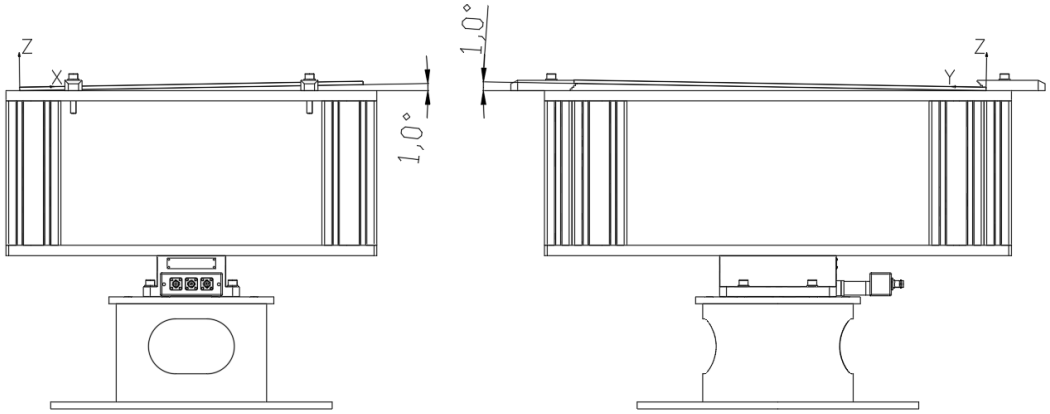
Slika 13. Referentni sklop ispitnog uzorka i stezne naprave

Za potrebe analize odstupanja, napravljen je 3D model ispitnog uzorka koji se ovisno o potrebi može rotirati i translatirati u odnosu na ishodište koordinatnog sustava predviđenog za obradu. Nakon što je sklop stezne naprave i ispitnog uzorka pretvoren u *Multi Body* model, napravljen je novi *Body* ploče te je naredbom *Translate* definirana translacija ploče, a naredbom *Rotate* definirana rotacija ploče oko X, Y i Z osi koordinatnog sustava (slika 14).



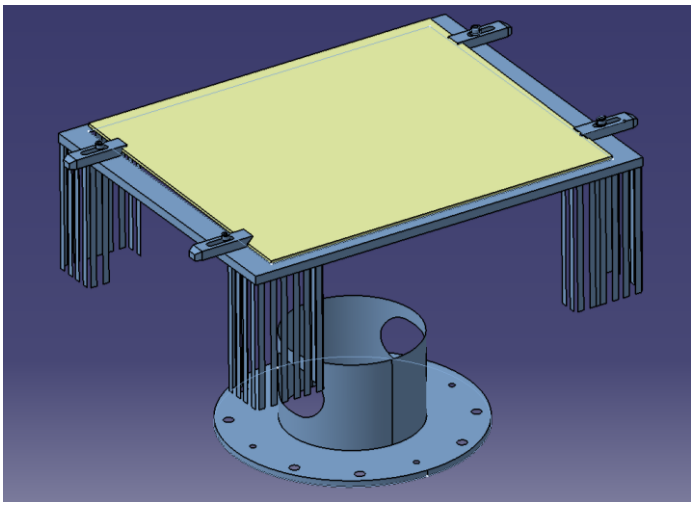
Slika 14. Naredbe za translaciju i rotaciju ispitnog uzorka

Slika 15 a) prikazuje ploču zakrenutu za 1,0° oko Y osi, što na krajnjem rubu daje grešku položaja po visini za 8,73 mm, a slika 15 b) prikazuje ploču zakrenutu za 1,0° oko X osi, što na krajnjem rubu daje grešku položaja po visini za 10,47 mm.



Slika 15. Ispitni uzorak zakrenut oko a) Y osi i b) X osi koordinatnog sustava predviđenog za obradu

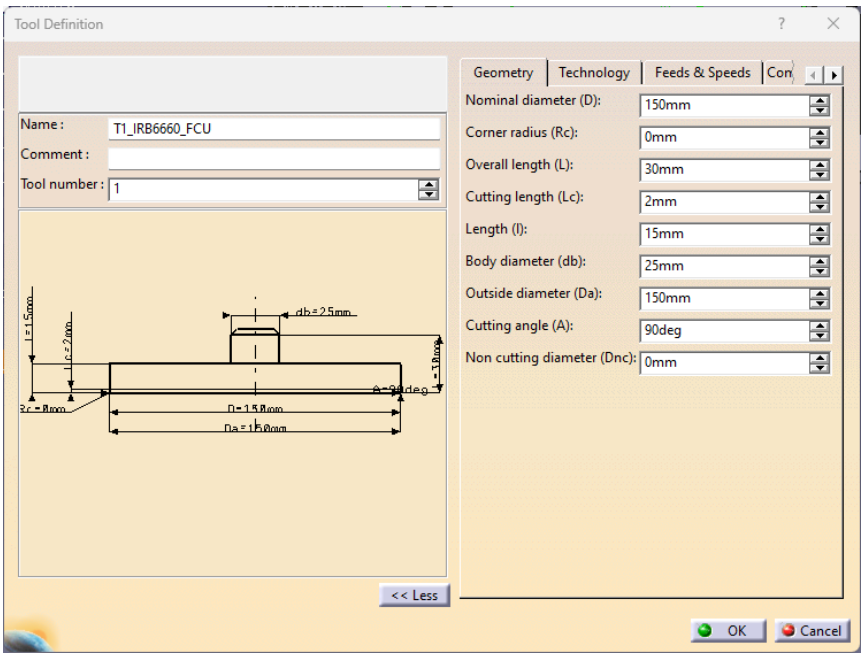
S obzirom da će se 3D modeli zakrenutih ispitnih uzoraka koristiti kao simulirani rezultati digitalizacije, naredbom *Extract* su na sklopu stezne naprave i zamaknutom ispitnom uzorku izdvojene površine koje su vidljive prilikom postupka 3D skeniranja (slika 16). 3D modeli ispitnih uzoraka izrađeni su na opisani način, jer se prilikom postupka digitalizacije prikupljaju samo podaci o geometriji površine. Time je osigurano da površine koje ne bi bile vidljive u rezultatima digitalizacije ne utječu na analizu odstupanja. Iz istog je razloga dio površine gornje ploče koji se nalazi ispod ispitnog uzorka uklonjen naredbom *Split*. Ispitni uzorci koji predstavljaju simulaciju rezultata digitalizacije su pohranjeni u obliku STL datoteka te su kasnije korišteni za analizu odstupanja.



Slika 16. Simulacija rezultata digitalizacije sklopa stezne naprave i zamaknutog uzorka

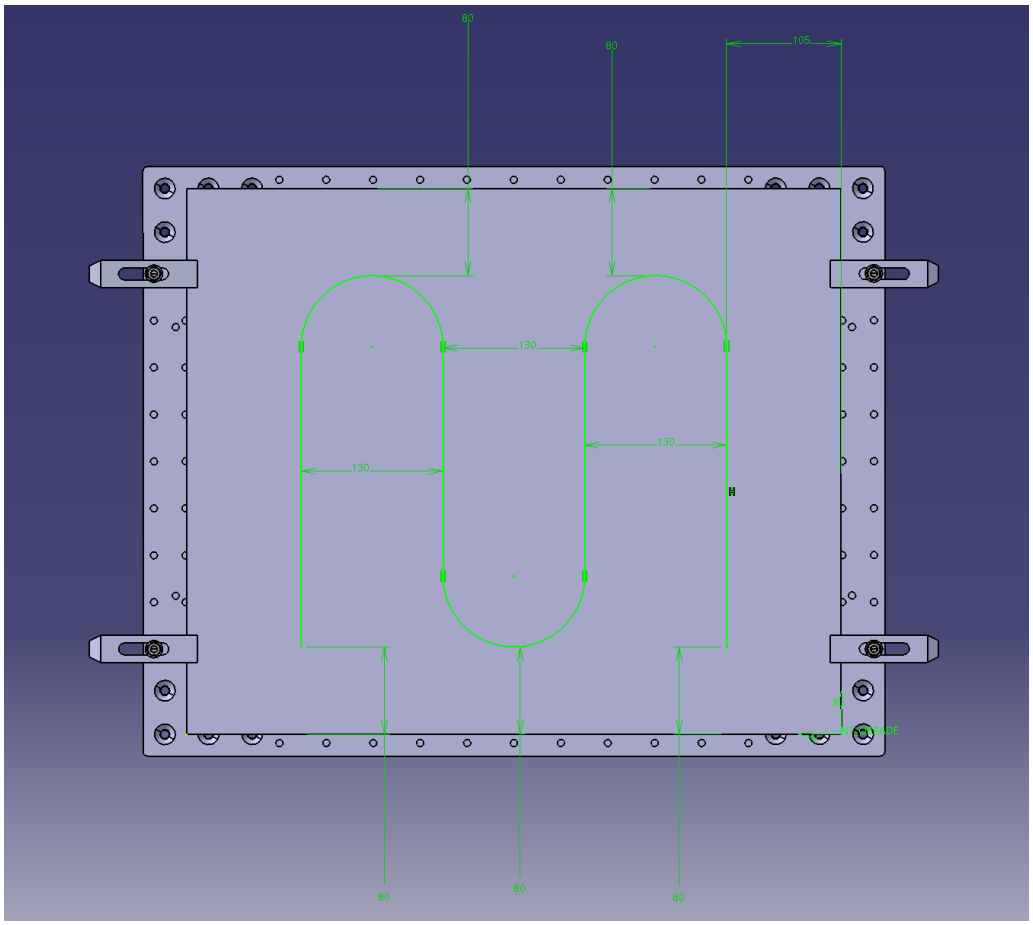
4.2. Putanja alata

Za definiranje putanje alata korišten je CAD/CAM sustav CATIA V5R21. Putanja alata za brušenje referentnog ispitnog uzorka generirana je primjenom naredbe *Curve Following* u sučelju *Prismatic Machining.* S obzirom da su pri izradi programskog koda za korekciju putanje alata potrebni samo podaci o koordinatama točaka na površini obratka, prilikom definiranja geometrije, promjer alata je postavljen na 150 mm, a ostali parametri ostavljeni su nepromijenjeni (slika 17).



Slika 17. Geometrija alata

Na referentnom ispitnom uzorku definirana je putanja po kojoj će se alat kretati tijekom operacije brušenja. S obzirom da promjer alata iznosi 150 mm, putanja je odmaknuta od rubova ispitnog uzorka za 105 mm kako bi se izbjegla kolizija između alata i steznih čeljusti tijekom postupka brušenja. Slika 18 prikazuje definiranu putanju alata na referentnom ispitnom uzorku. Nul-točka obrade definirana je u donjem desnom uglu ispitnog uzorka.



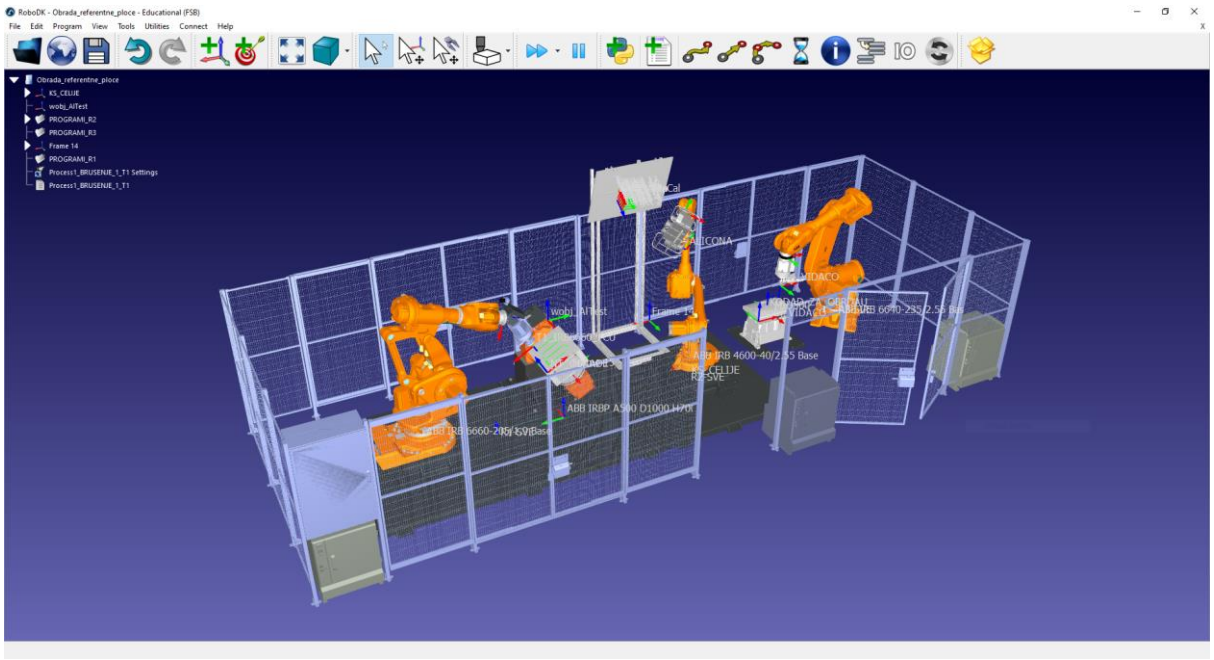
Slika 18. Definirana putanja alata na referentnom ispitnom uzorku

Korišteni softver CATIA V5R21 nema post-procesor potreban za generiranje RAPID koda koji se koristi za definiranje gibanja industrijskog robota ABB IRB 6660-205/1.9. Zbog toga, dobiveni APT kod je učitan u programski paket RoboDK u kojem je generiran RAPID kod.

4.3. Generiranje RAPID koda

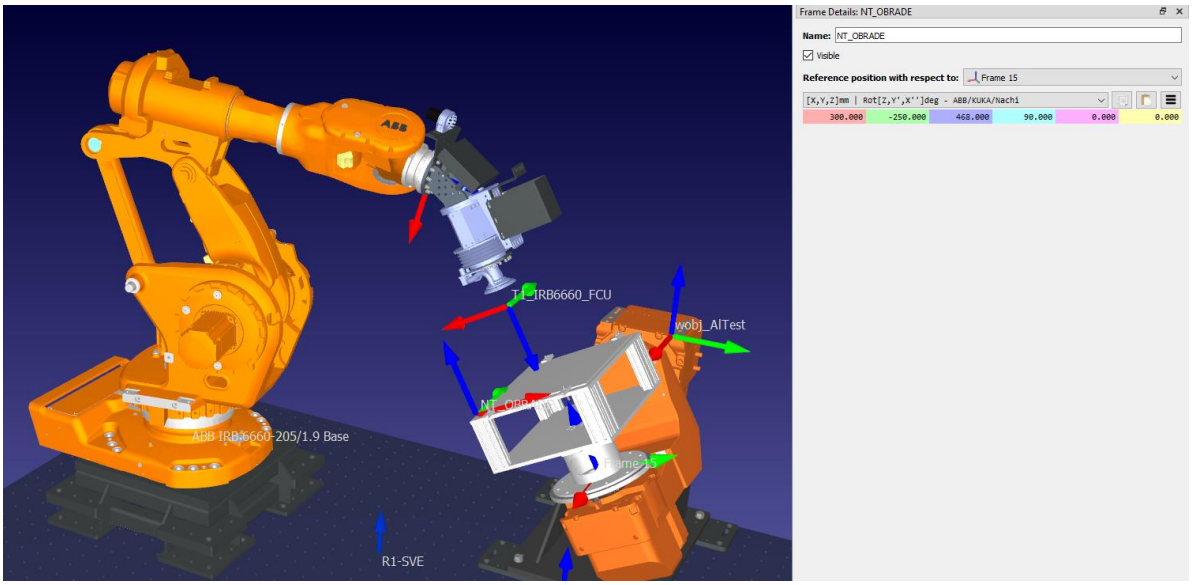
RAPID kod za definiranje gibanja robota tijekom procesa brušenja ispitnog uzorka generiran je u programskom paketu RoboDK. RoboDK je softver za *off-line* programiranje i simulaciju rada robota. Nudi mogućnost programiranja raznih modela robota različitih proizvođača te generiranje programskih kodova potrebnih za definiranje gibanja robota.

U sklopu provedbe projekta ARCOPS kreiran je 3D model robotske ćelije u kojem je moguće simulirati brušenje ispitnih uzoraka (slika 19).



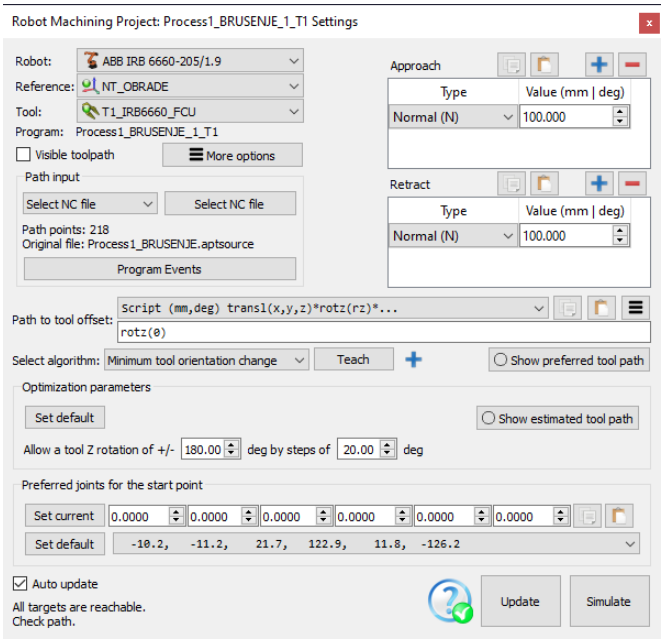
Slika 19. 3D model robotske ćelije ARCOPS u programskom paketu RoboDK

Kako bi se simuliralo brušenje, u softveru RoboDK je u postojeću robotsku ćeliju učitan 3D model ispitnog uzorka. Nakon što je ispitni uzorak postavljen u središte koordinatnog sustava okretno-nagibnog stola, potrebno je definirati nul-točku obrade koja odgovara nul-točki definiranoj u softveru CATIA V5R21. Za određivanje položaja nul-točke, očitane su koordinate nul-točke obrade u odnosu na referentni koordinatni sustav koji se nalazi u središtu baze stezne naprave. Naredbom *Add a Reference Frame* napravljen je novi koordinatni sustav koji predstavlja nul-točku obrade. Pozicija koordinatnog sustava nul-točke definirana je u odnosu na koordinatni sustav okretno-nagibnog stola te je zamaknuta za 300 mm u smjeru X osi, -250 mm u smjeru Y osi te 468 mm u smjeru Z osi . Također, da bi se orijentacija nul-točke obrade poklapala s onom definiranom u softveru CATIA V5R21, koordinatni sustav je rotiran za 90° oko osi Z (slika 20).



Slika 20. Definiranje nul-točke obrade

Nakon što je definirana nul-točka obrade, u softver RoboDK je potrebno učitati APT kod. Prije pokretanja simulacije, u izborniku *Robot Machining Project* (slika 21) definirani su nul-točka obrade te robot i alat kojim će se vršiti obrada. U slučaju kada se okretno-nagibni stol nalazi u početnoj poziciji, tijekom pokretanja simulacije program javlja da robot ne može pozicionirati vrh alata u sve točke na ispitnom uzorku. Zbog navedenog, okretno-nagibni stol je zakrenut za 30° oko osi X označene crvenom bojom.

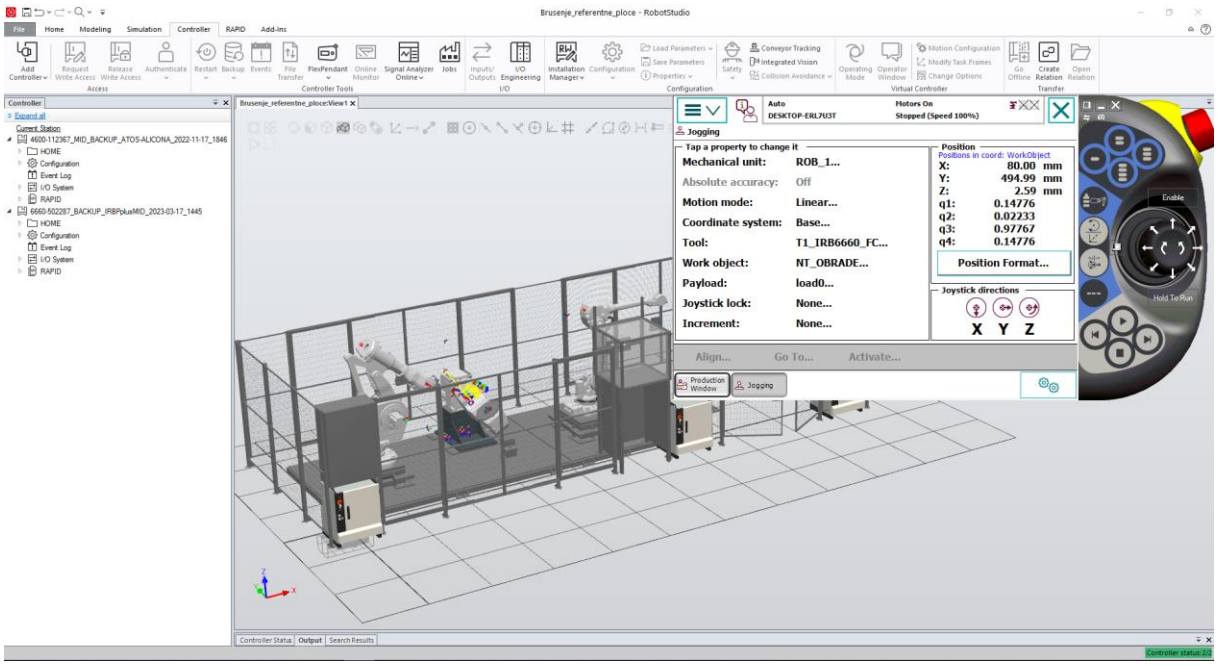


Slika 21. Izbornik *Robot Machining Project*

Završetkom simulacije te njenom provjerom, naredbom *Generate Robot Program* generiran je RAPID kod za definiranje gibanja robota. S obzirom da postoji mogućnost da post-procesor ugrađen u RoboDK neće izvršiti simulaciju gibanja na identičan način kao ABB upravljački sustav koji robot koristi, učitavanjem generiranog koda na robot može doći do singularnosti ili kolizije prouzrokovane razlikom u gibanju zglobova robota ili okretno-nagibnog prigona. Kako bi se osiguralo da gibanje robota u potpunosti odgovara njegovom gibanju u simulaciji, RAPID kod je potrebno učitati i simulirati na modelu ćelije u programskom paketu ABB RobotStudio. U odnosu na RoboDK, ABB RobotStudio može vjerno simulirati rad ABB robota, a učitavanjem modela ćelije u softver može se formirati digitalna kopija ćelije.

4.4. ABB RobotStudio

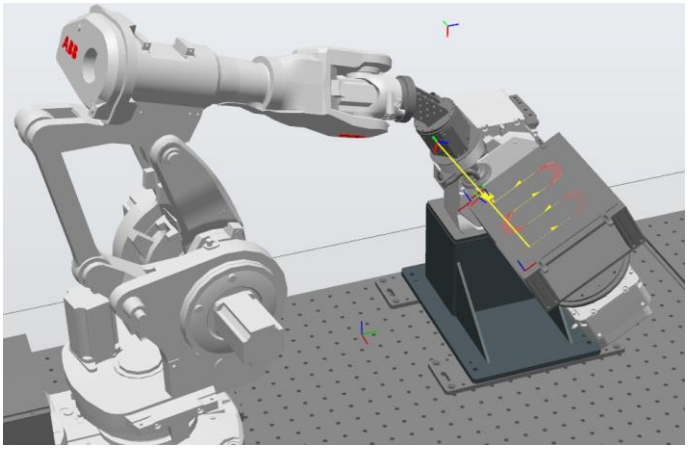
ABB RobotStudio je programski paket koji služi za *off-line* programiranje i simulaciju ABB industrijskih robota. Softver ima ugrađenu podršku za virtualizaciju upravljačkog sustava kojim je moguće izraditi identičnu kopiju stvarnog upravljačkog sustava. Zbog toga se može sa sigurnošću pretpostaviti da će kretnje robota u simulaciji odgovarati stvarnim kretnjama robota. Mogućnost testiranja robotske ćelije u virtualnom okruženju omogućava provjeru grešaka, čime se smanjuju troškovi i vrijeme potrebno za implementaciju robotskih sustava u proizvodnom pogonu [17]. Slika 22 prikazuje 3D model ARCOPS robotske ćelije u korisničkom sučelju softvera ABB RobotStudio.



Slika 22. 3D model robotske ćelije ARCOPS u programskom paketu ABB RobotStudio

Prije učitavanja RAPID koda u postojeću ARCOPS robotsku ćeliju ubačen je 3D model sklopa stezne naprave i ispitnog uzorka u obliku STL datoteke. 3D model sklopa učitan je naredbom *Import Geometry*, a na okretno-nagibni stol je postavljen naredbom *Attach To* i odabirom okretnog-nagibnog stola ABB IRBP A. Na isti način učitan je brusni alat te je priključen na prirubnicu robota ABB IRB 6660/205-1.9. S obzirom da je u softveru RoboDK nagib okretno-nagibnog stola promijenjen za 30°, u softveru ABB RobotStudio je također njegov nagib promijenjen za isti iznos.

Nakon što je robotska ćelija prilagođena za obradu ispitnog uzorka, u ABB RobotStudio je učitan RAPID kod generiran u softveru RoboDK. U programskom jeziku RAPID razlikuju se sistemski i programski moduli. Sistemski moduli koriste se za definiranje tipova podataka i rutina koji su važni za funkcioniranje ćelije, a nisu u užem smislu povezani s tehnološkim procesom. Programski moduli sadržavaju podatke kojima se definiraju procesi i rutine koji su u užem smislu povezani s tehnološkim procesom (npr. operacije brušenja). U izborniku *Controller* označen je robot ABB IRB 6660/205-1.9 te je naredbom *Load Module* učitan programski modul za brušenje ispitnog uzorka, te sistemski modul kojim su definirani parametri alata. Slika 23 prikazuje robotsku ćeliju nakon što je učitan programski modul za brušenje ispitnog uzorka. Na ispitnom uzorku prikazana je putanja alata, a u svakoj točki putanje je prikazan koordinatni sustav vrha alata robota TCP (eng. *tool centre point*). Pokretanjem simulacije provjereno je da robot ispravno prati definiranu putanju alata te je zaključeno kako nisu potrebne modifikacije programa.



Slika 23. Simulacija brušenja ispitnog uzorka

Slika 24 prikazuje dio programskog modula u kojem je definirana glavna rutina (eng. *Main*) koja se prva pokreće kada robot radi u automatskom načinu rada. Glavna rutina u sebi može sadržavati putanju gibanja alata, a kod složenijih aplikacija (npr. obrada više različitih pozicija)

može sadržavati pozive na druge rutine koje onda u zavisnosti o uvjetima sadrže instrukcije za gibanje alata u specifičnim situacijama. S obzirom da će se generirani RAPID kod primjenjivati kao referentni u programskom kodu za korekciju putanje alata, potrebno je izdvojiti točke u kojima se alat nalazi u zahvatu s obratkom. U slučaju kada je potrebno u RAPID kodu dodati komentar, dodaje se znak "!" . Na taj način je prije linije koda u kojoj alat ulazi u zahvat postavljen komentar "! Pocetak", a nakon zadnje linije koda u kojoj je još alat u zahvatu postavljen komentar "! Kraj".

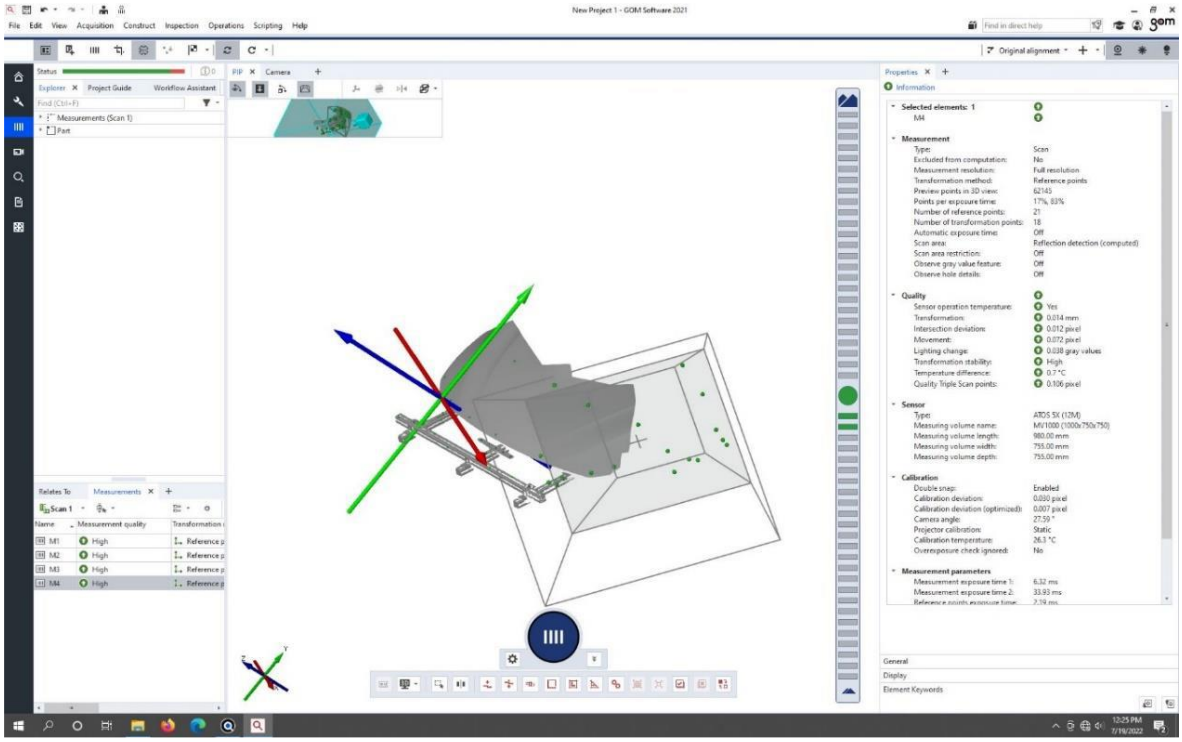
U glavnom programu, instrukcijom *MoveL* definirano je linearno kretanje vrha alata robota po putanji od točke do točke. U prvoj uglatoj zagradi definirane su koordinate točaka u odnosu na ishodište koordinatnog sustava predviđenog za obradu. U drugoj uglatoj zagradi je definirana orijentacija vrha alata robota u obliku kvaterniona. Kvaternioni su način opisivanja orijentacije u trodimenzionalnom prostoru koji se u području robotike često primjenjuju za definiranje orijentacije vrha alata robota, a sastoje se od jedne realne komponente i tri imaginarne komponente [18]. U trećoj uglatoj zagradi nalaze se dodatne informacije o konfiguraciji robota kako bi se uz koordinate i orijentaciju vrha alata robota jednoznačno definirao položaj alata u odnosu na obradak, a time i položaj robota i vanjskih prigona. Konfiguracija je definirana brojevima koji označavaju u kojem se kvadrantu karakteristične osi robota nalaze. U četvrtoj zagradi definiran je položaj vanjskih osi, u ovom slučaju okretno-nagibnog stola, dok preostali podaci iza uglatih zagrada definiraju brzine gibanja, zahtijevanu točnost praćenja putanje, te zadano ishodište koordinatnog sustava obrade [19]. Za potrebe korekcije putanje alata korišteni su podaci u prvoj i drugoj uglatoj zagradi nakon instrukcije *MoveL*.

```
PROC Main()
    ConfJ \On;
    ConfL \Off;
    ! Program generated by RoboDK v5.6.7 for ABB IRB 6660-205/1.9 on 07/11/2023 10:04:58
    ! Using nominal kinematics.
    ! --------------------------------------------------------------
    ! Generated on utorak, 7. studeni 2023. 10:00:45
    ! CATIA APT VERSION 1.0
    ! --------------------------------------------------------------
    ! BRUSENJE
    ! BRUSENJE
    ! *CATIA0
    ! BRUSENJE
    ! 1.00000     0.00000     0.00000  -250.00000
    ! 0.00000     1.00000     0.00000  -300.00000
    ! 0.00000     0.00000     1.00000   468.00000
    ! OPERATION NAME : Tool Change.1
    ! Start generation of : Tool Change.1
    ! TOOLCHANGEBEGINNING
    ! TOOLCHANGEEND
    ! End of generation of : Tool Change.1
    ! OPERATION NAME : Curve Following.1
    ! Start generation of : Curve Following.1
    Mirka 70.0;
    ! Show T1_IRB6660_FCU
    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;
    ! Pocetak
    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 [[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 [[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 [[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 [[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 [[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 [[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 [[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;
    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;
    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;
    ! Kraj
    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;
    ! End of generation of : Curve Following.1
ENDPROC
```

Slika 24. Dio glavne rutine RAPID koda

5. KOREKCIJA PUTANJE ALATA

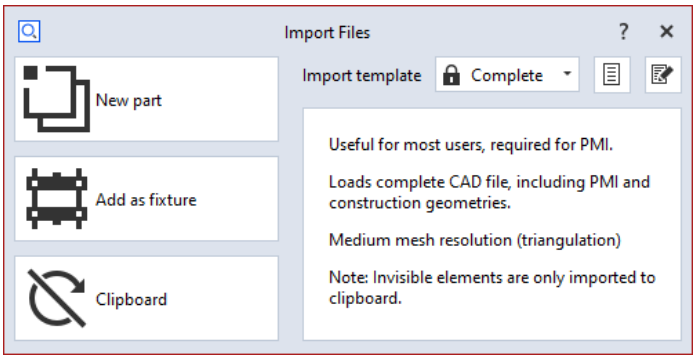
Za analizu odstupanja položaja i orijentacije kod stezanja ispitnih uzoraka primijenjen je programski paket GOM Inspect. GOM Inspect je softver njemačke tvrtke GOM *metrology GmbH* koji se primjenjuje se za prikupljanje i obradu podataka tijekom procesa digitalizacije, analizu rezultata digitalizacije, usporedbu digitalnog prikaza realnih objekata s referentnim CAD modelima, mjerenje te praćenje i analizu objekata u stvarnom vremenu (eng. *live tracking*). Također, nudi mogućnost analize geometrije, dimenzija i tolerancija razmatranog ispitnog uzorka te generiranje izvješća o svakoj provedenoj analizi. Slika 25 prikazuje korisničko sučelje programskog paketa GOM Inspect.



Slika 25. Korisničko sučelje programskog paketa GOM Inspect

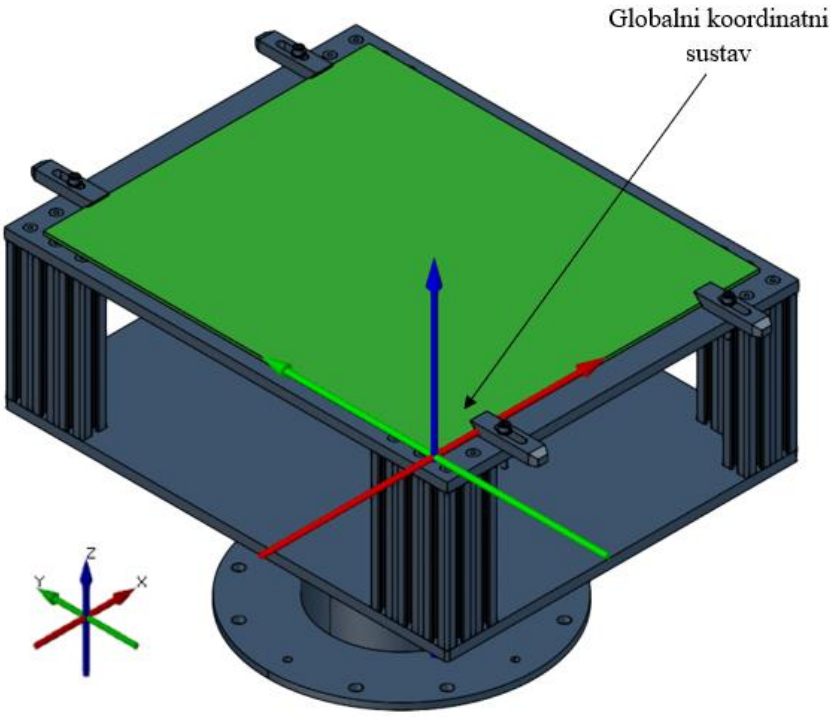
5.1. Priprema podataka za analizu

Prvi korak pri formiranju postava u kojem će se analizirati odstupanja položaja i orijentacije kod stezanja ispitnih uzoraka je definiranje stezne naprave. U GOM Inspect je naredbom *Import* učitana STEP datoteka stezne naprave. Nakon što je 3D model stezne naprave učitan, potrebno je u izborniku *Import Files* odabrati opciju *Add as fixture* (slika 26). Na taj način je stezna naprava definirana kao dio mjerne okoline (eng. *Measuring environment*) te se površine njenog 3D modela prilikom analize deformacija neće uzimati u obzir.



Slika 26. Izbornik naredbe *Import Files*

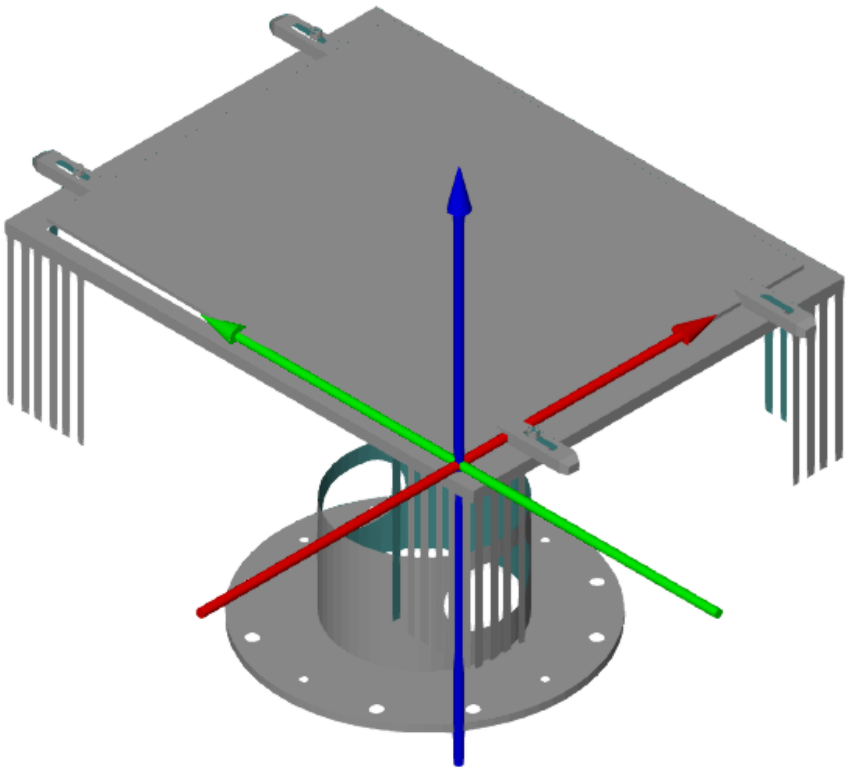
Nakon što je definirana stezna naprava, u softveru je učitan 3D model pravilno stegnutog ispitnog uzorka. Prilikom učitavanja CAD modela u obliku STEP datoteke, potrebno je u izborniku *Import Files* odabrati opciju *New part*. Na taj je način pravilno stegnuti ispitni uzorak označen kao referentni element (eng. *Nominal Element*). Zbog jednostavnijeg razlikovanja elemenata, boja 3D modela referentnog ispitnog uzorka je promijenjena u zelenu.



Slika 27. Referentni CAD model s prikazanim globalnim koordinatnim sustavom

S obzirom da je referentni ispitni uzorak označen kao *Nominal Element*, softver prepoznaje koordinatni sustav definiran u CATIA-i V5R21 kao globalni koordinatni sustav te sve koordinate kontrolnih elemenata izračunava u odnosu na njega. U ovom slučaju, globalni koordinatni sustav se nalazi na gornjem uglu ispitnog uzorka te se podudara s nul-točkom obrade (slika 27). Kasnije je u softveru moguće konstruirati lokalni koordinatni sustav prema kojem se mogu definirati nove koordinate kontrolnih elemenata.

CAD modeli pogrešno stegnutih ispitnih uzoraka su spremljeni u obliku STL datoteke, a prilikom učitavanja u softver, u izborniku *Import STL* potrebno je odabrati značajku *Mesh*. Na taj način ispitni uzorak postaje stvarni element (eng. *Actual Element*) te je označen sivom bojom (slika 28).



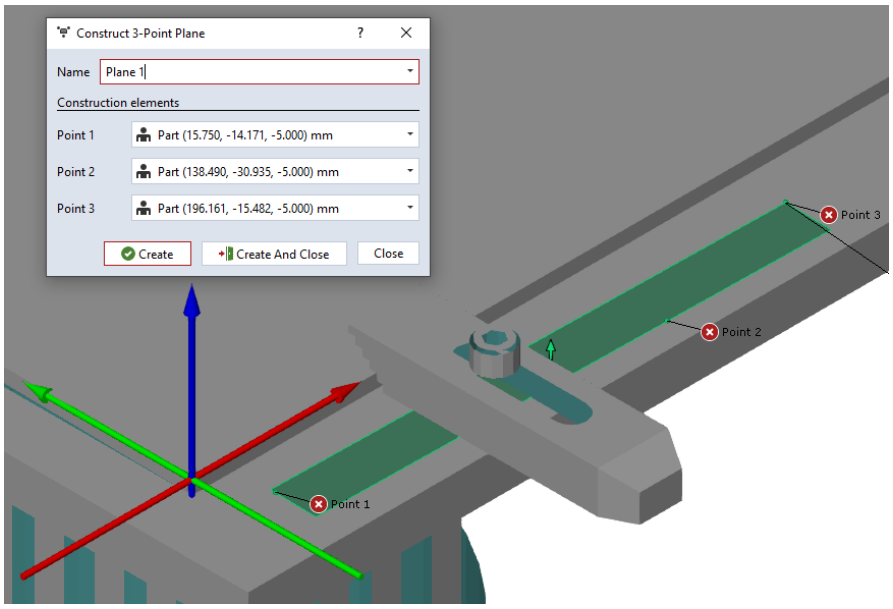
Slika 28. Simulacija skena pogrešno stegnutog ispitnog uzorka

U korisničkom sučelju programskog paketa GOM Inspect se s lijeve strane nalazi stablo u kojem su definirani svi elementi projekta te sve naredbe izvršene na njima. Kontrolni elementi i naredbe koje se odnose na *Nominal Element* su u stablu označeni plavom bojom, a kontrolni elementi koji se odnose na *Actual Element* su označeni zelenom bojom. Navedeni elementi su na isti način označeni i u izbornicima naredba.

5.2. Poravnanje

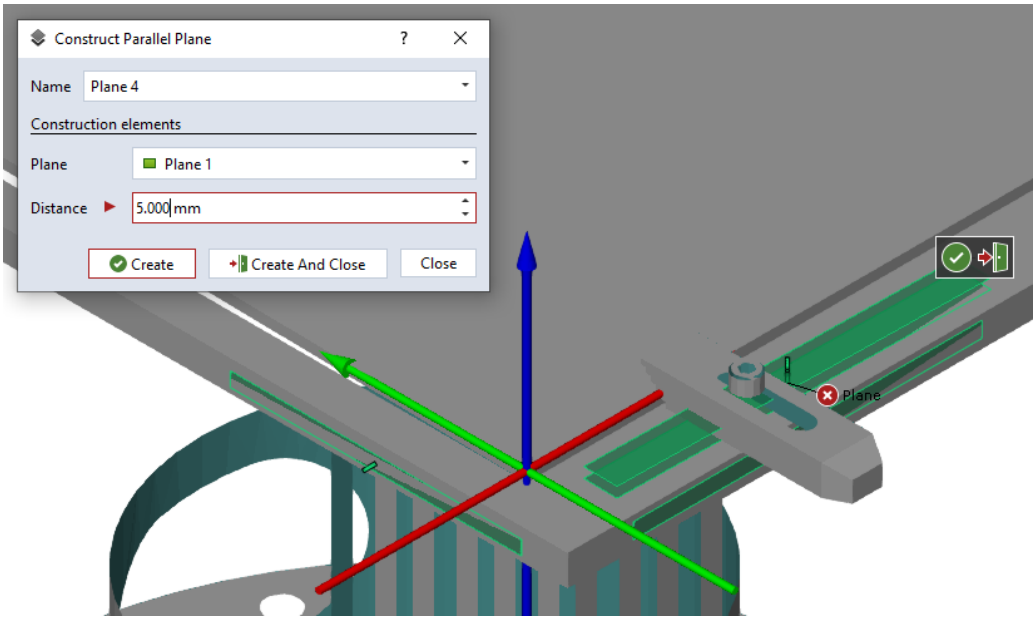
U svrhu analize odstupanja zakrenutih ispitnih uzoraka u odnosu na referentni CAD model, učitane elemente je potrebno poravnati. S obzirom da će se u ovom diplomskom radu mjeriti odstupanje prilikom pogrešnog stezanja ispitnih uzoraka, elementi su poravnati na način da se preklapaju stezne naprave. Da bi se to postiglo, na elementima je izvršeno glavno poravnanje (eng. *Main Alignment*).

Kako bi se osiguralo da se stezne naprave u potpunosti podudaraju te da postoje samo odstupanja ispitnih uzoraka, za glavno poravnanje korištena je naredba *Main Alignment By Coordinate Systems*. Za navedenu naredbu potrebno je konstruirati lokalni koordinatni sustav koji se u potpunosti preklapa s globalnim koordinatnim sustavom. Za potrebe definiranja ishodišta te smjera osi lokalnog koordinatnog sustava, na simuliranom skenu sklopa stezne naprave i pogrešno stegnutog ispitnog uzorka konstruirane su tri ravnine. Ravnine su na steznoj napravi konstruirane naredbom *Construct 3-Point Plane* (slika 29). Kako bi se ovom naredbom konstruirala ravnina, na površini je potrebno definirati 3 točke. Ravnine su konstruirane na način da je ravnina 1 (XY') paralelna s XY ravninom, ravnina 2 (XZ') paralelna s XZ ravninom te ravnina 3 (YZ') paralelna s YZ ravninom globalnog koordinatnog sustava. S obzirom da je u primjeru na slici 29 riječ o ravnoj plohi na idealnom CAD modelu, točke za definiranje ravnine nalaze se blizu jedna drugoj. U slučaju da se umjesto CAD modela koristi skenirani uzorak, kako bi se postigla što veća točnost, poželjno je prilikom konstruiranja ravnine točke definirati na što većoj udaljenosti.



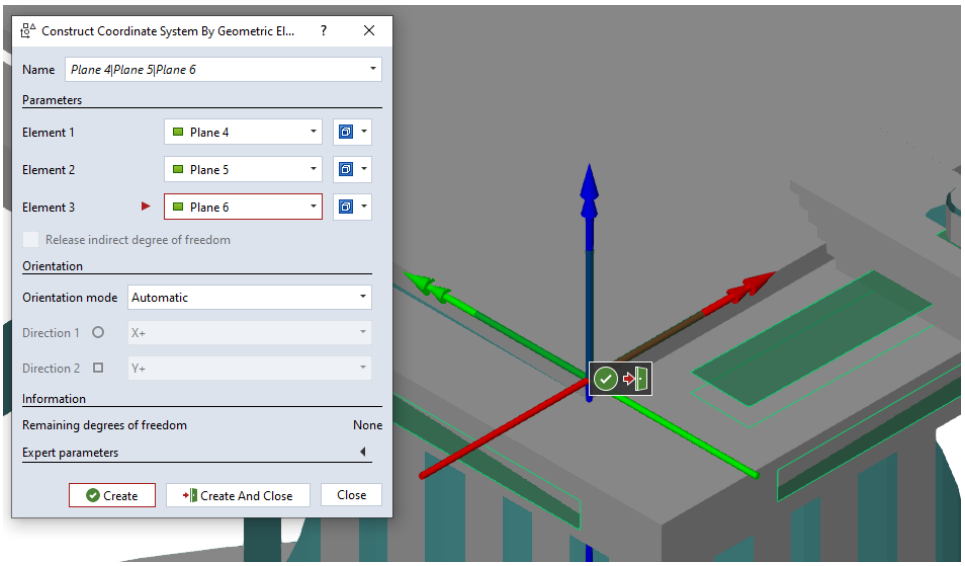
Slika 29. Konstruiranje ravnine naredbom *Construct 3-Point Plane*

Konstruirane ravnine su zatim translatirane kako bi se u potpunosti preklapale s ravninama globalnog koordinatnog sustava. Naredbom *Construct Parallel Plane* (slika 30) konstruirane su ravnine paralelne s ravninama XY', XZ' i YZ'. Ravnina 4 (XY") je paralelna s ravninom XY' te je zamaknuta od nje za 5 mm u pozitivnom smjeru osi Z, ravnina 5 (XZ") je paralelna s ravninom XZ' te je zamaknuta od nje za 40 mm u pozitivnom smjeru osi Y, a ravnina 6 (YZ") je paralelna s ravninom YZ' te je zamaknuta od nje za 20 mm u pozitivnom smjeru osi X.



Slika 30. Konstruiranje ravnine naredbom *Construct Parallel Plane*

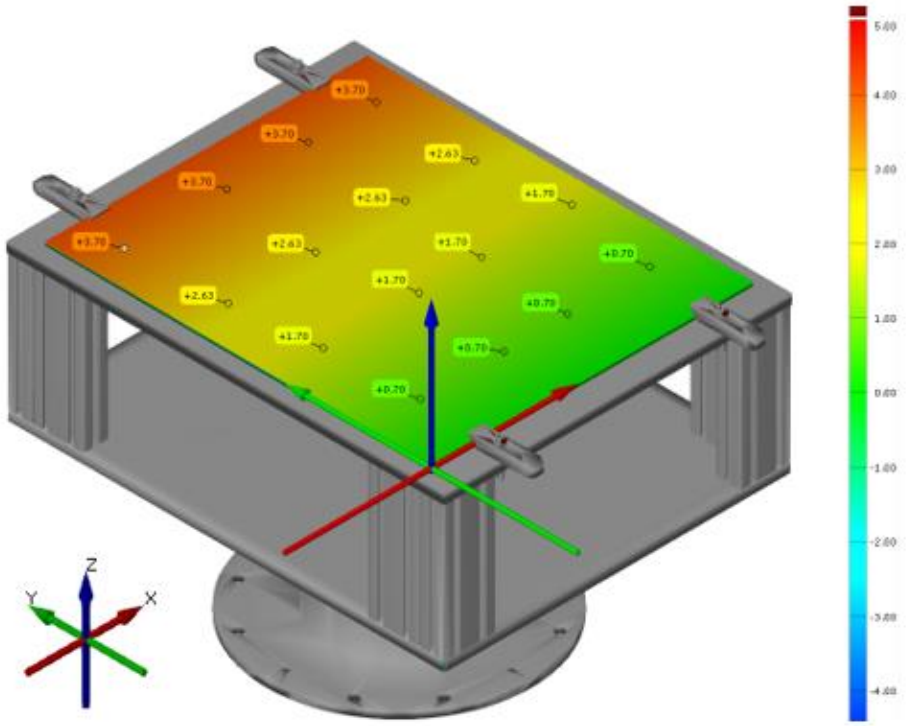
Nakon što su konstruirane ravnine, sljedeći korak je konstrukcija lokalnog koordinatnog sustava. U izborniku naredbe *Construct Coordinate System By Geometric Elements* odabiru se ravnine u odnosu na koje će osi novog koordinatnog sustava biti okomite (slika 31). Označene su ravnine XY", XZ" i YZ", a orijentaciju osi lokalnog koordinatnog sustava nije potrebno podešavati jer se već poklapa s orijentacijom osi globalnog koordinatnog sustava.



Slika 31. Konstruiranje lokalnog koordinatnog sustava naredbom *Construct Coordinate System By Geometric Elements*

Završetkom konstrukcije lokalnog koordinatnog sustava, može se izvršiti glavno poravnanje referentnog CAD modela i zamaknutog ispitnog uzorka. U izborniku *Main Alignment By Coordinate Systems* definirano je da se novonastali lokalni koordinatni sustav poravna s globalnim koordinatnim sustavom. Softver poravnava označene koordinatne sustave na način da se njihove pozicije i orijentacije u potpunosti preklapaju. Nakon što je izvršeno glavno poravnanje, na ispitnom postavu se može provesti postupak kreiranja kontrolnih elemenata i analiza odstupanja ispitnog uzorka u odnosu na referentni CAD model.

Ispravnost postupka poravnanja provjerena je naredbom *Surface Comparison on Actual* (slika 32). Također, naredbom *Deviation Label* postavljene su kontrolne točke na površinu zamaknutog ispitnog uzorka. Analizom je utvrđeno da su odstupanja kontrolnih točaka na istoj udaljenosti od osi X jednaka te se povećavaju u smjeru osi Y. S obzirom da je u navedenom primjeru analiziran ispitni uzorak zakrenut oko osi X, zaključeno je da je glavno poravnanje ispravno.



Slika 32. Analiza odstupanja naredbom *Surface Comparison on Actual*

5.2.1. Automatizacija postupka poravnanja

U GOM Inspect-u moguće je generirati skripte koje nastaju snimanjem različitih korisničkih operacija u softveru (eng. *macro*). Snimanjem se generira kod u programskom jeziku Python, kojim je uz kasnije izmjene i dorade moguće automatizirati proces digitalizacije i obrade mjernih podataka.

Da bi se izbjeglo konstruiranje kontrolnih elemenata te definiranje poravnanja prilikom izrade svakog novog projekta, pomoću funkcije *Scripting* automatiziran je prethodno opisan postupak poravnanja. Nakon što je u izborniku naredbe *Edit Script* pokrenuto snimanje, snimljene su sve naredbe potrebne za pravilno poravnanje uzoraka te je snimanje završeno. Snimljena skripta za poravnanje može se primijeniti prilikom izrade svakog novog projekta koji uključuje primijenjenu steznu napravu i ispitni uzorak. Slika 33 prikazuje programski kod za poravnanje ispitnih uzoraka s referentnim CAD modelom. Prvim dijelom programskog koda opisane su naredbe potrebne za konstruiranje ravnine na površini stezne naprave. U sljedećem koraku opisane su naredbe potrebne za konstruiranje ravnina paralelnih u odnosu na prethodne. Treći dio koda opisuje konstruiranje lokalnog koordinatnog sustava, a u četvrtom dijelu je definirano glavno poravnanje.

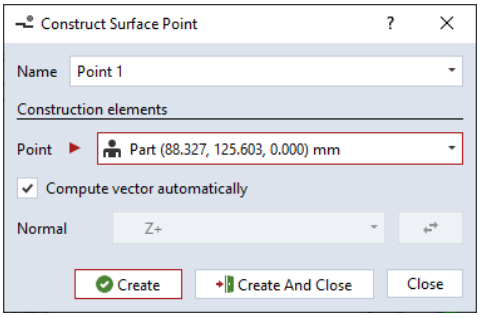
Slika 33. Programski kod za poravnanje stezne naprave neovisno o poziciji i orijentaciji ispitnog uzorka

5.3. Konstruiranje kontrolnih točaka

Ovisno o potrebi, u programskom paketu GOM Inspect moguće je provoditi različite vrste analize objekata. Softver nudi mogućnosti mjerenja koordinata točaka u odnosu na definirani koordinatni sustav, udaljenosti te iznose kutova između pojedinih kontrolnih elemenata.

Za potrebe ovog diplomskog rada potrebno je konstruirati točke na površini referentnog uzorka koje odgovaraju koordinatama točaka putanje alata u generiranom NC programu. Uz svaku od kontrolnih točaka vezana je informacija o koordinati točke te smjeru vektora normale na površinu u toj točki. Dobivene točke potrebno je projicirati na površinu zamaknutog ispitnog uzorka, a podaci o koordinatama točaka i orijentaciji vektora normala na površinu zamaknutog ispitnog uzorka koristit će se prilikom korekcije putanje alata.

Kontrolne točke na površini referentnog CAD modela konstruirane su naredbom *Construct Surface Point*. U izborniku *Construct Surface Point* (slika 34) je za svaku točku moguće unijeti koordinate u odnosu na definirani koordinatni sustav te odrediti smjer i orijentaciju vektora normale.



Slika 34. Izbornik naredbe *Construct Surface Point*

S obzirom da je na ispitnom uzorku potrebno kreirati niz kontrolnih točaka čije se koordinate poklapaju s koordinatama točaka putanje alata prema NC programu, izrađen je programski kod kojim se iz generiranog RAPID koda izdvajaju koordinate točaka putanje alata te se u GOM Inspect-u konstruiraju na površini ispitnog uzorka referentnog CAD modela.

Programski kod započinje učitavanjem modula *gom* u kojem su sadržane sve naredbe i parametri specifični za GOM Inspect Pro te biblioteka *libLeginRepalust* u kojoj su definirane funkcije koje će se pozivati prilikom izrade programa za korekciju putanje alata (slika 35). Također, definirana je lokacija na kojoj je pohranjena datoteka referentnog RAPID koda te je komentarima "! Pocetak" i "! Kraj" označen dio koda u kojem se alat nalazi u zahvatu s obratkom. Linijom koda *lstTockePutanjeZahvat*=[] stvorena je prazna lista u koju će se pohranjivati podaci iščitani iz RAPID koda.

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

lstTockePutanjeZahvat=[]
```

Slika 35. Definiranje modula i ulaznih podataka

Na slici 36 je prikazan dio programskog koda kojim se učitavaju potrebne linije RAPID koda. Programski kod čita sve linije RAPID koda te započinje kopiranje linija nakon komentara "! Pocetak", a završava prije komentara "! Kraj". Metodom *.append()* se svaka linija koda koja zadovoljava uvjet da se nalazi između traženih komentara dodaje u listu označenu varijablom *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 36. Učitavanje linija RAPID koda

Na slici 37 je prikazan dio programskog koda kojim se učitavaju podaci pohranjeni u prethodno stvorenoj listi. S obzirom da je na površini ispitnog uzorka potrebno konstruirati kontrolne točke koje odgovaraju koordinatama točaka putanje alata te konstruirati pripadajuće vektore normala, iz linija RAPID koda potrebno je izdvojiti podatke o koordinatama točaka i kvaternionima.

```
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.

    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
```

1

2

3

Slika 37. Konstruiranje kontrolnih točaka na referentnom ispitnom uzorku

Programski kod na slici 38 prikazuje naredbe pomoću kojih se iz linija RAPID koda izdvajaju podaci potrebni za generiranje kontrolnih točaka i vektora normala na površinu ispitnog uzorka idealnog CAD modela. Navedeni programski kod definiran je u biblioteci *libLeginRepalust*. Programski kod u svakoj od linija RAPID koda prvo izdvaja podatke koji se nalaze između prve i posljednje uglate zagrade u liniji. Izdvojeni dio linije je metodom *.split()* podijeljen u listu simbolima "],[". Na prvom mjestu ([0]) stvorene liste nalaze se podaci o koordinatama točaka, a na drugom mjestu ([1]) se nalaze podaci o kvaternionima. Da bi se svaka komponenta

mogla koristiti zasebno, izdvojeni podaci su ponovno podijeljeni u listu simbolom ",". Izdvojenim podacima o koordinatama točaka putanje alata pridodana je varijabla *_koordinate_tocaka*, a podacima o kvaternionima je pridodana varijabla *_kutevi_tocaka_quart*. Podaci u stvorenim listama definirani su kao podatak tipa *string*, odnosno u obliku niza znakova (npr. brojevi, slova, interpunkcijski znakovi). Za potrebe daljnjih proračuna podaci su pretvoreni u decimalne brojeve (*float*).

```
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]
```

Slika 38. Izdvajanje koordinata točaka i kvaterniona

U dijelu programskog koda na slici 37 označenim brojem 2, koordinata X definirana je varijablom *koord_x* te se odnosi na prvi element unutar prve liste *(_koordinate_tocaka*), koordinata Y definirana je varijablom *koord_y* te se odnosi na drugi element unutar prve liste, a koordinata Z je definirana varijablom *koord_z* te se odnosi na treći element unutar prve liste. Jedinični vektori definirani su varijablama *jed_vec_i*, *jed_vec_j* i *jed_vec_k*, a programski kod za pretvorbu kvaterniona u jedinične vektore pohranjen je u biblioteci *libLeginRepalust*, te je definiran funkcijom *quat2jv* (slika 39). Budući da je Z os koordinatnog sustava vrha alata robota orijentirana prema obratku, a vektori normale na površinu ispitnog uzorka su suprotno orijentirani, orijentacija jediničnih vektora je prilikom definiranja varijabli *jed_vec_i*, *jed_vec_j* i *jed_vec_k* promijenjena množenjem i, j i k komponenti s -1.

```
def quat2jv(q1, q2, q3, q4, chsVec):

    if chsVec == "x":
        i = 1-2*pow(q3,2)-2*pow(q4,2)
        j = 2*q2*q3+2*q1*q4
        k = 2*q2*q4-2*q1*q3
    elif chsVec == "y":
        i = 2*q2*q3-2*q1*q4
        j = 1-2*pow(q2,2)-2*pow(q4,2)
        k = 2*q3*q4+2*q1*q2
    elif chsVec == "z":
        i = 2*q2*q4+2*q1*q3
        j = 2*q3*q4-2*q1*q2
        k = 1-2*pow(q2,2)-2*pow(q3,2)
    else:
        i = float("NaN")
        j = float("NaN")
        k = float("NaN")

    return(i,j,k)
```

Slika 39. Programski kod za pretvorbu kvaterniona u jedinične vektore

S obzirom da su programski kodovi za izvršavanje naredbi specifični za GOM Inspect Pro, postupak konstruiranja kontrolnih točaka snimljen je funkcijom *Scripting*. Da bi se na površini referentnog CAD modela konstruirale točke čije koordinate odgovaraju koordinatama točaka putanje alata, programski kod za konstruiranje točke naredbom *Construct Surface Point* je modificiran. Na slici 40 je u prvom primjeru prikazana snimljena naredba za konstruiranje točaka na površini referentnog CAD modela, a u drugom primjeru je prikazana modifikacija naredbe. Naredba je modificirana na način da je u identifikacijskoj oznaci točke umjesto rednog broja stavljen oznaka %s. Na taj način će se prilikom pokretanja programa svakoj točki uz oznaku *Point* dodijeliti novi redni broj počevši od 1. Uz to, u funkcijama *gom.Vec3d* su umjesto definiranih parametara za smjer vektora normale postavljene varijable *jed_vec_i*, *jed_vec_k* i *jed_vec_j*, a za koordinate točke su postavljene varijable *koord_x*, *koord_y* i *koord_z*.

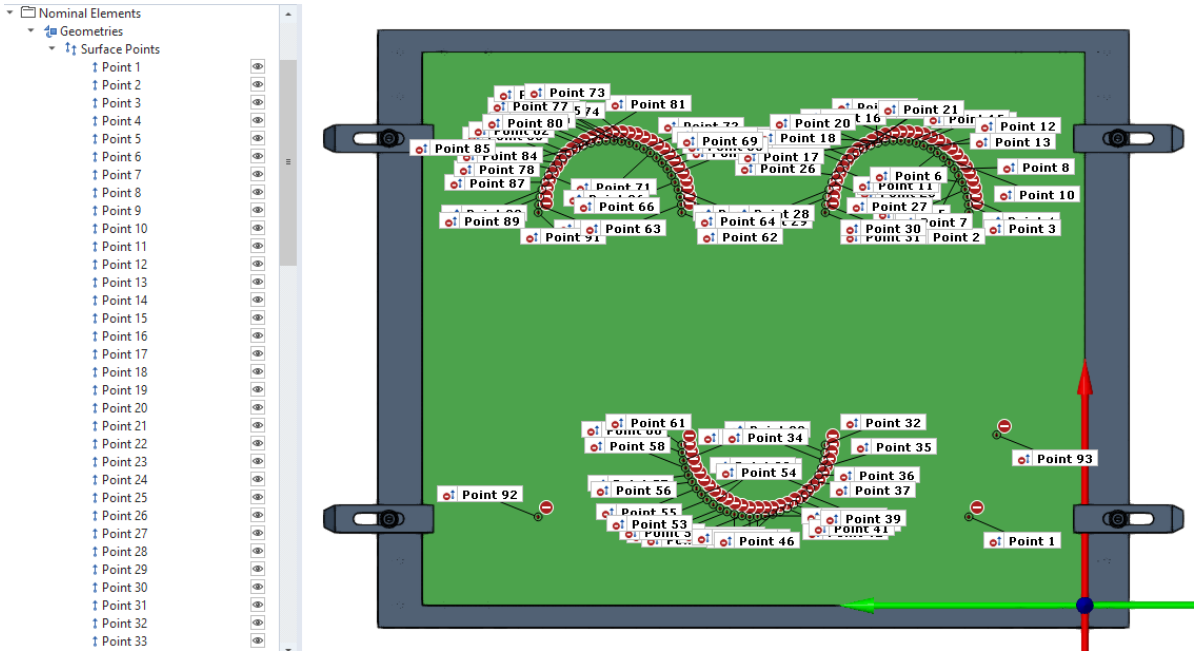
```
MCAD_ELEMENT=gom.script.primitive.create_surface_point (
    name='Point 1',
    point={'interpolated': True, 'normal': gom.Vec3d (0.0, 0.0, 1.0), 'point': gom.Vec3d (154.1151027, 79.80660946, 0.0), 'target': gom
    )

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),
```

1

2

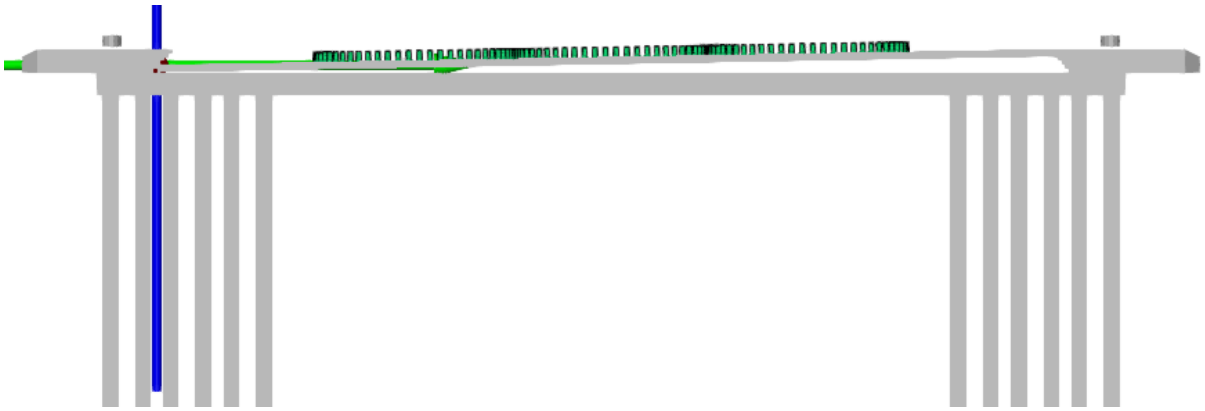
Slika 40. Modifikacija naredbe za konstruiranje kontrolnih točaka i vektora normala

Pokretanjem programskom koda je za svaku liniju RAPID koda u kojoj se alat nalazi u zahvatu konstruirana točka s identifikacijskom oznakom *Point* i rednim brojem, koordinatama točke koje odgovaraju koordinatama točaka putanje alata te s vektorima normala u smjeru pozitivnom s obzirom na površinu obratka, ali suprotne orijentacije u odnosu na Z os vrha alata robota (slika 41).



Slika 41. Kontrolne točke konstruirane na referentnom ispitnom uzorku

Nakon što su na referentnom CAD modelu konstruirane točke s pripadajućim vektorima normala, točke je potrebno projicirati na površinu zakrenutog ispitnog uzorka. U stablu (slika 41, lijeva strana) su klikom na *Surface Points* označene sve točke, a projicirane su naredbom *Measuring Principle: Project Point Onto Actual* (slika 42). Za vektore normala na projicirane točke se u GOM Inspect-u mogu dobiti koordinate ishodišta te orijentacija, a navedene informacije koristit će se kao ulazni podaci u programskom kodu za korekciju putanje alata.



Slika 42. Projicirane točke i vektori normala na površini zamaknutog ispitnog uzorka

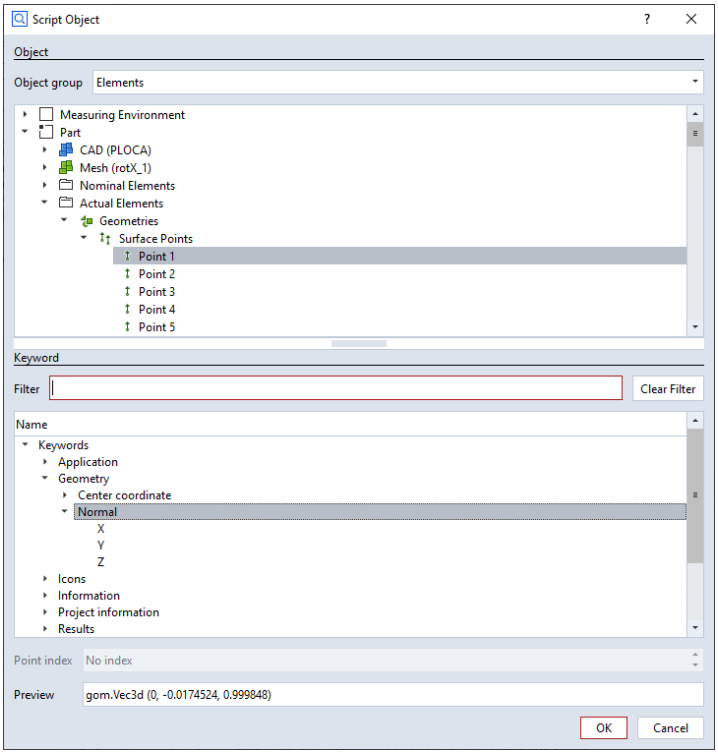
5.4. Programski kod korekcije putanje alata

U programski kod za korekciju putanje alata prvo su učitani programski modul *gom* te biblioteke *libLeginRepalust* i *numpy* (slika 43). *Numpy* je Python biblioteka koja sadrži matematičke funkcije potrebne za rad s vektorima i matricama. Uz to, na početku programskog koda definirane su lokacija referentnog RAPID koda te lokacija na koju ću se pohranjivati ažurirani RAPID kod.

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

Slika 43. Učitavanje programskih modula i biblioteka u programski kod za korekciju putanje alata

Kako bi se dobili podaci o koordinatama točaka i orijentaciji vektora normala na referentnom CAD modelu i ispitnom uzorku, u korisničkom sučelju *Script Editor* desnim klikom miša odabrana je naredba *Insert Element Value*. Navedenom naredbom otvara se izbornik *Script Object* u kojem je moguće odabrati sve postojeće elemente u projektu (slika 44). Do potrebnih elemenata dolazi se odabirom *Surface Points* u padajućem izborniku u odjeljku *Geometry*.



Slika 44. Izbornik *Script Object*

S obzirom da *python* modul u GOM Inspect-u nema dobru dokumentaciju, kako bi se došlo do načina na koji su zadani parametri te da bi se stvorile potrebne linije koda, za točke na referentnom CAD modelu i na ispitnom uzorku otvoreni su padajući izbornici *Center coordinate* i *Normal* te su označene osi X, Y i Z. Na taj način stvorene su linije koda označene na slici 45 brojevima od 1 do 4. Oznake "*Nom*" odnose se na referentni CAD model, a oznake "*Act*" se odnose na zamaknuti ispitni uzorak. Linije koda označene brojem 1 prikazuju funkcije kojima su definirane X, Y i Z koordinate kontrolnih točaka, a linije koda označene brojem 2 prikazuju funkcije kojima su definirane duljine i smjerovi projekcija jediničnih vektora referentnog CAD modela na X, Y i Z osi globalnog koordinatnog sustava. Pod brojem 3 prikazane su funkcije kojima su definirane koordinate kontrolnih točaka, a pod brojem 4 su prikazane funkcije kojima su definirane duljine i smjerovi projekcija jediničnih vektora zamaknutog ispitnog uzorka.

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

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

        xNom = gom.app.project.inspection[ptName].center_coordinate.x          } 1
        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                     } 2
        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     } 3
        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                } 4
        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 45. Definiranje koordinata točaka i vektora normala

Navedeni podaci pohranjeni su u tipu podataka rječnika (eng. *dictionray*). Rječnik se primjenjuje za pohranu podataka u obliku ključeva i vrijednosti. Ključevi mogu biti i podaci tipa *string* i brojevi, a bitno je da imaju jedinstvenu oznaku kako bi se svaki ključ mogao povezati s pripadajućom vrijednosti [20]. Slika 46 prikazuje način na koji su podaci za točku 1 pohranjeni u obliku rječnika. Ključevi su definirani simbolima kao što su na primjer *"Xnom"*, *"Ynom"* i *"Znom"* te je svakom ključu pridružena vrijednost.

{'id': 1,
'Xnom': 80.0, 'Xact': 80.0,
'Ynom': 105.0, 'Yact': 105.0,
'Znom': 0.0, 'Zact': 1.001805567040258,
'iNom': -0.0, 'iAct': 2.5036305945950676e-16,
'jNom': -0.0, 'jAct': -0.05966273672725364,
'kNom': 1.0, 'kAct': 0.9982185922162612
}

Slika 46. Podaci o točki 1 u obliku rječnika

Na slici 47 je prikazan dio programskog koda kojim se definiraju podaci u referentnom RAPID kodu koji će biti zamijenjeni prilikom ažuriranja na temelju analize odstupanja koordinata i vektora normala. U prvom dijelu koda definirano je da će se ažurirati dio RAPID koda koji se nalazi između oznaka "! Pocetak" i "! Kraj". Iz navedenog dijela RAPID koda svaka od linija je izdvojena te joj je dodijeljena oznaka "ORIGINAL LINE". Također, kako bi se olakšala kontrola novog koda, definirano je da se linije referentnog RAPID koda ne brišu, već da se označe kao komentar. U svakoj od originalnih linija, koordinate kontrolnih točaka izdvojene su na način da se traže podaci u dijelu linije koda koji započinje s prvom pojavom simbola "[[", a završava sa simbolom ']'. Istom metodom izdvojeni su i kvaternioni iz referentnog RAPID koda, samo što su u ovom slučaju izdvojeni podaci koji se nalaze unutar uglatih zagrada koje slijede nakon uglatih zagrada u kojima se nalaze koordinate kontrolnih točaka. Izdvojeni podaci o koordinatama kontrolnih točka označeni su varijablom *_srch_koordinate_tocaka_str*, a podaci o kvaternionima označeni su varijablom *_srch_kvarternioni_str*.

```
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())

            _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)
```

Slika 47. Izdvajanje podataka iz referentnog RAPID koda

Vektori normala na površinu zakrenutog ispitnog uzorka suprotne su orijentacije u odnosu na Z os koordinatnog sustava alata. Zbog navedenog, potrebno je matricom rotacije (slika 48) definirati koordinatni sustav alata u kojem će Z os biti istog smjera, ali suprotne orijentacije u odnosu na jedinični vektor normale na površinu zakrenutog ispitnog uzorka.

$$R = \begin{bmatrix} i_x & i_y & i_z \\ j_x & j_y & j_z \\ k_x & k_y & k_z \end{bmatrix}$$

Slika 48. Komponente matrice rotacije

Da bi se definirano koordinatni sustav, najprije je potrebno odrediti vektor normale i pomoćni vektor za definiciju orijentacije. X os koordinatnog sustava vrha alata robota na referentnom ispitnom uzorku korištena je kao vektor za definiciju orijentacije. Zbog toga, u prvom dijelu programskog koda na slici 49, iz linija referentnog RAPID koda izdvojeni su kvaternioni označeni varijablama *_orig_quat[]*.

```
_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]

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

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 49. Transformacija matrice rotacije u kvaternione

Pomoću izdvojenih podataka, programskim kodom na slici 39 izračunate su komponente jediničnog vektora koji predstavlja X os koordinatnog sustava vrha alata. Funkcijom *np.zeros(3)* stvoren je 1x3 vektor te su u njega unese izračunate i_x, j_x i k_x komponente. Jedinični vektor koji definira Z os koordinatnog sustava vrha alata određen je korištenjem vrijednosti projekcija vektora normale zamaknutog ispitnog uzorka. Ponovno je funkcijom *np.zeros(3)* stvoren 1x3 vektor gdje je na prvo mjesto ([0]) unesena vrijednost iz rječnika povezena s ključem "*iAct*", na drugo mjesto ([1]) je unesena vrijednost povezana s ključem "*jAct*", a na treće mjesto ([2]) je unesena vrijednost povezana s ključem "*kAct*" Kako bi se promijenila orijentacija vektora, sve tri komponente pomnožene su s -1. Nakon što su stvoreni vektor za definiciju orijentacije (stara X os) te vektor normale na površinu (Z os), korištenjem naredbe za vektorski produkt *np.cross()* stvoren je vektor koji predstavlja Y os. Ponovnom implementacijom funkcije *np.cross()* te izračunom vektorskog produkta između prethodno definiranih vektora, dobiven je vektor koji predstavlja X os koordinatnog sustava alata. Prije no što se uvrste u matricu rotacije, oba dobivena vektora su normalizirana, tj. pretvorena u jedinične vektore dijeljenjem s vlastitom duljinom. Primjenom izračunatih komponenta, funkcijom *np.matrix()* stvorena je matrica rotacije kojom je definiran koordinatni sustav alata na zamaknutom ispitnom uzorku. Programski kod kojim je definirana matrica rotacije prikazan je na slici 50.

```
def jv2rotmtx(vecNorm2Surf, vecOri):

    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]]])

    return rotMtx
```

Slika 50. Programski kod za definiciju matrice rotacije

Dobivenom matricom rotacije definiran je koordinatni sustav vrha alata robota kojemu je smjer Z osi određen vektorom normale na površinu zamaknutog ispitnog uzorka te je suprotno orijentiran u odnosu na normalu površine. Prije no što se u RAPID kodu referentne vrijednosti koordinata točaka i kvaterniona zamjene novim, ažuriranim vrijednostima, potrebno je matricu rotacije pretvoriti u kvaternione. Za pretvorbu je korišten programski kod prikazan na slici 51. Programski kod je preuzet iz izvora [21], a s obzirom da je originalna verzija napisana u programskom jeziku Java, programski kod je modificiran kako bi ispravno funkcionirao u programskom jeziku Python.

```
def rotmtx2quat(rotMtx):

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

    if tr > 0:
        s = np.sqrt(tr+1.0) * 2
        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
        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
        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
        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 51. Pretvorba matrice rotacije u kvaternione

Na slici 52 je prikazan programski kod kojim se u referentni RAPID kod na mjesto koordinata točaka i kvaterniona ubacuju novi podaci. Prvim dijelom koda definirani su podaci označeni varijablom *_repl_koordinate_tocaka_str* u kojima se nalaze X, Y i Z koordinate kontrolnih točaka na površini zamaknutog ispitnog uzorka. Na isti su način definirani podaci označeni varijablom *_repl_kvaternioni_str* u kojima se nalaze w, x, y i z komponente kvaterniona. Nakon toga je korištena metoda *replace()* kojom se u svakoj liniji koda podaci označeni varijablom *_srch_koordinate_tocaka_str* mijenjaju podacima označenima varijablom *_repl_koordinate_tocaka_str*. Također, podaci označeni varijablom *_srch_kvaternioni_str* mijenjaju se s podacima označenima varijablom *_repl_kvaternioni_str*. Programski kod se ponavlja sve dok se ne ažurira svaka točka putanje alata, a na kraju se ažurirani RAPID kod pohranjuje na lokaciju definiranu na početku koda.

```
    _repl_koordinate_tocaka_str= "%.3f,%.3f,%.3f" % (_matchingPt["Xact"], _matchingPt["Yact"], _matchingPt["Zact"])
    print("REPL PT DATA:", _srch_koordinate_tocaka_str, "->", _repl_koordinate_tocaka_str)

    _repl_karternioni_str= "%.8f,%.8f,%.8f,%.8f" % (_act_quat[0], _act_quat[1], _act_quat[2], _act_quat[3])
    print("REPL ANGLE DATA:", _srch_karternioni_str, "->", _repl_karternioni_str)

    repl_line = line.replace(_srch_koordinate_tocaka_str, _repl_koordinate_tocaka_str)

    repl_line = repl_line.replace(_srch_karternioni_str, _repl_karternioni_str)

    print("UPDATED LINE:",repl_line.lstrip())

    outFile.write(repl_line)

    print("\n")

    numLineToUpdate +=1
else:
    outFile.write(line)
```

Slika 52. Ažuriranje koordinata točaka i kvaterniona u RAPID kodu

Na slici 53 prikazan je dio ažuriranog koda u kojem su vidljive ažurirane koordinate točaka putanje alata te kvaternioni.

```
! Pocetak
 ! ORIGINAL LINE: MoveL [[80.000,105.000,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],[
MoveL [[80.000,104.968,1.833],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+09
 ! ORIGINAL LINE: MoveL [[355.000,105.000,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000]
MoveL [[355.000,104.968,1.833],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
 ! ORIGINAL LINE: MoveL [[362.028,105.381,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000]
MoveL [[362.028,105.349,1.840],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
 ! ORIGINAL LINE: MoveL [[368.973,106.520,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000]
MoveL [[368.973,106.488,1.860],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
 ! ORIGINAL LINE: MoveL [[375.755,108.403,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000]
MoveL [[375.755,108.370,1.892],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
 ! ORIGINAL LINE: MoveL [[382.293,111.008,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000]
MoveL [[382.293,110.974,1.938],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
 ! ORIGINAL LINE: MoveL [[388.511,114.304,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000]
MoveL [[388.511,114.269,1.995],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
 ! ORIGINAL LINE: MoveL [[394.336,118.254,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],
MoveL [[394.336,118.218,2.064],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
 ! ORIGINAL LINE: MoveL [[399.700,122.810,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000]
MoveL [[399.700,122.773,2.144],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
 ! ORIGINAL LINE: MoveL [[404.541,127.920,0.000],[0.00000000,-0.00000000,1.00000000,0.00000000]
MoveL [[404.541,127.881,2.233],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
 ! ORIGINAL LINE: MoveL [[408.800,133.523,0.000],[0.00000000,0.00000000,1.00000000,-0.00000000]
MoveL [[408.800,133.482,2.331],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
 ! ORIGINAL LINE: MoveL [[412.428,139.553,0.000],[0.00000000,0.00000000,1.00000000,0.00000000],
MoveL [[412.428,139.510,2.436],[-0.00000000,0.00000000,0.99996192,0.00872653],[-1,1,-2,0],[9E+0
```

Slika 53. Ažurirani RAPID kod

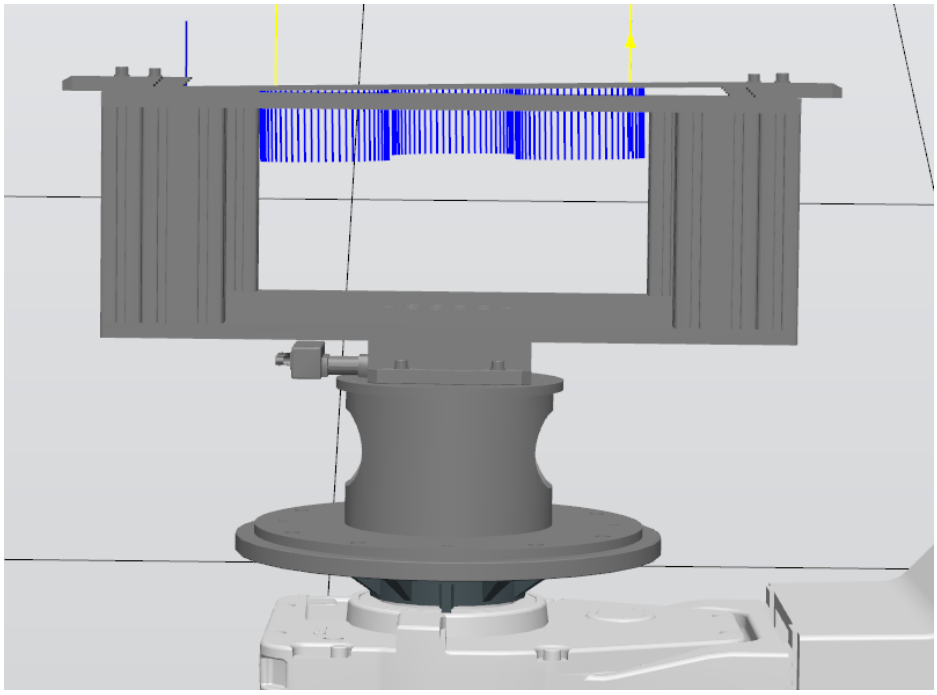
5.5. GOM Inspect predložak

Prilikom kreiranja novog projekta, nakon što se u softver učitaju i definiraju stezna naprava te CAD model referentnog ispitnog uzorka, za prikaz podataka o odstupanju kontrolnih točaka potrebno je pokrenuti prethodno snimljene skripte. Pomoću snimljenih skripta će se na ispitnim uzorcima konstruirati kontrolni elementi te izvršiti glavno poravnanje. Nakon što se za prvi ispitni uzorak izvrše potrebne naredbe, moguće je ukloniti ispitni uzorak te projekt spremiti u obliku predloška (eng. *Template*). Stvaranjem predloška se postupak formiranja mjernog postava i analize u potpunosti automatizira. Otvaranjem spremljenog projekta u obliku predloška potrebno je samo učitati STL datoteku ispitnog uzorka bez pokretanja skripti. Na taj način softver na svakom novom ispitnom uzorku izvršava sve naredbe definirane na prvom ispitnom uzorku te kao izlaz daje podatke o odstupanjima koji se koriste u programskom kodu za korekciju RAPID koda.

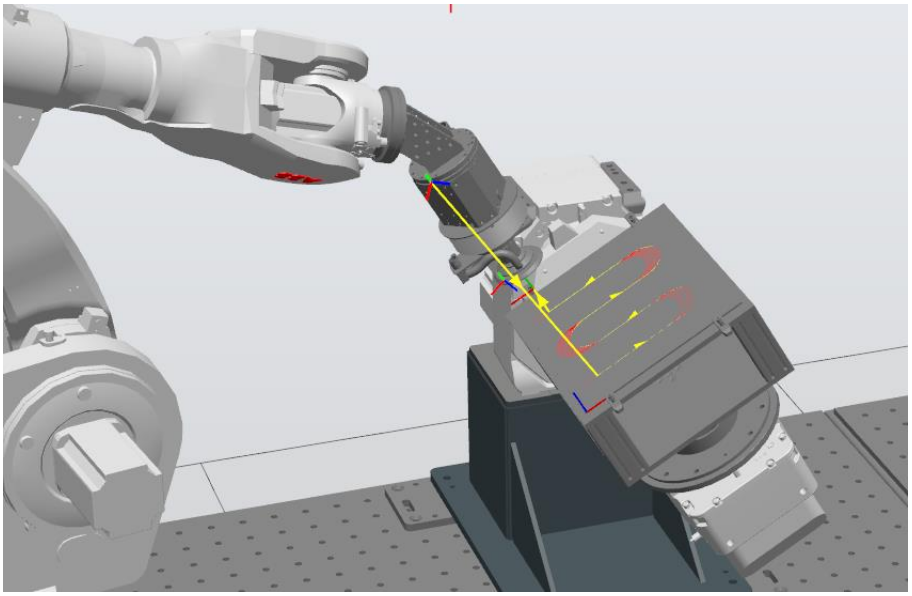
6. SIMULACIJA KORIGIRANOG NC PROGRAMA

Da bi se provjerila ispravnost rada programskog koda za korekciju putanje alata, ažurirani RAPID kodovi učitani su u programskom paketu ABB RobotStudio te je simulirano brušenje zamaknutih ispitnih uzoraka. Za prvi ispitni uzorak korišten je 3D model ploče zakrenute za 1˚ oko osi X koordinatnog sustava obrade, što na krajnjem rubu daje grešku položaja po visini za 10,47 mm. Prvi ispitni uzorak učitan je u predložak programskog paketa GOM Inspect u kojem se na površinu ploče projiciraju točke putanje alata te je pokretanjem programskog koda za korekciju putanje alata RAPID kod ažuriran. 3D model sklopa stezne naprave i ispitnog uzorka spremljen je u obliku STL datoteke te je učitan u programski paket ABB RobotStudio (slika 54).

Nakon što je ispitni uzorak postavljen na okretno-nagibni stol, naredbom *Load Module* učitan je ažurirani RAPID kod. Slika 55 prikazuje korigiranu putanju alata na zakrenutom ispitnom uzorku. Korigirana putanja alata nalazi se na površini ispitnog uzorka, a pokretanjem simulacije utvrđeno je da je putanja korigirana na ispravan način.

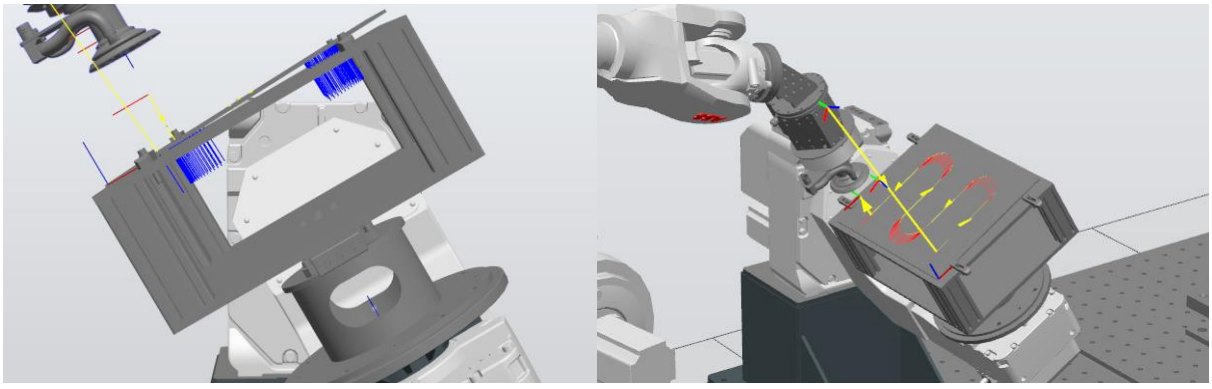


Slika 54. Zakrenuti ispitni uzorak na okretno-nagibnom stolu

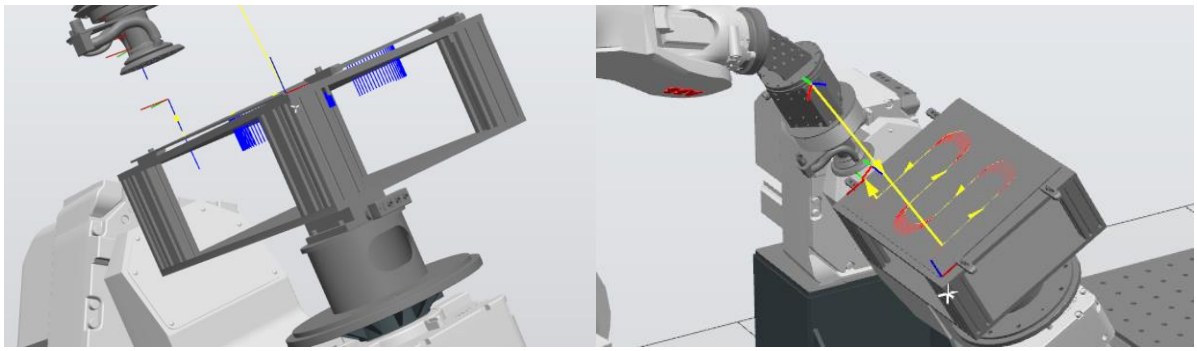


Slika 55. Korigirana putanja alata na površini zakrenutog ispitnog uzorka

S ciljem verifikacije ispravnosti rada programskog koda za korekciju putanje alata napravljeni su još ispitni uzorak zakrenut oko Y osi za 1˚, što na krajnjem rubu daje grešku položaja po visini za 8,73 mm (slika 56) te ispitni uzorak zakrenut oko osi X, a nakon toga osi Y koordinatnog sustava obrade, što na krajnjem uglu daje grešku položaja po visini za 19,20 mm (slika 57). Neovisno o načinu zakreta ispitnog uzorka, koordinate točaka alata nalaze se na površini ispitnog uzorka te je pokretanjem simulacije nagib alata korigiran na ispravan način.

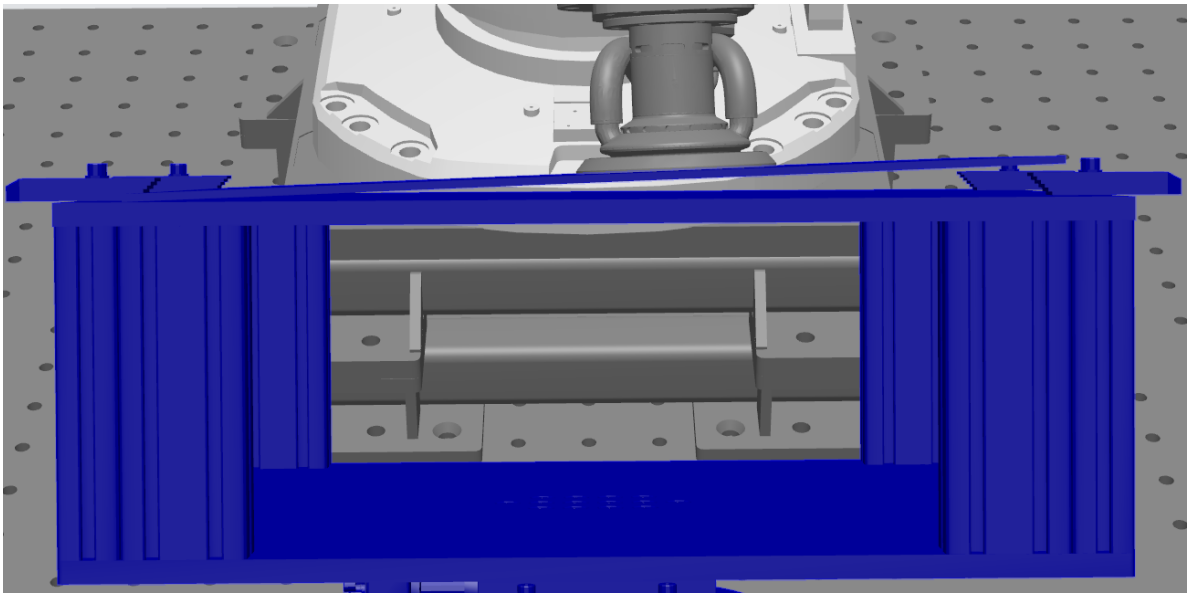


Slika 56. Korigirana putanja alata na površini ispitnog uzorka zakrenutog oko Y osi



Slika 57. Korigirana putanja alata na površini ispitnog uzorka zakrenutog oko X i Y osi

Kako bi se provjerila funkcionalnost programskog koda za korekciju putanje alata za odstupanja pri kojima se rub ispitnog uzorka nalazi iznad steznih čeljusti (slika 58, desna strana), izrađen je 3D model sklopa stezne naprave na kojoj je ispitni uzorak zakrenut oko osi X za 2˚, što na krajnjem rubu daje grešku položaja po visini za 20,94 mm. Navedeni 3D model i ažurirani RAPID kod učitani su u ABB RobotStudio te je simulacijom potvrđeno da programski kod ispravno korigira koordinate točaka i nagib alata i u slučaju kada se ispitni uzorak nalazi iznad steznih čeljusti.



Slika 58. Simulacija obrade ispitnog uzorka zakrenutog za 2˚ oko X osi

7. ZAKLJUČAK

U ovom je diplomskom radu opisana simulacija primjene digitalizatora u korekciji putanje alata prema odstupanju položaja i orijentacije kod stezanja obratka. U sklopu rada izrađen je programski kod koji na temelju analize rezultata digitalizacije ažurira koordinate točaka te kvaternione kojima je definirana orijentacija vrha alata robota u RAPID kodu. Korekcijom putanje alata bi se unatoč greškama prilikom stezanja mogla postići ujednačena kvaliteta obrađene površine. Također, skraćenjem vremena potrebnog za pravilno pozicioniranje i stezanje pripremka, povećala bi se produktivnost robotske ćelije.

U CAD/CAM sustavu CATIA V5R21 izrađeni su 3D modeli ispitnih uzoraka te je generiran APT kod za brušenje ispitnog uzorka koji je učitan u programskom paketu RoboDK. Dobiveni program spremljen je u obliku RAPID koda koji je potreban za definiranje gibanja ABB industrijskih robota. Zatim je u programskom paketu GOM Inspect formiran postav u kojem je 3D model sklopa stezne naprave i ispravno pozicioniranog ispitnog uzorka korišten kao idealni CAD model, a 3D modeli zamaknutih ispitnih uzoraka korišteni su kao simulirani rezultati digitalizacije. Postupak pripreme mjernog postava u programskom paketu GOM Inspect je u potpunosti automatiziran, no programski kod za poravnanje elemenata analize vrijedi samo u konkretnom slučaju. U slučaju promjene stezne naprave ili ispitnog uzorka, postupak konstruiranja kontrolnih elemenata te poravnanja je potrebno ponoviti.

U programskom jeziku Python napravljen je kod koji iz generiranog RAPID koda izuzima koordinate putanje i orijentaciju vrha alata robota te ih u GOM Inspect-u na idealnom CAD modelu kreira u obliku kontrolnih točaka te pripadajućih vektora normala na površinu. Dobivene kontrolne točke su zatim projicirane na površinu zamaknutog ispitnog uzorka te je izrađen programski kod kojim se na temelju analize odstupanja projiciranih točaka referentne vrijednosti koordinata i orijentacije alata zamjenjuju novim vrijednostima.

Ispravnost programskog koda za korekciju putanje alata verificirana je u programskom paketu ABB RobotStudio. U ARCOPS robotsku ćeliju učitani su dislocirani ispitni uzorci te pripadajući ažurirani RAPID kod. Simulirana je obrada uzoraka zakrenutih oko X i Y osi koordinatnog sustava predviđenog za obradu te je potvrđeno da programski kod ispravno korigira koordinate točaka putanje te nagib alata. Unatoč činjenici da je simulacijom potvrđena ispravnost programskog koda za korekciju putanje alata, funkcionalnost sustava je potrebno provjeriti u stvarnim uvjetima. Također, potrebno je provjeriti utjecaj veličine odstupanja na ujednačenost kvalitete obrađene površine kao i na trošenje alata.

LITERATURA

[1] Haleem, A., Javaid, M., Singh, R. P., Rab, S., Suman, R., Kumar, L., Khan, I. H. (2022). Exploring the potential of 3D scanning in Industry 4.0: An overview. *International Journal of Cognitive Computing in Engineering, 3*, 161-171. ISSN 2666-3074. https://doi.org/10.1016/j.ijcce.2022.08.003

[2] Javaid, M., Haleem, A., Singh, R. P., Suman, R. (2021). Industrial perspectives of 3D scanning: Features, roles and its analytical applications. Sensors International, 2, 100114. ISSN 2666-3511. https://doi.org/10.1016/j.sintl.2021.100114

[3] https://www.pre-scient.com/knowledge-center/product-development-by-reverse-engineering/scanners-scanning.html, pristupljeno 21. listopada 2023.

[4] Ebrahim, M. A.-B. (2015). "3D Laser Scanners' Techniques Overview." *International Journal of Science and Research (IJSR), 4*(10). ISSN 2319-7064.

[5] https://hermary.com/learning/principles-of-laser-triangulation/, pristupljeno 21. listopada 2023.

[6] https://www.aniwaa.com/guide/3d-scanners/3d-scanning-technologies-and-the-3d-scanning-process/, pristupljeno 21. listopada 2023.

[7] https://www.movimed.com/knowledgebase/what-is-structured-light-imaging/, pristupljeno 22. listopada 2023.

[8] https://www.artec3d.com/learning-center/what-is-photogrammetry, pristupljeno 22. listopada 2023.

[9] https://library.e.abb.com/public/4bc775cd7374c09ac1257bf30030c01a/ROB0053%20EN_D_HR.pdf, pristupljeno 24. listopada 2023.

[10] https://new.abb.com/products/3HAC020536-013/irb-6660, pristupljeno 24. listopada 2023.

[11] https://new.abb.com/products/3HAC020536-014/irc5-controller?HR, pristupljeno 24. listopada 2023.

[12] https://search.abb.com/library/Download.aspx?DocumentID=ROB0109EN_G&LanguageCode=en&DocumentPartId=&Action=Launch, pristupljeno 24. listopada 2023.

[13] https://new.abb.com/products/3HAC020536-017/irb-4600?HR, pristupljeno 24. listopada 2023.

[14] https://www.qa-group.com/files/downloads/quality-analysis-optical-metrology-technical-equipment.pdf, pristupljeno 25. listopada 2023.

[15] https://topomatika.hr/proizvodi/3d-skeneri/atos-5-5x-industrijsko-mjeriteljstvo/, pristupljeno 24. listopada 2023.

[16] https://search.abb.com/library/Download.aspx?DocumentID=ROB10080EN_R3&LanguageCode=en&DocumentPartId=&Action=Launch, pristupljeno 27. studenog 2023.

[17] https://new.abb.com/products/robotics/robotstudio, pristupljeno 25. listopada 2023.

[18] https://www.mecademic.com/academic_articles/quaternions-in-industrial-robotics/, pristupljeno 10. studenog 2023.

[19] Technical reference manual - RAPID Instructions, Functions and Data types, pristupljeno 16. studenog 2023.

[20] https://docs.python.org/3/tutorial/datastructures.html, pristupljeno 16. studenog 2023.

[21] https://www.euclideanspace.com/maths/geometry/rotations/conversions/matrixToQuaternion/index.htm, pristupljeno 16. studenog 2023.

PRILOZI

I. CD-R disc