Izgradnja 3D modela objekata pomoću robotske ruke i 3D kamere

Kovačević, Mihovil

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA

Sveučilišni studij

IZGRADNJA 3D MODELA OBJEKATA POMOĆU ROBOTSKE RUKE I 3D KAMERE

Završni rad

Mihovil Kovačević

Osijek, 2021.

SADRŽAJ

1. UVOD

Izgradnja 3D modela objekta odnosi se na proces stvaranja virtualnog trodimenzionalnog prikaza stvarnog objekta pomoću specijaliziranog programskog rješenja i opreme. 3D model objekta može odražavati veličinu, oblik i teksturu stvarnog objekta. 3D modeli sastavni su dio mnogih industrija, uključujući virtualnu stvarnost, video igre, 3D ispis, marketing, filmove te znanstvena i medicinska snimanja.

U ovom se radu izgradnja 3D modela objekta ostvaruje pomoću robotske ruke i 3D kamere. Pomakom robotske ruke kamera se dovodi na različite pozicije te se u svakoj poziciji uzimaju slike. Obradom tih slika dobiva se 3D model objekta.

Ovaj rad se sastoji od četiri poglavlja. U drugom poglavlju opisane su robotska ruka, kamera, programska podrška i koncepti korišteni za potrebe ovog završnog rada. U trećem poglavlju analizira se postupak transformacije oblaka točaka iz koordinatnog sustava kamere u koordinatni sustav baze robota iz općenitog pristupa i konkretnog pristupa programskog rješenja. Pri analizi programskog rješenja objašnjeno je i upravljanje robotskom rukom. Na kraju su prikazani rezultati 3D modela objekata, odnosno procijenjena je uspješnost provođenja cjelokupnog sustava. Zaključak rada dan je u četvrtom poglavlju.

1.1. Zadatak završnog rada

Zadatak završnog rada je izraditi programsko rješenje u ROS okviru za izgradnju 3D modela objekata pomoću robotske ruke i 3D kamere, koje se sastoji od modula za upravljanje robotskom rukom, te modula za spremanje i transformaciju 3D slike u koordinatni sustav baze robota. Izrađeni sustav eksperimentalno ispitati izgradnjom 3D modela nekog objekta.

2. KOMPONENTE ROBOTSKOG SUSTAVA

Za potrebe ovog završnog rada, korišten je robotski sustav koji se sastoji od robotske ruke Universal Robots UR5 i kamere Intel RealSense LiDAR Camera L515 povezanih unutar ROS-a. Komponente su detaljnije opisane u nastavku.

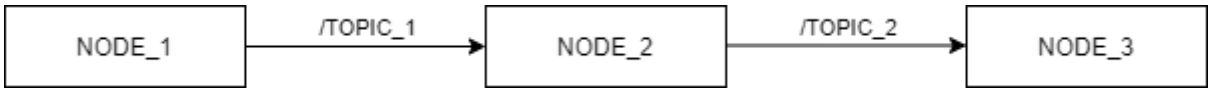
2.1. Robotski operacijski sustav

Robotski operacijski sustav (engl. *Robot Operating System,* ROS) je programski okvir namijenjen programiranju robota. Izvorno je prototipiran u laboratoriju za umjetnu inteligenciju sveučilišta Stanford, ali službeno ga je kreirao i razvio Willow Garage 2007. Trenutno ga dalje razvija i održava Open Source Robotics Foundation [1].

ROS se sastoji od četiri glavna svojstva: infrastrukture, alata, mogućnosti i ekosustava. Pod infrastrukturom se u osnovi podrazumijeva višeprocesna arhitektura te biblioteke koje su ugrađene i otvorenog koda. Od mnogih korisnih alata koje nudi, najčešće je naglasak na vizualizacijske. ROS dolazi s implementacijama puno uobičajenih algoritama robotike koji su potrebni svakom robotu, stoga nije potrebno razvijati vlastite. Za nadogradnju postojećih i kreiranje vlastitih koriste se službeno podržani programski jezici C++ i Python. Budući da je otvorenog koda i oko njega postoji velika zajednica, kod i paketi se međusobno dijele.

Problem kompatibilnosti starijih robota i novijih izdanja ROS-a rezultat je još uvijek značajnih promjena između inačica istog. Kompatibilnost softvera i biblioteke s određenom verzijom ROS-a također je važna. ROS nije prikladan za zadatke koji u potpunosti moraju biti skalabilni i sigurni ili ih treba pokretati na operacijskom sustavu koji nije Ubuntu.

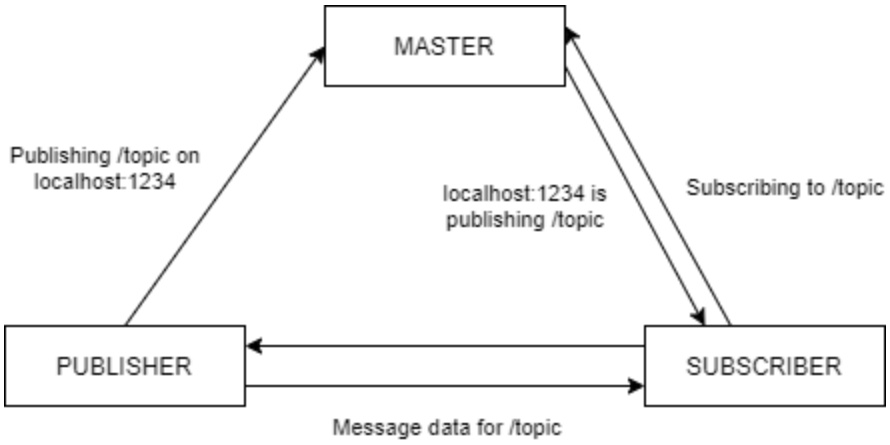
Programiranje u ROS-u potiče razdvajanje koda u različite module (komponente) od kojih svaki obavlja specifičan zadatak. Svaki takav program naziva se čvor (engl. *node*). Način na koji će različiti čvorovi međusobno komunicirati je kroz teme (engl. *topic*). Tema je imenovani kanal pomoću kojeg čvorovi mogu slati i primati podatke. Podaci koji se šalju preko tema nazivaju se poruke (engl. *message*). Sve poruke i sve teme moraju imati određen tip podatka. Na slici 2.1. prikazan je pojednostavljeni dijagram čvorova.



Slika 2.1. Pojednostavljeni dijagram čvorova

Bilo koji čvor može objaviti poruku na bilo kojoj temi i bilo koji čvor može se pretplatiti na bilo koju temu. Čvorovi koji objavljuju poruku na temu imaju ulogu izdavača (engl. *publisher*), a čvorovi koji se pretplaćuju na temu imaju ulogu pretplatnika (engl. *subscriber*). Ne postoji ograničenje broja čvorova koji mogu objaviti i / ili se pretplatiti na određenu temu, a isto tako čvor može objaviti i / ili se pretplatiti na više tema.

Poslužiteljski čvor (engl. *master*) je virtualni server koji nadgleda sve čvorove. Mora biti pokrenut kako bi cijeli ROS sustav funkcionirao ispravno. Ne obrađuje niti jedan podatak koji se objavljuje, već samo prati meta informacije, odnosno koji su čvorovi pokrenuti, koji čvorovi izdaju na koju temu, koji se čvorovi pretplaćuju na koju temu i koje su njihove IP adrese. IP adresa poslužiteljskog čvora je uvijek poznata, stoga se svaki izdavač i pretplatnik može povezati. Na slici 2.2. prikazan je koncept poslužitelja. U nastavku će biti objašnjen postupak koji se odvija neopaženo prilikom korištenja ovog koncepta.



Slika 2.2. Koncept poslužitelja

Izdavač obavještava poslužiteljski čvor da izdaje na određenu temu i šalje svoju IP adresu. Zatim pretplatnik zahtijeva pretplatu na spomenutu temu, pri čemu poslužiteljski čvor vraća informaciju pretplatniku da spomenuti izdavač s određenom IP adresom izdaje na tu temu. Pretplatnik će tada

uspostaviti peer-to-peer vezu s izdavačem te će izdavač slati poruke vezane uz spomenutu temu izravno pretplatniku.

2.2. LiDAR Camera L515 – Intel RealSense

Intel RealSense nudi najširi spektar tehnologija računalnog vida, od LiDAR-a (svjetlosno zamjećivanje i klasifikacija), mjerenja dubine i istodobne lokalizacije te mapiranja do autentifikacije lica. Uz Intel RealSense pakete za razvoj programa otvorenog koda, omogućeno je brzo i jednostavno ostvarenje željenog rješenja [2].

Intel RealSense LiDAR Camera L515 pogodna je za zadatke koji zahtijevaju podatke o dubini u visokoj razlučivosti i visokoj točnosti. Daje precizna volumetrijska mjerenja objekata te donosi dodatnu razinu preciznosti i točnosti u cijelom svom operativnom opsegu. U kombinaciji s kvalitetnom FHD RGB kamerom nudi rješenja za robusnije ručno skeniranje. Na slici 2.3. prikazana je razmatrana kamera, a u tablici 2.1. navedene su njezine specifikacije.

Tablica 2.1. Specifikacije modela LiDAR Camera L515

Naziv modela	**LiDAR Camera L515**
Domet	0.25m – 9m
Masa	100g
Vidno polje	70°H x 43°V x 55°D
Rezolucija RGB slike	1920 × 1080 @30fps
Rezolucija dubinske slike	1024 × 768 @30fps
Dimenzije	61mm x 25mm
Konekcija	USB 3.1
Programska podrška	Intel RealSense SDK

Slika 2.3. Intel RealSense LiDAR Camera L515

2.3. UR5 – Universal Robots

Universal Robots službeno su 2005. godine osnovali Esben Østergaard, Kasper Støy i Kristian Kassow s ciljem stvaranja dostupnosti robotske tehnologije malim i srednjim poduzećima. Osnivanje tvrtke omogućeno je ulaganjem Syddansk Innovation [3].

Universal Robots UR5 lagani je industrijski kolaborativni robot koji omogućuje automatizaciju ponavljajućih i/ili opasnih zadataka s nosivosti do 5kg. Ovaj robot opće namjene izrađen je imajući u vidu svestranost i prilagodljivost. Dizajniran je za integraciju u širok spektar primjena. Idealan je za optimizaciju kolaborativnih procesa male težine, poput podizanja i spuštanja predmeta te testiranja. Jednostavan je za programiranje, nudi brzu postavku, kolaborativan je i siguran.

UR5 je 6-osna robotska ruka. Na svih 6 osi moguća je rotacija od 360 stupnjeva. Šest osi čine baza, rame, lakat, zglob 1, zglob 2 i zglob 3. Na slici 2.4. prikazan je razmatrani robot, a u tablici 2.2. navedene su njegove specifikacije.

Tablica 2.2. Specifikacije modela UR5

Naziv modela	**UR5**
Domet	850mm
Nosivost	5kg
Promjer baze	149mm
Masa	20.6kg
Radna temperatura	0 – 50°C
Konekcija	Profinet EtherNet/IP, USB 2.0, USB 3.0

Potrošnja energije	200 – 570W
Programska podrška	Polyscope (grafičko korisničko sučelje), MoveIt paket

Slika 2.4. Universal Robots UR5

3. IZGRADNJA 3D MODELA

U ovom se radu pretpostavlja kamera montirana pri vrhu robotskog alata. Kamera je u odnosu na vrh alata robota nepomična. Baza robota je statična, odnosno pozicija ishodišta njezinog koordinatnog sustava se ne mijenja. U svrhu izgradnje 3D modela objekta, zglobovi robota se pomiču te se iz nekoliko različitih položaja uzimaju slike predstavljene oblakom točaka. Transformacijom svih oblaka točaka iz koordinatnog sustava kamere u koordinatni sustav baze robota, dobiva se cjeloviti 3D model objekta.

3.1. Transformacija oblaka točaka iz koordinatnog sustava kamere u koordinatni sustav baze robota

Transformacijom oblaka točaka iz koordinatnog sustava kamere u koordinatni sustav baze robota definirane su prostorne pozicije svake točke oblaka točaka u odnosu na bazu robota.

Takva transformacija se provodi slijedećim postupkom:

$${}^{A}P = {}^{A}T_{C} \cdot {}^{C}P$$

$${}^{R}P = {}^{R}T_{A} \cdot {}^{A}P$$

Pri čemu su:

${}^{C}P$ – Matrica vrijednosti prostornih pozicija svih točaka jednog oblaka točaka u koordinatnom sustavu kamere. Ova matrica ima četiri retka, odnosno svaki stupac predstavlja homogene koordinate jedne točke koje su proširene četvrtim elementom koji ima vrijednost 1 kako bi se matrica mogla množiti s matricom homogene transformacije s lijeva ${}^{A}T_{C}$.

$$\begin{bmatrix} x & \cdots \\ y & \cdots \\ z & \cdots \\ 1 & \cdots \end{bmatrix}$$

${}^{A}T_{C}$ – Matrica homogene transformacije koja sadrži vrijednosti rotacije i translacije kamere u odnosu na vrh alata robota.

$$\begin{bmatrix} r00 & r01 & r02 & tx \\ r10 & r11 & r12 & ty \\ r20 & r21 & r22 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

${}^{A}P$ – Matrica vrijednosti prostornih pozicija svih točaka jednog oblaka točaka u koordinatnom sustavu vrha alata robota. Ova matrica ima četiri retka, odnosno svaki stupac predstavlja homogene koordinate jedne točke koje su proširene četvrtim elementom koji ima vrijednost 1 kako bi se matrica mogla množiti s matricom homogene transformacije s lijeva ${}^{R}T_{A}$.

$$\begin{bmatrix} x & \cdots \\ y & \cdots \\ z & \cdots \\ 1 & \cdots \end{bmatrix}$$

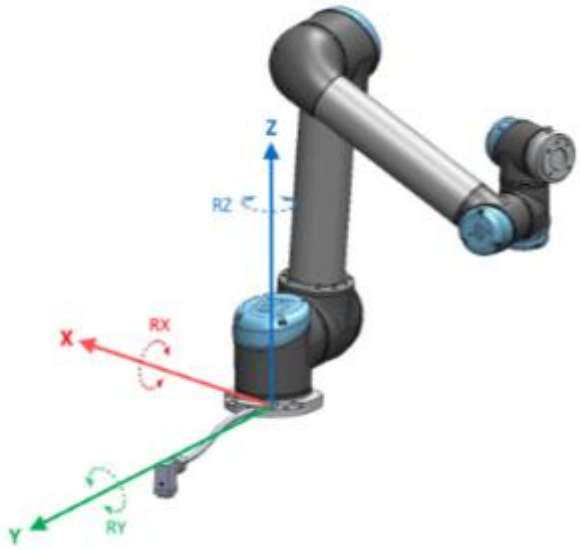
${}^{R}T_{A}$ – Matrica homogene transformacije koja sadrži vrijednosti rotacije i translacije vrha alata robota u odnosu na bazu robota.

$$\begin{bmatrix} r00 & r01 & r02 & tx \\ r10 & r11 & r12 & ty \\ r20 & r21 & r22 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

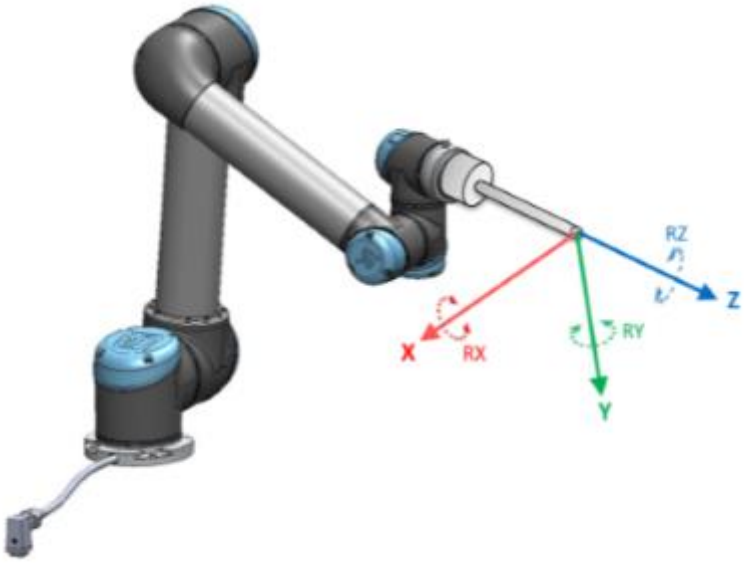
${}^{R}P$ – Matrica vrijednosti prostornih pozicija svih točaka jednog oblaka točaka u koordinatnom sustavu baze robota. Ova matrica ima četiri retka, odnosno svaki stupac predstavlja homogene koordinate jedne točke koje su proširene četvrtim elementom koji ima vrijednost 1. Za korištenjue ove matrice kao matrice oblaka točaka u koordinatnom sustavu baze robota, četvrti element svakog stupca koji ima vrijednost 1 mora biti uklonjen, te matrica mora biti transponirana.

$$\begin{bmatrix} x & \cdots \\ y & \cdots \\ z & \cdots \\ 1 & \cdots \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} x & y & z \\ \vdots & \vdots & \vdots \end{bmatrix}$$

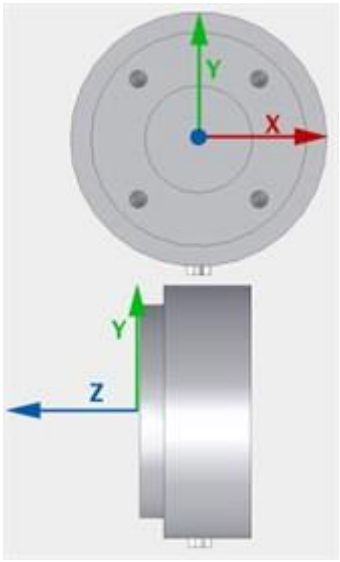
U nastavku na slikama prikazani su koordinatni sustavi korištenih komponeti, točnije na slici 3.1. prikazan je koordinatni sustav baze robota, na slikama 3.2. i 3.3. koordinatni sustav vrha alata robota te na slici 3.4. koordinatni sustav kamere. Na slici 3.5. prikazan je eksperimentalni postav na kojem su ucrtani svi predhodno spomenuti koordinatni sustavi.



Slika 3.1. Koordinatni sustav baze robota



Slika 3.2. Koordinatni sustav vrha alata robota



Slika 3.3. Bliži pogled koordinatnog sustava vrha alata robota

Slika 3.4. Koordinatni sustav kamere

Slika 3.5. Eksperimentalni postav

Za prethodno navedene modele matrica potrebno je prikupiti odgovarajuće stvarne vrijednosti kako bi se transformacija mogla provesti pravilno.

${}^{C}P$ – Vrijednosti matrice se dobivaju snimkom kamere. Takva matrica nije u odgovarajućem obliku za množenje s matricom homogene transformacije s lijeva, stoga svaki redak mora biti proširen četvrtim elementom koji ima vrijednost 1, te matrica mora biti transponirana.

$$\begin{bmatrix} x & y & z \\ \vdots & \vdots & \vdots \end{bmatrix} \quad \rightarrow \quad \begin{bmatrix} x & \cdots \\ y & \cdots \\ z & \cdots \\ 1 & \ldots \end{bmatrix}$$

$${}^{C}P = \begin{bmatrix} x & \cdots \\ y & \cdots \\ z & \cdots \\ 1 & \ldots \end{bmatrix}$$

${}^{A}T_{C}$ – Rotacijski dio vrijednosti se dobiva na temelju odnosa koordinatnih osi kamere i vrha alata robota, a translacijski dio vrijednosti se dobiva mjerenjem prostornih udaljenosti od središta leće kamere do središta vrha alata robota. S obzirom da je kamera u odnosu na vrh alata robota nepomična, jednom određene vrijednosti ove transformacijske matrice ostaju iste za svaku transformaciju.

$$ {}^{A}T_{C} = \begin{bmatrix} 0 & 1 & 0 & -0.11 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 1 & -0.04 \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

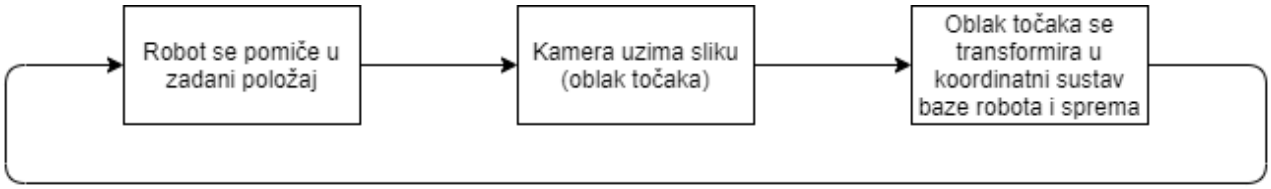
${}^{R}T_{A}$ – Vrijednosti se dobivaju izravno iz kinematike robota, odnosno pomoću programske funkcije koja vraća vrijednosti pozicije i orijentacije vrha alata robota u odnosu na bazu robota. Dobivene vrijednosti orijentacije u obliku kvaterniona potrebno je konvertirati u 3D rotacijsku matricu.

$$ {}^{R}T_{A} = \begin{bmatrix} r00 & r01 & r02 & tx \\ r10 & r11 & r12 & ty \\ r20 & r21 & r22 & tz \\ 0 & 0 & 0 & 1 \end{bmatrix} $$

3.2. Programsko rješenje

Programsko rješenje predstavljeno u nastavku pokriva automatizirana rješenja za prikupljanje oblaka točaka pomoću kamere Intel Realsense LiDAR Camera L515, manipulaciju robotske ruke Universal Robots UR5 te transformaciju oblaka točaka iz koordinatnog sustava kamere u koordinatni sustav baze robota.

Programsko rješenje izrađeno je u programskom jeziku Python uz korištenje nekoliko modula i biblioteka [4]. Podijeljeno je u dvije funkcije: glavna funkcija, te funkcija za spremanje i transformaciju oblaka točaka. Na slici 3.6. prikazan je dijagram toka programskog rješenja.



Slika 3.6. Dijagram toka programskog rješenja

Prilikom korištenja Python skripte unutar ROS okvira potrebno je navesti odgovarajući interpreter.

U primjeru 3.1. naveden je opisani programski kod.

```
#!/usr/bin/env python
```

Primjer 3.1. Interpreter

Korišteni moduli i biblioteke:

- rospy omogućava povezivanje s temama, uslugama i parametrima ROS-a [5]
- subprocess omogućava pokretanje novih procesa, spajanje na njihov ulaz/izlaz i dobivanje njihovih povratnih informacija
- glob omogućava pronalazak datoteka čiji nazivi odgovaraju određenom uzorku
- open3d omogućava brzi razvoj softvera koji se bavi 3D podacima [6]
- numpy omogućava korištenje opsežnih matematičkih funkcija, matematičkih konstanti, generatora slučajnih brojeva, postupaka linearne algebre i slično [7]
- moveit_commander između ostalih pruža i klasu MoveGroupCommander koja omogućava manipulaciju robota [8]

U primjeru 3.2. naveden je opisani programski kod.

```
import rospy
import subprocess
import glob
import open3d
import numpy
import moveit_commander
```

Primjer 3.2. Moduli i biblioteke

Zadatak glavne funkcije je pomicanje robotske ruke u pozicije koje omogućuju kameri dohvaćanje slika objekta. Na početku je potrebno inicijalizirati čvor i postaviti varijable group_name, move_group i joint_goal kako bi se omogućila komunikacija s robotom, odnosno zadavanje pozicija, pomicanje zglobova, zaprimanje informacija o trenutnoj poziciji i slično.

Svakom zglobu robotske ruke zadaje se pozicija izražena u radijanima te se poziva naredba za izvršenje pomicanja zglobova. Nakon što je robot postavljen u željenu poziciju, poziva se funkcija za spremanje i transformaciju oblaka točaka. Ponavlja se isto s različitim vrijednostima za poziciju zglobova kako bi se dobilo više različitih slika.

Vrijednosti pozicija zglobova za tri slučaja navedena u kodu odabrani su kao pozicije koje kameri omogućuju dohvaćanje zadovoljavajućih slika objekta za potrebe ovog rada, nakon ručnog ispitivanja više različitih pozicija zglobova pri čemu je objekt uvijek bio jedanko pozicioniran u odnosu na robota. Pri tome se podrazumijeva da je objekt postavljen na točno određenu poziciju u radnom prostoru robota.

U primjeru 3.3. naveden je opisani programski kod.

```
def create_3d_model():
    rospy.init_node('robot_control', anonymous=True)
    group_name = 'manipulator'
    move_group = moveit_commander.MoveGroupCommander(group_name)
    joint_goal = move_group.get_current_joint_values()

    joint_goal[0] = 0
    joint_goal[1] = -11*numpy.pi/36
    joint_goal[2] = 5*numpy.pi/12
    joint_goal[3] = -numpy.pi/6
    joint_goal[4] = -11*numpy.pi/36
    joint_goal[5] = numpy.pi/2

    move_group.go(joint_goal, wait=True)
    move_group.stop()

    rospy.sleep(1)

    save_and_transform_point_cloud('first')

    joint_goal[0] = 0
    joint_goal[1] = -25*numpy.pi/36
    joint_goal[2] = -5*numpy.pi/12
    joint_goal[3] = -8*numpy.pi/9
    joint_goal[4] = 11*numpy.pi/36
    joint_goal[5] = numpy.pi/2

    move_group.go(joint_goal, wait=True)
    move_group.stop()

    rospy.sleep(1)

    save_and_transform_point_cloud('second')

    joint_goal[0] = 4*numpy.pi/9
    joint_goal[1] = -5*numpy.pi/36
    joint_goal[2] = numpy.pi/18
    joint_goal[3] = -numpy.pi/18
    joint_goal[4] = -19*numpy.pi/36
```

```
    joint_goal[5] = numpy.pi/2

    move_group.go(joint_goal, wait=True)
    move_group.stop()

    rospy.sleep(1)

    save_and_transform_point_cloud('third')
```

Primjer 3.3. Glavna funkcija

S obzirom da je funkcija za spremanje i transformaciju oblaka točaka kompleksnija, u nastavku će biti objašnjena postupno.

Spomenuta funkcija ima parametar 'name' koji će se koristiti pri imenovanju elemenata kako bi funkcija mogla biti opetovano pozvana i davati jedinistvene nazive za svaki spremljeni element.

Za spremanje oblaka točaka koristi se predefinirani čvor pointcloud_to_pcd iz paketa pcl_ros. Čvor se pretplati na željenu temu ROS-a, u ovom slučaju tema korištene kamere i sprema poruke oblaka točaka u .pcd datoteke. Pomoću parametra ~prefix(str) dodaje se željeni prefiks imenu spremljene .pcd datoteke.

Spomenuti čvor koji je namijenjen pokretanju iz terminala može se pokrenuti pomoću Python skripte koristeći naredbu Popen() iz biblioteke subprocess. S obzirom da se nakon pokretanja čvor izvodi beskonačno dugo, potrebno je nakon dovoljno prikupljenih podataka prekinuti izvođenje naredbom kill().

U primjeru 3.4. naveden je opisani programski kod.

```
def save_and_transform_point_cloud(name):
    save = subprocess.Popen(['rosrun', 'pcl_ros', 'pointcloud_to_pcd', 'input:=/c
    amera/depth/color/points', '_prefix:={0}'.format(name)])
    rospy.sleep(1)
    save.kill()
```

Primjer 3.4. Funkcija za spremanje i transformaciju – prvi dio

Idući korak je spremanje trenutne pozicije i orijentacije vrha alata robota i kreiranje ${}^{R}T_{A}$ matrice.

Dobivene vrijednosti orijentacije u obliku kvaterniona potrebno je konvertirati u 3D rotacijsku matricu. Dobivene vrijednosti pozicije ne zahtijevaju izmjene, odnosno direktno predstavljaju translaciju. Zatim se ${}^{R}T_{A}$ matrica stvara prema modelu iz prethodnog poglavlja.

U primjeru 3.5. naveden je opisani programski kod.

```
    group_name = 'manipulator'
    move_group = moveit_commander.MoveGroupCommander(group_name)
    current_pose = move_group.get_current_pose().pose

    quaternion = [current_pose.orientation.w, current_pose.orientation.x, current
    _pose.orientation.y, current_pose.orientation.z]
    translation = [current_pose.position.x, current_pose.position.y, current_pose
    .position.z]

    q0 = quaternion[0]
    q1 = quaternion[1]
    q2 = quaternion[2]
    q3 = quaternion[3]

    r00 = 2 * (q0 * q0 + q1 * q1) - 1
    r01 = 2 * (q1 * q2 - q0 * q3)
    r02 = 2 * (q1 * q3 + q0 * q2)

    r10 = 2 * (q1 * q2 + q0 * q3)
    r11 = 2 * (q0 * q0 + q2 * q2) - 1
    r12 = 2 * (q2 * q3 - q0 * q1)

    r20 = 2 * (q1 * q3 - q0 * q2)
    r21 = 2 * (q2 * q3 + q0 * q1)
    r22 = 2 * (q0 * q0 + q3 * q3) - 1

    tx = translation[0]
    ty = translation[1]
    tz = translation[2]

    RTA = numpy.array([[r00, r01, r02, tx],
                       [r10, r11, r12, ty],
                       [r20, r21, r22, tz],
                       [0, 0, 0, 1]])
```

Primjer 3.5. Funkcija za spremanje i transformaciju – drugi dio

Iz prethodno spremljenog oblaka točaka potrebno je izdvojiti točke u obliku matrice.

Pomoću funkcije glob() iz biblioteke glob dohvaća se odgovarajući naziv prethodno spremljenog oblaka točaka. Funkcija read_point_cloud() iz biblioteke open3d.io čita taj oblak točaka i tada je moguće izdvojiti točke u obliku matrice pomoću funkcije asarray() iz biblioteke numpy. Ujedno se izvadaju i boje u obliku matrice koje će se na kraju pridodati konačnom rezultatu.

U primjeru 3.6. naveden je opisani programski kod.

```
    filename = glob.glob('{0}*.pcd'.format(name))[0]
    pcd = open3d.io.read_point_cloud(filename)
    xyz = numpy.asarray(pcd.points)
    rgb = numpy.asarray(pcd.colors)
```

Primjer 3.6. Funkcija za spremanje i transformaciju – treći dio

Nakon izvdojenih potrebnih podataka, spremljene oblake točaka se uklanja jer nemaju više svoju ulogu.

U primjeru 3.7. naveden je opisani programski kod.

```
    while glob.glob('{0}*.pcd'.format(name)):
        filename = glob.glob('{0}*.pcd'.format(name))[0]
        subprocess.run(['rm', '-r', filename])
```

Primjer 3.7. Funkcija za spremanje i transformaciju – četvrti dio

Poznato je kako bi mogli množiti dvije matrice, broj stupaca prve matrice mora biti jednak broju redaka druge matrice.

Matrica koja sadrži izdvojene točke iz prethodno spremljenog oblaka točaka nije u odgovarajućem obliku za množenje s matricom homogene transformacije s lijeva ${}^{A}T_{C}$, stoga svaki redak mora biti proširen četvrtim elementom koji ima vrijednost 1, te matrica mora biti transponirana koristeći funkcije iz biblioteke numpy. Tada predstavlja matricu ${}^{C}P$.

Nakon učinjenog, matrica ${}^{C}P$ će imati četiri retka što zadovoljava uvjet množenja matrica jer matrica ${}^{A}T_{C}$ ima četiri stupca. Vrijednosti matrice ${}^{A}T_{C}$ su ručno izmjerene i zadovoljavaju svaki slučaj, odnosno uvijek su iste.

U primjeru 3.8. naveden je opisani programski kod.

```
    xyzk = numpy.insert(xyz, 3, 1, axis=1)
    CP = numpy.transpose(xyzk)

    ATC = [[0, 1, 0, -0.11], [-1, 0, 0, 0], [0, 0, 1, -0.04], [0, 0, 0, 1]]

    AP = numpy.dot(ATC, CP)
```

Primjer 3.8. Funkcija za spremanje i transformaciju – peti dio

Nakon dobivenog rezultata množenjem matrica ${}^{R}T_{A}$ i ${}^{A}P$, potrebno je ukloniti prethodno dodani niz vrijednosti 1 i transponirati kako bi dobili matricu koja odgovara modelu matrice ${}^{R}P$, odnosno kako bi ju mogli koristiti za kreiranje oblaka točaka u koordinatnom sustavu baze robota.

U primjeru 3.9. naveden je opisani programski kod.

```
    RP = numpy.dot(RTA, AP)
    RP = RP[:-1]
    RP = numpy.transpose(RP)
```

Primjer 3.9. Funkcija za spremanje i transformaciju – šesti dio

Za objekt oblaka točaka koristi se PointCloud() iz biblioteke open3d.geometry. Atributu points se pridružuje prethodno izračunata matrica ${}^{R}P$, a atributu colors se pridružuje prethodno izdvojena matrica boja rgb. Pomoću naredbe write_point_cloud() iz biblioteke open3d.io stvara se konačni rezultat, točnije oblak točaka u koordinatnom sustavu baze robota.

Dobiveni oblak točaka sprema se kao .ply datoteka radi mogućnosti vizualizacije u programima za obradu 3D modela. Polygon File Format (.ply) predstavlja 3D format datoteke koji pohranjuje grafičke objekte opisane kao zbirka poligona.

U primjeru 3.10. naveden je opisani programski kod.

```
    ply = open3d.geometry.PointCloud()
    ply.points = open3d.utility.Vector3dVector(RP)
    ply.colors = open3d.utility.Vector3dVector(rgb)
    open3d.io.write_point_cloud('RP_{0}.ply'.format(name), ply)
```

Primjer 3.10. Funkcija za spremanje i transformaciju – sedmi dio

3.3. Eksperimentalna evaluacija

Razvijeni sustav je ispitan pokusima u kojima je objekt postavljen na stol ispred robota u poziciju koja je prethodno odabrana kao prikladna u odnosu na pozicije u koje će robot postaviti kameru za dohvaćanje slika tog objekta. Nakon provođenja programskog rješenja spremljene su tri datoteke s ekstenzijom .ply koje sadrže matrice obojenih 3D oblaka točaka transformiranih u koordinatni sustav baze robota.

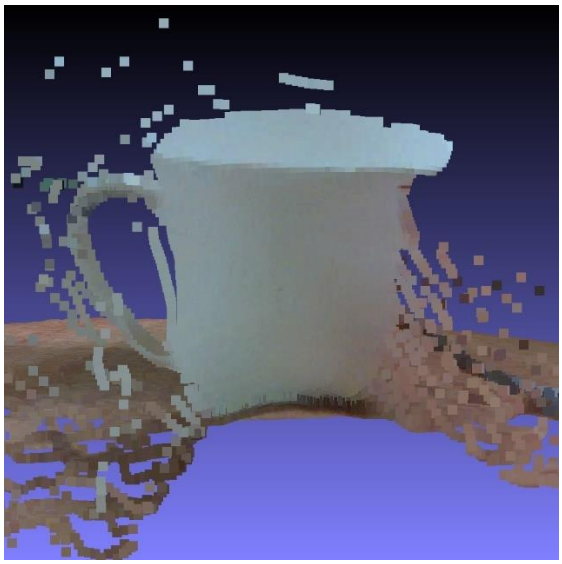
Transformirane točke vizualiziraju se u programu MeshLab [9]. MeshLab je sustav otvorenog koda za obradu i uređivanje 3D mreža trokuta. Pruža skup alata za uređivanje, čišćenje, ispravljanje, pregled, iscrtavanje, teksturiranje i pretvaranje mreža. Nudi značajke za obradu

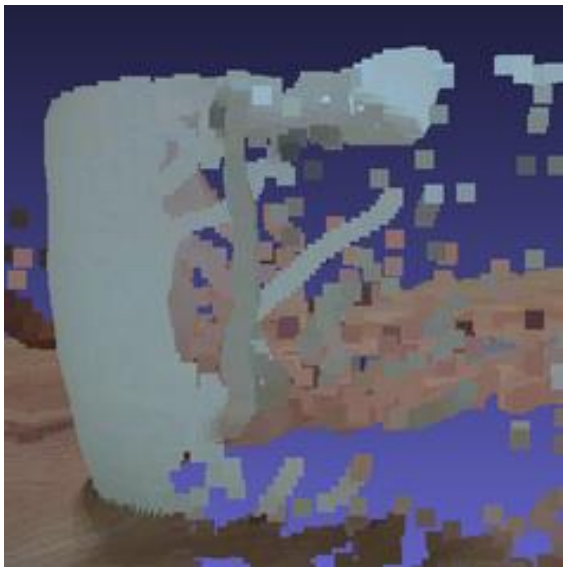
sirovih podataka proizvedenih pomoću alata/uređaja za 3D digitalizaciju i za pripremu modela za 3D ispis. Dobivene datoteke s ekstenzijom .ply potrebno je otvoriti programom MeshLab kako bi dobili krajnji rezultat 3D modela.

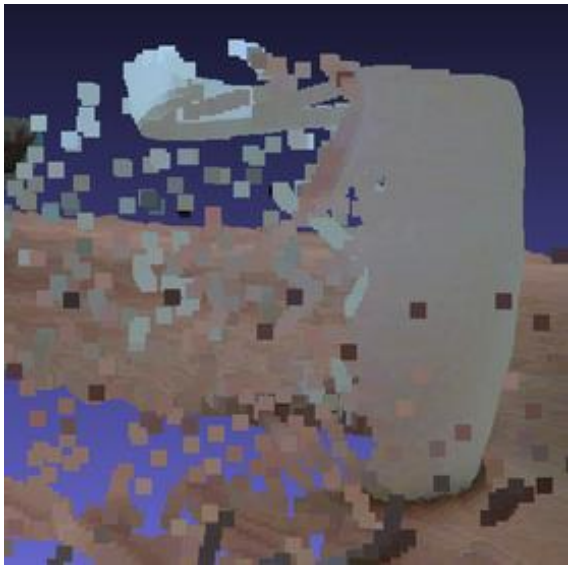
Za uzimanje svakog prikazanog oblaka točaka kamera je morala biti na jako maloj udaljenosti u odnosu na objekt, što zahtijeva veliku preciznost pri prikupljanju vrijednosti za transformacijske matrice korištene u programskom rješenju. Posljedica nemogućnosti ostvarenja tako velike preciznosti su vidljiva odstupanja u poziciji oblaka točaka koji zajedno predstavljaju 3D model objekta.

Promatrani objekt je šalica – u nastavku su prikazani rezultati. Svaki oblak točaka prvo je prikazan zasebno, a zatim spojen s ostalim – u oba slučaja s prednje strane, stražnje strane i bočno. U zasebnim prikazima vidljivi su samo dijelovi okrenuti prema kameri, dok se na cjelovitom modelu vide sve stjenke objekta.

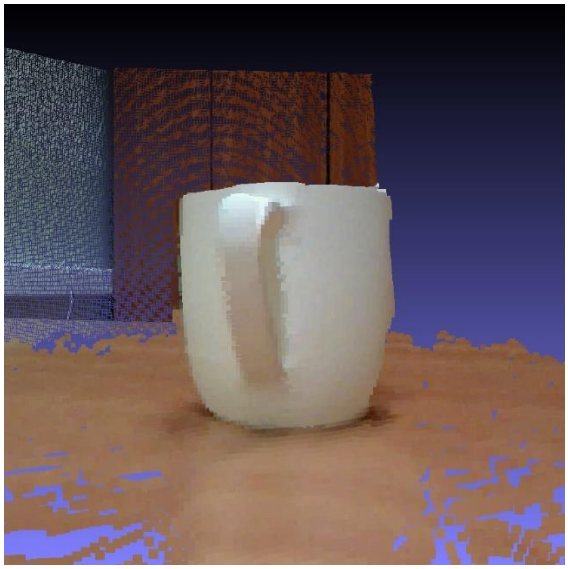
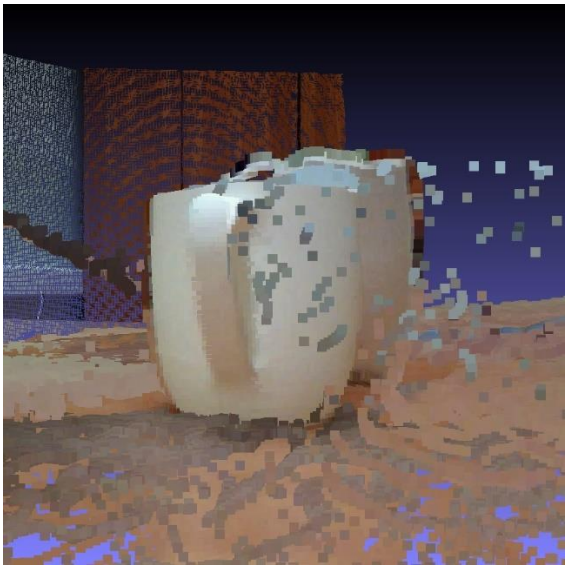
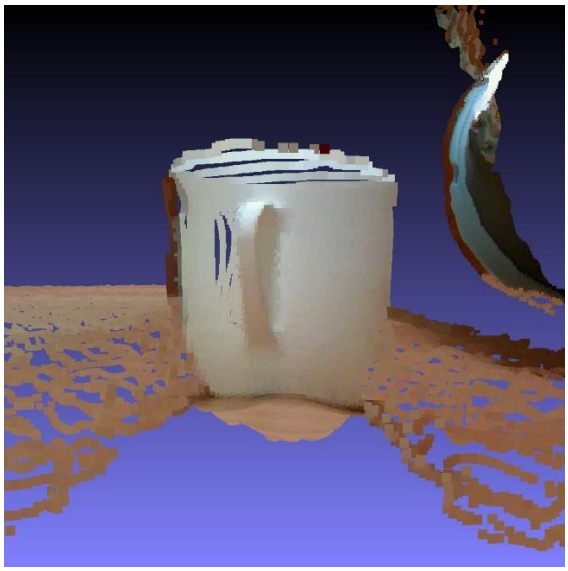
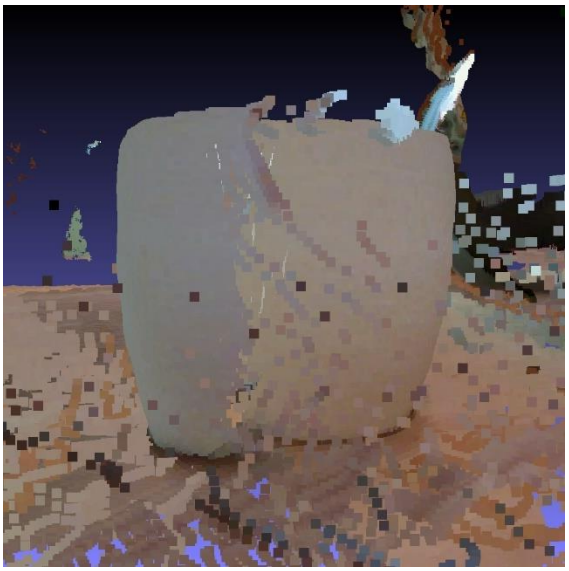
Idućih osam slika predstavlja prvi oblak točaka zasebno i spojen s ostalim – redom s prednje strane, stražnje strane i bočno.

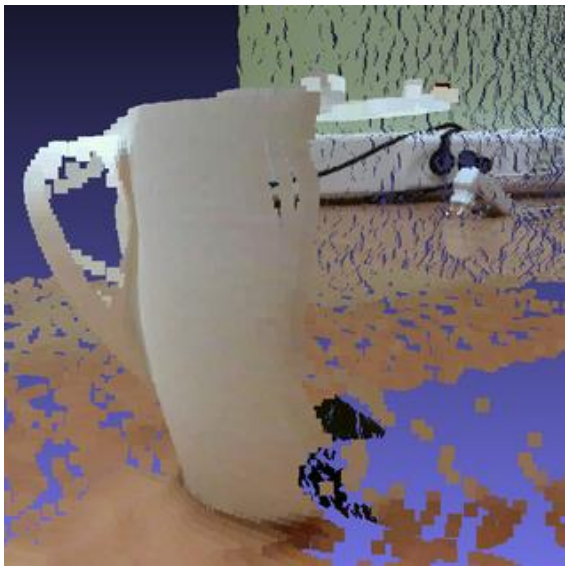


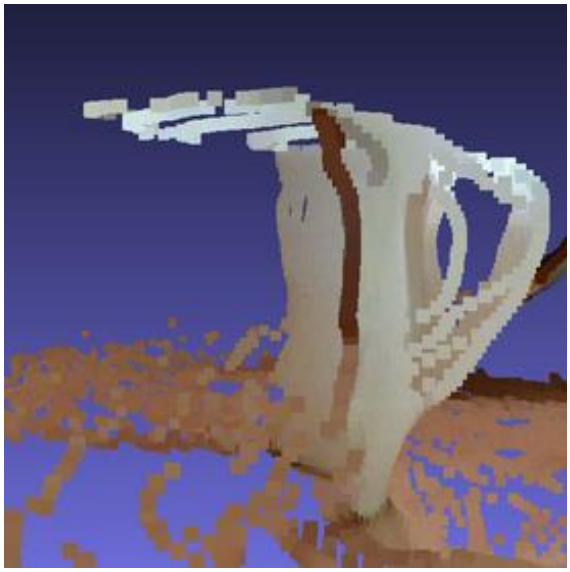




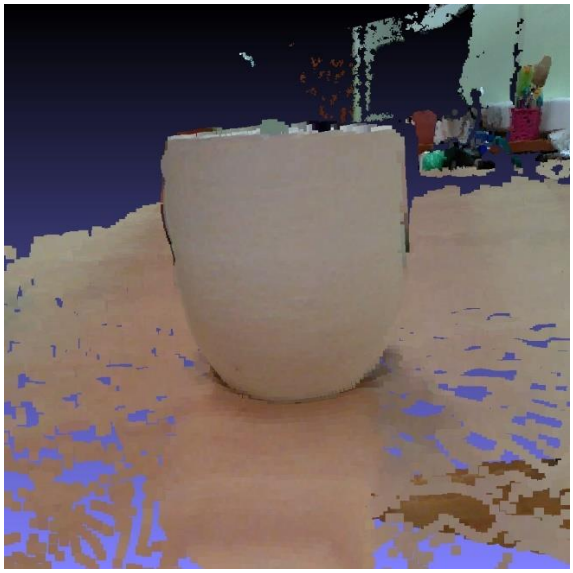

Idućih osam slika predstavlja drugi oblak točaka zasebno i spojen s ostalim – redom s prednje strane, stražnje strane i bočno.

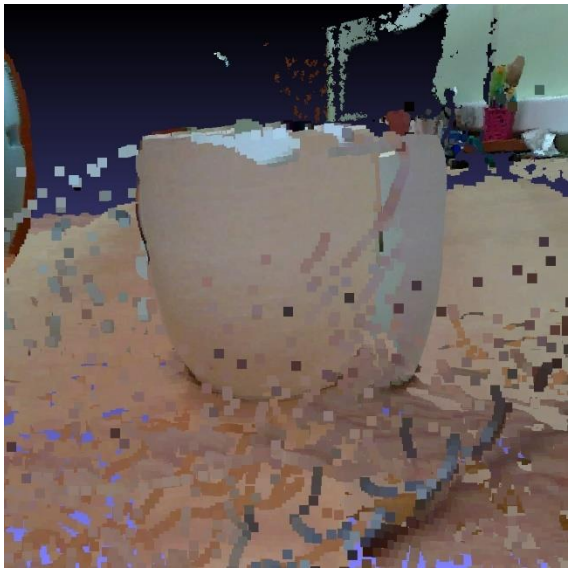





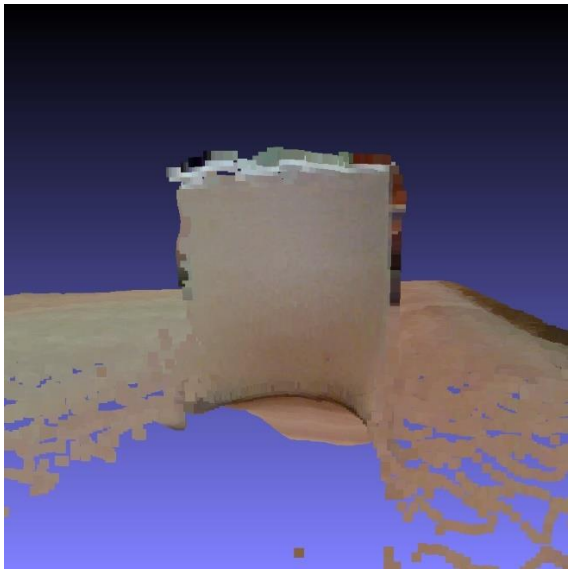


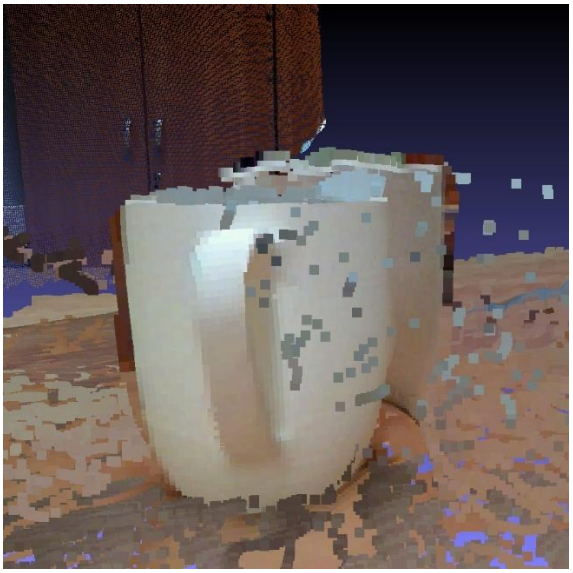


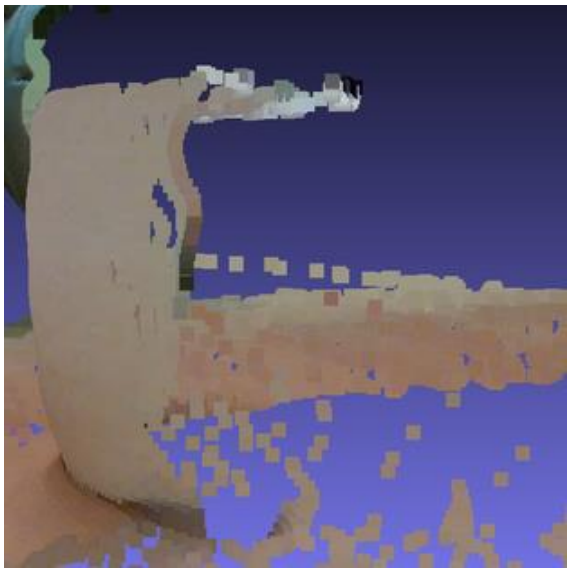

Idućih osam slika predstavlja treći oblak točaka zasebno i spojen s ostalim – redom s prednje strane, stražnje strane i bočno.

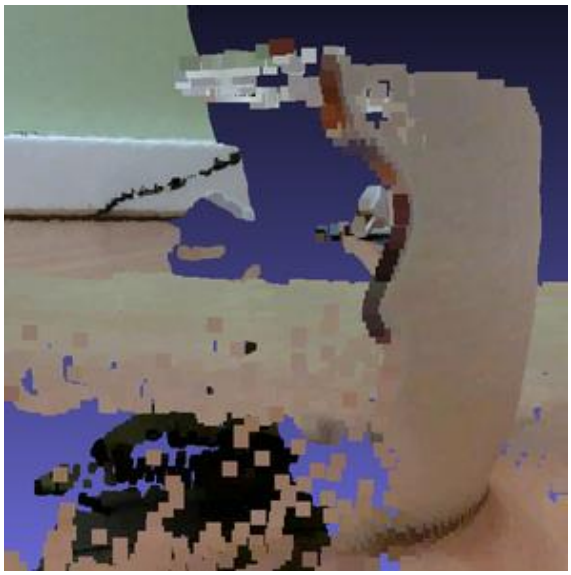













4. ZAKLJUČAK

U ovom radu opisana je teorijska podloga i programsko rješenje za kreiranje 3D modela objekata uz pomoć robotske ruke i 3D kamere. Sustav je temeljen na ROS platformi uz programsko rješenje izrađeno u programskom jeziku Python, pri čemu je korišteno šest biblioteka - rospy za inicijalizaciju čvora i komunikaciju s čvorom robota, subprocess za pokretanje naredbe terminala koja sprema sliku kamere, glob za pronalazak datoteka čiji nazivi odgovaraju određenom uzorku kako bi se automatizirao proces transformacije, open3d za izdvajanje točaka i boja iz oblaka točaka, numpy za aritmetičke operacije s matricama, te moveit_commander za manipulaciju robota i dobivanje vrijednosti pozicije i orijentacije vrha alata robota.

Razvijeni sustav podrazumijeva da je objekt pozicioniran na određeno mjesto u odnosu na robota te da objekt mora biti takav da stane u vidno polje kamere, u suprotnom je potrebno promijeniti poziciju i orijentaciju kamere. Proces spremanja i transformacije oblaka točaka u koordinatni sustav baze robota je automatiziran.

Provedeni pokusi pokazali su značajna odstupanja kod transformacije oblaka točaka snimljenih iz različitih kutova u zajednički koordinatni sustav. Bolji rezultati postigli bi se primjenom odgovarajuće metode registracije oblaka 3D točaka, kao što je iterativni algoritam najbliže točke (engl. *Iterative Closest Point*, ICP) [10].

LITERATURA

[1] ROS, Open Source Robotics Foundation, dostupno na: https://www.ros.org/ [7. srpnja 2021.].

[2] Intel RealSense, dostupno na: https://www.intelrealsense.com/ [7. srpnja 2021.].

[3] Universal Robots, dostupno na: https://www.universal-robots.com/ [7. srpnja 2021.].

[4] Python documentation, dostupno na: https://docs.python.org/ [23. kolovoza 2021.].

[5] rospy documentation, dostupno na: http://wiki.ros.org/rospy [23. kolovoza 2021.].

[6] Open3D, dostupno na: http://www.open3d.org/ [30. kolovoza 2021.].

[7] NumPy, dostupno na: https://numpy.org/ [30. kolovoza 2021.].

[8] MoveIt, dostupno na: https://moveit.ros.org/ [30. kolovoza 2021.].

[9] MeshLab, dostupno na: https://www.meshlab.net/ [4. rujna 2021.].

[10] P. J. Besl i N. D. McKay, A Method for Registration of 3-D Shapes, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, svez. 14, br. 2, 1992.

SAŽETAK

Razmatra se sustav za kreiranje 3D modela objekata. Sačinjavaju ga fizičke komponente robotska ruka Universal Robots UR5 i 3D kamera Intel RealSense LiDAR L515, te programsko rješenje realizirano korištenjem programskog jezika Python koji su povezani unutar ROS okvira. Konačni 3D model objekta predstavlja skup oblaka točaka transformiranih iz koordinatnog sustava kamere u koordinatni sustav baze robota. Rezultati eksperimentalne evaluacije prikazani su obojenim 3D oblakom točaka dobivenim fuzijom tri snimke promatranim iz različitih kutova.

Ključne riječi: 3D kamera, 3D model, 3D oblak točaka, robotska ruka, ROS

ABSTRACT

Creating 3d models of objects using a robotic hand and a 3d camera

A system for creating 3D models of objects is considered. It consists of the physical components robotic arm Universal Robots UR5 and 3D camera Intel RealSense LiDAR L515, and a software solution implemented using the Python programming language that are connected within the ROS framework. The final 3D model of the object represents a set of point clouds transformed from a camera coordinate system to a robot base coordinate system. The results of the experimental evaluation are presented by a colored 3D point cloud obtained by fusion of three RGB-D images acquired from different angles.

Keywords: 3D camera, 3D model, 3D point cloud, robotic arm, ROS

ŽIVOTOPIS

Mihovil Kovačević rođen je 11. rujna 1998. godine u Slavonskom Brodu. Odrastao je u Županji gdje je i pohađao osnovnu školu. Prirodoslovno matematički smjer Gimnazije Županja upisuje 2013. godine te završava 2017. godine. Iste godine upisuje preddiplomski sveučilišni studij Računarstvo na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija Osijek.