Stvaranje 3D modela prostorija fuzijom 3D oblaka točaka snimljenih 3D kamerom

Tomašić, Branimir

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

Diplomski studij

Automobilsko računarstvo i komunikacije

STVARANJE 3D MODELA PROSTORIJA FUZIJOM 3D OBLAKA TOČAKA SNIMLJENIH 3D KAMEROM

Diplomski rad

Branimir Tomašić

Osijek, 2020.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac D1: Obrazac za imenovanje Povjerenstva za diplomski ispit

Osijek, 22.09.2020.

Odboru za završne i diplomske ispite

Imenovanje Povjerenstva za diplomski ispit

Ime i prezime studenta:	Branimir Tomašić
Studij, smjer:	Diplomski sveučilišni studij Automobilsko računarstvo i komunikacije
Mat. br. studenta, godina	D-27ARK, 21.09.2019.
OIB studenta:	28105846961
Mentor:	Prof.dr.sc. Robert Cupec
Sumentor:	Mateja Hržica
Sumentor iz tvrtke:	
Predsjednik Povjerenstva:	Izv. prof. dr. sc. . Damir Filko
Član Povjerenstva 1:	Prof.dr.sc. Robert Cupec
Član Povjerenstva 2:	Izv. prof. dr. sc. Emmanuel-Karlo Nyarko
Naslov diplomskog rada:	Stvaranje 3D modela prostorija fuzijom 3D oblaka točaka snimljenih 3D kamerom
Znanstvena grana rada:	**Obradba informacija (zn. polje računarstvo)**
Zadatak diplomskog rada:	Izraditi računalni program koji spaja više 3D oblaka točaka snimljenih 3D kamerom u jedan primjenom iterativnog algoritma najbliže točke.Sumentor s FERIT-a: Mateja Hržica
Prijedlog ocjene pismenog dijela ispita (diplomskog 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: 2 bod/boda Jasnoća pismenog izražavanja: 3 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene mentora:	22.09.2020.

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 ORIGINALNOSTIRADA

Osijek, 12.10.2020.

Ime i prezime studenta:	Branimir Tomašić
Studij:	Diplomski sveučilišni studij Automobilsko računarstvo i komunikacije
Mat. br. studenta, godina upisa:	D-27ARK, 21.09.2019.
Turnitin podudaranje [%]:	2

Ovom izjavom izjavljujem da je rad pod nazivom**: Stvaranje 3D modela prostorija fuzijom 3D oblaka točaka snimljenih 3D kamerom**
izrađen pod vodstvom mentora Prof.dr.sc. Robert Cupec

i sumentora Mateja Hržica

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:

ZAHVALA

Zahvaljujem se mentoru prof. dr. sc. Robertu Cupecu i sumentorici mag.ing. el. Mateji Hržici na velikoj pomoći, konstruktivnim kritikama i korisnim savjetima koje su mi pružili tijekom pisanja ovog rada.

Također veliko hvala mojoj obitelji, rodbini i prijateljima na pruženoj podršci i velikom razumijevanju tijekom pisanja ovog rada, ali i tijekom cijelog studija.

Od srca hvala i mojoj djevojci Željki, koja mi je tijekom pisanja rada bila ogromna pomoć, bila mi je i „lektor“ i „statističar“, hvala joj i na razumijevanju i motivaciji koju mi je pružila, ne samo sada nego i uvijek.

SADRŽAJ

1. UVOD

Ulaskom virtualne stvarnosti u domove preko igraćih konzola počinje se širiti upotreba RGB-D kamera. Kamere postaju sve lakše, pouzdanije i jeftinije zbog čega se one počinju koristiti i u ostale svrhe. U ovom radu bit će prikazano korištenje upravo jedne takve kamere u svrhu stvaranja 3D modela prostorija. 3D modeli predstavljaju stvarna fizička tijela, pretvaranjem njihovih stranica i rubova u virtualne točke u 3D prostoru. Na slici 1.1 prikazana je usporedba stvarnog objekta, RGB slike (a), dubinske slike objekta (b), objekt prikazan oblakom točaka u 3D prostoru (c) i modela dobivenog obradom točaka u 3D prostoru (d).

Slika 1.1 a) RGB slika , b) dubinska slika, c) oblak točaka u 3D prostoru i d) 3D model. (Slika preuzeta sa stranice: a), c) https://www.researchgate.net/figure/This-figure-illustrates-the-differences-between-our-method-and-MLS-We-acquired-multiple_fig3_261019453, d) https://free3d.com/imgd/s31925-culver-black-mug-52773.png)

Uporaba 3D modela prostora je raznovrsna, od najjednostavnijih uporaba za prikazivanje prostora od strane arhitekata i trgovaca nekretninama, do onih složenijih za mapiranje prostorije za autonomne robote i testiranje novih algoritama kretanja i robotskog vida. Širokoj uporabi 3D modela prostorija doprinosi način njihovog dobivanja, odnosno količina podataka koja se uštedi ovim načinom prikazivanja prostora.

U drugom poglavlju ovog rada postavljena je teorija za dubinsku sliku, oblak točaka te aktivnu viziju. Predstavljeni su i neki od najkorištenijih pristupa aktivne vizije. U trećem poglavlju navedene su metode koje se koriste za poravnanje oblaka točaka te su pojašnjene metode koje se koriste u ovom diplomskom radu. Četvrto poglavlje je predodređeno za implementaciju algoritma za izradu 3D modela prostorija. Također, navedene su sve vanjske biblioteke koje su bile potrebne, kao i baza 3D modela prostorija koja se koristila u simulacijama. Peto poglavlje eksperimentalna je evaluacija algoritma pojašnjenog u prethodnom poglavlju. Završno poglavlje je zaključak diplomskog rada, gdje su predloženi mogući budući smjerovi unaprjeđenja predloženog algoritma.

1.1. Zadatak diplomskog rada

Zadatak diplomskog rada je izraditi računalni program koji spaja više 3D oblaka točaka snimljenih 3D kamerom u jedan primjenom iterativnog algoritma najbliže točke - ICP (engl. *Iterative Closest Point*).

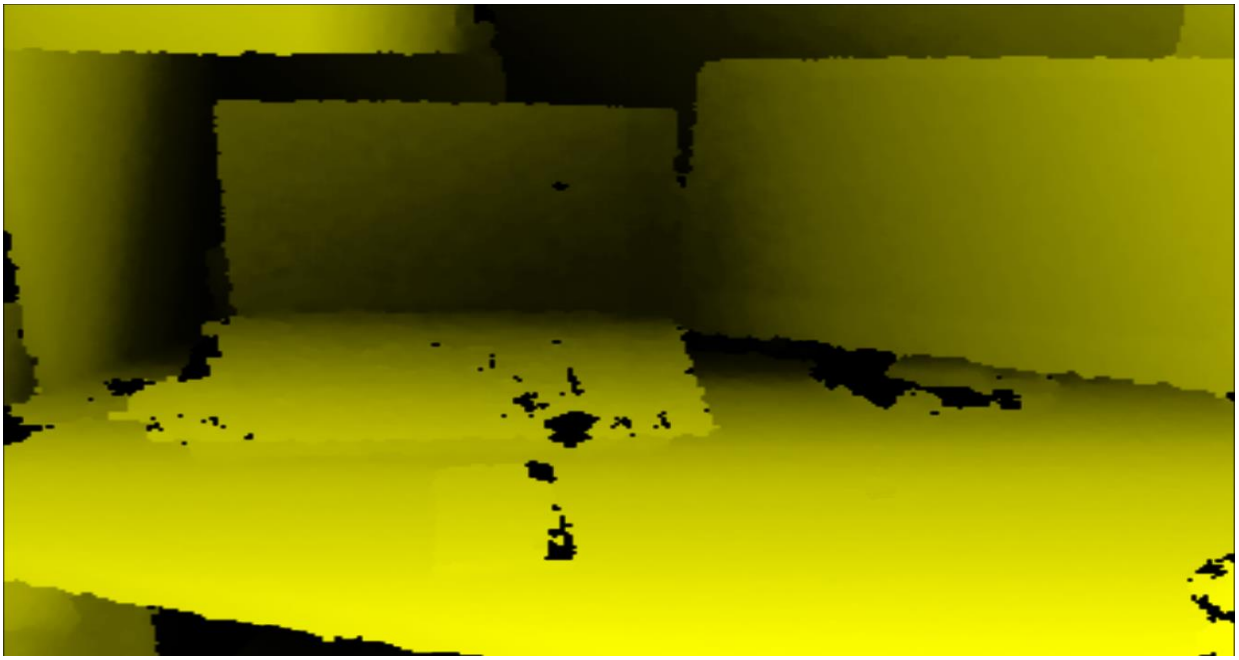
2. SNIMANJE PROSTORA POMOĆU 3D SENZORA

Postoji mnogo različitih 3D senzora za snimanje prostora, a najčešće korišteni su LIDAR (slika 2.1 lijevo), RBG-D kamera (slika 2.1 sredina) i stereo kamera (slika 2.1 desno). LIDAR se može podijeliti u dvije skupine: engl. *scannless* LIDAR koji snima cijelu scenu sa svakom zrakom i engl. *scanning* LIDAR koji snima točku po točku sa svakom zrakom svjetlosti, odnosno laserom. U RGB-D kamerama se tipično nalazi engl. *scannless* LIDAR te se one još nazivaju i engl. *time-of-flight* kamere (ToF). Postoje i RGB-D kamere koje rade na principu strukturirane svjetlosti. Princip rada za ToF kamere vrlo je jednostavan, za dobivanje dubine mjeri se vrijeme potrebno da se zraka svjetlosti odbije od objekta i vrati nazad do senzora. Ovakva slika naziva se dubinska slika i nalazi se na jednom kanalu izlaza, dok se na drugom kanalu RGB-D kamere slikaju standardne slike u boji. Snimanje dubine ToF kamerama vrlo je brzo jer se koristi samo jedna varijabla za izračun, vrijeme puta zrake svjetlosti. Iz ovog razloga ovakav je sustav vrlo povoljan za uporabu u stvarnom vremenu (engl. *real-time*). Jedan od najvećih nedostataka ovih kamera je problem reflektirajućeg materijala, zbog kojih je mogućnost pogreške pri snimanju dubine velika [1]. Drugi pristup je korištenje stereo kamere (stereo vizija), odnosno dvije kamere postavljene u istoj ravnini. Izračun udaljenosti ovim pristupom je složeniji jer je potrebno pronaći istu točku, za koju se računa udaljenost, snimljenu iz dva različita pogleda s dvije kamere. Zatim se pomoću pomaka odabrane točke na te dvije slike i parametara kamera može izračunati pozicija točke u prostoru [4].

Slika 2.1 LIDAR (Sick), RGB-D kamera (Asus Xtion Pro) i stereo kamera (Omega).
(Slike preuzete sa stranice: https://www.sick.com/at/en/detection-and-ranging-solutions/3d-lidar-sensors/mrs1000/c/g387152, https://www.asus.com/3D-Sensor/Xtion_PRO/, https://arcure.net/omega-stereo-camera-for-indoor-outdoor-applications)

2.1. Dubinska slika

Izlaz koji se dobije snimanjem kamerom koja u sebi ima senzor dubine (engl. *depth sensor*) naziva se dubinska slika. U njoj se nalaze vrijednosti koje odgovaraju udaljenosti odgovarajuće točke od kamere. Najčešće su prikazane u formatu s dvije boje, gdje se jednom bojom prikazuju najbliže točke na slici, a crnom bojom najudaljenije točke na slici, kao što je prikazano na slici 2.2, koja je dobivena Asus Xtion Pro Live kamerom. No, pri spajanju dubinske slike i RGB slike teško je dobiti vrijednosti koje u sebi nemaju pogrešku. Pogreška nastaje zbog interpolacije točaka s velikom razlikom u dubini, koje se nalaze na rubu objekta. Zbog toga primjenjuje se RANSAC metoda, koja će biti pojašnjena u sljedećem poglavlju, te se više dubinskih slika i RGB slika spajaju kako bi se dobio jedan oblak točaka [3].

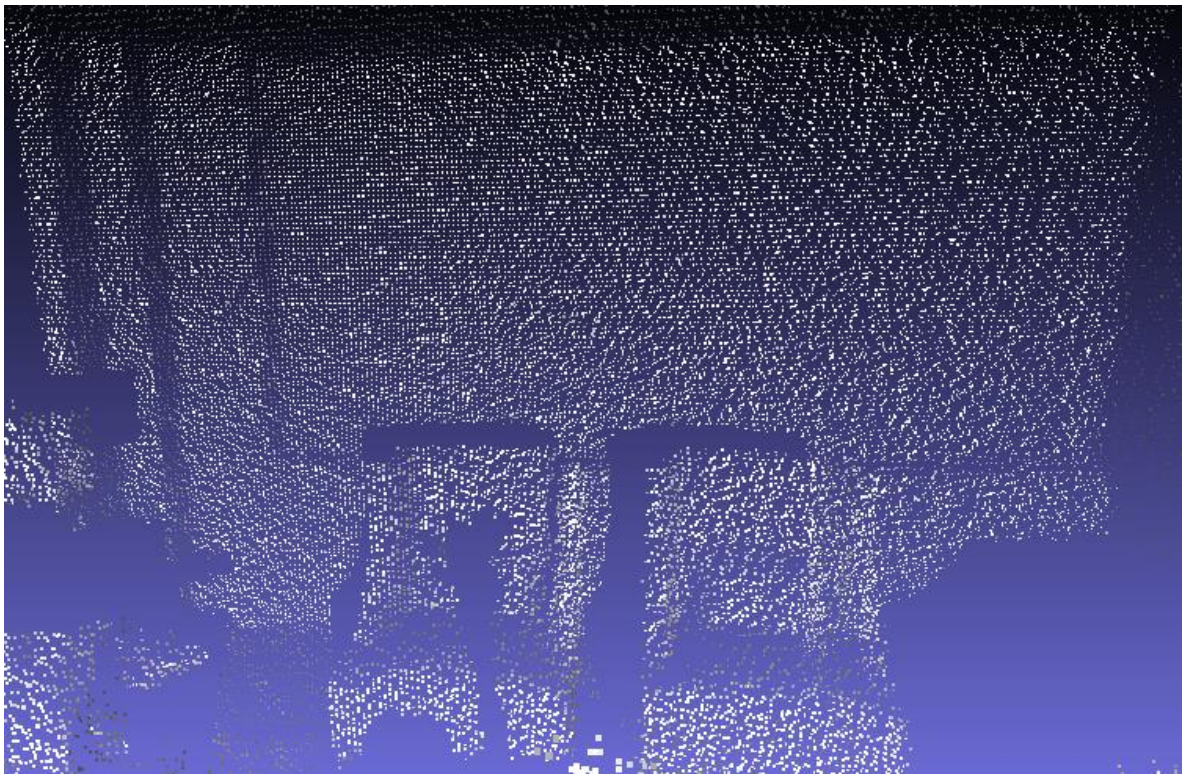


Slika 2.2 Dubinska slika.

2.2. Oblak točaka

Oblakom točaka moguće je prikazati sliku dobivenu RGB-D kamerom s vrlo velikom uštedom podataka. Pretvorbom u oblak točaka jedini potrebni podatci su pozicija i boja. Također, oblaci točaka dobiveni iz kamere su organizirani (strukturirani ili poredani), što znači da se iz jedne točke oblaka dobiva točno jedna točka u prostoru RGB slike i dubinske slike te da je svakoj od njih moguće jednostavno pronaći susjedne točke. Ovo svojstvo vrlo je važno u obradi oblaka točaka jer susjedne točke u 3D prostoru ujedno predstavljaju i susjedne točke na slici. Kako bi se

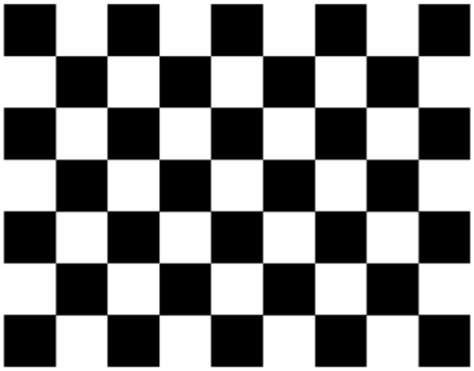
jedna scena prikazala oblakom točaka potrebno ju je slikati iz više različitih pogleda te nakon toga na dobivene slike primijeniti jednu od tehnika poravnanja oblaka točaka [1]. Prikaz jednog oblaka točaka moguće je vidjeti na slici 2.3. Dobiven je iz 3D modela prostorije i prikazan preko aplikacije Meshlab.



Slika 2.3 Oblak točaka.

2.3. Kalibracija RGB-D kamere

Postupak kalibracije kamere koristi se kako bi se kroz izračun unutarnjih parametara kamere poboljšala točnost dobivenih mjernih veličina snimljenih kamerom, kao što je udaljenost objekta od kamere. Razni postupci kalibracije kamere su do sada razvijeni i implementirani u raznim programskim bibliotekama, kao što su OpenCV i OpenNI te Matlab. Iako su tehnike različite sve se koriste slikama dobivenim kamerom, slike u boji (RGB) i dubinske slike te od korisnika zahtijevaju prikupljanje istih [4]. Također, mnogim je kalibracijskim postupcima zajednički kalibracijski panel koji nalikuje šahovskoj ploči, slika 2.4, sastavljena od niza crnih i bijelih kvadrata proizvoljnog broja redaka i stupaca te proizvoljne veličine samih kvadrata. Kalibracijski panel se koristi pri prikupljanju slika za kalibraciju u svrhu pokretanja jednog dijela kalibracije, algoritma za detekciju rubova ili vrhova.



Slika 2.4 Kalibracijski panel.
(Slika preuzeta sa stranice:
https://wiki.ros.org/camera_calibration/Tutorials/MonocularCalibration?action=AttachFile&do=view&target=check-108.pdf)

2.4. Aktivna vizija

Sustav aktivne vizije počeo se razvijati u kasnim 1980-tim godinama kada se došlo do zaključka da je pasivna vizija problematična te da se pomoću nje ne može doći do oblika objekta iz sjenčanja, kontura, tekstura i pokreta. U [5] teoretski su predstavili prvu ideju aktivne vizije i pružili kvalitetnu podlogu, uz matematička objašnjenja, za daljnji rad u ovom području. Od tada razvijeno je mnogo sustava aktivne vizije, a neki od najpoznatijih bit će predstavljeni u ovom poglavlju.

2.4.1. Pristup s autonomnim kamerama

Pristup se temelji na kamerama, stereo kamerama, koje se mogu same okretati u svojoj okolini. Za modeliranje kretanja praćenog objekata koristi se Kalmanov filtar. Točnije, koristi se žarišna duljina kamere koja minimizira nesigurnost mjerenja. U novije vrijeme sve je integrirano u jednom sustavu. Za dobivanje procjene položaja kamera-objekt i kontrole zumiranja, čiji se algoritmi nalaze u samoj kameri (engl. *ad hoc*), koristi se Pan-Tilt-Zoom mehanizam kamere prikazan na slici 2.5 [6]. Problem ovog sustava je pogreška koja se javlja pri svakom pomaku kamere. Zbog toga potrebna je česta kalibracija kamere.

Slika 2.5 Pan-Tilt-Zoom kamera Panasonic.
(Slika preuzeta sa stranice: https://na.panasonic.com/us/audio-video-solutions/broadcast-cinema-pro-video/professional-ptz-cameras/aw-ue150-4k-60p)

2.4.2. *Master/slave* pristup

Master/slave pristup koristi statičku kameru, master kamera, koja nadzire široko vidno polje i prati svaki pokret objekta od interesa. Kako bi što bolje pratio objekte, njihov se položaj šalje pokretnoj kameri, slave kameri, gdje su objekti od interesa promatrani u visokoj rezoluciji, dok je ostatak scene zamućen. Jedna od izvedbi ovog sustava prikazana je na slici 2.6. Obje kamere su kalibrirane tako da imaju zajedničku točku reference, kako bi razmjena podataka između njih bila jednostavnija. Iako je ovaj pristup jednostavan i učinkovit, nedostatak je potreba mapiranja dijelova slike kako bi pokretna kamera znala na kojoj se lokaciji nalazi objekt [6].

Slika 2.6 Master/slave kamera Hikvision.
(Slika preuzeta sa stranice: https://www.securityworldmarket.com/int/News/Product-News/hikvision-to-launch-a-new-master-and-slave-tracking-system)

2.4.3. Pristup mreže aktivnih kamera

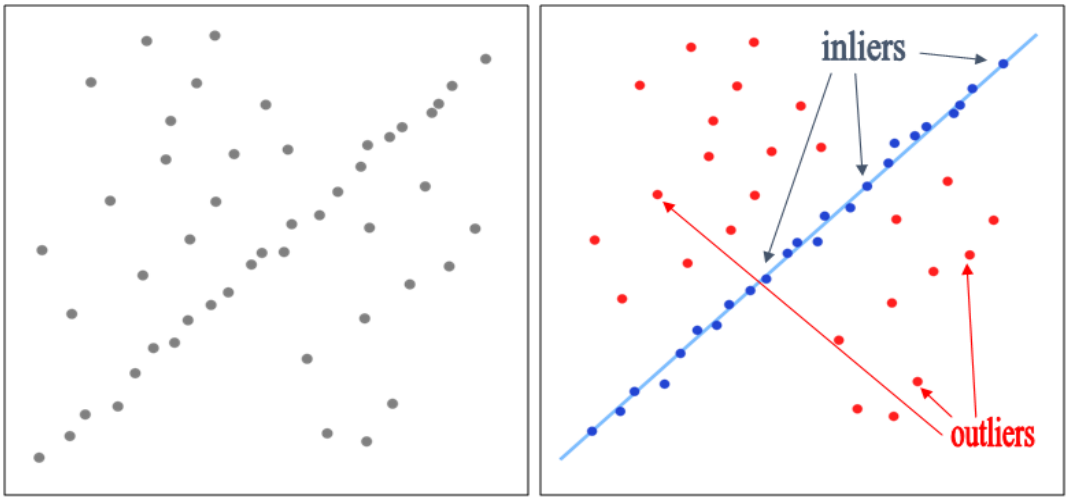
Kako bi se pokrio što veći prostor i zadržala visoka rezolucija razvio se pristup mreže kamera. Pristup se može temeljiti ili na pristupu s autonomnim kamerama ili na master/slave pristupu, a glavni problemi ovog pristupa su raspodjela i predaja poslova. Problem raspodjele poslova je odlučivanje koji resursi kamere će biti iskorišteni za pojedini zadatak, odnosno problem raspoređivanja kamera. S druge strane, problem predaje poslova mora opisati model po kojemu će kamere predavati poslove jedna drugoj. Pristup mreže aktivnih kamera teorijski zvuči privlačan, no kako bi ga se ostvarilo potrebna je velika količina novca za stvaranje takvog sustava i njegovo održavanje [6].

3. PORAVNANJE OBLAKA TOČAKA

Poravnanje oblaka točaka je proces sparivanja dva ili više različitih oblaka točaka u jedan, tj. proces rotiranja, skaliranja i translatiranja jednog oblaka točaka kako bi se određene točke poklopile s njima odgovarajućim točkama drugog oblaka. Nakon snimanja RGB-D kamerom kao izlaz dobivaju se dva kanala sa slikama, jedan za RGB sliku, a drugi za dubinsku sliku. Zatim slijedi izdvajanje značajki s obje slike, metodama kao što su SURF, SIFT ili ORB i njihovo sparivanje. No kako postoje pogreške pri sparivanju značajki s RGB i dubinske slike potrebno je koristiti RANSAC. Na kraju kako bi se dva oblaka točaka spojila koristi se iterativni algoritam najbliže točke (ICP) [3].

3.1. RANSAC

RANSAC (engl. *RANdom SAmple Consensus*) algoritam se koristi kako bi se smanjio utjecaj krivo detektiranih značajnih točaka. To je iterativni postupak kojim se procjenjuju parametri matematičkog modela koji najbolje odgovaraju promatranim podatcima [7]. Glavna pretpostavka algoritma je da se unutar promatranih podataka nalaze engl. *inliers*, točke koje odgovaraju zadanom matematičkom modelu i odudarajuće točke (engl. *outliers*).



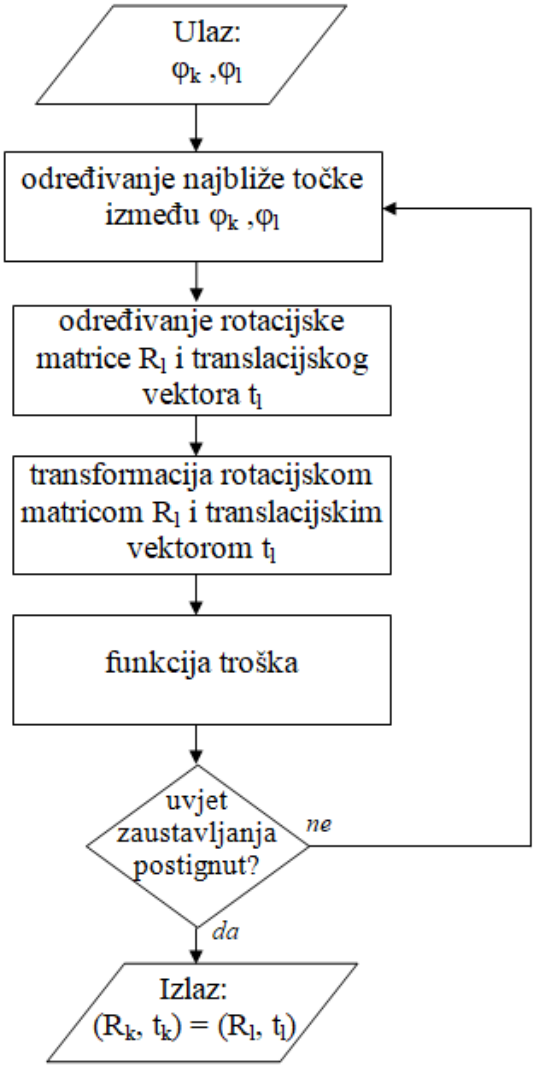
Slika 3.1 Primjer RANSAC algoritma.
(Slika preuzeta iz [8])

Za pronalaženje odgovarajućeg matematičkog modela, RANSAC u svakoj iteraciji uzima najmanji broj točaka potrebnih za definiranje tog modela slučajnim odabirom. Za detekciju pravaca u 2D skupu točaka RANSAC algoritam uzima dvije točke slučajnim odabirom. Kroz te dvije točke algoritam provlači pravac te ispituje jesu li ostale točke udaljene od toga pravca

unutar određenog praga. Ukoliko jesu, proglašava ih inlierima, a ostale točke koje su udaljenije od pravca proglašava odudarajućim točkama. Nakon zadanog broja iteracija za model postavlja onaj pravac koji je imao najviše inliera. Najveća prednost RANSAC algoritma je otpornost na šum, odnosno odudarajuće točke, dok je najveći nedostatak točno postavljanje praga. Ako je prag premali točke koje su zapravo inlieri bit će proglašene odudarajućim točkama i obratno, ako je prag prevelik točke koje su zapravo odudarajuće točke bit će proglašene inlierima [8].

3.2. Iterativni algoritam najbliže točke (ICP)

ICP (iterativni algoritam najbliže točke) jedan je od najkorištenijih algoritama za poravnanje trodimenzionalnih oblaka točaka. Stoga se razvilo puno tipova ovog algoritma: točka-u-točku (engl. *point-to-point*), točka-u-ravninu (engl. *point-to-plain*), linearni, nelinearni ICP algoritam itd. Pojednostavljeni prikaz dijagrama toka standardnog ICP algoritma prikazan je na slici 3.2.



Slika 3.2 Dijagram toka ICP algoritma.

Kao ulaz u algoritam koristi se izvorni i referentni oblak točaka (φ_k, φ_l), nakon čega se za točke referentnog oblaka točaka određuju najbliže točke prvog oblaka točaka. Rezultat ovog koraka su parovi korespondentnih točaka, gdje je jedan element para točka iz prvog oblaka točaka, a drugi element njoj najbliža točka iz drugog oblaka točaka. Sljedeći korak je određivanje rotacijske matrice R_l i translacijskog vektora t_l te transformacija drugog oblaka točaka koristeći R_l i t_l. Pomoću njih se zatim izračunava funkcija troška (engl. *cost function*) prikazana sljedećim izrazom:

$$E(R_l, t_l) = \sum_{m \in A_{kl}} (p_{k,m} - (R_l p_{l,m} - t_l))^2 \tag{3-1}$$

gdje $p_{k,m}$ predstavlja točku prvog oblaka točaka, $p_{l,m}$ točku drugog oblaka točaka, a A_{kl} predstavlja skup korespondentnih točaka između ta dva oblaka točaka. Zatim se provjerava je li uvjet za zaustavljanje postignut, odnosno je li funkcija troška manja od određenog praga. Ukoliko je, algoritam kao izlaz daje proračunatu rotacijsku matricu R_l i translacijski vektor t_l poravnatih oblaka točaka. U slučaju da uvjet za zaustavljanje nije postignut, odnosno funkcija troška nije manja od određenog praga, algoritam se vraća na određivanje novih najbližih točaka. Uvjet za zaustavljanje može biti i dostignuti broj iteracija [9]. Tehnika izračunavanja funkcije troška pokazala se najboljom za poravnanje dva oblaka točaka. U funkciju se također može implementirati odbacivanje odudarajućih točaka prije samog poravnanja kao i dodavanje težina na točke, što je glavna razlika između linearnog i nelinearnog ICP-a te točka-u-točku i točka-u-ravninu ICP-a [10]. Razlika između točka-u-točku i točka-u-ravninu ICP-a je korištenje normale u točki u ICP algoritmu točka-u-ravninu. Normala 3D točke je jedinični vektor koji je ortogonalan na ravninu dobivenu iz bliske okoline promatrane točke. Time točka postaje orijentirana 3D točka što omogućava lakše poravnanje dvaju pogleda. Točka-u-ravninu ICP algoritam minimizira udaljenost točke jednog oblaka točaka u odnosu na ravninu u kojoj leži točka drugog oblaka točaka okomitu na normalu pridruženu toj točki. S druge strane točka-u-točku ICP algoritam minimizira udaljenost točaka jednog oblaka točaka od njima najbližih točaka drugog oblaka točaka [11].

3.3. TEASER

TEASER je najnovija (engl. *state-of-the-art*) metoda poravnanja 3D oblaka točaka. Metoda se bazira na dva algoritma TRIM (engl. *Translation and Rotation Invariant Measurements*) i TIM (engl. *Translation Invariant Measurements*). Pomoću algoritma TRIM izračunava se skala scene

iz čega se zatim pomoću TIM algoritma izračunava rotacija te se na kraju pomoću izračunate rotacije dobiva translacija.

3.4. Metoda izračunavanja optimalnih položaja kamere

U ovom radu koristiti će se nova metoda za izračunavanje optimalnog položaja kamere predložena u [11]. Koristi se trenutni pogled kamere kako bi se odabrao novi optimalni kut gledanja za isto gledište. Točnije, određuje se optimalna orijentacija pantilt mehanizma koja je definirana kutom azimuta (engl. *pan*) i elevacije (engl. *tilt*). Svaka nova slika mora se preklapati s nekom od prethodno spremljenih slika, čime informacija o područjima preklapanja ostaje očuvana te omogućuje pouzdano poravnanje slika. Također, sljedeći kut gledanja mora pokriti što je moguće veći prostor još neistražene scene. Stoga se odabir novog smjera optimalnog kuta gledanja zasniva na sljedeća dva kriterija:

1) što veći postotak još neistražene scene treba biti sadržan u sljedećem pogledu i
2) dio scene koji je potpuno obuhvaćen i sljedećim i nekim od prethodnih pogleda sadrži dovoljno informacije za uspješno poravnanje.

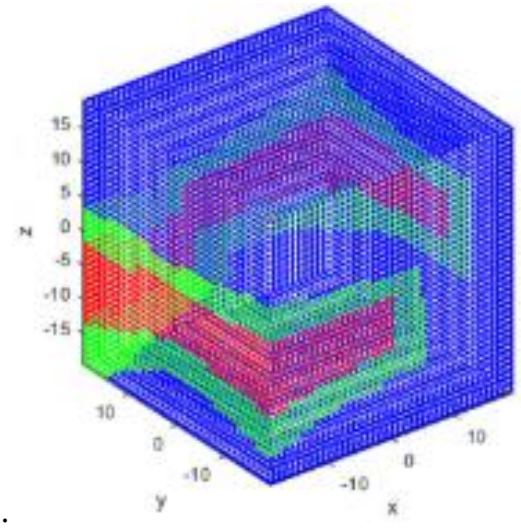
Za dvije gore navedene pretpostavke matematička formula glasi:

$$\varphi(\theta_i) = w_\chi \chi(\theta_i) + \max_{\varphi \in \Phi} \big(\psi(\theta_i, \varphi)\big) \tag{3-2}$$

gdje je w_χ empirijski težinski faktor, $\chi(\Theta_i)$ količina još neistraženog prostora obuhvaćenog pogledom Θ_i, a samo razmatrani pogled s dovoljno informacija o preklapanju s nekim od prijašnjih pogleda koji zadovoljava:

$$\max_{\varphi \in \Phi} \big(\psi(\theta_i, \varphi)\big) > w_\psi \tag{3-3}$$

gdje je $\psi(\Theta_i, \varphi)$ izračunati sadržaj preklopljenih informacija (OIC, engl. *overlap information content*), koji predstavlja preklapanje pogleda Θ_i sa svim prethodnim pogledima φ, a w_ψ predstavlja eksperimentalni prag. Za sljedeći pogled bit će odabran onaj pogled za koji je vrijednost $\varphi(\Theta_i)$ iz formule (3-2) najveća. Kako bi se utvrdio postotak još neistražene scene za sljedeći pogled svi prijašnji pogledi su spremljeni koristeći jediničnu kocku sa središtem u trenutnom gledištu. Nakon utvrđivanja sljedećeg pogleda kocka se ažurira. Prikaz jedinične kocke s deset spremljenih pogleda nalazi se na slici 3.3.



Slika 3.3 Jedinična kocka s 10 spremljenih pogleda.
(Slika preuzeta iz [11])

Crvenom bojom su prikazana područja sredine pogleda koja neće biti korištena za sljedeći pogled jer bi izazvala redundantne poglede. Zelenom bojom je prikazan cijeli pogled i područja koja će biti korištena za odabir novog pogleda, a plavom bojom je označeno još neistraženo područje scene.

3.5. Metoda određivanja korespondencija između točaka

Za određivanje korespondencije između točaka u 3D oblaku točaka, u ovom radu, koristiti će se engl. *point-to-plain* ICP algoritam predstavljen u [13]. Uobičajen pristup kod poravnanja oblaka točaka je smanjivanje udaljenosti između dvaju odgovarajućih točaka snimljenih iz dva različita pogleda na što manju vrijednost. To se postiže smanjivanjem funkcije:

$$E(R_l, t_l) = \sum_{(m,m')\in A_{kl}} w(m,m')[{}^{k}n_{m,k}^{T} R_k^T (R_l\, {}^{l}p_{m',l} + t_l - R_k^0\, {}^{k}p_{m,k} - t_k^0)]^2 \qquad (3\text{-}4)$$

gdje R_l i t_l predstavljaju rotacijsku matricu i translacijski vektor trenutnog pogleda, a R_k^0 i t_k^0 rotacijsku matricu i translacijski vektor prethodno spremljenog pogleda. Te varijable predstavljaju položaj kamere iz kojega je snimljen trenutni (*l*-ti), odnosno prethodni (*k*-ti) oblak točaka u odnosu na zajednički koordinatni sustav scene. Varijabla *w* je težina pridružena točki ovisna o nesigurnosti mjerenja položaja točke, dok varijabla $n_{m,k}$ predstavlja normalu točke *m*. Normala točke množi se s rotacijskom matricom i translacijskim vektorom *l*-tog i *k*-tog pogleda

kako bi se dobila udaljenost točke m' transformirane u koordinatni sustav točke m od ravnine u kojoj leži točka m, pri čemu je ta ravnima okomita na normalu točke m.

Svaki novi snimljeni oblak točaka poravnava se s jednim od prethodno snimljenih oblaka točaka, a kako bi se pogledi uspješno poravnali vrlo je važno točno odrediti koja točka trenutno snimljenog oblaka točaka odgovara istoj toj točki prethodno snimljenog oblaka točaka. Stoga algoritam za korespondenciju točaka ima tri uvjeta koja moraju biti ispunjena kako bi se točke smatrale korespondentnima:

1) euklidska udaljenost dvaju točaka $p_{m',l}$ i $p_{m,k}$ promatranih u istoj ravnini dobivenih iz dva različita pogleda mora biti manja od zadanog praga d_a (δ_{max} je 3^o u ovom radu),

$$\left\|p^l_{m',l} - p^l_{m,k}\right\| \leq d_a, \text{ gdje je } d_a = 2\left\|p^l_{m,k}\right\|\delta_{max} \tag{3-5}$$

2) razlika kutova normala dvaju točaka ${}^l n^T_{m',l}$ i ${}^l n^T_{m,k}$ mora biti manja od zadanog praga ω (10^o u ovom radu),

$$arccos({}^l n^T_{m',l}\ {}^l n_{m,k}) \leq \omega \tag{3-6}$$

3) razlika udaljenosti točaka ${}^l p_{m',l}$ i ${}^l p_{m,k}$ od kamere mora biti manja od praga r_z (0.03m u ovom radu).

$$\left\|\mathrm{p}^{\mathrm{l}}_{\mathrm{m}',\mathrm{l}}\right\| - \left\|p^l_{m,k}\right\| \leq r_z \tag{3-7}$$

Nakon što su korespondentne točke pronađene, svakoj točki dodaje se ICF (engl. *information content factor*). Veći ICF se pridružuje onoj točki koja leži na ravnini manjoj od najveće ravnine u području scene koje je zajedničko za oba pogleda, odnosno onoj točki koja je korisnija za poravnanje. Za poravnanje dva oblaka točaka potrebno je da oni sadrže dio scene s dvije ili više neparalelnih ravnina te da obje ravnine sadrže korespondentne točke. Kako bi se postigla računalna učinkovitost, korespondencije se uspostavljaju samo za manji broj nasumično izabranih točaka. Vjerojatnost izbora neke točke proporcionalna je ICF-u te točke. Rezultat ovog postupka omogućuje algoritmu bolji odabir točaka koje će se koristiti u poravnanju dva oblaka točaka.

Algoritmi temeljeni na minimizaciji najmanje kvadratne pogreške (engl. *least-squares optimization*) osjetljivi su na odudarajuće podatke te se stoga, kako bi se broj pogrešno pridruženih točaka smanjio, određuju su točke s višeznačnim korespondencijama. Radi se o

točkama koje zadovoljavaju tri prethodno navedena kriterija korespondencije, ali ne leže na istoj ravnini. Zbog toga je uvedena formula:

$$|n_m^T(p_{m'} - p_m)| \geq d_{p2p} \tag{3-8}$$

gdje je d_{p2p} eksperimentalno određen prag, za čiji se izračun koristi prag d_a definiran u (3-5). Na kraju se sve točke koje nemaju višeznačne korespondencije transformiraju i spajaju s najbližom odgovarajućom točkom u prozoru željenog pogleda čija je veličina definirana formulom:

$$w_w = 2f\delta_{max} \tag{3-9}$$

gdje je f žarišna duljina kamere. Rezultat je lista korespondentnih točaka dobivena iz dvaju različitih pogleda [11].

3.6. Metoda poravnanja više oblaka točaka

U ovom dijelu bit će pojašnjena metoda poravnanja oblaka točaka koja se koristi u ovom radu. Za poravnanje svih pogleda kao referentni pogled uzima se prvi snimljeni pogled te se ostali pogledi poravnavaju s obzirom na njegov koordinatni sustav. Susjednim pogledima smatraju se oni pogledi kojima OIC prelazi određeni prag, što je definirano formulom:

$$\psi(\varphi_k, \varphi_l) > w_\psi \tag{3-10}$$

gdje Φ predstavlja skup svih prikupljenih pogleda te je podijeljen u podskupove Φ_i, $i=2,\ldots,n_\Phi$, gdje Φ_1 sadrži samo prvi pogled. φ_l predstavlja jedan pogled iz podskupa Φ_1, a φ_k predstavlja jedan pogled iz podskupa Φ_k, ako on nije sadržan u skupu prethodnih pogleda i ima susjedan pogled u tom skupu. Pogledi se poravnavaju rekurzivnim postupkom, gdje je svaki pogled iz nekog podskupa Φ_i poravnat s jednim od svojih susjednih pogleda iz podskupa Φ_{i-1}. Poravnanje dva susjedna pogleda vrši se ICP algoritmom pojašnjenim u prethodnom potpoglavlju. Nakon posljednjeg poravnavanja svaki je pogled poravnat sa svojim susjedom. Kako bi se svi pogledi poravnali u jedan jedinstveni pogled za završno poravnanje koristi se formula:

$$T_k' = T_l' \cdot (T_l^0)^{-1} T_k \tag{3-11}$$

gdje T_k' i T_l' predstavljaju matrice homogene transformacije završnih pozicija dva pogleda, T_l^0 predstavlja početnu poziciju referentnog pogleda, a T_k predstavlja transformacijsku matricu

trenutnog pogleda nakon zadnjeg poravnanja opisanu s rotacijskom matricom i translacijskim vektorom [11]:

$$T_k = \begin{bmatrix} R_k & t_k \\ 0 & 1 \end{bmatrix} \tag{3-12}$$

4. IMPLEMENTACIJA ALGORITMA ZA IZGRADNJU 3D MODELA PROSTORIJA

Algoritam za izgradnju 3D modela prostorija napisan je u programskom jeziku C++, u razvojnom okruženju Visual Studio 2013. Kompletni algoritam se sastoji od dva dijela. U prvom dijelu se iz modela prostorije pomoću aktivne vizije stvaraju oblaci točaka, za predefinirani broj pogleda. Oblaci točaka se spremaju te se koriste u drugom dijelu algoritma. U ovom dijelu oblaci točaka se spajaju ICP algoritmom te se na kraju nastoji dobiti potpuni 3D model oblaka točaka prostorije. Za realizaciju algoritma koristi se nekoliko vanjskih biblioteka, koje će biti pojašnjene u nastavku ovog poglavlja. Cilj ovog diplomskog rada bio je u već postojeći programski kod, koji trenutno radi u simulacijskom režimu rada, unijeti mjerni šum koji je neizbježan kod rada sa stvarnim RGB-D kamerama. Time se osiguravaju realni uvjeti tijekom simulacije što dovodi do vjerodostojnijeg testiranja ispravnosti algoritma za poravnanje slika. Mjerni šum može izazvati neispravne korespondencije točaka što dovodi do neispravno poravnatih slika. Ispravnost ovog algoritma u konačnici utječe na ispravnost algoritma za stvaranje 3D modela prostorije.

4.1. RVL biblioteka (Robot Vision Library)

RVL biblioteka nastala je na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija u Osijeku. U njoj su sadržani različiti alati za aplikacije temeljene na računalnom vidu u području robotike. Neki od alata su: prepoznavanje objekata i prostora, segmentacija RGB slika itd. RVL biblioteka za pokretanje zahtijeva uključivanje još nekih biblioteka kao što su: OpenCV, OpenNI 2, PCL i VTK [14].

4.1.1. OpenCV

OpenCV (*Open Source Computer Vision Library*) je biblioteka otvorenog pristupa za aplikacije računalnog vida i strojnog učenja. Biblioteka ima preko 2500 razvijenih algoritama od onih osnovnih do novih (engl. *state-of-the-art*) algoritama računalnog vida i strojnog učenja. Neki od njih su: detekcija i prepoznavanje lica, stvaranje 3D oblaka točaka iz stereo kamera itd. [15].

4.1.2. OpenNI 2

OpenNI 2 je biblioteka otvorenog pristupa za obradu dubinskih slika snimljenih RGB-D kamerom. Aplikacija nudi mogućnosti različitih prikaza dubine (po boji) kao i prikaz spojene

dubinske slike s RGB slikom i njihov usporedni prikaz. Uz to aplikacija nudi i zrcaljenje slike, snimanje scene te još mnoge druge opcije [16].

4.1.3. VTK

VTK (*Visualization Toolkit*) je biblioteka otvorenog pristupa za računalnu 3D grafiku, modeliranje, procesuiranje i crtanje. Podržava široki spektar algoritama vizualizacije i naprednih tehnika modeliranja. Napravljena je da se može koristiti na svim platformama i iskoristiti prednosti niti (engl. *threads*) i procesa paralelne raspodjele memorije, koji utječu na brzinu i skalabilnost izvođenja algoritama [17].

4.2. PCL biblioteka (Point Cloud Library)

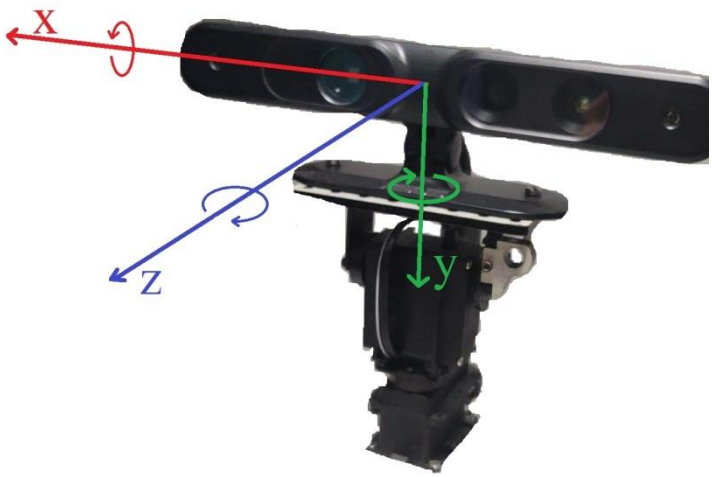
PCL je biblioteka otvorenog pristupa za obradu 2D i 3D slika i oblaka točaka. U biblioteci se nalaze brojni algoritmi, od kojih su neki: određivanje značajnih točaka, poravnanje oblaka točaka, segmentacija, vizualizacija itd. PCL biblioteka je stvorena za istraživačke svrhe i moguće je pokretanje na većini operacijskih sustava [18].

4.3. RVLLocalizationDemo

RVLLocalizationDemo je glavna aplikacija stvorena za razvoj i ispitivanje novih metoda 3D modeliranja. Sastoji se od dva dijela, aktivne vizije i rekonstrukcije prostora. Aktivna vizija se izvršava kružnim procesom gdje je prvi korak pomicanje kamere u zadani pogled pomoću pan-tilt mehanizma te snimanje tog pogleda (metoda *goTo_captureImage* u klasi *PanTilt*), a zatim se snimljeni pogled obrađuje (metoda *InformationContent* u klasi *View* i metoda *UnexploredRegions* u klasi *VNLocalizer*). Nakon toga, temeljem svih prethodno snimljenih pogleda generiraju se kandidati za sljedeći pogled te se ocjenjuju kako bi se dobio kandidat s najvećom ocjenom koji se potom odabire za novi pogled (metoda *CalculateScore* u klasi *VNLocalizer*). Odabirom novog pogleda aktivna vizija se vraća na prvi korak te se ponavlja sve dok se ne snimi *n* pogleda, gdje je *n* definiranih od strane korisnika.

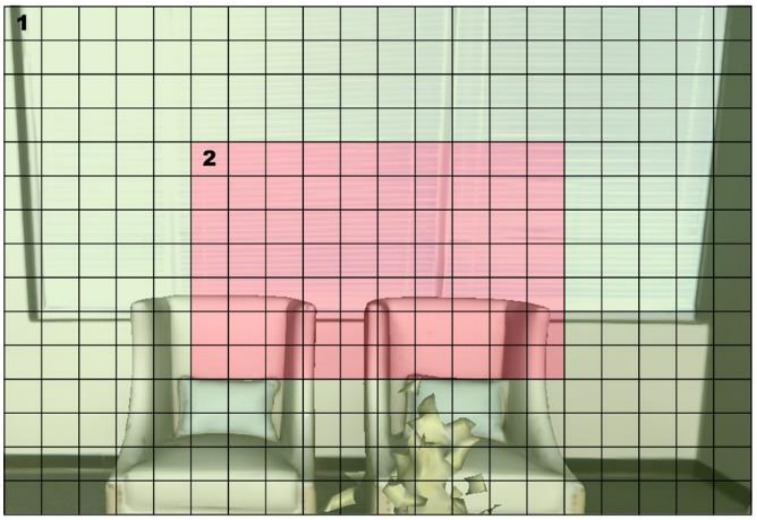
Glavni uvjet rada programa je stacionarna kamera koja se okreće oko svoje x osi (*tilt*) i y osi (*pan*), prikazana na slici 4.1. Prilikom rada sa stvarnom kamerom okretanje oko osi odrađuju motori glave i postolja na kojima se nalazi kamera. Takav sustav podložan je odstupanju zbog rada stvarnih motora te se stoga u simulaciju unosi nesigurnost tog mehanizma. Nesigurnost pan-tilt mehanizma dobivena je entropijski te iznosi +/- 3°, a implementirana je kao rotacija oko

nasumično odabrane osi kamere za nasumično odabrani kut. Slučajnim se odabirom odabire jedna os kamere oko koje se vrši rotacija u iznosu nesigurnosti mehanizma, +/- 3°. Uz to, zbog postizanja što realnije simulacije, prilikom generiranja dubinske slike dodaje se mjerni šum koji se pojavljuje u radu sa stvarnom dubinskom kamerom (funkcija *GenerateVTKPolyDataDepthImage_Kinect* u aplikaciji *RVLLocalizationDemo*).



Slika 4.1 Osi kamere, x os (*tilt*), y os (*pan*) i z os (*roll*).

Obradom snimljenih pogleda, zbog postizanja uspješne registracije, pogledi se dijele na područja od interesa. Ovo je važan korak jer za uspješno poravnanje pogleda ICP algoritam mora imati dovoljno točnih informacija. Dakle, pogledi koji se poravnavaju moraju imati dovoljno preklapanja (ne smiju biti razdvojeni, ali ne smiju biti ni previše preklopljeni). Projekcijom pogleda na jediničnu kocku moguće je odrediti područja interesa (metoda *CreateProjectionCube* u klasi *ProjectionCube*). Ćelije unutarnjeg dijela pogleda projiciranog na jediničnu kocku označavaju se crvenom bojom i pridružuje im se vrijednost 2, dok se ćelije vanjskog dijela pogleda projiciranog na jediničnu kocku označavaju zelenom bojom i pridružuje im se vrijednost 1. Podjela pogleda na područja interesa prikazana je na slici 4.2. Ovim postupkom osigurava se preklapanje u vanjskom dijelu pogleda, čime se sprječava gomilanje pogleda i redundancija informacije. Nakon što se novi pogled projicira na jediničnu kocku, gdje je središte jedinične kocke pozicija kamere, a središte pogleda žarište kamere, provjerava se preklapa li se taj pogled s nekim od prethodno spremljenih pogleda. Ukoliko preklapanje postoji, provjerava se u kojem se dijelu pogledi preklapaju te se zapisuje ono područje s većim indeksom područja (1 ili 2).



Slika 4.2 Područje interesa jednog pogleda.
(Slika preuzeta iz [11])

Kandidati za sljedeći pogled su oni elementi kocke koji se nalaze na vanjskom dijelu pogleda, odnosno imaju indeks područja 1. Kako bi se odredio najbolji kandidat, svaki kandidat se ocjenjuje na temelju dva kriterija. Koliko još neistraženog dijela jedinične kocke trenutni pogled prekriva te koliko pogodnih informacija on sadrži. Prvi kriterij koristi indekse područja kako bi se odabrao najbolji kandidat. Indeks područja 0 označava još neistraženi dio jedinične kocke te što više tih indeksa područja pogled pokrije to je on bolji kandidat. Drugi kriterij na novom pogledu traži ravninu okomitu na dominantnu ravninu prethodnog pogleda, čime se postiže ispravnost rada ICP algoritma. Najbolje ocijenjeni kandidat odabire se za novi pogled, odnosno kamera se okreće tako da odabrani kandidat bude u središtu novog pogleda. Nakon *n* snimljenih pogleda program prelazi iz aktivne vizije u rekonstrukciju prostora.

Rekonstrukcija prostora se provodi u dvije faze, gdje je prva uzorkovanje oblaka točaka, a druga određivanje korespondencije između preklapajućih oblaka točaka na temelju kojih će se izvršiti fuzija. Uzorkovanje oblaka točaka (metoda *ImageSampling* u klasi *VNLocalizer*) bitna je faza za točniji rad ICP algoritma. U fazi uzorkovanja točke koje u sebi sadrže bitne informacije za fuziju imaju veću težinu prilikom odabira. Odnosno, količina informacija koje točka u sebi sadržava određena je temeljem ravnine kojoj pripada. Ukoliko se preklapanje točaka dogodi u istoj ravnini pogreška translacije i rotacije može biti prevelika, stoga je potrebno osigurati preklapanje u najmanje dvije ravnine. Uzorkovanje oblaka točaka se sastoji od četiri koraka. Na početku iz svih prethodno spremljenih oblaka točaka detektiraju se preklapajuće točke između pogleda (metoda *ImagesOverlap* u klasi *VNLocalizer*). Nakon toga slijedi izračun matrice kovarijance preklapajućih točaka (metoda *OverlappingImagesCovariance* u klasi *VNLocalizer*), time se vodi

računa da se ne odbace one točke koje u sebi sadrže dovoljno informacija za uspješnu fuziju. Detekcijom dominantne ravnine u jednom pogledu, točke koje pripadaju drugoj dominantnoj ravnini prilikom uzorkovanja imaju veći faktor. Izračun ovog faktora, ICF-a (metoda *InforamtionContentFactor* u klasi *VNLocalizer*) pretposljednji je korak uzorkovanja oblaka točaka. Posljednji korak je samo uzorkovanje točaka (metoda *SamplePoints* u klasi *VNLocalizer*). Druga faza rekonstrukcije prostora je određivanje korespondencija (metoda *DataAssociation* u klasi *VNLocalizer*). Za svaku točku iz k-tog oblaka točaka traži se njena korespondentna točka u l-tom oblaku uzorkovanih točaka. Korespondentne točke su one točke za koje se u okolini promatrane točke (k-tog pogleda) pronađe odgovarajuća preklapajuća točka (l-tog pogleda) koja pripada istoj ravnini i ima najmanju euklidsku udaljenost od promatrane točke.

Na temelju pronađenih korespondentnih točaka provodi se točka-u-ravninu ICP algoritam (metoda *MultiPointCloudReg* u klasi *VNLocalizer*) s korisnički definiranim brojem iteracija. ICP algoritam pomoću transformacijske matrice *T* (u kojoj se nalazi rotacijska matrica *R* i translacijski vektor *t*) transformira promatrani oblak točaka kako bi se on što bolje poklopio s njegovim preklapajućim oblakom točaka. Nakon što su parovi oblaka točaka poravnati potrebno ih je poravnati u odnosu na globalni koordinatni sustav (metoda *MultiPointCloudCorrection* u klasi *VNLocalizer*). Nakon završenog broja iteracija svi oblaci točaka međusobno se preklapaju i poravnavaju u odnosu na globalni koordinatni sustav (metoda *FinalPoseCorrection* u klasi *VNLocalizer*). Ovim postupkom fuzija oblaka točaka je završena.

Sve metode koje se pozivaju sadržane su u klasama: *PanTilt, View, ProjectionCube* i *VNLocalizer,* definiranima u *RVLLocalization*. U klasi *PanTilt* koristi se metoda *goTo_captureImg* koja za ulazne parametre uzima kuteve azimuta i elevacije (engl. *pan* i *tilt*) kako bi se snimio trenutni pogled, 2D polje u koje će biti pohranjena dubinsku sliku i strukturu podataka koja predstavlja RGB sliku trenutnog pogleda (*dPan_in, dTilt_in, depthImage_array, pRGBImage*). Kao rezultat metoda popunjava 2D polje i strukturu s dijelom scene na koji je usmjerena kamera. Iz klase *VNLocalizer* pozivaju se sljedeće metode:

1) *UnexploredRegions* za ulazne parametre uzima jediničnu kocku, rotacijsku matricu i translacijski vektor koordinatnog sustava kamere u odnosu na globalni koordinatni sustav (*cube, R, t*). U ovom slučaju koordinatni sustav je u središtu jedinične kocke s osima okomitim na stranice te kocke. Uspoređuje se indeks područja ćelije trenutnog pogleda projiciranog na jediničnu kocku i indeks područja odgovarajuće ćelije u jediničnoj kocki s prethodno spremljenim pogledima te se kao rezultat dobije njihova maksimalna

vrijednost. Time se odabiru elementi koji bi mogli biti iskorišteni za sljedeći pogled koristeći aktivnu viziju.

2) *CalculateScore* za ulazne parametre uzima niz parova točaka, jediničnu kocku, veličinu jedinične kocke, prag preklapanja, neistraženo područje scene, faktor preklapanja, najbolje *pan* i *tilt* vrijednosti dobivene za sljedeći pogled i putanju do direktorija za spremanje (*viewDirectionCandidates, cube, cubeSize, overlapInfoThrIn, wUnexploredIn, wOverlapIn, bestPanTilt, resultsFolder*). Metoda računa koliko još neistraženog područja vidi novi pogled zbrajanjem elemenata jedinične kocke koje pogled obuhvaća, a prijašnji pogledi nisu. Ukupni rezultat pojedinog pogleda računa se pomoću korisnički definiranih faktora *wOverlapIn* i *wUnexploredIn* te što su oni veći više pridonose ukupnom rezultatu. Kao rezultat metoda daje vrijednosti *pan*-a i *tilt*-a za pogled kojem je rezultat metode najveći te ih sprema u datoteku zadanu *resultsFolder*-om.
3) *ImageSampling* je metoda za uzorkovanje slika te se u njoj pozivaju metode:
    1. *ImagesOverlap* je metoda koja kao rezultat stvara listu preklopljenih točaka između dva preklapajuća pogleda k i l te ju sprema u varijablu *O_kl_list* unutar objekta klase *Views*.
    2. *OverlappingImagesCovariance* je metoda koja, pomoću varijable *O_kl_list*, kao rezultat u objekt klase *Views* sprema matricu kovarijance preklapajućih točaka *overlappingView_kl*.
    3. *InformationContentFactor* je metoda koja, preko varijable *overlappingView*_kl, računa ICF za svaku točku te ju sprema u varijablu *overlappingViews_covariances* objekta klase *Views*.
    4. *SamplePoints* je metoda koja pomoću varijabla *O_kl_list* i *overlappingView_kl* uzorkuje točke te ih sprema u objekt *C_kl_list* klase *Views*.
4) *DataAssociation* je metoda koja se koristi za određivanje korespondentnih točaka između dva pogleda te kao rezultat u objekt *correspodance_list* klase *Views* sprema točke koje su međusobno korespondentne.
5) *MultiViewRegistration* za ulazne parametre uzima broj iteracija, *bool* parametre iz .cfg datoteke za mogućnost prikaza 3D modela i sparivanja dvaju pogleda, putanju do direktorija za spremanje i *bool* parametar za odabir geometrijskog prostora (*noIterations, visualizeRoom, visualizeCorrespondences, resultsFolder, bUVDspace*). Odabir geometrijskog prostora uveden je u program kako bi se moglo što bolje ispitati utjecaj šuma na algoritam [19]. Metoda također poziva i metode za poravnavanje dva pogleda međusobno (*MultiPointCloudReg*), poravnavanje dva pogleda u odnosu na globalni

koordinatni sustav (*MultiPointCloudCorrection*) i poravnavanje svih pogleda međusobno u odnosu na globalni koordinatni sustav (*FinalPoseCorrection*). Kao rezultat metoda daje prepravljene vrijednosti rotacijske matrice i translacijskog vektora (*R, t*) koordinatnog sustava kamere u odnosu na globalni koordinatni sustav (jediničnu kocku).

Sljedeća klasa je klasa *ProjectionCube*. Iz nje se poziva metoda *CreateProjectionCube* čiji je ulazni parametar pola veličine jedinične kocke (*cubeHalfSizeIn*), a kao rezultat metoda daje jediničnu kocku podijeljenu u ćelije (dva puta ulazni parametar plus jedna ćelija) s vrijednostima elemenata jednakim nuli. Kada se pogled projicira na jediničnu kocku tada više elemenata slike (engl. *pixel*) može pripadati jednoj ćeliji. Posljednja klasa je *View*, a metode su:

1) *LoadCameraFile_RGBD_PanTilt2Cube* čiji je ulazni parametar putanja do direktorija u kojemu su spremljeni prethodno snimljeni 3D oblaci točaka (*filePath*), a kao rezultat metoda ili daje vrijednosti *pan*-a i *tilt*-a za svaki pogled posebno te ih sprema u datoteku zadanu *filePath*-om ili iz datoteke zadane *filePath*-om učitava *pan* i *tilt* za svaki prethodno spremljeni pogled.
2) *LoadOverlapingViews* ima isti ulazni parametar, putanju do direktorija (*filePath*), a kao rezultat metoda ili popunjava varijablu objekta *view* iz klase *View* i daje poglede koji imaju preklapanje s trenutnim pogledom te ih sprema u datoteku zadanu *filePath*-om ili iz datoteke zadane *filePath*-om učitava s kojim pogledom trenutni pogled ima preklapanje.

Za realizaciju programskog rješenja bilo je potrebno dodati mjerni šum kamere u navedeni program za stvaranje 3D modela prostorije. U tu svrhu dodana je nova varijabla *noiseLevel* koja po svojoj vrijednosti odgovara proračunatoj vrijednosti šuma stvarne RGB-D kamere. Zbog lakšeg odabira razine šuma, bez potrebe mijenjanja koda, dodan je interaktivni dio koji od korisnika traži unos količine šuma, gdje su opcije: idealni slučaj (bez šuma), željeni slučaj (mali šum), realni slučaj (šum kamere) i neželjeni slučaj (veliki šum). Ovaj dio koda prikazan je na slici 4.3.

```
float noiseLevel = 0.003;
int num = 0;
do {
    printf("Enter noise level: \n\t0 - No noise \n\t1 - Small noise");
    printf("\n\t2 - Default noise(camera noise) \n\t3 - High noise\n");
    scanf("%d", &num);
} while (num > 3 || num < 0);
switch (num)
{
case 0: noiseLevel = 0.0;   break;
case 1: noiseLevel = 0.001; break;
case 3: noiseLevel = 0.005; break;
default: noiseLevel = 0.003; break;
}
```

Slika 4.3 Kod za korisnički unos šuma.

Dodavanje mjernog šuma na simulirane snimke odrađuje se u funkciji *GenerateVTKDepthImage_Kinect*. Deklaracija funkcije je prikazana na slici 4.4. Izlaz iz nje, *renderDepthImg*, vraća se nazad u funkciju koja je poziva *GenerateVTKPolyDataDepthImage_Kinect*, čija je deklaracija prikazana na slici 4.5. Funkcija *GenerateVTKDepthImage_Kinect* stvara dubinsku sliku u milimetrima, a funkcija *GenerateVTKPolyDataDepthImage_Kinect* stvara sve potrebne parametre za tu dubinsku sliku. Ovim funkcijama je ujedno odrađena i simulacija kamere. Parametri *w* i *h* predstavljaju dimenzije slika snimljenih kamerom, parametar *f* predstavlja žarišnu duljinu kamere, a parametri *cx* i *cy* predstavljaju dimenzije dubinskih slika snimljenih kamerom.

```
cv::Mat GenerateVTKDepthImage_Kinect(
    vtkSmartPointer<vtkRenderWindow> renWin,
    double scale,
    double clipnear,
    double clipfar,
    int w = 640,
    int h = 480,
    double f = 543.1221626989097f,
    double cx = 317.2825290065861f,
    double cy = 240.955527515504f,
    float *R = NULL,
    float *t = NULL,
    float noiseLevel = 0.003);
```

Slika 4.4 Deklaracija funkcije *GenerateVTKDepthImage_Kinect*.

```
cv::Mat GenerateVTKPolyDataDepthImage_Kinect(
    vtkSmartPointer<vtkPolyData> pd,
    double scale = 1.0f,
    double clipnear = 0.0f,
    double clipfar = 0.0f,
    int w = 640,
    int h = 480,
    double f = 543.1221626989097f,
    double cx = 317.2825290065861f,
    double cy = 240.955527515504f,
    float *R = NULL,
    float *t = NULL,
    float noiseLevel = 0.003);
```

Slika 4.5 Deklaracija funkcije *GenerateVTKPolyDataDepthImage_Kinect.*

No kako šum nije uvijek uniforman i pošto Gaussova razdioba bolje opisuje mjerni šum od jednolike razdiobe, potrebno je implementirati generator slučajnih brojeva iz Gaussove razdiobe. Korištenjem već gotovih funkcija za stvaranje Gaussove razdiobe te množenjem dobivenog slučajnog broja iz razdiobe s odabranom razinom šuma, stvarni šum je implementiran u algoritam. Inicijalizacija stvaranja slučajnog broja iz Gaussove razdiobe prikazana je na slici 4.6, a postupak unosa šuma u mjerenje je prikazan na slici 4.7.

```
std::default_random_engine generator;
std::normal_distribution<double> distribution(0.0, 1.0);
double randGauss = 0.0;
double noise = 0.0;
```

Slika 4.6 Inicijalizacija Gaussove razdiobe.

```
randGauss = distribution(generator);
noise = noiseLevel*randGauss;
tempd = ((d[y * width + x ]+ noise) * (1.0 / clipfar - 1.0 / clipnear) * clipnear + 1.0) / clipnear;
renderedDepthImg.at<uint16_t>(y, x) = (uint16_t)((1.0 / tempd) * scale); //in milimeters
```

Slika 4.7 Unos razine mjernog šuma u mjerenje.

Kako bi se pojednostavilo unošenje željenih 3D modela prostorija u algoritam i kako bi se nakon završetka algoritma pronašli njegovi rezultati, u posebnoj .cfg datoteci bilo je potrebno unijeti putanje do željenih direktorija na vlastitom računalu, što je prikazano na slici 4.8. Parametar *SceneSequenceFileName* pokazuje na direktorij u kojemu se nalazi tekstualna datoteka s popisom svih oblaka točaka snimljenih za jednu prostoriju. Ova putanja se koristi kada su oblaci točaka već obrađeni navedenim algoritmom. Parametar *ActiveVisionPLYFilePath* pokazuje na direktorij u kojemu se nalazi 3D model prostorije, a *ResultsFolder*, u slučaju uključenog parametra *activeVision*, pokazuje na direktorij u koji se spremaju oblaci točaka dobiveni algoritmom. U slučaju isključenog parametra *activeVision* u direktorij se spremaju zapisi

rezultata dobivenih algoritmom, npr. konačna rotacijska matrica *R* i translacijski vektor *t*. Ove dvije putanje se koriste za prvi dio algoritma.

```
Save PLY : no

SceneSequenceFileName: D : \Diplomski\Rez\room1\sceneSequence.txt

%ActiveVisionPLYFilePath: C : \Dip\replica\room_2\mesh.ply
ActiveVisionPLYFilePath : C : \Dip\replica\office_3\mesh.ply

ResultsFolder : D : \Diplomski\Rez\test
```

Slika 4.8 Postavljanje putanja na odgovarajuće direktorije.

U istoj .cfg datoteci potrebno je navesti parametre aktivne vizije, koji su prikazani na slici 4.9. *overlapInfoThreshold* govori o pragu minimalnog postotka informacija iz nekog prethodnog pogleda kojeg mora sadržavati novi pogled kako bi se pogledi poravnali. Ukoliko je vrijednost ispod praga pogledi se ne preklapaju dovoljno dobro te trenutni pogled neće biti kandidat za novi pogled. *activeVision* je parametar koji pokreće aktivnu viziju, dok *activeVisionSimulation* služi za pokretanje programa u simulacijskom režimu rada. Ako je postavljen u *no* program koristi stvarnu RGB-D kameru na pantilt mehanizmu, a ako je parametar postavljen u *yes* tada program učitava 3D model prostorije. Unutar modela stvaraju se različiti pogledi i odgovarajući oblaci točaka za tu prostoriju simuliranjem okretanja kamere. Kada je parametar *visualizeCorrespondences* aktiviran, prilikom izvođenja programa prikazuju se redom svi pogledi na scenu odabrani aktivnom vizijom, a parametrom *visualizeRoom* aktivira se prikaz završnog spajanja svih pogleda u jedan te se prikazuje 3D oblak točaka prostorije dobivene obradom 3D modela navedenim algoritmom. Zadnja dva parametra *numberOfViews* i *numberOfIterations* određuju koliko će različitih pogleda biti stvoreno i broj iteracija ICP algoritma koji spaja poglede u jedinstveni 3D oblak točaka.

```
overlapInfoThreshold: 0.01
wUnexplored : 1
wOverlap : 1
activeVision : yes
activeVisionSimulation : yes
visualizeCorrespondences : no
visualizeRoom : yes
numberOfViews : 25
numberOfIterations : 2
```

Slika 4.9 Parametri aktivne vizije.

No, kako je to zasebna datoteka potrebno je sve navedene parametre iz .cfg datoteke stvoriti i inicijalizirati na predefinirane vrijednosti, slika 4.10 i slika 4.11.

```
pParamData = pParamList->AddParam("overlapInfoThreshold", RVLPARAM_TYPE_FLOAT, &overlapInfoThreshold);
pParamData = pParamList->AddParam("wUnexplored", RVLPARAM_TYPE_FLOAT, &wUnexplored);
pParamData = pParamList->AddParam("wOverlap", RVLPARAM_TYPE_FLOAT, &wOverlap);
pParamData = pParamList->AddParam("activeVision", RVLPARAM_TYPE_BOOL, &bActiveVision);
pParamData = pParamList->AddParam("activeVisionSimulation", RVLPARAM_TYPE_BOOL, &bActiveVisionSimulation);
pParamData = pParamList->AddParam("visualizeCorrespondences", RVLPARAM_TYPE_BOOL, &bVisualizeCorrespondences);
pParamData = pParamList->AddParam("visualizeRoom", RVLPARAM_TYPE_BOOL, &bVisualizeRoom);
pParamData = pParamList->AddParam("numberOfViews", RVLPARAM_TYPE_INT, &noGeneratedViews);
pParamData = pParamList->AddParam("numberOfIterations", RVLPARAM_TYPE_INT, &noIterations);
```

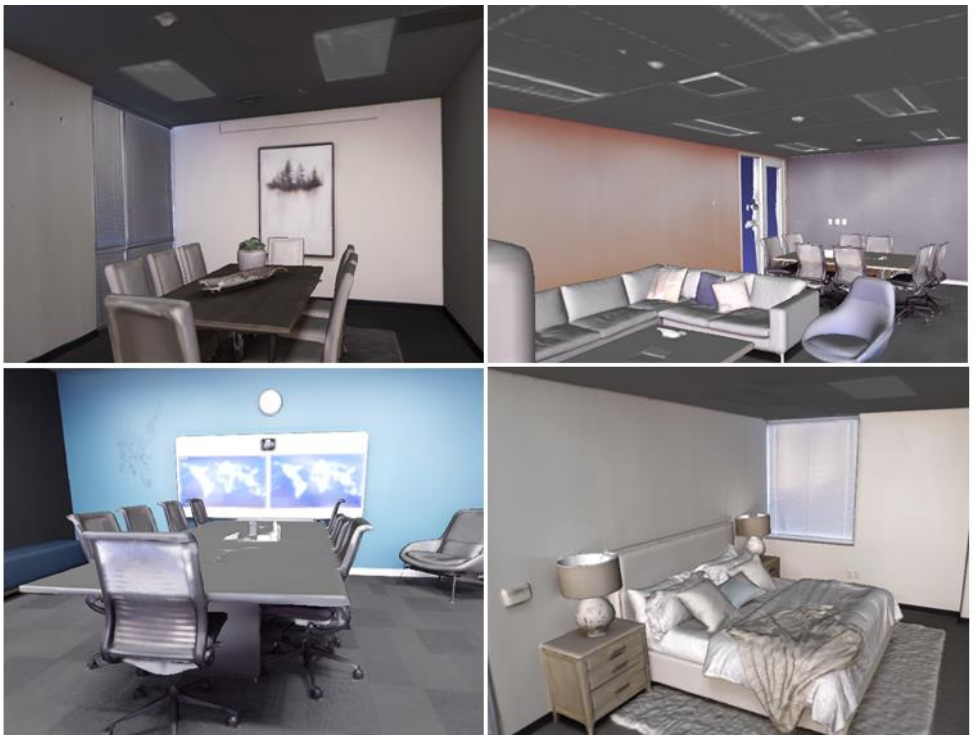
Slika 4.10 Kreiranje parametara .cfg datoteke za aktivnu viziju u algoritmu.

```
float overlapInfoThreshold = 0.01f;
float wUnexplored = 1.0f;
float wOverlap = 1.0f;
bool bActiveVision = false;
bool bActiveVisionSimulation = false;
bool bVisualizeCorrespondences = false;
bool bVisualizeRoom = false;
int noGeneratedViews = 2;
int noIterations = 10;
```

Slika 4.11 Inicijalizacija parametara .cfg datoteke za aktivnu viziju u algoritmu.

5. EKSPERIMENTALNA EVALUACIJA

U svrhu prikazivanja točnosti algoritma, za testiranje njegove ispravnosti i procjenu pogreške korišteni su različiti modeli prostorija iz Replica baze (*office_1*, *office_4*, *room_1* itd). Replica je baza različitih modela unutrašnjih prostora. Svaki model je visoke kvalitete s visokom rezolucijom i velikim rasponom tekstura, uključujući staklene te zrcalne površine kao i drvene te tekstilne površine [20].



Slika 5.1 Replica DataSet modeli prostorija.

Kako bi se simuliralo stvaranje 3D model prostorije iz danog 3D modela iz Replica baze bilo je potrebno prikupiti oblake točaka snimljene iz više različitih pogleda. Broj pogleda također utječe na ispravnost i mogućnost procjene pogreške te stoga i on varira (10, 20 i 30) za isti model prostorije. Zadnji parametar koji utječe na ispravnost testa i pojavu pogreške je mjerni šum te se i njegove vrijednosti mijenjaju tijekom testova (0, 0.003, 0.005).

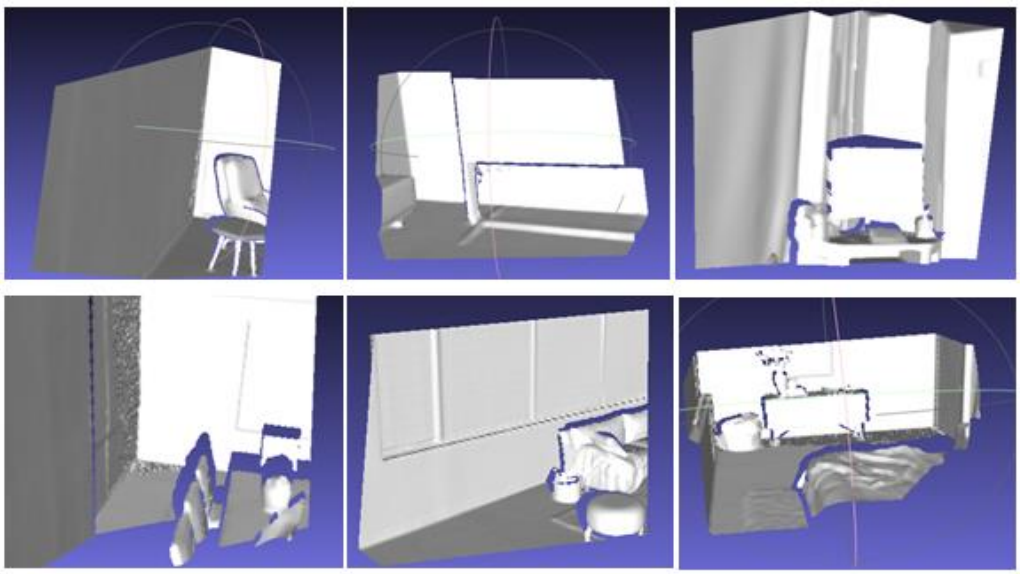
Kako bi program mogao raditi s proizvoljnim brojem pogleda potrebno je tijekom stvaranja slika, aktivne vizije, napraviti njihovo spremanje. Potrebno ih je spremiti pod pravim imenom, sl-0000, gdje broj predstavlja redni broj slike. Uz to, potrebno je stvoriti i novi .txt dokument koji govori o pantilt parametrima za svaki pogled posebno, sl-0000-O (primjer sadržaja dokumenta: 0 0 0 -35.000000 -31.000000 0.000000 0 A, gdje je -35 parametar *pan*, a -31

parametar *tilt* u stupnjevima, dok ostali parametri nisu relevantni za ovaj rad). Također, stvoren je još jedan .txt dokument koji govori o broju ostalih pogleda koji imaju predefinirani koeficijent preklapanja s nekim od do sada snimljenih pogleda u nekom dijelu trenutnog pogleda, sl-0000-overlap (primjer sadržaja dokumenta: 2, gdje taj parametar znači da postoje dva prije snimljena pogleda koja se preklapaju s njim). Prikaz navedenih dokumenata, spremljenih u jednoj mapi, vidljiv je na slici 5.2.

sl-00000	06-Sep-20 20:07	PLY File	4,325 KB
sl-00000-O	06-Sep-20 20:07	Normal text file	1 KB
sl-00000-overlap	06-Sep-20 17:41	Normal text file	0 KB
sl-00001	06-Sep-20 20:07	PLY File	4,326 KB
sl-00001-O	06-Sep-20 20:07	Normal text file	1 KB
sl-00001-overlap	06-Sep-20 17:41	Normal text file	1 KB

Slika 5.2 Dokumenti spremljeni nakon algoritma aktivne vizije.

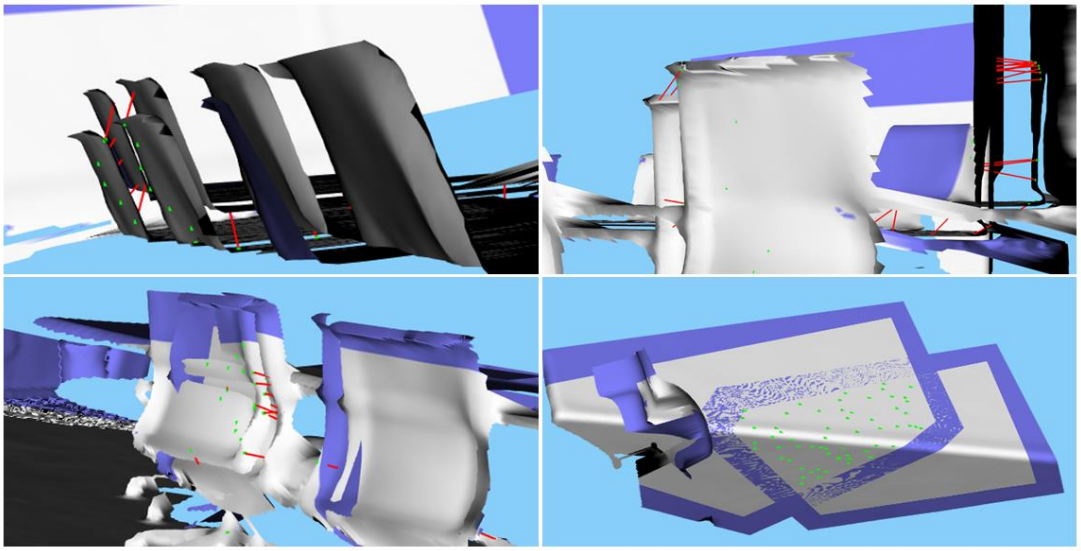
U prvom testu korišten je velik broj scena bez šuma kako bi se na najjednostavnijem testu utvrdila ispravnost algoritma. Na slici 5.3 prikazani su, nasumičnim odabirom, dijelovi prostorije dobiveni iz svih modela prostorija koje su korištene u ovim testovima.



Slika 5.3 Dijelovi prostorije dobiveni algoritmom bez mjernog šuma.

Nakon prikupljanja određenog broja pogleda, algoritam prelazi na spajanje pojedinih dijelova prostorije s dijelovima koji imaju zajednički dio, preklopljeno područje. Dio izlaza algoritma prikazan je na slici 5.4. Dva pogleda se sjedinjuju u jedan te se značajne točke prikazuju zelenom bojom, a u slučaju da se te točke ne nalaze na istom mjestu klasa *Visualizer* (sadržana unutar

RVLPCSegment aplikacije) ih spaja crvenom linijom te u sljedećim koracima nastoji minimizirati udaljenost između tih točaka. Crvene linije nisu vidljive na donjem desnom dijelu slike 5.4 jer je algoritam ispravno sparo značajne točke na oba pogleda, dok na ostalim dijelovima slike to nije tako te su crvene linije vidljive.



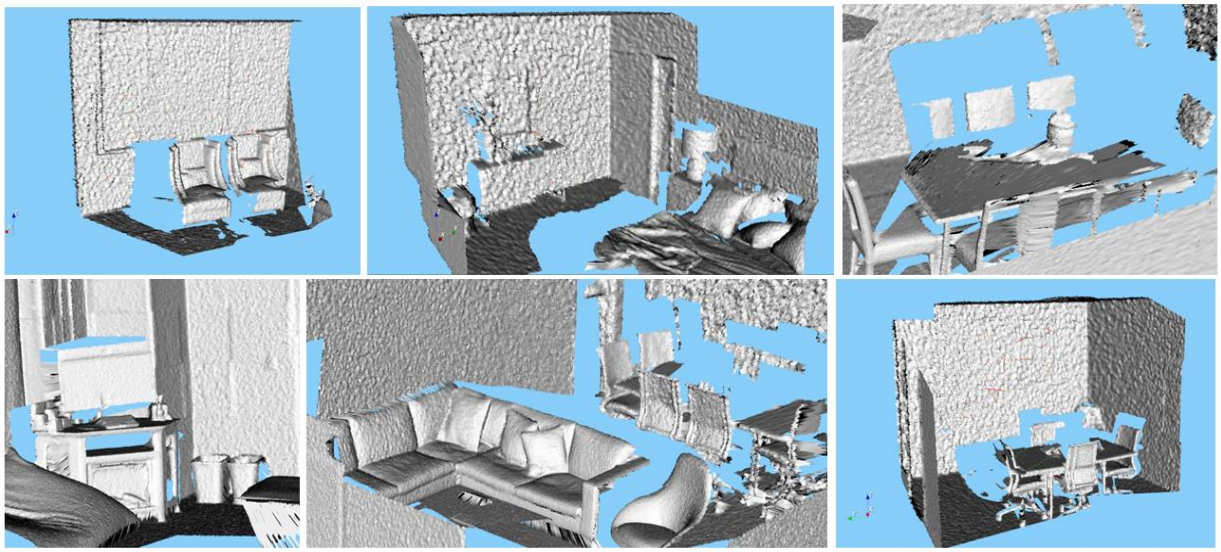
Slika 5.4 Dio izlaza iz algoritma za korespondenciju točaka.

Na slici 5.5 na lijevoj strani su prikazane rekonstruirane prostorije dobivene navedenim algoritmom, dok su na desnoj strani prikazani odgovarajući modeli iz Replica baze. Modeli korišteni iz Replica baze su: *office_4*, *office_3*, *office_1*, *room_2*, *room_1* i *room_0* te su tim redom i prikazani na slici.

Slika 5.5 Prikaz algoritmom rekonstruirane 3D prostorije (lijevo) i odgovarajući model iz Replica baze (desno).

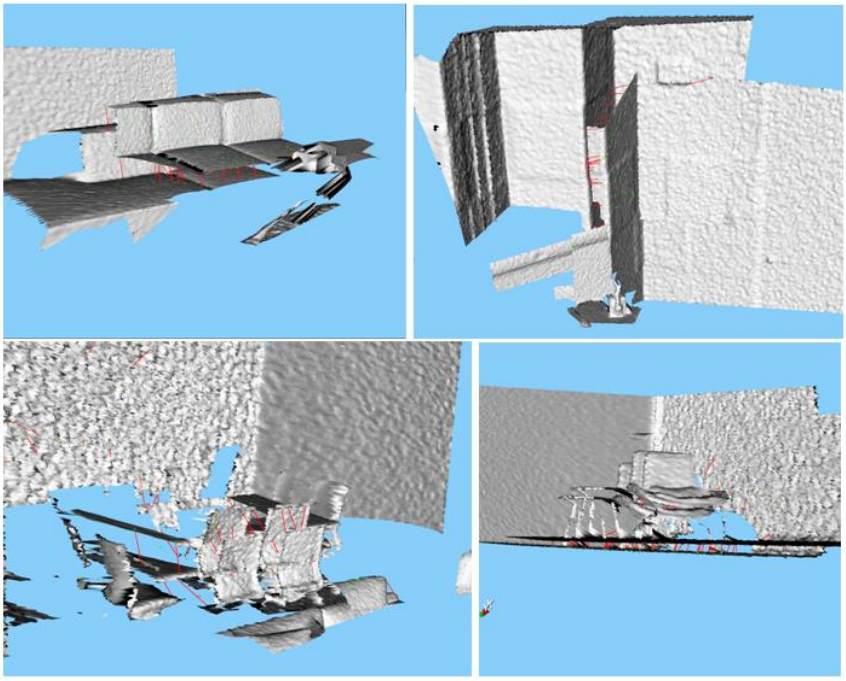
S obzirom da je prvi test najjednostavniji, algoritam se pokazao ispravnim i očekivani rezultat je postignut. Registracija i preklapanje su u potpunosti bili uspješni. To je i vidljivo na dobivenom 3D modelu prostorije prikazanom na prethodnoj slici. Jedini vidljivi nedostatak algoritma je problem zaklanjanja. Kada jedan objekt zaklanja onaj iza njega čini model nepotpunim u odnosu na onaj iz Replica baze, no to nije moguće izbjeći kada se scene snima iz jedne točke gledišta.

U sljedećem testu simuliran je stvarni šum, razine 0.003, koji se pojavljuje pri mjerenju dubine u kamerama zasnovanim na triangulaciji, a uz to korištena je i srednja količina pogleda (20). Slika 5.6 na identičan način kao i slika 5.3 prikazuje dijelove prostorija, odabranih nasumično, dobivenih iz svih modela prostorija koje su korištene u ovim testovima.



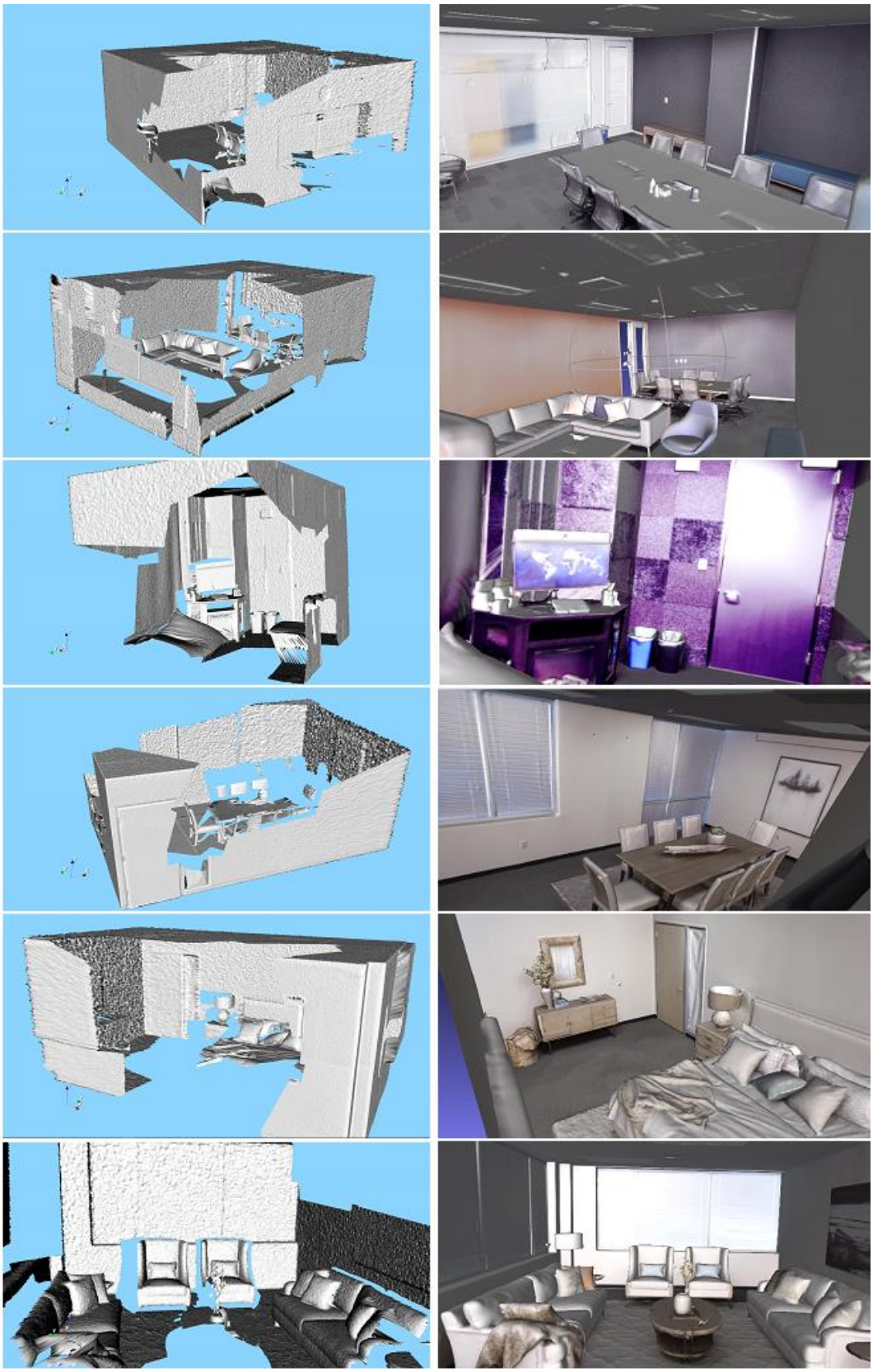
Slika 5.6 Dijelovi prostorije dobiveni algoritmom s razinom mjernog šuma od 0.003.

Zbog prisutnosti šuma očekivano je da će algoritam za poravnanje imati veću razinu pogreške nego u prvom testu. Iako je upravo to vidljivo na svim dijelovima slike 5.7, osim na gornjem desnom dijelu slike, rezultati poravnanja su zadovoljavajući.



Slika 5.7 Dio izlaza iz algoritma za korespondenciju točaka s razinom mjernog šuma 0.003.

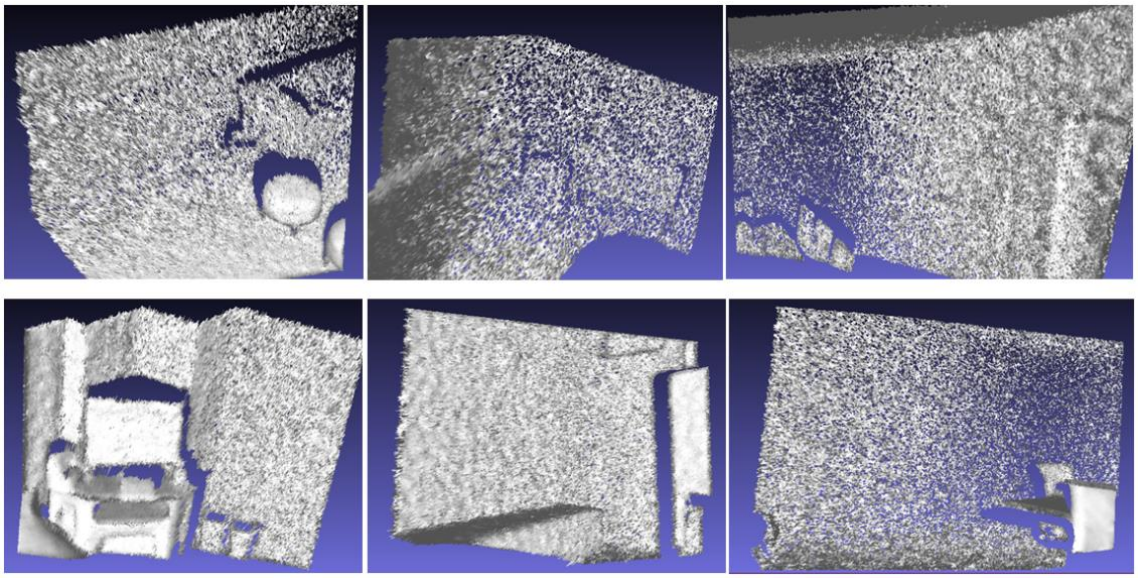
Slika 5.8 prikazuje prostorije rekonstruirane s algoritmom i mjernim šumom s razinom 0.003. Navedeni šum se koristi kako bi se u ovom testu simulirao šum stvarne kamere. Ovako rekonstruirane prostorije prikazane su na lijevoj strani slike, dok su odgovarajući modeli (istim redom kao na slici 5.5) iz Replica baze prikazani na desnoj strani slike.



Slika 5.8 Prikaz algoritmom rekonstruirane 3D prostorije s razinom mjernog šuma od 0.003 (lijevo) i odgovarajući model iz Replica baze (desno).

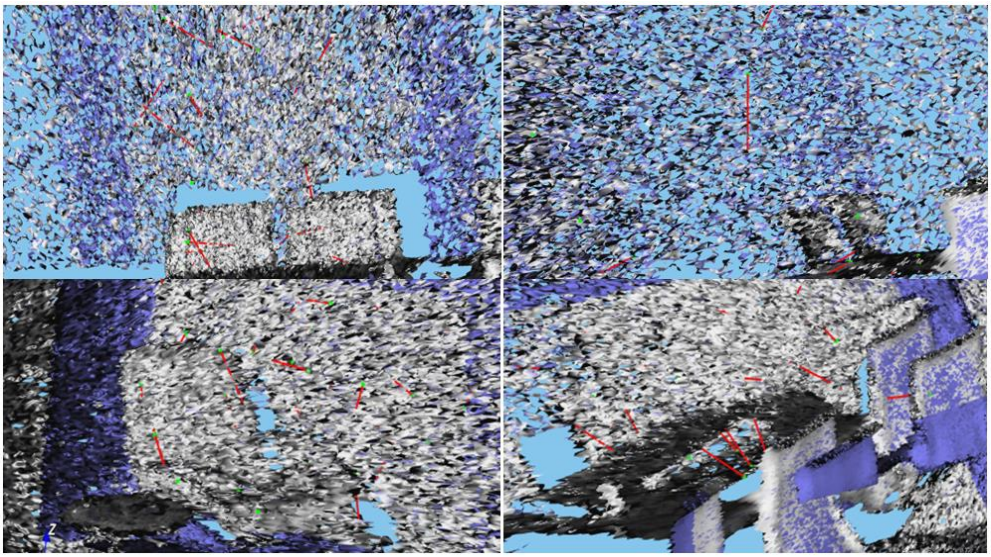
Uočeni nedostatak se i dalje pojavljuje te uključujući i njega dobiveni model i dalje postiže očekivane rezultate. Iako je to na slici 5.7. slabo vidljivo, algoritam daje zadovoljavajuće rezultate za korespondenciju točaka i u potpunosti uspješno stvara 3D model, uz gubitak određenih detalja na njemu.

Posljednji test je stvoren da ispita granice algoritma s razinom šuma od 0.005 i malim brojem pogleda (10). Pretpostavka za ovaj test je neuspješno sparivanje značajnih točaka dvaju pogleda te model prostorije koji nije zadovoljavajuć. Na slici 5.9 prikazani su nasumični dijelovi prostorije dobiveni iz svih modela prostorija korištenih u ovom testu.



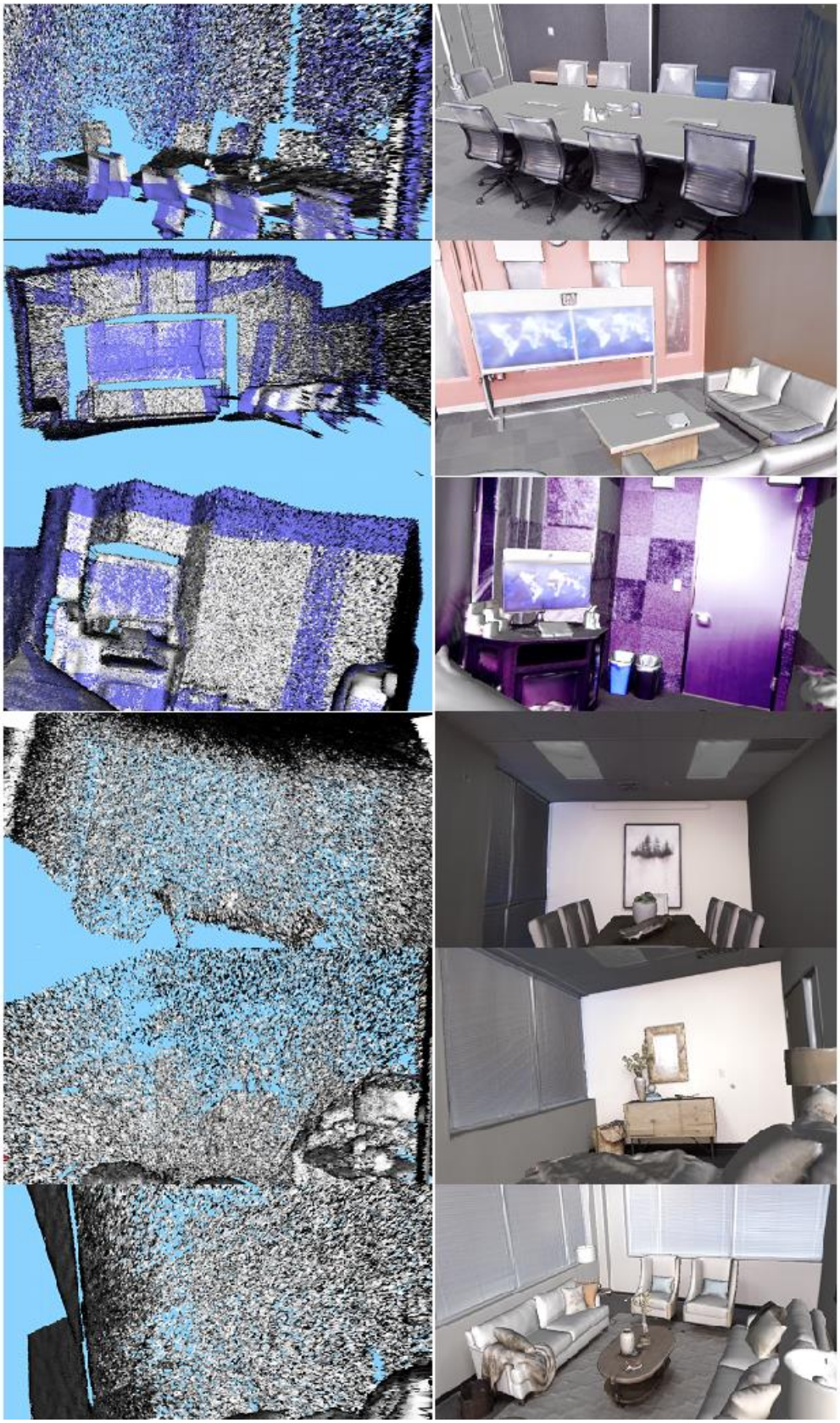
Slika 5.9 Dijelovi prostorije dobiveni algoritmom s razinom mjernog šuma od 0.005.

Kao što je pretpostavljeno mjerni šum je izazvao značajne pogreške tijekom provođenja algoritma za sparivanje značajnih točaka, što je prikazano na slici 5.10 no ipak je algoritam uspješno spojio dva pogleda, čime je ovaj dio pretpostavke bio netočan.



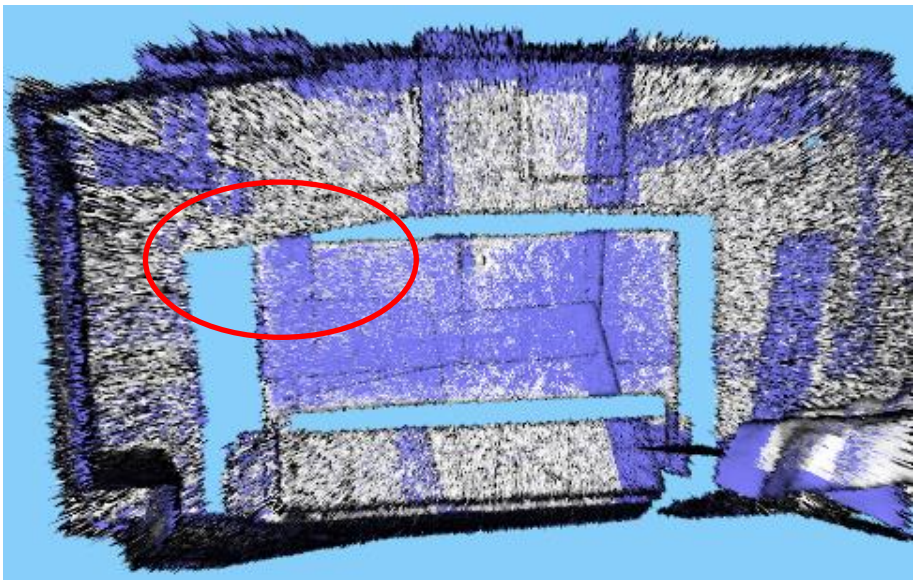
Slika 5.10 Dio izlaza iz algoritma za korespondenciju točaka s razinom mjernog šuma 0.005.

3D modeli rekonstruiranih prostorija vidljivi su na slici 5.11 te iako su nepotpuni jer pokazuju samo dio prostorije, imaju prihvatljivu točnost korespondencije točke.



Slika 5.11 Prikaz algoritmom rekonstruirane 3D prostorije s razinom mjernog šuma od 0.005 (lijevo) i odgovarajući model iz Replica baze (desno).

S obzirom na broj pogleda koji se koristio u ovom testu algoritam je pokazao da se s time može nositi pa čak i uz veliki mjerni šum. Iako 3D modeli prostorija nisu potpuni, dijelovi prostorija se mogu prepoznati u usporedbi s njihovim odgovarajućim modelima. Uz nedostatak dijelova prostorija vidljiv je i minimalan pomak u prostornim ravninama te zakrivljenost inače ravnih dijelova, najbolje vidljivo na slici 5.12. Zaokruženi dio je inače ravni dio zida, ali sada je prikazan kao zakrivljen.



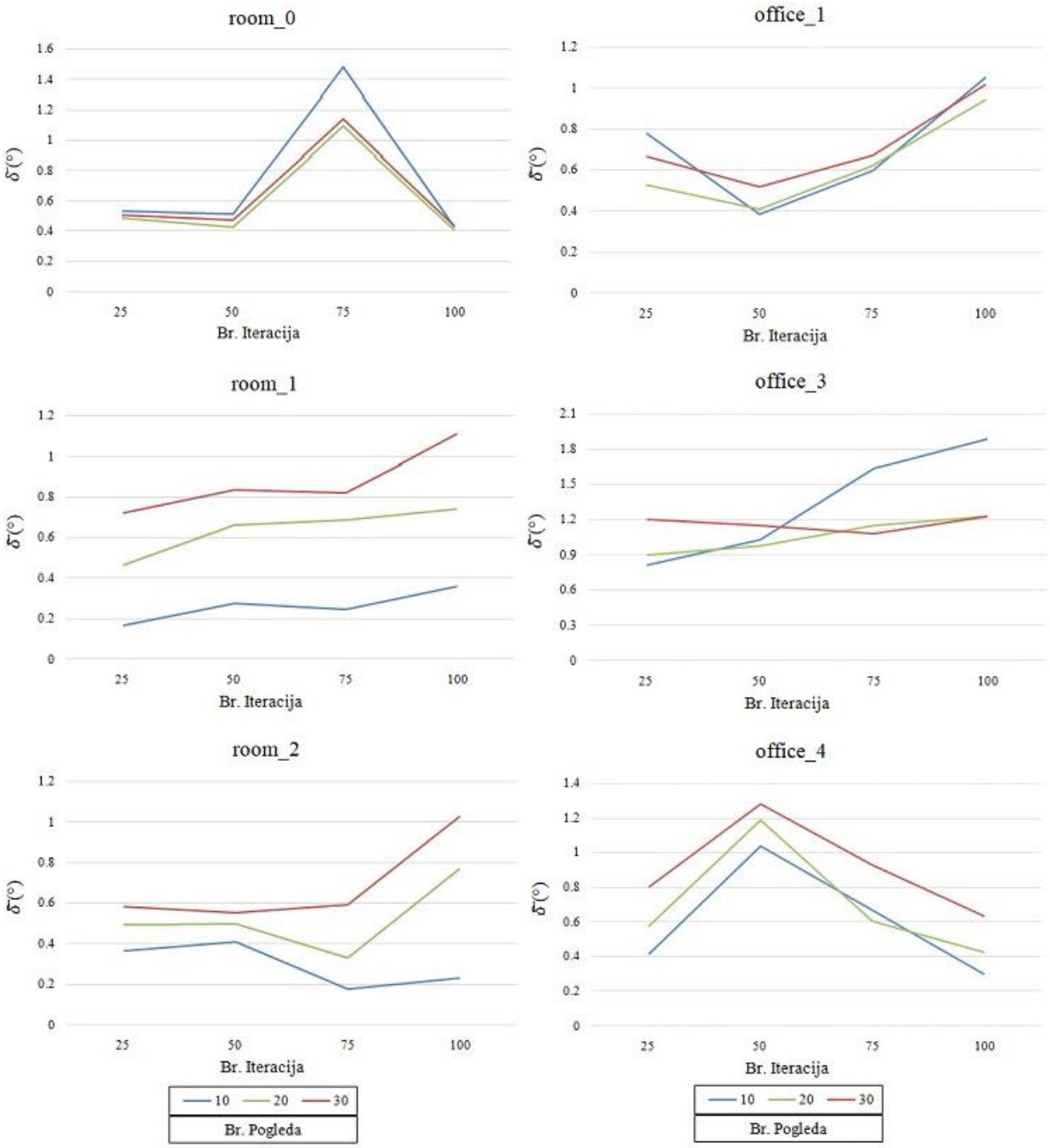
Slika 5.12 Prikaz pogrešnog poravnanja 3D modela pri mjernom šumu razine 0.005.

Kako bi se kvantitativno ocijenila točnost razmatranog algoritma izračunat je kut pomaka između procijenjenog i stvarnog pogleda, u ovom radu označen s δ. Prikaz pogreške proveden je snimanjem šest različitih soba iz Replica baze (*room_0*, *room_1*, *room_2*, *office_1*, *office_3*, *office_4*) u tri različita broja pogleda (10, 20, 30) kroz četiri različite iteracije (25, 50, 75, 100). Rezultati srednje i maksimalne pogreške po broju pogleda za određenu iteraciju su prikazani u tablici 5.1. Iz navedene tablice moguće je vidjeti kako je maksimalna srednja pogreška 1.8803°, dok je minimalna srednja pogreška 0.1888°, a maksimalna pogreška koja se pojavljuje iznosi 3,5170°. Prosječna srednja vrijednost pogrešaka za sve sobe po pogledima iznosi 0.6574° za 10 pogleda, 0.6916° za 20 pogleda i 0.8325° za 30 pogleda, odnosno ukupna prosječna srednja vrijednost pogreške rotacije iznosi 0.7271°.

Tablica 5.1 Prosječna i maksimalna pogreška rotacije za šest 3D modela prostorija iz Replica baze za tri različita broja pogleda u četiri iteracije.

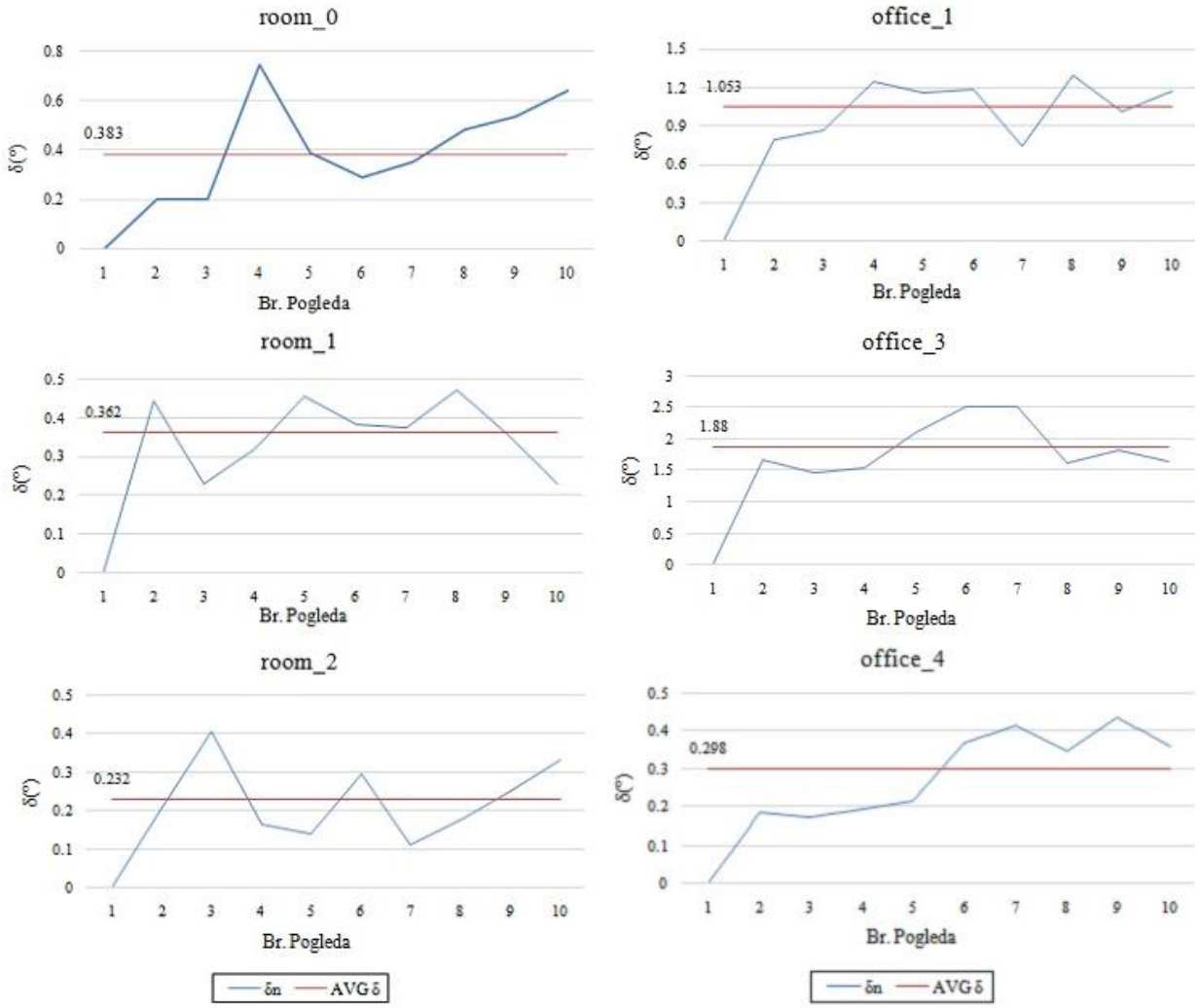
		Broj pogleda					
		10		20		30	
		Pogreška rotacije (°)					
Naziv sobe	***Iteracija***	AVG	MAX	AVG	MAX	AVG	MAX
room_0	25	0,5313	0,7245	0,4843	0,7245	0,5090	0,7706
	50	0,5134	0,9269	0,4266	0,9269	0,4727	1,4610
	75	1,4839	3,5170	1,0961	3,5170	1,1424	3,5170
	100	0,4260	0,7458	0,4067	0,7458	0,4335	0,8009
room_1	25	0,1688	0,2979	0,4657	1,1834	0,7218	1,4024
	50	0,2790	0,5213	0,6638	1,1758	0,8344	1,6985
	75	0,2500	0,4280	0,6879	1,3434	0,8196	1,3434
	100	0,3623	0,4715	0,7425	2,2175	1,1089	2,2175
room_2	25	0,3641	0,6546	0,4959	0,9612	0,5821	1,6717
	50	0,4091	0,4972	0,4971	0,7834	0,5543	1,0494
	75	0,1754	0,3071	0,3305	1,0910	0,5925	3,2987
	100	0,2324	0,4069	0,7712	1,9366	1,0295	1,9684
office_1	25	0,7823	1,3368	0,5282	1,3368	0,6676	1,4288
	50	0,3833	0,9233	0,4075	0,9756	0,5198	1,1709
	75	0,5961	0,9740	0,6220	0,9740	0,6739	0,9740
	100	1,0525	1,2970	0,9410	1,6778	1,0195	1,6778
office_3	25	0,8075	1,6333	0,8959	1,7728	1,1997	2,5351
	50	1,0242	1,4189	0,9743	1,4551	1,1486	1,7819
	75	1,6333	1,9193	1,1453	1,9193	1,0823	1,9193
	100	1,8803	2,5182	1,2242	2,5182	1,2306	2,5182
office_4	25	0,4164	0,5995	0,5732	1,1084	0,8000	2,1442
	50	1,0375	1,6630	1,1873	1,6630	1,2786	1,6630
	75	0,6695	1,1297	0,6065	1,1297	0,9267	3,1009
	100	0,2984	0,4364	0,4246	0,8270	0,6314	1,5286

Detaljniji prikaz podataka, iz gore navedene tablice 5.1, vidljiv je na slici 5.13, gdje su prikazani grafovi svih 3D prostorija uzetih iz Replica baze. Na grafovima su prikazane srednje pogreške rotacije kroz iteracije (25, 50, 75 i 100). Plava linija označava trend srednje pogreške rotacije za 10 pogleda, zelena za 20 pogleda, a crvena za 30 pogleda. Iz navedenih grafova je vidljivo kako se s povećanjem broja iteracija rotacijska pogreška ne smanjuje monotono. U nekim grafovima (*room_2*, *office_1* i *office_3*) pogreške rotacije su najznačajnije za 100 iteracija, dok je u drugim grafovima (*room_0* i *office_4*) ta pogreška najmanja. Također, nije vidljiva ni povezanost pogreške rotacije s brojem pogleda. Na nekim grafovima (*room_0*, *office_1* i *office_3*) srednja rotacijska pogreška kroz većinu iteracija je najveća za liniju koja predstavlja 30 pogleda, dok je na drugim grafovima (*room_1*, *room_2* i *office_4*) srednja rotacijska pogreška najveća za liniju koja predstavlja 10 pogleda.



Slika 5.13 Srednja pogreška rotacije po iteracijama.

Nadalje, na slici 5.14 prikazane su pogreške rotacije za prvih deset pogleda nakon 100-te iteracije za svaki 3D model prostorije. Plava linija prikazuje pogrešku rotacije za svaki pogled, dok crvena linija prikazuje srednju pogrešku rotacije za tih deset pogleda. Na grafovima je vidljivo kako pogreška rotacije nema povezanost koju je moguće pripisati broju pogleda.



Slika 5.14 Pogreška rotacije za prvih 10 pogleda nakon 100-te iteracije.

6. ZAKLJUČAK

Algoritam za stvaranje 3D modela prostorije fuzijom 3D oblaka točaka, realiziran u okviru RVL biblioteke, stvara točne 3D modele prostorija korištenjem 3D modela prostorija iz Replica baze. Zadatak ovoga rada bio je u već postojećem algoritmu dodati šum koji se pojavljuje u stvarnim uvjetima, sa stvarnim kamerama, te testirati ispravnost algoritma u takvim uvjetima. Veće pogreške u rekonstruiranim 3D modelima prostorija nastaju tek kada mjerni šum prelazi razinu od 0.005. Također, prikazano je da pogreške rotacije nisu povezane s brojem korištenih pogleda niti monotono opadaju s brojem iteracija koje se koriste za dobivanje 3D modela prostorije. Glavni nedostatak ovog algoritma je statičnost kamere, čime se stvaraju praznine u modelima iza objekata.

6.1. Budući rad

U budućem radu moguće je unaprijediti trenutne nedostatke algoritma kao što su zaklanjanje objekata. Taj problem bi se mogao riješiti pomicanjem kamere te ponovnim pokretanjem algoritma, čime bi se objekti koji su u prvom položaju kamere bili zaklonjeni u ovom, drugom položaju, bili vidljivi. Moguće je također u potpunosti osposobiti algoritam da bude primjenjiv u stvarnim uvjetima te da rekonstrukcija 3D modela prostora bude odrađena stvarnom RGB-D kamerom.

LITERATURA

[1] 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, str. 267-278, Pro. 2015.

[2] A. Ortis, F. Rundo, G. Di Giore, S. Battiato, „Adaptive Compression of Stereoscopic Images“, 17th International Conference on Image Analysis and Processing (ICIAP), Lecture in Computer Science, Napoli, Italija, br. 8156, str. 391-399, Ruj. 2013.

[3] C. Li, C. Tao, G. Liu, „3D Visual SLAM Based on Multiple Iterative Closest Point“, Mathematical Problems in Engineering, br. 2015, str. 1-11, Lip. 2015.

[4] C. Jing, J. Potgieter, F. Noble, R. Wang, „A comparison and analysis of RGB-D cameras' depth performance for robotics application“, 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP), Auckland, Novi Zeland, str. 1-6, Stu. 2017.

[5] J. Aloimonos, I. Weiss, A. Bandyopadhyay, „Active Vision“, International Journal of Computer Vision, br. 1, sv. 4, str. 333-356, Sij. 1988.

[6] M. Al Haj, C. Fernández, Z. Xiong, I. Huerta, J. Gonzàlez, X. Roca, „Beyond the Static Camera: Issues and Trends in Active Vision“, Visual Analysis of Humans, T. Moeslund, A. Hilton, V. Krüger, L. Sigal Springer, London, 2011., str. 11-30.

[7] M. A. Fischler, R. C. Bolles, „Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography“, Communications of the ACM, br. 24, sv. 6, str. 381–395, Lip. 1981.

[8] M. Vranješ, R. Grbić, „Nadzirano učenje – 2.dio“, predavanje Strojno učenje u sustavima autonomnih i umreženih vozila, Fakultet elektrotehnike, računarstva i informacijskih tehnologija (FERIT), Osijek, Pro. 2019. https://loomen.carnet.hr/pluginfile.php/1600842/mod_resource/content/8/02b%20Nadzirano%20u%C4%8Denje%20-%20Klasifikacija%20i%20regresija.pdf, (13.07.2020.)

[9] R. Marani, V. Renò, M. Nitti, T. D'Orazio, E. Stella, „A Modified Iterative Closest Point Algorithm for 3D Point Cloud Registration“, Computer-Aided Civil and Infrastructure Engineering, br. 31. sv. 7, str. 515-534, Srp. 2016.

[10] S. Rusinkiewicz, M. Levoy, „Efficient variants of the ICP algorithm“, Proceedings Third International Conference on 3-D Digital Imaging and Modeling, Quebec City, Quebec, Kanada, str. 145-152, Lip. 2001.

[11] M. Hržica, R. Cupec, I. Petrović, „Active vision for 3D indoor scene reconstruction using a 3D camera on pantilt mechanism“, 2020.

[12] H. Yang, J. Shi, L. Carlone, „TEASER: Fast and Certifiable Point Cloud Registration“, Sij. 2020.

[13] Y. Chen, G. Medioni, „Object modeling by registration of multiple range images“, IEEE International Conference on Robotics and Automation (ICRA), Sacramento, CA, SAD, br. 3., str. 2724-2729, Tra. 1991.

[14] R. Cupec, E. K. Nyarko, D. Filko, „Robot Vision Library – Instruction for Using Poit Cloud Segmentation Tool“, Grupa za robotiku i 3D računalni vid, Sveučilište J. J. Strossmayera Osijek, Fakultet elektrotehnike, računarstva i informacijskih tehnologija (FERIT)

[15] OpenCV: About, dostupno na: https://opencv.org/about/, (14.07.2020.)

[16] OpenNI 2, dostupno na: https://github.com/occipital/OpenNI2, (14.07.2020.)

[17] VTK, About: Overview, dostupno na: https://vtk.org/about/#overview, (14.07.2020.)

[18] Point Cloud Library | The Point Cloud Library (PCL), dostupno na: https://pointclouds.org/, (14.07.2020.)

[19] Y. Zhong, Y. Dai, H. Li, „3D Geometry-Aware Semantic Labeling of Outdoor Street Scenes“, 24th International Conference on Pattern Recognition (ICPR), Peking, Kina, 2018, pp. 2343-2349.

[20] J. Straub et al., „The Replica Dataset: A digital replica of indoor spaces“, pp. 1-10, Lip 2019.

SAŽETAK

U radu je objašnjen algoritam za stvaranje 3D modela prostorije fuzijom 3D oblaka točaka. Korištenjem navedenog algoritma, koji je implementiran unutar RVL (Robot Vision Library) biblioteke, implementirana je simulacija mjernog šuma koji se javlja u stvarnim mjerenjima s RGB-D kamerama. Algoritam radi na principu aktivne vizije koja se temelji na dva kriterija: što veći postotak još neistražene scene mora biti sadržan u sljedećem pogledu i dio scene koji je potpuno obuhvaćen sljedećim i jednim od prethodnih pogleda sadrži dovoljno informacije za uspješno poravnanje. Za izračunavanje korespondencije točaka u 3D oblaku točaka koristi se ICP algoritam koji ima tri uvjeta za ispravnu korespondenciju: euklidska udaljenost između dviju točaka mora biti manja od one uzrokovane rotacijom kamere za 3°, razlika normala tih točaka mora biti manja od 10° i razlika udaljenosti od kamere obje točke mora biti manja od 0,03m. Algoritam je testiran na šest različitih 3D modela prostorija preuzetih iz Replica baze. Dobiveni modeli uspoređeni su s originalnim. Maksimalna i srednja pogreška rotacije su prikazane za različite poglede kroz različite iteracije 3D modela prostorija iz Replica baze te su prikazani odgovarajući grafovi za njih.

Ključne riječi: 3D model prostorije, RGB-D kamera, 3D oblak točaka, ICP, RANSAC, TEASER, RVL, PCL

ABSTRACT

Title: **CREATING 3D ROOM MODEL BY FUSING 3D POINT CLOUDS CAPTURED WITH 3D CAMERA**

An algorithm for creating 3D room model by fusing 3D point clouds was explained in this paper. This algorithm, which is implemented as a part of RVL (Robot Vision Library), was used to simulate noise which occurs in real-time measurements with RGB-D cameras. The algorithm works on the principle of active vision which has two criteria: next view has to incorporate the largest possible percentage of yet unexplored scene and part of the scene that is completely incorporated in both the next and one of the previous views contains enough information for succesful registration. An ICP algorithm, which has three conditions for succesful registration, was used to calculate point correspondence in 3D point clouds. Those conditions are: Euclidean distance between two points has to be less than the point shift caused by camera rotation of 3°, difference in point normals of these points has to be less than 10° and difference in distances from the camera of these points has to be less than 0,03m. The algorithm was tested on six different 3D room models from Replica Dataset. Point cloud registration was tested and the created models were compared with the original ones. Maximal and average rotation errors were calculated for different views in different iterations of 3D room models from the Replica Dataset. These results were presented in the form of graphs for each 3D room model.

Key words: 3D room model, RGB-D camera, 3D point cloud, ICP, RANSAC, TEASER, RVL, PCL

ŽIVOTOPIS

Branimir Tomašić rođen je 20. Siječnja 1997. godine u Našicama. Osnovnu školu je upisao i završio u mjestu Koška, u Osnovnoj školi „Ivane Brlić Mažuranić“. 2011. godine upisuje se u Prirodoslovno – matematičku gimnaziju u Osijeku. Nakon završetka srednje škole, 2015. godine upisuje Elektrotehnički fakultet u Osijeku, preddiplomski sveučilišni studij Računarstva. te ga 2018. završava u roku. Potom upisuje diplomski studij Automobilsko računarstvo i komunikacije na fakultetu novog naziva, Fakultet elektrotehnike, računarstva i informacijskih tehnologija. Trenutno je na završnoj godini diplomskog studija.

_____

Branimir Tomašić

DODATNI PRILOZI

Programski kod priložen je na CD-u uz rad.