Digitalizacija robotskog radnog prostora u simulacijskom softveru pomoću in-hand 3D kamere

Piršić, Antonio

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Antonio Piršić

Zagreb, 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE

ZAVRŠNI RAD

Mentori:

Doc. dr. sc. Marko Švaco, dipl. ing.
Dr. sc. Filip Šuligoj, dipl. ing.

Student:

Antonio Piršić

Zagreb, 2023.

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

Zahvaljujem se mentorima doc.dr.sc. Marku Švaci i dr. sc. Filipu Šuligoju na pomoći te svim svim savjetima i opremi koju su mi ustupili prilikom izrade ovog završnog rada.

Zahvaljujem se i roditeljima Mariu i Đurđi te djevojci Amandi na podršci tijekom studiranja i pisanja završnog rada.

Antonio Piršić

SVEUČILIŠTE U ZAGREBU
FAKULTET STROJARSTVA I BRODOGRADNJE
Središnje povjerenstvo za završne i diplomske ispite
Povjerenstvo za završne i 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: **602 – 04 / 23 – 6 / 1**	
Ur.broj: **15 - 1703 - 23 -**	

ZAVRŠNI ZADATAK

Student: **Antonio Pirišić** JMBAG: **0035216236**

Naslov rada na hrvatskom jeziku: **Digitalizacija robotskog radnog prostora u simulacijskom softveru pomoću in-hand 3D kamere**

Naslov rada na engleskom jeziku: **Digitization of the robotic workspace in a simulation software using an in-hand 3D camera**

Opis zadatka:

Simulacijski robotski softvera za planiranje gibanja i raznih trajektorija je koristan alat u procesu programiranja robotskih sustava. U načelu, glavni izazov off-line i on-line programiranja robota se svodi na nemogućnost poklapanja idealiziranog i konstruiranog modela s izvedbom u realnom svijetu. Kako bi se što praktičnije mogle koristiti razne funkcije simulacijskog softvera, predlaže se digitalizacija oblaka točaka radnog prostora robota primjenom 3D kamere za potrebu planiranja trajektorija u simulacijskom softveru.

Zadatak uključuje:

- Pripremu radnog okruženja Universal robot UR5e s in-hand kalibriranom 3D kamerom (Realsense 435d).
- Programiranje Python skripte koja omogućava dohvaćanje oblaka točaka i vizualizaciju istog unutar simulacijskog softvera RoboDK na temelju trenutne pozicije i orijentacije stvarnog robota i poznate transformacije između prirubnice robota i 3D kamere.
- Izradu simulacije robotskog programa koji dovodi TCP (*eng., tool center point*) robota u kontakt s digitaliziranom površinom.
- Pokretanje simuliranog robotskog programa na realnom robotu primjenom metoda dostupnih u RoboDK.

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

Zadatak zadan:	Datum predaje rada:	Predviđeni datumi obrane:
30. 11. 2022.	**1. rok:** 20. 2. 2023. **2. rok (izvanredni):** 10. 7. 2023. **3. rok:** 18. 9. 2023.	**1. rok:** 27. 2. – 3. 3. 2023. **2. rok (izvanredni):** 14. 7. 2023. **3. rok:** 25. 9. – 29. 9. 2023.

Zadatak zadao:

Doc. dr. sc. Marko Švaco

Dr. sc. Filip Šuligoj

Predsjednik Povjerenstva:

Prof. dr. sc. Branko Bauer

SADRŽAJ

POPIS SLIKA

POPIS TABLICA

SAŽETAK

U ovom završnom radu se integrira sustav za digitalizaciju radnog prostora robota primjenom 3D kamere. Korišteni su Universal Robots 5e robot i Intel Realsense D435i kamera. Sva korištena oprema nalazi se u Regionalnom centru izvrsnosti za robotske tehnologije (CRTA) u sklopu Fakulteta strojarstva i brodogradnje. Opisan je postupak izrade skripte za dohvaćanje oblaka točaka predmeta kojeg kamera snima, te vizualizaciju tog oblaka unutar simulacijskog softvera RoboDK. Riješen je problem kalibracije kamere i vrha alata. Koriste se matrice homogene transformacije za potrebe transformacije i dovođenja u odnos kamere (samim tim i oblaka točaka) i koordinatnog sustava baze robota. Naposljetku, kao zadatak koji robot izvršava je dolazak vrha alata u kontakt s površinom snimljenog predmeta na naredbu provedenu iz RoboDK softvera.

Ključne riječi: robot, kamera, kalibracija, RoboDK, Universal Robots, oblak točaka, alat

SUMMARY

In this bachelor thesis, a system for digitizing the robot workspace using a 3D camera is integrated. The Universal Robots 5e robot and the Intel Realsense D435i camera were used. All the equipment used is located at the Regional Center of Excellence for Robotics Technologies (CRTA) within the Faculty of Mechanical Engineering and Naval Architecture. It describes the process of creating a script for retrieving the point cloud of the object captured by the camera, and visualizing that cloud within the RoboDK simulation software. The problem of calibrating the camera and tooltip was solved. Homogeneous transformation matrices are used for transformation and bringing the camera (therefore the cloud of points) into relation with the coordinate system of the robot base. Finally, the task of the robot is to bring the tip of the tool in contact with the surface of the captured object in response to the command executed by the RoboDK software.

Key words: robot, camera, calibration, RoboDK, Universal Robots, point cloud, tool

1. UVOD

U današnjem svijetu, roboti su neizbježni dio mnogih industrijskih i proizvodnih procesa, kao i u mnogim drugim područjima kao što su zdravstvo, znanstvena istraživanja, pa čak i u zabavnoj industriji. Razvoj novih tehnologija koje bi unaprijedile i olakšale rad robota postaje ključan čimbenik u njihovoj primjeni i poboljšanju kvalitete proizvoda..

Jedna od takvih tehnologija je primjena RGB-D kamera za skeniranje objekata i stvaranje oblaka točaka. U ovom radu istražujemo primjenu D435 kamere na UR5e robotu za stvaranje oblaka točaka predmeta, te njihovo integriranje u softverski alat RoboDK. Cilj nam je stvoriti točan i pouzdan model skeniranog predmeta, te ga precizno pozicionirati u stvarnom svijetu pomoću robota UR5e.

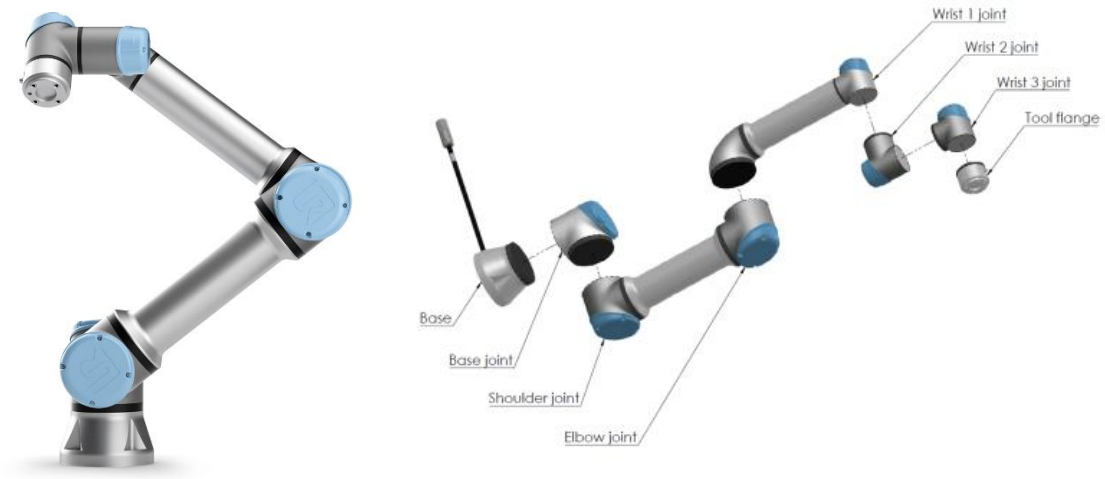
Robotski vid se odnosi na sposobnost robota da vide i interpretiraju svoju okolinu korištenjem senzora za vid. Ti senzori omogućuju robotima da prikupljaju podatke o svojoj okolini, poput oblika, veličine, boje i udaljenosti objekata. Korištenje robotskog vida ima mnoge prednosti u industriji, uključujući točniju i bržu inspekciju proizvoda, smanjenje rizika od ljudskih pogrešaka i mogućnost raditi u opasnim okruženjima.

Ova tehnologija ima širok spektar primjena u industriji, od automatskog sortiranja i pakiranja do proizvodnje složenih dijelova. Uz navedene primjene, ova tehnologija ima mogućnosti primjene u područjima poput medicinske tehnologije, umjetnosti, arheologije i drugdje. Stoga, razvoj novih tehnologija za stvaranje oblaka točaka predmeta, te njihova integracija s robotima i softverima, postaje sve važnija te ima potencijal za poboljšanje i automatiziranje mnogih industrijskih i drugih procesa.

2. UNIVERSAL ROBOT UR5e

UR5e robot je robotski sistem koji se koristi u industrijskom okruženju za automatizaciju proizvodnje. Ovaj robotski sistem je dizajniran kako bi se lako integrirao u postojeće proizvodne linije i kako bi se omogućilo precizno i pouzdano izvođenje različitih zadataka.

Težak je 20.6 kg te mu je nosivost 5 kg. Doseg mu je 850 mm. [1]



Slika 1. UR5e robot s prikazom dijelova

Robot se sastoji od kontrolera, robotske ruke i različitih tipova alata koji se mogu montirati na robotsku ruku. Kontroler je odgovoran za upravljanje robotskom rukom i alatom, a koristi se za programiranje robota i kontrolu njegovih pokreta. Vrlo je jednostavan za uporabu te ljudi bez iskustva u programiranju mogu pomoću kontrolera u obliku tableta lako koristiti robota.

Slika 2. Kontroler UR5e robota

Robotska ruka se sastoji od pet zglobova koji omogućuju robotu da ima širok raspon pokreta i fleksibilnost. To omogućuje robotu da se kreće u prostoru i da se prilagodi različitim situacijama i zahtjevima.

Dizajniran je tako da može raditi u različitim okruženjima, uključujući one s ograničenjima prostora. Robotska ruka je izrađena od visokokvalitetnih materijala koji osiguravaju izdržljivost i dugotrajnost.

UR5e se koristi u različitim industrijama, uključujući automobilsku, elektroničku, proizvodnju hrane i pića, te proizvodnju medicinskih proizvoda. Njegove glavne primjene su automatizacija proizvodnje, skladištenje i logistika.

Također ima mogućnost rada u suradnji s ljudima, što omogućuje da se roboti i ljudi međusobno dopunjuju u procesima proizvodnje. Ova funkcionalnost je ključna u smanjenju rizika od ozljeda na radu i povećanju efikasnosti proizvodnje.

Lako se integrira u postojeće proizvodne linije i omogućuje precizno i pouzdano izvođenje različitih zadataka. Programiranje UR5e robota se može obaviti korištenjem različitih softverskih alata, poput Universal Robots Polyscope alata za programiranje ili treće strane alate kao što su ROS ili RoboDK. Ovi alati omogućuju programiranje robota u različitim jezicima, kao što su C++, Python ili URScript.

UR5e se može koristiti u kombinaciji s različitim tipovima senzora, kao što su kamere i laserski senzori, kako bi se osigurala dodatna preciznost i funkcionalnost. Senzori se mogu koristiti za praćenje objekata te njegovog položaja u prostoru .

Osim toga, pošto je UR5e kompatibilan s različitim tipovima alata, koji se mogu montirati na robotsku ruku omogućuju robotu da se prilagodi različitim zahtjevima i da se koristi za različite vrste radova, kao što su paletizacija, sastavljanje dijelova, pakiranje, mjerenje, provjera kvalitete, zavarivanje, glodanje i sl.

Sve u svemu, UR5e robot je fleksibilan i multifunkcionalan, što ga čini idealnim za automatizaciju proizvodnje u različitim industrijama. Njegova sposobnost rada u suradnji s ljudima i mogućnost integracije s različitim tipovima senzora i alata čine ga izvrsnim rješenjem za povećanje efikasnosti i smanjenje rizika ozljeda na radu.

Slika 3. UR5e robot u industrijskom pogonu

3. DEPTH KAMERA

3.1. Općenito o depth kamerama

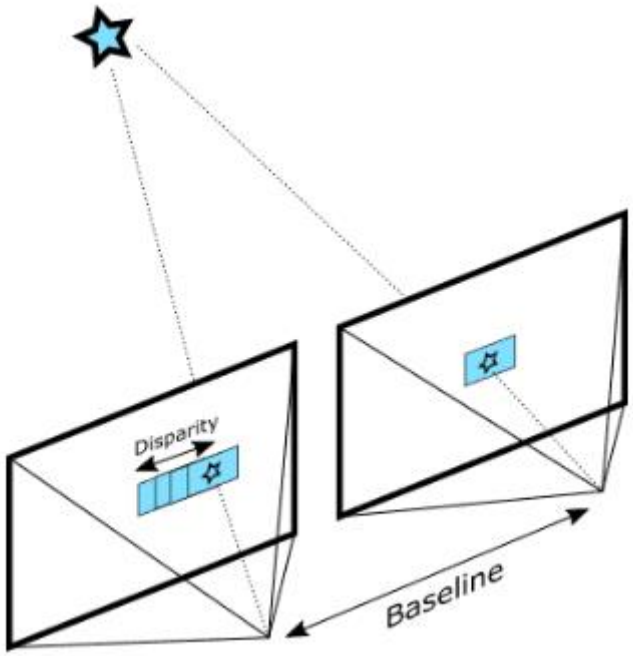
Postoje različiti tipovi 3D senzora koji se razlikuju po načinu dobivanja podataka o prostoru, načinu na koji obrađuju podatke, rezoluciji, daljina koju kamera može dohvatiti, itd. Te kamere omogućuju robotima percepciju prostor u 3 dimenzije. . Zbog korisnih informacija koje kamere za dubinu pružaju, one su postigle važnost u aplikacijama poput robota upravljanih vizijom, inspekcije i nadzora. 3D podaci su bolji od 2D podataka jer su manje podložni smetnjama kao što je npr. promjena svjetlosnih uvjeta na koje su kamere osjetljive. Stoga roboti postaju pouzdaniji i precizniji prilikom izvođena zadataka. [2]

Načini na koje kamere detektiraju dubinu:

1. Stereo senzori
2. Time-of-Flight senzori
3. Senzor strukturiranog svjetla (Structured light)
4. LiDAR (Light Detection Ranging) senzori

3.1.1. Stereo senzori

Stereo senzori pokušavaju kopirati ljudski vid koristeći dvije kamere koje gledaju isti predmet, a razmaknute su za određenu udaljenost. Slike s ovih kamera se prikupljaju i koriste se za izdvajanje i uspoređivanje vizualnih podataka kako bi se dobila tzv. karta razlike između pogleda kamera. Informacije o razlici su suprotne dubini i lako se mogu koristiti za dobivanje karte dubine. Neki senzori ovog tipa obično rade samo u okruženjima bogatim vizualnim podacima i imaju probleme u okruženjima bez podataka (npr. bijelom zidu). Prednost ovog tipa senzora je da obično podržava veće rezolucije slika dubine od drugih tipova kamera. Računanje i uspoređivanja podataka se obavljaju na kameri, što znači da obično zahtijevaju više snage za obradu u usporedbi s drugim metodama. Također, radni domet stereo kamere je ograničen udaljenosti između kamera, jer se pogreška povećava kvadratno s udaljenošću objekata od senzora. [2]



Slika 4. Princip rada stereo kamere

3.1.2. Time-of-Flight senzori

Ovi senzori koriste svjetlost tako da ju pošalju do predmeta te računaju dubinu tako što mjere vrijeme koje je potrebno svakom fotonu da se vrati do senzora. To znači da svaki piksel odnosi se na jedan snop svjetlosti projiciran od strane uređaja, što pruža veću gustoću podataka, uže sjene iza objekata i lakšu kalibraciju. Budući da koriste projekciju svjetlosti, ovi senzori su osjetljivi na različite vrste površina, kao što su vrlo reflektirajuće ili vrlo tamne. U tim slučajevima, obično se pojavljuju nevažeći podaci. Međutim, oni su mnogo otporniji na niske svjetlosne ili tmurne uvjete od stereo senzora koji ovise o svjetlosti u sceni. [2]

3.1.3. Senzori strukturiranog svjetla (Structured light)

Senzori strukturiranog svjetla (SL) koriste poznati uzorak koji se projicira od strane senzora IF na scenu. Način na koji se taj uzorak deformira koristi se za izgradnju karte dubine. Ovaj tip senzora ne zahtijeva vanjski svjetlosni izvor i uglavnom se koristi u unutarnjim okruženjima zbog njihove niske otpornosti na sunčevo svjetlo, budući da može doći do interferencije s projiciranim svjetlosnim uzorkom. [2]

3.1.4. LiDAR (Light Detection Ranging) senzori

Slični su ToF senzorima, također koriste skeniranje, detekciju svjetlosti i mjerenje udaljenosti, ali su precizniji od ToF senzora jer koriste više impulsa IR-a dok ToF senzori obično koriste samo jedan impuls svjetlosti za dobivanje slike. Mapiraju okoliš točka po točku umjesto skupa točaka (kao kod ToF-a), na temelju tog ukupni šum za svaku točku se smanjuje, što daje preciznije rezultate. Osim toga, LiDAR obično može dobiti mjerenja na većoj udaljenosti od tradicionalnih ToF ili stereo senzora. Međutim, LiDAR senzori su obično skuplji od ostalih 3D senzora i, budući da rade skeniranje, mogu biti osjetljivi na dinamički uvjete. [2]

3.2. Intel RealSense D435i kamera

Intel® RealSense™ kamera D435i je stereo kamera koje pruža kvalitetnu dubinu za različite aplikacije. Njegov široki kut gledanja i raspon od 10 metara je savršen za aplikacije poput robota gdje je važno vidjeti što više scene moguće. U odnosu na 435 model, 435i model koristi IMU (The inertial measurement unit) tj. detekciju kretanja i rotaciju u 6 stupnjeva slobode. IMU kombinira različite senzore sa žiroskopom za detekciju rotacije i kretanja u 3 osi. [4]

Slika 5. D435i kamera i njezine komponente

Tablica 1. Specifikacije D435i kamere

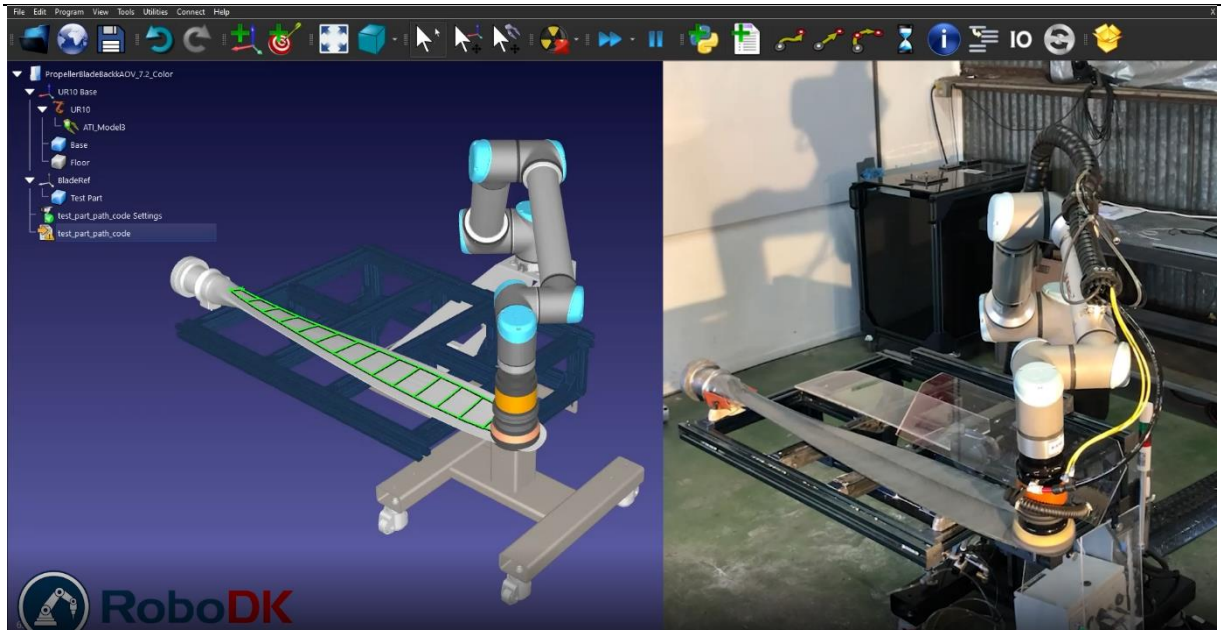
SPECIFIKACIJE	DETALJI
Vrsta kamere	Stereo Depth
Senzor slike	Global Shutter, rezolucija 1280 x 720
Raspon kamere	0.25 m do 10 m
Vidno polje	92° x 65° x 77°
Maksimalna brzina snimanja	90 fps
Dimenzije	70 mm x 25 mm x 25 mm
Težina	30 g
Radna temperatura	0°C do 40°C

4. ROBODK

RoboDK je simulacijski softver koji pruža okruženje za dizajniranje, programiranje i simulaciju industrijskih robota. U današnjoj brzoj i izuzetno konkurentnoj industriji, imati sposobnost simuliranja i testiranja robota i programskih kodova bez fizičkog pokretanja stroja može donijeti značajnu prednost. RoboDK nudi niz usluga svojim korisnicima, uključujući povećanje učinkovitosti, smanjenje vremena i troškova proizvodnje te poboljšanje sigurnosti. Softver ima korisniku prijateljsko sučelje i široku biblioteku robota i alata, što ga čini jednostavnim i za neiskusne korisnike. U RoboDK se može programirati korištenjem različitih programskih jezika, uključujući Python, C# i C++, što čini dostupnim većini korisnika. Softver također podržava rad više robota u tandemu, omogućujući korisnicima da simuliraju složene industrijske procese bez problema. Sposobnost testiranja i validacije robot programskih kodova bez fizičkog pokretanja na stroju može značajno smanjiti vrijeme i troškove povezane s ispravljanjem i fini-tuniranjem programskog koda robota. Također eliminira potrebu za zaustavljanjem proizvodnje i zaustavljanjem stroja, što može imati značajan utjecaj na produktivnost i rentabilnost.

Analizirana je upotreba RoboDK softvera u odnosu na Robot Operating System (ROS) za rješavanje dijagnostičkih zadataka u off-line programiranju robota. Izvršena je usporednu analizu performansi ova dva softvera u različitim scenarijima programiranja, kao što su kreiranje putanja, planiranje kretanja i simulacije.

Zaključno je da je RoboDK je pružio bolju kvalitetu simulacije i preciznije planiranje putanje u odnosu na ROS. Također je primijećeno da je RoboDK jednostavniji za korištenje i ne zahtijeva značajno programiranje, što ga čini prikladnijim za manje stručne korisnike. [3]



Slika 6. RoboDK sučelje i izvedba u stvarnom svijetu

Druga prednost korištenja RoboDK-a je sposobnost simuliranja složenih industrijskih procesa i identificiranja potencijalnih sudara i drugih problema prije nego se oni dogode. Ovo može pomoći povećati sigurnost stroja i smanjiti rizik od nesreća ili oštećenja opreme. Osim toga, softver daje detaljne izvještaje za svaki korak simulacije, što omogućuje korisnicima da brzo i lako isprave bilo koje probleme ili nedostatke u programu. To također omogućuje korisnicima da razumiju korake u svom procesu i da se fokusiraju na ključne probleme. Također podržava integraciju s različitim CAD i CAM softverima. [5]

5. ROBOTSKI VID

Ideja o robotskom vidu potječe još iz 1960-ih godina, kada su znanstvenici prvi puta počeli razvijati računalne modele koji bi mogli prepoznavati objekte na slikama.

Tijekom 1970-ih i 1980-ih godina, razvijeni su prvi sustavi za prepoznavanje lica i ruku, a krajem 1980-ih godina razvijeni su prvi sustavi za prepoznavanje objekata u 3D prostoru. U 1990-ima godinama, razvijeni su algoritmi koji su omogućili prepoznavanje objekata u realnom vremenu i u dinamičnim scenama, a krajem 2000-ih godina, razvijeni su i algoritmi za prepoznavanje emocija na licima. Robotski vid predstavlja važnu grana računalne vizije koja se kontinuirano razvija i primjenjuje u različitim područjima. [6]

Vizija robota je područje robotike koje se fokusira na omogućavanje robotima da percipiraju svoje okruženje i donose odluke na temelju te percepcije. Dok nije postojao robotski vid, roboti su bili samo strojevi koji se kreću na temelju programa stvorenih od strane ljudi. Strogo su slijedili naredbe u kodu te je to bilo idealno za ponavljajuće zadatke koji su ljudima fizički iscrpljujući. Pošto se industrija razvija tako se razvijaju i roboti. Dolazi do uvođenja robotskog vida koji poboljšava razinu preciznosti i točnosti robota u pametnim automatiziranim procesima. Roboti mogu obavljati zadatke kao što su pregled, identifikacija, brojanje, mjerenje ili očitavanje barkodova. Strojno učenje je također primjenjivo na robotski vid što ga još više unaprjeđuje. [7]

Robotski vizijski sustavi koriste kamere i druge uređaje za snimanje slika i prikupljanje vizualnih podataka iz okoline. Vrsta korištenog uređaja za snimanje slike ovisi o primjeni i zahtjevima zadatka robota. Na primjer, neki roboti koriste stereo kamere za percepciju dubine, dok drugi koriste jednobojne kamere za bolji kontrast.

Nakon što su vizualni podaci snimljeni, oni se obrađuju pomoću algoritama za izdvajanje značajnih informacija te se informacije koriste za usmjeravanje radnji robota.

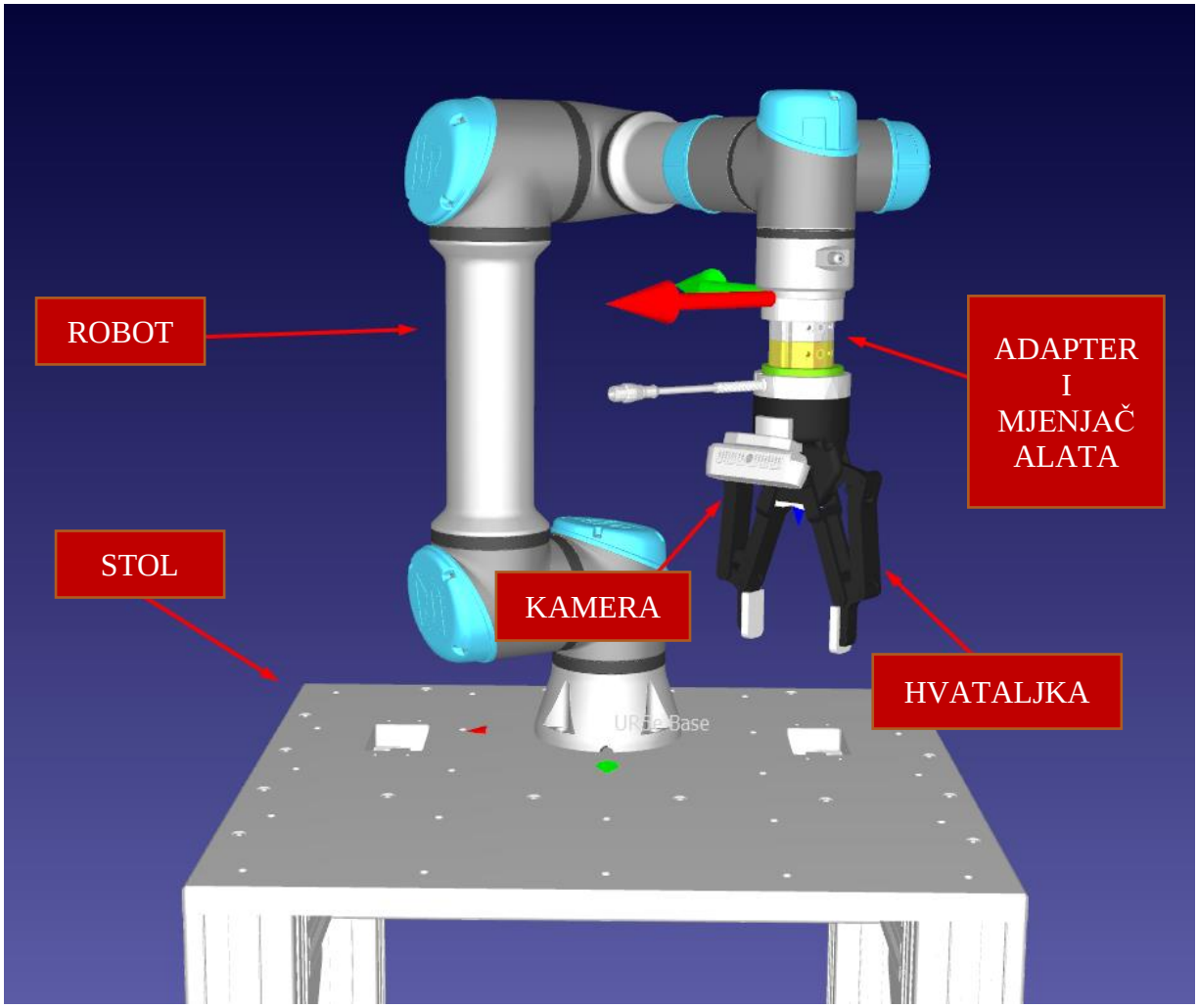
Obrada slike u stvarnom vremenu ključna je za mnoge aplikacije robotike, jer omogućuje robotu da brzo reagira na promjene kao npr. da otkrije proizvod koji nije dovoljno obrađen i uklanja ga s proizvodne trake. Unatoč napretku tehnologije, još uvijek postoje mnogi izazovi u hvatanju i obradi vizualnih podataka kao što su uvjeti slabog osvjetljenja ili prašina u proizvodnim pogonima.

Slika 7. Ilustrativni prikaz robotskog vida

6. STVARANJE RADNOG OKRUŽENJA U ROBODK

Kako bi prikaz u RoboDK bio što sličniji onom u stvarnom svijetu rekreiran je implementiranjem 3D modela korištenih komponenti. RoboDK ima biblioteku s mnoštvom modela robota i alata pa tako ima i model korištenog UR5e robota. Također u biblioteci se nalazi i alat koji je spojen na robota, a to je Robotiq 2F-140 hvataljka. Kako bi se spojila hvataljka i prirubnica robota koriste se adapter i modul za brzu zamjenu alata marke Schunk. Njihove 3D modele je također moguće naći na internet stranicama kompanije Schunk. Isto vrijedi i za kameru čiji je model skinut s interneta i dodan u RoboDK. Zadnja komponenta koja je dodana je stol na koji je smješten robot koji mi je ustupljen od strane mentora.

Nakon što su svi dijelovi dodani u RoboDK mijenjanjem njihovih koordinata u odnosu na koordinatni sustav robota svi su dijelovi složeni na robota.



Slika 8. Prikaz složenih komponenti u RoboDK

Slika 9. Prikaz složenih komponenti u stvarnom svijetu

7. IZRADA PYTHON SKRIPTE

Skripta je izrađena u Python programskom jeziku te implementirana u RoboDK kako bi pokrenula proces dohvaćanja oblaka točaka predmeta kojeg je kamera snimila te ga pretvara u format koji podržava RoboDK (.obj).

Skripta koristi više Python biblioteka, a jedna od tih je „pyrealsense2“. To je biblioteka za Intel RealSense kamere koja omogućuje pristup funkcijama kamere te podacima o slici koju kamera snima. „Numpy“ je numerička biblioteka za Python koja pruža podršku za rad s matricama i raznim matematičkim funkcijama. Kao što postoji robotski vid, tako postoji i računalni vid. Za tu vrstu vida zadužena je OpenCV biblioteka koja pruža funkcije za obradu slika, uključujući osnovne operacije poput rezanja i mijenjanja veličine, kao i naprednije funkcije poput detekcije objekata i prepoznavanja lica. U skripti je korištena i Open3D biblioteka, tj. dvije funkcije iz te biblioteke. „Read_point_cloud“ je funkcija koja iz datoteke procesuira podatke oblaka točaka i pohranjuje ih u obliku podataka koji se mogu koristiti za daljnju obradu i vizualizaciju. "Draw_geometries" je funkcija u Open3D biblioteci u Pythonu koja prikazuje popis 3D geometrijskih objekata u interaktivnom prozoru za vizualizaciju. Objekti mogu biti oblaci točaka, mreže trokuta itd. Funkcija prima popis objekata kao ulaz i otvara interaktivni prozor za vizualizaciju koji prikazuje te objekte. Prozor podržava razne oblike interakcije, uključujući rotaciju, zumiranje i odabir objekata.

```
import pyrealsense2 as rs
import numpy as np
import cv2
from open3d.cpu.pybind.io import read_point_cloud
from open3d.cpu.pybind.visualization import draw_geometries
```

Slika 10. Python kod - uvoženje biblioteka

Drugi dio koda se odnosi na stvaranje „pipeline-a“ te konfiguriranje istog za dobivanje „stream-a“ s kamere. Na temelju podataka o tipu kamere postavljaju se podržane rezolucije snimanja dubine i boje. U ovom slučaju obje rezolucije su 640x480 pixela. Nakon dohvaćanja tih podataka počinje prijenos slike s kamere te dohvaćanje podataka o skali dubine sa senzora dubine kamere. U skripti se također može namjestiti do koje daljine kamera dohvaća podatke što je u ovom slučaju namješteno na 5 metara. Pošto kamera koristi različite leće za sliku u boji i sliku dubine moramo ih pomoću skripte poravnati u jedno kako bi se podaci sinkronizirali.

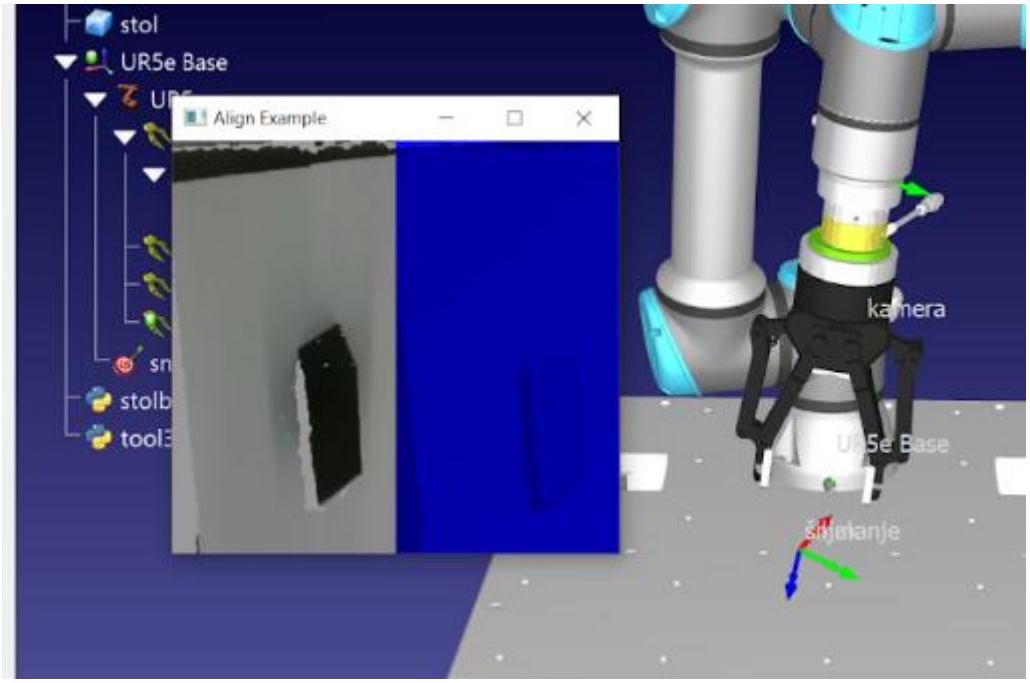
```
pipeline = rs.pipeline()

config = rs.config()

colorizer = rs.colorizer()

pipeline_wrapper = rs.pipeline_wrapper(pipeline)
pipeline_profile = config.resolve(pipeline_wrapper)
device = pipeline_profile.get_device()
device_product_line = str(device.get_info(rs.camera_info.product_line))

config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)

profile = pipeline.start(config)

depth_sensor = profile.get_device().first_depth_sensor()
depth_scale = depth_sensor.get_depth_scale()

clipping_distance_in_meters = 5
clipping_distance = clipping_distance_in_meters / depth_scale

align_to = rs.stream.color
align = rs.align(align_to)
```

Slika 11. Python kod – konfiguriranje kamere za snimanje

Treći dio koda se odnosi na petlju za prikupljanje slike s kamere. To je beskonačna petlja koja prikuplja „frejmove“ i poravnava ih. Dobivamo sliku dubine i sliku boje. Pozadina dalja od 5 metara (clipping distance) se boja u sivo kako bi se uklonila. Naposljetku, dobivamo sliku dubine u boji. Otvara se prozor koji nam prikazuje snimku s kamere.

```
try:
    while True:
        frames = pipeline.wait_for_frames()

        aligned_frames = align.process(frames)

        aligned_depth_frame = aligned_frames.get_depth_frame()
        color_frame = aligned_frames.get_color_frame()

        if not aligned_depth_frame or not color_frame:
            continue

        depth_image = np.asanyarray(aligned_depth_frame.get_data())
        color_image = np.asanyarray(color_frame.get_data())

        grey_color = 153
        depth_image_3d = np.dstack((depth_image,depth_image,depth_image))
        bg_removed = np.where((depth_image_3d > clipping_distance) | (depth_image_3d <= 0), grey_color, color_image)

        depth_colormap = cv2.applyColorMap(cv2.convertScaleAbs(depth_image, alpha=0.03), cv2.COLORMAP_JET)
        images = np.hstack((bg_removed, depth_colormap))

        cv2.namedWindow('Align Example', cv2.WINDOW_NORMAL)
        cv2.imshow('Align Example', images)
```

Slika 12. Python kod – prozor koji emitira sliku s kamere



Slika 13. Prozor - emitirana slika s kamere

Zadnji dio koda počinje tako da uvodimo naredbu da pritiskom tipke „q“ ili „ESC“ na tipkovnici zatvaramo prozor. Ako pritisnemo tipku „e“ od slike s kamere se stvara 3D model format (out.ply).

Format PLY (Polygon File Format ili Stanford Triangle Format) je format za 3D geometrijske modele, često se koristi za pohranjivanje informacija o 3D modelima, kao što su točke, rubovi i lica. Format PLY razvijen je na Sveučilištu Stanford i često se koristi za pohranjivanje mreža trokuta, iako ga također može koristiti i za druge vrste 3D modela. Format je tekstualan, a svaki red u datoteci predstavlja vrh, plohu ili neke druge aspekte modela. [8]

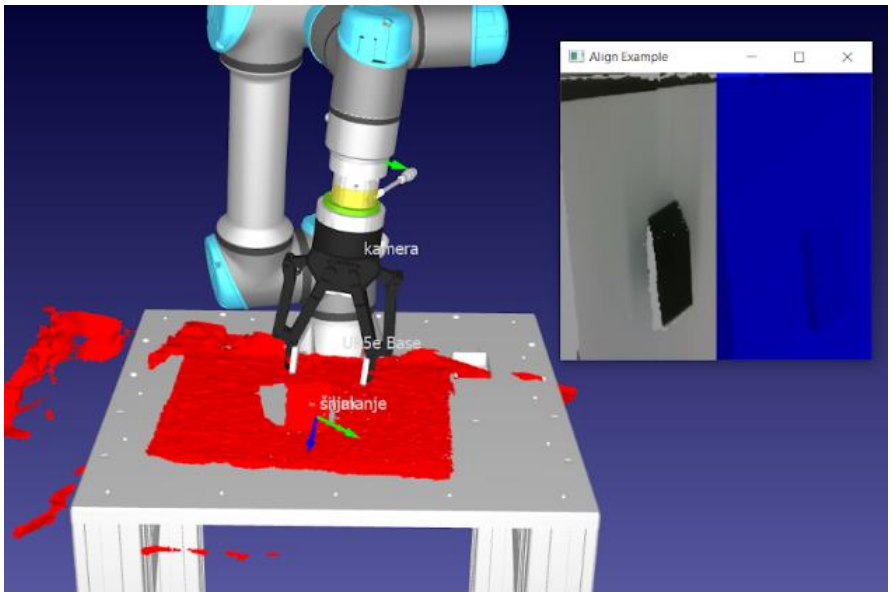
RoboDK ne podržava format PLY i zbog toga ga se moralo pretvoriti u drugi podržani format. U ovom slučaju to je OBJ format. OBJ datoteka (.obj) sadrži informacije o geometriji 3D objekata. Datoteke se koriste za razmjenu informacija, CAD-a i 3D ispisa. Jedna datoteka može definirati više objekata. Objekti u OBJ datoteci definirani su poligonskim licima (koja su definirana vrhovima ili točkama) i normalama, krivuljama i površinama. OBJ je vektorska datoteka što čini objekte skalabilnima te nema maksimalne veličine datoteke.

To je učinjeno uz pomoć Python biblioteke PyMeshLab. To je popularna biblioteka za obradu i analizu trokutastih mreža. Uz pomoć njega moguće je lako izvršiti širok raspon operacija s trokutastim mrežama kao što su čišćenje i popravak, pojednostavljivanje, transformacija i analiza. Dizajnirana je da bude laka za korištenje i moćan je alat za sve koji rade s trokutastim mrežama.

Kad se završi transformacija formata 3D modela, taj model se učitava u RoboDK i to u odnosu na koordinatni sustav kamere. Radi lakšeg razlikovanja 3D modela od ostalih značajki u RoboDK skripta boja model u crvenu boju.

```
            key = cv2.waitKey(1)

            if key & 0xFF == ord('q') or key == 27:
                cv2.destroyAllWindows()
                break
            if key == ord("e"):
                a
                frames = pipeline.wait_for_frames()
                colorized = colorizer.process(frames)
                ply = rs.save_to_ply("out.ply")

                ply.set_option(rs.save_to_ply.option_ply_binary, False)
                ply.set_option(rs.save_to_ply.option_ply_normals, True)

                print("Saving to out.ply...")

                ply.process(colorized)
                print("Done")

                import pymeshlab

                ms = pymeshlab.MeshSet()
                ms.load_new_mesh("out.ply")
                ms.save_current_mesh("out.obj")

                from robolink import *  # RoboDK API
                from robodk import *  # Robot toolbox

                RDK = Robolink()

                frame = RDK.Item('kamera')
                part = RDK.AddFile(r'C:\Users\Antonio\Desktop\završni\out.obj', frame)
                part.setColor([1, 0, 0, 1])
    finally:
        pipeline.stop()
```

Slika 14. Python kod - stvaranje, konverzija formata i učitavanje u RoboDK 3D modela

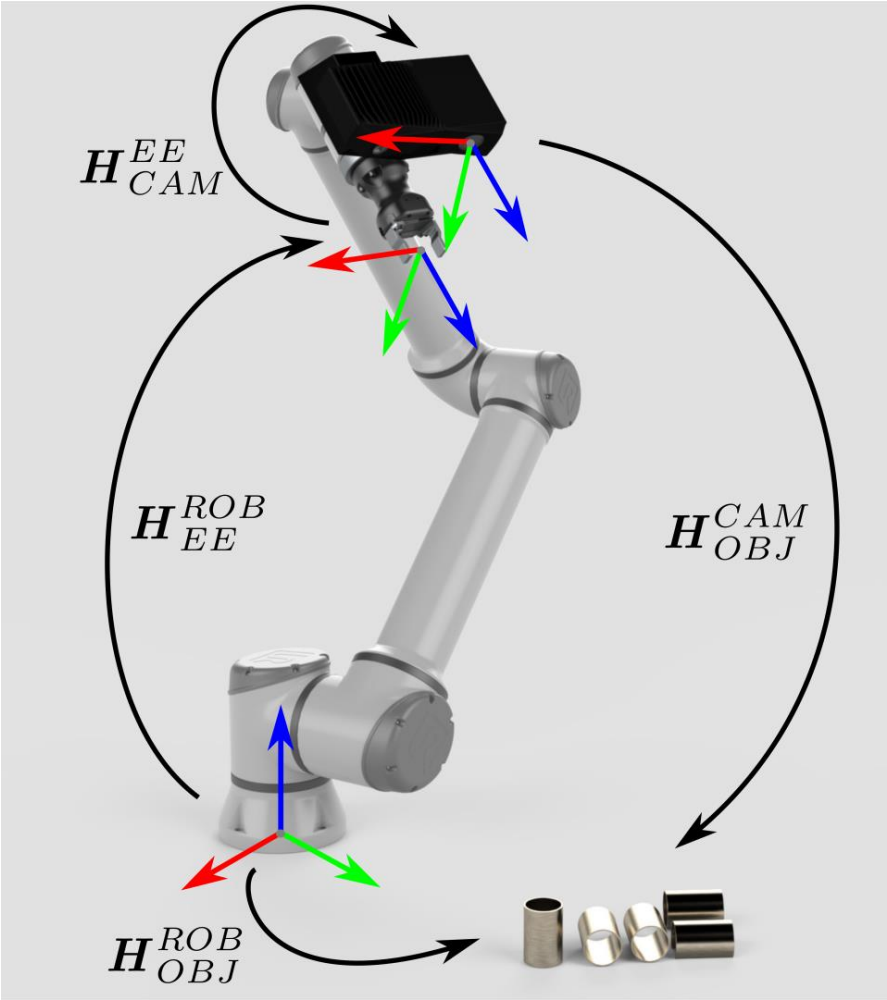


Slika 15. Prikaz snimljenog i učitanog predmeta u RoboDK

8. KALIBRACIJA

8.1. Općenito o eye-in-hand kalibraciji kamere

Kad je kamere postavljenja na robotsku ruku to se naziva eye-in-hand tj. oko na ruci. Kako bi alat mogao precizno dolaziti u kontakt s površinom predmeta mi trebamo znati položaj kamere u odnosu na prirubnicu robota. Zbog toga se radi kalibracija kamere kako bi kao rezultat dobili rotaciju i translaciju koordinatnog sustava kamere u odnosu na koordinatni sustav prirubnice. Obično se taj rezultat dobiva tako što se uslika set slika predmeta poznate geometrije u različitim pozama robota. [9]



Slika 16. Prikaz problema kalibracije kamere

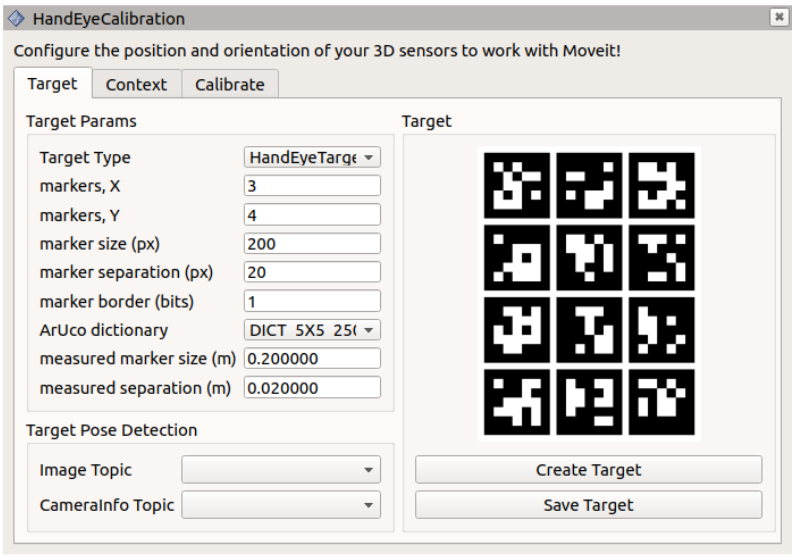
Najčešće se kao predmet poznate geometrije koristi kalibracijska ploča koja je lako vidljiva kameri. U eye-in-hand slučaju kalibracije imamo poznatu pozu prirubnice u odnosu na bazu robota, također imamo poznatu pozu predmeta u odnosu na kameru i imamo nepoznatu pozu predmeta u odnosu na bazu robota koja je uvijek konstanta. Kako bi zatvorili krug jedina nepoznata je poza kamere u odnosu na prirubnicu.

8.2. Rješavanje eye-in-hand kalibracije kamere

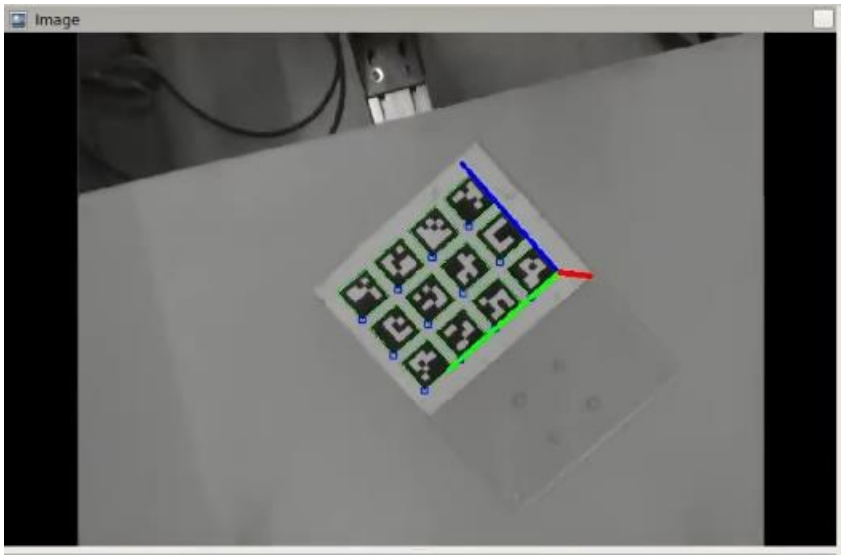
Problem kalibracije je vrlo zahtjevan za rješavanje te sam se odlučio da ga riješim pomoću MoveIt-a.

MoveIt! je moćan open-source softver za manipulaciju robota razvijen unutar ROS (Robot Operating System). Pruža jednostavnu za uporabu platformu za integraciju planiranja kretanja, manipulacije i percepcije za robote u jednom sustavu. Omogućuje brzu i jednostavnu izradu naprednih aplikacija za robote u širokom spektru domena, od industrijske automatizacije do medicinske robotike. Pored toga, MoveIt! pruža skup alata za vizualizaciju i ispravljanje pogrešaka u planovima kretanja i za testiranje i potvrđivanje performansi robota.

Za MoveIt postoji paket koji rješava problem kalibraciju kamere. Potrebno ga je instalirati i pokrenuti. Također je potrebno skinuti UR5e model za ROS i instalirati potrebne pakete za D435i kameru. Prilikom pokretanja prvi korak je da napravimo kalibracijsku ploču koju je zatim potrebno spremiti i isprintati. Nakon printanja mjerimo veličinu kvadrata i prostora između svakog kvadrata te unosimo tražene podatke.



Slika 17. Sučelje unutar MoveIt-a – izrada kalibracijske ploče

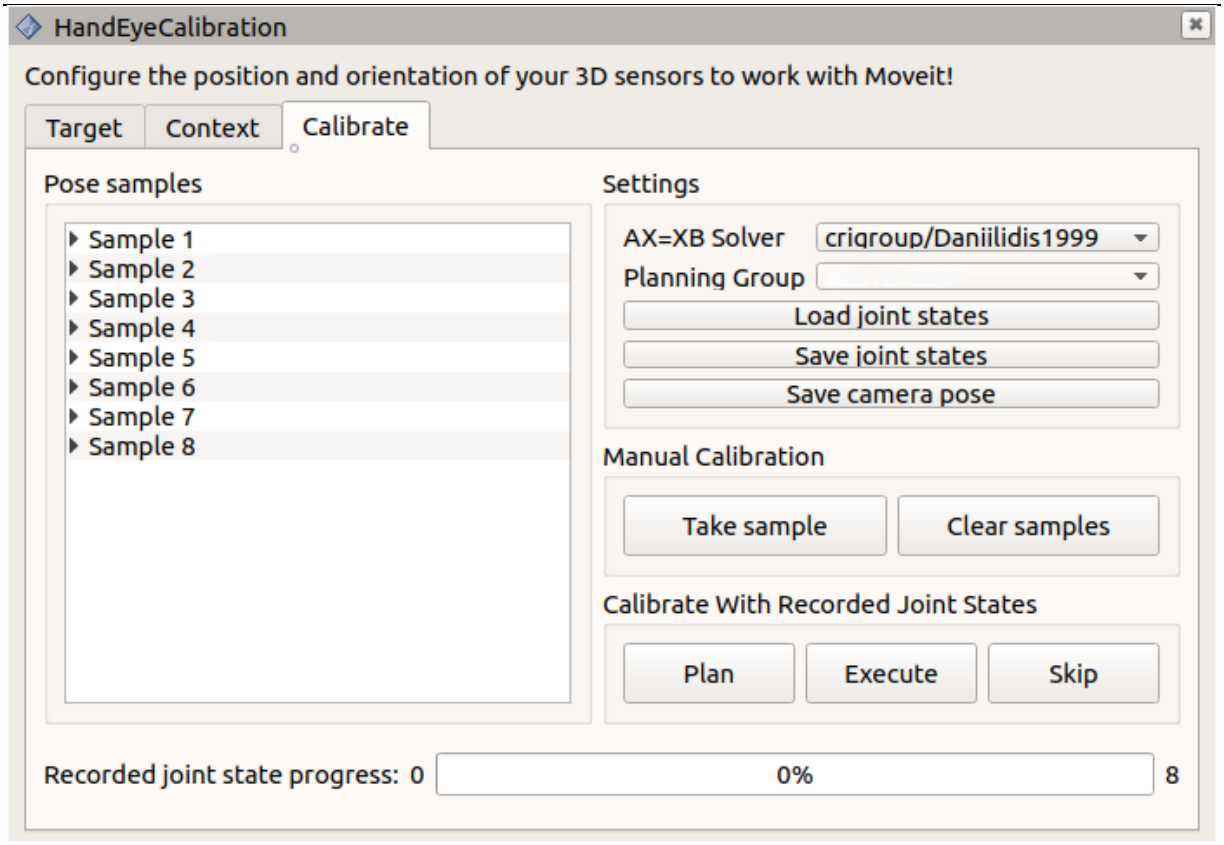


Slika 18. Prikaz kako kamera vidi kalibracijsku ploču

Nakon toga moramo postaviti koordinatne sustave kamere, kalibracijske ploče, prirubnice i baze robota. Tad možemo početi s kalibracijom. Prikupljamo 15 različitih uzoraka i pri svakom uzorku mijenjamo pozu i orijentaciju robota da gleda na kalibracijsku ploču iz drugog ugla.

Slika 19. Proces kalibracije

Uzorci su uzimani tako da se robot pomicao u različite poze te su ručno uneseni njegovi položaji zglobova u MoveIt. Podaci o rotaciji svakog zgloba u određenoj pozi možemo naći na kontroleru UR5e robota. Svakom sljedećom pozom rezultat konvergira u točnije rješenje. Prve rezultate dobivamo već nakon 5 poza, dok nakon 15 poza dobivamo prilično točno rješenje s malim odstupanjima. Kad smo dobili rješenje unesemo ga u RoboDK i kamera je kalibrirana.



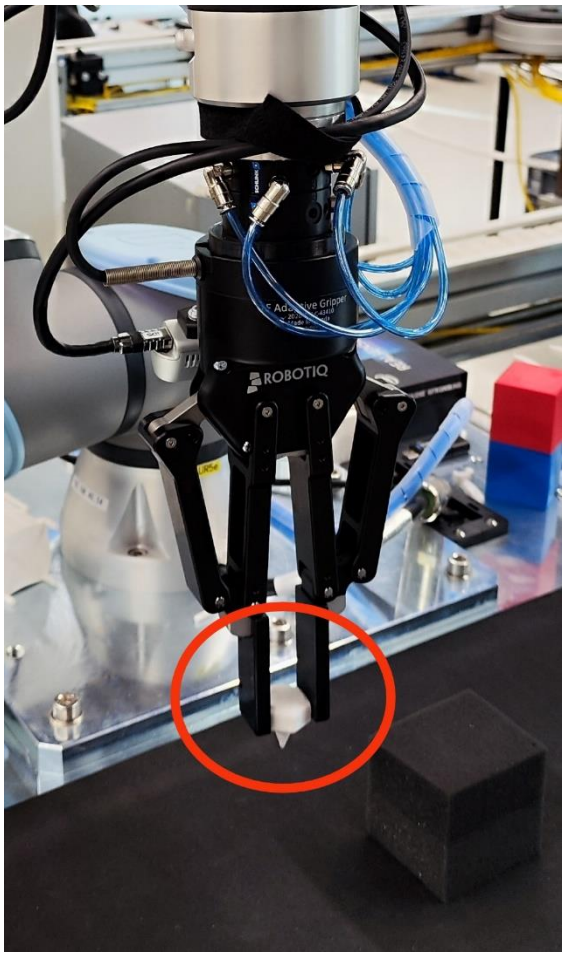
Slika 20. Sučelje unutar MoveIt-a – prikupljanje uzoraka i računanje kalibracije

8.3. Kalibracija šiljka

Kalibracijski šiljak koji je stisnut u hvataljku je korišten kao alat koji dolazi u kontakt s površinom predmeta snimljenog kamerom. Kako bi precizno znali gdje se alat nalazi u odnosu na prirubnicu robota treba provesti kalibraciju. Kalibracija je provedena pomoću sučelja za kalibraciju alata u Universal Robots Polyscope-u.

Slika 21. Sučelje za kalibraciju alata unutar kontrolera UR5e

Proces kalibracije se odvija pomoću 2 kalibracijska šiljka. Jedan postavimo na ravnu površinu, dok je drugi šiljak u hvataljci robota. Vrh šiljka u hvataljci dovodimo u kontakt s vrhom šiljka koji je statičan na površini i uvijek zadržava istu poziciju. Potrebne su nam 4 različite poze robota iz različitih uglova da bi dobili konačnu kalibraciju.



Slika 22. Kalibracijski šiljak stisnut u hvataljci

9. SPAJANJE UR5e ROBOTA SA RAČUNALOM

Kako bi robot izvršavao komande koje mu zadamo unutar RoboDK potrebno ga je povezati. Unutar RoboDK postoji sučelje za povezivanje robota koje koristi TCP/IP protokol.

9.1. TCP/IP protokol

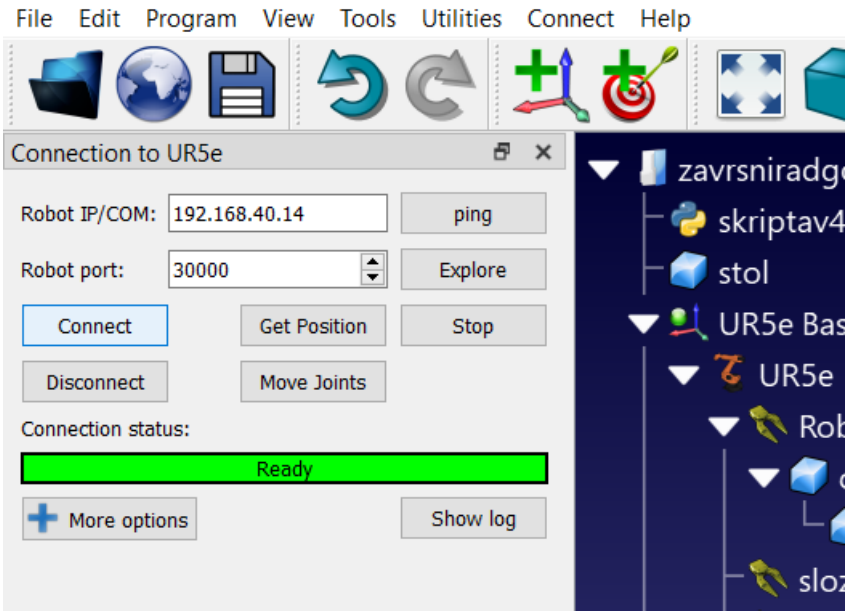
TCP/IP protokol je skup pravila i procedura koji se koriste za komunikaciju između računala na mreži. Ovaj protokol je osnova za funkcionalnost interneta, a omogućava računalima da međusobno razmjenjuju podatke putem paketa koji se šalju preko mreže.

TCP (Transmission Control Protocol) je jedan od osnovnih protokola u TCP/IP arhitekturi, a koristi se za pouzdanu i sigurnu komunikaciju između računala. On uspostavlja konekciju između računala i omogućava slanje podataka u segmentima koji se ponovno šalju ako se izgube ili oštete tijekom prijenosa.

IP (Internet Protocol) je drugi osnovni protokol u TCP/IP arhitekturi, a koristi se za adresiranje i usmjeravanje paketa preko mreže. IP adrese su jedinstveni identifikatori za računala na mreži, a omogućuju im da šalju i primaju pakete. [10]

9.2. Povezivanje UR5e sa RoboDK

Većina robota u CRTA-i imaju svoju IP adresu kako bi ih mogli kontrolirati s bilo kojeg računala koje je povezano Ethernet kabelom. Tako svoju ima i UR5e te ju koristimo kako bi se povezali unutar RoboDK. Prije samog povezivanje potrebno je robota namjestiti na „remote control mode-u“ odnosno na način uporabe s vanjskog kontrolera. Inače kad koristimo kontroler UR5e koji je fizički povezan s robotom on je u „local control mode-u“. Unesemo IP adresu robota i spojimo se. Zelena boja nam signalizira da je robot spojen.

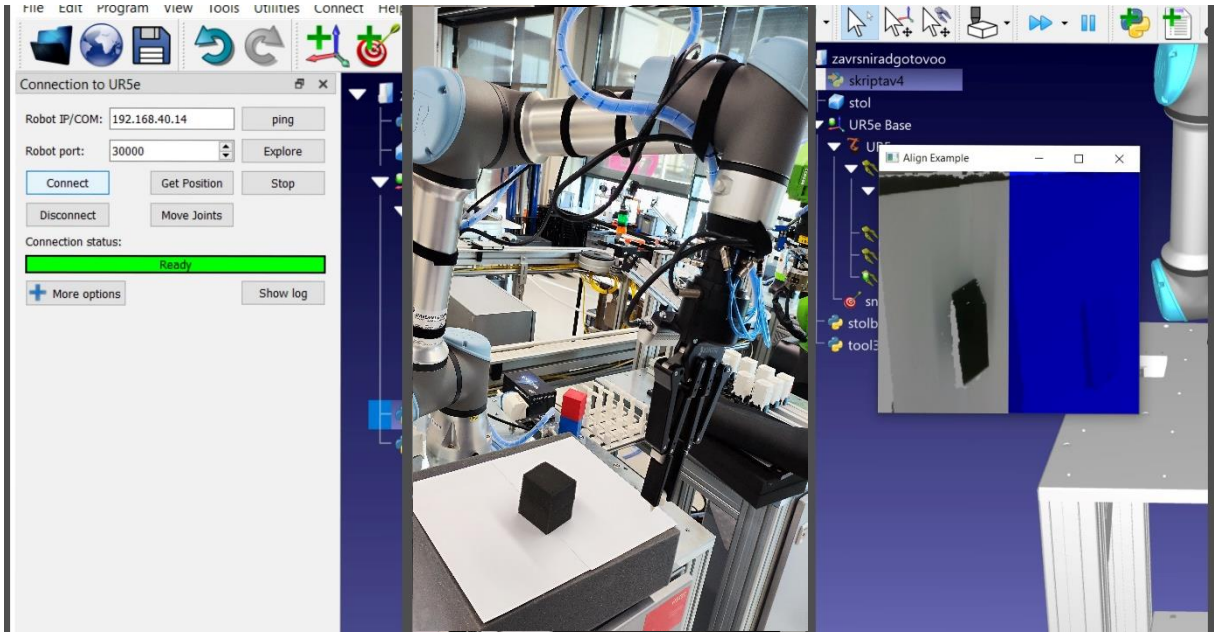


Slika 23. Sučelje unutar RoboDK za povezivanje s UR5e

10. CIJELI PROCES

10.1. Snimanje predmeta

Prvo što trebamo napraviti je spojiti robota sa RoboDK i odabrati opciju „Get Position“ kako bi dobili trenutnu pozu robota iz stvarnosti u RoboDK. Drugi korak je namjestiti otprilike kameru da snima predmet koji je u ovom slučaju crna kocka. Pokrećemo skriptu koja otvara na računalu prikaz slike koju kamera vidi i gledamo da li je predmet u okviru kamere.

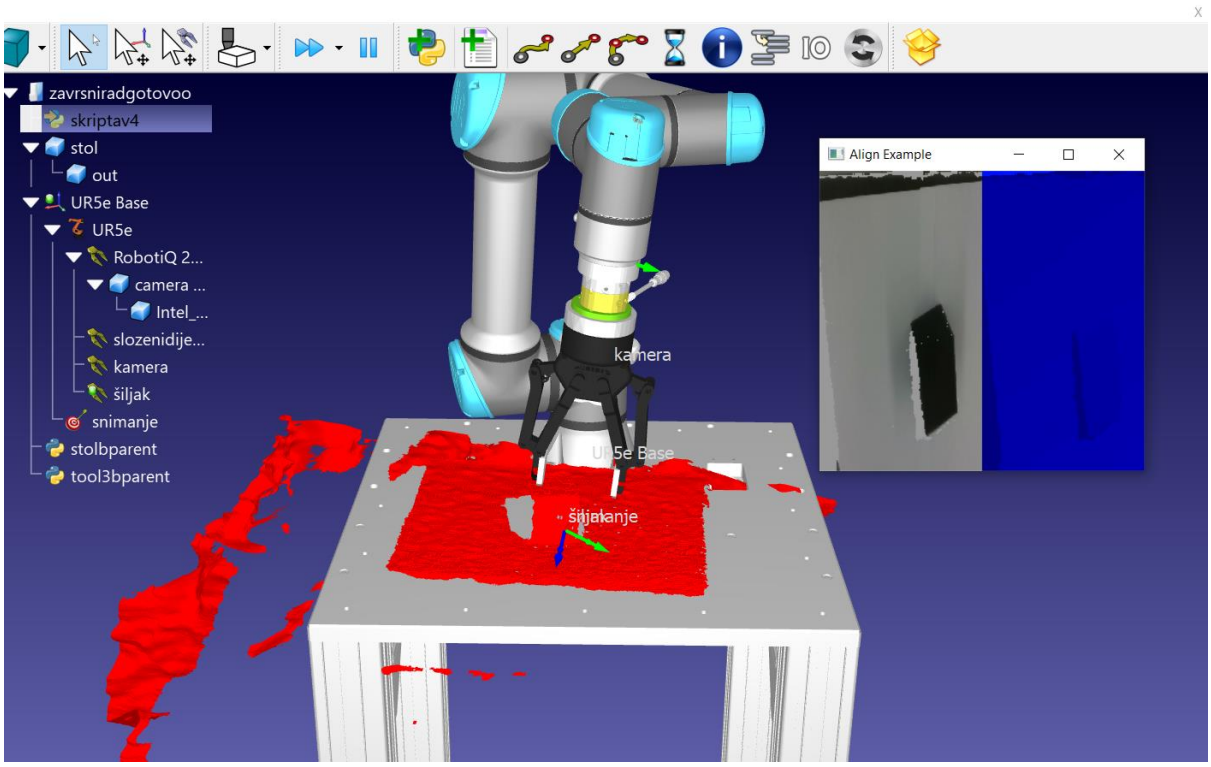


Slika 24. Proces snimanja predmeta

a) spajanje robota sa RoboDK: b) namještanje kamere iznad predmeta; c) pokretanje skripte da bi vidjeli što kamera vidi

10.2. Stvaranje oblaka točaka

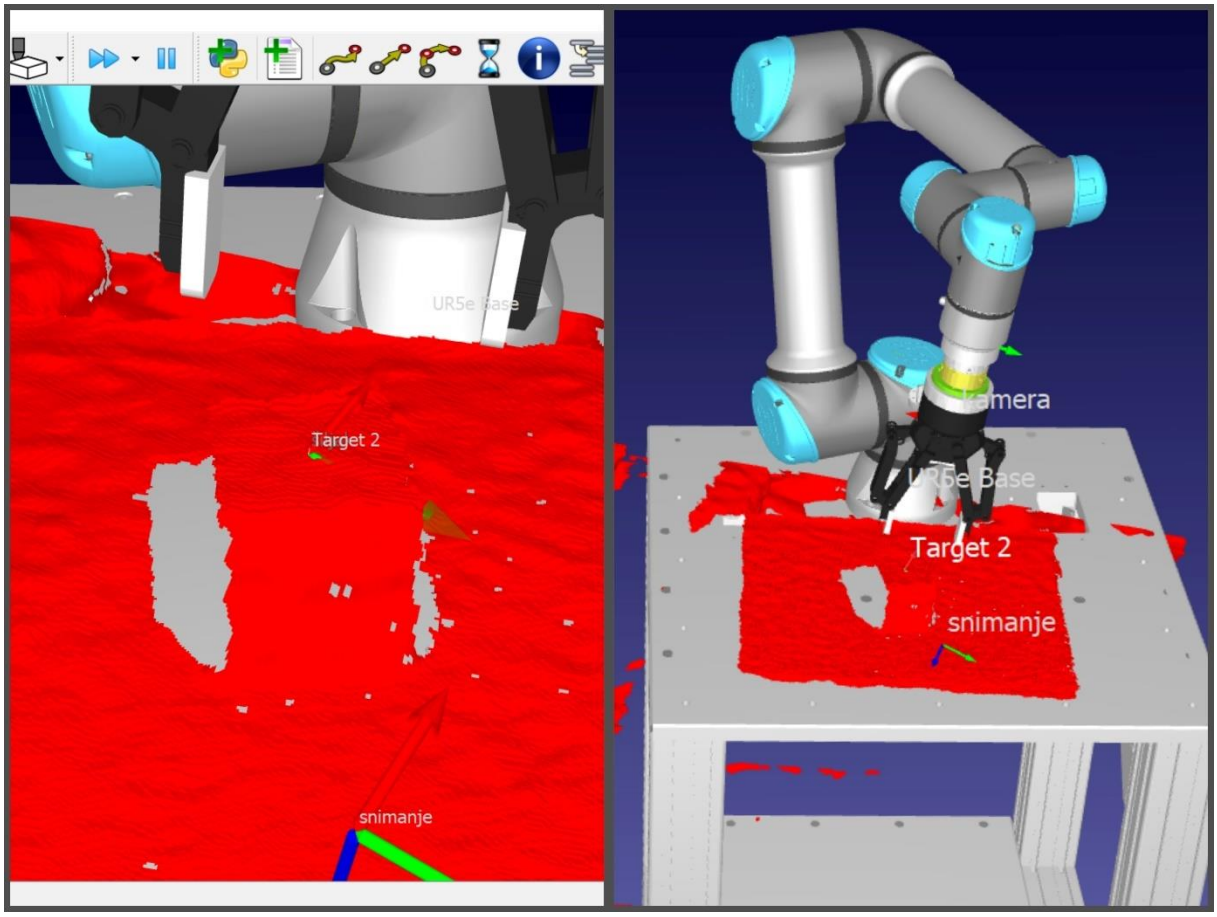
Nakon što smo pozicionirali predmet (crna kocka) u fokus kamere pritiskom tipke „E“ pokrećemo stvaranje oblaka točaka i konverziju u .obj format te se on dodaje u RoboDK u crvenoj boji.



Slika 25. Učitani 3D model predmeta u RoboDK

10.3. Dodavanje targeta

Na uspješno učitanom modelu crne kocke dodajemo metu u koju robot tj. šiljak u hvataljci mora otići. To radimo tako što otvorimo sučelje za dodavanje meta pritiskom kombinacije tipki CTRL+SHIFT+T. Stisnemo na površinu predmeta u RoboDK i meta je stvorena.



Slika 26. Dodavanje meta u RoboDK

a) dodana meta pod nazivom „Target 2“; b) robot s alatom (šiljak) u meti

10.4. Odlazak robota u metu

U sučelju za spajanje sa UR5e robotom unutar RoboDK odaberemo opciju „Move joints“ koja će, iz prethodne poze u kojoj se robot nalazio i iz koje smo snimali predmet, izvršiti naredbu za pomicanje robota u novu pozu koja je zapravo odlazak u metu koja je u ovom slučaju na površini crne kocke.

Zbog toga što kamera ne može dovoljno dobro snimiti površinu predmeta tj. 3D model predmeta nema glatku površinu ne možemo skroz precizno doći u kontakt s površinom crne spužvaste kocke u stvarnom svijetu. Ponekad šiljak dođe iznad kocke, a ponekad dođe u koliziju s kockom i upravo je to posljedica neravne površine snimljenog 3D modela.

Slika 27. Proces odlaska robota u kontakt s površinom crne kocke

1) robot u stanju mirovanja u pozi iz koje je snimao predmet; 2) robot nakon naredbe „Move joints“ pokrenute iz RoboDK kreće prema predmetu; 3), 4), 5), 6), 7), 8) robot se kreće prema predmetu; 9) robot dolazi u kontakt s površinom predmeta sa šiljkom u hvataljci

11. ZAKLJUČAK

U ovom završnom radu je realiziran zadatak digitalizacije robotskog radnog prostora u RoboDK softveru korištenjem Intel Realsense D435i 3D kamere pričvršćene na robotsku ruku UR5e robota.

Napravljena je skripta u Python programskom jeziku koja dohvaća oblak točaka, transformira ga u format koji je podržan od strane RoboDK te ga vizualizira unutar RoboDK simulacijskog softvera. Brzina digitalizacije zavisi od performansi korištenog računala tj. da se 3D model vizualizira u RoboDK. Kamera ima problema sa snimanjem predmeta s oštrim bridovima i ravnim površinama te predmeta malih dimenzija. Digitalizirana površina 3D modela postaje hrapava zato što je potreban veći broj točaka kako bi razmak između njih bio manji te bi kao rezultat površina bila ravna.

Kamera je kalibrirana pomoću MoveIt biblioteke u ROS-u. Kalibracija se pokazala kao najzahtjevniji dio ovog završnog rada jer uključuje niz postupaka koji se moraju napraviti da bi rezultati bili precizni i pouzdani. Kalibracija alata (šiljka) se pokazala jednostavnom zahvaljujući sučelju unutar Universal Robots Polyscope-a.

Uspješno je realizirano dovođenje alata u kontakt s površinom predmeta no preciznost nije svaki put bila na nivou zbog, već spomenute, hrapavosti površine 3D modela. To je moguće poboljšati tako što bi se koristila depth kamera veće rezolucije koja bi bolje mogla snimiti predmet.

LITERATURA

[1] https://www.universal-robots.com/products/ur5-robot/

[2] https://www.aivero.com/overview-of-depth-cameras/

[3] Garbev, Atanas, and Atanas Atanassov. "Comparative Analysis of RoboDK and Robot Operating System for Solving Diagnostics Tasks in Off-Line Programming." 2020 International Conference Automatics and Informatics (ICAI). IEEE, 2020.

[4] https://www.intelrealsense.com/depth-camera-d435i/

[5] https://unchainedrobotics.de/en/products/softwares-en/simulation-softwares-en/robodk

[6] Szeliski, Richard. Computer vision: algorithms and applications. Springer Nature, 2022.

[7] https://www.tm-robot.com/en/robot-vision-system/

[8] https://en.wikipedia.org/wiki/PLY_(file_format)

[9] Strobl, Klaus H., and Gerd Hirzinger. "Optimal hand-eye calibration." 2006 IEEE/RSJ international conference on intelligent robots and systems. IEEE, 2006.

[10] Kurose, James, and Keith Ross. "Computer networks: A top down approach featuring the internet." (2010).