Registracija medicinskih 3D snimki rana korištenjem TEASER++ biblioteke

Gavran, Iva

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

Sveučilišni studij

REGISTRACIJA MEDICINSKIH 3D SNIMKI RANA KORIŠTENJEM TEASER++ BIBLIOTEKE

Završni rad

Iva Gavran

Osijek, 2021.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 10.09.2021.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime studenta:	Iva Gavran
Studij, smjer:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4307, 24.09.2018.
OIB studenta:	08550345191
Mentor:	Izv. prof. dr. sc. . Damir Filko
Sumentor:	
Sumentor iz tvrtke:	
Naslov završnog rada:	Registracija medicinskih 3D snimki rana korištenjem TEASER++ biblioteke
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Predložena ocjena završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 3 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	10.09.2021.
Datum potvrde ocjene Odbora:	22.09.2021.

Potpis mentora za predaju konačne verzije rada u Studentsku službu pri završetku studija:	Potpis:
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 30.09.2021.

Ime i prezime studenta:	Iva Gavran
Studij:	Preddiplomski sveučilišni studij Računarstvo
Mat. br. studenta, godina upisa:	R4307, 24.09.2018.
Turnitin podudaranje [%]:	9

Ovom izjavom izjavljujem da je rad pod nazivom**: Registracija medicinskih 3D snimki rana korištenjem TEASER++ biblioteke**

izrađen pod vodstvom mentora Izv. prof. dr. sc. . Damir Filko

i sumentora

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD

Liječenje medicinskih rana nije nimalo jednostavno. Proces analize rane nerijetko je netočan i subjektivan zbog ljudskog faktora u njemu. Ako se iz prvog pokušaja ne utvrdi ispravan tretman rane, liječenje se može odužiti te rezultirati velikim novčanim izdatcima. Kreiranje automatskog sustava, koji bi omogućio točnu dijagnostiku, a time i preciznu rekonstrukciju rane, isključilo bi probleme netočnosti i subjektivnosti. Proces analize odvijao bi se brže, a rezultati bi bili pouzdaniji i precizniji. Takav sustav moguće je stvoriti pomoću robotskog manipulatora i 3D kamera [1]. Međutim, kvalitetna registracija 3D snimki može biti kreirana isključivo korištenjem algoritma koji će dovoljno dobro, brzo i precizno poravnati oblake točaka. Algoritam koji trenutno pokazuje najbolju izvedbu naziva se TEASER++ te će se on primjenjivati u ovom radu prilikom registracije snimki rana.

TEASER++ najnovija je metoda poravnanja 3D oblaka točaka. Pokazuje neusporedivo točnije rezultate u odnosu na prijašnje metode koje su bile robusne, ali prespore ili pak brze, ali nedovoljno efikasne. TEASER++ po prvi puta objedinjuje točnost i brzinu. Nadalje, rješava registraciju globalno, ne oslanjajući se na inicijalnu pretpostavku, te tolerira ekstreman broj odudarajućih točaka (engl. outliners). Njegova izvedba dominantna je nad svim do sada primjenjivanim algoritmima [2].

Ovaj završni rad započinje teorijskom obradom registracije 3D oblaka točaka i nekih od algoritama koji su se do sada koristili. Nakon toga, obrađuje se TEASER++ algoritam te opisuje njegova implementacija. Na posljetku, prikazuje se eksperimentalni dio registracije snimki rana te analiziraju dobiveni rezultati.

Zbog cjelokupne praktičnosti i prilagodbe raznim okruženjima, eksperimentalni dio rada izveden je na operacijskom sustavu Linux koristeći programski jezik Python.

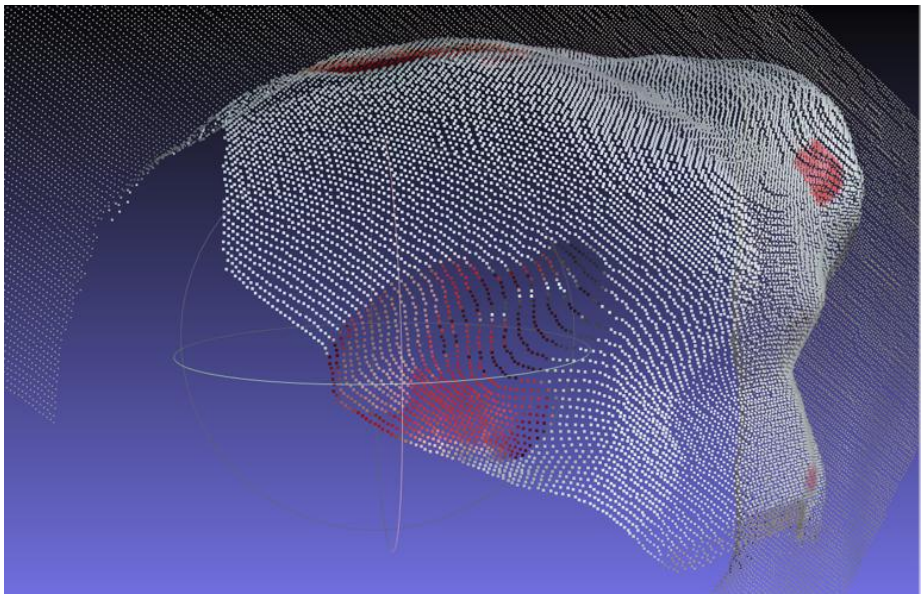
1.1. Zadatak završnog rada

Zadatak završnog rada je pokušati registrirati medicinske 3D snimke rana korištenjem TEASER++ biblioteke u Linux operacijskom sustavu koristeći Python programski jezik te analizirati prednosti i nedostatke navedene registracije.

2. REGISTRACIJA 3D OBLAKA TOČAKA

2.1. Oblak točaka

Oblak točaka rezultat je izmjere 3D kamerom kojom se dobije digitalna snimka predstavljena skupom točaka u 3D prostoru koje prikazuju vanjsku površinu izmjerenog objekta [3]. Svaka točka predstavljena je x, y i z koordinatom. Oblaci kreirani 3D kamerama strukturirani su te se lako može odrediti susjedna točka ili pak iz koje je točke oblaka dobivena koja točka u prostoru RGB slike [4]. Primjer oblaka točaka prikazanog u programu MeshLab vidljiv je u nastavku na slici 2.1.. Slika prikazuje scenu rane korištenu u eksperimentalnom dijelu ovog rada.



Slika 2.1. Primjer oblaka točaka

2.2. Registracija oblaka točaka

Registracija oblaka točaka svođenje je nekoliko oblaka točaka, snimljenih iz nekoliko perspektiva, u zajednički koordinatni sustav i proračunavanje transformacijske matrice koja to omogućava. Izvodi se snimanjem istog prostora između stajališta i njihovim prepoznavanjem u procesu obrade [3].

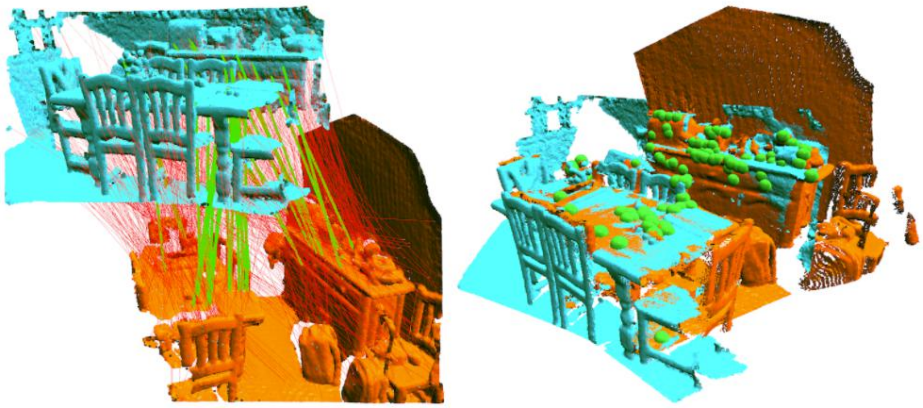
Drugim riječima, registracija rješava problem procjene transformacije između dvaju oblaka. Procjenjuje transformacijsku matricu te tako omogućuje spajanje parcijalnih dijelova istih 3D scena u kompletni 3D oblak točaka. Nadalje, omogućuje razne oblike transformacije

poput skaliranja, rotiranja ili translatiranja. Time zauzima jednu od ključnih uloga u aplikacijama koje su vezane za računalni vid [5]. Budući da je oblak točaka postao primarni format podatka koji prezentira 3D svijet, a senzori imaju limitirane sposobnosti, upravo se od registracijskih algoritama zahtjeva generiranje velike 3D scene.

Primjena registracije je razna. Prvenstveno se primjenjuje u 3D rekonstrukciji koja je aktualna u autonomnoj vožnji te robotici. Nadalje, dio registracije je i 3D lokalizacija. Locirati poziciju agenta u 3D prostoru vrlo je važno, primjerice za procjenu udaljenosti automobila. Na kraju, registracija je iznimno korisna kod procjene položaja što poprilično doprinosi u donošenju odluka u robotici [5].

Upravo zbog raznolike primjene, registracija je postala vrlo zanimljiva u sferama robotike i računalnog vida. Samim time, postala je i jedan od fundamentalnih problema zbog pogrešaka do kojih može doći. U praksi je izrazito problematično to što su korespondencije često nepoznate, a u slučajevima kada jesu poznate, znaju sadržavati veliki broj odudarajućih točaka. Iz tih razloga potrebno je primijeniti odgovarajuće algoritme za korekciju tih pogrešaka. Neki od dosadašnjih vrlo poznatih algoritama koji se primjenjuju u tu svrhu su RANSAC i ICP te TEASER i TEASER++ (brža verzija TEASER-a).

Primjena registracije oblaka točaka pomoću TEASER++ algoritma prikazana je na slici 2.2.



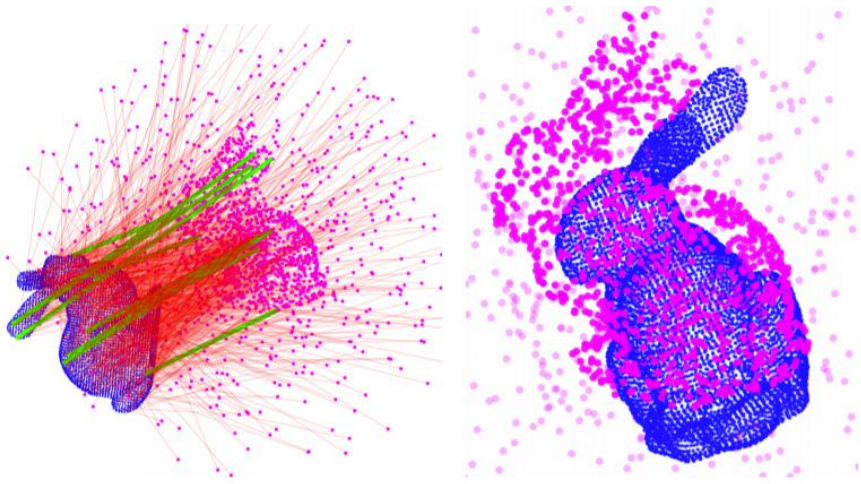
Slika 2.2. Registracija oblaka točaka; zelene linije predstavljaju točke modela, a crvene linije odudarajuće točke (slika je preuzeta sa stranice [8])

2.2.1. RANSAC

RANSAC (Random Sample Consensus) popularni je algoritam koji se često primjenjuje u robotici te problemima vezanim za robotski vid. Efikasan je prilikom registracije koja ima malu razinu šuma i mali broj odudarajućih točaka. Međutim, njegovo vrijeme izvedbe se eksponencijalno povećava s povećanjem broja odudarajućih točaka te zbog toga može imati lošu izvedbu prilikom njihovog velikog postotka. To je izrazito problematično jer je toleriranje velikog broja odudarajućih točaka od krucijalne važnosti u aplikacijama kada su korespondencije nepoznate [6].

Općenito govoreći, RANSAC je iterativni algoritam koji ima dvije faze: generalnu hipotezu te evaluacijsku hipotezu. U generalnoj hipotezi, RANSAC odabire točke slučajnim odabirom te procjenjuje određeni parametar iz uzorka. Hipoteza svakom iteracijom pretpostavlja odgovaraju li točke matematičkom modelu ili su u pitanju odudarajuće točke. Važno je naglasiti kako RANSAC uzima dio podataka, a ne cijeli model. Nakon dovoljnog broja iteracija, u drugoj fazi zvanoj evaluacijska hipoteza, odabire se ona ravnina koja ima najviše odgovarajućih točaka modela [7]. Međutim, odabir praga stvara određene poteškoće. Primjerice, ako je postavljeni prag premalen, točke modela bit će prepoznate kao odudarajuće točke, a ukoliko je prevelik, odudarajuće točke bit će proglašene točkama modela.

Na slici 2.3 lijevo su prikazana dva oblaka prije registracije. Izvorni oblak obojen je plavom bojom, a ciljani oblak ružičastom. 95% točaka su odudarajuće (crvene linije), a 5% točaka su točke modela (zelene linije). Na slici 2.3 desno prikazan je rezultat nakon primjene RANSAC algoritma. Može se uočiti da procjena izvornog oblaka otprilike odgovara, no nije dovoljno precizna.

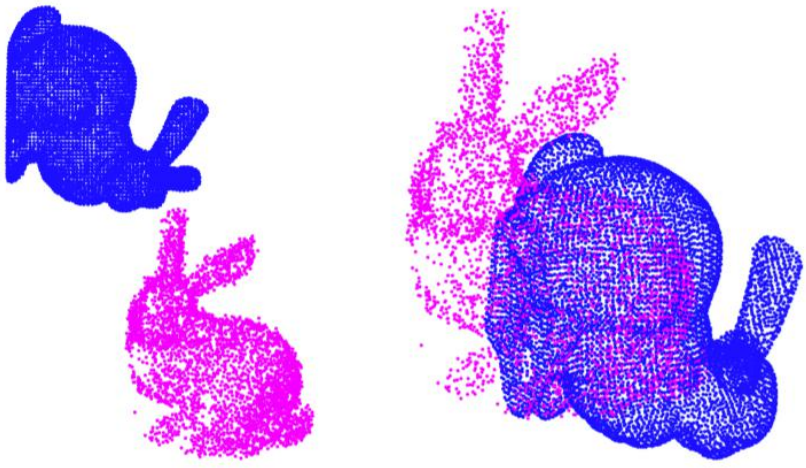


Slika 2.3. Registracija primjenom RANSAC algoritma (slika je preuzeta iz dokumenta [2])

2.2.2. ICP

Dva oblaka točaka mogu se pokušati poravnati i korištenjem popularnog algoritma ICP-a (Iterative Closest Point). Može se reći da je on najrasprostranjenija intuitivna metoda registracije. Njegov način djelovanja, ugrubo govoreći, sastoji se od dva koraka: prvi korak je pronaći parove točaka između dva oblaka točaka, a drugi korak uključuje izračunavanje transformacije, koja uključuje rotaciju i translaciju, kako bi se minimizirala Euclidova distanca između spojenih točaka [9].

Govoreći detaljnije, algoritam radi na principu određivanja najbliže točke izvornog oblaka točaka referentnom oblaku. To rezultira parovima točaka u kojima je jedan element iz izvornog oblaka, a drugi najbliža točka iz referentnog oblaka. Nakon toga se, uz određene procese transformacije, dolazi do funkcije troška čiji rezultat mora biti manji od praga. Ukoliko je manji, kao rezultat se dobiva poravnati oblak točaka. U suprotnom, algoritam ponovno određuje nove najbliže točke ili, ako je došao do maksimalnog broja iteracija, zaustavlja svoje djelovanje. ICP također ima mnogo nesavršenosti koje su vidljive kod registracije koja sadrži nepoznate ili nesigurne korespondencije. U tom slučaju, ICP se oslanja na sposobnost inicijalnog pogotka za nepoznatu transformaciju što rezultira poprilično lošom izvedbom [2]. Slika 2.4 prikazuje nedostatke prilikom primjene ICP algoritma.



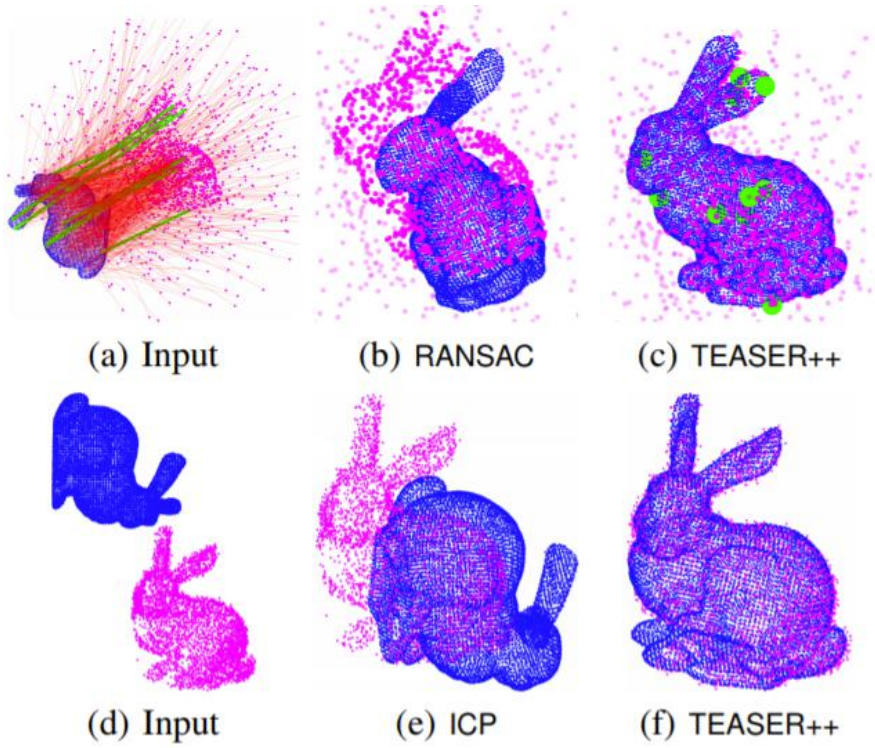
Slika 2.4. Registracija primjenom ICP algoritma
(slika je preuzeta iz dokumenta [2])

3. TEASER++ ALGORITAM

TEASER++ (Truncated least squares Estimation And SEmidefinite Relaxation) prvi je certificirani algoritam koji omogućuje brzu registraciju dvaju 3D oblaka točaka u prisutnosti velikog broja odudarajućih korespondencija. Motivacija za kreiranje ovog algoritma uslijedila je nakon uočavanja nedostataka koje imaju ostali algoritmi, primjerice RANSAC (implicitno pretpostavlja prisutnost malog broja outlinera) ili pak ICP (inicijalno pretpostavlja nepoznatu transformaciju).

Generalno se može zaključiti kako su dosadašnji algoritmi heuristički, to jest brzi, ali nedovoljno efikasni, ili su takozvane globalne metode koje su robusne, no imaju eksponencijalno vrijeme izvršavanja u najgorem slučaju. Došlo je do potrebe za stvaranjem pristupa koji može globalno rješavati registraciju (ne oslanjati se na inicijalnu pretpostavku), tolerirati ekstreman broj odudarajućih točaka, izvršavati naredbe u polinomijalnom vremenu, koji je brz u praksi te osigurava garanciju formalne performanse. S ciljem ispunjavanja tih uvjeta, kreirano je nekoliko doprinosa. Jedan od njih je i TEASER koji je otkrivao pravu transformaciju između oblaka točaka u prisutnosti odudarajućih točaka. Konačno je u sljedećem doprinosu došlo do kreiranja TEASER++ algoritma, brze verzije TEASER-a, koji koristi GNC (Graduated Non-Convexity) kako bi pretpostavio rotaciju bez rješavanja velikog SPD-a (SemiDefinite Programming) te se može koristiti u *real-time* aplikacijama.

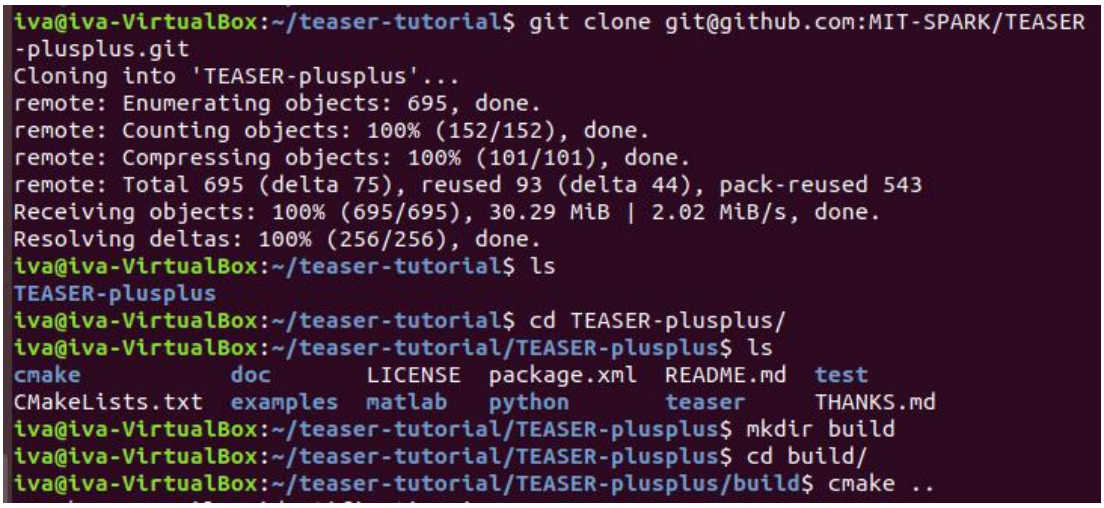
Zadnji doprinos je proširena evaluacija detekcije objekta i povezivanja oblaka točaka. Dokazano je da oba algoritma, TEASER i TEASER++, dominiraju nad ostalim popularnim algoritmima poput RANSAC-a te su robusni nad više od 99% odudarajućih točaka kada je skala poznata [2]. Nadalje, TEASER++ može se pokrenuti u milisekundama te je trenutno najbrži robusni algoritam za registraciju. Neusporedivo je robusniji od ICP-a te je puno točniji od Go-ICP-a [2]. Slika 3.1 prikazuje nadmoć TEASER++-a nad ICP i RANSAC algoritmom.



Slika 3.1. Usporedba algoritma TEASER++ s RANSAC i ICP algoritmom (slika je preuzeta iz dokumenta [2])

3.1. Implementacija TEASER++ algoritma

TEASER++ biblioteku moguće je preuzeti putem poveznice [8]. Na njoj se nalazi pristup *GitHub* stranici na kojoj je predstavljen ovaj algoritam te svi dodatci koji ga sačinjavaju. Važno je naglasiti kako je krucijalni korak prije pokretanja instalacije TEASER++ algoritma instalacija *Anaconda* i *Git* programa. Treba napomenuti i kako implementacija ovog algoritma može zahtijevati dodatnu instalaciju paketa koje prosječan korisnik ne koristi te se time vrijeme same instalacije može znatno produžiti. Koraci instalacije u terminalu prikazani su na slici 3.2..



Slika 3.2. Koraci instalacije TEASER++ biblioteke

Nadalje, za korištenje FPFH značajki iz *Open3D* paketa potrebno je instalirati noviju verziju *Open3D-a*. U primjeru s poveznice korištena je verzija 0.9.0.0, a trenutna potrebna verzija je 0.13.0.0 zbog ostalih nadogradnji.

Nakon što je TEASER++ instaliran, u mapi *example* ponuđeno je nekoliko primjera koji mogu poslužiti kao obrazac za implementaciju specifične primjene. Ponuđeni primjeri napisani su jezicima C++ i Python. Za potrebe ovog rada analizirani su isključivo oni napisani u Pythonu. Za korištenje svakog od navedenih primjera potrebna su različita okruženja. Drugim riječima, prije njihovog pokretanja potrebno je koristiti *conda* naredbu kako bi se moglo odabrati ono okruženje koje odgovara primjeru kojeg se želi pokrenuti. U nastavku su navedene naredbe koje je potrebno izvesti kako bi se pokrenuli navedeni primjeri. One su potrebne samo prilikom prvog pokretanja. Nakon toga je dovoljno izvesti samo *conda activate* naredbu.

```
sudo apt install cmake libeigen3-dev libboost-all-dev
conda create -n teaser_test python=3.6 numpy
conda activate teaser_test
conda install -c open3d-admin open3d=0.9.0.0
git clone https://github.com/MIT-SPARK/TEASER-plusplus.git
cd TEASER-plusplus && mkdir build && cd build
cmake -DTEASERPP_PYTHON_VERSION=3.6 .. && make teaserpp_python
cd python && pip install .
cd ../.. && cd examples/teaser_python_ply
python teaser_python_ply.py
```

Slika 3.3. Naredbe za pokretanje teaser_python_ply.py primjera [8]

```
sudo apt install cmake libeigen3-dev libboost-all-dev
conda create -n teaser_3dsmooth python=3.6 numpy
conda activate teaser_3dsmooth
conda install -c open3d-admin open3d=0.9.0.0
conda install scikit-learn
git clone https://github.com/MIT-SPARK/TEASER-plusplus.git
cd TEASER-plusplus && mkdir build && cd build
cmake -DTEASERPP_PYTHON_VERSION=3.6 .. && make teaserpp_python
cd python && pip install .
cd ../.. && cd examples/teaser_python_3dsmooth
python teaser_python_3dsmooth.py
```

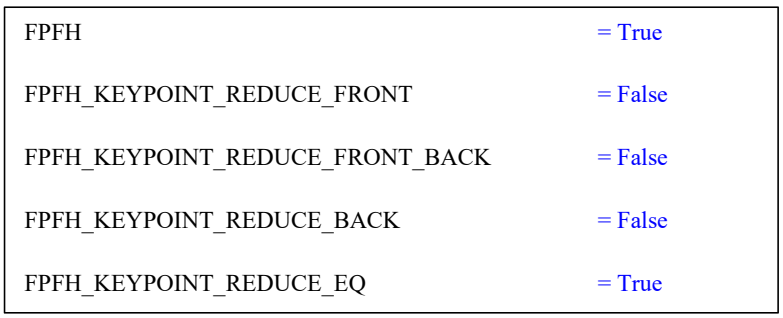
Slika 3.4. Naredbe za pokretanje teaser_python_3dsmooth.py primjera [8]

4. EKSPERIMENTALNA EVALUACIJA

4.1. Programska podrška

Kako bi se realizirala registracija snimki rana potrebna za ispunjavanje zadatka ovog rada, bilo je potrebno kreirati programsku podršku u skladu s tim. Ona je napravljena izmjenom primjera koji se nalazi u teaser_python_3dsmooth direktoriju kojem se može pristupiti putem TEASER-plusplus/examples/teaser_python_3dsmooth/ putanje. Taj primjer naziva se teaser_python_3dsmooth.py te koristi podatke iz datoteke gt.log kojoj se može pristupiti putem putanje TEASER-plusplus/examples/example_data/3dmatch_sample/gt.log te ju je također bilo potrebno izmijeniti.

Prva izmjena je bila dodavanje FPFH konstante za kontrolu programskog toka te prigodnog koda kojim ta konstanta upravlja. Taj kod služi za dobivanje podotipkanog, smanjenog oblaka točaka te FPFH značajki tih točaka. Takvo smanjeni oblak omogućuje obradu većih oblaka koje uglavnom nije moguće obraditi zbog zauzimanja previše radne memorije. Pripadni kod koji je označen FPFH konstantom preuzet je s [10].



Slika 4.1. Definiranje konstanti

Dodane su još četiri konstante koje omogućuju različitu redukciju većeg oblaka točaka; FPFH_KEYPOINT_REDUCE_FRONT, FPFH_KEYPOINT_REDUCE_FRONT_BACK, FPFH_KEYPOINT_REDUCE_BACK i FPFH_KEYPOINT_REDUCE_EQ kao što je i prikazano na slici 4.1.. Sufiksi tih konstanti označavaju sljedeće: FRONT - redukcija sprijeda, FRONT_BACK - redukcija sprijeda i straga, BACK - redukcija straga te EQ - ravnomjerno preko cijelog oblaka.

Ravnomjerna redukcija davala je najbolje rezultate tako da je ona primjenjivana u svim registracijama u ovom radu. *Kod* ravnomjerne redukcije prikazan je na slici 4.2.. Redukcija je iznimno važna u procesu registracije. Ukoliko su oblaci različite veličine, nužno je njihove podotipkane oblake, to jest FPFH značajke, svesti na jednaku veličinu. U suprotnom, registracija neće biti moguća.

```
if FPFH:
    frag1_key, frag1 = preprocess_point_cloud(frag1_pc, voxel_size=VOXEL_SIZE1)
    frag2_key, frag2 = preprocess_point_cloud(frag2_pc, voxel_size=VOXEL_SIZE2)
    frag1_key_n = len(frag1_key.points)
    frag2_key_n = len(frag2_key.points)
    if frag1_key_n < frag2_key_n:
      if FPFH_KEYPOINT_REDUCE_EQ:
        print("Reducing keypoint2 point cloud equaly through the whole cloud.")
        frag2_key.points=o3d.utility.Vector3dVector(np.array(frag2_key.points)[spread_idx
        (frag2_key_n, frag1_key_n)])
        frag2.data = frag2.data[:, spread_idx(frag2_key_n, frag1_key_n)]
    else:
      if FPFH_KEYPOINT_REDUCE_EQ:
        print("Reducing keypoint1 point cloud equaly through the whole cloud.")
        frag1_key.points = o3d.utility.Vector3dVector(np.array(frag1_key.points)[spread_idx
        (frag1_key_n, frag2_key_n)])
        frag1.data = frag1.data[:, spread_idx(frag1_key_n, frag2_key_n)]
```

Slika 4.2. Redukcija značajki ravnomjernom redukcijom

Nadalje, *kodu* su dodane VOXEL_SIZE1 i VOXEL_SIZE2 konstante tipa float koje omogućuju mijenjanje veličine voksela. Voksel u trodimenzionalnoj grafici predstavlja najmanji dio trodimenzionalog prostora neke scene koji se može obrađivati ili prikazivati. Koristi se u vizualizaciji i analizi 3D podataka [11]. Veličina voksela utječe na veličinu podotipkanog oblaka, to jest značajki. Veza je obrnuto proporcionalna, to jest što je voksel veći, manji je podotipkani oblak. Veličinu voksela važno je dovoljno dobro procijeniti kako bi registracija bila izvediva. Ukoliko je voksel premalen, postotak obrađenih točaka bit će prevelik te će zbog nedostatka memorije biti nemoguće odraditi registraciju. S druge strane, ukoliko je prevelik, to će rezultirati malim brojem obrađenih točaka te registracija neće biti valjana. Veličina voksela korištena u registraciji u ovom radu bila je od 0.005 do 0.007 m. Slika 4.3. prikazuje funkciju u kojoj se vrši redukcija s obzirom na veličinu voksela.

```
def preprocess_point_cloud(pcd, voxel_size):
    print(":: Downsample with a voxel size %.3f." % voxel_size)
    pcd_down = pcd.voxel_down_sample(voxel_size)

    radius_normal = voxel_size * 2
    print(":: Estimate normal with search radius %.3f." % radius_normal)
    pcd_down.estimate_normals(o3d.geometry.KDTreeSearchParamHybrid(radius=radius_normal,
    max_nn=30))

    radius_feature = voxel_size * 5
    print(":: Compute FPFH feature with search radius %.3f." % radius_feature)
    pcd_fpfh = o3d.pipelines.registration.compute_fpfh_feature(
        pcd_down, o3d.geometry.KDTreeSearchParamHybrid(radius=radius_feature, max_nn=100))
    return pcd_down, pcd_fpfh
```

Slika 4.3. Redukcija s obzirom na veličinu voksela

Budući da je za testiranje medicinskih snimki rana u ovom radu potrebna manipulacija ulaznim oblacima točaka, važno je omogućiti dodavanje šuma jednom i drugom oblaku te manipuliranje transformacijskim matricama. Zbog toga je u *kod* bilo potrebno dodati konstante NOISE_BOUND1 i NOISE_BOUND2, koje označavaju dodani šum na ulazne oblake, te transformacijske matrice T1 i T2. Ukoliko nije bilo potrebe za transformacijom, koristila se jedinična matrica. Za rotaciju je korištena matrica s ortonormiranim stupcima koja rotira oko x osi za devedeset stupnjeva. Točne matrice korištene u kodu za različite vrste transformacije prikazane su u nastavku na slici 4.4..

```
# default (no transform)
T1 = np.array(
    [[1.0,  0.0,  0.0,  0.0],
     [0.0,  1.0,  0.0,  0.0],
     [0.0,  0.0,  1.0,  0.0],
     [0.0,  0.0,  0.0,  1.0]])

# translation
T1 = np.array(
    [[1.0,  0.0,  0.0, -1.9],
     [0.0,  1.0,  0.0,  6.6],
     [0.0,  0.0,  1.0, -4.0],
     [0.0,  0.0,  0.0,  1.0]])

# rotation
T2 = np.array(
    [[1.0,  0.0,  0.0,  0.0],
     [0.0,  0.0, -1.0,  0.0],
     [0.0,  1.0,  0.0,  0.0],
     [0.0,  0.0,  0.0,  1.0]])
```

Slika 4.4. Matrice korištene prilikom transformacija scena rana (prva: bez transformacije, druga: translacija, treća: rotacija)

Izmjene su izvršene i na *pair_eval_helper()* funkciji kako bi joj se mogla predati imena datoteka ulaznih oblaka točaka. Nadalje, dodani su informativni ispisi veličine, to jest broja točaka ulaznih oblaka, transformacijskih matrica te količine šuma. Dodan je i poziv spomenute funkcije za izračun FPFH značajki te također spomenuta redukcija većeg od dobivenih podotipkanih oblaka, to jest značajki.

```
#izmjena pair_eval_helper funkcije
def pair_eval_helper(scene_path, desc_path, gt_mat, fragment1_name, fragment2_name)

#informativni ispisi
print("Made downsampled " + fragment1_name + ".ply point cloud which consists of " +
str(len(frag1_key.points)) + " points (" + str(len(frag1_key.points)*100/len(frag1_pc.points))
+ " % of " + str(len(frag1_pc.points)) + ").")
print("Made downsampled " + fragment2_name + ".ply point cloud which consists of " +
str(len(frag2_key.points)) + " points (" + str(len(frag2_key.points)*100/len(frag2_pc.points))
+ " % of " + str(len(frag2_pc.points)) + ").")

print("T1 matrix")
print(T1)
frag1_pc.transform(T1)
print("T2 matrix")
print(T2)
frag2_pc.transform(T2)

print("Noise1 " + str(NOISE_BOUND1))
transposed_points = np.transpose(np.asarray(frag1_pc.points))
transposed_points += (np.random.rand(3, N1) - 0.5) * 2 * NOISE_BOUND1
frag1_pc.points = o3d.utility.Vector3dVector(np.transpose(np.asarray(transposed_points)))

print("Noise2 " + str(NOISE_BOUND2))
transposed_points = np.transpose(np.asarray(frag2_pc.points))
transposed_points += (np.random.rand(3, N2) - 0.5) * 2 * NOISE_BOUND2
frag2_pc.points = o3d.utility.Vector3dVector(np.transpose(np.asarray(transposed_points)))
```

Slika 4.5. Izmjena pair_eval_helper funkcije te informativni ispisi

U *main()* funkciju dodan je fiksni, zakodirani poziv čitanja (0,0) para iz izmijenjene gt.log datoteke. Taj par predstavlja jediničnu transformacijsku matricu (bez rotacije, translacije i skaliranja) budući da je svaki par oblaka koji se koristi u ovom radu u približno istom položaju.

Na posljetku je dodana kontrola imena ulaznih oblaka točaka rana koji su smješteni u TEASER-plusplus/examples/example_data/3dmatch_sample/ direktoriju te je dodano vraćanje vrijednosti estimirane transformacijske matrice koja nam je potrebna za njen ispis.

4.2. Testiranje registracije

TEASER++ registracija ispitivana je na snimkama rana izrađenima pomoću 3D kamera.

Registracija je testirana koristeći 3 scene rana na stražnjici te nakon toga 3 scene rana na nozi. Te scene u daljnjem tekstu navedene su kao scena 1, scena 2 i scena 3. Dodani šum uvijek je iznosio 0.001, a rotacija je bila od 90 stupnjeva.

Poredak registracije scena rana na stražnjici bio je sljedeći:

1. Scena 1 bez šuma i bez transformacije + scena 1 s dodanim šumom i rotacijom
2. Scena 1 bez šuma i bez transformacije + scena 1 s dodanim šumom i translacijom
3. Scena 1 bez šuma, translatirana + scena 1 bez šuma, rotirana
4. Scena 1 sa šumom, translatirana + scena 1 sa šumom, rotirana
5. Scena 1 bez šuma i bez transformacije + scena 2 bez šuma i bez transformacije
6. Scena 1 bez šuma i bez transformacije + scena 3 bez šuma i bez transformacije
7. Scena 1 sa šumom, translatirana + scena 2 sa šumom, rotirana
8. Scena 1 sa šumom, translatirana + scena 3 sa šumom, rotirana

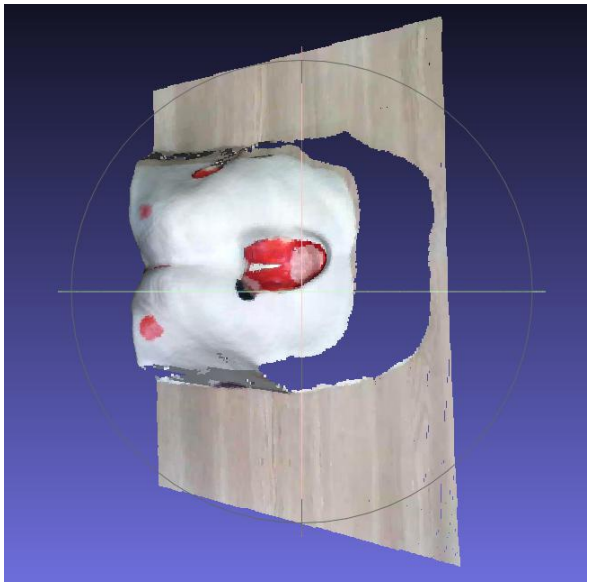
Nakon iscrpnog testiranja registracije na scenama rana na stražnjici, scene rana na nozi izvedene su na dvije najkompliciranije registracije što je bilo dovoljno za usporedbu i potvrdu rezultata.

Poredak registracije scena rana na nozi bio je sljedeći:

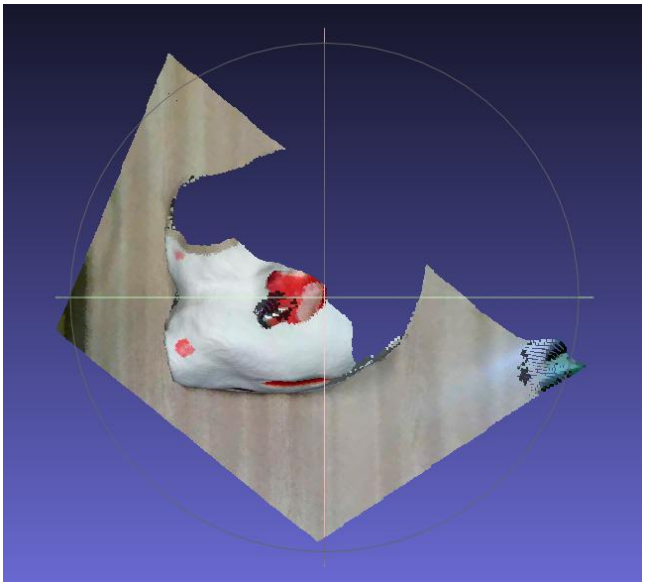
1. Scena 1 sa šumom, translatirana + scena 2 sa šumom, rotirana
2. Scena 1 sa šumom, translatirana + scena 3 sa šumom, rotirana

4.2.1. Registracija scena rana na stražnjici

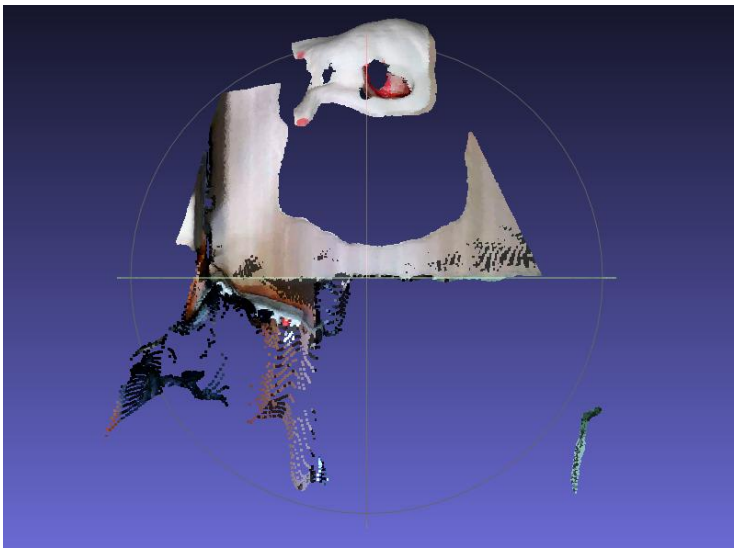
Tri scene rana korištene u postupku registracije prikazane su na slikama 4.5., 4.6. i 4.7..



Slika 4.5. Rana na stražnjici, scena 1



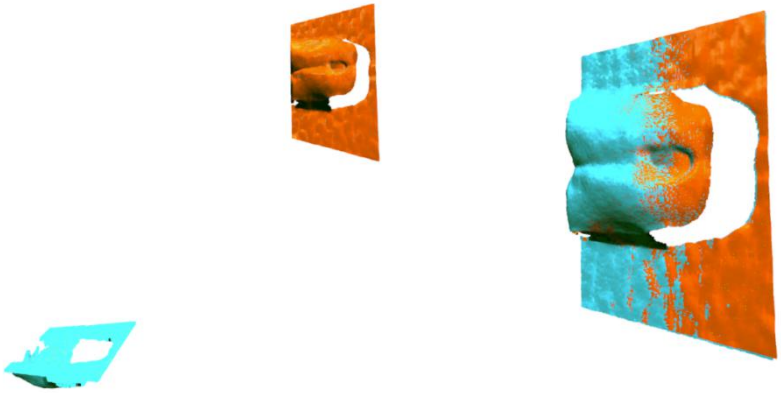
Slika 4.6. Rana na stražnjici, scena 2



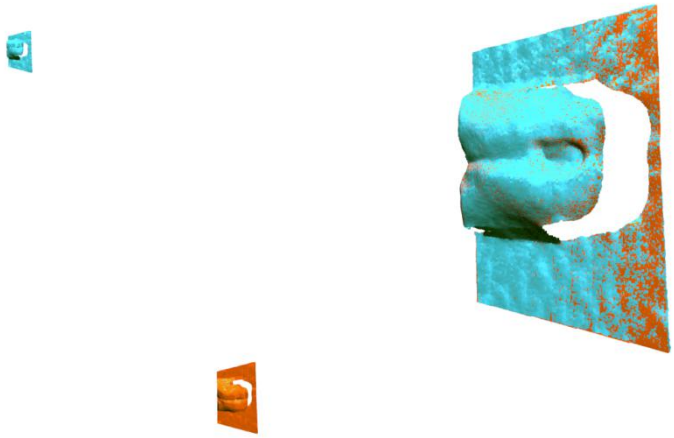
Slika 4.7. Rana na stražnjici, scena 3

U nastavku su prikazani rezultati svake od 8 registracija. Prva navedena scena prikazana je narančastom bojom, a druga navedena plavom.

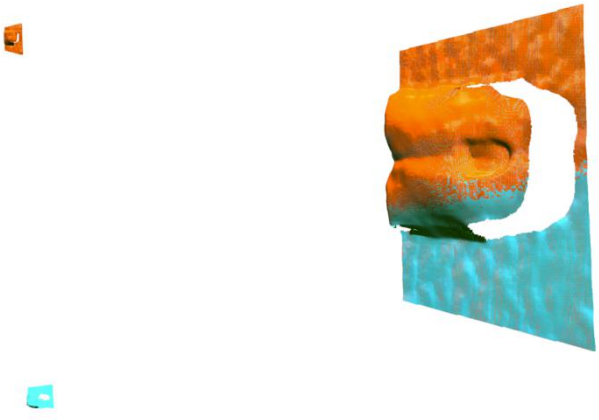
1. Scena 1 bez šuma i bez transformacije + scena 1 s dodanim šumom i rotacijom



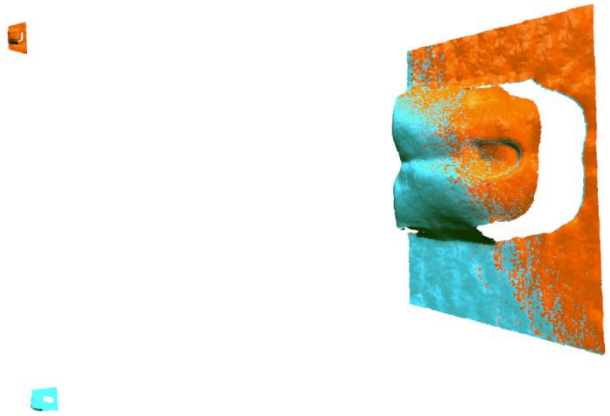
2. Scena 1 bez šuma i bez transformacije + scena 1 s dodanim šumom i translacijom



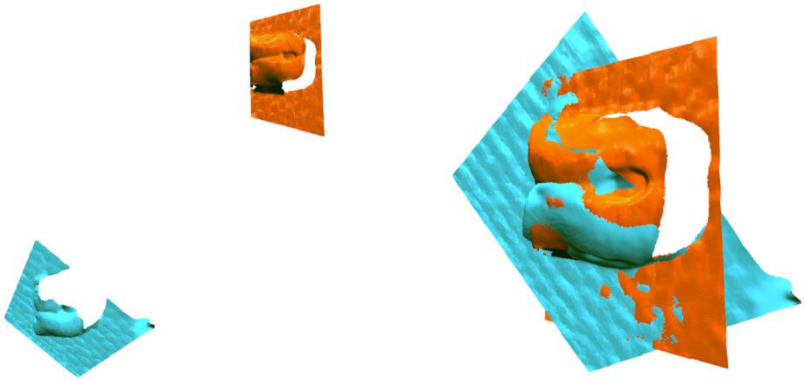
3. Scena 1 bez šuma, translatirana + scena 1 bez šuma, rotirana



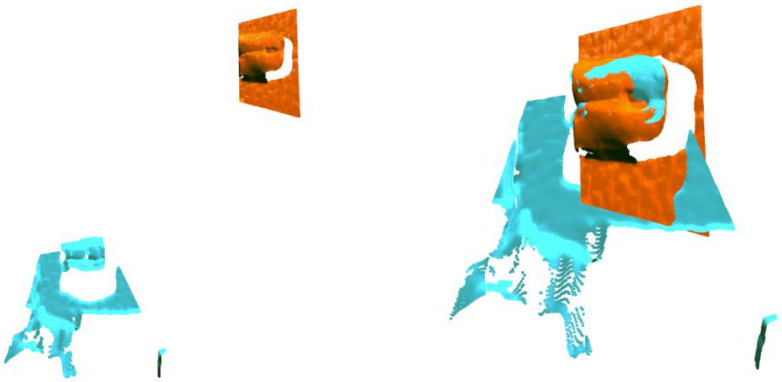
4. Scena 1 sa šumom, translatirana + scena 1 sa šumom, rotirana



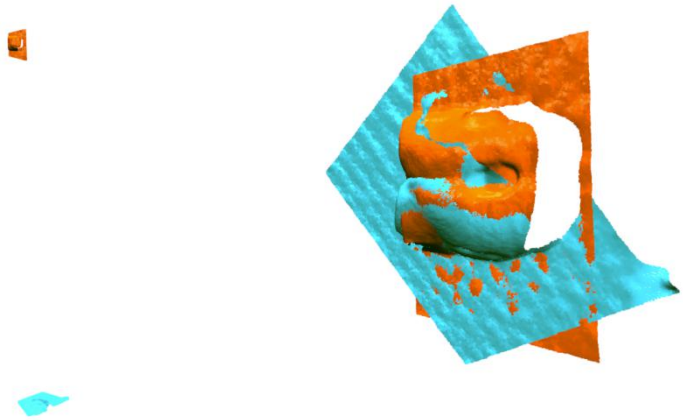
5. Scena 1 bez šuma i bez transformacije + scena 2 bez šuma i bez transformacije



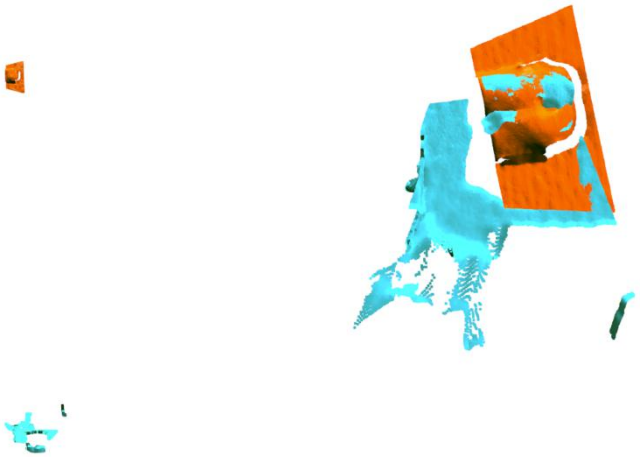
6. Scena 1 bez šuma i bez transformacije + scena 3 bez šuma i bez transformacije



7. Scena 1 sa šumom, translatirana + scena 2 sa šumom, rotirana



8. Scena 1 sa šumom, translatirana + scena 3 sa šumom, rotirana



Tablica 4.1. prikazuje brojčane rezultate nakon svake registracije. U njoj se jasno vidi kako je postotak značajki koji je uzet u obzir prilikom registracije obrnuto proporcionalan veličini voksela. Nadalje, vidi se kako trajanje registracije proporcionalno ovisi o broju spojenih korespondencija.

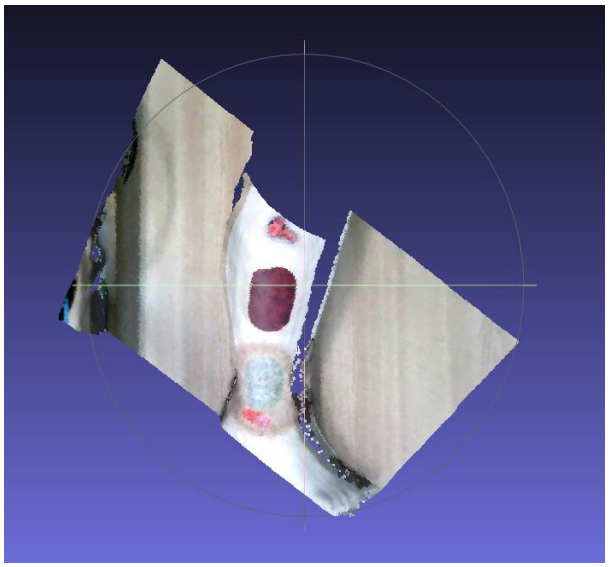
Stupci u tablici imenovani su na sljedeći način: Voksel označava veličinu oba voksela, Značake 1 postotak značajki uzet u obzir kod prve scene, Značajke 2 postotak značajki uzet u obzir kod druge scene, Iteracije broj izvedenih iteracija, Korespondencije broj spojenih, prepoznatih korespondencija, a Trajanje označava trajanje TEASER++ registracije izraženo u sekundama.

Tablica 4.1. Brojčani rezultati svih registracija scena rana na stražnjici

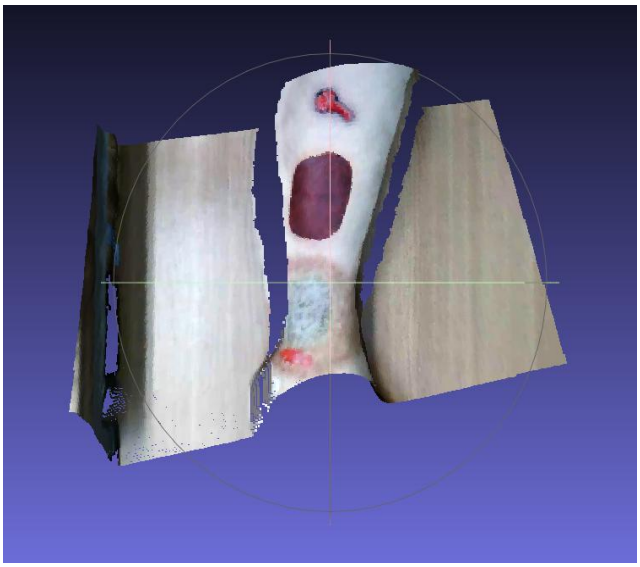
	Voksel	Značajke 1	Značajke 2	Iteracije	Korespon-dencije	Traja-nje
1. Scena 1 bez šuma i bez transformacije + scena 1 s dodanim šumom i rotacijom	0.007	11%	11%	15	2329	2.66
2. Scena 1 bez šuma i bez transformacije + scena 1 s dodanim šumom i translacijom	0.007	11%	11%	8	2756	8.21
3. Scena 1 bez šuma, translatirana + scena 1 bez šuma, rotirana	0.007	11%	11%	8	4750	61.08
4. Scena 1 sa šumom, translatirana + scena 1 sa šumom, rotirana	0.007	13%	13%	12	2965	8.17
5. Scena 1 bez šuma i bez transformacije + scena 2 bez šuma i bez transformacije	0.006	15%	15%	19	2803	2.56
6. Scena 1 bez šuma i bez transformacije + scena 3 bez šuma i bez transformacije	0.006	15%	16%	14	1799	0.54
7. Scena 1 sa šumom, translatirana + scena 2 sa šumom, rotirana	0.006	16%	16%	19	3101	2.97
8. Scena 1 sa šumom, translatirana + scena 3 sa šumom, rotirana	0.005	25%	27%	26	3582	2.86

4.2.2. Registracija scena rana na nozi

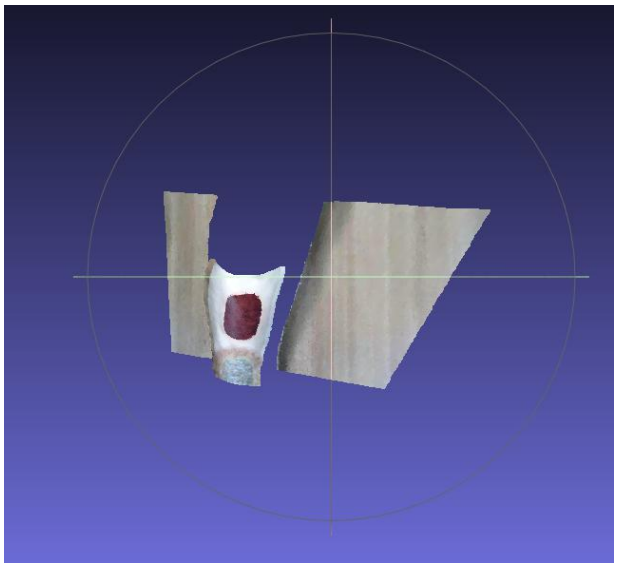
Tri scene rana korištene u postupku registracije prikazane su na slikama 4.8., 4.9. i 4.10..



Slika 4.8. Rana na nozi, scena 1



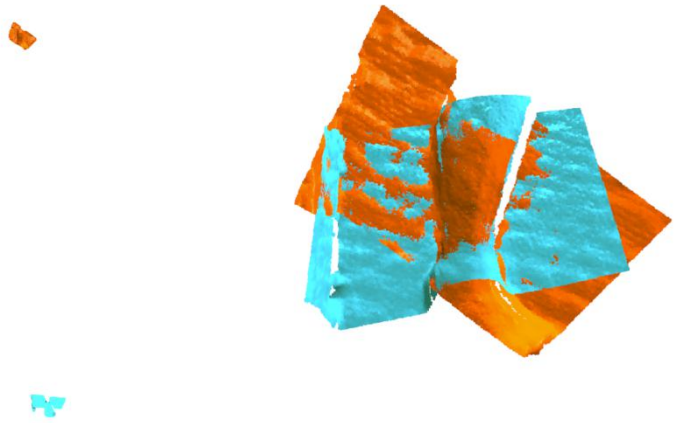
Slika 4.9. Rana na nozi, scena 2



Slika 4.10. Rana na nozi, scena 3

U nastavku su prikazani rezultati 2 registracije. Prva navedena scena prikazana je narančastom bojom, a druga navedena plavom.

1. Scena 1 sa šumom, translatirana + scena 2 sa šumom, rotirana



2. Scena 1 sa šumom, translatirana + scena 3 sa šumom, rotirana

Tablica 4.2 prikazuje brojčane rezultate registracija scena rana na nozi. Vokseli označavaju veličinu voksela, Značajke 1 postotak značajki uzet u obzir kod prve scene, a Značajke 2 broj značajki uzet u obzi kod druge scene.

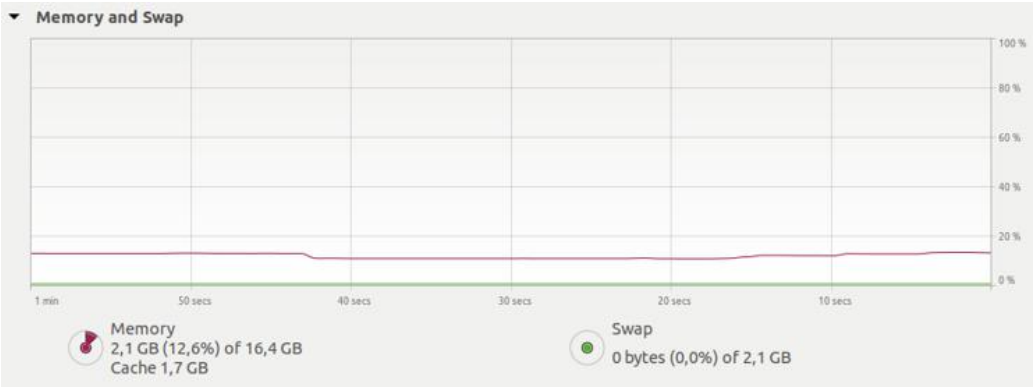
Tablica 4.2. Brojčani rezultati svih registracija scena rana na nozi

	Vokseli	Značajke 1	Značajke 2
1. Scena 1 sa šumom, translatirana + scena 2 sa šumom, rotirana	0.005	19%	19%
2. Scena 1 sa šumom, translatirana + scena 3 sa šumom, rotirana	0.005	12%	13%

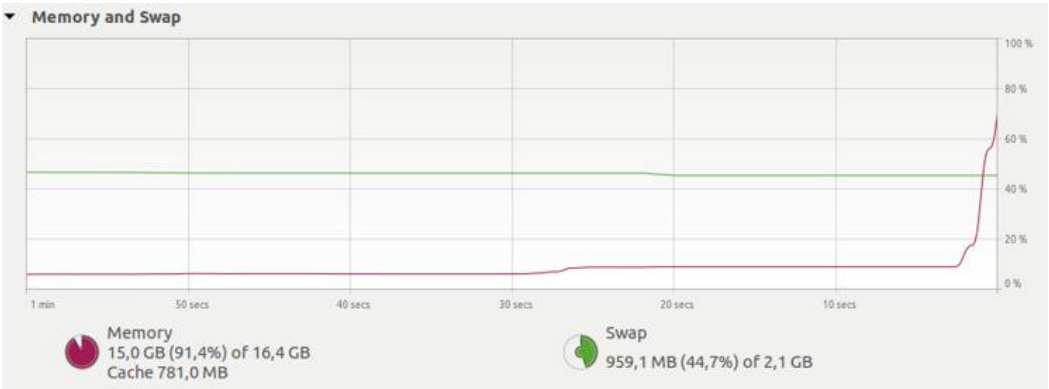
4.2.3. Uočeni nedostatci prilikom registracije

Glavni nedostatak prilikom registracije snimki rana bilo je veliko zauzeće memorije. Snimci koji su imali 60 i više MB nisu mogli biti registrirani te su registrirani samo oni snimci veličine do 5 MB, iako je registracija izvedena na radnoj stanici natprosječnog memorijskog kapaciteta. Zauzeće memorije na računalu prije pokretanja registracije dvije snimke od 2 i 4 MB, uzimajući u obzir 11% značajki, iznosilo je 1.7 GB, a zauzeće prilikom te registracije 2.1 GB kao što je i prikazano na slici 4.11. što znači da je registracija u tom slučaju zauzela 0.4 GB. S druge strane, prilikom pokušaja registracije snimki od 75 i 80 MB, uzimajući u obzir 5% značajki, memorija je došla do svog maksimalnog zauzeća od 16 GB te se program srušio (slika

4.12.). Budući da ni sami primjeri navedeni u TEASER++ biblioteci nemaju preko 5 MB, može se pretpostaviti kako TEASER++ još uvijek nije dovoljno razvijen za registraciju velikih snimki.



Slika 4.11. Zauzeće memorije prilikom registracije snimki od 2 i 4 MB



Slika 4.12. Zauzeće memorije netom prije rušenja programa prilikom registracije snimki od 75 i 80 MB

Sljedeći nedostatak bio je odabir veličine voksela. Njegovo smanjivanje dovodi do rušenja programa zbog nedovoljno memorije, a njegovo povećavanje do nemogućnosti registracije. Budući da se odabire ručno te proizvoljno, bilo je vremenski naporno, ponekad i neizvedivo, određivati idealnu veličinu voksela.

5. ZAKLJUČAK

Zadatak ovog rada bio je registrirati 3D snimke rana TEASER++ algoritmom što je uspješno i izvršeno. Ključan dio prilikom programiranja bio je izjednačiti veličinu oblaka točaka ukoliko su različiti, to jest reducirati veći oblak, te izvršiti redukciju na pravilan način, ravnomjernim podotipkavanjem. Većina snimki registrirana je veoma dobro čak i nakon dodanog šuma na obje snimke te dodane transformacije u obliku rotacije i translacije. Nadalje, vrijeme registracije bilo je iznimno brzo. Moglo bi se zaključiti kako je TEASER++ opravdao očekivanja kao brz i učinkovit algoritam. Međutim, velike količine memorije koje se zauzimaju prilikom registracije onemogućavaju registraciju snimaka većih veličina. Čak i kada se te snimke pokušaju smanjiti podotipkavanjem, broj značajki koje su uzete u obzir bude toliko malen da se registracija uopće ne može pravilno izvesti.

Sve u svemu, TEASER++ pokazao se kao veoma uspješan algoritam, bez obzira na broj odudarajućih točaka, šumova ili transformacija. U odnosu na dosadašnje algoritme, može se sa sigurnošću reći da je revolucionaran te će zasigurno doprinijeti u rješavanju mnogih dosadašnjih problema u vezi registracije 3D snimaka.

LITERATURA

[1] D. Filko, E. K. Nyarko, „Vision4Wounds”, dostupno na: https://vision4wounds.ferit.hr/

[31. 8. 2021.]

[2] H. Yang, J. Shi, L. Carleone, „TEASER: Fast and Certifiable Point Cloud Registration”, 2020.

[3] A. Jovanovski, „Oblak točaka i autocad civil 3d”, dostupno na: https://www.academia.edu/37048771/Oblak_tocaka_i_autocad_civil_3d [31. 8. 2021.]

[4] K. Chen, Y. K. Lai, S. M. Hu, „3D indoor scene modeling from RGB-D data: a survey“, Computational Visual Media, br. 1, sv. 4, 2015.

[5] X. Huang, G. Mei, J. Zhang, R. Abbas, „A comprehensive survey on point cloud registration”, The University of Sydney, University of Technology Sydney, 2021.

[6] H. Yang, L. Calone, „A Polynomial-time Solution for Robust Registration with Extreme Outlier Rates”, Robotics: Science and Systems (RSS), 2019.

[7] S. Choi, W. Yu, T. Kim, „Performance evaluation of RANSAC family”, 2009.

[8] „GitHub - MIT-SPARK/TEASER-plusplus”, dostupno na: GitHub - MIT-SPARK/TEASER-plusplus: A fast and robust point cloud registration library [1. 9. 2021.]

[9] C. Li, C. Tao, G. Liu, „3D Visual SLAM Based on Multiple Iterative Closest Point“, Mathematical Problems in Engineering, 2015.

[10] „Global registration - Open3D 0.13.0 documentation”, dostupno na: http://www.open3d.org/docs/release/tutorial/pipelines/global_registration.html [30. 8. 2021.]

[11] „Glosbe riječnik - voxel”, dostupno na: https://hr.glosbe.com/en/hr/voxel [1. 9. 2021.]

SAŽETAK

Naslov: Registracija medicinskih 3D snimki rana korištenjem TEASER++ biblioteke

Cilj ovog završnog rada je pokušati registrirati 3D snimke rana TEASER++ bibliotekom. Rad je započeo instalacijom Anaconda i Git programskih paketa. Nakon toga, instalirana je TEASER++ biblioteka te svi dodatni paketi nužni za njeno pokretanje. Nadalje, bilo je potrebno napisati programski kod svojstven za rješavanje problema registracije 3D snimki rana poput izjednačavanja veličine oblaka točaka te odabira značajki, to jest podotipkavanja. Kada je kod uspješno kreiran, provedena je registracija snimki rana na stražnjici te na nozi. Registracija je bila uspješna, no isključivo na snimkama manje veličine zbog prevelikog zauzeća memorije prilikom pokušaja registracije većih snimaka. Postepeno je na snimke dodavan šum te transformacija u obliku rotacije i translacije. Registracija je i nakon toga bila relativno uspješna. Vrijeme izvođenja bilo je iznimno brzo tijekom svih testiranja.

Ključne riječi: TEASER++, oblak točaka, 3D snimke rana, registracija 3D oblaka točaka

ABSTRACT

Title: 3D point cloud registration of the medical wounds scenes using TEASER++ library

The aim of this final thesis is to try to register 3D point clouds of the wounds using TEASER++. First thing that had to be done was the installation of Anaconda and Git software packages. Furthermore, the TEASER++ library was installed with all additional packages necessary for its launch. Afterwards, it was necessary to program a solution for 3D point cloud registration of the wounds, such as equalizing the size of point clouds by making the larger cloud smaller and subsampling point clouds. When the code was successfully created, point clouds of the wounds on the lower back region and on the leg were registered. Registration was successful, but only on the smaller scenes due to lack of memory when trying to register larger scenes. Noise and transformation, in the form of the rotation and translation, were gradually added to the point clouds. The registration was still relatively successful after that. Execution time was constantly fast during testing.

Key words: TEASER++, point clouds, 3D scenes of wounds, 3D point cloud registration

ŽIVOTOPIS

Iva Gavran rođena je u Požegi. Završila je Osnovnu školu Antuna Kanižlića u Požegi te nakon toga Opću gimnaziju u Požegi s odličnim uspjehom. Fakultet elektrotehnike, računarstva i informacijskih tehnologija u Osijeku upisuje 2018. godine. U slobodno vrijeme voli proučavati umjetnu inteligenciju, psihologiju te antropologiju.

_____

Potpis autora

DODATNI PRILOZI

Programski kod priložen je na CD-u uz rad.