Usporedba primjenjivosti standardnih industrijskih robota u neurokirurgiji

Koren, Petar

Master's thesis / Diplomski rad

2017

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

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:235:015488

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Petar Koren

Zagreb, 2017.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

DIPLOMSKI RAD

Mentor:

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

Student:

Petar Koren

Zagreb, 2017.

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

Zahvaljujem se mentoru Prof. dr. sc. Bojanu Jerbiću na pomoći i sugestijama prilikom izrade diplomskog rada.

Posebno bih se zahvalio kolegi Dr. sc. Marku Švaci, dipl. ing. na savjetima, pomoći, strpljenju i razumijevanju koje je pokazao tokom izrade ovog rada.

Naposljetku bih se želio zahvaliti svojoj obitelji na velikoj potpori i inspiraciji tijekom cjelokupnog studija.

Petar Koren

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

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

DIPLOMSKI ZADATAK

Student: **Petar Koren** Mat. br.: 0035183482

Naslov rada na hrvatskom jeziku: **Usporedba primjenjivosti standardnih industrijskih robota u neurokirurgiji**

Naslov rada na engleskom jeziku: **Comparison of standard industrial robots for neurosurgery application**

Opis zadatka:

Primjena i razvoj novih robotskih sustava u području neurokirurgije bilježe značajan rast. Zbog specifičnih zahtjeva u području neurokirurgije u kojima se tradicionalno koriste stereotaktički okviri kao što su Leksellov, CRW ili ZD okvir, danas je razvijen niz robotskih sustava sa specifičnom kinematikom kao što su Neuromate, Neuroarm, MARS, itd. S druge strane razvijaju se robotizirane platforme bazirane na standardnim 6R (revolutnim) strukturama kao što su ROSA, Aqrate ili Pathfinder.

U radu je potrebno ispitati primjenjivost tri verzije industrijskog robota KUKA KR6 Agilus sa dosegom: 700, 900 i 1100 mm. U 3D virtualnom okruženju operacijske sale potrebno je identificirati radni prostor robota u kojem je moguće u potpunosti zadovoljiti sve operativne zahtjeve manipulacije neurokirurškim instrumentima u intrakranijalnom prostoru pacijenta. Osnovni ulazni parametri su maksimalan vertikalni i horizontalan kut manipulacije oko centra radne sfere, kut rotacije oko osi alata, mogućnosti linearne kretnje po osi neurokirurškog alata (svrdlo, biopsijska sonda), dimenzija radnog (operativnog) prostora robota.

Identificirani radni prostor za sve tri robotske konfiguracije (R700, R900 i R1100) potrebno je grafički prikazati kao 2D i 3D radne mape u ovisnosti o ulaznim parametrima.

Identificirani radni prostor za robot Kuka KR6 Agilus R900 potrebno je eksperimentalno verificirati na sustavu RONNA u Laboratoriju za projektiranje izradbenih i montažnih sustava.

Zadatak zadan:	Rok predaje rada:	Predviđeni datum obrane:
17. studenog 2016.	19. siječnja 2017.	25., 26. i 27. siječnja 2017.

Zadatak zadao: v. d. predsjednika Povjerenstva:

Prof. dr. sc. Bojan Jerbić

Prof. dr. sc. Biserka Runje

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

POPIS ALGORITAMA

POPIS OZNAKA

Oznaka	Jedinica	Opis
x	mm	Pozicija na x osi Kartezijevog koordinatnog sustava
x_{start}	mm	Početna vrijednost x pozicije
x_{end}	mm	Završna vrijednost x pozicije
$x_{division}$	-	Broj podjela između x_{start} i x_{end}
y	mm	Pozicija na y osi Kartezijevog koordinatnog sustava
y_{start}	mm	Početna vrijednost y pozicije
y_{end}	mm	Završna vrijednost y pozicije
$y_{division}$	-	Broj podjela između y_{start} i y_{end}
z	mm	Pozicija na z osi Kartezijevog koordinatnog sustava
z_{start}	mm	Početna vrijednost z pozicije
z_{end}	mm	Završna vrijednost z pozicije
$z_{division}$	-	Broj podjela između z_{start} i z_{end}
A_n	-	n-ti zglob robota
θ_i	°	Zakret zgloba oko z osi
d_i	mm	Pomak segmenta po z osi
a_i	mm	Pomak segmenta po x osi
α_i	°	Zakret zgloba oko x osi
r	mm	Polumjer kuglinog isječka
W^F	-	Matrica homogene transformacije prirubnice
W^T	-	Matrica homogene transformacije vrha alata
F^T	-	Matrica homogene transformacije vrha alata u prirubnicu
ϑ	°	Kut rotacije oko osi alata
α	°	Kut vertikalne rotacije
α_{start}	°	Iznos početnog kuta vertikalne rotacije
α_{end}	°	Iznos završnog kuta vertikalne rotacije
$\alpha_{division}$	-	Broj podjela između α_{start} i α_{end}

β	°	Kut horizontalne rotacije
β_{start}	°	Iznos početnog kuta horizontalne rotacije
β_{end}	°	Iznos početnog kuta horizontalne rotacije
$\beta_{division}$	-	Broj podjela između β_{start} i β_{end}

POPIS KRATICA

Oznaka	Opis
MIK	Minimalno invazivna kirurgija
RIK	Računalno integrirana kirurgija
RPK	Računalno pomagana kirurgija
VUK	Vizijski upravljana kirurgija
CT	Kompjutorska tomografija
MR	Magnetska rezonanca
$RA-MIK$	Robotski asistirana minimalno invazivna kirurgija
RM sustav	Riechert – Mundinger sustav
ZD sustav	Zamorano – Duchovny sustav
OpenRAVE	Open Robotics Automation Virtual Environment
XML	EXtensible Markup Language
CSV	Comma – separated values
TCP	Tool Center Point
F/R	Front/Rear orijentacija prvog zgloba robota
U/D	Elbow Up/Down orijentacija drugog zgloba robota
F/N	Flip/Non-Flip orijentacija četvrtog zgloba robota

SAŽETAK

Predmet istraživanja rada je usporedba primjenjivosti tri Kuka Agilus industrijska robota u neurokirurgiji. Zadatak je ispitati mogućnost dohvata svakog od robota, te identificirati optimalno radno područje u kojemu će robot moći zadovoljiti sve operativne zahtjeve u intrakranijalnom prostoru pacijenta. Uvodni dio rada bazira se na kratkom pregledu područja stereotaktičkih okvira i robotskih stereotaktičkih sustava. Kako bi se uočile ključne razlike između Agilus robota dan je uvid u njihove osnovne karakteristike. U svrhu identificiranja optimalnog radnog prostora razvijen je algoritam u Python i OpenRAVE programskim paketima, te je opisan primjerima, slikama i pseudokodovima. Istraživani intrakranijalni prostor pacijenta bazira se na geometrijskom obliku kuglinog isječka kojeg zatvaraju standardni stereotaktički okviri. Eksperimentalni dio rada obuhvaća simulacije različitih ulaznih parametara te prikaz 2D i 3D grafičkih mapa radnog prostora za sve tri robotske konfiguracije.

Ključne riječi: *robotika, neurokirurgija, stereotaksija, Kuka Agilus*

SUMMARY

The topic of this thesis is the comparison of the applicability of three Kuka Agilus industrial robots in neurosurgery. The Task was to examine the reach of each of three robots and identify the optimal working area in which robot will meet all operational requirements in the intracranial space of the patient. The introductory part focuses on a short overview of stereotactic frames and robotic stereotactic systems. In order to determine the key differences between Agilus robots, their basic characterictics will be presented. To identify the optimal workspace, an algorythm has been developed in Python and OpenRAVE software packages, and it has been described with examples, pictures and pseudocodes. The studied intercranial area of the patient is based on the geometrical shape of spherical cone closed by standard stereotactic frames. The experimental part covers simulations with different input parameters, as well as 2D and 3D graphic maps of the working area for all three robot configurations.

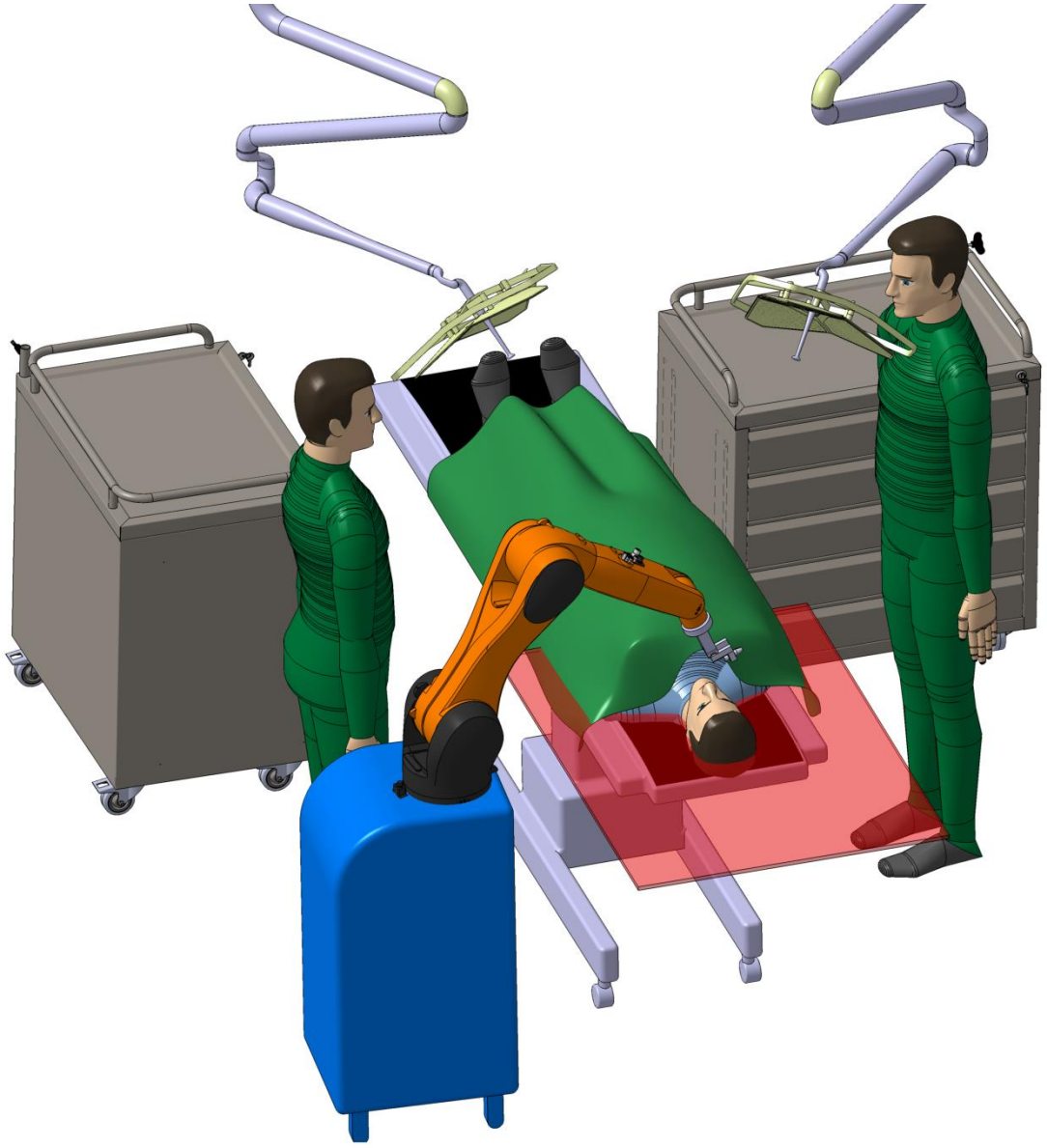
Key words: *robotics, neurosurgery, stereotactic, Kuka Agilus*

1. UVOD

Razvojem ljudskih vještina, znanja i tehnologija dolazi do ekspanzije na svim područjima čovjekovog djelovanja. Još u 15. stoljeću, poznati talijanski arhitekt, izumitelj, mislilac, Leonardo da Vinci, dao je veliki obol na području strojarstva i tehnike općenito [1]. Potreba za masovnom proizvodnjom, a osobito u automobilskoj industriji, u svrhu ubrzanja i olakšanja rada, dovela je do automatizacije pogona proizvodnje, te razvoja robotike. U 20. stoljeću, točnije 1921. godine, češki dramatičar Karel Čapek skovao je riječ robot, od češke riječi „*robotnik"* što u prijevodu znači rob, radnik. Robotika je grana inžinjerske znanosti koja se bavi projektiranjem, konstruiranjem i upravljanjem robotima [2].

Upotreba robota najčešća je u industriji, za vojne svrhe, svemirska istraživanja, a danas sve više u kućanstvima te za medicinske svrhe. Upravo je potonje nabrojano, izazov s kojim se inženjeri u suradnji s medicinskim stručnjacima, sve više hvataju ukoštac. Područja medicine s primjenom robota u nekom obliku su: oftalmologija, urologija, ginekologija, kardiologija, neurologija i dr. Kirurška robotika je najfascinantnije, a ujedno najsloženije i najsofisticiranije interdisciplinarno područje medicinskog inženjerstva. Neurokirurgija je osobito prikladna za primjenu robota, ali zbog slojevitosti anatomske građe, visoke osjetljivosti i delikatne funkcionalnosti tkiva, zahvati ne dozvoljavaju niti minimalne pogreške [3].

Cilj ovoga rada je iskoristiti robotski sustav za identifikaciju radnog prostora operacijske sale u kojemu je moguće u potpunosti zadovoljiti sve operativne zahtjeve manipulacije neurokiruškim instrumentima u intrakranijalnom prostoru pacijenta. Motiv rada temelji se na upotrebi standardnih industrijskih Kuka robota umjesto robotskih sustava specifične kinematike. Takav pristup problemu doveo je do razvoja pouzdanog, preciznog, mobilnog robotskog sustava za neuronavigaciju u slučaju RONNA-e.



Slika 1 – Identifikacija radnog prostora s maksimalnim dosegom robota u intrakranijalnom prostoru pacijenta

2. PLAN RADA

Kako je navedeno ranije, upotreba robota u neurokirurgiji osobito je prikladna, ali i izrazito kompleksna. Roboti se u neurokiruškim operacijama koriste na više načina i s različitim stupnjem samostalnosti. U pripremi i tijeku neurokiruške operacije tradicionalno se koriste stereotaktički okviri kao što su *Leksellov, CRW* ili *ZD* okvir. Kao zamjena za navedene instrumente razvijen je niz robotskih sustava sa specifičnom kinematikom kao što su *Neuromate*, *Neuroarm*, *MARS*. S druge strane razvijaju se robotizirane platforme sa standardnim revolutnim robotima kao što su *ROSA*, *Aqrate* ili *Pathfinder*.

Ideja ovoga rada je kreirati 3D virtualno okruženje u *OpenRave* softveru, te izraditi algoritam za kretanje robota kojim bi se nadomjestila upotreba standardnih stereotaktičkih okvira. Programiranje će biti izvršeno u *Python* programskom jeziku. U radu će se ispitati primjenjivost tri verzije industrijskog robota KUKA Agilus s dosegom: 700, 900 i 1100 mm. Osnovni ulazni parametri u programu bit će:

- maksimalan vertikalan i horizontalan kut manipulacije oko centra radne sfere
- kut rotacije oko osi alata
- mogućnosti linearne kretnje po osi neurokirurškog alata
- dimenzija operativnog prostora robota

Sklapanje CAD modela robota potrebnih za kreiranje virtualnog okruženja izvršit će se u CATIA programskom paketu, zatim će uslijediti konfiguracija XML datoteka potrebnih za rad OpenRAVE računalnog alata te pisanje algoritma za kretanje robota u Python programu. Za sve tri robotske konfiguracije bit će grafički prikazane radne mape u ovisnosti o ulaznim parametrima. Naposljetku, identificirani radni prostor za robot Kuka KR6 Agilus R900, koji je sastavni dio RONNA sustava, bit će eksperimentalno verificiran u Laboratoriju za projektiranje izradbenih i montažnih sustava na Fakultetu strojarstva i brodogradnje.

3. PREGLED NEUROKIRUŠKIH SUSTAVA

Robotizirana kirurgija definirana je od SAGES – MIRA Robotic Consensus Grupe kao „Kirurški zahvat ili tehnologija koja dodaje uređaj poboljšan kompjuterskom tehnologijom u interakciju između kirurga i pacijenta tijekom operacije i pretpostavlja određeni stupanj slobode kontrole koja je do sada bila u potpunosti rezervirana za kirurge. Ova definicija obuhvaća mikromanipulatore, tele – operirane endoskope i konzolno – manipulatorske uređaje. Glavni elementi su poboljšanje sposobnosti kirurga: vizijski, rukovanje tkivom ili dobivanje povratnih informacija prilikom rukovanja tkivom i izmjena tradicionalnog direktnog lokalnog dodira između kirurga i pacijenta."

Minimalno invazivna kirurgija (**MIK**) izvorno se odnosi na laparoskopske zahvate (laserska kirurgija), gdje se kroz abdomensku šupljinu pristupa kroz 3 – 5 malih rezova ($0.5 - 3\ cm$). Ovakav postupak prvi puta je zabilježen na ljudima 1910. godine, a od tada su razvijene i brojne druge metode kako bi se pristupilo različitim dijelovima ljudskog tijela. U današnje vrijeme sve je popularnija alternativa otvorenim zahvatima što u većini slučajeva za posljedicu ima smanjenje vremena oporavka i opasnosti tijekom zahvata.

Računalno integrirana kirurgija (**RIK**) najčešće je upotrebljavani izraz kojim se pokriva cijelo područje intervencijskih medicinskih tehnologija od obrade slike i širih primjena u stvarnosti do automatskog uklanjanja tkiva. Potpodručje *Računalno pomagana kirurgija* (**RPK**) uobičajeno znači da digitalni sustav koji je uključen ne sudjeluje u fizičkom dijelu operacije nego doprinosi kvaliteti operacije poboljšavajući vizualizaciju ili navigaciju.

Vizijski upravljana kirurgija (**VUK**) djelomično pokriva područje RPK-a te postoji i prije upotrebe robotike u medicini. Ideja stereotaksije potječe od 1906. godine iako je prvi podkorteksni zahvat izveden 1947. Navedena tehnika izvorno je bila namijenjena za poboljšanje operacija tumora mozga i postala je popularna od 70 – ih godina zbog pojave jeftinijih računala i naprednijih programa za obradu slike. VUK označava real – time registraciju (korelaciju i mapiranje) područja operacije (operacijske sale) do predoperacijskog registriranja pacijenta (snimanje i bilježenje podataka na CT-u ili MRI-u), time se definira referentni koordinatni sustav koji bi pomogao prilikom izvođenja zadatka (stereotaktička kirurgija). To dovodi do napredne vizualizacije i može biti korišteno kako bi se popravila slobodna navigacija, točnije pozicioniranje opreme ili upravljanje robotskih sustava.

Tijekom zadnjih desetljeća, deseci istraživačkih projekata fokusiraju se na probleme kirurgije mozga i kralježnice. Mogućnost da se operacije izvedu na manjoj skali pomoću robota čini mikrokirurgiju mogućom. Upotrebom slika dobivenih iz medicinskih uređaja (CT, MRI) može se dobiti povećana preciznost za navigaciju i pozicioniranje kirurškog alata u ciljnu točku. Također postoji i opcija upotrebe napredne digitalne obrade signala za kontroliranje ili bilježenje prostornih točaka koje su od interesa. To može biti korisno za simulacije operacija bez opasnosti. U konačnici robotizirana oprema može poboljšati ergonomske zahtjeve tijekom zahvata. Glavne prednosti robotskih neurokiruških sustava su:

- povećana preciznost
- velika kontrola kvalitete
- stabilnost i robusnost
- standardizacija, planiranje i reprodukcija operacija
- ušteda vremena (nakon upoznavanja sustava)
- korištenje MIK tehnika (npr. u kirurgiji lubanje)

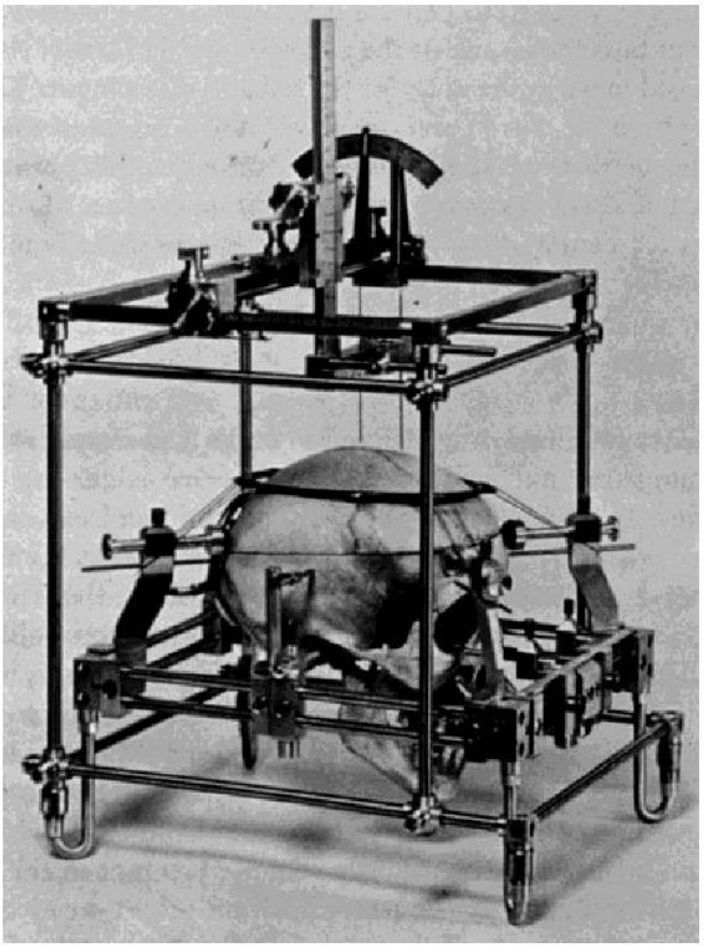
U slučaju neurokirugije postoji velika potreba za visokom preciznošću i kirurzi tradicionalno koriste optičke leće i specijalne alate kako bi poboljšali svoje sposobnosti. RIK pruža razne mogućnosti za poboljšanjem i povećanjem ljudske spretnosti. RA – MIK (Robotski asistirana – minimalno invazivna kirurgija) obećava značajne rezultate u slučaju zahvata na mozgu iz dva glavna razloga. Prvi, lubanja pruža kruti okvir što ju čini jednostavnijom za registraciju i povezivanje stvarnih obilježja na glavi s onima dobivenim skeniranjem prije operacije (osnova učinkovite vizijski upravljane kirurgije). Drugo, kompaktnost glave dopušta minimalne pomake mekog tkiva tijekom zahvata što za rezultat ima veću iskoristivost predoperativnog planiranja.

Robotski sustavi u neurokirurgiji dijele se na dvije glavne skupine: **stereotaktički okviri** i **stereotaktički robotski sustavi.**

3.1. Stereotaktički okviri

Stereotaksija (grč. stereos – trodimenzinalni, lat. tactus – dodirivati) je neurokiruški zahvat pri kojem se upotrebljava poseban uređaj za ciljano navođenje posebnih instrumenata (sonde, hvataljki i sl.) kroz malen otvor u lubanji u željena područja mozga i kralježnične moždine [4].

Ruski anatomičar Zernov izveo je prvu stereotaktičku operaciju 1889. s ciljem da izmjeri i mapira ljudski cerebralni giri. Spiegel i Wycis izveli su prvi stereotaktički zahvat modernog doba 1947. godine. Od tada je predstavljen veliki broj raznovrsnih mehanizama za stereotaktičku neurokirurgiju.



Slika 2 – Stereotaktički okvir razvijen od strane Spiegel i Wycis

Postoje tri osnovna kinematička postava za ručne stereotaktičke okvire: **translacijski, lučno – središnji, burr – hole postolja.** Svaka od skupina bit će objašnjena na temelju jednog primjera iz grupe. Većina ručnih stereotaktičkih okvira napravljena je od metala kojega je moguće sterilizirati kako bi se izbjegla upotreba korištenja sterilnih pokrivača u toku operacije. Ručni okviri mogu se lako prebacivati između raznih operacijskih sala, prostorni zahtjevi su im vrlo mali, stoga su vrlo prihvaćeni među neurokirurzima. Registracija između pacijenta i koordinatnog sustava vizijskog sustava radi se preko lokalizacijske jedinice (sustava markera).

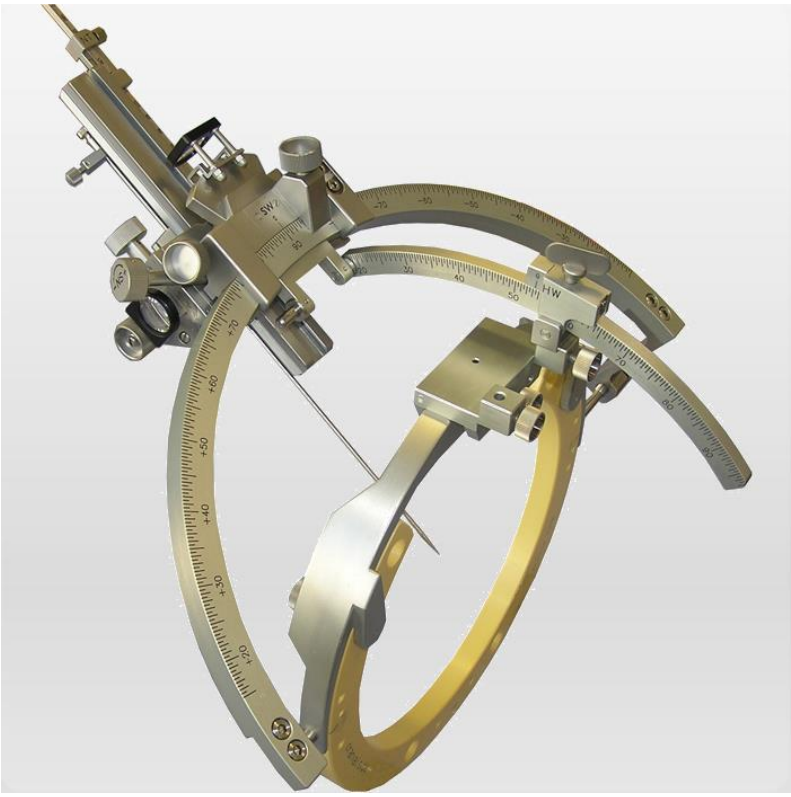
Tablica 1 daje pregled u nekolicinu ručnih sustava s datumima predstavljanja i njihovim kinematičkim lancem. Broj uređaja koji postoje puno je veći od prikazanoga u tablici.

Tablica 1 – Pregled ručnih stereotaktičkih uređaja podijeljenih prema kinematičkom principu s datumom predstavljanja

Ime uređaja	Kinematika	Godina predstavljanja
Spiegel & Wycis	Translacijski	1947.
Riechert - Mundinger	Translacijski	1955.
Talairach	Translacijski	1958.
Leksell	Središte luka	1949.
Cosman – Robert - Wells	Središte luka	1991.
Zamoranow – Duchovny	Središte luka	1994.
Austin & Lee	Burr hole postolje	1956.
Pelorus	Burr hole postolje	1985.
Nexframe	Burr hole postolje	2004.

3.1.1. Translacijski : Riechert – Mundinger sustav

Translacijski sustavi bili su prvi uređaji korišteni za stereotaktičke zahvate (npr. Spiegel i Wycis). Njihovo ime potječe od ideje da je svaka ciljna točka definirana s tri međusobno okomite translacijske koordinate. Jedno od ograničenja u starijim uređajima bilo je vođenje samo jedna trajektorija do ciljne točke. Ukoliko bi ta trajektorija vodila kroz vitalna područja mozga, operacija se ne bi mogla biti izvršiti. Iz tog razloga su u novije translacijske sustave dodane i osi rotacije. RM sustav koristi kružnu baznu jedinicu koja je učvršćena za pacijentovu glavu sa šiljcima s oštrim vrhovima. Četiri pločasta markera mogu biti postavljena u svhu obsvljsnjs registracije na prstenu putem CT ili MRI snimaka. RMov prsten s instrumentom fiksiran je u tri točke, što sustavu omogućuje veću stabilnost i točnost. Matematika uključena u planiranje ciljne točke nešto je kompleksnija nego kod lučnih sustava, zbog spregnutosti rotacijskih i translacijskih osi [5].

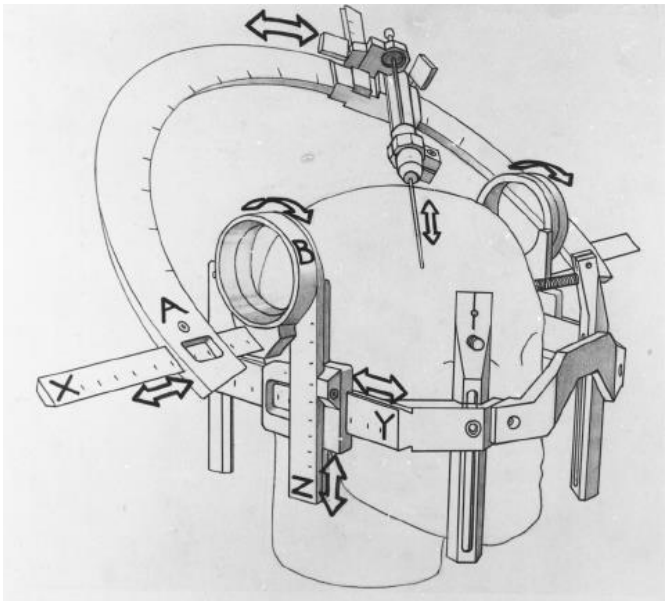


Slika 3 – Riechert – Mundinger sustav

3.1.2. Središte luka : Leksell okvir

Lars Leksell bio je švedski liječnik i profesor neurokirurgije te izumitelj radiokirurgije. Uređaj je predstavio 1949. godine kao prvi lučno – kvadratični stereotaktički sustav. On se sastoji od dva glavna dijela, kružnog luka na kojemu se nalazi pomični alat te dijela koji se fiksira na lubanju pacijenta. Kada se okvir fiksira, ovisno o ciljnoj točki, pomicanje instrumenta na kružnom luku kao rezultat uvijek daje orijentaciju k središtu centra sfere. Time se omogućuje lako namještanje okvira prema zahtjevima zahvata. Translacijski i rotacijski dio međusobno su odvojivi.

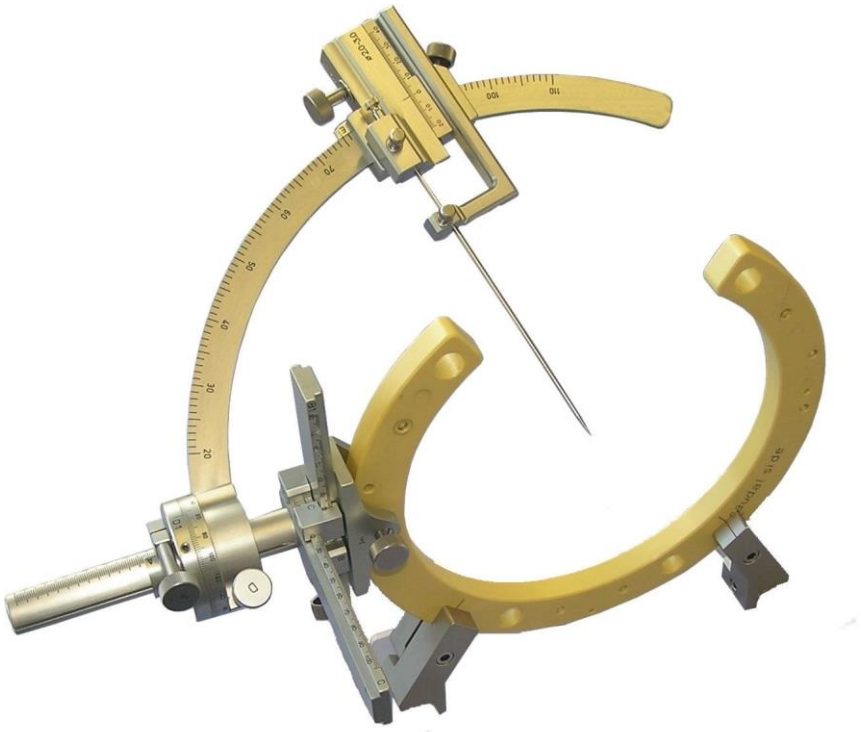
Slika 4 – Bazni dio Leksell okvira s referentnim koordinatnim sustavom (lijevo) i pomični luk s vodilicom (desno)



Slika 5 – Shema gibanja segmenata Leksell okvira

3.1.3. ZD okvir

Zamorano – Duchovny (ZD), predstavljen 1994. godine, je kao i Leksell, središte luka sustav. Kružni bazni prsten fiksiran je za pacijentovu glavu s četiri šiljka s oštrim vrhovima. Kod okvira se koristi otvoreni serijski kinematički lanac s translacijskom i dvije rotacijske osi. Tri translacijske osi u bazi sustava su međusobno okomite. One se koriste za pozicioniranje sonde u Kartezijevom koordinatnom sustavu. Kirurški postupak zasniva se na fiksiranju okvira na glavu pacijenta, potom se snima 3D slika i planira trajektorija u kirurškom programu za planiranje. S obzirom da translacijske i rotacijske osi nisu međusobno spregnute, kirurg može lakše namjestiti rotacijsku os bez potrebe ponovnog podešavanja translacijskih osi.



Slika 6 – ZD okvir

3.1.4. Burr hole postolje: Nexframe sustav

Iako su ovi sustavi generalno imali nedostatak u vidu preciznosti, trenutno proživljavaju renesansu. To uglavnom proizlazi iz integracije sustava navigacije u neurokirurgiji što značajno poboljšava preciznost. Trenutno je veliko pitanje je li preciznost burr hole uređaja na istom nivou sustava baziranih na okvirima. U sustavima burr hole postolja, bazna jedinica uređaja direktno se vijcima fiksira za lubanju. Lokacija burr holea bira se tako da je ciljna točka okomita na površinu lubanje. Odabir te lokacije od presudne je važnosti s obzirom na to da burr hole uređaji mogu kompenzirati samo male pogreške u pomaku. Nexframe (Medtronic Inc., SAD) predstavlja uobičajeno korišteni burr hole sustav. Sastoji se od dva dijela: Nexframe baze, koja je fiksirana za pacijentovu glavu s dva vijka i Nexframe tornja koji se pričvršćuje za baznu jedinicu (Slika 7).

Slika 7 – Nexframe stereotaktički burr hole sustav

3.2. Stereotaktički robotski sustavi

Kasnih 1980 – ih razvijeno je nekoliko neurokirurških sustava, što čini neurokirurgiju jednim od najranijih područja primjene robotske kirurgije. Stereotaktička neurokirurgija pruža idealne uvjete za primjenu robotskih pomagala iz sljedećih razloga:

- Ograničeno kretanje ciljne točke: Robotska kirurgija najuspješnija je kada su pokreti ciljne točke ograničeni. U usporedbi s mekim tkivom (npr. jetra), lubanja ograničava pokrete ciljne točke tijekom operacije.
- Kruto fiksiranje lubanje: U stereotaktičkoj neurokirurgiji, bazna jedinica fiksirana je za glavu pacijenta. Ovaj kruti spoj može se iskoristiti za utvrđivanje transformacije između koordinatnih sustava robota i pacijenta
- Potrebe za velikom preciznošću: Mali pomaci od planirane trajektorije mogu dovesti do značajnog oštećenja mozga, posebice u područjima mozga zaduženim za govor. Stoga, velika preciznost je od presudne važnosti u stereotaktičkoj neurokirurgiji.

Mnogi robotski sustavi nastaju na sveučilištima i razvojnim ustanovama, a rijetki od njih postanu komercijalno primjenjivi. Neki od najuspješnijih komercijalnih robotskih sustava su Neuromate, Neuroarm, ROSA, Pathfinder itd.

3.2.1. Neuromate

Predstavljen 1987. godine, Neuromate je jedan od prvih robotskih sustava u kirurgiji uopće. Trenutni vlasnik ovog sustava je tvrtka Renishaw (Renishaw PLC, UK). Neuromate je sustav specifične kinematike jer osim robota s pet stupnjeva slobode gibanja, ima produženu ruku na koju se fiksira lubanja pacijenta. Sterilnost sustava postiže se omatanjem robotske ruke. Dva su načina upravljanja, na bazi okvira i bez okvira. Pri testiranju sustava autori su ustanovili prosječnu pogrešku od $0.86\,mm$ sa standardnim odstupanjem od $0.32\,mm$ za način rada s okvirom. Način rada bez okvira ra rezultat ima srednju pogrešku od $1.95\,mm$ sa standardnim odstupanjem od $0.44\,mm$. Mobilnost sustava je limitirana pošto ne uključuje mobilno postolje. U metodi bez okvira, sustav za praćenje potreban je kako bi se odredila transformacija između koordinatnog sustava robota i pacijenta.

Slika 8 – Neuromate

3.2.2. PathFinder

PathFinder (Prosurgic, UK) prvi puta se pojavio 2001. godine. Bazira se na industrijskom robotu koji je montiran na pomični stalak. Kao većina industrijskih robota, PathFinder ima 6 stupnjeva slobode gibanja, s držačem alata na izvršnom dijelu robota. U standardnom kirurškom postupku PathFindera izvršavaju se slijedeći koraci: fiksiraju se markeri za registraciju na glavu pacijenta. Oni se ili uvijaju u lubanju ili lijepe na kožu s ljepljivom trakom. Nakon toga se dobavlja 3D slika pacijenta s markerima. U idućem koraku kirurg planira ciljnu točku i trajektoriju robota s programom za kirurško planiranje na temelju slika dobivenih iz CTa ili MRIa. Robot se postavlja u blizinu pacijenta. Pacijentova glava je fiksirana za operacijski stol Mayfield hvataljkom. Optički sustav za praćenje određuje poziciju markera za registraciju iz raznih kutova. Na temelju tih informacija dobiva se prijela između stereotaktičkog koordinatnog sustava i koordinatnog sustava robota. Robot se

automatski pozicionira prema preoperativnom planu. Na držač alata moguće je pričvrstiti razne alate te robot sam izvršava kirurški zahvat. U testiranju s fantomom dobiveni su rezultati preciznosti: srednje pogreške od $2.7\,mm$, minimalne pogreške od $1.8\,mm$ i maksimalne pogreške od $3.2\,mm$. Prostorni zahtjevi u operacijskim salama značajno su manji od onih koje zahtjeva Neuromate sustav, ali i dalje daleko veći od potreba koje imaju ručni stereotaktički sustavi. Zahvaljujući prijenosnom stalku, PathFinder se lako može transportirati između operacijskih sala. Za upravljanje sustavom potreban je optički sustav za praćenje.

Slika 9 – PathFinder

3.2.3. ROSA

Sustav proizašao iz francuske tvrtke Medtech, trenutno korišten na područjima Europe, Sjeverne Amerike, Azije i Australije. ROSA je revolutni robot sa šest stupnjeva slobode gibanja, postavljen na prijenosni stalak u kojemu se nalazi kontrolna jedinica. Sustav je predviđen samo za pozicioniranje sonde u prostoru i nije za interakciju s pacijentom. Umetanje alata ili druge interakcije pripadaju pod odgovornost kirurga. Kirurški postupak s ROSA sustavom je slijedeći: uzima se 3D slika pacijenta i trajektorija sonde planira se na temelju dobivenih podataka. Lubanja pacijenta fiksirana je za operacijski stol pomoću hvataljke. Robot se potom pozicionira u neposrednu blizinu pacijenta i fiksira se za operacijski stol. Kočnice pomične jedinice se uključuju kako bi se minimizirali mogući neželjeni pomaci. Registracija se radi putem laserskog skenera koji je učvršćen za izvršni dio robota. On utvrđuje udaljenost od pacijentove glave i time se dobiva 3D prikaz površine. Dobiveni 3D podaci registriraju se u preoperativnu sliku pacijentove glave. Za potrebe registracije nisu potrebni markeri za registraciju ni stereotaktički okviri. Robot je opremljen senzorom sile i momenata na izvršnom dijelu. Moguć je takozvani haptički način rada u kojemu kirurg rukom pomiče izvršni dio robota. U tom slučaju robot se giba samo po unaprijed definiranoj trajektoriji. U 2014. godini objavljeni su rezultati kliničkih ispitivanja i ustanovila se srednja pogreška od $1.22\,mm$ sa standardnim odstupanjem od $0.73\,mm$. Prostorni zahtjevi manji su nego kod PathFindera što omogućava bolju mobilnost sustava. Zahvaljujući načinu rada s haptičkom suradnjom dobilo se na dodatnoj sigurnosti. Sustav zahtjeva laserski skener kako bi se obavila registracija što može pridodati dodatnu pogrešku na ukupnu točnost.

Slika 10 – ROSA

3.2.4. RONNA

RONNA robotski stereotaktički sustav za neuronavigaciju razvijen je u Hrvatskoj od strane tima s Fakulteta strojarstva i brodogradnje s Katedre za projektiranje izradbenih i montažnih sustava uz suradnju liječnika iz Kliničke bolnice Dubrava. RONNA sustav sastoji se od programa za planiranje i navigaciju, dva robota (glavni i pomoćni), stereovizijskog sustava, lokalizacijske jedinice (sustava markera), alata za rad (griperi, vodilice, itd.) i napredne programske podrške za donošenje upravljačkih odluka iz nejasne okoline. Roboti koji se koriste su standardni industrijski što omogućuje robusnost i manju cijenu razvijenog sustava. Glavni robot (Kuka KR6) je revolutni robot s maksimalnim dosegom 900mm, nosivosti 6kg i ponovljivosti ±0.03 mm. Taj robot se koristi kao alternativa klasičnoj stereotaktičkoj metodi zbog svoje čvrstoće i krutosti. Pomoćni robot (UR5) ima nisku apsolutnu točnost i ponovljivost od ±0.1 mm s maksimalnim dosegom od 850 mm i nosivosti od 5 kg. Kuka robot koristi se za preciznu navigaciju instrumenata prema ciljnoj točki operacije koju izvodi pomoćni robot ili kirurg. Na slijedećoj slici (Slika 11) može se vidjeti RONNA sustav s pripadajućom stereovizijskom kamerom (Polaris) i ekranima za nadzor rada sustava.

Slika 11 – RONNA

Lokalizacijska jedinica sastoji se od markera koji se postavljaju na pacijentovu glavu pomoću posebnog titanskog vijka čime se dobiva kruti spoj s lubanjom. Sustav se sastoji od 3 markera vidljivih na CTu ili MRIu. Nakon snimanja na CTu i dobivanja 3D modela pacijenta, pomoću spomenutih markera kreira se koordinatni sustav pacijenta.

Prvi korak u radu sustava je priprema pacijenta gdje se na pacijenta kruto postavlja sustav za lokalizaciju. Potom pacijent ide na snimanje na CT ili MRI kako bi se dobila 3D slika na temelju koje se u posebno razvijenom programu za kirurško planiranje točno planiraju trajektorije za operaciju. Program za planiranje operacije potom šalje informacije robotima, jer da bi sustav radio robot mora identificirati poziciju i orijentaciju markera koji definiraju koordinatni sustav pacijenta. Taj postupak dijeli se u dva koraka: približna i konačna lokalizacija. Prvi korak radi modul RONNAvision. RONNAvision modul standardna je stereovizijska kamera postavljena na pomoćnog robota. Robot lokalizira markere pričvršćene na pacijentovoj glavi i šalje točne informacije o koordinatama glavnom robotu. U drugom koraku lokalizacije glavni robot koristi specijalno razvijeni stereovizijski sustav. Stereovizijski modul (RONNAstereo) sastoji se od dvije kamere visoke rezolucije s veoma uskim vidnim poljem. Kamere su međusobno okomite i sjecište optičkih osi im je u fokusu obje kamere. Idući korak je izvršavanje planirane trajektorije. RONNA sustav može raditi u dva načina: interaktivnom ili automatskom, što ovisi o tipu operacije i izboru kirurga. U oba slučaja glava pacijenta fiksirana je na Mayfield hvataljki. U slučaju interaktivnog rada, robot samo pomaže kirurgu. U tom slučaju pomoćni robot ima zadatak samo približno odrediti lokaciju markera na pacijentu putem svog stereovizijskog sustava. To pomaže glavnom robotu da nađe markere na pacijentovoj glavi. Nakon te lokalizacije i kalibracije od glavnog robota mogu se odvojiti kamere i spojiti vodilica. Robot dolazi na planiranu trajektoriju, te je na kirurgu da stavi sondu ili drugi alat u intrakranijalni prostor pacijenta. U slučaju automatskog rada, obje robotske ruke su aktivne te umjesto kirurga, pomoćni robot obavlja kretnje stavljanja alata u lubanju pacijenta. RONNA sustav obavio je prvo kliničko ispitivanje 2016. godine u Kliničkoj bolnici Dubrava (Zagreb) gdje je obavljena biopsija tumora u interaktivnom načinu rada. U ovome trenutku klinička ispitivanja su u tijeku te nisu poznati rezultati o ukupnoj točnosti primjene sustava [6].

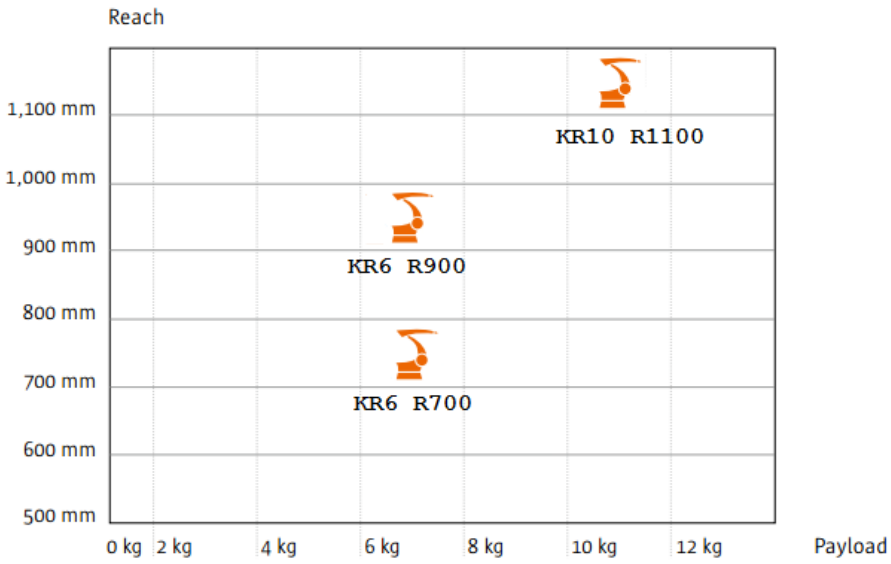
Slika 12 - Prvo kliničko ispitivanje RONNA sustava

4. KUKA AGILUS ROBOTI

Kuka Agilus je serija opsežnih malih robota koje odlikuje velika brzina, točnost i ponovljivost. U ovaj skup pripadaju roboti s pet ili šest stupnjeva slobode gibanja. Performanse koje odlikuju Agilus robote su slijedeće:

- Velika brzina – ističu se u poslovima rukovanja, *pick and place* radnjama
- Integrirani sustav napajanja – EtherCAT/EtherNet sabirnice, ventili za komprimirani zrak, jednostavna integracija prihvatnice
- Široki raspon položaja montiranja – zahvaljujući kočnicama u svim osima roboti ispunjavaju zahtjeve svih pozicija (Agilus s pet stupnjeva slobode gibanja može se montirati na pod i strop, a Agilus sa šest stupnjeva dodatno i na zid)
- Mali troškovi održavanja – nisu potrebna podmazivanja nakon određenog vremena upotrebe što omogućuje kontinuiran rad
- Jednostavan kontroler za upravljanje

Roboti koji će se koristiti u ispitivanju ovoga rada su Kuka Agilus KR6 R700, KR6 R900 i KR10 R1100. Odabrani Agilus roboti na slici su prikazani prema dosegu i nosivosti (Slika 13) [7].



Slika 13 – Podjela odabranih Agilus robota prema dosegu i nosivosti

4.1. Tehničke karakteristike

U tablici (Tablica 2) su prikazane najbitnije karakteristike za sva tri navedena robota.

Tablica 2 – Karakteristike Agilus KR 6 R700, KR 6 R900 i KR 10 R1100 robota

		KR 6 R700	KR 6 R900	KR 10 R1100
Broj stupnjeva slobode gibanja		6		
Masa		$50kg$	$52kg$	$54kg$
Maksimalna nosivost		$6kg$	$6kg$	$10kg$
Ponovljivost		$\pm 0{,}03mm$		
Radni doseg		$706{,}7mm$	$901mm$	$1101mm$
Opseg kretanja zglobova	$A1$	$+ / -170°$		
	$A2$	$+45° / -190°$		
	$A3$	$+156° / -120°$		
	$A4$	$+ / -185°$		
	$A5$	$+ / -120°$		
	$A6$	$+ / -350°$		
Upravljačka jedinica		$KR\ C4\ compact$		
Temperaturno radno područje		$5 - 45℃$		
Privjesak za učenje		$Kuka\ smartPAD$		

Slika 14 – Kuka Agilus KR6 R700 (lijevo), KR6 R900 (sredina), KR10 R1100 (desno)

Slika 15 – Upravljačka jedinica KR C4 compact i smartPAD privjesak za učenje

4.2. Primjena

Agilus roboti se zbog dobrih tehničkih karakteristika primjenjuju u mnogim industrijskim granama, na poslovima kao što su:

- Rukovanje, utovar / istovar
- Sastavljanje
- Pakiranje
- Paletiziranje
- Mjerenje, testiranje, kontroliranje
- Premazivanje
- Umetanje i montaža
- Opsluživanje alatnih strojeva

4.3. Kinematički model

Kinematički model robota izveden je u Denavit – Hartenberg (DH) notaciji. Slijedi prikaz koordinatnih osi svakog zgloba (Slika 16), te DH parametri za sva tri Agilus robota.

Tablica 3 – DH parametri za Kuka KR6 R700

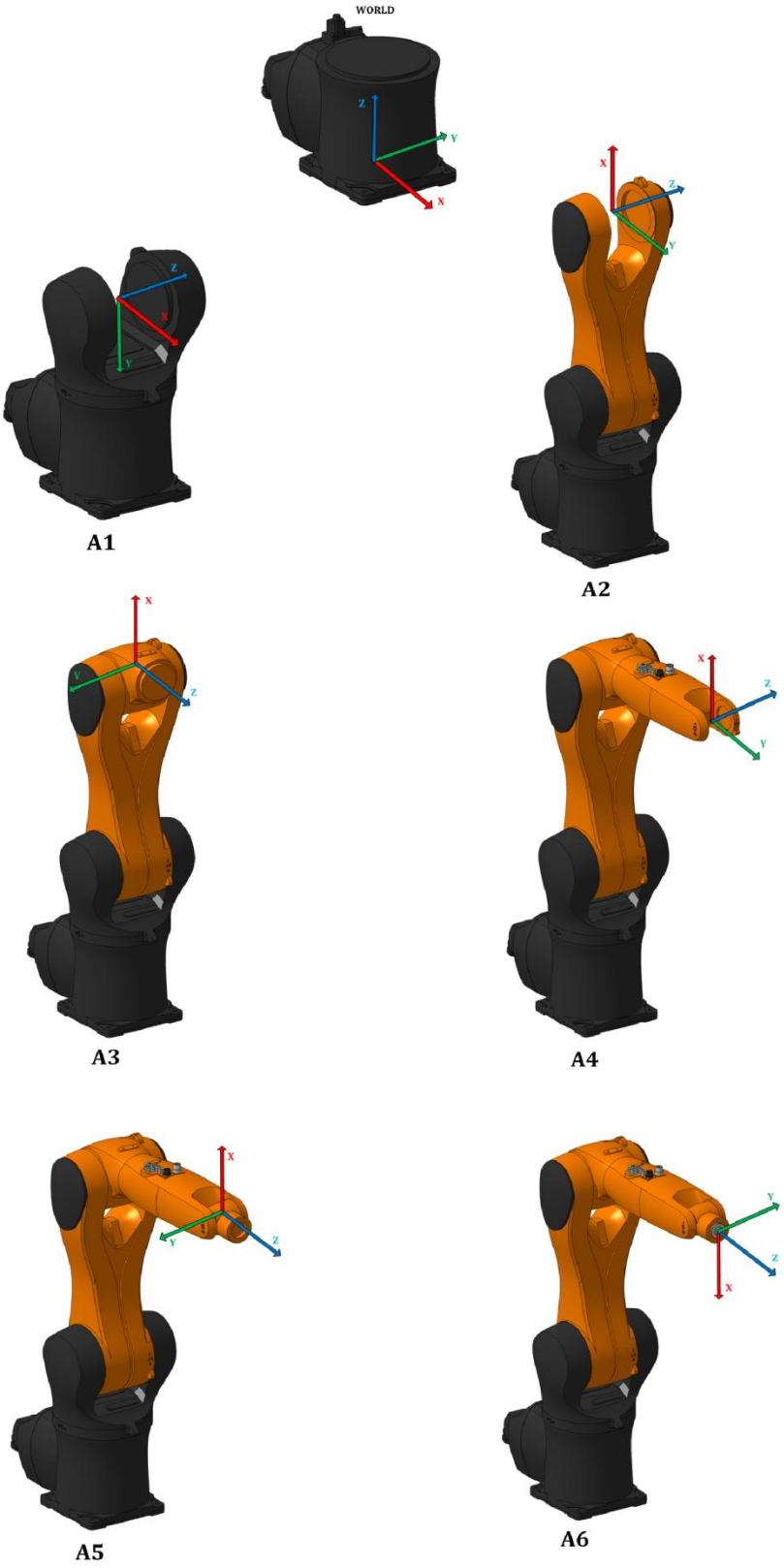
Zglobovi robota	Parametri			
	$\theta_i\ [deg]$	$d_i\ [mm]$	$a_i\ [mm]$	$\alpha_i\ [deg]$
A_1	θ_1	400	25	90
A_2	θ_2	0	315	0
A_3	$\theta_3\ -90$	0	35	90
A_4	$\theta_4\ +\ 90$	365	0	-90
A_5	$\theta_5\ -\ 90$	0	0	90
A_6	θ_6	80	0	180

Tablica 4 - DH parametri za Kuka KR6 R900

Zglobovi robota	Parametri			
	$\theta_i\ [deg]$	$d_i\ [mm]$	$a_i\ [mm]$	$\alpha_i\ [deg]$
A_1	θ_1	400	25	90
A_2	θ_2	0	455	0
A_3	$\theta_3 - 90$	0	35	90
A_4	$\theta_4 + 90$	420	0	-90
A_5	$\theta_5 - 90$	0	0	90
A_6	θ_6	80	0	180

Tablica 5 - DH parametri za Kuka KR10 R1100

Zglobovi robota	Parametri			
	$\theta_i\ [deg]$	$d_i\ [mm]$	$a_i\ [mm]$	$\alpha_i\ [deg]$
A_1	θ_1	400	25	90
A_2	θ_2	0	560	0
A_3	$\theta_3 - 90$	0	35	90
A_4	$\theta_4 + 90$	515	0	-90
A_5	$\theta_5 - 90$	0	0	90
A_6	θ_6	80	0	180



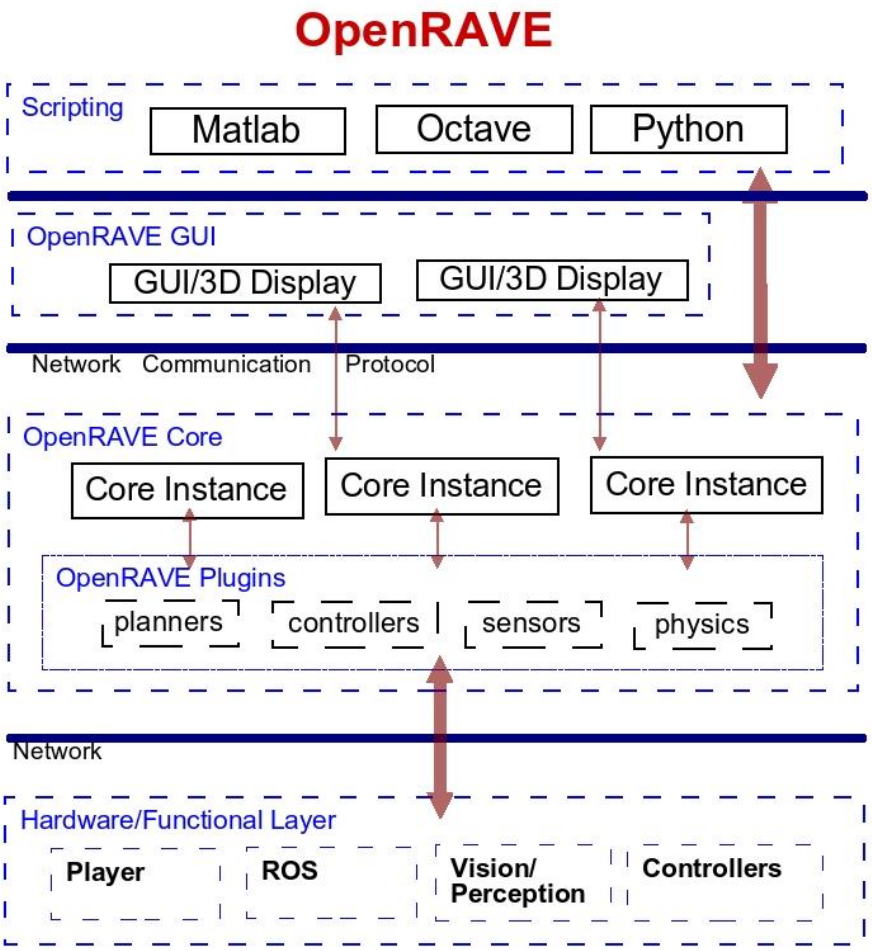
Slika 16 – Prikaz koordinatnih osi svakog zgloba Agilus robota

5. RAZVOJ SIMULACIJSKOG MODELA ROBOTA

Prilikom projektiranja automatskih sustava, posebice složenijih kao što je robotska asistencija u operacijskoj sali, potrebno je mogućnost pogreške svesti na minimum. Iz tih razloga došlo je do razvoja softvera kojima je omogućena 3D vizualizacija, planiranje te kontroliranje robotskog sustava u virtualnom okruženju. Jedan od takvih programa je OpenRAVE, otvorena platforma u koju je moguće implementirati proizvoljne CAD modele. U interakciji s Python programskim jezikom omogućeno je stvaranje virtualne okoline u kojoj će željeni roboti rješavati zadane kompleksne probleme. Kako bi se CAD modeli mogli implementirati i koristiti u OpenRAVE računalnom paketu, potrebno je podesiti njihove koordinatne sustave pomoću CATIA programskog paketa. Detaljno kreiranje manipulatora nalazi se u prilogu (Prilog I) radi preglednosti rada. Slijedi kratak opis OpenRAVE i Python programskih paketa.

5.1. OpenRAVE

OpenRAVE (**Open R**obotics **A**utomation **V**irtual **E**nvironment) programski paket kreiran je od strane Rosen Diankova u Institutu za robotiku na Carnegie Mellon sveučilištu. Pruža mogućnost kreiranja virtualnog okruženja u kojemu je moguće razvijati, implementirati i testirati algoritme za proizvoljne automatske sustave. Fokus programa je simulacija i analiza kinematičkih i geometrijskih lanaca vezanih uz planiranje pokreta. Najbitnija tehnologija koju OpenRAVE pruža je alat pod nazivom IKFast**,** a odnosi se na računanje kinematike robota. Za razliku od sličnih alata, IKFast je u mogućnosti analitički riješiti kinematičke jednadžbe kompleksnih kinematičkih lanaca, a konačan rezultat su izuzetno stabilna rješenja koja se mogu izvoditi unutar $5\mu s$ na današnjim procesorima. U konfiguraciji manipulatora i okoline OpenRAVE koristi .xml format zapisa [8].



Slika 17 – OpenRAVE arhitektura rada

5.2. Python

Python je interpreterski, interaktivni, objektno orijentirani programski jezik, kojega je 1990. godine razvio Guido van Rossum. Python nije donio revolucionarne značajke u programiranju, već je na optimalan način ujedinio ideje i načela rada drugih programskih jezika. Nalazi se između tradicionalnih, skriptnih i sistemskih jezika. Python nudi jednostavnost i lako korištenje (poput Matlab-a), uz napredne programske alate [9].

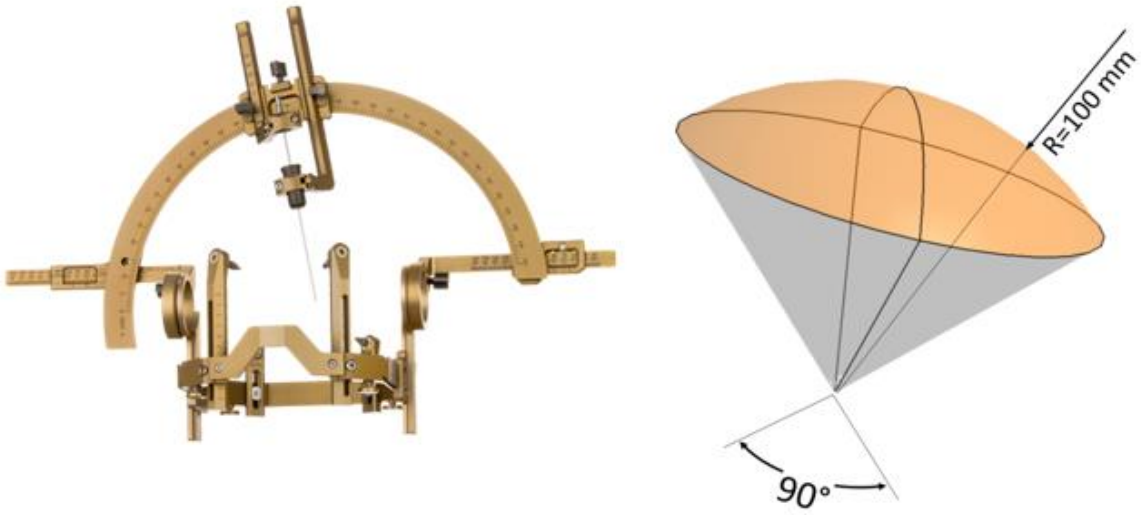
6. EKSPERIMENTALNO ISPITIVANJE ROBOTSKOG RADNOG PROSTORA

Jedan od važnijih zahtjeva kod stereotaktičke neurokirurgije je precizna prostorna navigacija instrumenata zbog čega se koriste stereotaktički okviri te u novije vrijeme stereotaktički robotski sustavi. Spomenuto je kako RONNA sustav za stereotaktičku navigaciju upotrebljava industrijske robote i pritom ne koristi dodatne produžene ruke na kojima pacijent mora biti fiksiran. Nakon lokalizacije i određivanja svih potrebnih koordinatnih sustava, potrebno je pronaći optimalnu poziciju između robota i pacijenta.

Prva stavka je definiranje radnog prostora. Nakon toga nužno je izračunati transformacijsku matricu pozicije vrha alata u prirubnicu robota. Zbog specifičnog oblika radnog prostora, u prostornom navođenju robota koristi se sferni koordinatni sustav kojega je potrebno transformirati u Kartezijev koordinatni sustav. Orijentacija alata mora biti unaprijed definirana stoga je potrebno za svaku točku u radnom prostoru izračunati koordinatne sustave orijentacije. Trajektorije u radnom području odabiru se prema kriteriju najmanje utrošenog vremena gibanja. Prije gibanja u svaku točku vrše se ispitivanja mogućnosti dolaska u točku i linearnog pomaka u intrakranijalni prostor. Ukoliko se pokaže nemogućim to ostvariti, vrši se rotacija oko osi alata. Navedeni postupci ponavljaju se za sve točke u predodređenom radnom prostoru. Time se kreira algoritam zasnovan na eksperimentalnoj validaciji mogućnosti prostornog pozicioniranja robotske ruke i testiranju mogućnosti u intrakranijalnom radnom području kojega opisuju standardni stereotaktički okviri.

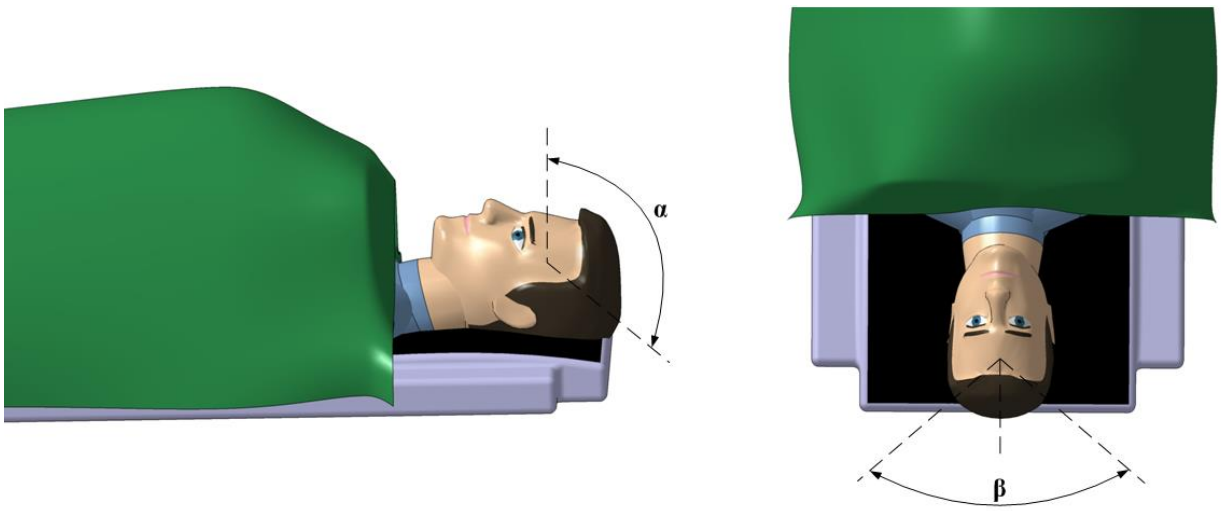
6.1. Radni prostor

Radni prostor robota tijekom operacije ograničen je na područje oko lubanje pacijenta te se orijentira prema referentnom koordinatnom sustavu u samoj lubanji (točka operacije). Prostor u kojem robot djeluje geometrijski je određen prema standardnim stereotaktičkim okvirima i zatvara prostor kuglinog isječka (Slika 18).



Slika 18 – Kružni prsten Leksell okvir zatvara radni prostor standardan za stereotaktičku kirurgiju **[10]**

Osim radijusa, varijabilni faktori u definiranju radnog prostora su alfa i beta kutovi (Slika 19). Navedena tri parametra određuju oblik kuglinog isječka, tj. gibanje robota.

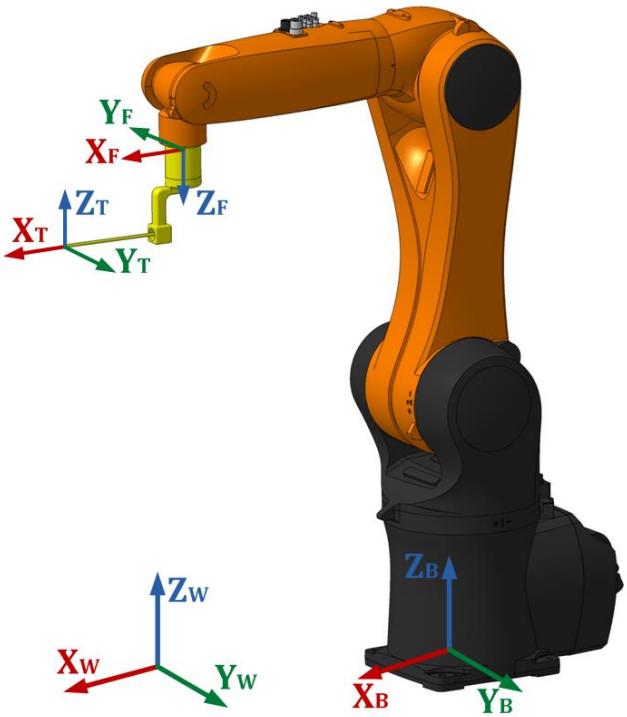


Slika 19 – Definiranje granica alfa (vertikalnog) i beta (horizontalnog) kuta

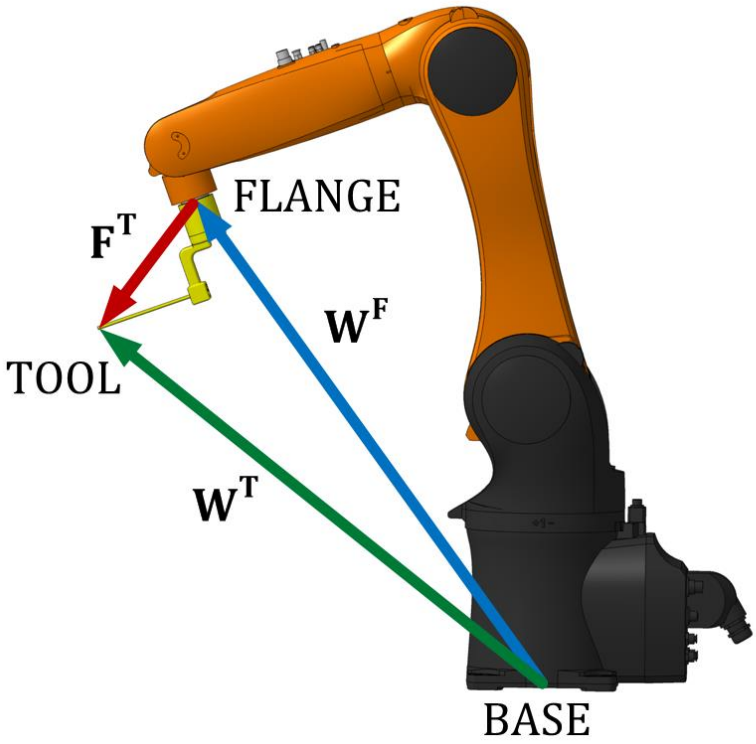
6.2. Transformacija sustava vrha alata u prirubnicu robota

Robot u tijeku pripreme i operacije može promijeniti nekoliko različitih vrsta alata (stereovizijski, vodilica, bušilica). S obzirom na visoke zahtjeve preciznosti nužna je točna manipulacija alatom. S obzirom da je moguće kontrolirati samo izvršni član robota (prirubnica), a ne i sam alat, potrebno je napraviti transformacijske matrice iz TCP-a (Tool Center Point) u samu prirubnicu.

Koordinatni sustavi koji su potrebni za računanje transformacija su sustavi baze, prirubnice i TCP-a (Slika 20).



Slika 20 – Prikaz koordinatnih sustava korištenih za transformaciju iz TCP-a u prirubnicu robota (World, Base, Flange, Tool)



Slika 21 - Vektori vrha alata i prirubnice iz baze robota

Jednadžba koja proizlazi iz prethodne slike (Slika 21) je sljedeća:

$$W^F \times F^T = W^T \quad \textbf{(1)}$$

Iz jednadžbe (1) potrebno je izlučiti transformaciju prihvatnice u vrh alata, odnosno F^T. S obzirom da se radi o matricama, slijedeći pravila o množenju i dijeljenju, potrebno je jednadžbu (1) pomnožiti s lijeve strane inverzom matrice W^F:

$$(W^F)^{-1} \backslash W^F \times F^T = W^T \quad \textbf{(2)}$$

Iz toga proizlazi:

$$F^T = (W^F)^{-1} \times W^T \quad \textbf{(3)}$$

Kada je izračunata matrica transformacija između prirubnice i vrha alata preostaje pozicije u koje se želi postaviti alat množiti s dobivenom matricom. Time će se dobiti pozicija u koju

treba poslati prirubnicu kako bi vrh alata bio na željenoj poziciji. Postupak je identičan prethodnom, samo se ovoga puta traži položaj prirubnice, odnosno W^F. Jednadžba (1) množi se s desne strane inverzom matrice F^T:

$$W^F \times F^T = W^T \backslash (F^T)^{-1} \quad \textbf{(4)}$$

Kao rezultat dobije se položaj prirubnice:

$$W^F = W^T \times (F^T)^{-1} \quad \textbf{(5)}$$

Izračunate transformacije bit će prikazane na konkretnom primjeru. Sve vrijednosti pozicija u matricama (četvrti stupac) prikazane su u metrima:

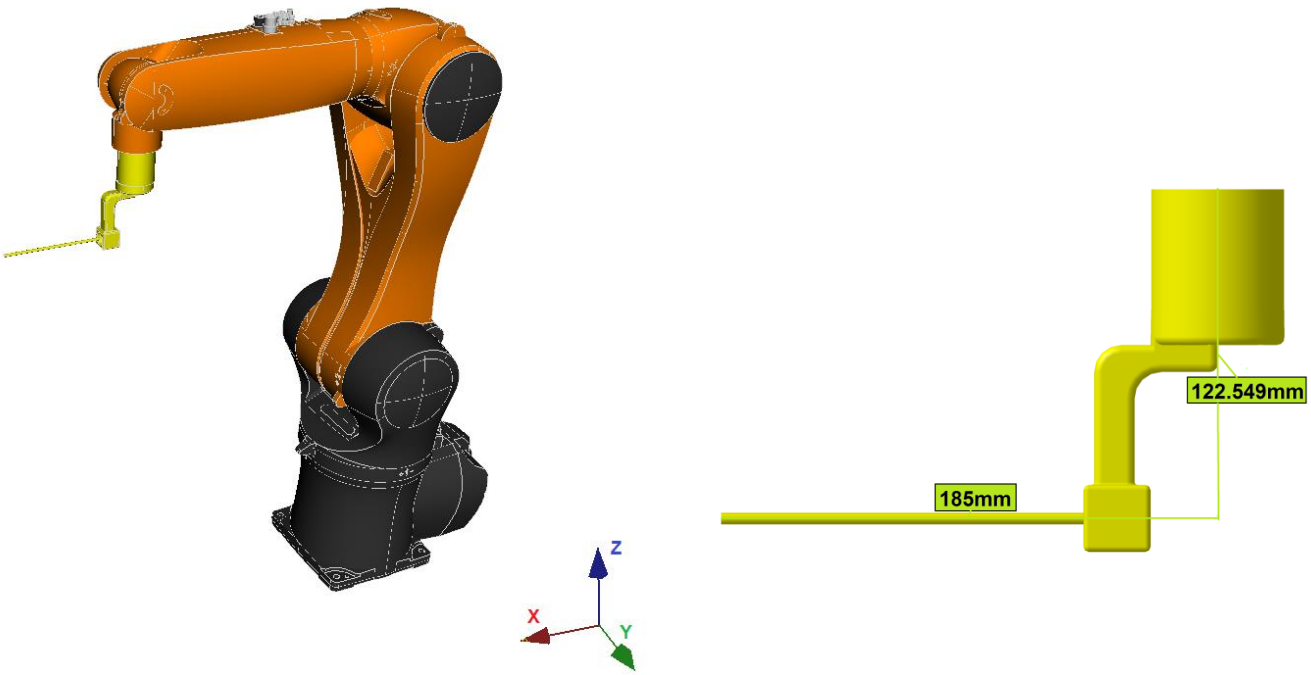
$$W^F = \begin{bmatrix} 1 & 0 & 0 & 0.445 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0.81 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(6)}$$

$$W^T = \begin{bmatrix} 1 & 0 & 0 & 0.631 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.687 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(7)}$$

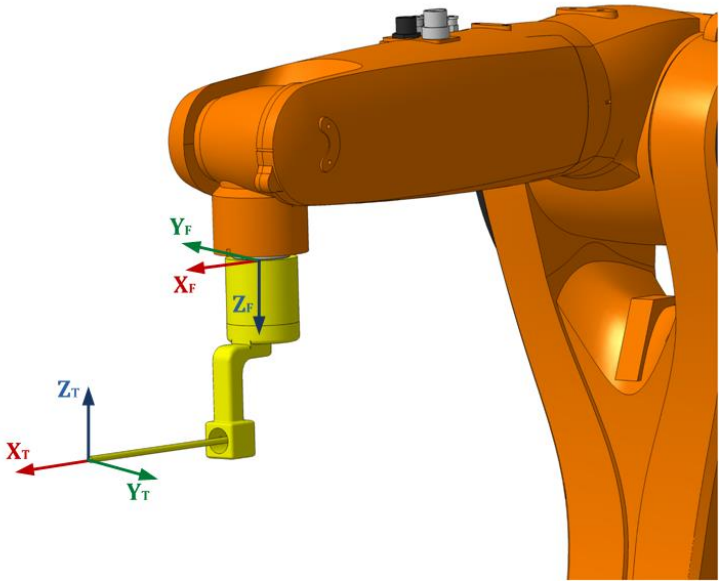
$$(W^F)^{-1} = \begin{bmatrix} 1 & 0 & 0 & -0.445 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0.81 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(8)}$$

$$F^T = (W^F)^{-1} \times W^T = \begin{bmatrix} 1 & 0 & 0 & 0.185 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0.1225 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(9)}$$

Prema podacima iz Catia-e i OpenRAVE-a (Slika 22) mogu se vidjeti dimenzije i orijentacija vodilice u odnosu na globalni koordinatni sustav OpenRAVE-a te potom usporediti s dobivenom matricom transformacije alata F^T.



Slika 22 – Orijentacija i dimenzije vodilice u usporedbi s $\boldsymbol{F^T}$



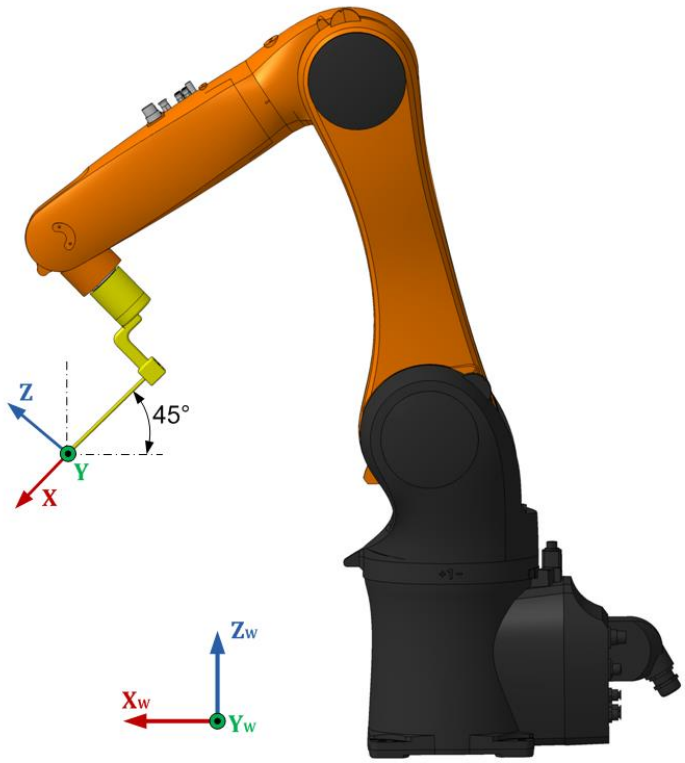
Slika 23 – Koordinatni sustav alata ($\boldsymbol{W^T}$) u odnosu na koordinatni sustav prirubnice robota ($\boldsymbol{W^F}$)

Nakon prikaza transformacija vrha alata u prirubnicu robota, sljedeći korak je pokazati dobivanje koordinata prihvatnice postavljanjem vrha alata u konkretnu točku (*Point*).

$$Point = W^T = \begin{bmatrix} 0.7071 & 0 & 0.7071 & 0.47071 \\ 0 & 1 & 0 & 0 \\ -0.7071 & 0 & 0.7071 & 0.37071 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(10)}$$

$$(F^T)^{-1} = \begin{bmatrix} 1 & 0 & 0 & -0.185 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0.1225 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(11)}$$

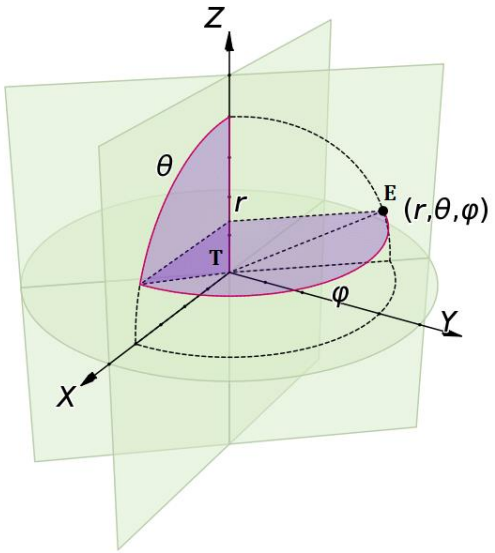
$$W^F = W^T \times (F^T)^{-1} = \begin{bmatrix} 0.7071 & 0 & -0.7071 & 0.4258 \\ 0 & -1 & 0 & 0 \\ -0.7071 & 0 & -0.7071 & 0.5889 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad \textbf{(12)}$$



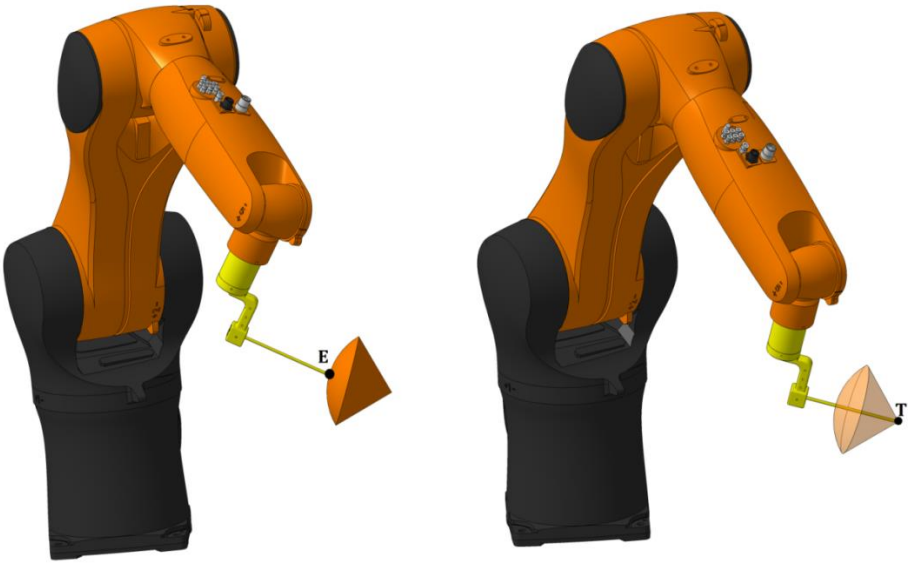
Slika 24 – Pozicioniranje vrha alata u *Point* točku

6.3. Transformacija sfernog u Kartezijev koordinatni sustav

Lubanja svakog čovjeka specifičnog je i nepravilnog oblika, ali okviri koji se koriste u stereotaksiji, kao što se može vidjeti na slici (Slika 18), zatvaraju poznati geometrijski oblik, oblik kuglinog isječka. Iz tog razloga zadavanje putanja robotu najintuitivnije i najjednostavnije je ukoliko se koristi sferni koordinatni sustav te se nakon toga izračuna transformacija za x, y i z točke Kartezijevog sustava. Sferni sustav definiran je pomoću dva kuta (azimut, elevacija) i radijusom sfere.



Slika 25 – Sferni koordinatni sustav (**E** – točka na sferi (**Entry**), **P** – ishodište sfere (**Target**)) **[11]**



Slika 26 – Robot s vrhom alata u ishodištu i na plaštu kuglinog isječka (**E** –**Entry**, **T** –**Target**)

Slijedi prikaz algoritma korištenog za transformaciju sfernog u Kartezijev koordinatni sustav:

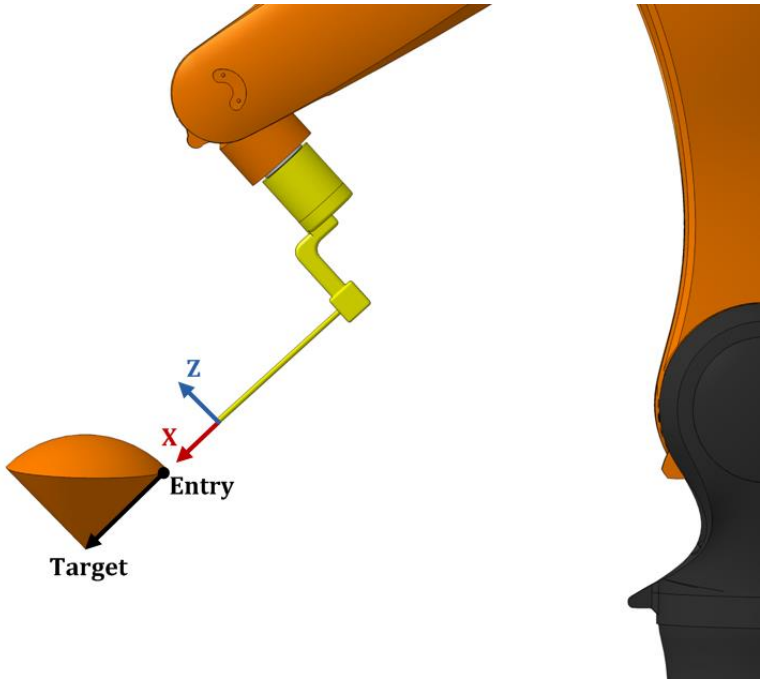
Algoritam 1 – Transformacija koordinatnih sustava

Algoritam: Transformacija sfernog u Kartezijev koordinatni sustav

```
1:  Function sph2cart (azimuth, elevation, radius)
2:  Input:azimuth ⟶ azimut
3:        elevation ⟶ elevacija
5:        radius ⟶ radijus sfere
7:  Output:X,Y,Z ⟶ pozicije u Kartezijevom koordinatnom
8:                  sustavu
9:
10: r_cos ⟵ radius x cosinus(elevation)
10: X ⟵ r_cos x cosinus(azimuth)
11: Y ⟵ r_cos x sinus(azimuth)
12: Z ⟵ r x sinus(elevation)
13: Return X,Y,Z
```

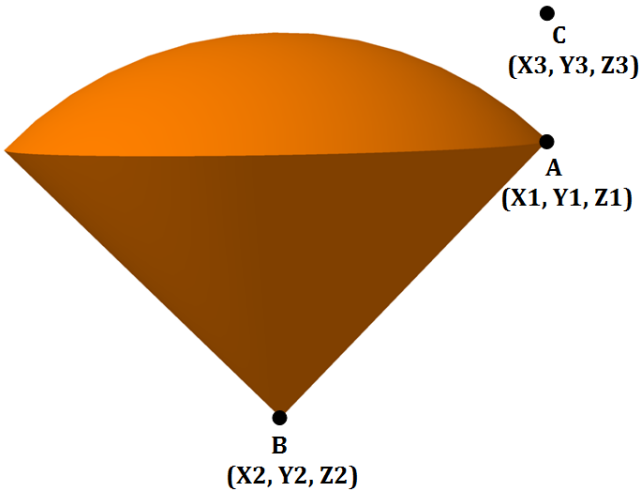
6.4. Orijentacijska matrica alata

Nakon što su pronađene x, y, z koordinate točke u koju je potrebno smjestiti vrh alata, nužno je napraviti odgovorajuću orijentaciju alata. Orijentacija alata mora biti unaprijed definirana prema ishodištu kuglinog isječka tako da je X os TCP-a na Entry → Target pravcu (Slika 27). Kako bi se to postiglo, potrebno je kreirati koordinatni sustav za svaku točku na plaštu kuglinog isječka. Najlakši način za kreiranje sustava je odabir triju točaka kroz koje se formiraju vektori s pripadajućim smjerovima. Matematičkim operacijama manipuliranja vektorima kao što su norma i vektorski produkt dolazi se do ciljanog koordinatnog sustava. U sljedećih nekoliko slika i pseudokodom bit će objašnjen postupak kreiranja koordinatnog sustava.



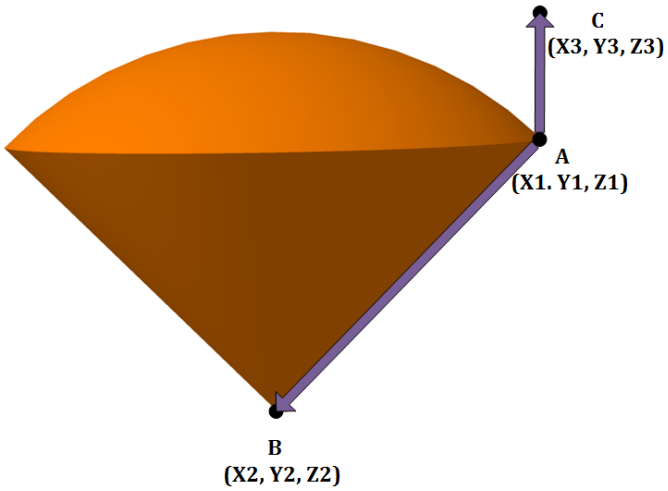
Slika 27 – Orijentacija alata prema ishodištu kuglinog isječka

Točke se odabiru na način da prva točka bude u ishodištu kuglinog isječka (Target), što znači da područje na kojem će se izvršiti zahvat mora biti poznato. Druga točka odabire se na plaštu isječka (Entry). Treća točka jednaka je prethodnoj po x i y koordinatama te je translatirana za proizvoljan iznos u pozitivnu stranu z osi. Sljedeća slika prikazuje navedene tri točke (Slika 28).



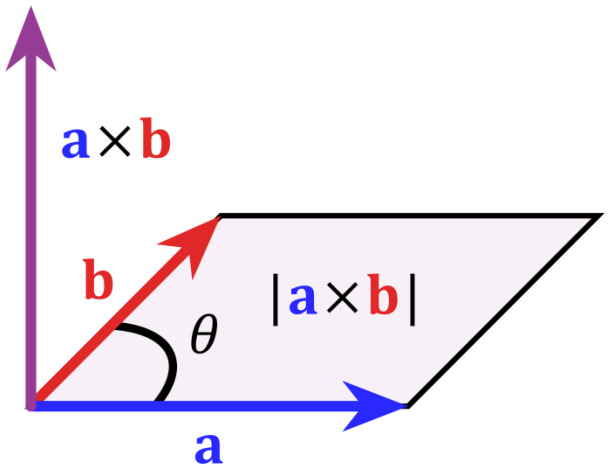
Slika 28 – Odabir tri točke za kreiranje koordinatnog sustava

Vektori s poznatim smjerovima mogu se generirati nakon što su odabrane sve tri točke. Smjer prvog vektora je od točke A (Entry) do B (Target) (prema Algoritam 2: AB_vektor), a smjer drugoga od točke A do C (prema Algoritam 2: AC_vektor) (Slika 29).



Slika 29 – Vektori za kreiranje koordinatnog sustava

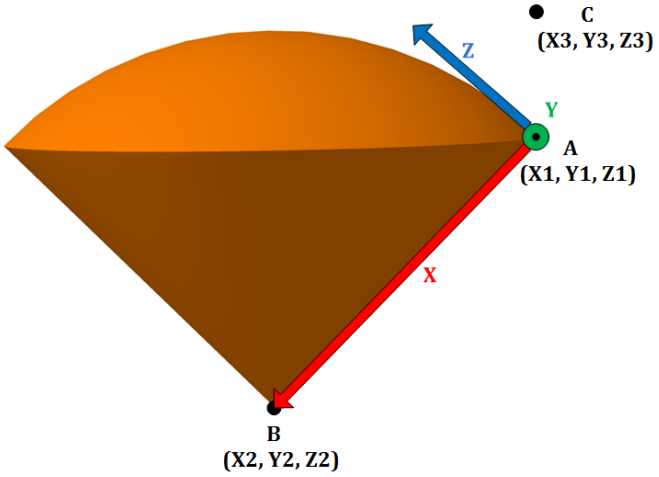
Dva vektora tvore ravninu, a treći vektor okomit na tu ravninu dobije se pomoću vektorskog produkta (prema Algoritam 2: Y).



Slika 30 – Vektorski produkt dva vektora

S novim vektorom te postojećim koji ide od točke A do točke B, kreira se nova ravnina. Postupak vektorskog produkta potrebno je ponoviti na novu ravninu (prema Algoritam 2: Z).

Rezultat je koordinatni sustav sa stalnom orijentacijom prema ishodištu (prema Algoritam 2: R_M)(Slika 31).



Slika 31 – Koordinatni sustav s orijentacijom prema ishodištu

Algoritam 2 – Orijentacijska matrica alata

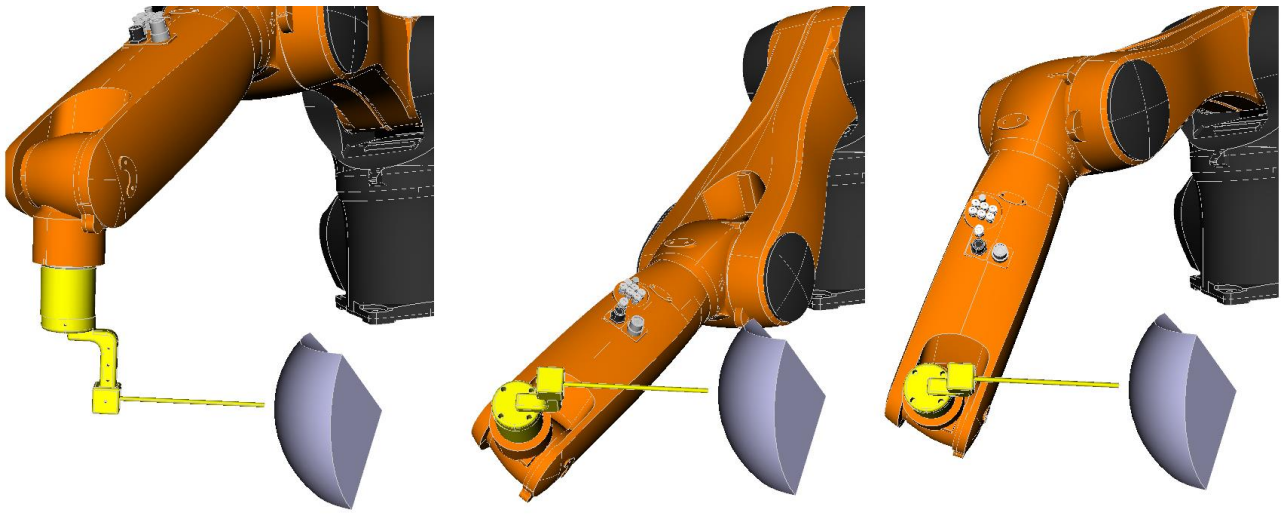
Algoritam: Kreiranje orijentacijske matrice alata

```
1:  Function orientation_matrix (kuka, tool_coord, sfera_pos)
2:  Input:kuka ⟶ inicijalizacija robota
3:        tool_coord ⟶ transformacijska matrica (4 × 4) vrha
4:                     alata
5:        sfera_pos ⟶ transformacijska matrica (4 × 4)
6:                              ishodišta sfere
7:  Output:R_M ⟶ transformacijska matrica (4 × 4) alata
8:
9:  A ⟵ tool_coord(0:3,3) //X,Y,Z pozicija alata
10: B ⟵ sfera_pos(0:3,3) //X,Y,Z pozicija centra sfere
11: C ⟵ copy(A)
12: C(2) ⟵ C(2) + 0.2 //C = A s razlikom Z koordinate
13: AB_vektor ⟵ B - A
14: AC_vektor ⟵ C - A
15: AB ⟵ AB_vektor / norma(AB_vektor)
16: AC ⟵ AC_vektor / norma(AC_vektor)
17: Y⟵ vektorski_produkt(AC/AB)/norma(vektorski_produkt(AC/AB))
18: //vektorski produkt na vektore AC I AB (Y os)
19: Z ⟵ vektorski_produkt(AB,B) // Z os
20: R_M ⟵ eye(4) //kreiranje jedinične matrice 4 × 4
21: R_M(0:3,0) ⟵ AB //prvi stupac matrice je vektor AB
22: R_M(0:3,1) ⟵ Y //drugi stupac matrice je os Y
23: R_M(0:3,2) ⟵ Z //treći stupac matrice je os Z
24: R_M(0:3,3) ⟵ A //četvrti stupac matrice je vektor A
25:       (pozicija alata u prostoru)
25: Return R_M
```

6.5. Ispitivanje najkraće trajektorije i mogućnost linearnog pomaka

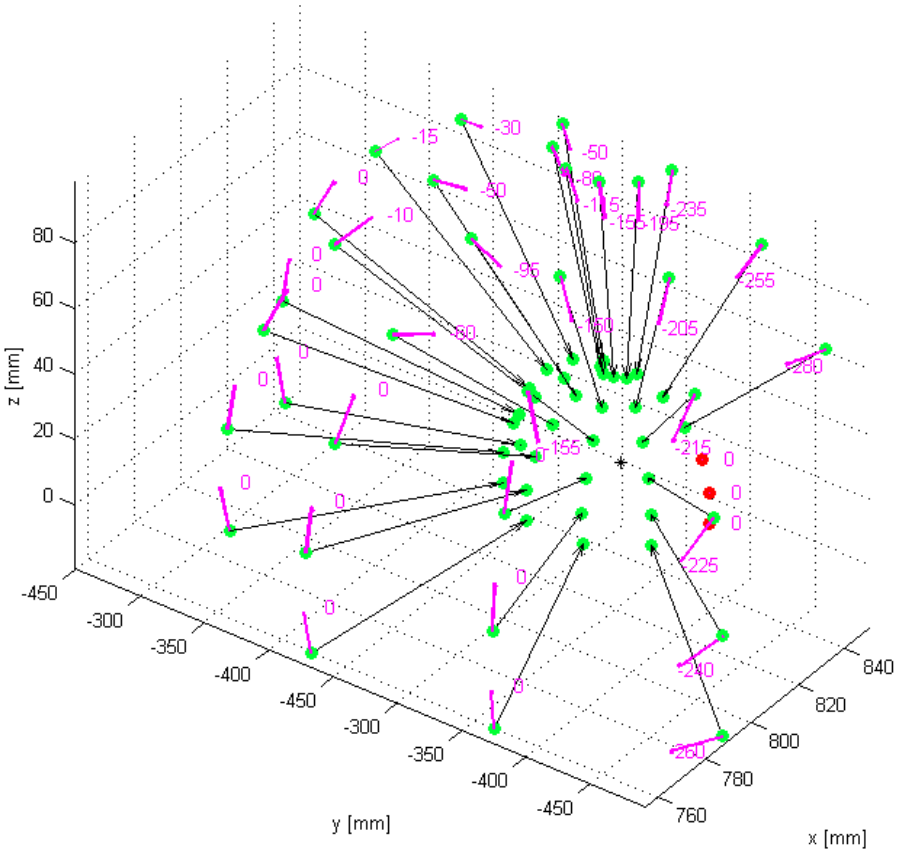
Nakon što se pronađu koordinate i izračuna orijentacijska matrica za svaku točku u koju robot treba doći, potrebno je i ispitati postoji li moguća trajektorija za dolazak u tu točku. Ovisno o ograničenjima zglobova robota, najčešće postoji nekoliko mogućih rješenja za odlazak u sljedeću točku. Rješenja koja OpenRave nudi su u obliku 1×6 vektora koji sadrže kutove rotacija svakog zgloba (prema Algoritam 3: solutions). Iz grupe mogućih rješenja, potrebno je odabrati najbolje koje zahtijeva najmanje utrošenog vremena. Nakon pronalaska najboljeg rješenja, slijedeći korak je ispitati može li robot u istoj konfiguraciji ostvariti linearni pomak prema ciljnoj (Target) točki ili je ciljna točka predaleko od robota ili bi robot došao u koliziju

sam sa sobom. Ukoliko se pokaže nemogućim ostvariti početnu (Entry) i ciljnu točku (Target) izvršava se rotacija oko osi alata. Kut rotacije oko osi alata (prema Algoritam 3: theta) varijabilan je faktor te se njegov korak može podešavati po želji. Drugi mogući ishodi ovog algoritma su: ako se niti uz promjenu kuta theta ne može ostvariti početna i ciljna točka (prema Algoritam 3: theta = theta_limit), uzima se trajektorija koja može dati bilo kakav rezultat. Također, ako ne postoji niti jedna moguća trajektorija za početnu točku, izlazi se iz petlje te se radi rotacija oko osi alata. U slučaju da niti tada nema mogućih rješenja (prema Algoritam 3: length(solutions) = 0) izlazi se iz petlje te se gibanje zapisuje kao neuspješno.



Slika 32 – Rotacija oko osi alata

Na sljedećoj slici može se vidjeti prikaz Entry ($100mm$) i Target ($30mm$) trajektorija s vektorom koji označava smjer z osi TCP-a (ljubičasta boja). Uz svaki vektor nalazi se ispis theta vrijednosti. Sve točke su obojane prema tome je li gibanje bilo uspješno (zeleno) ili neuspješno (crveno) (Slika 33).



Slika 33 – Entry/Target trajektorije s ispisom theta vrijednosti

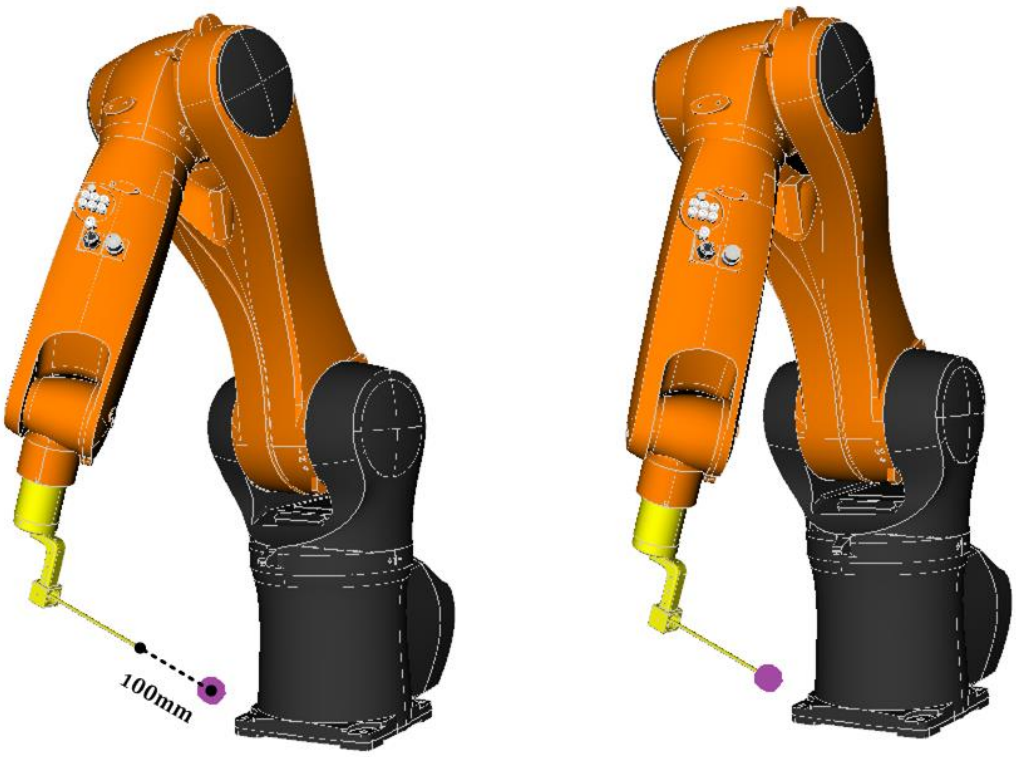
Algoritam 3 – Solutions

Algoritam: Traženje najkraće trajektorije gibanja robota između točaka

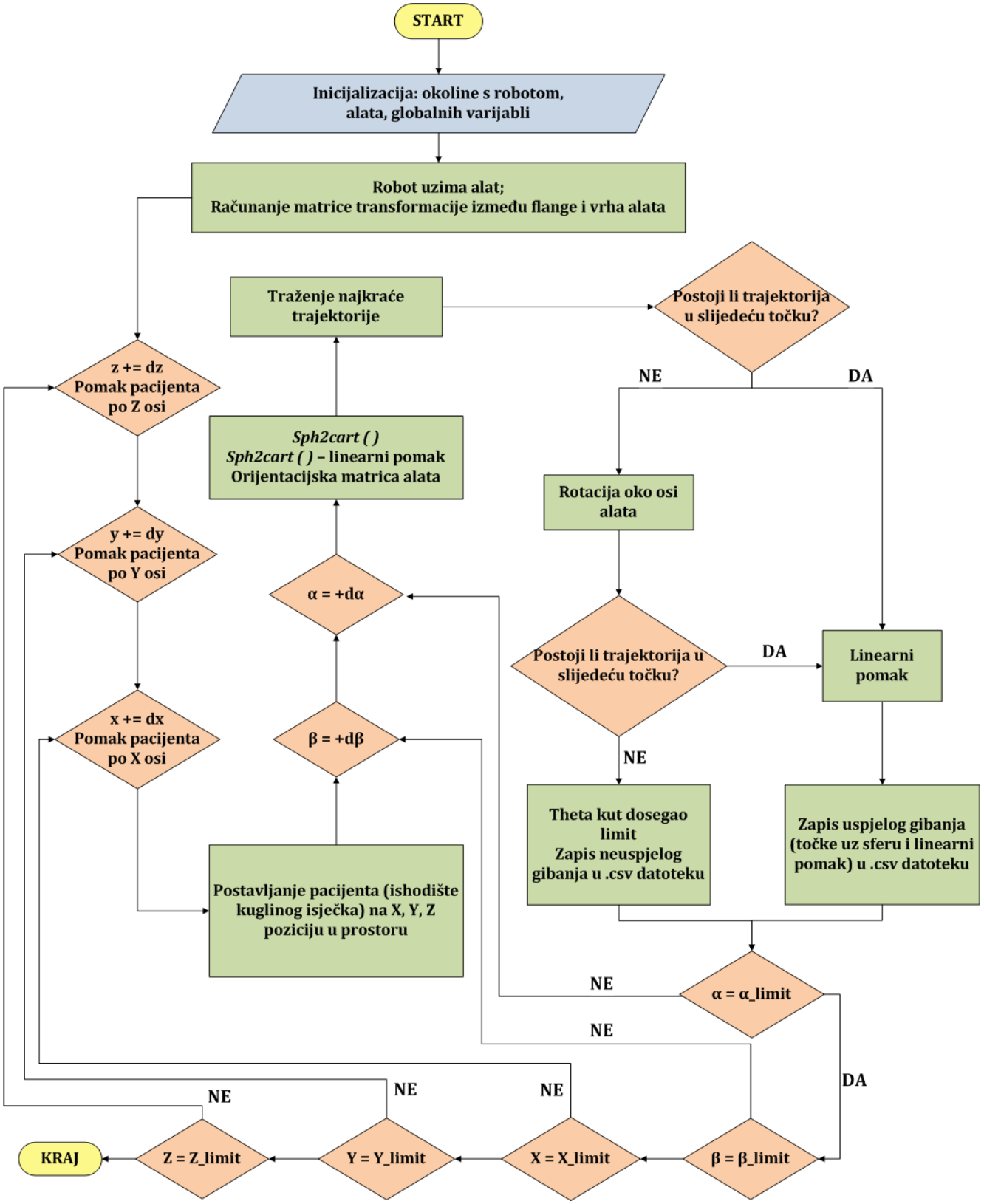
```
Function shortest_trajectory (solutions, theta, kukaruka,
      kuka,smjer, sphere_center, vodilica)
Input: solutions ⟶ n x 6 vektora mogućih trajektorija
       theta ⟶ kut rotacije oko osi alata
       kukaruka ⟶inicijalizacija manipulatora za gibanje
       kuka ⟶   inicijalizacija robota
       smjer ⟶ putanja linearnog gibanja
       sphere_center ⟶ centar sfere (X,Y,Z)
       vodilica ⟶ inicijalizacija vodilice
Output:limit kuta theta
       planning Error ⟶ ne postoji moguća trajektorija
       gibanje po najkraćoj trajektoriji
minimum ⟵ array([]) //prazan vektor
for x from 0 to length(solutions):
    trajektorija ⟵ (J1(x),J2(x),J3(x),J4(x),J5(x),J6(x)) //kutovi
                                rotacije svakog zgloba
    vrijeme ⟵ trajektorija.GetDuration() //racunanje
      vremena izvršenja svake trajektorije
    minimum ⟵ append(vrijeme) //zapis vremena na kraj
      vektora
end for
if theta ⟵ theta_limit: //theta kut dosegao limit
    print 'Limit kuta theta'
    Return True //ne postoji trajektorija niti uz
    rotaciju oko osi alata (nedohvatljiva točka)
end if
if length(solutions) ⟵ 0://nema moguće trajektorije
    print 'Planning Error'
    Return False //izlaz iz petlje, promjena theta kuta
end if
else:
    min ⟵ index(min(minimum))//lokacija minimuma
    cilj ⟵ (J1(min),J2(min),J3(min),J4(min),J5(min),J6(min))
    kukaruka.MoveHandStraight//pravocrtno gibanje
    if linear_move < goal_linear_move:
        Return False //nije ispunjeno linearno gibanje,
        izlaz iz petlje, promjena theta kuta
    else:
        kukaruka.MoveManipulator
        Return True//postoje trajektorije za Entry/Target
```

6.6. Linearni pomak

Linearni pomak izvršuje se tek nakon što se u algoritmu (Algoritam 3) ispitaju najkraće trajektorije gibanja i mogućnost samog linearnog pomaka. Razlog naknadnog izvršavanja linearnog pomaka je zapisivanje položaja vrha alata u .csv datoteku. Ukoliko linearni pomak nije do kraja izvršen, potrebno je to zabilježiti radi naknadnog crtanja grafičkih mapa uspješnosti svakog robota.



Slika 34 – Linearni pomak u intrakranijalnom prostoru pacijenta



Slika 35 – Dijagram toka programa za traženje optimalnog radnog područja robota

7. USPOREDBA REZULTATA

U ovom poglavlju prikazani su eksperimentalni rezultati različitih konfiguracija ulaznih parametara sustava.

Inicijalna testiranja (devet simulacija) provedena su za sve tri robotske konfiguracije s jednakim parametrima:

- $x_{start},\ x_{end},\ x_{division}$
- $y_{start},\ y_{end},\ y_{division}$
- $z_{start},\ z_{end},\ z_{division}$
- $linearni\ pomak$
- $theta_{division}$

Alfa i beta kutovi su varijabilni faktori koji će se simulirati u tri slučaja:

a) $alfa = -10°\ do\ 80°\ i\ beta = 170°$

b) $alfa = -10°\ do\ 85°\ i\ beta = 180°$

c) $alfa = -20°\ do\ 88°\ i\ beta = 200$

7.1. Ograničenja zglobova robota

Konfiguracija zglobova Agilus robota definirana je od strane proizvođača (Tablica 2). Poznato je da robot može zauzeti nekoliko konfiguracija u istoj točki u prostoru. Tijekom operacije nisu dozvoljene sve konfiguracije, stoga ih je potrebno ograničiti. U sljedećoj tablici je primjer mogućih konfiguracija robota u istoj točki prirubnice (Tablica 6).

Položaj točke u kojoj se nalazi prirubnica (Slika 36):

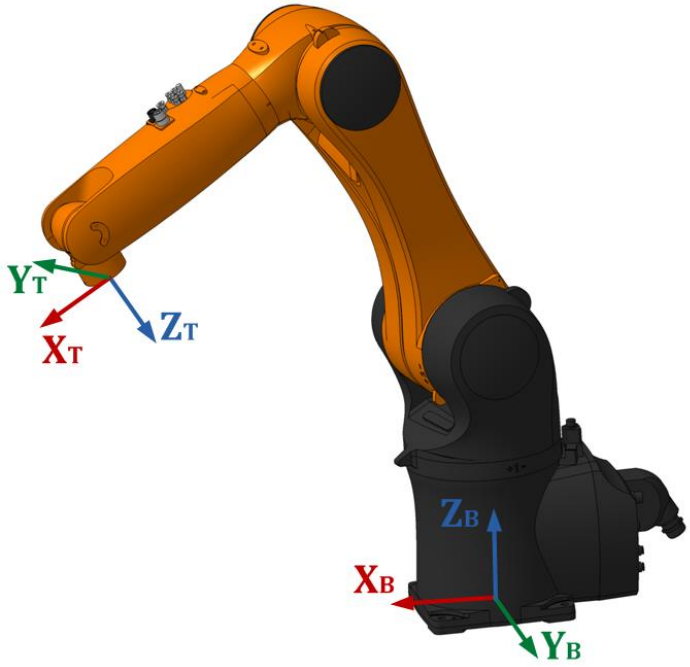
- $x = 550mm$
- $y = 150mm$
- $z = 550mm$

Radi boljeg razumijevanja tablice slijede objašnjenja:

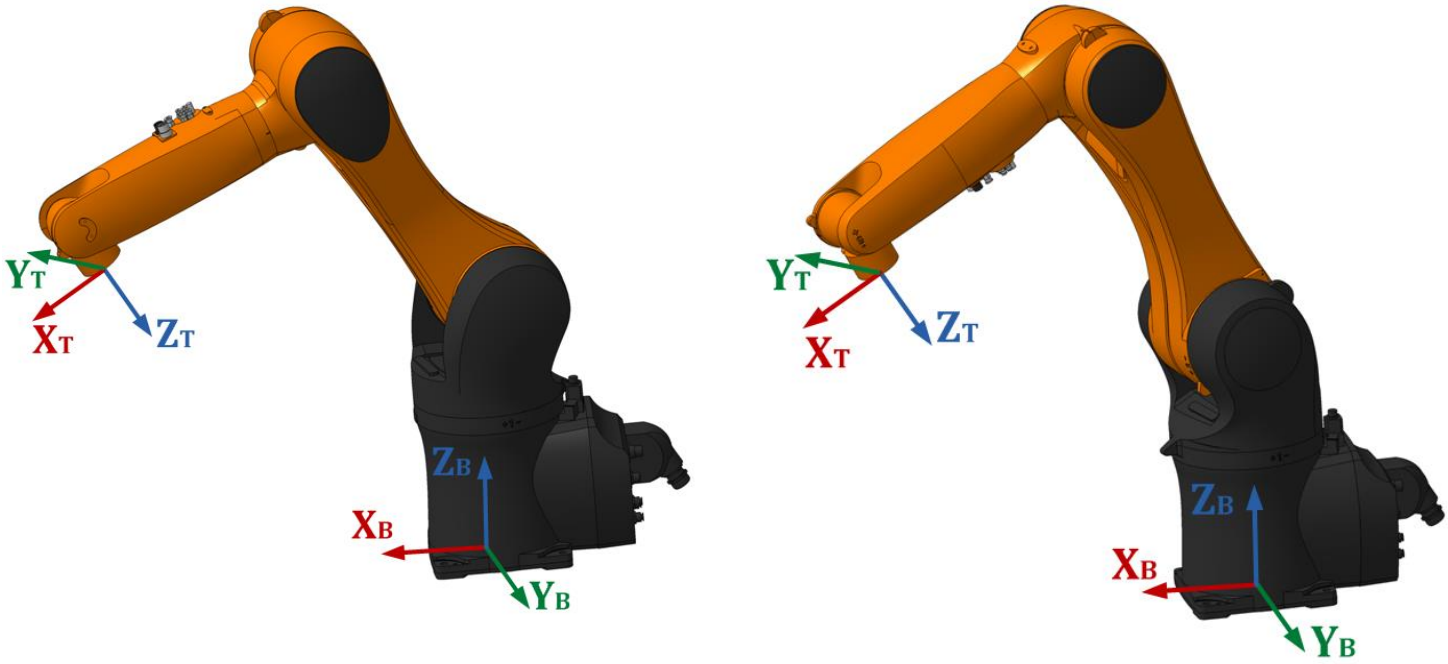
- F/R (Front/Rear) – prema kojoj strani u odnosu na x os je okrenut prvi zglob robota (A_1)
- U/D (Elbow Up/Down) – način na koji je okrenuta druga os robota (A_2)
- F/N (Flip/Non-Flip) – rotacija četvrte osi robota (A_4)

Tablica 6 – Konfiguracija zglobova s dolaskom prirubnice u istu točku

Broj	F/R	U/D	F/N	A_1	A_2	A_3	A_4	A_5	A_6
1	✓	✓	×	$-14.9°$	$-63°$	$93.9°$	$177.5°$	$-89°$	$-4.2°$
2	✓	✓	✓	$-14.9°$	$-63°$	$93.9°$	$-2.5°$	$89°$	$175.8°$
3	×	×	✓	$165.1°$	$-122.8°$	$-75.2°$	$177.5°$	$101.9°$	$175.2°$
4	×	×	×	$165.1°$	$-122.8°$	$-75.2°$	$-2.5°$	$-101.9°$	$-4.8°$
5	✓	✓	×	$-14.9°$	$-63°$	$93.9°$	$-182.5°$	$-89°$	$-4.2°$
6	×	×	✓	$165.1°$	$-122.8°$	$-75.2°$	$-182.5°$	$101.9°$	$175.2°$
7	✓	✓	✓	$-14.9°$	$-63°$	$93.9°$	$-2.5°$	$89°$	$-184.2°$
8	×	×	✓	$165.1°$	$-122.8°$	$-75.2°$	$177.5°$	$101.9°$	$-184.8°$
9	×	×	✓	$165.1°$	$-122.8°$	$-75.2°$	$-182.5°$	$101.9°$	$-184.8°$

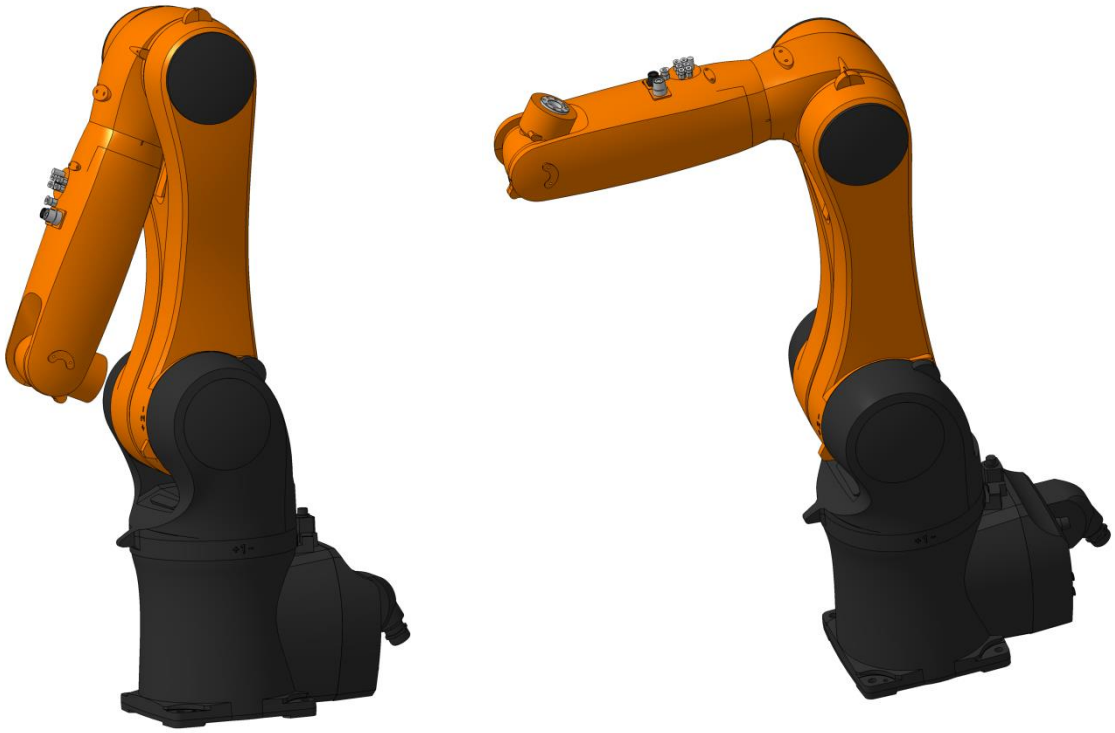


Slika 36 – Primjer konfiguracije osi robota (prema **Tablica 6**: Broj 1)



Slika 37 – Rear, Elbow Down i Flip konfiguracija robota (prema **Tablica 6**: Broj 3 i 2)

Tijekom operacije na robota su spojeni brojni kabeli i sterilni pokrivač. Potrebno je spriječiti konfiguracije s prethodne slike (Slika 37) i stoga su stavljena ograničenja na osi A_1 i A_4. Osi A_3 i A_5 također se ograničavaju kako bi robot izbjegavao dolaske u limite i koliziju sa samim sobom.



Slika 38 – Robot u limitu osi $\boldsymbol{A_3}$ i $\boldsymbol{A_5}$

Tablica svih ograničenja korištenih u simulacijama (Tablica 7):

Tablica 7 – Ograničenja na zglobove robota

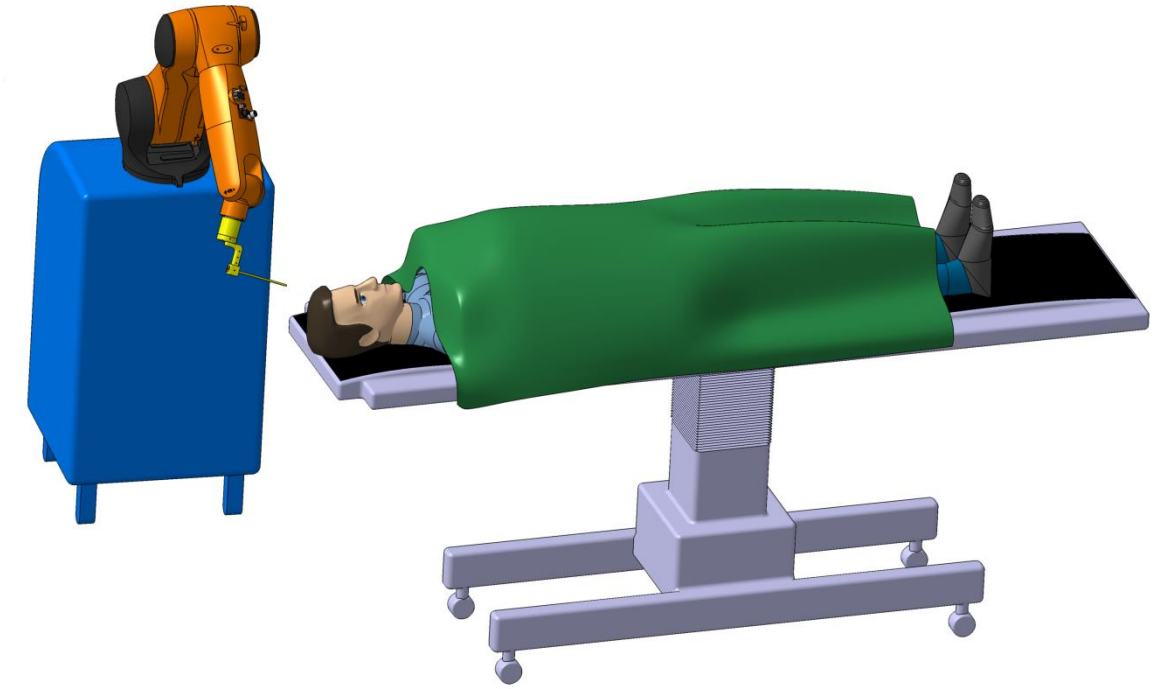
Os robota	Standardna konfiguracija	Ograničenja
A_1	$+ / -170°$	$+ / -100°$
A_2	$+45° / -190°$	$+45° / -190°$
A_3	$+156° / -120°$	$+136° / -70°$
A_4	$+ / -185°$	$+ / -120°$
A_5	$+ / -120°$	$+ / -115°$
A_6	$+ / -350°$	$+ / -350°$

7.2. Inicijalni eksperimentalni rezultati

U ovim simulacijama položaj između robota i pacijenta je istovjetan, a robot se nalazi pokraj glave pacijenta (Slika 39). U svakom odjeljku prikazana je tablica s ulaznim parametrima tog eksperimenta.

Pojedina tablica sadrži:

- vrstu vodilice
- početne i završne vrijednosti x,y,z koordinata (u mm) i broj podjela između početnih i završnih vrijednosti
- početne i završne vrijednosti alfa i beta kutova (u stupnjevima) i broj podjela između početnih i završnih vrijednosti
- podjela theta kuta (u stupnjevima)
- duljinu linearnog kretanja (u mm)
- broj trajektorija
- procesor na kojemu su izvršene simulacije

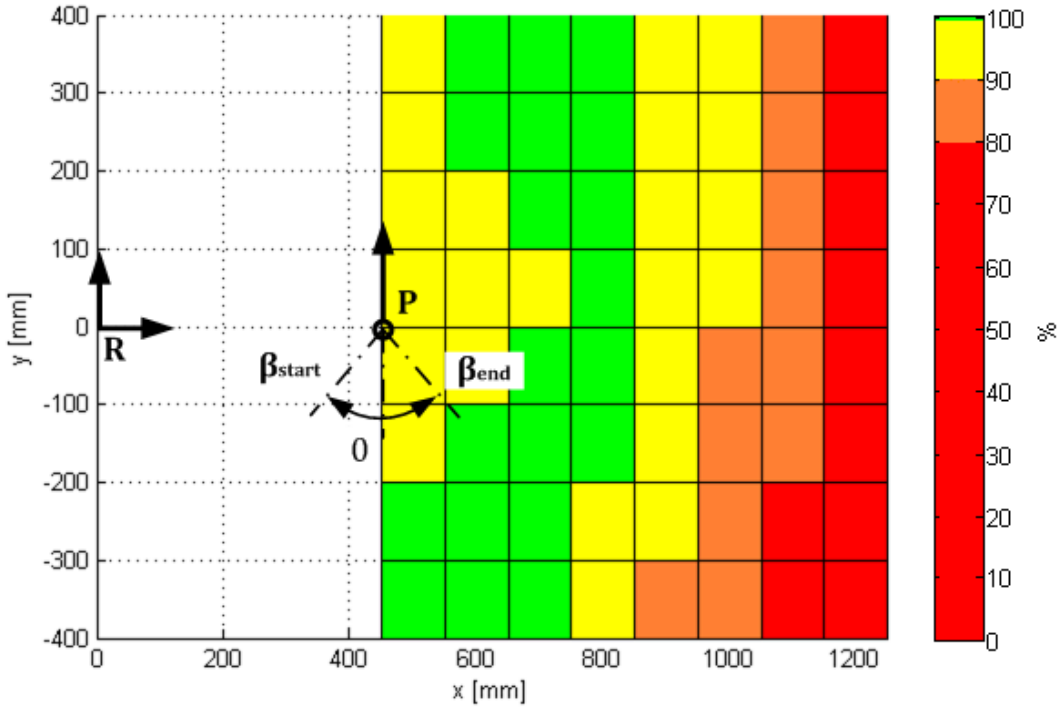


Slika 39 – Položaj robota pokraj glave pacijenta

Uspješnost simulacije bit će prikazana na dva grafa. Prvi od njih prikazivat će uspješnost na svim visinama ispitivanja, nijansama boja od 0% (plava) do 100% (crvena). Zbog bolje preglednosti u poglavlju je kao primjer prikazan samo jedan takav graf, dok se ostali mogu pronaći u prilogu (Prilog III). Drugi graf je aritmetička sredina svih visina ispitivanja te je zbog bolje preglednosti pojednostavljen i prikazuje vrijednosti pomoću četiri boje:

- 0 – 80% - crvena
- 80 – 90% - narančasta
- 90 – 100% - žuta
- 100% - zelena

Pozicija i orijentacija robota u odnosu na orijentaciju pacijenta i područje ispitivanja prikazana je na slici (Slika 40).



Slika 40 – Položaj i orijentacija robota u odnosu na orijentaciju pacijenta i područje ispitivanja (R – robot, P -pacijent)

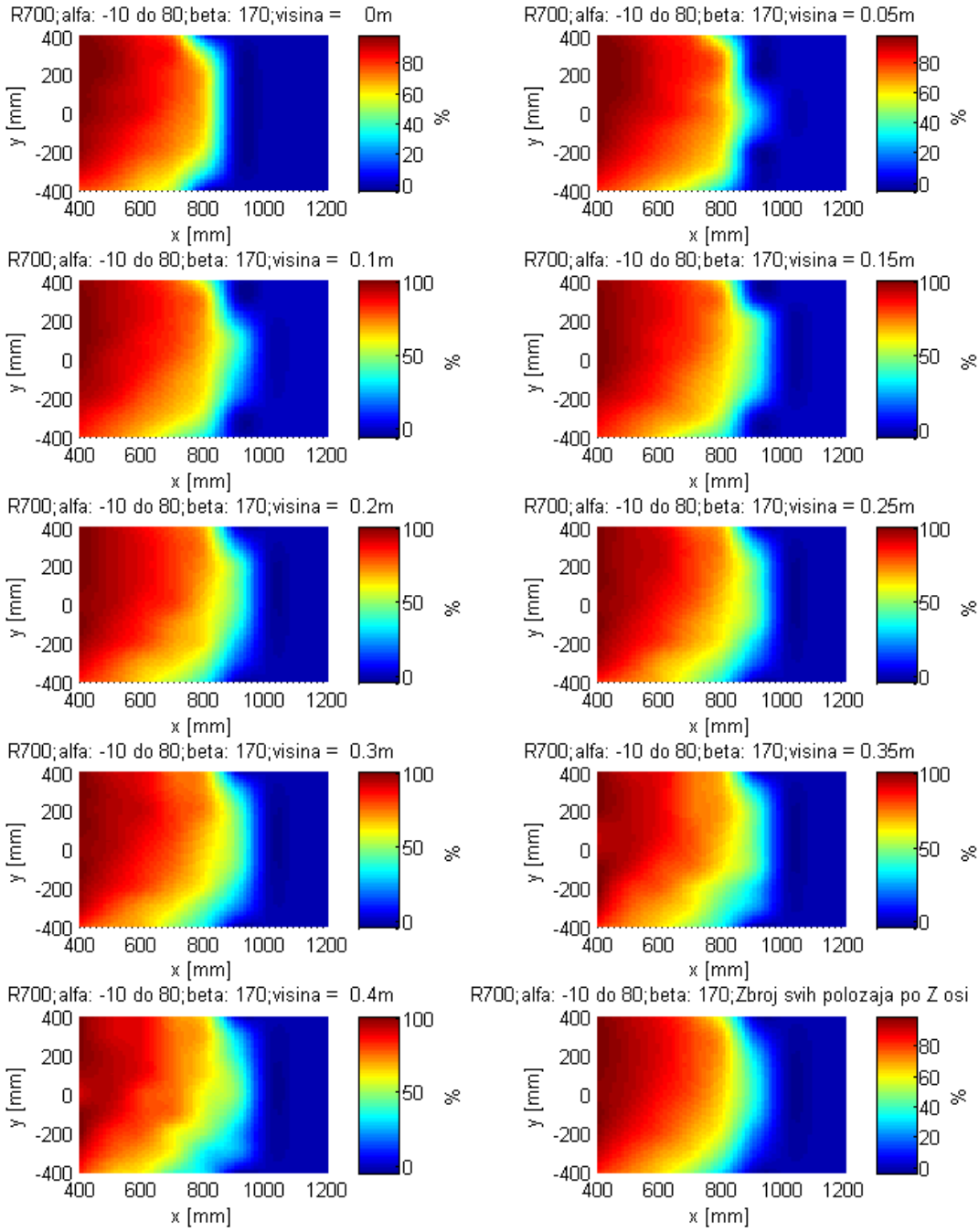
7.2.1. Simulacija a)

Tablica 8 – Parametri eksperimentalne simulacije a)

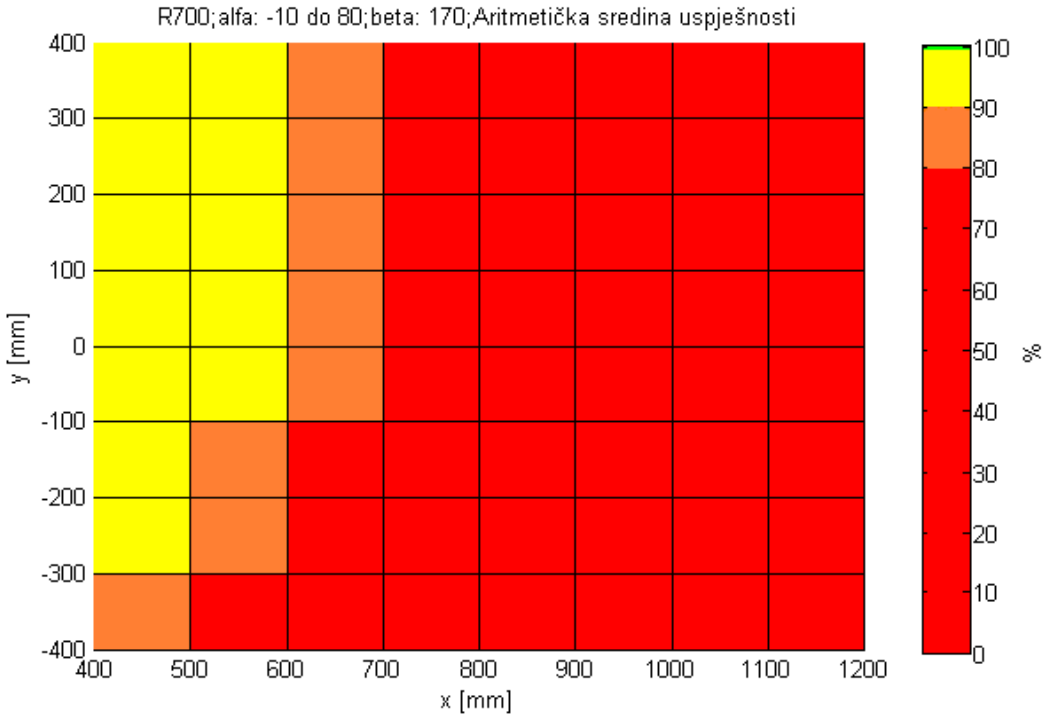
Parametar		Vrijednost
$Vrsta\ vodilice$		S biopsijskom sondom
x	x_{start}	450
	x_{end}	1250
	$x_{division}$	8
y	y_{start}	-400
	y_{end}	400
	$y_{division}$	8
z	z_{start}	0
	z_{end}	400
	$z_{division}$	8
α	α_{start}	-10
	α_{end}	80
	$\alpha_{division}$	5
β	β_{start}	-85
	β_{end}	85
	$\beta_{division}$	5
$\vartheta_{division}$		5
$duljina\ linearnog\ gibanja$		80
$broj\ trajektorija$		26 244
$procesor$		$Intel\ B\ 960\ 2.20GHz$

7.2.1.1. Kuka Agilus KR6 R700

Vrijeme trajanja simulacije: $33h\ i\ 56min$



Slika 41 – Uspješnost robota R700 na svim visinama (simulacija **a**)



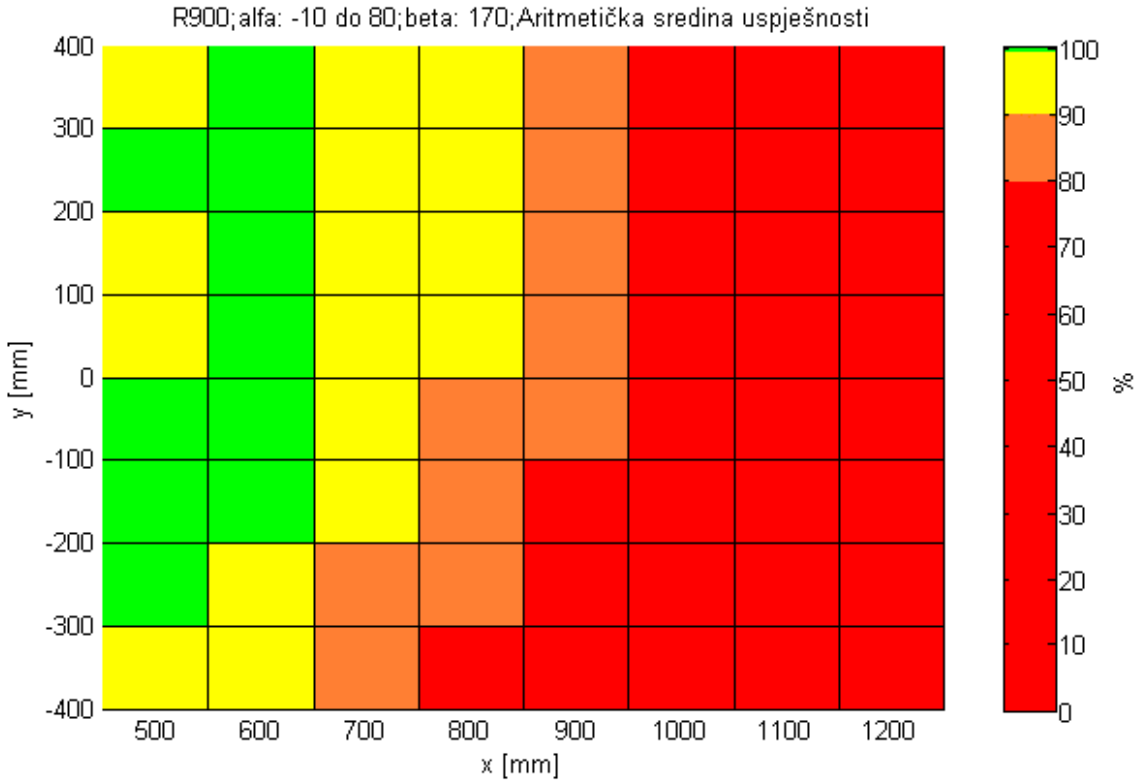
Slika 42 – Pojednostavljeni prikaz uspješnosti robota R700 (simulacija **a**)

Simulacija a) parametarski je najmanje zahtjevna od sva tri inicijalna eksperimenta. Težinu svake simulacije određuju alfa i beta kutovi s obzirom da su ostali parametri identični. Prva slika KR700 robota (Slika 41) prikazuje područja uspješnosti na svim visinama. Na njoj je vidljivo da robot ima nekoliko visina na kojima je uspješnost 100%. Logično je, s obzirom na doseg R700 robota, da se najuspješnije točke nalaze blizu ishodišta robota. Valja primijetiti da robot R700 u područjima x_{start} ima veću uspješnost na pozitivnoj strani y osi, što nije slučaj s preostala dva robota. Područja od $x = 600\,mm$ postaju predaleka meta za robota ovako malog dosega, kao i dijelovi početnih visina $z = 0\,i\,50\,mm$. Razlog lošeg dosega na malim visinama je nemogućnost dohvata početnih α kutova ($\alpha = -10°, 0°$).

S obzirom da preostale dvije simulacije imaju parametre većih raspona, za očekivati je još lošije rezultate, što je vidljivo dalje u odlomku.

7.2.1.2. Kuka Agilus KR6 R900

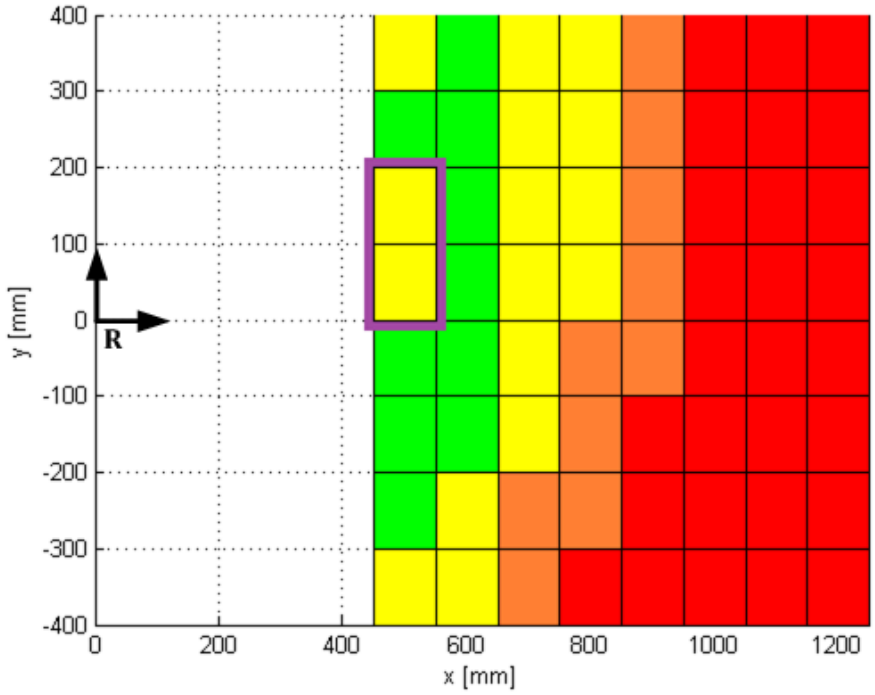
Vrijeme trajanje simulacije: $24h\ i\ 26min$



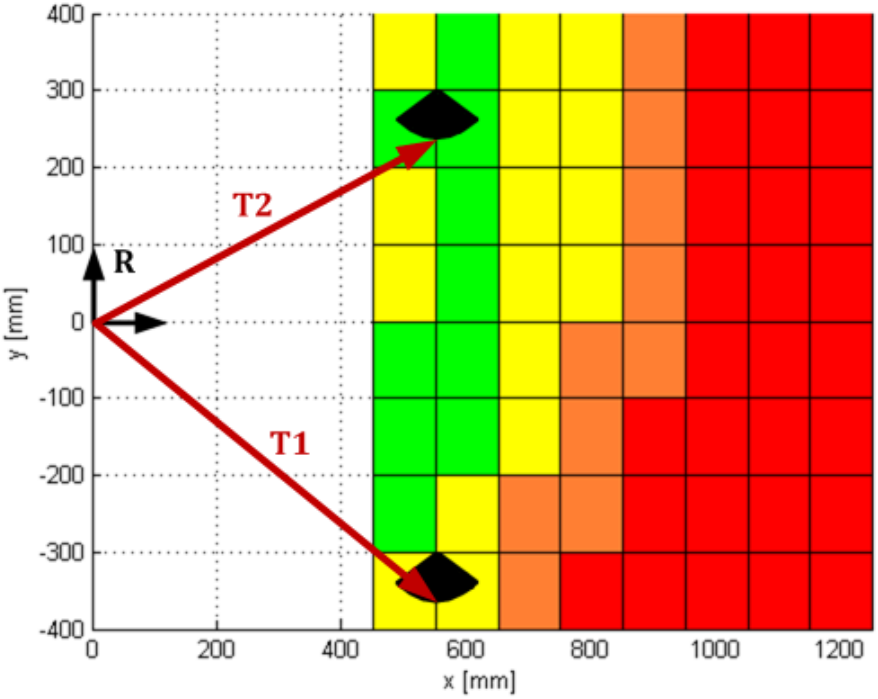
Slika 43 - Pojednostavljeni prikaz uspješnosti robota R900 (simulacija **a**)

Robot R900, kojeg se koristi na RONNA sustavu, pokazuje poprilično bolje rezultate. Kako je spomenuto ranije, u područjima x_{start} dva veća robota ponašaju se drugačije od R700 robota te je na tom području uspješnost veća na negativnoj strani y osi. Razlog je onemogućeno kretanje robota uslijed kolizije sa samim sobom u područjima kada Target točka bude na x_{start} i $y = 0\ i\ 100\ mm$. U tim rubnim slučajevima, radni prostor je vrlo blizu robota te robot nije u mogućnosti doći u Entry točku s ovim iznosom linearnog pomaka (Slika 44). Kritične trajektorije su na kutovima β_{start} i $\alpha = 8°\ i\ 26°$. Uspješnost bi mogla biti veća smanjenjem linearnog pomaka Entry točke, no to bi moglo negativno utjecati na planiranje operativnog zahvata. Odmakom od ishodišta na vrijednosti $x = 550\ mm$, područje sa 100% uspješnosti dominantnije je u pozitivnom dijelu y osi (skup trajektorija T2)(Slika 45). U tom dijelu radni prostor nalazi se dovoljno daleko pa nema opasnosti od kolizije robota sa samim sobom. S

druge strane, trajektorije u $y = -300mm$ postaju predaleke da bi robot mogao potpuno izvršiti sve kretnje (skup trajektorija T1) (Slika 45).



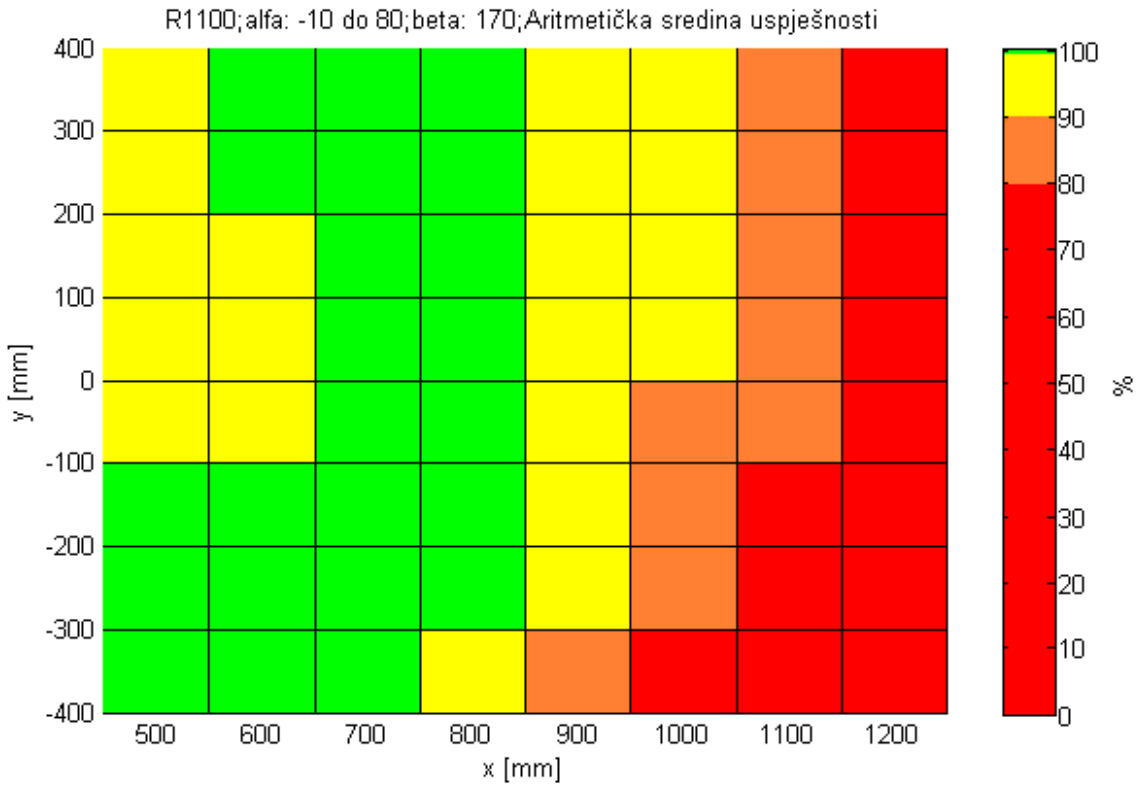
Slika 44 – Područje smanjenih radnih mogućnosti u blizini robota



Slika 45 – Skupovi trajektorija T1 i T2

7.2.1.3. Kuka Agilus KR10 R1100

Vrijeme trajanja simulacije: $30h\ i\ 15min$



Slika 46 - Pojednostavljeni prikaz uspješnosti robota R1100 (simulacija **a**)

Robot R1100 pokazao je puno bolje rezultate od svojih prethodnika. Naglasak kod ovog robota je radni prostor 2.3 puta veći nego kod robota R900. Svoju izvrsnost pokazao je na velikom području negativne y osi, a na cijelom radnom prostoru između $x = 650\,mm$ i $x = 850\,mm$ ima skoro pa 100% učinkovitost. Nažalost, zbog većih dimenzija ne može izvršiti veliki broj trajektorija u prostoru oko x_{start} i $y = 0\,mm$. Odličnim radom na području daljem od ishodišta lakše je naći poziciju u kojoj će robot minimalno smetati djelokrugu rada kirurga.

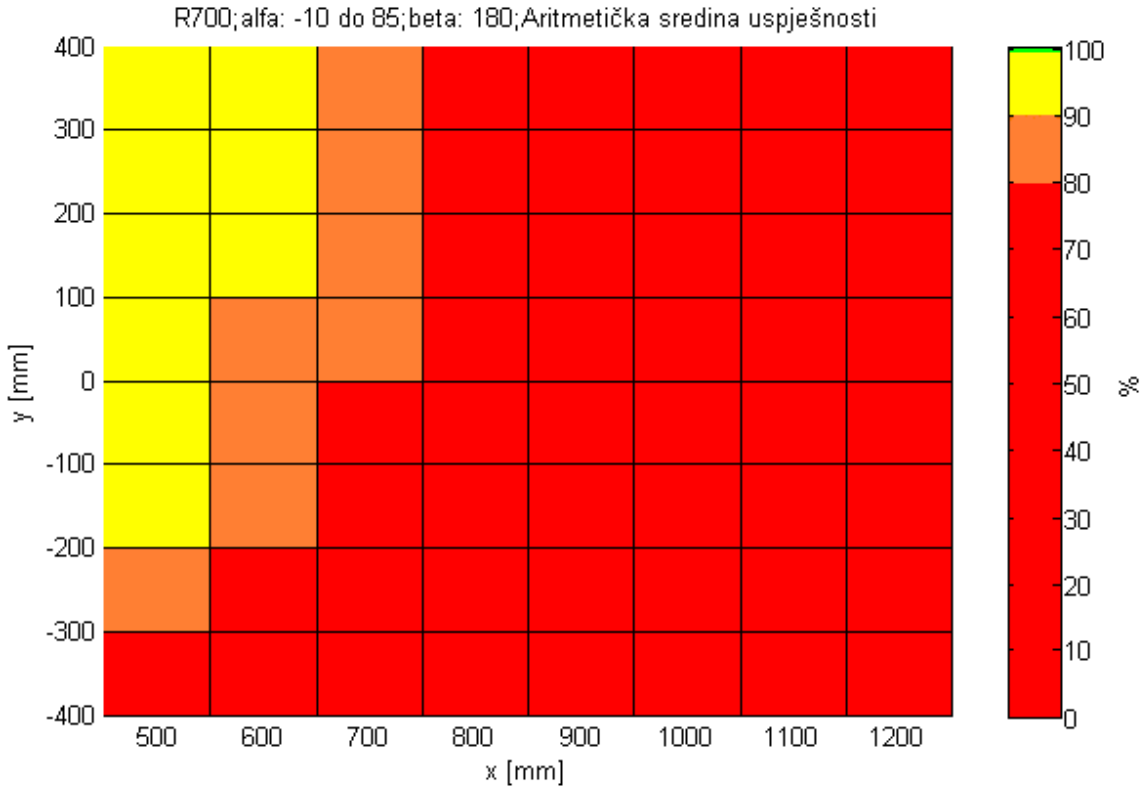
7.2.2. Simulacija b)

Tablica 9 - Parametri eksperimentalne simulacije b)

Parametar		Vrijednost
$Vrsta\ vodilice$		S biopsijskom sondom
x	x_{start}	450
	x_{end}	1250
	$x_{division}$	8
y	y_{start}	-400
	y_{end}	400
	$y_{division}$	8
z	z_{start}	0
	z_{end}	400
	$z_{division}$	8
α	α_{start}	-10
	α_{end}	85
	$\alpha_{division}$	5
β	β_{start}	-90
	β_{end}	90
	$\beta_{division}$	5
$\vartheta_{division}$		5
$duljina\ linearnog\ gibanja$		80
$broj\ trajektorija$		26 244
$procesor$		$Intel\ i7-6800K\ 3.40GHz$

7.2.2.1. Kuka Agilus KR6 R700

Vrijeme trajanja simulacije: $15h\ i\ 12min$

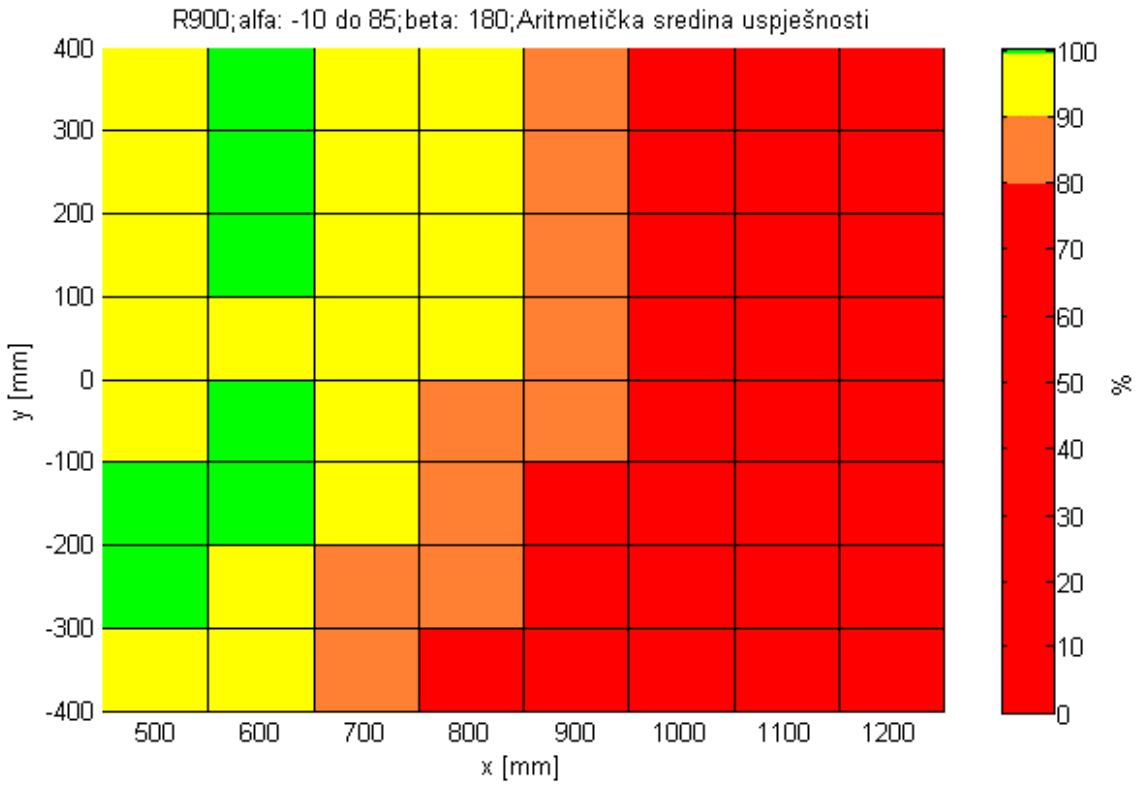


Slika 47 - Pojednostavljeni prikaz uspješnosti robota R700 (simulacija **b**)

U odnosu na prethodnu simulaciju, simulacija b) ima male izmjene kutova. S ovom konfiguracijom parametara robot R700 nije uspio niti na jednoj visini izvršiti skup trajektorija sa 100% uspješnosti (Slika 80). Nadalje, povećan je prostor u blizini ishodišta u kojem robot ne može postići konačne trajektorije.

7.2.2.2. Kuka Agilus KR6 R900

Vrijeme trajanja simulacije: $17h\ i\ 22min$

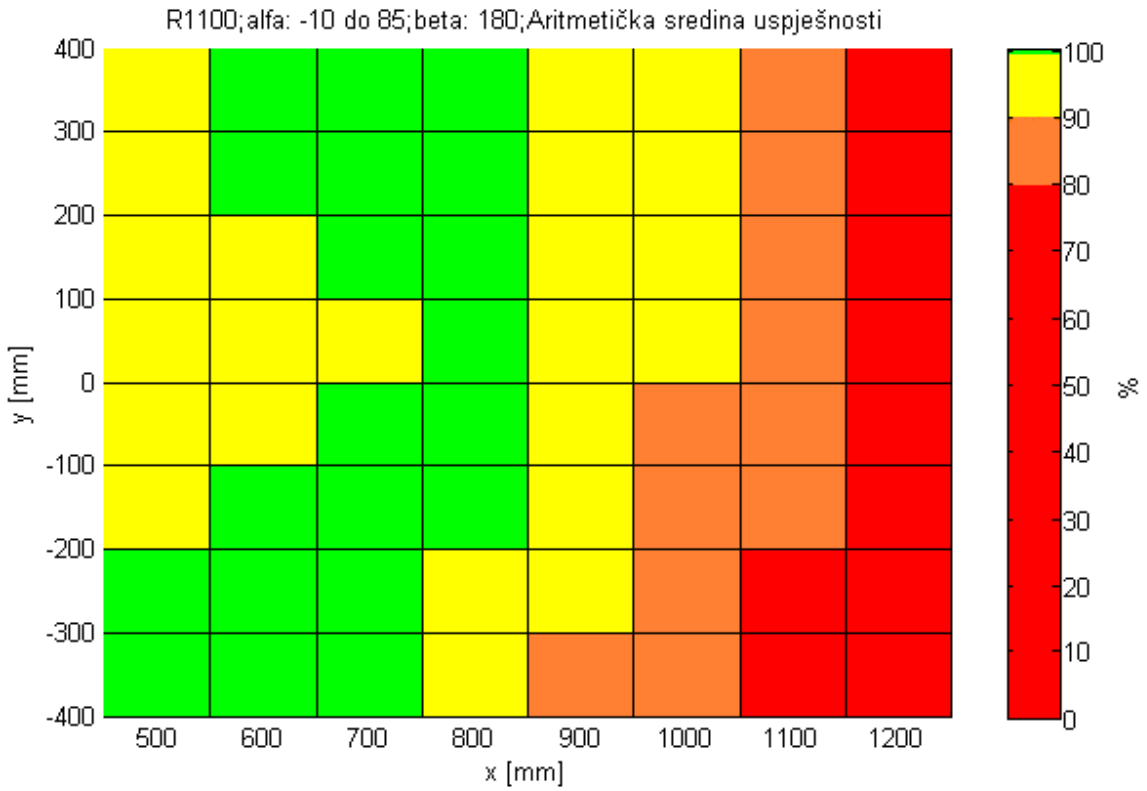


Slika 48 - Pojednostavljeni prikaz uspješnosti robota R900 (simulacija **b**)

Promjena parametera donijela je određene promjene i kod R900 robota te se njegov prostor 100% uspješnosti smanjio za 30%.

7.2.2.3. Kuka Agilus KR10 R1100

Vrijeme trajanja simulacije: $18h\ i\ 15min$



Slika 49 - Pojednostavljeni prikaz uspješnosti robota R1100 (simulacija **b**)

Povećanje raspona kutova α i β utjecalo je negativno i na radno područje R1100 robota te je pridonijelo smanjenju optimalnog radnog prostora od 13%.

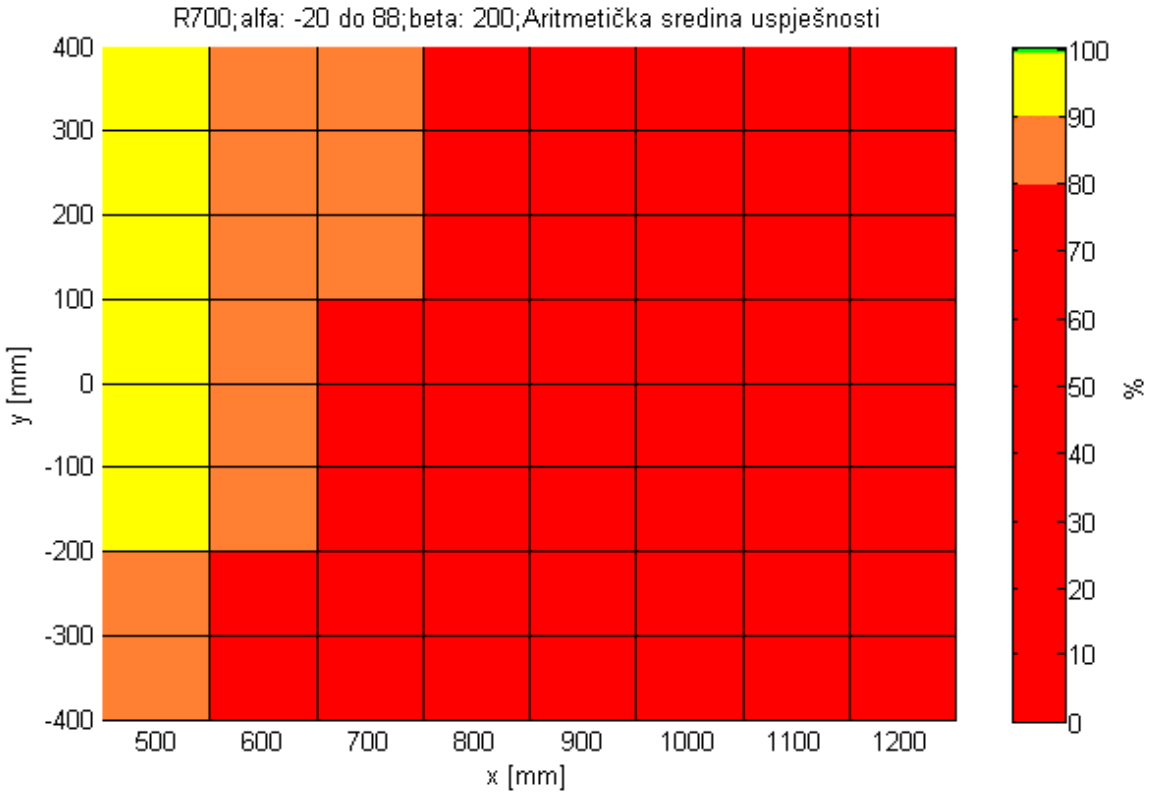
7.2.3. Simulacija c)

Tablica 10 - Parametri eksperimentalne simulacije c)

Parametar		Vrijednost
$Vrsta\ vodilice$		S biopsijskom sondom
x	x_{start}	450
	x_{end}	1250
	$x_{division}$	8
y	y_{start}	-400
	y_{end}	400
	$y_{division}$	8
z	z_{start}	0
	z_{end}	400
	$z_{division}$	8
α	α_{start}	-20
	α_{end}	88
	$\alpha_{division}$	5
β	β_{start}	-100
	β_{end}	100
	$\beta_{division}$	5
$\vartheta_{division}$		5
$duljina\ linearnog\ gibanja$		80
$broj\ trajektorija$		26 244
$procesor$		$Intel\ i5-2400\ 3.10GHz$

7.2.3.1. Kuka Agilus KR6 R700

Vrijeme trajanja simulacije: $14h\ i\ 29min$

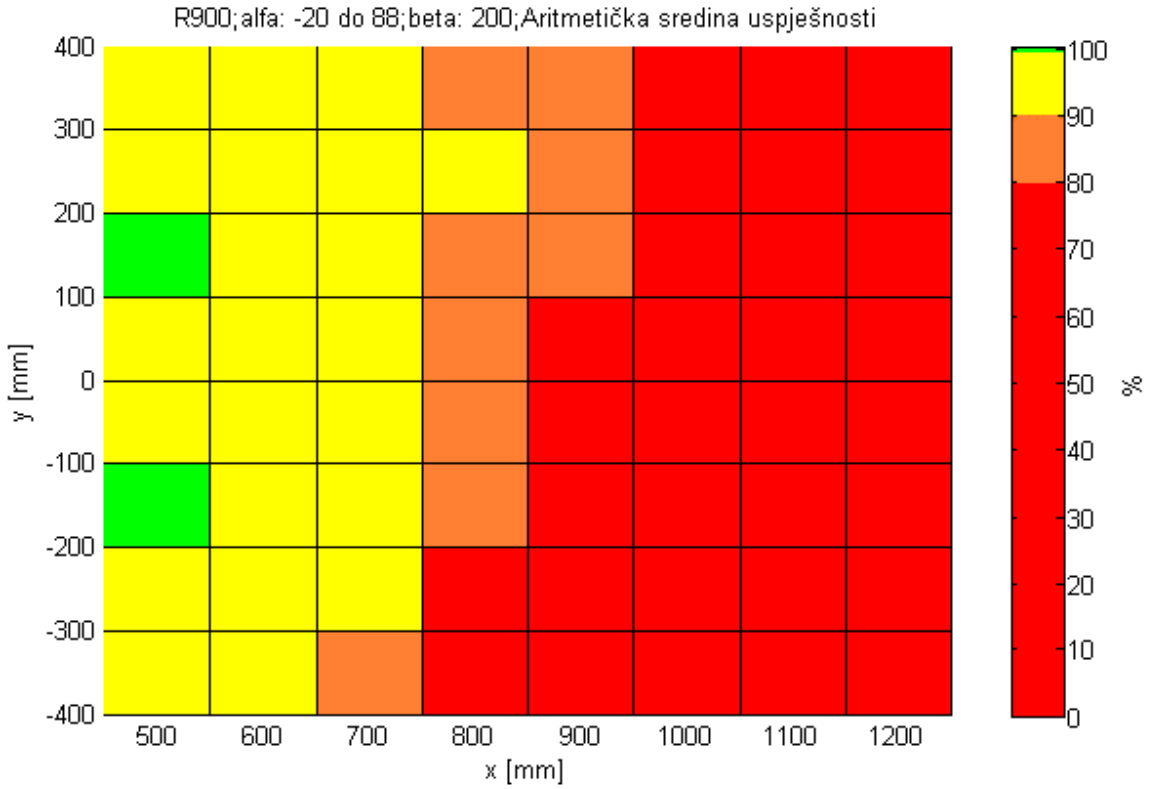


Slika 50 - Pojednostavljeni prikaz uspješnosti robota R700 (simulacija **c**)

Posljednja inicijalna simulacija donosi ekstremne vrijednosti kutova α i β, stoga nije neočekivan loš rezultat robota R700, koji niti s jednostavnijim parametrima nije pokazivao radna područja sa 100% uspješnosti.

7.2.3.2. Kuka Agilus KR6 R900

Vrijeme trajanja simulacije: 17*h*

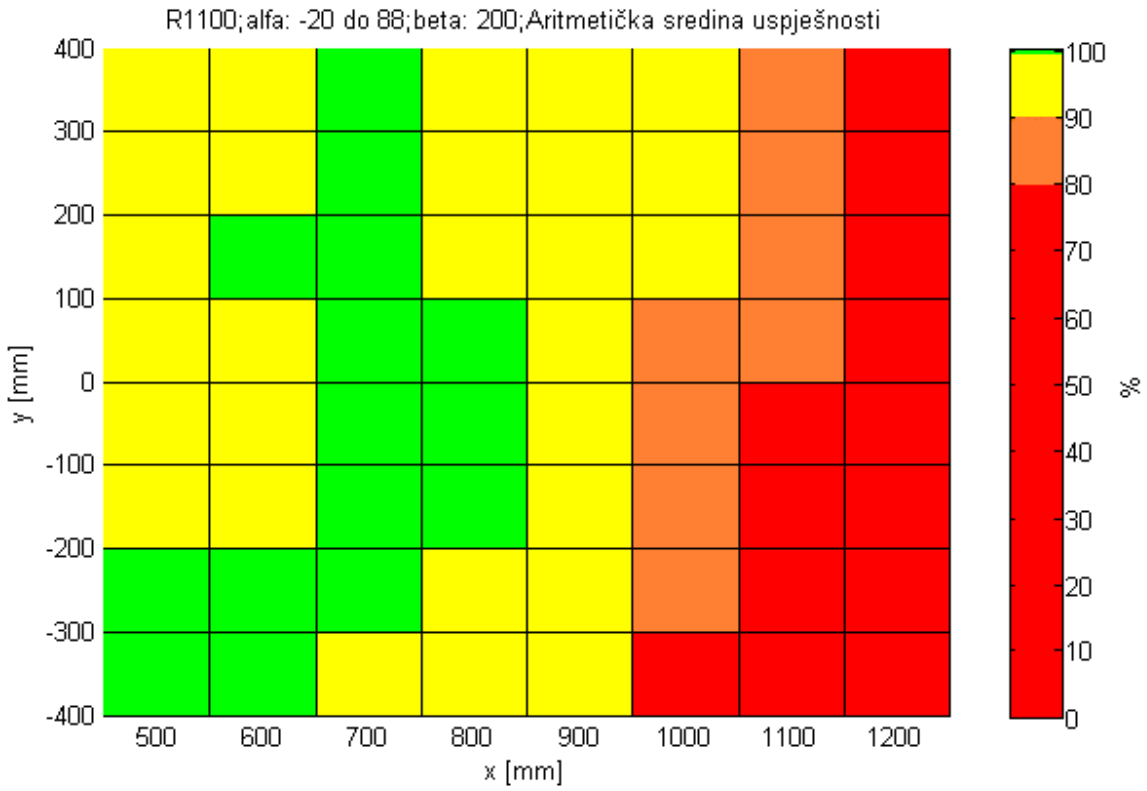


Slika 51 - Pojednostavljeni prikaz uspješnosti robota R900 (simulacija **c**)

Radno područje s novim parametrima donijelo je velike promjene kod robota R900. Područje s potpunom uspješnosti drastično se smanjilo te sada obuhvaća samo 20% prostora u odnosu na simulaciju a).

7.2.3.3. Kuka Agilus KR10 R1100

Vrijeme trajanja simulacije: $17h\ i\ 57min$



Slika 52 - Pojednostavljeni prikaz uspješnosti robota R1100 (simulacija **c**)

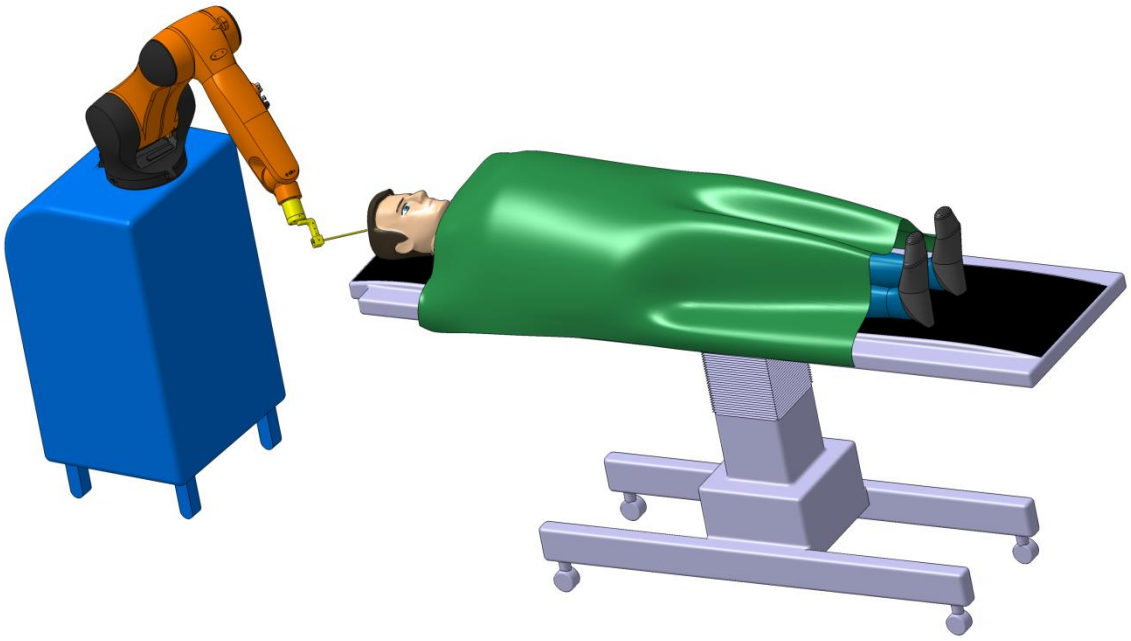
Bez obzira na zahtjevne iznose parametara robot R1100 i dalje pokazuje vrlo dobre rezultate. U odnosu na simulaciju b) prostor blizu ishodišta robota se poboljšao jer su s podjelom β kuta na pet dijelova izbjegnute kritične trajektorije kada je $\beta = 90°$.

Nakon prvih eksperimentalnih rezultata koji su dali grubi uvid u područja u kojima bi roboti mogli raditi s najvećom učinkovitosti, napravljeno je nekoliko simulacija s varijacijama koje su utjecale na ishod optimalnog radnog područja. Konačan cilj je povećati radno područje sa 100% uspješnosti.

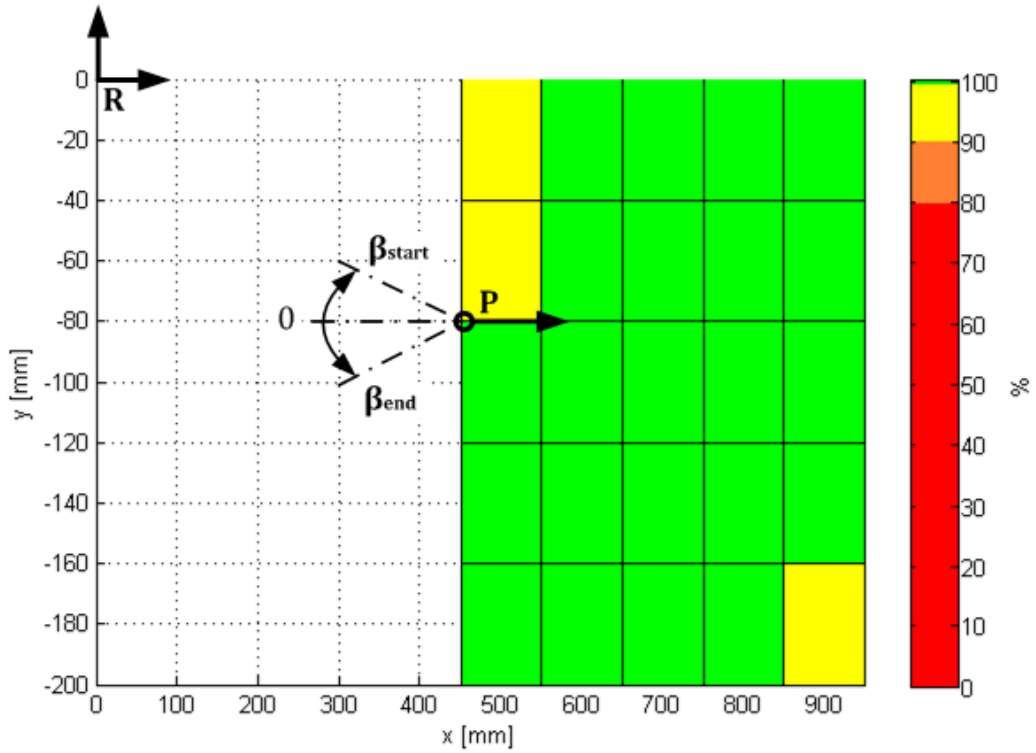
Neke od izmjena su:

- Promjena alata - alat čiji je TCP bliži prirubnici robota
- Linearni pomak – smanjena je duljina linearnog pomaka
- Položaj između robota i pacijenta – robot se nalazi iza glave pacijenta (Slika 53) [12]

7.3. Eksperimentalni rezultati s varijacijama



Slika 53 – Položaj robota iza glave pacijenta



Slika 54 - Položaj i orijentacija robota u odnosu na orijentaciju pacijenta i područje ispitivanja (R – robot, P -pacijent)

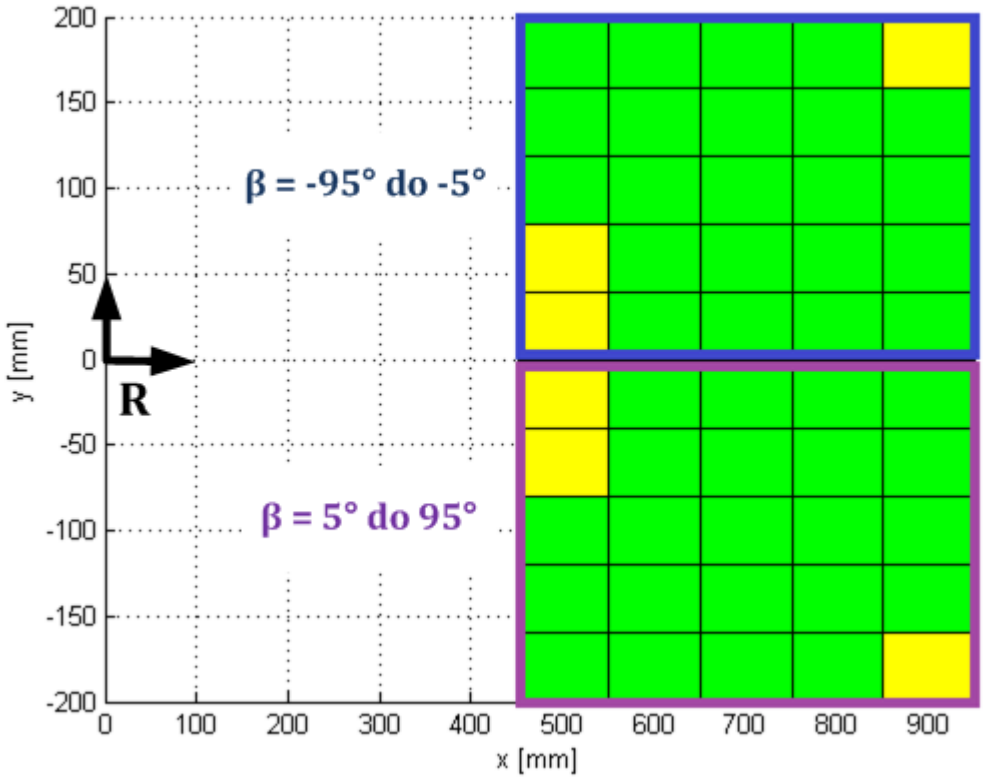
7.3.1. Položaj robota iza glave pacijenta i promjena linearnog pomaka

U ovome odjeljku prikazani su rezultati za položaj robota iza glave pacijenta te usporedba dvaju linearnih pomaka od $50mm$ i $80mm$.

Tablica 11 - Parametri simulacije s promjenom linearnog pomaka

Parametar		Vrijednost
$Robot$		$R900$
$Vrsta\ vodilice$		S biopsijskom sondom
x	x_{start}	450
	x_{end}	950
	$x_{division}$	5
y	y_{start}	-200
	y_{end}	0
	$y_{division}$	5
z	z_{start}	0
	z_{end}	300
	$z_{division}$	6
α	α_{start}	-5
	α_{end}	85
	$\alpha_{division}$	10
β	β_{start}	5
	β_{end}	95
	$\beta_{division}$	9
$\vartheta_{division}$		5
$duljina\ linearnog\ gibanja$		50 *i* 80
$broj\ trajektorija$		27 720

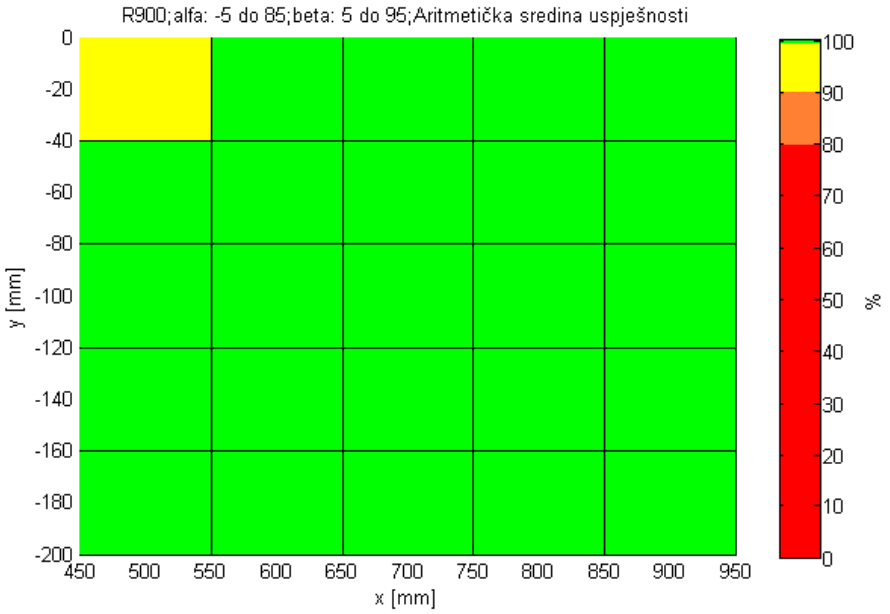
Zbog specifičnog položaja robota u odnosu na pacijenta, pretraživao se prostor samo u negativnom smjeru y osi. Situacija bi bila identična i u pozitivnu stranu za kutove $\beta_{start} = -95$ i $\beta_{end} = -5$ (Slika 55).



Slika 55 – Simetričnost radnog područja u poziciji robota iza glave pacijenta

Ovakav prostorni raspored između pacijenta i robota daje nove mogućnosti u planiranju operativnog zahvata. S obzirom da je robot R900 trenutno u upotrebi na RONNA sustavu, naglasak je bio na tome da se isprobaju njegove mogućnosti u prikazanom načinu rada. Zbog vjerodostojnijih rezultata radni prostor je fokusiran na manje područje nego ono u inicijalnim simulacijama, ali s gušćom razdiobom y područja te α i β kutova.

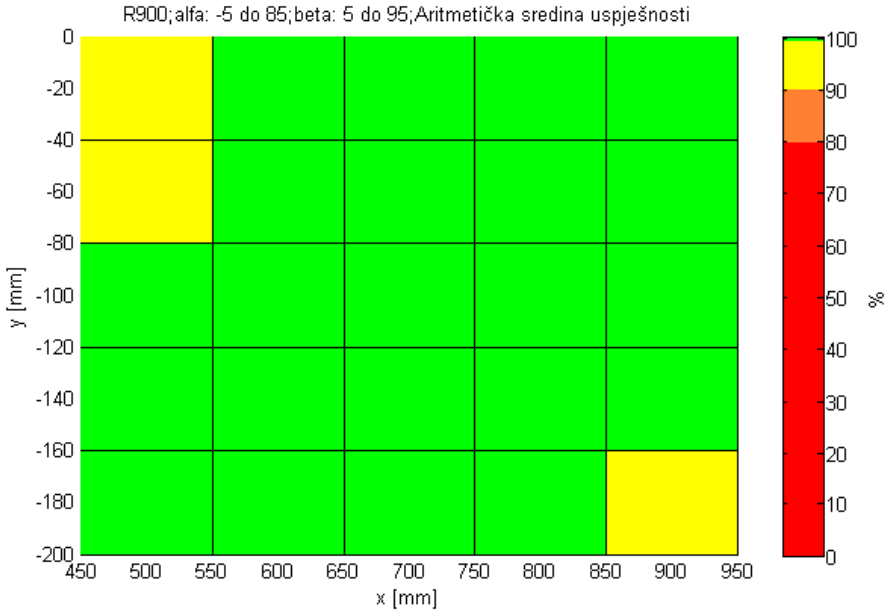
Vrijeme trajanja simulacije (*linearni pomak*: $50mm$**):** $10h\ i\ 54min$



Slika 56 - Pojednostavljeni prikaz uspješnosti robota R900 (linearni pomak: $\mathbf{50}\boldsymbol{mm}$)

Prva simulacija dala je 100% uspješne rezultate na gotovo cijelom radnom području. U ovom slučaju, kao i u prethodnima, postoji kritičan dio u kojem robot nema mogućnost dolaska u sve konačne trajektorije (problem nastaje od $z = 150mm$). Za ovaj primjer je to područje vrlo malo jer se radi o linearnom pomaku $50mm$ udaljenim od Target točke. Takav pomak može biti nedostatan u planiranju zahvata u intrakranijalnom području pacijenta. Za daljnje istraživanje u ovakvom načinu rada robota preporučuje se analizirati veći radni prostor (osobito u y smjeru) te veći raspon α kuta kako bi se dobila šira slika stvarnih mogućnosti robota.

Vrijeme trajanja simulacije (*linearni pomak*: $80mm$**):** $15h\ i\ 51min$

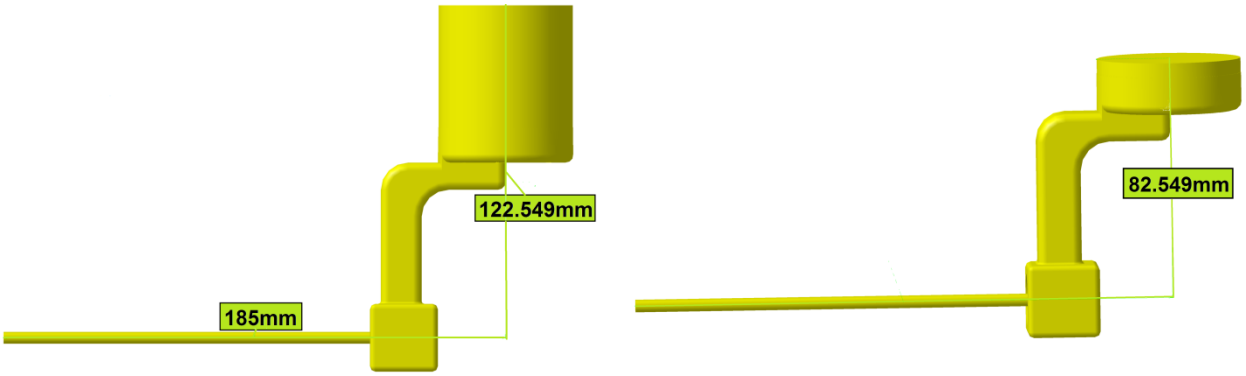


Slika 57 – Pojednostavljeni prikaz uspješnosti robota R900 (linearni pomak: $\boldsymbol{80mm}$)

Drugi primjer očekivano je lošiji jer se koristi linearni pomak od $80mm$ u odnosu na Target točku. Veći raspon pomaka utjecao je na smanjenje uspješnosti u najbližem (problem s kolizijom) te najdaljem (nedohvatljive trajektorije) području od robota. Dulji linearni pomak pogodniji je u planiranju operativnog zahvata te se za daljnju analizu preporučuje testiranje robota R1100 s parametrima korištenima u ovoj simulaciji. Prema dosadašnjim rezultatima predviđa se da bi robot R1100 imao više problema u području bliže ishodištu, a s druge strane 100% uspješnost bi trebao ostvariti u domeni široj od prikazane simulacije.

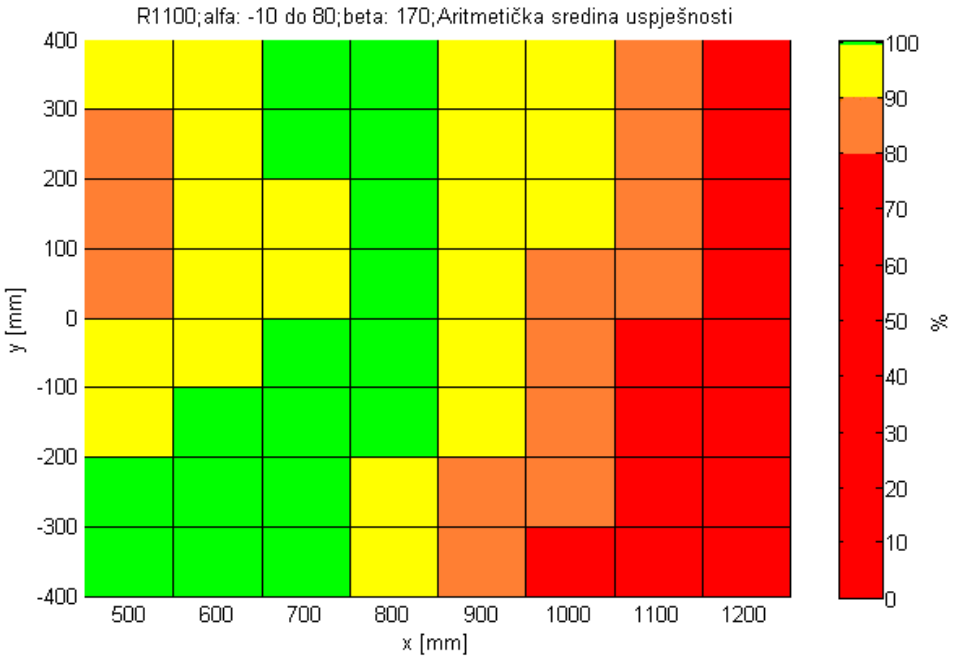
7.3.2. Robot pokraj glave pacijenta – TCP vodilice bliži prirubnici robota

U ovoj simulaciji koriste se ulazni parametri iz eksperimentalne simulacije a) (Tablica 8) s razlikom u promjeni alata. TCP nove vodilice je $40mm$ bliže prirubnici od dosadašnje (Slika 58).



Slika 58 – Skraćena verzija vodilice

Vrijeme trajanja simulacije: $25h\ i\ 37min$



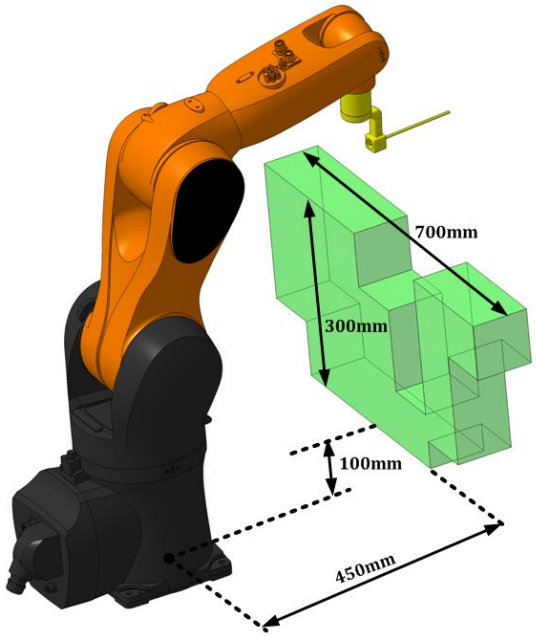
Slika 59 – Pojednostavljeni prikaz uspješnosti robota R1100 s kraćom vodilicom

Simulacija je provedena na robotu R1100 jer do sada pokazao najbolje rezultate pa je najzanimljiviji za promatranje. Eksperiment s promjenom dimenzije alata dao je rezultate lošije od očekivanih. Područja sa 100% uspješnosti smanjila su se za 23%, a dogodio se i pad na području $90\% - 100\%$. Za daljnja istraživanja preporuča se isprobati simulaciju na preostala dva manja robota.

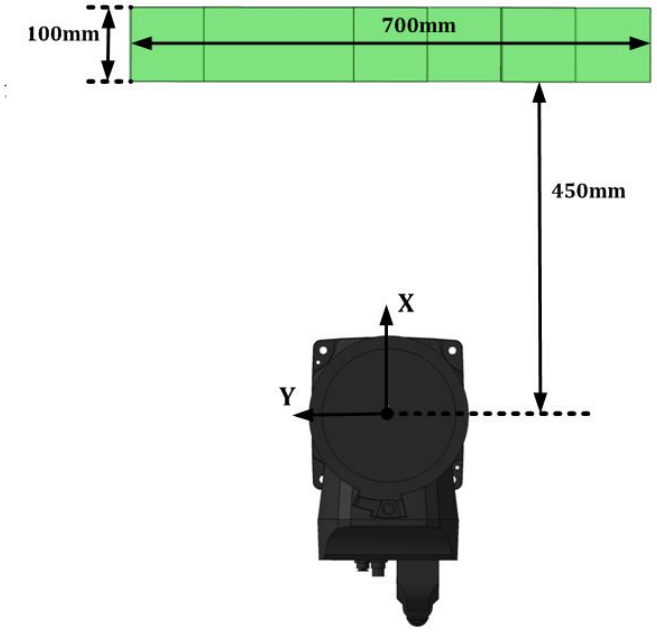
7.4. Usporedba radnih prostora robota

Nakon provedenih inicijalnih eksperimenata za sva tri slučaja ulaznih parametara, vidljivo je da svi roboti daju najbolje rezultate u simulaciji a), a robot R1100 pokriva najveću površinu sa 100% učinkovitosti.

Robot R700 nije se pokazao dobrim niti u jednom eksperimentu, a njegova je najveća učinkovitost s parametrima za simulaciju a) (Slika 60 i Slika 61). Doseg od $700mm$ nedovoljan je da robot postigne sve trajektorije na malim visinama ($z < 100mm$) osobito u području negativnog kuta α_{start}. Loši rezultati mogu se pripisati i nezahvalnom odabiru početne vrijednost na x osi, no u realnim uvjetima pacijenta se rijetko stavlja na područje bliže od $x_{start} = 400mm$ u odnosu na robota. Rezultati bi mogli biti bolji ukoliko se robota R700 postavi iza glave pacijenta.

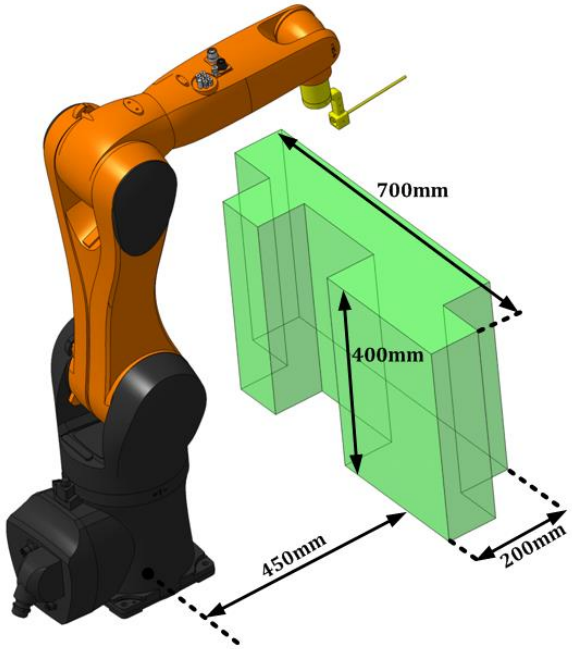


Slika 60 – Najveći radni prostor sa 100% uspješnosti za robota R700

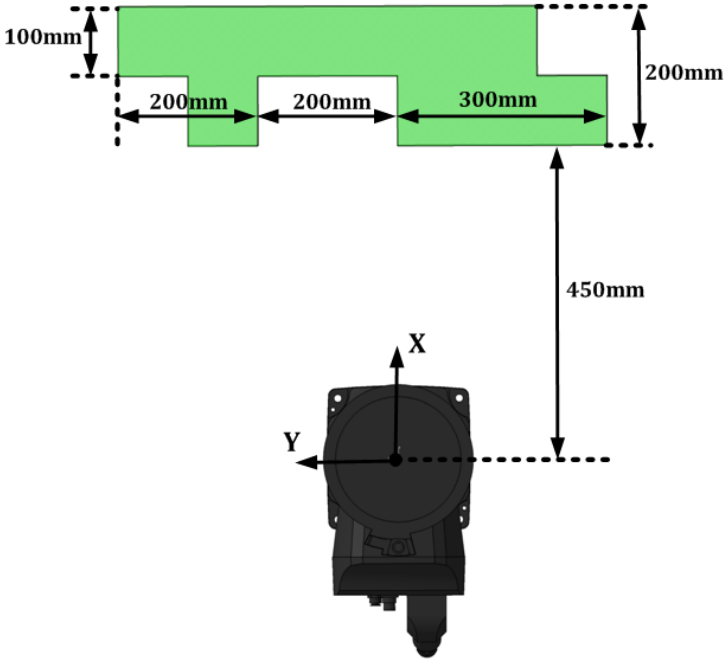


Slika 61 - Najveći radni prostor sa 100% uspješnosti za robota R700 (tlocrt)

Robot R900 obuhvaća prihvatljivo područje sa 100% uspješnosti u simulaciji s manje zahtjevnim parametrima (Slika 62 i Slika 63). Što su parametri simulacija postajali zahtjevniji, optimalno radno područje drastično je padalo. U simulaciji b) ono je obuhvaćalo 70% prvotnog radnog prostora te se u zahtjevnoj c) simulaciji smanjilo na svega 20%. Varijacije u kojima se robot nalazi iza glave pacijenta pokazale su se odličnima na isprobanom radnom prostoru, a promjena duljine linearnog gibanja pokazala je očekivane rezultate. Zanimljive varijacije za buduća istraživanja su: proširiti granice radnog područja te pronaći radni prostor u kojemu bi linearna kretnja iznosila više od $80mm$. Iznimno je važno pronaći područje u kojem robot može ostvarivati tako duge kretnje sa 100% učinkovitosti jer su to realni pomaci tijekom operacije.

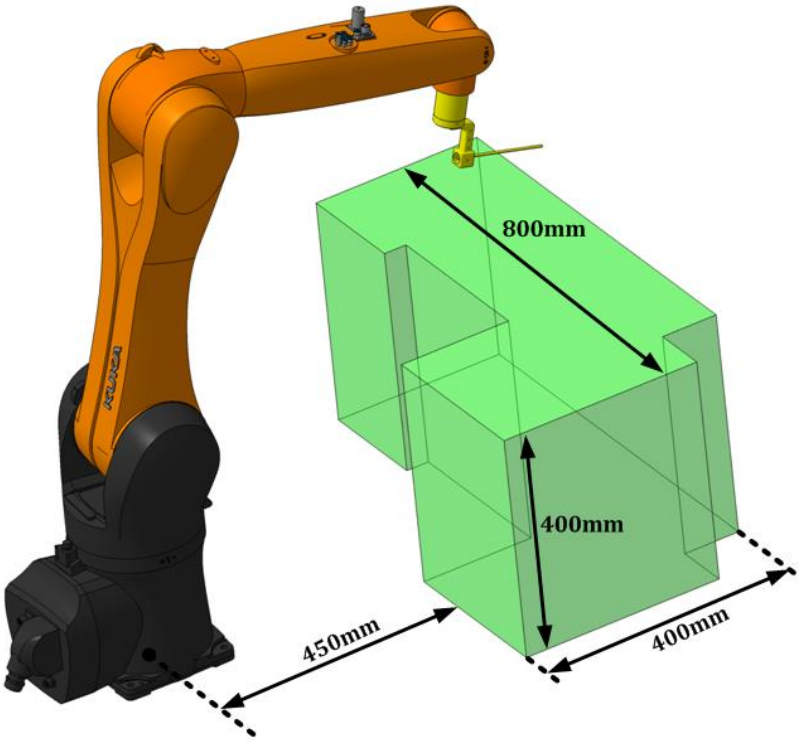


Slika 62 – Najveći radni prostor sa **100**% uspješnosti za robota R900

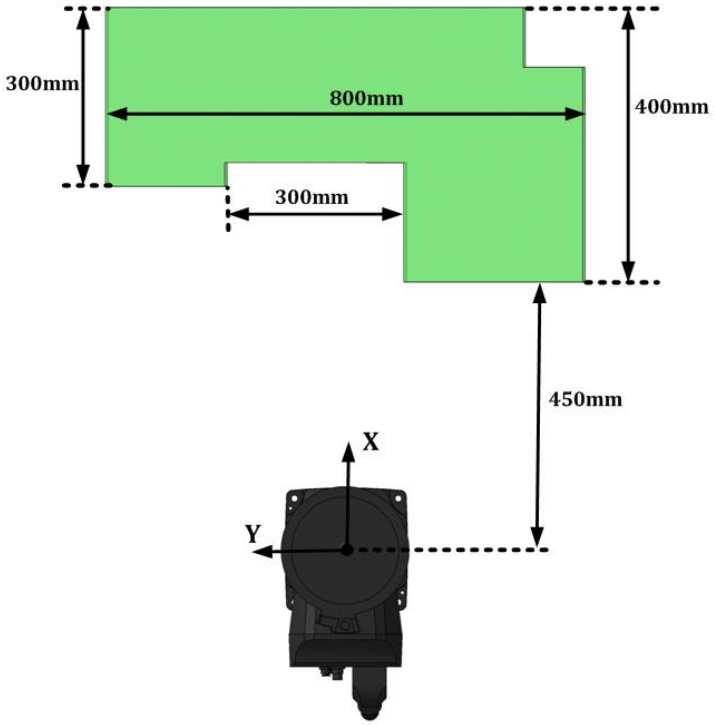


Slika 63 - Najveći radni prostor sa **100**% uspješnosti za robota R900 (tlocrt)

Robot R1100 pokazao je daleko najveće mogućnosti u identifikaciji radnog područja (Slika 64 i Slika 65). Promjene $alfa$ i $beta$ kutova u inicijalnim eksperimentima najmanje su utjecale na rad ovog robota. Nakon prve promjene radni prostor se smanjio za 13%, a nakon druge za 35% u odnosu na rezultate simulacije a). Razlog tomu je što robot ima bolju uspješnost na području daljem od ishodišta, stoga ni velike vrijednosti parametra β ne dovode robota u opasnost kolizije. Varijacija u kojoj se koristi skraćena vodilica nije polučila uspjehom. S obzirom da se robot pokazao uspješnim u većini slučajeva, bilo bi korisno ispitati njegovu učinkovitost s varijacijama linearnog pomaka u položaju iza glave pacijenta.



Slika 64 - Najveći radni prostor sa **100**% uspješnosti za robota R1100



Slika 65 - Najveći radni prostor sa **100**% uspješnosti za robota R1100 (tlocrt)

8. ZAKLJUČAK

Svrha ovog rada bila je identifikacija optimalnog radnog područja u kojem će robot moći zadovoljiti sve operativne zahtjeve u intrakranijalnom prostoru pacijenta. Umjesto robotskih sustava specifične kinematike rad se temeljio na upotrebi standardnih industrijskih Kuka robota. U istraživanju rada korištena su tri Kuka Agilus industrijska robota s dosegom: 700, 900 i 1100 mm. Virtualno okruženje operacijske sale kreirano je uz pomoć programskih paketa Python i OpenRave. Istraživani intrakranijalni prostor pacijenta bazirao se na geometrijskom obliku kuglinog isječka kojeg zatvaraju standardni stereotaktički okviri. Eksperimentalni dio rada obuhvaća simulacije različitih ulaznih parametara (α, β, ϑ, linearni pomak) te prikaz 2D i 3D grafičkih mapa radnog prostora za sve tri robotske konfiguracije. Nakon provedena tri inicijalna ispitivanja dobiven je uvid u mogućnosti pojedinog robota. Tako primjerice robot R700 pokazuje najlošije rezultate. Opseg njegovog radnog područja najmanje zadovoljava tražene kriterije. Robot R900 pokazao se nešto uspješniji u traženju optimalnog radnog prostora. Najbolji rezultat ostvaruje posljednji robot, R1100, koji u sva tri inicijalna ispitivanja ima najviše zastupljenosti 100% pokrivenog radnog područja. Nakon inicijalnih ispitivanja uslijedile su eksperimentalne simulacije s varijacijama. Dobiveni rezultati sveukupnog istraživanja postavljaju pitanja za nove istraživačke radove pri čemu se robot R1100 nameće kao idealni kandidat za buduća istraživanja.

LITERATURA

[1] Wikipedija, »Leonardo da Vinci,« 12 prosinac 2016. [Mrežno]. Available: https://en.wikipedia.org/wiki/Leonardo_da_Vinci. [Pokušaj pristupa 14 prosinac 2016].

[2] Wikipedija, »Robotika,« 19. sprnja 2016.. [Mrežno]. Available: https://hr.wikipedia.org/wiki/Robotika. [Pokušaj pristupa 14. prosinca 2016.].

[3] B. Jerbić, D. Chudy i G. Nikolić, »Primjena robota u neurokirurgiji,« 24. sprnja 2015. [Mrežno]. Available: http://stariweb.mef.hr/studmef/mef.hr/mef.hr-casopis/primjena-robota-u-neurokirurgiji.html. [Pokušaj pristupa 15. prosinca 2016.].

[4] M. leksikon, »Medicinski-leksikon,« [Mrežno]. Available: http://www.medicinski-leksikon.info/znacenje/stereotaksija.html. [Pokušaj pristupa 20. prosinca 2016.].

[5] Inomed, »RM stereotactic system,« [Mrežno]. Available: http://www.en.inomed.com/products/functional-neurosurgery/stereotactic-systems/rm-stereotactic-system/. [Pokušaj pristupa 21. prosinca 2016.].

[6] I. Stiperski, Mjerenje točnosti i preciznosti robota, Zagreb: FSB, 2016.

[7] KUKA, »KUKA small robots,« 11. studenog 2014.. [Mrežno]. Available: www.kuka-robotics.com. [Pokušaj pristupa 28. prosinca 2016.].

[8] R. Diankov, Automated Construction of Robotic Manipulation Programs, Pittsburgh, Pennsylvania: Carnegie Mellon University, 2010..

[9] Python, Python Software Production, [Mrežno]. Available: https://www.python.org/. [Pokušaj pristupa 19. prosinca 2016.].

[10] J. Vidaković, B.Jerbić, F.Šuligoj, M.Švaco i B. Šekoranja, »Position planning strategy for stereotactic robot,« 2016..

[11] Nipy, »nipy.org,« [Mrežno]. Available: http://nipy.org/dipy/theory/spherical.html. [Pokušaj pristupa 7. siječnja 2017.].

[12] J. Gonzalez-Martinez, J. Bulacio, S. Thompson, J. Gale, S. Smithason, I. Najm i W. Bingman, »Technique, Results, and Complications Related to Robot-Assisted Stereoelectroencephalography,« p. 12, veljača 2016..

[13] »OpenRAVE,« 18. ožujka 2013. [Mrežno]. Available: http://openrave.org/. [Pokušaj

pristupa 29. prosinca 2016.].

[14] A. Gasparetto i V. Zanotto, »Toward an optimal performance index for neurosurgical robot's design,« p. 18, 2 listopada 2009.

PRILOZI

I. Kreiranje manipulatora

II. Programski kod

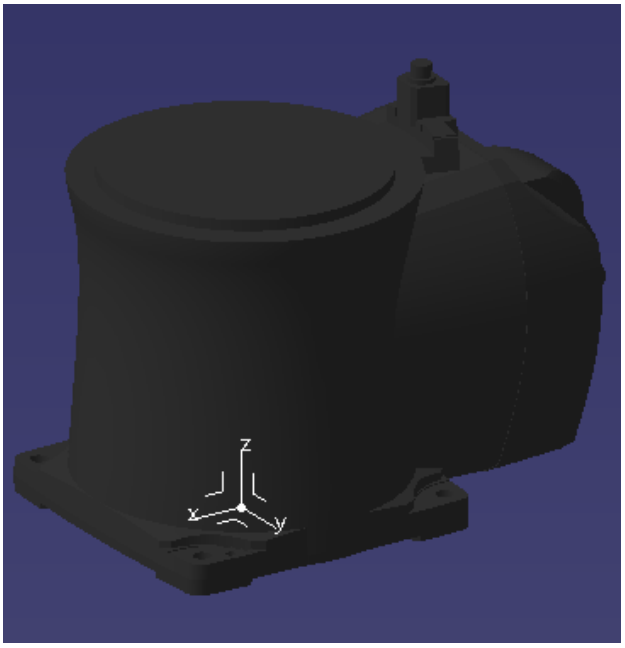
III. Grafovi uspješnosti

IV. CD-R disc

PRILOG I: KREIRANJE MANIPULATORA

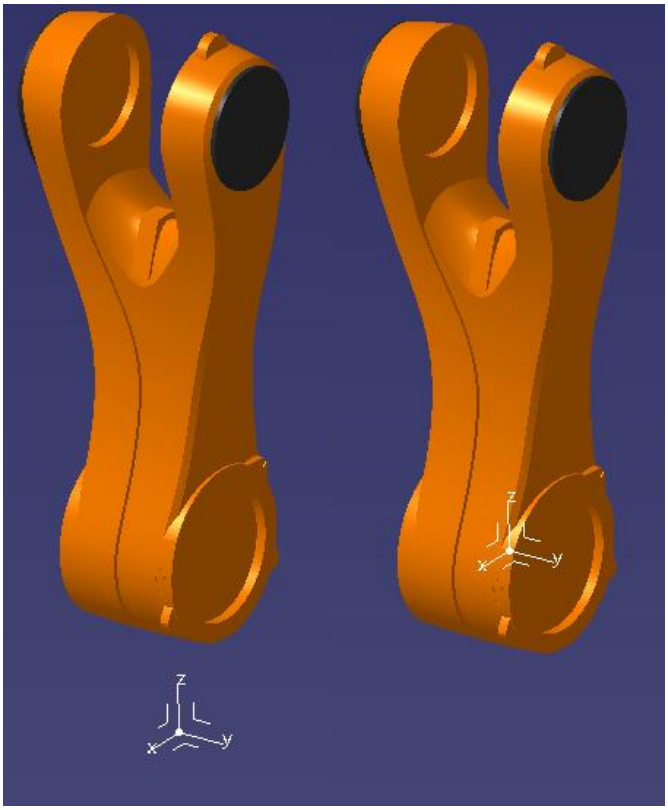
CATIA – konfiguracija koordinatnih sustava

Za sklapanje cjelovitog robota, svaki od njegovih zglobova mora imati pripadajući koordinatni sustav. U slučaju Agilus robota riječ je o šest revolutnih zglobova. Koordinatni sustav svake osi mora se nalaziti u točki rotacije zgloba. Razlog tomu je što OpenRAVE nudi rotaciju zglobova samo oko njihovih koordinatnih sustava. Na primjeru KR6 R900 robota biti će prikazana konfiguracija sustava.



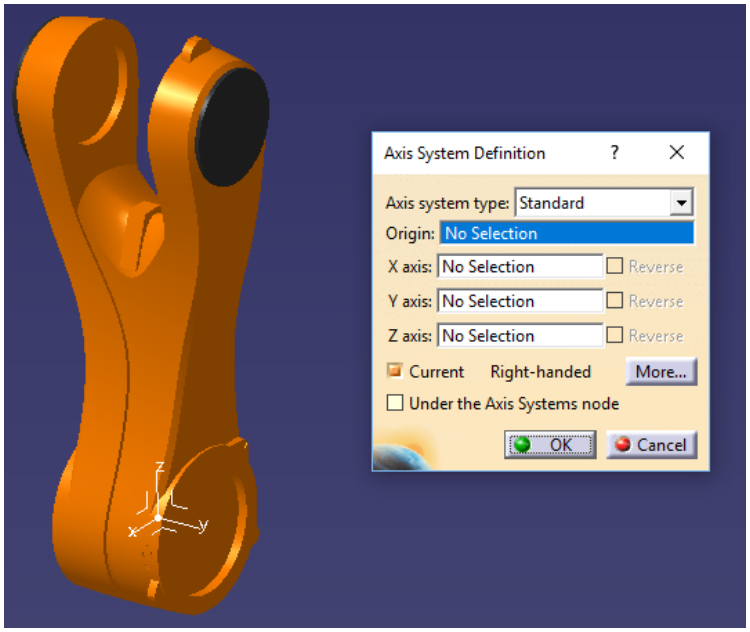
Slika 66 – Baza KR6 R900 robota s pripadajućim koordinatnim sustavom

Slijedeća slika (Slika 67) prikazuje drugu os robota te neispravan i ispravan položaj koordinatnog sustava.



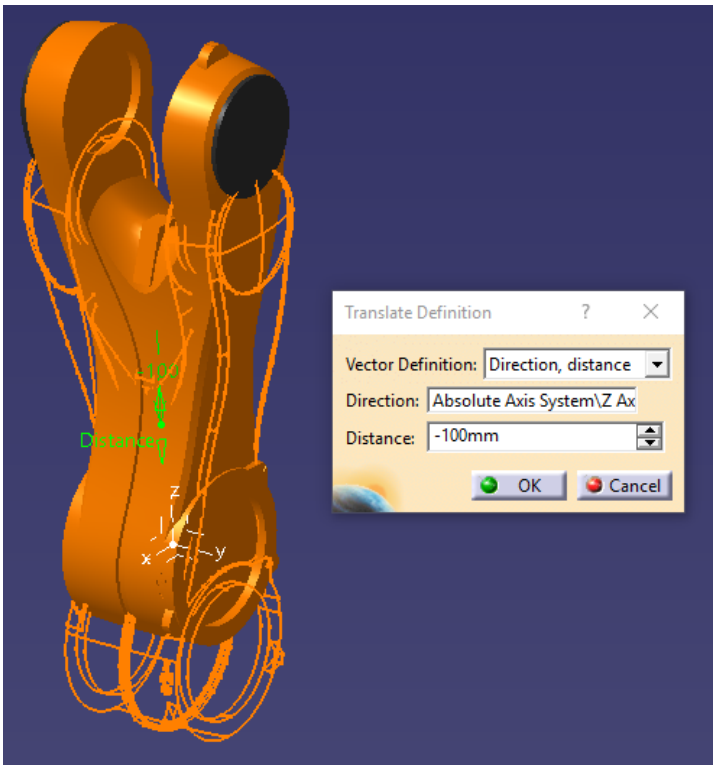
Slika 67 – Druga os - neispravan (lijevo) i ispravan(desno) polozaj koordinatnog sustava

Ponekada nakon otvaranja modela nije prikazan koordinatni sustav, te ga je potrebno dodati. Unutar PART dijela, u alatnoj traci nalazi se kartica **Insert**, te u padajućem izborniku **Axis System**. Klikom na Axis System otvorit će se **Axis System Definition** koji nudi par opcija o tipu sustava, ali je najčešće dovoljno pritisnuti OK i sustav će biti izgeneriran i vidljiv u stablu (Slika 68).



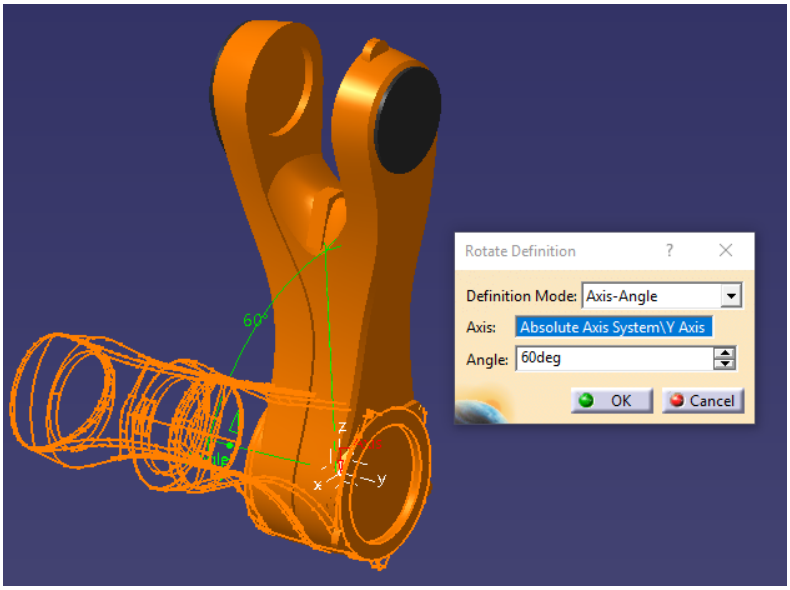
Slika 68 – Dodavanje koordinatnog sustava – Axis System Definition

Ukoliko koordinatni sustav nije odmah na traženom položaju, najčešće se koriste tri metode za njegovo postavljanje. Navedene funkcije mogu se naći pod **Insert → Transformation Features**. Prva od njih je translacija koordinatnog sustava. Nakon odabira **Translation** otvara se prozor **Translate Definition** u kojemu je potrebno odabrati smjer te iznos translacije.



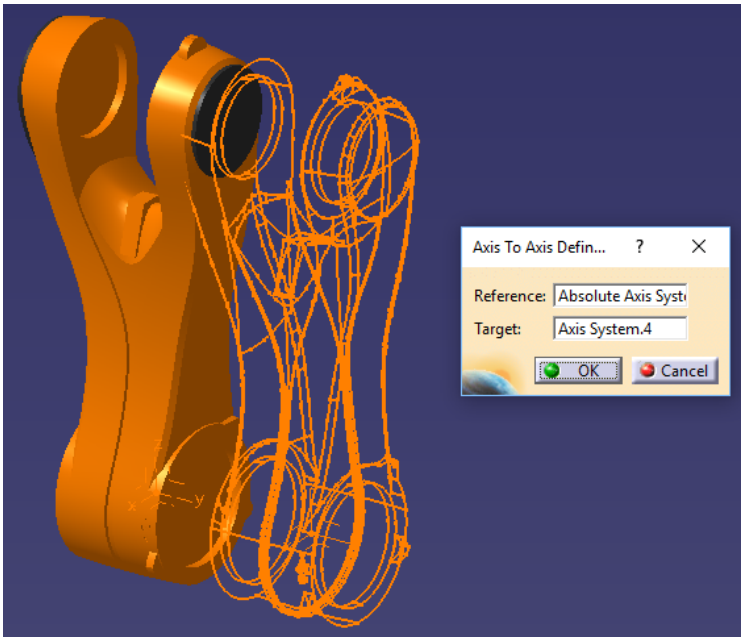
Slika 69 – Translacija koordinatnog sustava u smjeru Z osi

Druga mogućnost je rotirati koordinatni sustav oko željene osi. Odabirom na **Rotation** otvoriti će se prozor **Rotate Definition** u kojemu se odabire os i iznos kuta rotacije u stupnjevima.



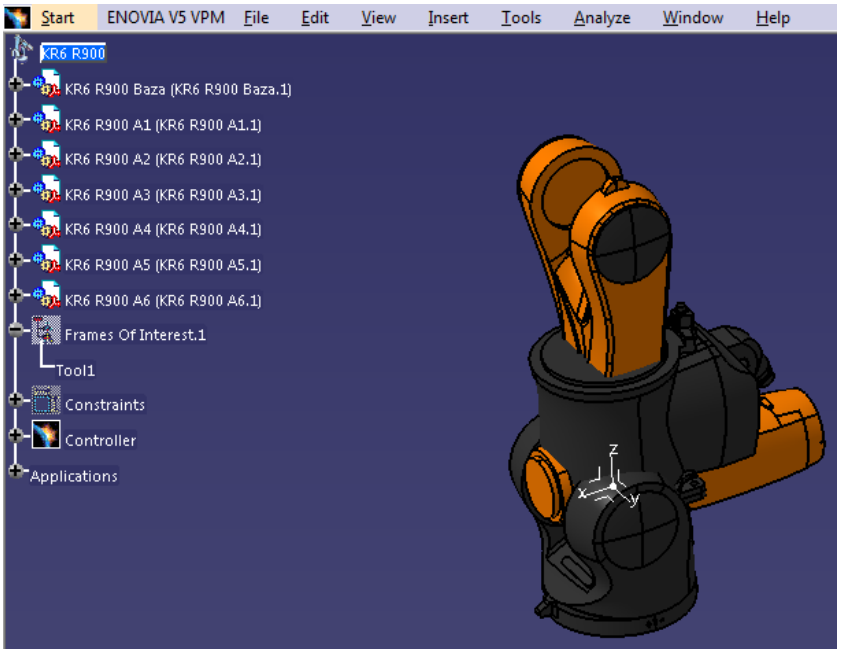
Slika 70 – Rotacija koordinatnog sustava oko Y osi

Posljednja mogućnost zasniva se na stvaranju novog koordinatnog sustava u željenoj točki te transformaciju starog sustava u novi. Naredba koja to omogućuje naziva se **Axis To Axis** koja otvara pripadajući prozor **Axis To Axis Definition**. U otvorenom prozoru jedine mogućnosti su odabrati dva koordinatna sustava, a to je referentni i ciljni sustav.



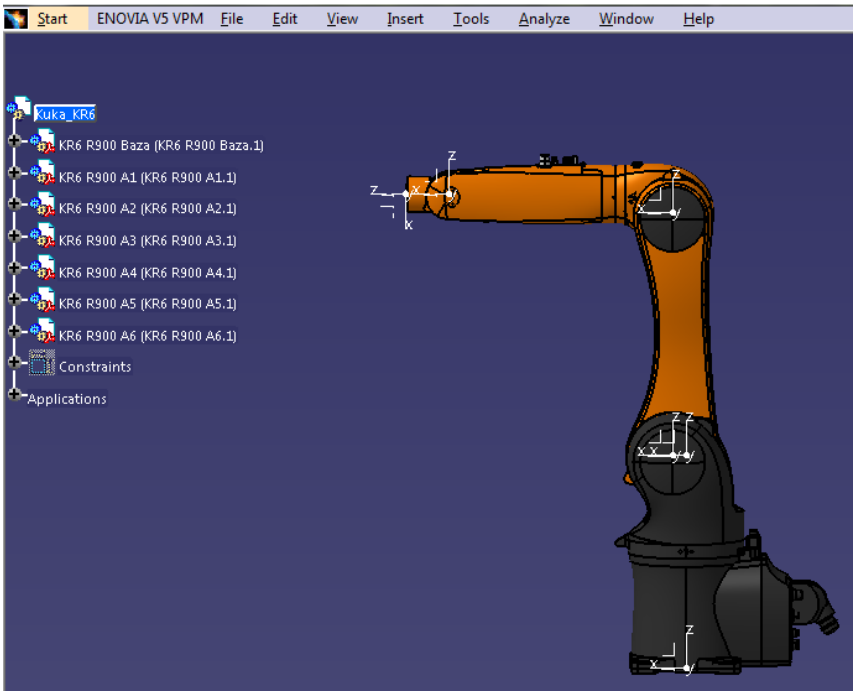
Slika 71 – Axis To Axis transformacija koordinatnog sustava

Podešavanjem svih koordinatnih sustava i pozicioniranja u točke rotacija, potrebno je složiti sklop (Assembly) robota kako bi se vidjele korelacije između pojedinih zglobova. Važno je napomenuti da svi dijelovi (Part) robota trebaju biti spremljeni u .**wrl** format. Kreiranjem sklopa robota dobije se odnos zglobova kao na slijedećoj slici (Slika 72).



Slika 72 – Sklop robota bez definiranih ograničenja

Nakon definiranja ograničenja (Constraints) sklop robota izgleda slijedeće (Slika 73).



Slika 73 - Sklop modela robota KR6 R900

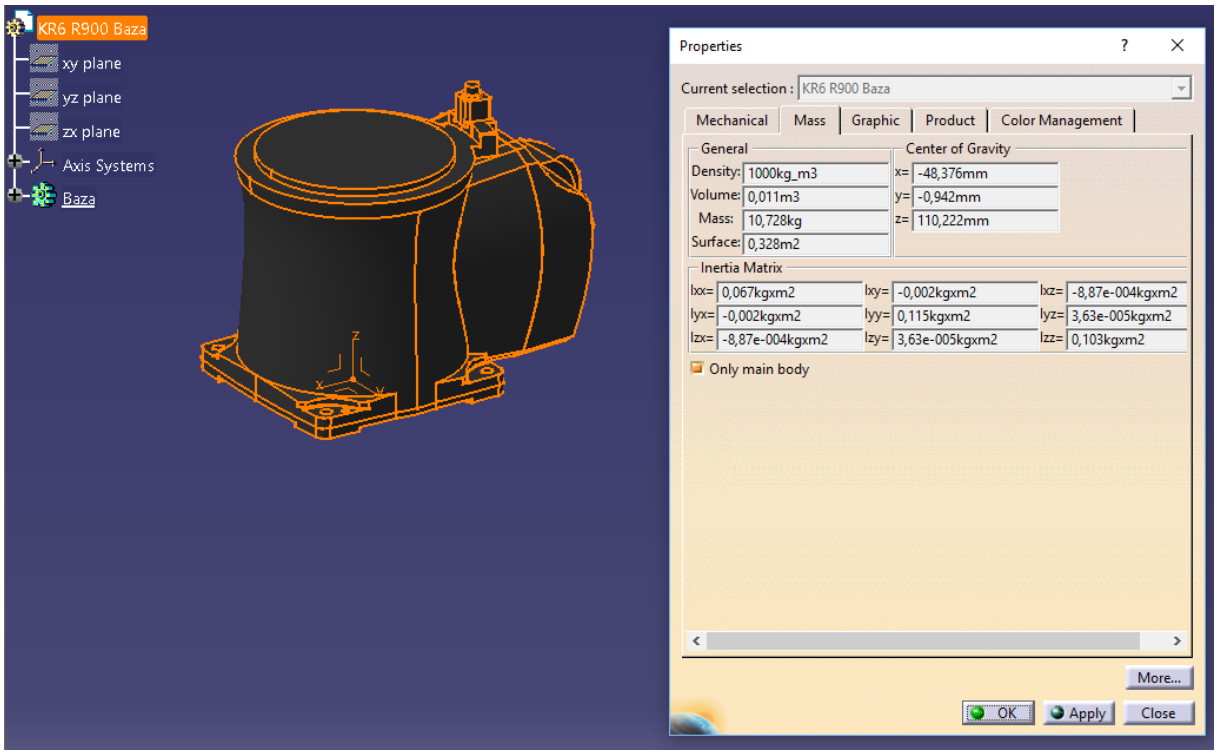
XML – konfiguracija robota i okoline

Konfiguracijom svih CAD dijelova, može se nastaviti s kreiranjem XML datoteka. U njima su sadržani svi dijelovi robota te su definirani zglobovi. Na primjeru KR6 R900 robota biti će prikazan dio koda za kreiranje tijela i zgloba robota, te okoline u kojoj će robot biti smješten. Osnovni dijelovi XML koda pri kreiranju tijela robota su [13]:

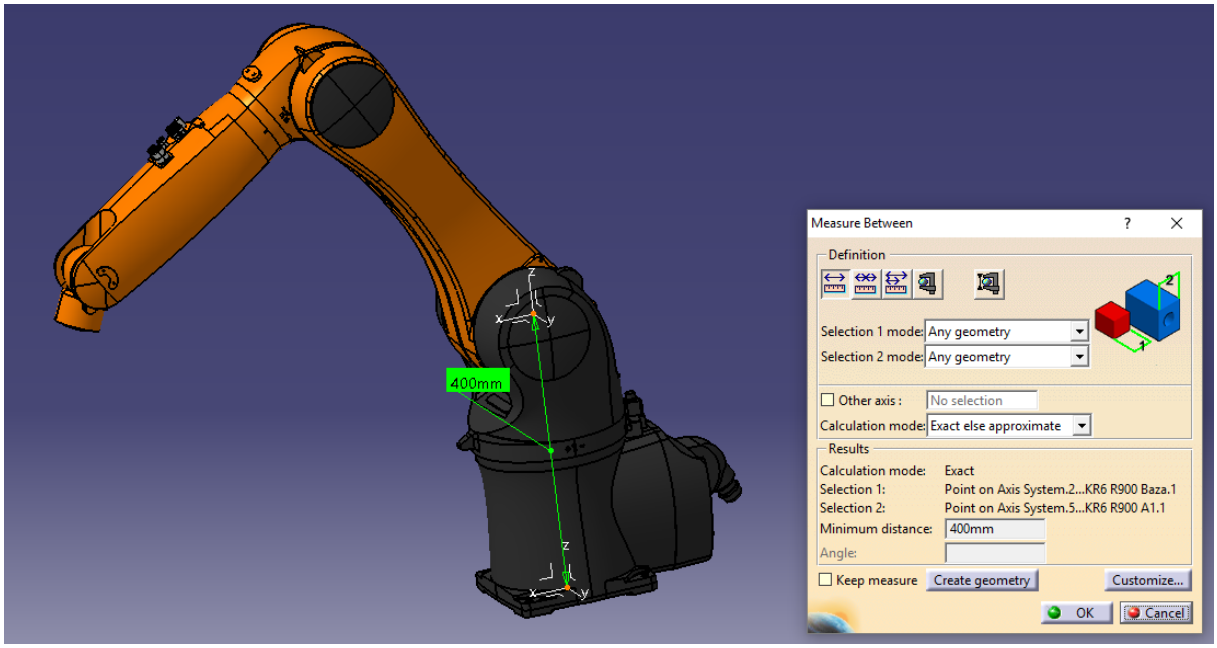
- Ime tijela robota – KinBody name
- Dio tijela robota:
  - Ime tijela - Body name
  - Ubacivanje geometrije (Geom) - Data,Render (.wrl datoteke)
  - Translacija koordinatnog sustava dijela u odnosu na globalni koordinatni sustav OpenRAVEa - $[X\ Y\ Z]$ $[m]$
  - Podaci o masi i težištu (mass) - total, com $[m]$
- Definiranje zgloba:
  - Ime zgloba - Joint name
  - Označavanje dijelova između kojih se definira zglob - Body
  - Dio na koji se odnosi rotacija – offsetfrom
  - Smjer rotacije - $[X\ Y\ Z]$; $[0\ 0\ -1] \rightarrow$ rotacija u negativnu stranu Z osi
  - Ograničenja na rotacije zglobova - limitsdeg (stupnjevi)
  - Ograničenje maksimalne brzine rotacije zgloba - maxveldeg $[°/s]$

```
<KinBody name="Kuka_KR6_R900">

      <Body name="base" type="dynamic">
            <Geom type="trimesh">
                  <Data>fsb/KR6_wrl/KR6_R900_Baza.wrl 1.0</Data>
                  <Render>fsb/KR6_wrl/KR6_R900_Baza.wrl 1.0</Render>
            </Geom>
            <Translation>0 0 0</Translation>
            <mass type="custom">
                  <total>10.728</total>
                  <com>-0.0484 -0.0009 0.1102</com>
            </mass>
      </Body>


      <Body name="axis1" type="dynamic">
```

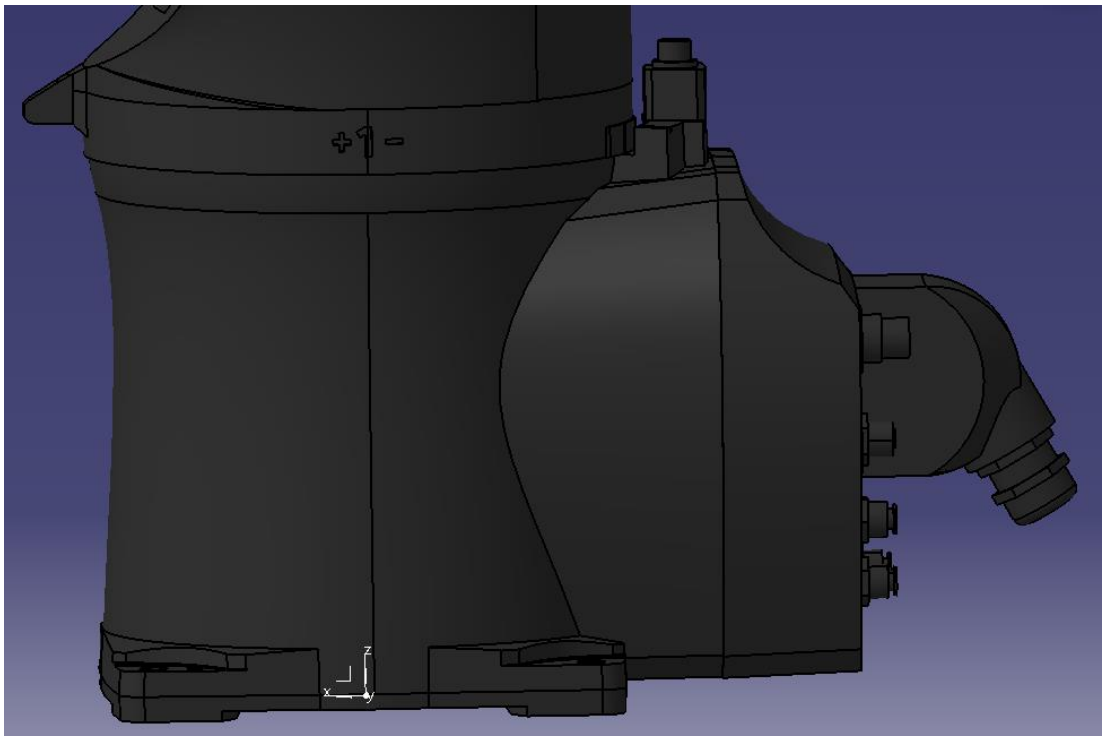
```
            <Geom type="trimesh">
                  <Data>fsb/KR6_wrl/KR6_R900_A1.wrl 1.0</Data>
                  <Render>fsb/KR6_wrl/KR6_R900_A1.wrl 1.0</Render>
            </Geom>
            <Translation>0 0 0.4</Translation>
            <mass type="custom">
                  <total>6.279</total>
                  <com>0.00795 0.0019 0.3327</com>
            </mass>
      </Body>

      <Joint name="J1" type="hinge">
            <Body>base</Body>
            <Body>axis1</Body>
            <offsetfrom>axis1</offsetfrom>
            <axis>0 0 -1</axis>
            <limitsdeg>-170 170</limitsdeg>
            <maxveldeg>270</maxveldeg>
      </Joint>
```



Slika 74 – Podaci o masi, težištu i inerciji u CATIA-i



Slika 75 – Udaljenost između prve osi i baze robota



Slika 76 – Prikaz smjera rotacije prve osi na modelu robota

Nakon što je konfigurirano tijelo robota, potrebno je napisati XML kod koji će sadržavati podatke o robotu potrebne za definiranje prihvatnice :

- Ime robota – Robot name
- Putanju na disku do XML datoteke tijela robota
- Ime manipulatora – Manipulator name:
  - Ime baze – base
  - Ime zvršnog člana – effector
  - Smjer prilaženja prihvatnice objektu – direction $[X\ Y\ Z]$

```
<Robot name="KUKA_KR6_mate">

      <KinBody file="fsb/xml/Kuka_KR6.xml"/>

      <Manipulator name="armf">
            <base>base</base>
            <effector>axis6</effector>
            <direction>0 0 1</direction>
      </Manipulator>

</Robot>
```

Preostaje kreirati okolinu u kojoj će robot djelovati. Značajke koje datoteka okoline sadrži su:

- Mogućnosti promjene boje i pogleda OpenRAVE sučelja
- Putanja do XML datoteke imena robota – Robot file
- Mogućnost translacije i rotacije robota u odnosu na globalni koordinatni sustav OpenRAVEa

```
<Environment>

      <!-- set the background color of the environment-->
      <bkgndcolor>1 1 1</bkgndcolor>
      <!-- set the initial camera translation-->
      <camtrans>1.5 1.5 2</camtrans>
      <!-- set the initial camera rotation specified by rotation-axis-->
      <camrotaxis>0.3 0.8 1.2 170</camrotaxis>

      <Robot file="fsb/xml/KUKA_KR6_mate.xml" name="KUKA_KR6">
            <translation>0 0 0</translation>
            <!--RotationAxis>0 0 1 90</RotationAxis-->
            <RotationAxis>0 0 0 90</RotationAxis>
      </Robot>

</Environment>
```

Na način koji je prikazan, kreiraju se i druga tijela kao što su alati, pribor ili proizvoljni CAD modeli. Osim robota, datoteka okoline može sadržavati i ostala tijela koja će predstavljati okruženje robota. XML datoteka okoline predstavlja zadnji stupanj u kreiranju robotskog sustava, i kasnije se poziva u programskom jeziku, ovom slučaju Python-u. Osim ubacivanja modela u OpenRAVE preko datoteke okoline, modele je moguće ubaciti i direktno u Python kodu, što će biti prikazano kasnije.

Osnovne naredbe u interakciji Pythona i OpenRAVEa

Na početku Python koda potrebno je pozvati naredbu za učitavanje OpenRAVE okoline te aktiviranje scene (eng. viewer). Nakon toga učitava se okolina koja je prethodno kreirana u .xml datoteci. Učitavanjem okoline može se pozvati naredba za definiranje robota

```
env=Environment()
env.SetViewer('qtcoin')
env.Load('C:/.../fsb/xml/KUKA_KR6_R700_lab.xml') #XML okolina

kuka = env.GetRobots()[0] #definiranje robota
```

Osim dodavanja pribora i alata u .xml datoteci okoline, isto je moguće učiniti direktno u Python kodu. Željenu komponentu moguće je postaviti na bilo koje mjesto u okruženju, definirajući transformacijsku matricu u koje će se postaviti njen koordinatni sustav. Ovaj pristup dodavanja u okruženje odvija se kroz tri faze. Prvo se učita adresa .xml datoteke objekta, zatim se definira matrica transformacija na koju će objekt biti postavljen, te se na kraju pozove funkcije za fizičko dodavanje tijela objekta. Na primjeru vodilice biti će prikazano direktno dodavanje objekta u okruženje.

```
point = numpy.array([[-0.70710678, 0., 0.70710678, 0.570711],
                     [0., 1, 0., -0.2],
                     [-0.70710678, 0, -0.70710678, 0.270711],
                     [0., 0., 0., 1.]])

vodilica = env.ReadRobotURI('fsb/xml/vodilica.robot.xml')
vodilica.SetTransform(const.pocetna)
add_tool = env.Add(vodilica)
```

Dva su najčešća načina upravljanja manipulatorom u OpenRAVE-u. Upravljanje pomoću unutarnjih koordinata, odnosno kontrola kuta zakreta svakog pojedinog zgloba robot,

jednostavniji je način upravljanja, ali je dovodođenje u željenu poziciju teže. Upravljanje unutarnjim koordinatama moguće je na dva načina. Prvi način je kontrola samo određenim zglobovima, a drugi način je upravljanje svim zglobovima manipulatora. Ukoliko se upravlja svim zglobovima, naredba za pokretanje sadrži vektor kojemu je duljina jednaka broju stupnjeva slobode robota. Svaka vrijednost odnosi se na pojedini zglob i unosi se u radijanima.

```
kuka = env.GetRobots()[0] #definiranje robota
kuka.SetDOFValues([0.5,2.58,-0.6],[0,1,2]) #zakret samo odabranih zglobova

kukaruka = interfaces.BaseManipulation(kuka)
kukaruka.MoveManipulator(goal = [0,-math.pi/2,math.pi/2,0,math.pi/2,0])
#vektor koji sadrži podatak o rotaciji svakog zgloba
```

Upravljanje pomoću vanjskih koordinata, temelji se na zadavanju matrice transformacija cilja u kojeg se želi poslati posljedna os robota, odnosno prihvatnica. Ukoliko robot koristi alat, što je najčešći slučaj, cilj je upravljati vrhom alata. Bitna napomena kod upravljanja vanjskim koordinatama je da prije primjene treba generirati inverzni kinematički model. Njegovo rješavanje omogućuje izračunavanje unutarnjih koordinata robota na temelju geometrije robota, te ga je potrebno izračunati samo jednom u kodu.

```
kuka = env.GetRobots()[0] #definiranje robota
ikmodelkuka =
databases.inversekinematics.InverseKinematicsModel(kuka,iktype=IkParameteri
zation.Type.Transform6D)
if not ikmodelkuka.load():
    ikmodelkuka.autogenerate() #solver za rješavanje inverzne kinematike
```

Kada je generiran inverzni kinematički model, može se upravljati robotom pomoću vanjskih koordinata.

```
kuka = env.GetRobots()[0] #definiranje robota
kukaruka = interfaces.BaseManipulation(kuka)
kukaruka.MoveToHandPosition(matrices=[vodilicacoord],seedik=5) #slanje
robota u mjesto prihvata alata
kuka.WaitForController(0) #stabilnost programa – čekanje da robot izvrši
kretnju
```

Kada je robot došao do pozicije prihvata alata, vrlo jednostavnim funkcijama predmet se može prihvatiti, a isto tako i ispustiti.

```
kuka.Grab(vodilica) #prihvat alata
kuka.Release(vodilica) #ispuštanje alata
```

PRILOG 2: PROGRAMSKI KOD

Python Main – glavni Python program

```
from openravepy import *
import math, numpy, time
from numpy.linalg import inv
from numpy import linalg as LA
import csv
import copy
import time
import method
import const

#---- postavljanje globalnih varijabli --
env=Environment()
env.SetViewer('qtcoin')

c = 1
if c == 0:
    env.Load('C:/Program Files (x86)/OpenRAVE-0.8.2/share/openrave-
0.8/fsb/xml/KUKA_KR6_R700_lab.xml')
    robot_name = '700'
elif c == 1:
    env.Load('C:/Program Files (x86)/OpenRAVE-0.8.2/share/openrave-
0.8/fsb/xml/KUKA_KR6_lab.xml')
    robot_name = '900'
else:
    env.Load('C:/Program Files (x86)/OpenRAVE-0.8.2/share/openrave-
0.8/fsb/xml/KUKA_KR10_R1100_lab.xml')
    robot_name = '1100'

t = 0
if t == 0:
    vodilica = env.ReadRobotURI('fsb/xml/Koren_vodilica.robot.xml')
    tool_name = 0
    TCP = numpy.array([185, 0, 122.55])
elif t == 1:
    vodilica = env.ReadRobotURI('fsb/xml/vodilica.robot.xml')
    tool_name = 1
    TCP = numpy.array([74.8, 0, 143])

pacijent = 0 #ukljucivanje pacijenta u prostor
if pacijent == 1:
    init_pacijent =
env.ReadRobotURI('fsb/xml/Krevet_s_pacijentom.robot.xml')
    Pacijent = method.init_pacijent(env,init_pacijent)

else:
    pass

########### INICIJALIZACIJA VODILICE ##########
Tool = method.init_tool(env, vodilica)

kuka = method.init_kuka(20, env)
kukaruka = interfaces.BaseManipulation(kuka)
```

```
manip = kuka.GetActiveManipulator()
traj = kukaruka.MoveManipulator(const.goal)

kuka.WaitForController(0)

ikmodelkuka =
databases.inversekinematics.InverseKinematicsModel(kuka,iktype=IkParameteri
zation.Type.Transform6D)
if not ikmodelkuka.load():
    ikmodelkuka.autogenerate()

#kuka uzima vodilicu
vodilicacoord = vodilica.GetTransform()
vodilicacoord[2, 3] += 0.00001

kukaruka.MoveToHandPosition(matrices=[vodilicacoord],seedik=5)
kuka.WaitForController(0)
kuka.Grab(vodilica)
kuka.WaitForController(0)

traj = kukaruka.MoveManipulator(const.goal,
outputtrajobj=True,execute=True)
kuka.WaitForController(0)

vodilica_coord = vodilica.GetLinks()[1].GetTransform()
mht = kuka.GetLinks()[6].GetTransform()
toolmatrix = numpy.dot(inv(mht), vodilica_coord)
print'======='

########### INICIJALIZACIJA SFERE ##########
sfera = method.init_sfera(env)

########### DEFINIRANJE KUTA VERTIKALNE ROTACIJE ###########
step_alfa = method.define_step_alfa();

########### DEFINIRANJE KUTA HORIZONTALNE ROTACIJE ###########
step_beta = method.define_step_beta();

limit = 0; #pocetni broj limit tocaka

for k in range(0, const.z_podjela + 1):#petlja za gibanje odsjecka po Z osi
    dz = k * const.dz_step / float(1000)

    for m in range(0, const.y_podjela + 1): #petlja za gibanje po Y osi
        dy = m * const.dx_step / float(1000)

        for z in range (0, const.x_podjela + 1): #gibanje odsjecka po X osi
            dx = z * const.dy_step / float(1000)

            sfera_pos = numpy.array([[1, 0, 0, dx + const.rxOffset],
                         [0, 0, -1, dy + const.ryOffset],
                         [0, 1, 0, dz + const.rzOffset],
                         [0., 0., 0., 1.]])

            pocetna_pacijent = numpy.array([[0,1, 0., dx + const.rxOffset],
                                 [-1, 0, 0, dy + const.ryOffset + 1.205],
                                 [0, 0, 1, dz + const.rzOffset - 0.1],
                                 [0., 0., 0., 1.]]) #polozaj pacijenta
```

```
            xOffset = sfera_pos[0,3];
            yOffset = sfera_pos[1,3];
            zOffset = sfera_pos[2,3];

            sfera.SetTransform(sfera_pos) #inicijalizacija sfere s pomakom
ishodista

            if pacijent == 1:
                init_pacijent.SetTransform(pocetna_pacijent)
#inicijalizacija pacijenta s pomakom ishodista sfere
            else:
                pass

            #Crtanje surface-a
            surface = (env.drawtrimesh
(points=numpy.array(((const.rxOffset,const.ryOffset,zOffset),
(const.rxOffset + const.x_direction_m,const.ryOffset,zOffset),
(const.rxOffset,const.ryOffset+const.y_direction_m,zOffset),
(const.rxOffset+const.x_direction_m,const.ryOffset+const.y_direction_m,zOff
set))),indices=numpy.array(((0,1,2),(2,1,3))),
colors=numpy.array((1,0,0,0.5))))

            #Crtanje ishodista sfere
            center_plot = (env.plot3(points=numpy.array
                          (((xOffset,yOffset,zOffset),
                          (xOffset,yOffset,zOffset))),
                          pointsize=0.015,
                          colors=numpy.array(((0,0,0),(0,1,0))),
                          drawstyle=1))

            #Crtanje ishodista sfere
            center_plot = (env.plot3(points=numpy.array
                          (((xOffset,yOffset,zOffset),
                          (xOffset,yOffset,zOffset))),
                          pointsize=0.015,
                          colors=numpy.array(((0,0,0),(0,1,0))),
                          drawstyle=1))

            print('Pozicija sfere u prostoru: %f;%f;%f' % (sfera_pos[0,3],
sfera_pos[1,3], sfera_pos[2,3]))

            for n in range(0, const.broj_H_rotacija + 1): #petlja za
kretanje robota po horizontalnoj ravnini [beta]

                angle_V = (n*step_beta + const.beta_2)*(math.pi/180)

                sphere_center = [xOffset, yOffset, zOffset]

                for i in range(0, const.broj_V_rotacija + 1): #petlja za
kretanje robota po visini odsjecka [alfa]

                    start = time.time()

                    angle = (i*step_alfa + const.alfa_2)*(math.pi/180)
#horizontalno gibanje
```

```
# ------ Funkcija za transformaciju u sferni koordinatni sustav ------
                    [X,Y,Z] = method.sph2cart(angle_V, angle, const.R_m);

                    # --- Rotacija sustava oko X osi ---
                    [X,Y,Z] = numpy.dot(const.z_trans, [X,Y,Z]);

                    vodilicacoord_tool_1 =
vodilica.GetLinks()[1].GetTransform()
                    vodilicacoord_tool_1[0,3] = X + xOffset  #tocka na
obodu promjenjiva po x
                    vodilicacoord_tool_1[1,3] = Y + yOffset #tocka na obodu
promjenjiva po y
                    vodilicacoord_tool_1[2,3] = Z + zOffset #tocka na obodu
promjenjiva po z

                    # --- Kreiranje tocke za linearni pomak ---
                    [Xc,Yc,Zc]= method.sph2cart(angle_V,angle,const.R_m_L);
                    [Xc,Yc,Zc] = numpy.dot(const.z_trans, [Xc,Yc,Zc]);

                    smjer = numpy.zeros(3)
                    smjer[0] = Xc - X
                    smjer[1] = Yc - Y
                    smjer[2] = Zc - Z

                    # ------ Kreiranje matrice za orijentaciju alata------
                    tool_orientation_matrix =
numpy.dot(method.create_orientationmatrix_for_tool(kuka,
vodilicacoord_tool_1, sfera_pos), inv(toolmatrix))

                    # ------ Trazenje mogucih putanja do cilja ------
                    ikparam =
IkParameterization(tool_orientation_matrix,IkParameterization.Type.Transfor
m6D)
                    solutions =
ikmodelkuka.manip.FindIKSolutions(ikparam,IkFilterOptions.CheckEnvCollision
s)

                    theta_deg = 0 #ako robot moze doci u tocku bez rotacije
oko osi alata, theta = 0

                    # --- Ispitivanje najkrace putanje ---
                    shortest_move =
method.find_linear_trajectory(solutions, theta_deg, kukaruka, kuka,smjer,
sphere_center, vodilica)

                    # --- Rotacijsko gibanje ---
                    while shortest_move == False:
                        for theta_deg in range(const.theta_start,
const.theta_stop, const.theta_step):
                            theta_rad = math.radians(theta_deg)
                            rotation =
numpy.array(method.create_orientationmatrix_for_tool(kuka,
vodilicacoord_tool_1, sfera_pos) * method.find_rotation_matrix(theta_rad))
                            rotation_move =
numpy.dot(rotation,inv(toolmatrix))
                        # ------ Trazenje mogucih putanja do cilja ------
                            ikparam =
IkParameterization(rotation_move,IkParameterization.Type.Transform6D)
```

```
                            solutions =
ikmodelkuka.manip.FindIKSolutions(ikparam,IkFilterOptions.CheckEnvCollision
s)

                            shortest_move =
method.find_linear_trajectory(solutions, theta_deg, kukaruka, kuka,smjer,
sphere_center, vodilica)

                            if shortest_move != False:
                                break

                    if theta_deg == const.theta_stop - const.theta_step:
                        theta_deg = 0
                        shortest_move = False
                        while shortest_move == False:
                            for theta_deg in range(const.theta_start,
const.theta_stop, const.theta_step):
                                theta_rad = math.radians(theta_deg)
                                rotation =
numpy.array(method.create_orientationmatrix_for_tool(kuka,
vodilicacoord_tool_1, sfera_pos) * method.find_rotation_matrix(theta_rad))
                                rotation_move =
numpy.dot(rotation,inv(toolmatrix))
                            # ------ Trazenje mogucih putanja do cilja ----
--
                                ikparam =
IkParameterization(rotation_move,IkParameterization.Type.Transform6D)
                                solutions =
ikmodelkuka.manip.FindIKSolutions(ikparam,IkFilterOptions.CheckEnvCollision
s)

                                shortest_move =
method.find_shortest_trajectory(solutions, theta_deg, kukaruka, kuka)

                                if shortest_move != False:
                                    break

                    if theta_deg == const.theta_stop - const.theta_step:
                        limit = limit + 1
                        position_sphere = [0,0,0,0,0,0]
                        position_linear = position_sphere
                        vodilicacoord_tool_sphere =
[vodilicacoord_tool_1[0,3],vodilicacoord_tool_1[1,3],
vodilicacoord_tool_1[2,3]]
                        vodilicacoord_tool_linear =
vodilicacoord_tool_sphere
                        theta_deg = 0
                        q = 0
                        w = q
                        theta_vector = vodilicacoord_tool_sphere
                        end = time.time()
                        elapsed_time = end - start
                        method.writing_CSV_file(position_sphere,
position_linear, vodilicacoord_tool_sphere, vodilicacoord_tool_linear,
theta_deg, sphere_center, angle, angle_V, tool_name, robot_name, TCP, q, w,
theta_vector, elapsed_time)
```

```
                    else:
                        position_sphere = kuka.GetDOFValues() #kutovi
rotacija zglobova robota uz sferu

                        vodilicacoord_tool_1 =
vodilica.GetLinks()[1].GetTransform()
                        vodilicacoord_tool_sphere =
vodilicacoord_tool_1[0:3,3]

                        #Ispitivanje uspjesnosti dolaska na cilj
                        q = 1
                        goal_sphere =
math.sqrt((vodilicacoord_tool_sphere[0]-sphere_center[0])**2 +
(vodilicacoord_tool_sphere[1]-sphere_center[1])**2 +
(vodilicacoord_tool_sphere[2]-sphere_center[2])**2)
                        if goal_sphere <= const.R_m - 0.01 :
                            q = 0
                        # --- Linearno gibanje ---
                        linearno_gibanje = method.linear_move(smjer,
kukaruka, kuka)

                        position_linear = kuka.GetDOFValues() #kutovi
rotacija zglobova robota s linearnimn pomakom

                        vodilicacoord_tool_2 =
vodilica.GetLinks()[1].GetTransform()
                        vodilicacoord_tool_linear =
vodilicacoord_tool_2[0:3,3]

                        theta_vector =
method.theta_vector(vodilicacoord_tool_1, vodilicacoord_tool_2)

                        #Ispitivanje uspjesnosti dolaska na cilj
                        w = 1
                        goal_linear =
math.sqrt((vodilicacoord_tool_linear[0]-sphere_center[0])**2 +
(vodilicacoord_tool_linear[1]-sphere_center[1])**2 +
(vodilicacoord_tool_linear[2]-sphere_center[2])**2)
                        if goal_linear <= const.R_m_L - 0.01:
                            w = 0

                        print('%f;%f;%f; kut beta: %f; kut alfa: %f' %
(vodilicacoord_tool_2[0,3], vodilicacoord_tool_2[1,3],
vodilicacoord_tool_2[2,3], math.degrees(angle_V), math.degrees(angle)))

                        #racunanje proteklog vremena trajektorije
                        end = time.time()
                        elapsed_time = end - start

                        method.writing_CSV_file(position_sphere,
position_linear, vodilicacoord_tool_sphere, vodilicacoord_tool_linear,
theta_deg, sphere_center, angle, angle_V, tool_name, robot_name, TCP, q, w,
theta_vector, elapsed_time)

print 'kraj'
```

Python method – funkcije koje se koriste i pozivaju u glavnom kodu

```
import copy
import math, numpy, time
from numpy.linalg import inv
from numpy import linalg as LA
import csv
import const
import collections

#Inicijalizacija kreveta s pacijentom
def init_pacijent(env, init_pacijent):
    init_pacijent.SetTransform(const.pocetna_pacijent)
    add_pacijent = env.Add(init_pacijent)
    return add_pacijent

#Inicijalizacija alata
def init_tool(env, vodilica):
    vodilica.SetTransform(const.pocetna)
    add_tool = env.Add(vodilica)
    return add_tool

#Inicijalizacija kuke
def init_kuka(brzinakuka, env):
    kuka = env.GetRobots()[0]
    spidur = kuka.GetDOFMaxVel()
    spidur = spidur / 100 * brzinakuka
    kuka.SetDOFVelocityLimits(spidur)
    return kuka

#Inicijalizacija sfere
def init_sfera(env):
    sfera_pos = numpy.array([[1, 0, 0, const.rxOffset],
                             [0, 0, -1, const.ryOffset],
                             [0, 1, 0, const.rzOffset],
                             [0., 0., 0., 1.]])

    sfera=env.ReadRobotURI('fsb/xml/sfera.robot.xml')
    sfera.SetTransform(sfera_pos)
    return sfera

#Definiranje koraka alfa kuta
def define_step_alfa():
    alfa_sum = abs(const.alfa_1)+ abs(const.alfa_2);
    return alfa_sum/const.broj_V_rotacija;

#Definiranje koraka beta kuta
def define_step_beta():
    beta_sum = abs(const.beta_1) + abs(const.beta_2);
    return beta_sum/const.broj_H_rotacija;

#Kreiranje sfernog koordinatnog sustava
def sph2cart(az, el, r):
    rcos_theta = r * math.cos(el)
    x = rcos_theta * math.cos(az)
    y = rcos_theta * math.sin(az)
```

```
    z = r * math.sin(el)
    return x, y, z

#Kreiranje orijentacijske matrice
def create_orientationmatrix_for_tool(kuka, vodilica_coord, sfera_pos):
    x1 = vodilica_coord[0:3,3]
    y1 = sfera_pos[0:3,3]
    z1 = copy.copy(x1)
    z1[2] += 0.2
    vector_x1y1 = y1 - x1
    vector_x1z1 = z1 - x1
    T1 = x1
    x1y1 = vector_x1y1/LA.norm(vector_x1y1)
    x1z1 = vector_x1z1/LA.norm(vector_x1z1)
    y1 = numpy.cross(x1z1,x1y1)/(LA.norm(numpy.cross(x1z1,x1y1)))
    z1 = numpy.cross(x1y1,y1)
    R_M1 = numpy.eye(4)
    R_M1[0:3,0] = x1y1
    R_M1[0:3,1] = y1
    R_M1[0:3,2] = z1
    R_M1[0:3,3] = T1
    return R_M1

#Trazenje i pomicanje najkracom trajektorijom linearnog pomaka
def find_linear_trajectory(solutions, theta_deg, kukaruka, kuka, smjer,
sphere_center, vodilica):
    # --- Ispitivanje najkrace putanje ---
    minimum = numpy.array([])
    d=math.sqrt(smjer[0]**2+smjer[1]**2+smjer[2]**2)
    stepsize=0.001
    maxsteps=d/stepsize

    for x in range(0, len(solutions)):
        traj =
kukaruka.MoveManipulator(goal=[solutions[x,0],solutions[x,1],solutions[x,2]
,solutions[x,3],solutions[x,4],solutions[x,5]],outputtrajobj=True,execute=F
alse)
        kuka.WaitForController(0)
        vrijeme = traj.GetDuration()
        minimum=numpy.append(minimum,vrijeme)

    if theta_deg == const.theta_stop - const.theta_step:
        print 'Limit kuta theta'
        return True

    if len(solutions) == 0:
        print 'Planning_Error'
        return False

    else:
        minN = minimum.tolist().index(min(minimum)) #trazenje minimuma
unutar vektora - pretvorba array u listu

cilj=[solutions[minN,0],solutions[minN,1],solutions[minN,2],solutions[minN,
3],solutions[minN,4],solutions[minN,5]]
        traj = kukaruka.MoveManipulator(goal =
cilj,outputtrajobj=True,execute=True)
        kuka.WaitForController(0)
```

```
kukaruka.MoveHandStraight(direction=smjer,stepsize=stepsize,minsteps=1,maxs
teps=maxsteps, execute = True)
        kuka.WaitForController(0)

        vodilicacoord_tool_2 = vodilica.GetLinks()[1].GetTransform()
        vodilicacoord_tool_linear = vodilicacoord_tool_2[0:3,3]

        goal_linear = math.sqrt((vodilicacoord_tool_linear[0]-
sphere_center[0])**2 + (vodilicacoord_tool_linear[1]-sphere_center[1])**2 +
(vodilicacoord_tool_linear[2]-sphere_center[2])**2)

        if goal_linear <= const.R_m_L - 0.01 and theta_deg ==
const.theta_stop - 2*const.theta_step:
            print 'nema boljeg pokusaja'
            return False

        if goal_linear <= const.R_m_L - 0.01:
            print 'neuspjeh'
            return False

        else:
            traj = kukaruka.MoveManipulator(goal =
cilj,outputtrajobj=True,execute=True)
            kuka.WaitForController(0)
            return True

#Definiranje linearnog gibanja
def linear_move(smjer, kukaruka, kuka):
    d=math.sqrt(smjer[0]**2+smjer[1]**2+smjer[2]**2)
    stepsize=0.001
    maxsteps=d/stepsize
    kukaruka.MoveHandStraight(direction=smjer, stepsize=stepsize,
minsteps=1,maxsteps=maxsteps)
    kuka.WaitForController(0)

def find_rotation_matrix(theta_rad):
    R = numpy.matrix([[1, 0, 0, 0],[0, math.cos(theta_rad),
math.sin(theta_rad), 0], [0, -math.sin(theta_rad), math.cos(theta_rad), 0],
[0, 0, 0, 1]])
    return R

def theta_vector(vodilicacoord_tool_1, vodilicacoord_tool_2):
    T = numpy.matrix([[1, 0, 0, 0],[0, 1, 0, 0],[0,0,1,0.015],[0,0,0,1]])
    Translate = numpy.dot(vodilicacoord_tool_2, T) #ovisno o tome je li
gibanje prema unutra ili prema van, stavlja se vodilicacoord_tool_1 ili 2
    return Translate[0:3,3]

def writing_CSV_file(position_sphere, position_linear,
vodilicacoord_tool_sphere, vodilicacoord_tool_linear, theta_deg,
sphere_center, angle, angle_V, tool_name, robot_name, TCP, q, w,
theta_vector, elapsed_time):
    deg = 180/math.pi

    const.position_sphere = numpy.append(const.position_sphere,
position_sphere)
    const.position_linear = numpy.append(const.position_linear,
position_linear)
```

```
    const.theta_value = numpy.append(const.theta_value, theta_deg)
    const.sphere_center = numpy.append(const.sphere_center, sphere_center )
    const.beta = numpy.append(const.beta, angle)
    const.alfa = numpy.append(const.alfa, angle_V)
    const.vodilicacoord_tool_sphere = numpy.append
(const.vodilicacoord_tool_sphere, vodilicacoord_tool_sphere)
    const.vodilicacoord_tool_linear = numpy.append
(const.vodilicacoord_tool_linear, vodilicacoord_tool_linear)
    const.q = numpy.append(const.q, q)
    const.w = numpy.append(const.w, w)
    const.Translate_Theta= numpy.append(const.Translate_Theta,theta_vector)
    const.elapsed_time = numpy.append(const.elapsed_time, elapsed_time)

    open_file = open('Robot_Points.csv',"wb")
    writer = csv.writer(open_file)
    x = 0
    y = 6
    z = 0
    p = 0

    for j in range(0,len(const.position_sphere)/6):

        data = writer.writerow((j+1,
robot_name,tool_name,TCP[0],TCP[1], TCP[2],
const.position_sphere[x]*deg, const.position_sphere[x+1]*deg,
const.position_sphere[x+2]*deg, const.position_sphere[x+3]*deg,
const.position_sphere[x+4]*deg, const.position_sphere[x+5]*deg,
const.vodilicacoord_tool_sphere[p], const.vodilicacoord_tool_sphere[p+1],
const.vodilicacoord_tool_sphere[p+2], const.q[z],
const.position_linear[x]*deg, const.position_linear[x+1]*deg,
const.position_linear[x+2]*deg, const.position_linear[x+3]*deg,
const.position_linear[x+4]*deg, const.position_linear[x+5]*deg,
const.vodilicacoord_tool_linear[p], const.vodilicacoord_tool_linear[p+1],
const.vodilicacoord_tool_linear[p+2], const.w[z], const.theta_value[z],
const.sphere_center[p], const.sphere_center[p+1], const.sphere_center[p+2],
const.beta[z]*deg, const.broj_H_rotacija , const.alfa[z]*deg ,
const.broj_V_rotacija, const.linear_move_distance,
const.Translate_Theta[p], const.Translate_Theta[p+1],
const.Translate_Theta[p+2],const.x_podjela+1, const.y_podjela+1,
const.z_podjela+1, const.elapsed_time[z]) )

        x = x + 6
        y = y + 6
        z = z + 1
        p = p + 3

    open_file.close()
```

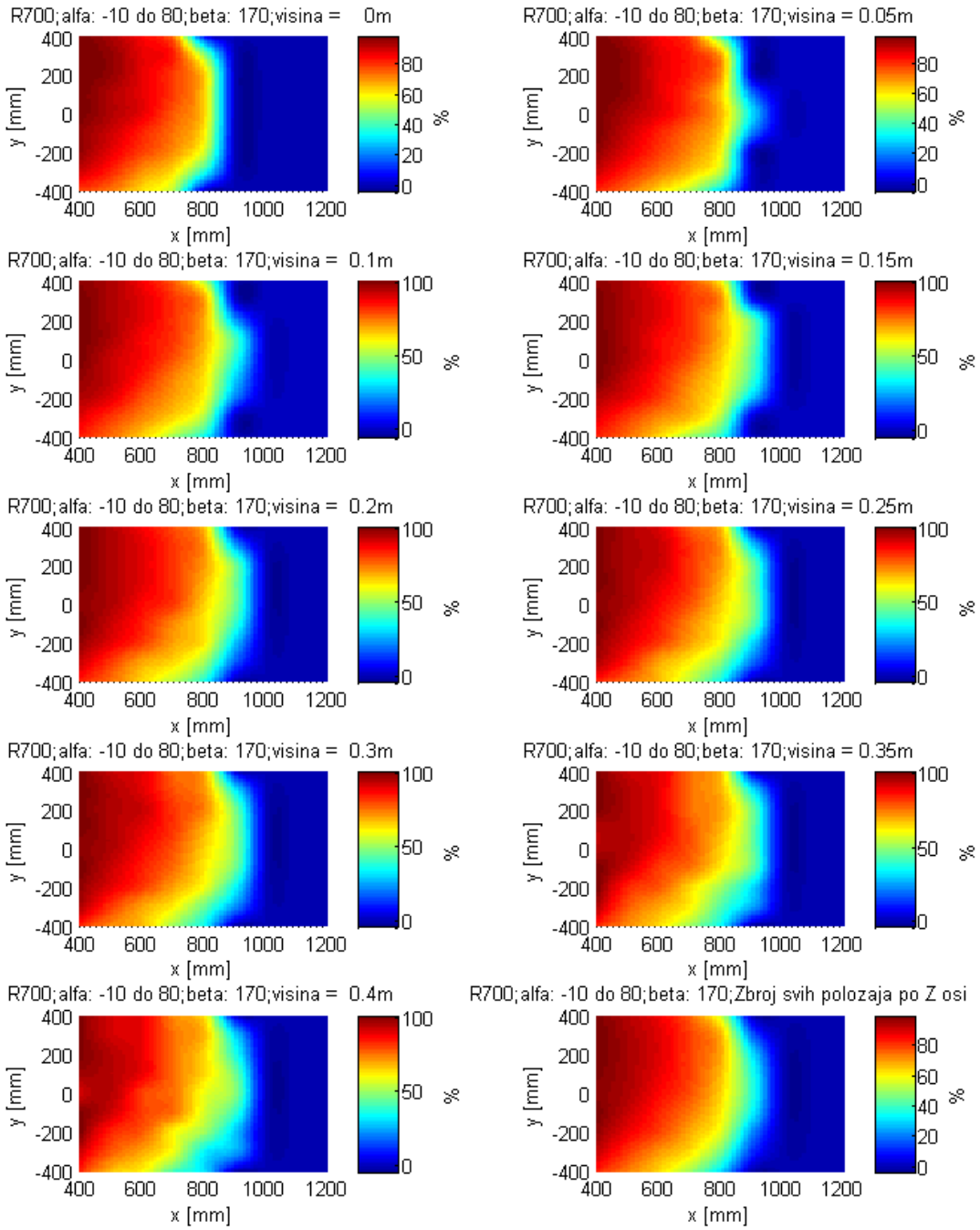
Python const – konstante koje se koriste i pozivaju u glavnom kodu

```
import math, numpy, time
from numpy.linalg import inv
from numpy import linalg as LA

point = numpy.array([[0.70710678, 0., 0.70710678, 0.470711],
                     [0., 1, 0., 0],
                     [-0.70710678, 0, 0.70710678, 0.370711],
                     [0., 0., 0., 1.]])

pocetna = numpy.array([[-1., 0, 0., 0.55],
                     [0, 1, 0, -0.15],
                     [0, 0, -1, 0.5],
                     [0., 0., 0., 1.]])

direction = numpy.array([[1, 0, 0., 0.5],
                     [0, 0, -1, 0.2],
                     [0, 1, 0, 0.4],
                     [0., 0., 0., 1.]])

y_trans = numpy.array([[1, 0, 0],
                     [0, math.cos(math.pi/2), -math.sin(math.pi/2)],
                    [0, math.sin(math.pi/2), math.cos(math.pi/2)]]);

z_trans = numpy.array([[math.cos(math.pi/2), math.sin(math.pi/2), 0],
                     [-math.sin(math.pi/2), math.cos(math.pi/2), 0],
                    [0, 0, 1]]);

goal = [0, -math.pi/2, math.pi/2, 0, math.pi/2, 0]
goal_2 = [0, -math.pi/2, math.pi/2, 0, math.pi/2, math.pi]

#konstante za kut alfa
alfa_1 = 80; #pocetni kut vertikalne rotacije
alfa_2 = -10; #zavrsni kut vertikalne rotacije

#konstante za kut beta
beta_1 = 85; #pocetni kut horizontalne rotacije
beta_2 = -beta_1;

#konstante za vertikalnu i horizontalnu rotaciju
broj_V_rotacija = 5;
broj_H_rotacija = 5;

#konstante za sferu
rxOffset = 0.4;
ryOffset = -0.4;
rzOffset = 0.2;

#pocetna tocka polozaja pacijenta
pocetna_pacijent = numpy.array([[0, 1, 0., rxOffset],
                     [-1, 0, 0, ryOffset+1.205],
                     [0, 0, 1, rzOffset-0.1],
                     [0., 0., 0., 1.]])
```

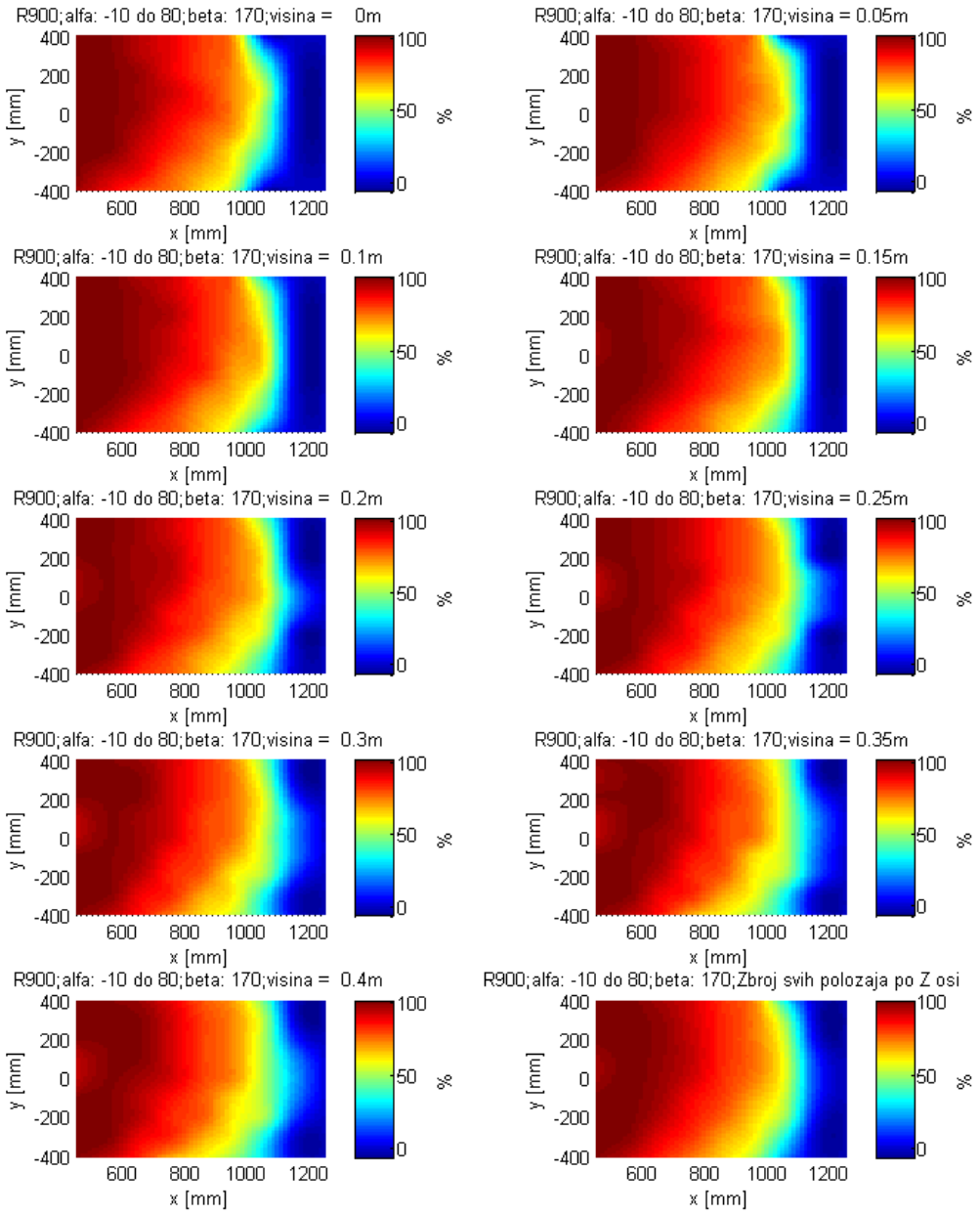
```
#konstante za kocku po kojoj se robot giba
x_direction = 800 #pomak u x smjeru [mm]
x_direction_m = x_direction / float(1000); #pomak u x smjeru [m]
x_podjela = 8
dx_step = x_direction/x_podjela

y_direction = 800 #pomak u y smjeru [mm]
y_direction_m = y_direction / float(1000) #pomak u y smjeru [m]
y_podjela = 8
dy_step = y_direction/y_podjela

z_direction = 400 #pomak u z smjeru [mm] (400)
z_direction_m = z_direction / float(1000) #pomak u z smjeru [m]
z_podjela = 1 #(8)
dz_step = z_direction/z_podjela

#konstante koje se odnose na valjak
maximum = 1; #odredjivanja koraka (360/max = stupnjevi rotacije)
rmax = 1; #odredjivanje koraka rotacije po obodu valjka
valjak_radius = 100; #radius valjka u [mm]
radius_valjak_m = valjak_radius/float(1000);
visina_valjka = 100; #visina valjka u [mm]
visina_valjka_m = visina_valjka/float(1000); #visina valjka u [m]
step_dx = visina_valjka_m/maximum;

#konstanta za radius odsjecka
R = 1; #radius odsjecka[mm]
R_m = R/float(1000); #radius odsjecka[m]

#konstanta za radius odsjecka - linearno gibanje
R_m_l = 80
R_m_L = R_m_l/float(1000) #radius koji odredjuje linearno gibanje (vrijedi:
R_m_L > R_m ili R_m_L < R_m)

linear_move_distance = R_m_l - R

#pocetni, zavrsni kut i korak zakretanja oko osi alata
theta_start = -5
theta_stop = -360
theta_step = -5

#konstante potrebne za kreiranje .csv datoteke
space_number = 30
position_sphere = numpy.array([])
position_linear = numpy.array([])
theta_value = numpy.array([])
sphere_center = numpy.array([])
cartesian_coord = numpy.array([])
beta = numpy.array([])
alfa = numpy.array([])
vodilicacoord_tool_sphere = numpy.array([])
vodilicacoord_tool_linear = numpy.array([])
q = numpy.array([])
w = numpy.array([])
Translate_Theta = numpy.array([])
limit = numpy.array([])
elapsed_time = numpy.array([])
```

PRILOG 3: GRAFOVI USPJEŠNOSTI

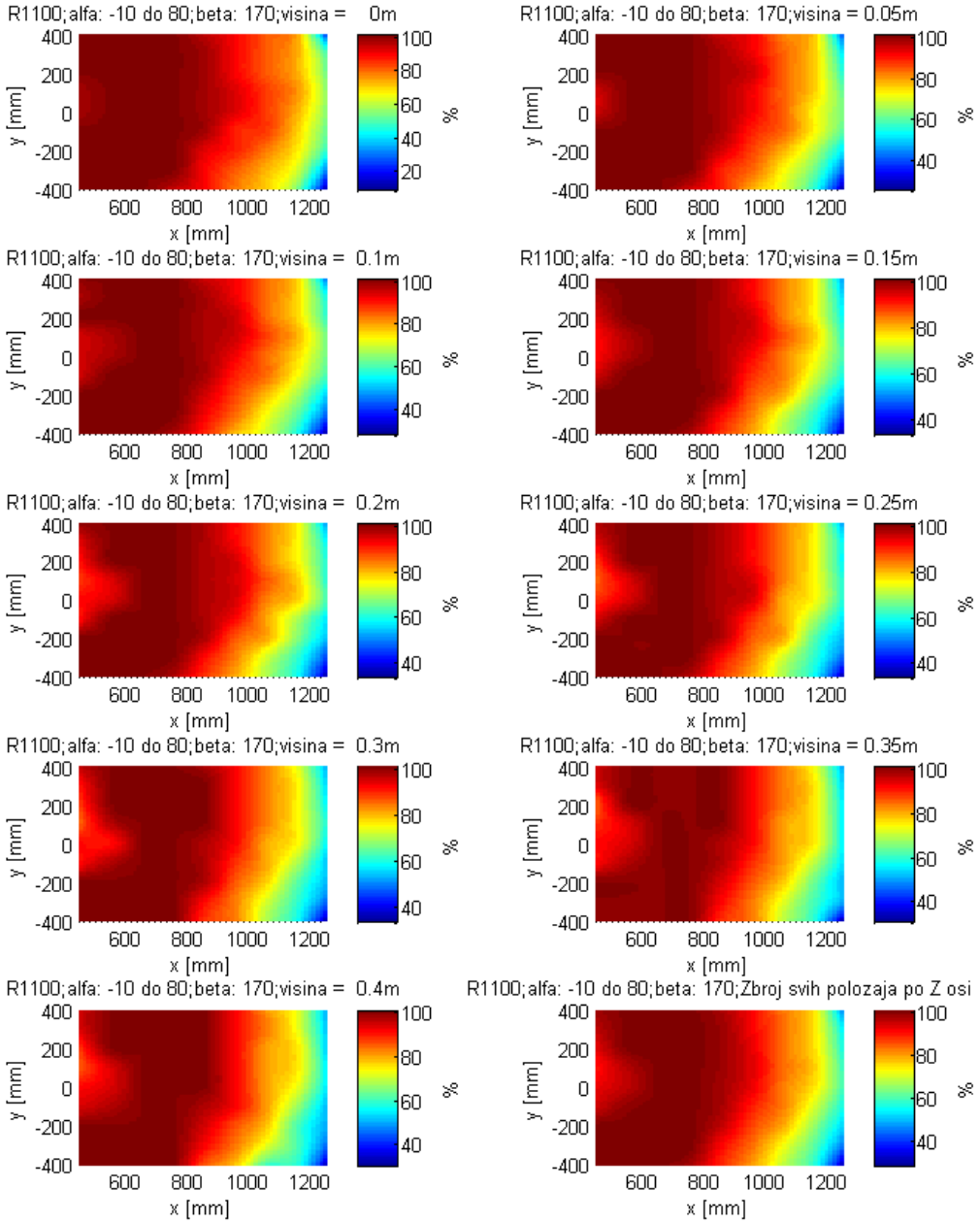
Simulacija a)



Slika 77 – Uspješnost robota R700 na svim visinama (simulacija a)

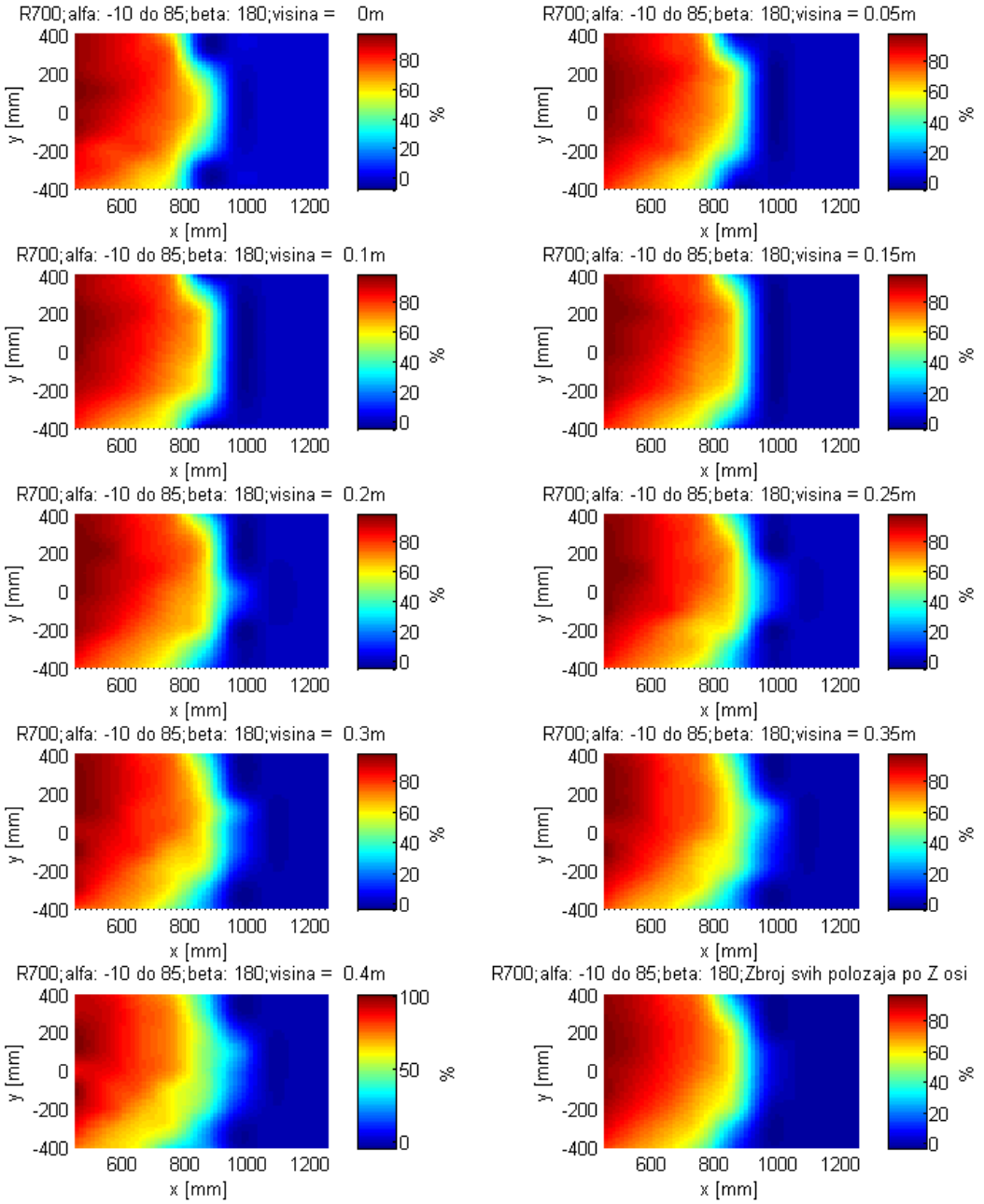


Slika 78 - Uspješnost robota R900 na svim visinama (simulacija a)

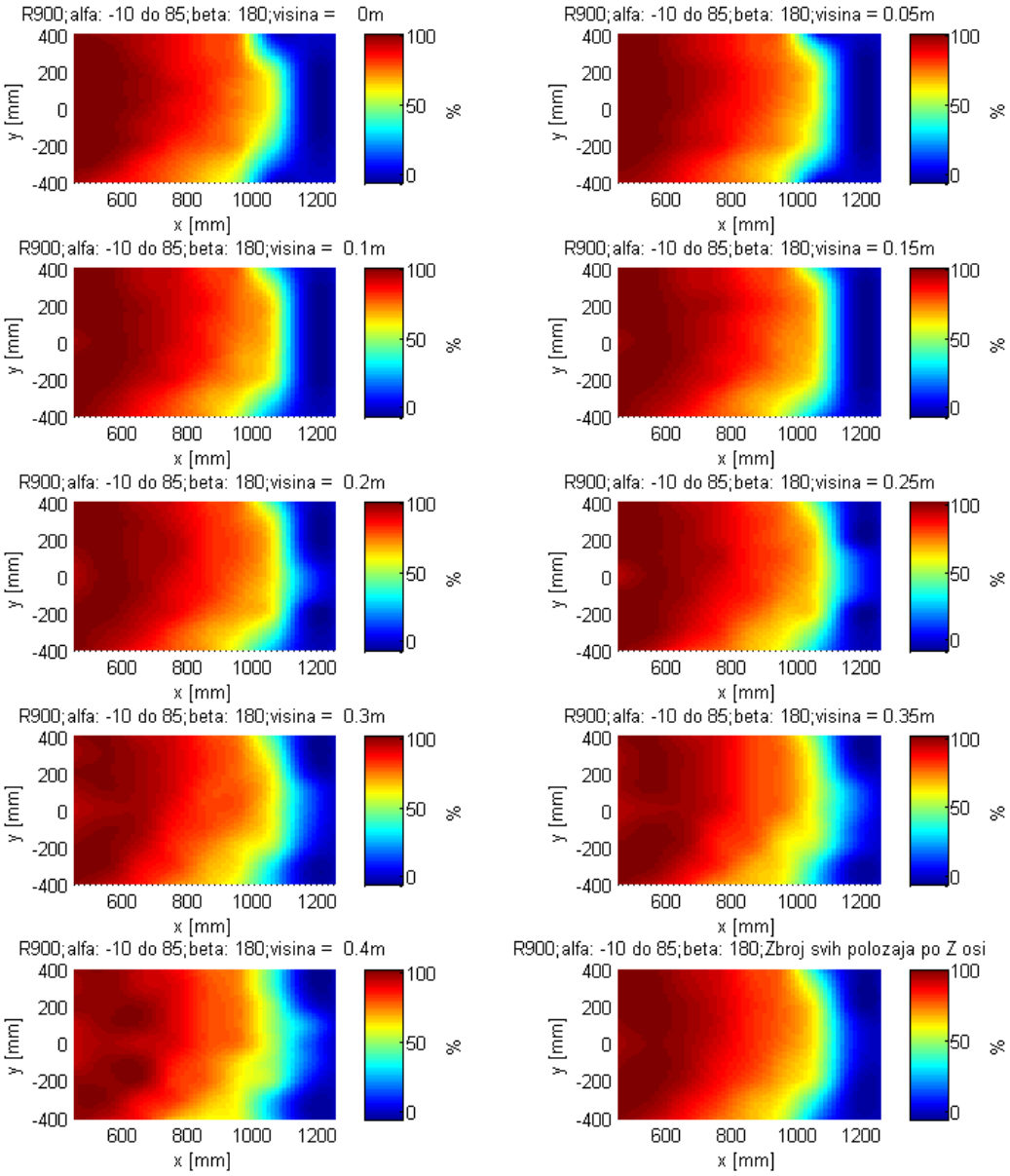


Slika 79 - Uspješnost robota R1100 na svim visinama (simulacija a)

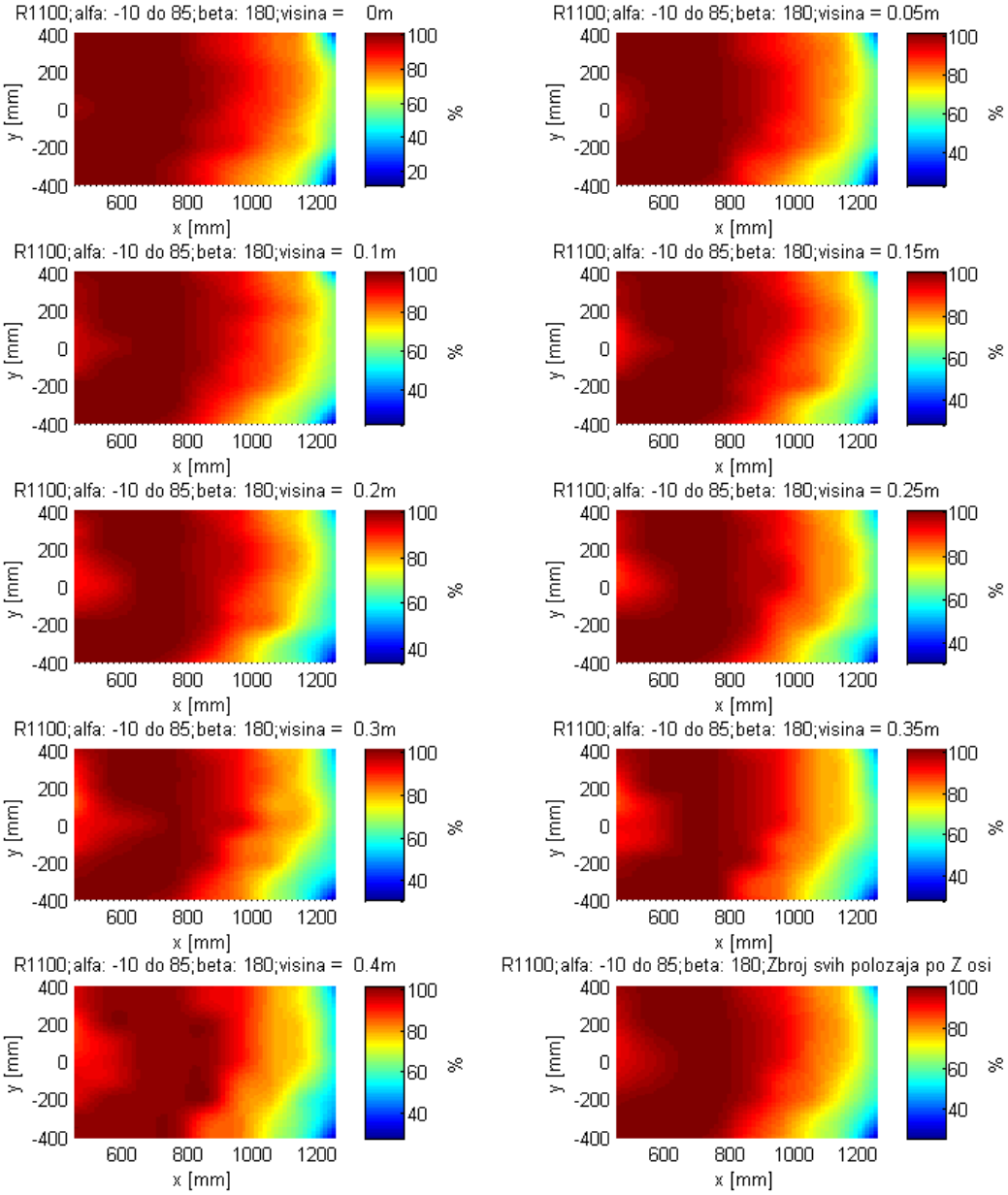
Simulacija b)



Slika 80 - Uspješnost robota R700 na svim visinama (simulacija b)

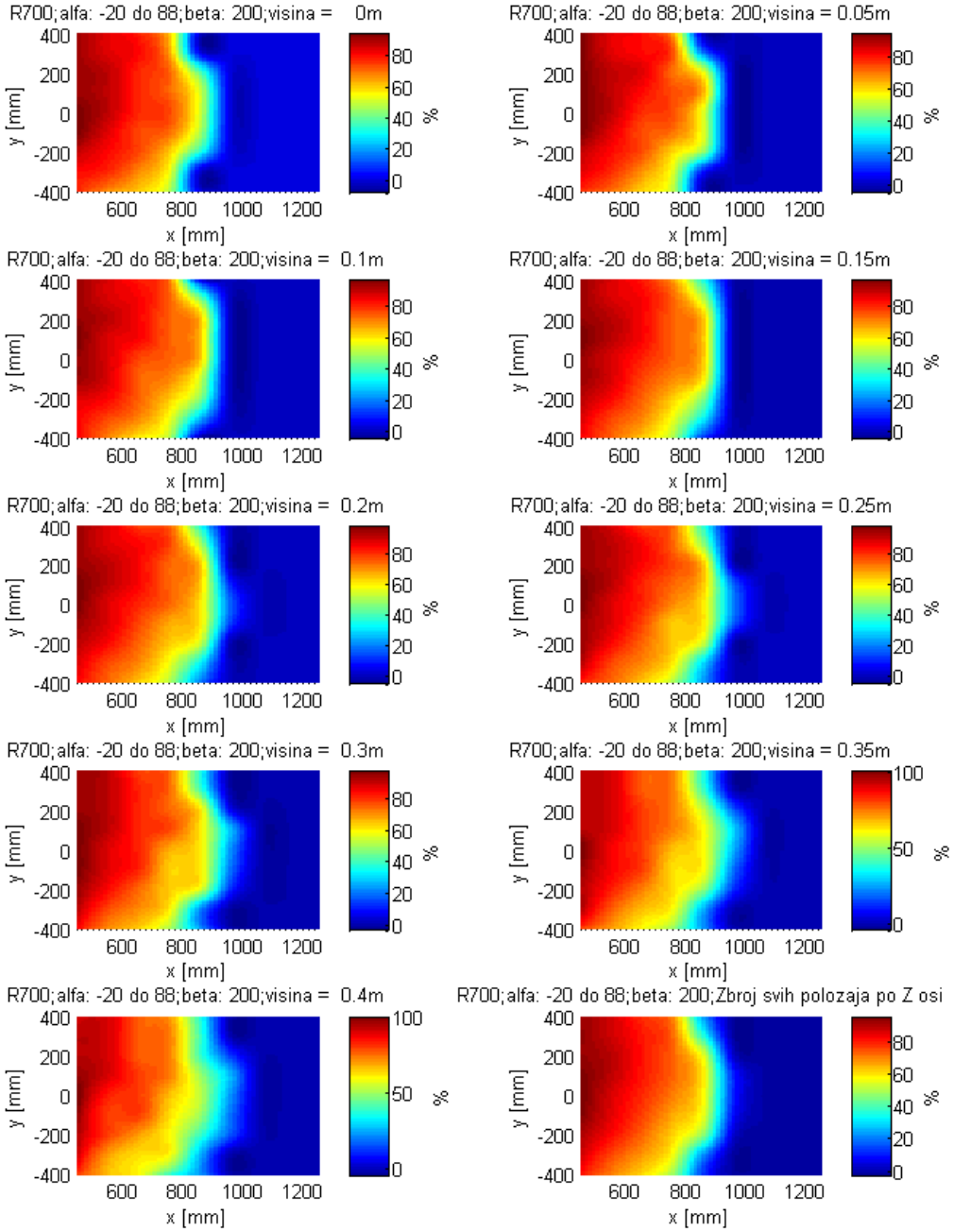


Slika 81 - Uspješnost robota R900 na svim visinama (simulacija b)

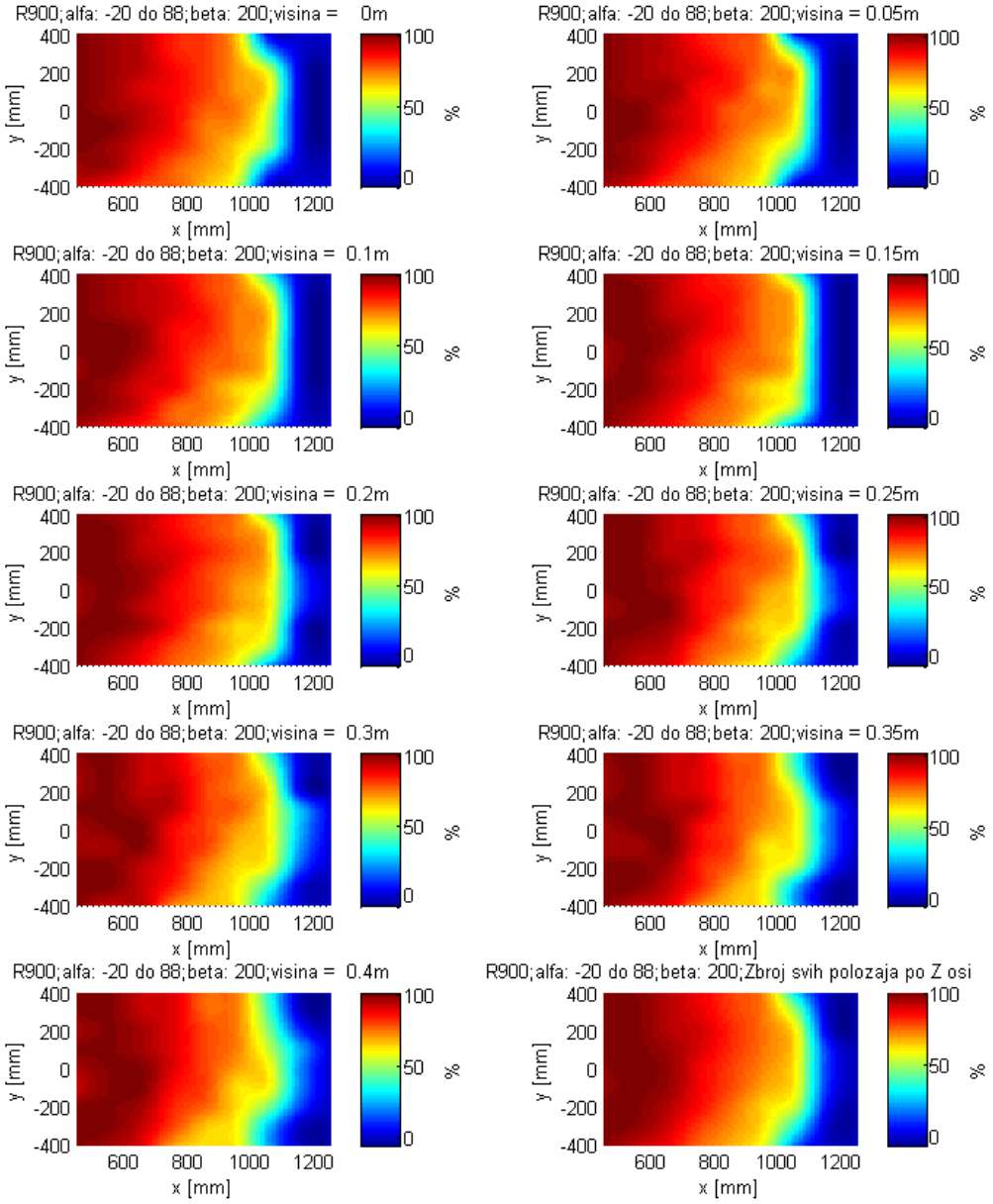


Slika 82 - Uspješnost robota R1100 na svim visinama (simulacija b)

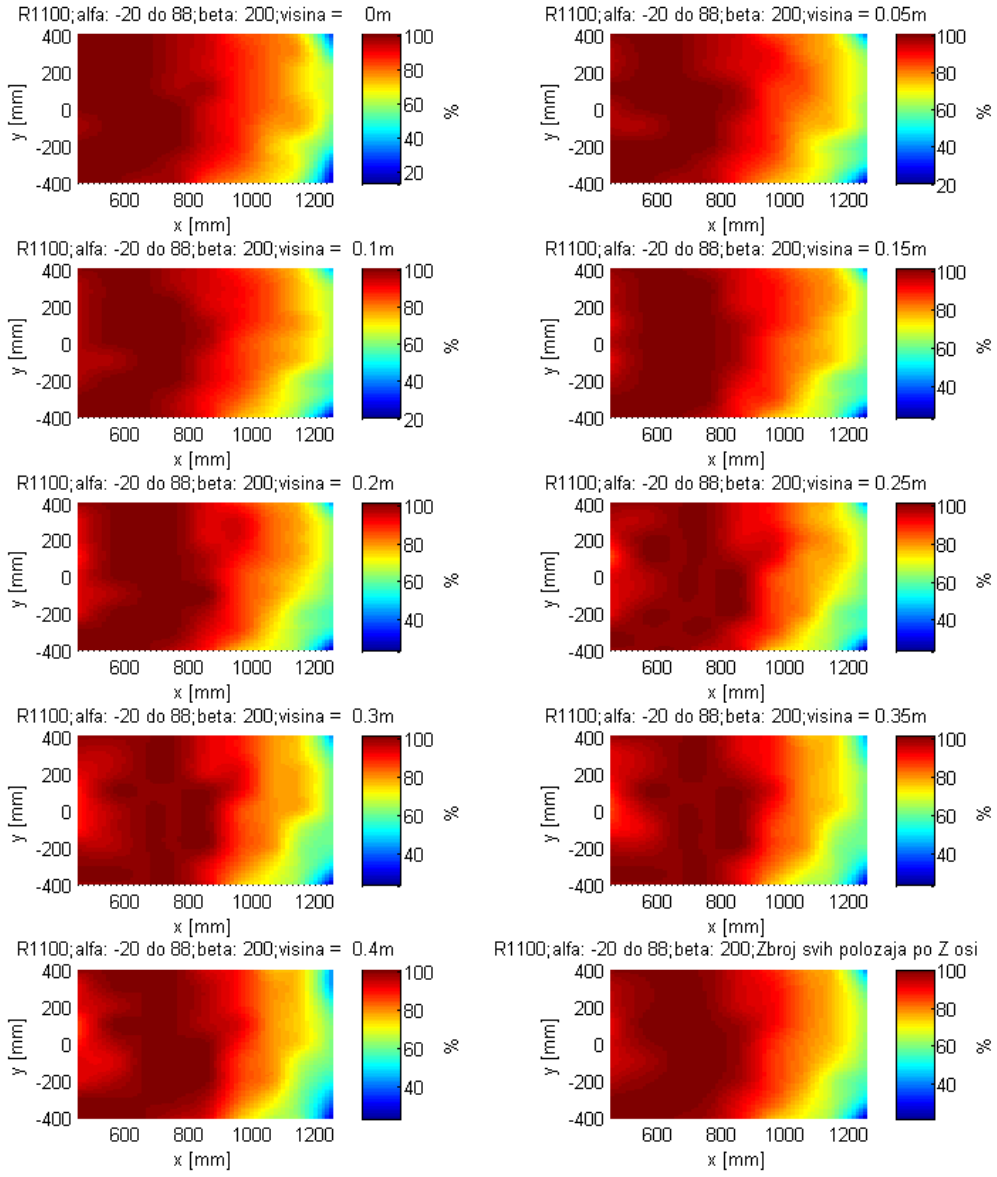
Simulacija c)



Slika 83 - Uspješnost robota R700 na svim visinama (simulacija c)



Slika 84 - Uspješnost robota R900 na svim visinama (simulacija c)



Slika 85 - Uspješnost robota R1100 na svim visinama (simulacija c)