Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Blažka Blatnik

Gradnja 3D modela s pomočjo oblaka točk in barvnih slik

diplomsko delo

univerzitetni študijski program
prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Danijel Skočaj

Ljubljana, 2018

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Za rekonstrukcijo 3D oblike predmetov se uporabljajo različni senzorji in pristopi. Prva skupina pristopov uporablja kot vir podatkov globinske kamere in 3D model rekonstruira iz oblakov 3D točk. Druga množica pristopov gradi 3D modele na osnovi množice barvnih slik. Oba pristopa imata prednosti in slabosti. Predvsem imata oba pristopa težave pri registraciji delnih 3D modelov določenih tipov površin, pri čemer ima prvi pristop težave z geometrijsko nerazgibanimi površinami, medtem ko drugi pristop zahteva bogato teksturo na predmetih. V diplomski nalogi uporabite za gradnjo 3D modelov oba pristopa. Razvijte postopek, ki bo zgradil 3D model iz množice 3D točk; delne 3D modele pridobljene z barvno-globinsko kamero Kinect naj postopek registrira z algoritmom ICP. Ta algoritem zahteva za uspešno delovanje in konvergenco h globalnemu minimumu dovolj dobro začetno poravnavo, ki jo zagotovite z upoštevanjem barvne informacije z algoritmom SfM.

Diplomsko delo je nastajalo ob neizmerni podpori najbližjih, ki so me spodbujali, motivirali in verjeli vame, ko sem to najbolj potrebovala.

Moji dragi Pavli

Kazalo

Povzetek

Naslov: Gradnja 3D modela s pomočjo oblaka točk in barvnih slik

Avtor: Blažka Blatnik

Na področju umetnega zaznavanja, računalniškega vida in robotike se pogosto uporabljajo različni viri podatkov o okolju. Edinstven in informativen vir je tudi oblak točk, ki predstavlja in opisuje 3D prostor okoli nas. Področja uporabe pa segajo precej preko robotike, saj je mogoče oblak točk zajeti in združiti v reprezentativen model objekta iz realnega prostora. S kasnejšo obdelavo lahko dosežemo, da je pridobljeni model dovolj natančno reprezentativen. Primerno obdelan objekt je zato mogoče tudi natisniti.

Cilj diplomske naloge je bil izboljšati algoritem ICP z uporabo podatkov z barvne slike za začetno oceno pozicije. Ocenjena pozicija je bila rezultat prilagojenega algoritma SfM.

Ključne besede: 3D, oblak točk, globinska kamera, sledenje, rekonstrukcija.

Abstract

Title: Building a 3D model with point cloud and color images

Author: Blažka Blatnik

In the field of artificial recognition, computer vision and robotics there are usually different means of getting representation of surrounding place. One of a kind and unique source of data is a point cloud representation which gives us 3D description of the surroundings. Use cases of point cloud far surpass robotics as it can be used to reconstruct a 3D model from the real world. With post processing we can obtain accurate models that can be later on printed with a 3D printer.

The goal of this thesis has been to improve ICP algorithm with the use of 2D data from colored images, to approximate camera positions. Approximate camera position is the end result of SfM algorithm.

Keywords: 3D, point cloud, depth camera, tracking, reconstruction.

Poglavje 1

Uvod

V robotiki se uporabljajo različni viri podatkov o okolju. Veliko raznolikih podatkov sistemu omogoča bolj informirano odločanje za navigacijo v prostoru in izbiro dejanj. Tip podatkov, ki omogoča opis prostora v treh dimenzijah, se imenuje oblak točk (prikazan na Sliki 1.1). Z njim lahko predstavimo prostor v treh dimenzijah, saj ima vsaka točka poleg dvodimenzionalne pozicije na sliki tudi dodatno dimenzijo: globino. Zajem tovrstnih podatkov nam omogočajo naprave, ki imajo globinsko zaznavo, na primer Microsoft Kinect, ki je prikazan na Sliki 1.2. Ena glavnih prednosti Kinecta je njegova cenovna dostopnost.

Predstavitve prostora z oblakom 3D točk lahko iz več različnih zornih kotov združimo in tako dobimo rekonstruirani model predmeta ali prostora. S primerno obdelavo lahko v naslednjem koraku tak model tudi natisnemo s 3D tiskalnikom. Tako lahko Kinect uporabljamo kot 3D skener.

1.1 Motivacija

Predstavitev prostora ali objekta s 3D modelom je uporabna ne samo v robotiki in 3D modeliranju, pač pa tudi pri industrijskem oblikovanju, računalniškem vidu, razvoju računalniških iger in topografskih meritvah [26]. Primer 3D modela je prikazan na Sliki 1.3. Tako lahko na primer zajamemo željeni

Slika 1.1: Primer oblaka točk.

Slika 1.2: Microsoft Kinect [5].

objekt, ga rekonstruiramo v 3D model in poljubno obdelujemo, dopolnjujemo in analiziramo.

3D rekonstrukcija mora, kar se da natančno, posnemati dejanske dimenzije opazovanega predmeta ali prostora, zato je potrebno zagotoviti, da algoritmi uporabljeni za rekonstrukcijo delujejo natančno in robustno, ter hkrati dovolj hitro.

Obstoječe implementacije v obliki programske opreme oz. celostno implementiranih cevovodov za rekonstrukcijo 3D modela z uporabo Kinecta v glavnem ne zajemajo vseh zgornjih zahtev. Če so hitre, niso natančne in robustne, če so natančne in robustne, pa običajno niso dovolj hitre. Poleg tega za svoje delovanje običajno uporabljajo samo globinsko sliko kamere, kar na geometrijsko nerazgibanih površinah (na primer na ravni steni) večinoma povzroči, da sistem izgubi trenutno lego in je potrebno celotni proces skeniranja ponoviti.

Glavna motivacija za izdelavo diplomske naloge je nadgraditi postopek ICP, za poravnavo oblakov točk, tako da bo upoštevana tudi barvna infor-

macija. Registracija je tako bolj odporna na monotono geometrijo, algoritem ICP pa z dodatnimi informacijami lažje in hitreje konvergira v globalni minimum.

Slika 1.3: Primer rekonstruiranega 3D modela [11].

1.2 Obstoječe implementacije

Za rekonstrukcijo 3D modela iz oblaka točk že obstajajo celostne implementacije procesa. Najbolj znani primeri so:

- **KinectFusion** in **KinFu**. KinectFusion [30] je produkt podjetja Microsoft in je namenjen za obdelavo oblaka točk zajetega z uporabo Kinecta. Algoritem sledi zaznani površini in med vsakima zajetima slikama izračuna transformacijsko matriko (t.j. rotacijo in translacijo) ter tako določi gibanje, kar mu omogoča sledenje in prepoznavanje površine. Slednje izvaja v štirih glavnih korakih [30]:

  1. *Pridobivanje površine:* v tem koraku se izračunata pozicija 3D točk v oblaku in njihove normale.
  2. *Poravnava:* v tem koraku sta ocenjeni pozicija in orientacija senzorja.
  3. *Globinska predstavitev:* v tem koraku se podatki o površini dopolnijo z volumnom.
  4. *Rekonstrukcija površine:* v zadnjem koraku se rekonstruira površina, rezultat je tridimenzionalna predstavitev scene.

Odprtokodna različica algoritma KinectFusion je algoritem KinFu [30], ki je del knjižnice PCL (Point Cloud Library [6]). Algoritem rekonstrukcijo rešuje na zelo podoben način kot KinectFusion in sicer v sledečih korakih:

1. *Filtriranje:* v tem koraku se zmanjša gostota oblaka točk, zato je kasnejša obdelava časovno in prostorsko manj zahtevna.
2. *Poravnava:* v tem koraku se oblaki točk poravnajo tako, da tvorijo željeni objekt.
3. *Rekonstrukcija:* v zadnjem koraku s pomočjo triangulacije tvorimo površino, ki predstavlja končni tridimenzionalni objekt.

Glavni prednosti algoritma KinectFusion sta robustnost na dinamične objekte v sceni in delovanje tudi v popolni temi. Algoritem je zelo občutljiv na hitre premike ter monotono geometrijo opazovane scene [33]. Slabost algoritma KinFu je pogojenost strojne in programske opreme ter pomanjkanje optimizacije, prednost pa odprtost za vse OpenNI senzorje (Natural Interaction) [30].

- **ElasticFusion** je algoritem za lokalizacijo (določanje rotacije in translacije za posamezno zajeto sliko), ki kot dodatno informacijo uporablja podatke o svetlobi. Pri registraciji si pomaga tudi z minimizacijo razlike v intenziteti med posameznimi slikovnimi točkami v barvni sliki.

  Velik poudarek daje reševanju problema *zaprtja zanke*, kar pomeni, da se zaveda, kdaj smo se pri skeniranju vrnili na že videno mesto. To rešuje z deljenjem že registriranih predelov na aktivne in neaktivne. Tako posamezen predel po določenem času neobiskanosti preide iz aktivnega v neaktivno stanje. Algoritem registracijo posameznih pogledov izvaja sproti in pri tem preverja, če novo zajeti oblak točk pripada aktivni ali neaktivni površini in s tem določa zaprtje zanke, kar izboljša poravnavo oblakov točk zajetih iz različnih smeri.

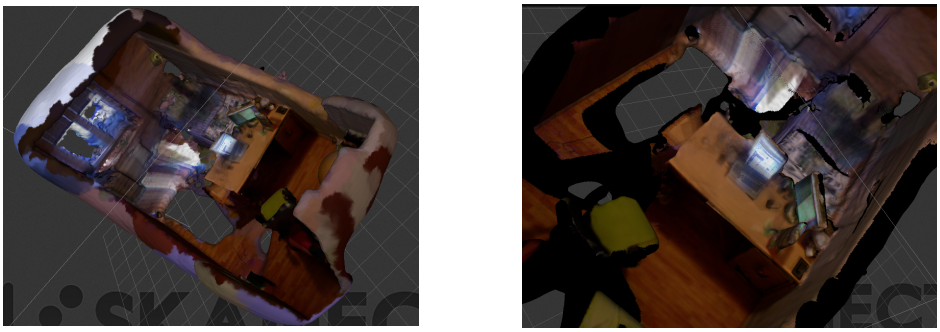
  ElasticFusion rešuje tako lokalne zanke (pomeni, da v krajšem času

večkrat obiščemo isto površino - npr. skeniranje z nihajočimi gibi levo-desno ali gor-dol) kot tudi globalne zanke (ko po dolgem času znova obiščemo že opazovano površino - npr. skeniranje sobe v krogu) [34].

- Algoritem **Kintinuous** nadgrajuje algoritem KinectFusion. Za registracijo oblakov točk primarno uporablja algoritem ICP (Iterative closest point [17]), ki iterativno zmanjšuje razdaljo med dvema oblakoma točk. V primeru, da algoritem ICP ne uspe poravnati dveh oblakov točk, se zanaša na algoritem FOVIS (Fast Odometry from Vison [33]), kar pomeni, da za določanje lege uporabi podatke iz barvne slike. FOVIS sistem se uporablja sekundarno, ker proizvede redkejšo rekonstrukcijo. Sekundaren način za lokalizacijo tako izboljšuje eno glavnih pomankljivosti algoritma KinectFusion in sicer nerobustnost na monotono geometrijo opazovane scene [33].

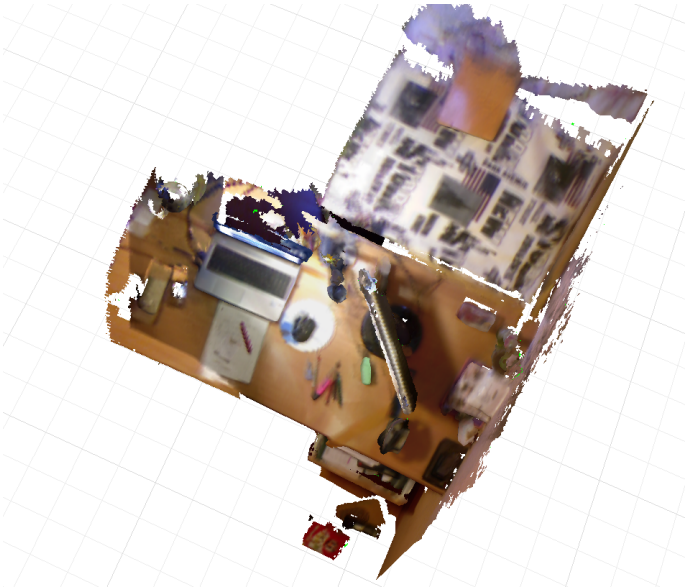
Prav tako obstajajo rešitve, ki v obliki brezplačne programske opreme ponujajo zajem in registracijo oblaka točk ter rekonstrukcijo 3D modela:

- **Skanect** [12] omogoča zajem oblaka točk z napravama Kinect ali Asus Xtion. Njegova glavna pomankjivost je, da skeniranje ne uspe, če opazovana površina ni dovolj geometrijskih razgibana. Zato je priporočeno skenirani sceni dodati več različnih geomretijskih teles. Končen rezultat uspešnega skeniranja je zadovoljivo natančno rekonstruiran 3D model s teksturo. Primera rekonstrukcije sobe sta prikazana na Sliki 1.4.



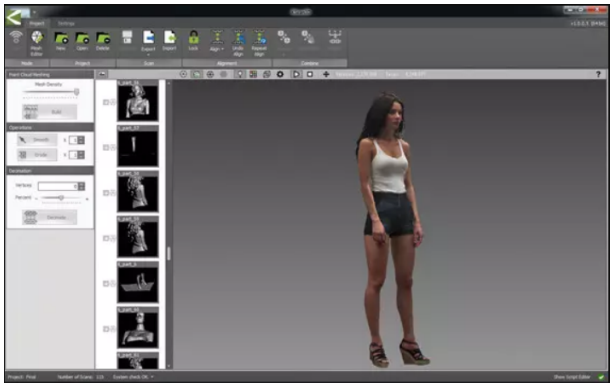
Slika 1.4: Primer rekonstrukcije sobe z uporabo programa Skanect.

- **ReconstructMe** [10] za zajem oblaka točk podpira veliko različnih globinskih senzorjev. Deluje hitreje kot Skanect in ne izgubi sledenja tako hitro, vendar skeniranje sobe v večini primerov še vedno spodleti, saj gre za sceno z veliko geometrijsko nerazgibanimi površinami (stene), kjer program skoraj praviloma izgubi pozicijo. Zaradi tega je težko rekonstruirati celotno sobo oz. skleniti zanko. Na Sliki 1.5 je primer rekonstrukcije, ki je izgubila sled trenutne lege senzorja, ko smo začeli skenirati steno.



Slika 1.5: Primer rekonstrukcije sobe z uporabo programa ReconstructMe.

- **KScan3D** [3] program je prvotno namenjen skeniranju objektov. Za vsak zajem slike skrbi uporabnik, zato posledično rekonstrukcija ni v realnem času. Model se rekonstruira in obdela po zajetih slikah in končni rezultat je lahko precej natančen, večinoma odvisen od uporabnika. Slika 1.6) prikazuje uporabo programa za rekonstrukcijo osebe.



Slika 1.6: Primer rekonstrukcije osebe z uporabo programa KScan3D [8].

1.3 Oris predlagane rešitve

Večina zgoraj predstavljenih algoritmov (in programov) se primarno zanaša na informacije dostopne iz globinske slike, ki so pri monotoni geometriji precej skope, lahko tudi nezadostne. Zato je v veliko primerih skeniranje sobe z veliko praznimi stenami neuspešno. Naša rešitev bi za registracijo oblakov točk uporabila podoben pristop kot algoritem Kintinuous, le da bi se zanašala najprej na vizualne podatke in nato še na algoritem ICP. S tem bi izboljšali eno glavnih pomankljivosti algoritma ICP in sicer potrebo po precej natančni začetni poravnavi. Končni proces predlagane rešitve je predstavljen na Sliki 1.7.

Najprej z uporabo globinskega senzorja zajamemo dva oblaka točk z različnih zornih kotov, ki se delno prekrivata. Oba oblaka točk sta tipa XYZRGBA, kar pomeni, da poleg pozicije vsake točke poznamo tudi njeno teksturo (RGB barvo in alpha kanal). S pomočjo teh podatkov lahko oblak točk pretvorimo v 2D barvno sliko. S tem se izognemo težavi sinhronizacije zajetega oblaka in zajete slike, ki bi jih bilo drugače potrebno pridobivati ločeno.

Pridobljeni barvni sliki uporabimo za oceno rotacijske matrike R in translacijskega vektorja t, ki jo pridobimo s pomočjo postopkov algoritma SfM (structure from motion [31]) in 2D-3D korespondenc. Iz njiju sestavimo pro-

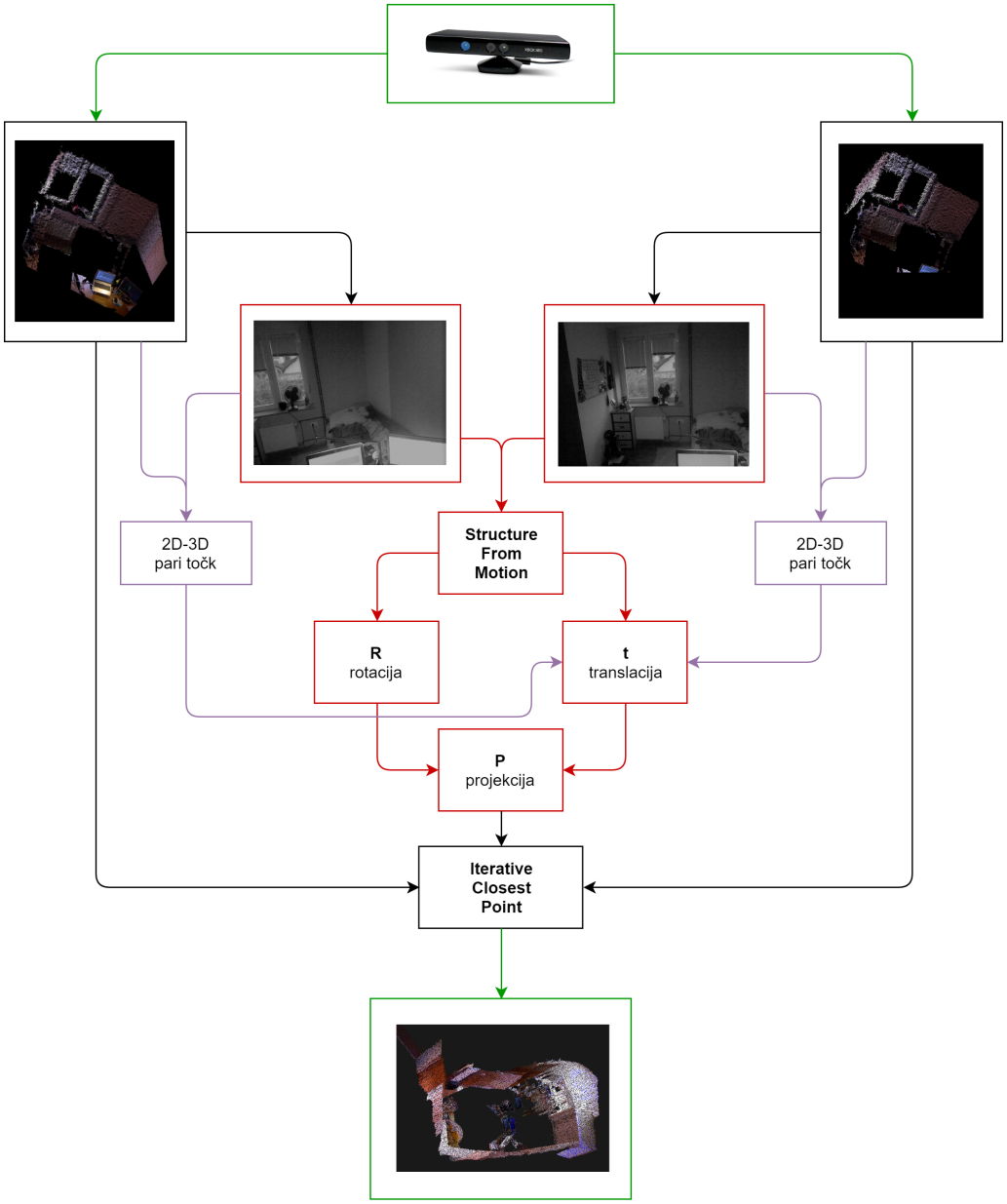
jekcijsko matriko P, ki nam preslika drugi zorni kot v koordinatni sistem prvega. Projekcijsko matriko uporabimo za začetno oceno transformacije pri algoritmu ICP.

S podano začetno oceno algoritmu znatno skrajšamo čas potreben za konvergiranje in hrati zmanjšamo možnost, da bo algoritem našel lokalni minimum, ki hkrati ni globalni.

1.4 Struktura diplomske naloge

Diplomska naloga je sestavljena iz petih poglavij, ki smiselno sledijo razvojnemu procesu končne rešitve.

- **Prvo poglavje** je uvod, kjer predstavimo motivacijo za idelavo naloge ter pregledamo obstoječe implementacije.
- **Drugo poglavje** najprej predstavi globinske kamere in pomembne podrobnosti o njihovem delovanju. Nato predstavi delovanje algoritma SfM, kot korak za delno poravnavo oblakov točk, in utemelji njegovo rabo. Sledi predstavitev algoritma ICP, njegove glavne lastnosti ter pomankljivosti. V zadnjem delu predstavimo potek integracije registriranih oblakov točk.
- **Tretje poglavje** je predstavitev končne implementacije. Najprej na kratko predstavi uporabljeno programsko opremo, nato pa še implementacijo končnega sistema.
- **Četrto poglavje** je namenjeno predstavitvi eksperimentalnih rezultatov delovanja končnega sistema.
- **Peto poglavje** je zaključek, ki predstavi sklepne ugotovitve ter možnosti izboljšave.



Slika 1.7: Cevovod končne implementacije.

Poglavje 2

Gradnja 3D modela

Implementiran sistem zajame oblak 3D točk s pomočjo globinskega senzorja, nato pa oblake točk iz več različnih pogledov preslika v koordinatni sistem objekta in jih združii. V tem poglavju predstavljamo opisane korake in metode za njihovo implementacijo.

2.1 Zajem podatkov

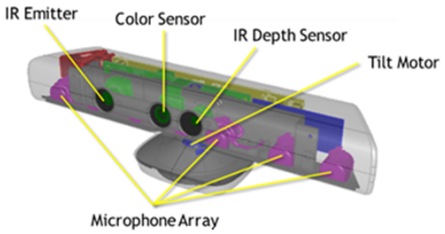
Globinski senzorji nam omogočajo predstavitev prostora z oblakom točk v treh dimenzijah, česar navadne kamere ne omogočajo. S tem naša zaznava pridobi novo dimenzijo - globino.

Delovanje globinskega senorja lahko v grobem delimo na dva dela. Na zajem dvodimenzionalne barvne slike in na zajem oblaka točk. Barvno sliko lahko zajamemo z navadno kamero, za oblak točk pa potrebujemo dva elementa. Prvi je IR (infra-rdeč) projektor, ki projicira vzorec točk v prostor, drugi pa je IR senzor, ki zaznava infrardečo svetlobo, v našem primeru gre za projicirane točke. Splošna globina se izračuna iz gostote točk in sicer velja, da so točke bližje izhodišča projekcije bolj goste in so z oddaljenostjo čedalje bolj redke. Vsako točko lahko s pomočjo triangulacije rekonstruiramo, saj poznamo njeno pozicijo v izvoru (IR projektor) in njeno projekcijo na sliki (IR senzor). Proeciranje točk je prikazano na Sliki 2.1. Ujemanje

točk med izvorom in projekcijo je možno izračunati, ker so točke razpršene v razpoznavnem in nesimetričnem vzorcu.

Slika 2.1: Proecirane IR točke v prostoru [9].

Primer nizkocenovne globinskege kamere je Microsoft Kinect, ki je bil sprva razvit za naravno interakcijo med človekom in igralno konzolo Xbox [36]. Ker je Kinect dovolj ugoden in hkrati dovolj natančen, je postal široko uporaben za različne raziskovalne namene na temo 3D zaznavanja. Kinect sestavljata dva senzorja (barvni in IR) ter projektor IR točk. Poleg treh (za globinsko kamero osnovnih) komponent ima Kinect tudi mikrofone in motor za samodejni nagib celotne naprave. Zgradba senzorja Kinect je prikazana na Sliki 2.2.



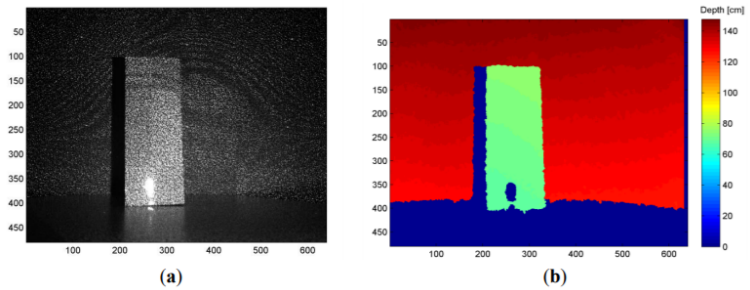
Slika 2.2: Zgradba kinect senzorja [13].

Tekstura zajeta z barvno kamero se globinski sliki doda programsko po tem, ko poznamo kalibracijo obeh kamer (globinske in IR). Tu se lahko pojavijo napake oz. manjša odstopanja in nekonsistentnosti, ki jih moramo pri delu z globinskimi kamerami upoštevati. Tako lahko točka na robu površine,

ki je bližje kameri, prevzame barvno lastnost površine, ki je neposredno za njo in obratno.

Senzor Kinect hkratno zajema globinsko ter barvno sliko s hitrostjo 30 slik na sekundo (ang. frame per second). Rezultat integracije barvne z globinsko sliko je barven oblak točk, ki vsebuje približno 300.000 točk [24].

Koncept delovanja senzorja Kinect je prikazan na Sliki 2.3. Laser v IR projektorju oddaja en sam žarek, ki se razbije na več delov, tako da projicira nesimetričen in razpoznaven (ter vedno enak) vzorec pik. Ta vzorec zajame IR senzor in ga primerja z v naprej pripravljenim in shranjenim referenčnim vzorcem, ki je bil zajet na znani razdalji. Med vzorcema se izračuna disparitet (ang. disparity), ki jo uporabimo za oceno oddaljenosti vsake točke.



Slika 2.3: Delovanje Kinect senzorja za določanje globine. a) Proekcija mreže IR točk. b) Končna globinska slika [24].

Poleg Kinect senzorja obstajajo tudi drugi cenovno ugodni globinski senzorji, na primer Asus Xtion, PrimeSense Carmine, Orbbec Astra, Intel Realsense SR300 in Persee sensor.

2.2 Približna poravnava oblakov točk

Kamere (globinske in navadne) imajo omejeno vidno polje ter posledično hkrati zajemajo samo del objekta ali scene, zato je potrebno za rekonstrukcijo večjega območja združiti več zaporednih posnetkov. Proces združevanja

več zaporednih oblakov točk, glede na prekrivajočo površino, imenujemo registracija. Rezultat registracije je združena celota, sestavljena iz več posnetkov z različnih zornih kotov. Za uspešno registracijo se morata obravnavani sliki vsaj delno prekrivati. Med njima mora obstajati zadostno število skupnih točk. Eden najbolj popularnih algoritmov za registracijo oblaka točk je algoritem ICP (Iterative closest point) [28]. Algoritem iterativno zmanjšuje razdaljo med dvema oblakoma točk, ter ju tako zbližuje, dokler ne konvergira. V kolikor konvergira v globalni minimum, sta oblaka točk pravilno poravnana - registrirana.

Glavna pomanjkjivost algritma ICP je, da za iskanje ključnih točk uporablja podatke pridobljene iz globinske slike. Te točke so lahko velikokrat pomanjkljive in celo zavajajoče, vsekakor pa ne uporabljajo vseh možnih informacij, ki jih lahko pridobimo z barvno 2D sliko. Algoritem ICP tako na ravni steni ne bo prepoznal nobene ključne točke, posledično ima mnogo implementacij za rekonstrukcijo največ težav prav na takšnih (geometrijsko) monotonih površinah. Prav tako bo nekatere ključne točke prepoznal na meji med predmetom in ozadjem, ki pa bodo pri različnih perspektivah na različnih mestih.

Pri obdelavi 2D slik teh pomanjkljivosti nimamo, saj si lahko pri prepoznavi ključnih točk pomagamo s prepoznavanjem vzorcev na stenah, predmetih ali njihovi okolici. V primeru, da opazovana površina kljub temu nima dovolj raznolikih in prepoznavnih vzorcev, si lahko pomagamo z dodajanjem vizualnih markerjev (preproga z vzorci ali namenske oznake). Primer namenske oznake je prikazan na Sliki 2.4.

S tem lahko izkoristimo dodatne informacije, ki jih dobimo z obdelavo 2D barvnih slik, in oblake točk iterativno grobo poravnamo pred algoritmom ICP. Tako zmanjšamo potrebno število korakov v katerih funkcija konvergira in hkrati zmanjšamo možnosti, da bi se algoritem ICP ustavil v lokalnem minimumu.

Slika 2.4: Primer oznake za dodano vizualno informacijo [7].

2.2.1 Algoritem SfM

SfM (structure from motion) je algoritem za rekonstrukcijo 3D modela iz njegovih projekcij na serijo več slik posnetih z različnih zornih kotov [31]. Glede na način izgradnje 3D modela se algoritem v grobem deli na tri vrste. Inkrementalni SfM inicializirajo slike, kot že ime samo pove, inkrementalno, globalni rešuje problem z vsemi slikami hkrati, hibridni pa uporablja mešanico obeh. Prvi je počasnješi in hitro kopiči napake pri rekonstrukciji, drugi je bolj učinkovit in natančen, a ima lahko težave z določanjem amplitude premika in izločanjem napačnih ujemanj med slikami [19]. Delovanje obeh metod se velikokrat vrednoti na velikem številu slik, ki med seboj niso zaporedno urejene (vsaka naslednja slika pokriva del trenutne), v našem primeru pa slike dodajamo inkrementalno in za vsako vemo, da se delno prekriva s predhodnjo. Zato uporabimo inkrementalno različico algoritma, pri kateri izpustimo vse korake, ki preverjajo, če se zaporedni sliki dejansko prekrivata.

Glavni potek algoritma SfM je predstavljen na shemi 2.5. V prvem koraku zajamemo sliko opazovane scene (v našem primeru jo pridobimo iz oblaka točk, ki ga zajema senzor Kinect). Na sliki poiščemo značilne točke in njihove opisnike s pomočjo katerih bomo lahko kjučne točke v nadaljni obdelavi ločili. Ko smo opisana koraka izvedli za vsaki dve sliki, med njima poiščemo pare točk, kar pomeni, da ocenimo kateri opisniki opisujejo projekcijo enake točke

iz prostora.

V primeru, da ge za prvi par slik, lahko samo približno določimo lego obeh kamer (ocenimo rotacijo R in translacijo t) in ju uporabimo za prvo triangulacijo točk. Prvi par služi za inicializacijo nadaljnega procesa [31].

V primeru, da gre za vsak naslednji par slik, najprej poiščemo točke, ki so bile v prejšnji fazi že triangulirane in tako iz 2D-3D parov izračunamo trenutno pozicijo kamere. Preostale točke, za katere še ne poznamo ocene 3D pozicije, trianguliramo.

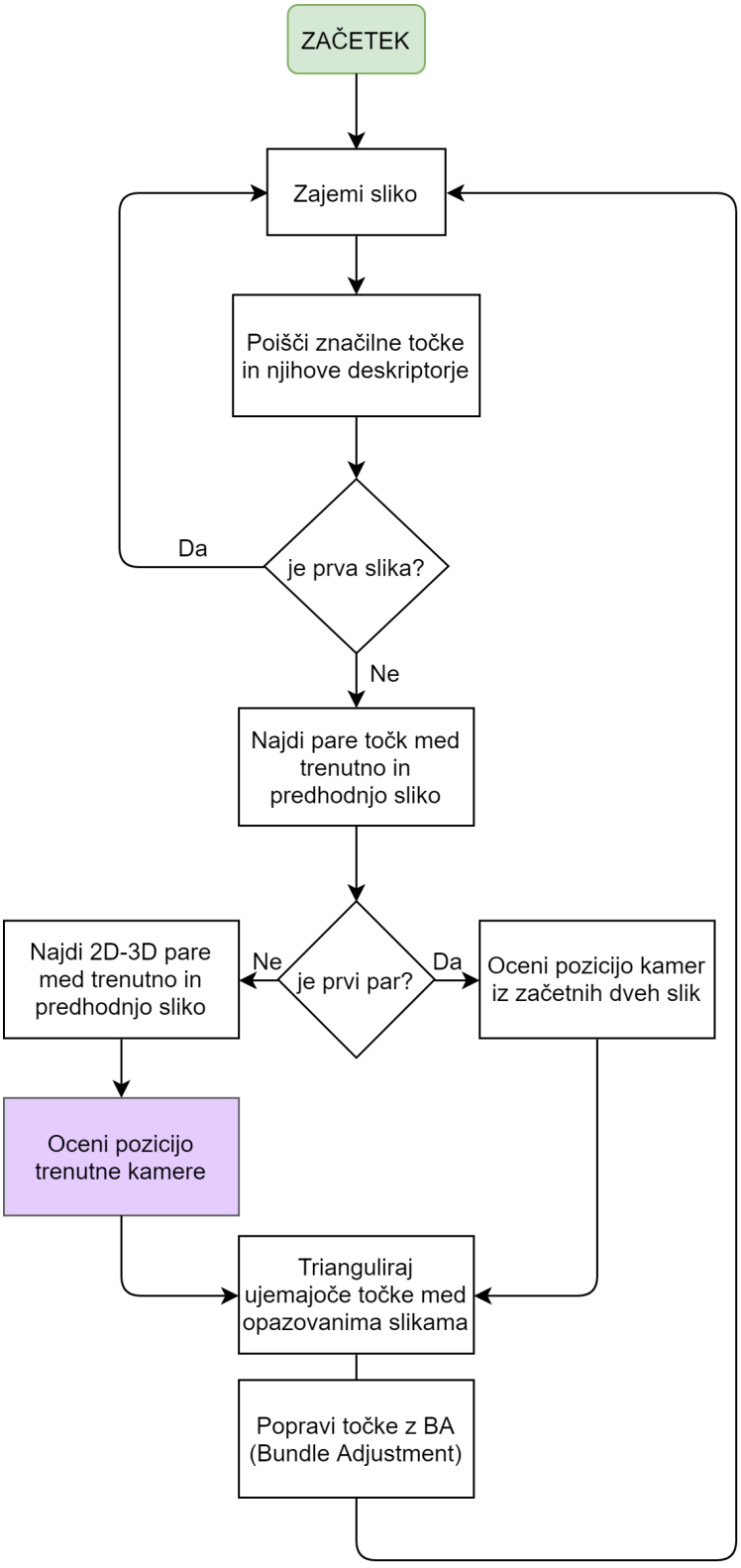
Po vsaki fazi triangulacije je pomembno popravljati napake projekcij, saj se drugače te kopičijo in lahko pripeljejo do popolnoma napačnega rezultata. To storimo s procesom imenovanim Bundle Adjustment, ki zmanjšuje reprojecijsko napako 3D točk nazaj v znane 2D točke. Sledi podrobnejši opis vsake izmed faz.

2.2.2 Iskanje značilnih točk

Vsako sliko lahko predstavimo z opisi njenih ključnih točk, s katerimi jo lahko potem primerjamo z naslednjo ali predhodnjo sliko, ter na podlagi tega določamo relacije med njimi. Kjučna točka na sliki je točka, ki jo lahko ponovno zaznamo pod spremenjeno orientacijo in pozicijo pogleda.

Za vsako sliko I določimo pare $F_i = \{(x_j, f_j)|j = 1...N_{F_i}\}$, kjer je $x_j \in R^2$ lokacija značilne točke, predstavljena z deskriptorjem f_j. Iskanje značilnih točk mora biti robustno na spremembe velikosti in geometrije zato, da lahko robustno določimo pare točk med dvema zaporednima slikama, ki točko gledata iz različnih zornih kotov [31].

Primer algoritma za določanje in deskripcijo kjučnih točk, invariantno na spremembo velikosti ali geometrije, je SIFT (scale-invariant feature transform [27]) ali njegova hitrejša različica SURF (Speeded Up Robust Features [15]). SURF v prvem koraku določi območja in merila (ang. scale), ki se jih lahko ponovno detektira z različnih zornih kotov. To doseže z iskanjem potencialnih ključnih točk, ki jih detektira s pomočjo algoritma za iskanje ključnih točk (npr. Hessian), iz katerih po določenih kriterijih kaskadno iz-



Slika 2.5: Shema glavnega poteka algoritma SfM

bira takšne, ki se ne spremenijo z večanjem ali manjšanjem opazovane slike - so stabilne. Za filtriranje takšnih točk uporablja Gausovo funkcijo, tako je velikostni prostor slike določen s funkcijo $L(x, z, \sigma)$, ki je rezultat konvolucije Gausove funkcije $G(x, y, \sigma)$ z opazovano sliko $I(x, y)$ [27]:

$$L(x, z, \sigma) = G(x, y, \sigma) * I(x, y), \tag{2.1}$$

kjer je $*$ konvolucija in

$$G(x, z, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2}. \tag{2.2}$$

Za učinkovito določanje pozicij stabilnih ključnih točk algoritem izračuna konvolucijo $D(x, y, \sigma)$ med sliko in razliko med dvema Gaussovima funkcijama (DoG - Difference of Gaussioan) različnih velikosti:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, z, k\sigma) - L(x, z, \sigma), \end{aligned} \tag{2.3}$$

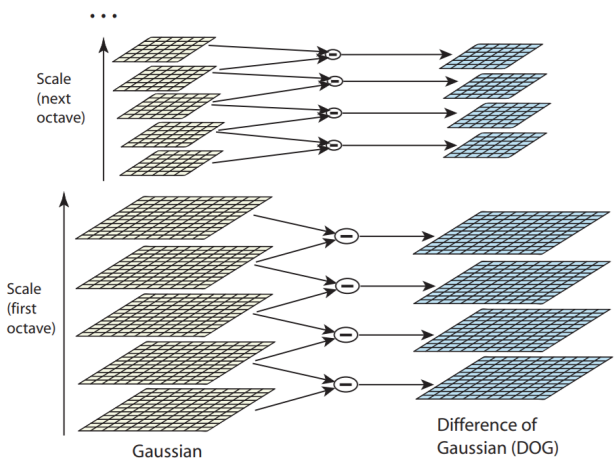
prikazano tudi shematsko na Sliki 2.6.

Algoritem SURF tako kot rezultat vrne značilne točke, katerih opis zajema velikost ter geometrijsko značilnost.

Ko algoritem najde točke, moramo določiti ustrezne pare. Za ujemanje parov značilnih točk uporabljamo metodo surove sile (ang. brute force). To pomeni, da moramo za vsako točko iz prve slike najti najbolj podobno točko iz druge slike, pri čemer primerjamo vse možne deskriptorje. Na koncu obdržimo samo simetrična ujemanja; veljavna so samo ujemanja, ko je bil A najboljši par za B in B najboljši par za A.

2.2.3 Ocena lege kamere

Oblak točk je za posamezen pogled podan v koordinatnem sistemu kamere. Za poravnavo dveh pogledov potrebujemo ocenjeno pozicijo obeh v 3D prostoru objekta. To lahko opišemo s središčno točko in rotacijo pogleda, kar nam predstavlja 6 prostostnih stopenj (6DOF). Navedeno nam opisujeta rotacijska matrika R in translacijski vektor t.



Slika 2.6: Pridobivanje več različnih velikosti slike z računanjem razlike (DoG) [27].

Rotacijska matrika R, dimenzij 3x3, in nam preslika 3D točko tako, da jo ustrezno zavrti okoli vsake izmed treh osi (x,y,z):

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} \\ a_{2,1} & a_{2,2} & a_{2,3} \\ a_{3,1} & a_{3,2} & a_{3,3} \end{bmatrix} \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}.$$

Translacija je geometrijska transformacija, ki premakne vsako podano točko za določeno razdaljo v določeno smer. V našem primeru govorimo o translacijskem vektorju oblike

$$t = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix},$$

kjer vsaka komponenta določa premik v posamezni smeri.

Če želimo sestaviti projekcijsko matriko P, ki nam bo določala relacijo med 2D točko na sliki in 3D točko v svetu, potrebujemo poleg rotacije in translacije tudi kalibracijsko matriko. To sestavljajo notranji (ang. intrinsic) parametri kamere in so specifični za vsako napravo posebej. Izmerimo jih

lahko s pomočjo postopka kalibracije. Kalibracijska matrika je oblike [23]

$$K = \begin{bmatrix} f_x & \alpha f_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.4}$$

in jo sestavlja 5 notranjih parametrov:

- $\boldsymbol{f_x}, \boldsymbol{f_y}$: goriščni razdalji v slikovnih elementih,
- $\boldsymbol{\alpha}$: koeficient ukrivljenosti (ang. skew) med x in y osema,
- $\boldsymbol{c_x}, \boldsymbol{y_y}$: središčna točka.

Rotacijska matrika in translacijski vektor skupaj s kalibracijsko matriko sestavljata transformacijsko matriko P

$$P = K[R|t]. \tag{2.5}$$

Pred začetkom algoritma SfM poznamo samo kalibracijsko matriko, rotacijo in transformacijo pa moramo oceniti iz parov projekcijskih točk, določenih v predhodnjem koraku. Relacijo med pari projekcij opisuje izvorna matrika E (ang. essential matrix), pri čemer predpostavlja, da kamera zadostuje modelu luknjičaste kamere (ang. pinhole camera model). Tako za par korespondenčnih projekcij y in y' iste 3D točke na dve različni sliki velja

$$(y')^T E y = 0, \tag{2.6}$$

pri čemer sta y in y' homogeni in normalizirani točki [37]. Ker zgornja enačba predpostavlja, da opazujemo model luknjičaste kamere, lahko razmerje med dvema množicama 3D točk opišemo kot

$$\tilde{x'} = R(\tilde{x} - t) \tag{2.7}$$

iz česar sledi

$$E = T_{\times} R. \tag{2.8}$$

Tako lahko inicializiramo začetno pozicijo, a se moramo zavedati, da je translacija izračunana iz izvorne matrike E samo enotski vektor, kar pomeni, da nam ni znano merilo premika, ampak samo smer [19].

V vseh naslednjih iteracijah lahko rotacijo in translacijo ocenimo iz 2D-3D korespondenc. Za vsako 3D točko v svetu velja, da nam njeno projekcijo na 2D sliko opisuje projecijska matrika P, kjer velja

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & \alpha f_x & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_{1,1} & a_{1,2} & a_{1,3} & t_1 \\ a_{2,1} & a_{2,2} & a_{2,3} & t_2 \\ a_{3,1} & a_{3,2} & a_{3,3} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.9}$$

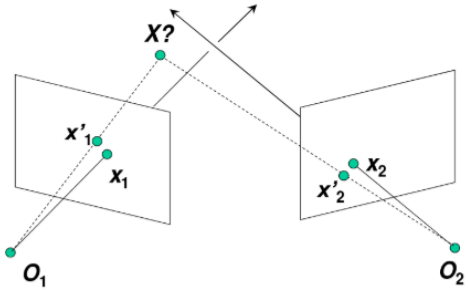
Tako lahko s pari 2D-3D točk in znani kalibracijsko matriko ocenimo rotacijo in translacijo, pri čemer translacija ne bo enotski vektor.

2.2.4 Triangulacija

Triangulacija je proces, ki z uporabo trigonometrije določa pozicijo neznane točke ali lokacije z uporabo pozicije dveh nepremičnih točk, ki imata znano oddaljenost [32].

V primeru računalniškega vida in obdelave slik to pomeni, da lahko določimo neznano točko v tridimenzionalnem prostoru, če poznamo dve njeni projekciji v dvo dimenzionalnem prostoru (na slikah). Slike se morajo glede na vsebino delno prekrivati, saj lahko samo tako algoritem na dveh paroma zaporednih slikah primerja projekcije iste točke iz prostora. Triangulacija predpostavlja, da poznamo matriko in lego kamere za oba pogleda, kar nam sestavlja projekcijsko matriko P.

V idealnem primeru brez šuma in merilnih napak je problem trivialen. Neznana točka leži na presečišču žarkov, ki tečeta čez središčno točko kamere in pripadajočo točko na dvodimenzionalni projekciji. Vendar je v realnosti veliko pogosteje, da se žarka ne sekata, kar je prikazano na Sliki 3.6. V tem primeru je potrebno oceniti točko, ki je najbližje idealni rešitvi.



Slika 2.7: Triangulacija: iskanje najboljšega približka presečišča.

2.2.5 Algoritem Bundle adjustment

Algoritem SfM sestavljata dva glavna koraka: ocenjevanje lege in triangulacija točk, ki sta med seboj tesno povezana. Pri določanju pozicije kamere potrebujemo 3D točke in pri določanju 3D točk potrebujemo pozicijo kamere.

Algoritem za inicializacijo vzame približek oz. oceno in skozi iteracije izmenično uporablja ocenjene točke in ocenjeno pozicijo. Brez vmesnega popravljanja meritev se hitro zgodi, da se začetne napake nakopičijo, kar pripelje do nepopravljivega stanja [31]. Zato na vsakem koraku iteracije uporabljamo tako imenovan Bundle adjustment, ki minimizira projekcijsko napako

$$min \sum_i \sum_j (\tilde{x}_i^j - K \left[R_i | t_i \right] X^j)^2. \tag{2.10}$$

2.3 Registracija oblakov točk

Algoritem ICP rešuje problem registracije dveh oblakov točk, kar pomeni, da aproksimira optimalno rotacijo in translacijo, kateri poravnata (registrirata) drugi oblak točk s prvim. Algoritem lahko uporabljamo za različne reprezentacije geometrijskih podatkov, med drugim množice točk, črte, krivulje in površine [17]. Rotacija in translacija nam rešujeta vseh 6 prostostnih stopenj, 3 za rotacijo in 3 za premik.

Predpostavimo, da imamo podana dva oblaka točk: P je oblak točk,

ki nam predstavlja podatke, X pa je oblak točk, ki nam predstavlja model. Želimo poiskati transformacijo, ki nam bo poravnala naše podatke P z našim modelom X. To pomeni, da želimo zmanjšati razdaljo med korespondenčnimi pari točk v obeh oblakih. Razdalja d med posamezno točko p v P in modelom X je definirana kot

$$d(p, X) = \min_{x \in X} \left\| x - p \right\|. \tag{2.11}$$

Točko v X, ki je na razdalji d od točke p označimo z y, tako velja

$$d(, y) = d(p, X), \tag{2.12}$$

kjer $y \in X$[17].

Če Y označuje množico vseh najbližjih točk in s C označimo operacijo iskanja najbližje točke (opisano po zgornjem postopku) velja

$$Y = C(P, X). \tag{2.13}$$

Tako lahko iz množice točk Y izračunamo registracijo po metodi najmanjših kvadratov Q,

$$(q, d) = Q(P, Y). \tag{2.14}$$

2.3.1 Algoritem ICP

Na začetku algoritem prejme oblak točk P, ki predstavlja podatke, in oblak točk X, ki predstavlja model. Rezultat algoritma bo transformacija, ki poravna podatke z modelom.

Po inicializaciji stanja sledi iteracija naslednjih korakov [21]:

- **Iskanje korespondenčnih točk v slikah**
  Algoritem izračuna množico najbližjih točk $Y = C(P, X)$. Iskanje najbližjih točk je lahko implementirano na več načinov, eden izmed njih je široko uporabljen algiritem K-D drevo (ang. K-D tree) [28].

- **Filtriranje napačno določenih korespondenc**
  Za bolj natančno določeno transformacijo in v izogib zavajajočim vrednostim, je potrebno izločiti pare točk, za katere po določenih kriterijih presodimo, da ne predstavljajo željenega ujemanja [18].

- **Izračun in aplikacija registracije**
  V tem koraku želi algoritem zmanjšati razdaljo med najdenimi korespondenčnimi točkami, kar pravzaprav pomeni, da z vsako iteracijo poskuša oblaka točk čim bolj približati. Najprej izračuna registracijo $(q, d) = Q(P, Y)$, nato pa jo aplicira na trenutno projecijsko matriko $P_{k+1} = q_k(P_0)$.

Algoritem ICP se zaključi, ko je doseženo preddoločeno maksimalno število iteracij oz., ko konvergira (sprememba v razdalji med dvema zaporednima korakoma iteracije je minimalna oz. pod določenim pragom)

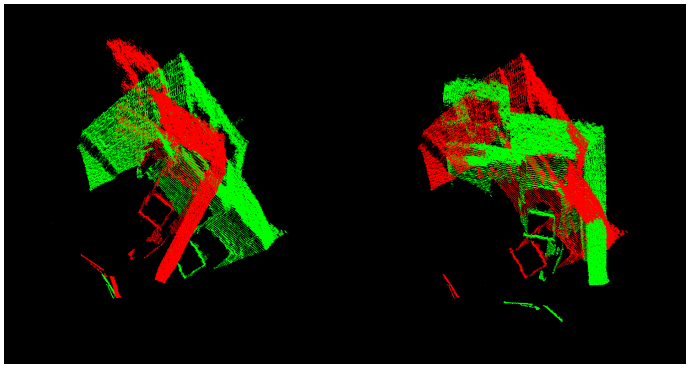
$$d_k - d_{k+1} < \theta. \tag{2.15}$$

Algoritem vedno konvergira v lokalni minimum, glede na funkcijo razdalje najmanjših kvadratov [16], ni pa lokalni venomer tudi globalni minimum. Uspešnost in natančnost algoritma sta tako odvisna od dveh faktorjev:

- **Oddaljenosti posameznih parov točk**
  Če je premik med oblakoma zadosti majhen, lahko z dovolj veliko gotovostjo trdimo, da bo iterativna minimizacija oddaljenosti točk uspešno konvergirala v lokalni minimum, ki bo hkrati tudi globalni minimum.

- **Ustreznega začetnega približka transformacije**
  Dobra začetna ocena transformacije lahko znatno vpliva na uspešnost algoritma, saj pravzaprav nadomesti in zadosti zgornjemu pogoju, ter tako zmanjša možnost konvergiranja v lokalni minimum. Prav tako velja, da bolj kot se začetna ocena razlikuje od dejanske rešitve, dlje časa rabi algoritem, da konvergira.

Iz zgornjih faktorjev lahko ugotovimo glavni pomankljivosti algoritma ICP: konvergiranje v lokalni, ki hkrati ni globalni minimum, in potreba po znatnem prekrivanju oblakov točk. Prva pomankljivost nam lahko prinese popolnoma napačne, neuporabne in celo zavajajoče rezultate, prikazane na Sliki 2.8, druga pa povečuje čas izračunavanja in zahteva nepotrebno redundandnost.



Slika 2.8: Primer slabe poravnave dveh oblakov točk z algoritmov ICP. Levo sta oblaka točk pred poravnavo, kjer lahko vidimo, da sta posneta iz različnih zornih kotov. Desno sta oblaka po konvergiranju algoritma, ki očitno ni našel globalnega minimuma.

2.3.2 Filter VoxelGrid

Vsak oblak točk zajet s senzorjem Kinect ima 307.200 (640 x 480) točk, kar pri združevanju več zaporednih oblakov točk pomeni, da dobimo zelo gosto in redundandno reprezentacijo objekta. Veliko število točk pomeni daljši čas obdelave tako za algoritem ICP (več točk pri računanju razdalj) kot tudi za rekonstrukcijo površine.

Zato pri delu z oblaki točk uporabljamo različne tipe filtrov, ki zmanjšajo količino obdelovanih podatkov (točk). Eden izmed njih je tudi VoxelGrid filter [22]. Namen filtra je zmanjšati gostoto oblaka točk z združevanjem več sosednjih točk v eno točko - cetroid. S tem postane oblak točk bolj enoten, vendar s tem igubimo določeno natančnost oz. podrobnosti. Kjub temu precej zmanjšamo čas in prostor potreben za obdelavo takega oblaka točk. Filter ne odpravi šuma ali napačno določenih točk v oblaku.

2.3.3 Filter PassThrough

Poleg redkejše reprezentacije neke površine nas v določenih primerih reprezentacija določenih delov sploh ne zanima. Eden takšnih primerov je skenira-

nje manjšega predmeta v bližini, kjer nas ne zanima velika (morda monotona) površina v ozadju (velikokrat ravna stena). Ne samo, da povečuje število točk za obdelavo, ampak s svojo številčno reprezentacijo zavede algoritem ICP, ki daje preveliko težo poravnavi ravnih površin v ozadju in na ta račun napačno poravna manjši predmet v ospredju. Za ta namen uporabljamo filter PassThrough [1], ki ohrani samo dele oblaka točk (glede na podano dimenzijo in razpon).

2.4 Integracija

Po registraciji oblakov točk lahko le te združimo v skupen oblak točk, ki reprezentira skenirani objekt. Naslednji korak za rekonstrukcijo 3D modela je triangulacija in rekonstrukcija površine.

Natančnejši oblak točk nam omogoča lepšo in bolj natančno rekonstrukcijo 3D modela. Zato je pomembno, da oblak točk po združitvi primerno obdelamo. Pogosta pojava pri registriranih oblakih točk sta šum, ki je navadno posledica merilnih napak, ter pojav "dvojnih sten"(ang. double wall), ki je posledica slabše poravnanih oblakov točk.

Ko imamo ustrezno filtriran in obdelan oblak točk, lahko rekonstruiramo površino. Za rekonstrukcijo površine mora imeti vsaka izmed točk ustrezno normalo. Normale točk nam povedo smer površine. Za bolj enotno površino je pomembno, da so normale točk usklajene z okolico; zato je potrebno normale točk zgladiti (ang. smoothening) [29].

Glede na normale točk lahko določimo površino objekta. Površina je sestavljena iz manjših trikotnikov, ki ustrezno povezujejo tri sosednje točke. V našem primeru integracije oblakov točk nismo implementirali, temveč smo za obdelavo uporabili obstoječo programsko opremo.

Poglavje 3

Implementacija

V tem poglavju bomo predstavili implementacijo končne rešitve ter uporabljeno programsko in strojno opremo.

Najprej predstavimo programsko in strojno opremo, nato pa celotno zgradbo in delovanje sistema. Pri tem sledimo cevovodu predstavljenem na začetku poglavja, omenimo korake, ki so vodili do končne rešitve in utemeljimo izbiro korakov za končno implementacijo.

3.1 Strojna in programska oprema

Končna rešitev je napisana v programskem jeziku C++ in uporablja dve trenutno najbolj uporabljeni knjižnici za obdelavo slik (OpenCV [2]) ter obdelavo oblakov točk (PCL - Point Cloud Library [6]). Za zajem oblaka točk uporablja senzor Microsoftov Kinect generacije v1 (in ne v2, ki ima precej izboljšane performance), predstavljen v poglavju 2.1.

Za lažjo kominikacijo med uporabnikom in programom je na senzorju Kinect nameščen gumb, ki sporoča, kdaj želi uporabnik zajeti novo sliko, kar lahko vidimo na Sliki 3.1. Komunikacija je vzpostavljena preko mikrokrmilnika Arduino UNO, ki se na računalnik poveže preko standardnega USB vhoda in se mu predstavi kot HID (Human Interface Device). Tako se lahko uporabnik prosto giblje po prostoru.

Za končno obdelavo in integracijo oblaka točk uporabljamo program MeshLab [4].

Slika 3.1: Arduino UNO (z gumbom) nameščen na Kinect senzor.

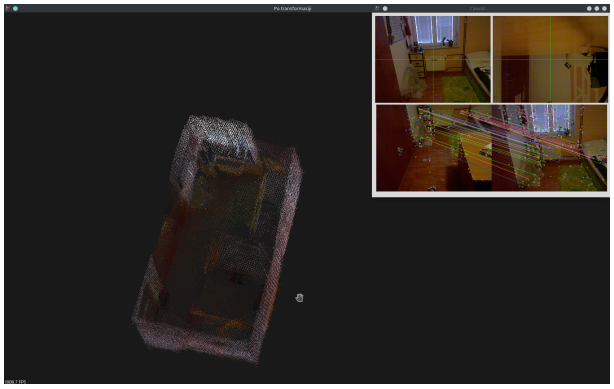
3.2 Zgradba in delovanje sistema

Implementacija sledi cevovodu predstavljenem v uvodni predstavitvi predlagane rešitve opisane v poglavju 1.3. Zajemanje podatkov ni zvezno, kar pomeni, da uporabnik sam sporoči sistemu, kdaj želi zajeti določeno sceno. To lahko stori z gumbom nameščenim na dno senzorja Kinect. Tako lahko uporabnik skeniranje prekine za poljuben čas in ga po želji nadaljuje od zadnje referenčne slike naprej. Prav tako tak način zajemanja podatkov uporabniku omogoča, da sam odloča koliko slik želi posneti za določeno območje. Tako zajemanje slik ni pogojeno na hitrost premikanja, prav tako s tem ni pogojena natančnost.

Tekom celotnega skeniranja lahko uporabnik spremlja trenutno stanje na zaslonu, kot je to prikazano na Sliki 3.2.

Algoritem SfM

Na začetku program inicializira stanje in uporabnika obvesti, da lahko prične z zajemanjem podatkov. Uporabnik lahko s pritiskom na gumb zajame trenutno opazovano sliko, ki se mu prikazuje na zaslonu. Poleg trenutne slike, ki jo vidi senzor, je prikazana tudi zadnja zajeta slika, ki jo uporabnik upo-



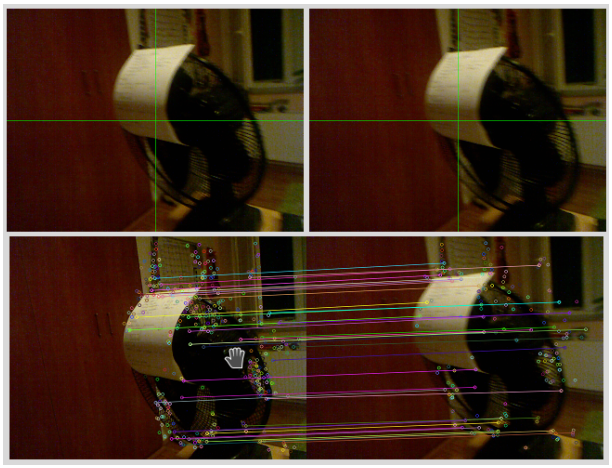
Slika 3.2: Prikaz uporabniškega vmesnika. V zgornjem desnem kotu sta prikazana trenutna slika senzorja Kinect ter predhodnje zajeta (referenčna) slika. Pod njima je prikazano ujemanje med trenutnima paroma slik. V ozadju so vizualizirani vsi koraki algoritma (delna poravnava, poravnava med izvajanjem algoritma ICP), ter vsi registrirani oblaki točk.

rablja za referenco. Izrisano ima tudi središče slike, tako da lahko lažje skrbi za ustrezno prekrivanje zaporednih posnetkov.

V programu se prikazuje aktualna slika v višji ločljivosti neposredno iz barvne kamere senzorja Kinect, za obdelavo se slika pridobi iz trenutno zajetega oblaka točk, saj se s tem izognemo morebitnim neskladjem med zajeto sliko in zajetim oblakom točk.

Ko uporabnik zajame dve delno prekrivajoči se sliki, program začne korake algoritma SfM. Sliki pretvori v sivine, nato pa na njiju poišče ključne točke in njihove opisnike. Za to uporablja SURF (Speeded up robust features) opisnik in 5 oktavno piramido.

Nato poišče pare projekcij med slikama. Za to uporablja knn (k-nearest neighbors) iskanje in uporabi le simetrična ujemanja. Oba koraka vizualizira uporabniku. Na sliki 3.4 lahko vidimo primer izrisa. Vsaka izmed zajetih slik ima označene zaznane kjučne točke (barvni krogci) ter pare projekcij iste točke iz prostora (barvne črte). Na sliki 3.3 je razvidno, da iskanje točk deluje tudi na zamegljeni sliki (posledica hitrega premikanja).



Slika 3.3: Prikaz ujemanja kljub zamegljeni sliki.

V naslednjem koraku uporabi točke iz parov korespondenčnih projekcij za oceno izvorne matrike (ang. essential matrix). Funkcija poleg točk z vsake slike potrebuje tudi kalibracijsko matriko kamere. Kalibracijska matrika uporabljena v programu je

$$K = \begin{bmatrix} 585,5 & 0,00130 * 585,5 & 327,9 \\ 0 & 568,5 & 246,2 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.1}$$

Metoda uporablja RANSAC, ki je algoritem za izločanje napačno določenih parov točk (outliers) [20].

Iz izvorne matrike nato izrazimo rotacijo R in smer translacije t. Tu se pojavi največja težava algoritma, saj je merilo premika nedoločljivo. Zato se zgornji koraki na prvih dveh slikah uporabljajo samo za inicializacijo sistema. Sicer pa je rotacija, ki jo dobimo preko zgoraj opisanega sistema, precej natančna, kar je razvidno iz Slik 3.5.

Z relativno rotacijo in translacijo lahko trianguliramo izbrane točke, kar pomeni, da ocenimo njihovo pozicijo v tridimenzionalnem prostoru. Za triangulacijo potrebujemo projecijski matriki obeh kamer pomnoženi s kalibra-

Slika 3.4: Prikaz detekcije in izrisa korespondenčnih projekcij. Zgoraj z dodano oznako in spodaj brez.

cijsko matriko. Pri inicializacijskem paru sta to

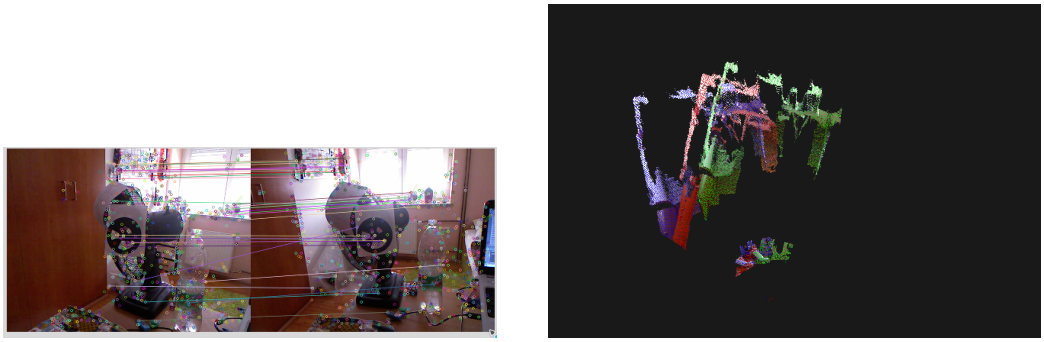
$$P_0 = K * \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{3.2}$$

in

$$P_1 = K * Rt. \tag{3.3}$$

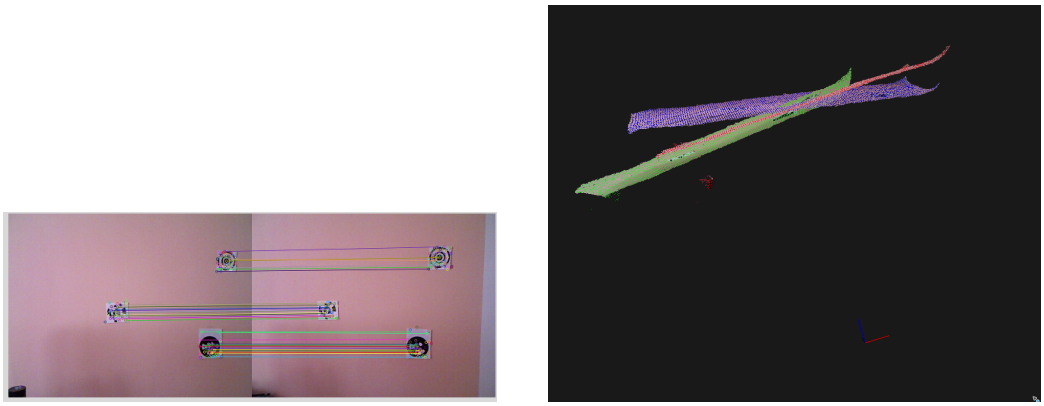
Triangulacija je bolj pravilna in zanesljiva, če sta inicializacijska posnetka bolj narazen, saj tako tvorita večji kot in so numerične napake manjše. Še vedno je potrebno izločiti napačno triangulirane točke, to so na primer takšne, ki ležijo za kamero (negativna z koordinata). Rezultat triangulacije je redek oblak točk, prikazan na Sliki 3.6.

V naslednjih iteracijah se ponavljajo zgoraj opisani koraki. Za pravilno translacijo moramo najti točke iz korespondenčnih parov projekcij, ki so v predhodnih korakih že bile triangulirane. Tako lahko določimo pravilno mero



(a) Poravnava preprostih oblakov točk z veliko identificeranimi skupnimi točkami.

(b) Poravnava oblakov točk zajetih okoli objekta z precej različnih kotov. Rotacija je pravilna klub majhnemu številu skupnih točk in več napačnimi pari.



(c) Poravnava oblakov točk na prazni ravni steni, obogateni z vizualnimi oznakami.

Slika 3.5: Poravnave oblakov točk z rotacijo pridobljeno z izvorno matriko. Rdeč oblak točk je začetni oblak točk, moder je drugi oblak točk pred rotacijo. Zelen oblak točk je drugi oblak točk po poravnavi. Na vseh primerih lahko vidimo, da je pridobljena rotacija zanesljiva.

Slika 3.6: Rezultat triangulacije.

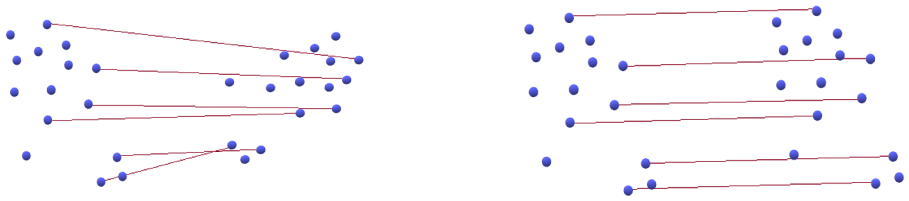
premika. Omenjeno omogoča algoritem PnP (Perspective-n-Point [25]), ki poleg 3D-2D parov točk potrebuje še kalibracijsko matriko kamere.

Knjižnica OpenCV ponuja implementacijo tako za triangulacijo kot tudi algoritem PnP, ki ju potrebujemo za določitev velikosti premika. Eksperimentalni rezultati so pokazali, da sta metodi preveč nestabilni in za več zaporednih posnetkov iste scene vračata nepredvidljivo različne rezultate. Zato končna rešitev podatek o velikosti premika pridobi iz globinske slike senzorja Kinect.

Za vsako točko na sliki poznamo njene koordinate v 3D prostoru (te lahko razberemo iz oblaka točk). Tako lahko s prej najdenimi pari projekcij na 2D slikah določimo pare 3D točk v prostoru. Pomembno je, da izločimo vse točke, ki nimajo podatka o globini (tehnična omejitev senzorja Kinect). Za oblaka točk A in B tako velja, da so njune skupne točke odaljene za enako razdaljo, če imata oblaka poravnani rotaciji, kar je razvidno iz Slike 3.7.

Algoritem Bundle adjustment, ki je sicer nujno potreben pri algoritmu SfM, pri naši implementaciji pravzaprav ne potrebujemo. Ta zadnji korak je namenjen zmanjševanju napak, ki se sicer hitro akumulirajo, ko gradimo model na podlagi predhodnjih izračunov. V naši implementaciji ocenjujemo rotacijo in translacijo samo na podlagi dveh zaporednih oblakov točk, kar nam omogoča izmerjena pozicija korespondenčnih 3D točk.

Tako lahko izračunamo povprečno odaljenost oblakov točk ter vrednost pomnožimo z prej izračunano smerjo premika. Na tem koraku imamo tako



Slika 3.7: Prikaz velikosti premika med dvema oblakoma točk. Levo je neporavnan oblak točk, kjer so razdalje med pari 3D točk različne. Desno je oblak točk po poravnavi rotacije, kjer so vsi pari točk enako oddaljeni.

ocenjeno rotacijo ter translacijo in z njima lahko oblak točk delno poravnamo.

Algoritem ICP

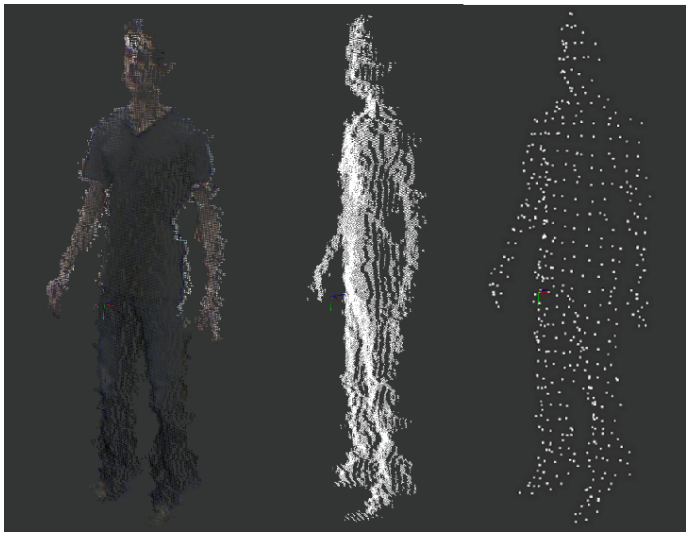
Naslednji korak cevovoda končne implementacije je registracija oblakov točk z algoritmom ICP. Implementacijo algoritma ponuja zgoraj omenjena knjižnica PCL.

Oblaka točk najprej delno poravnamo z zgoraj pridobljeno rotacijo in translacijo. Nato iz oblakov odstranimo vse neštevične (NaN) vrednosti, ki so večinoma posledica tehničnih omejitev senzorja Kinect. Poleg omejene globine zaznave senzor Kinect ne bo zaznal točk na površinah, ki preveč absorbirajo IR svetlobo.

V naslednjem koraku uporabimo filter VoxelGrid, ki nam zmanjša število točk, saj večje število točk znatno podaljša čas potreben za izračun razdalj med točkami v algoritmu ICP. Izbrana velikost lista v filtru je 0.04, kar nam znatno zmanjša število točk (iz 307.200 na povprečno 3.000 za posamezen oblak), a še vedno dovolj natančno reprezentira opazovani oblak točk. Primer filtriranja oblaka točk je prikazan na Sliki 3.8.

Algoritem ICP upošteva več parametrov, ki precej vplivajo na njegovo delovanje in so odvisni predvsem od lastnosti vhodnih podatkov. Tako lahko med drugim nastavimo največje število korakov algoritma, razliko, ki določa konvergiranje algoritma, in najdaljšo razdaljo med pari točk.

V našem primeru imamo približno poravnavo oblakov točk zelo natančno.



Slika 3.8: Prikaz rečenja oblaka točk za hitrejše delovanje algoritma ICP [14].

Eksprimentalni rezultati so pokazali, da lahko, kot pare točk, upoštevamo samo točke na razdalji ne več kor 15cm. Omejitev števila korakov je implementirana iterativno, pri čimer izvajamo algoritem po dva koraka hkrati in med posameznimi iteracijami preverjamo in prilagajamo parametre. V primeru, da je razlika v transformaciji dovolj majhna, zmanjšamo razdaljo med korespondenčnimi točkami, ter tako bližje kot sta oblaka točk, bolj povečujemo natančnost poravnave.

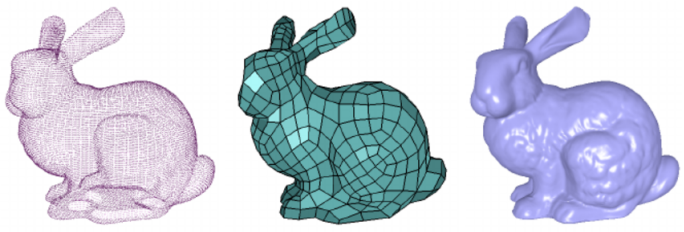
Algoritem prekinemo po 20 iteracijah oz., ko izračunana mera prileganja doseže določen prag.

V izogib akumuliranju napak novo zajeti oblak točk registriramo z združenim oblakom točk predhodnje zajetih scen.

Obdelava skeniranega objekta

Po končanem skeniranju združimo poravnane oblake točk in jih shranimo v datoteko v formatu PLY. Nato lahko oblak točk uvozimo v MashLab [4] ali kater drug pogram za obdelavo 3D objektov. V programu lahko oblak točk obdelamo tako, da ostranimo šum, nato pa rekonstruiramo površino

z izbranim algoritmom, ki točke v oblaku poveže z več trikotniki in tako ustvari površino. Površino lahko kasneje tudi zgladimo. Proces je prikazan na Sliki 3.9.



Slika 3.9: Prikaz korakov transformacije oblaka točk v 3D model [35].

Poglavje 4

Eksperimentalni rezultati

V tem poglavju bomo predstavili in interpretirali rezultate implementirane rešitve, s katerimi smo ovrednotili delovanje naše implementacije. Za meritev izberemo dva različna tipa objektov. Najprej analiziramo delovanje na večjih (soba), nato pa še na manjših objektih.

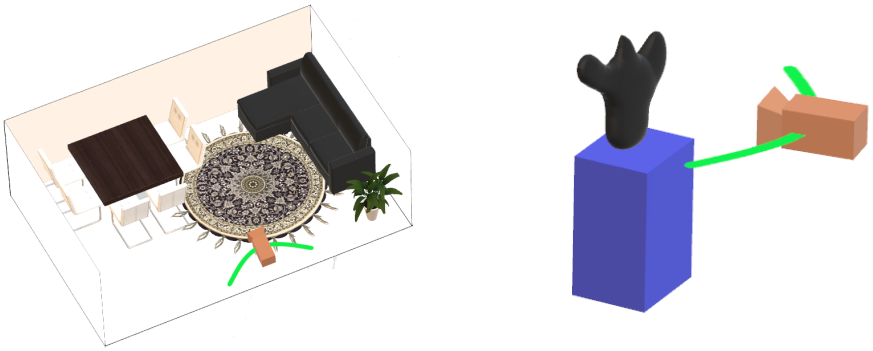
4.1 Postavitev eksperimentov

Natančnost končnega algoritma bomo določili s primerjanjem mer dejanskega in rekonstruiranega 3D modela. Glede na tip skeniranih objektov meritve delimo na dva dela.

Prvi tip meritev opravimo s skeniranjem prostora. Tu gibanje senzorja sledi izbočenemu loku (prikazano na Sliki 4.1 levo) in je zahtevana natančnost manjša, kar pomeni, da so napake manj opazne. Prav tako moramo pri tovrstnem skeniranju reševati problem geometrijske razgibanosti (stene). Za evalvacijo ocene bomo rekunstroirali celotno sobo in izmerili natančnost štirih dimenzij.

Drugi tip meritev manjših objektov se nanaša na skeniranje, kjer premikanje sledi ubočenemu loku (prikazano na Sliki 4.1 desno). Gre za manjše predmete, kjer je potrebna večja natančnost in so zato (sicer manjše) napake pri rekonstrukciji 3D modela relativno velike. Za evalvacijo ocene bomo re-

konstruirali kvader dimenzij 10cm x 7cm x 6cm in zvočnik dimenzij 6,9cm x 8,2cm x 3cm.



Slika 4.1: Prikaz loka skeniranja v odvisnosti od skeniranega objekta.

Objekte smo najprej skenirali in poravnali z zgoraj opisanim algoritmom, nato pa jih uvozili v MeshLab, kjer smo združen oblak točk še enkrat filtrirali (za izločanje napačno detektiranih točk in šuma), nato pa rekonstruirali površino. Pri skeniranju manjših objektov smo s programom MashLab odstranili tudi okoliške točke, ki jih ne potrebujemo za rekonstrukcijo.

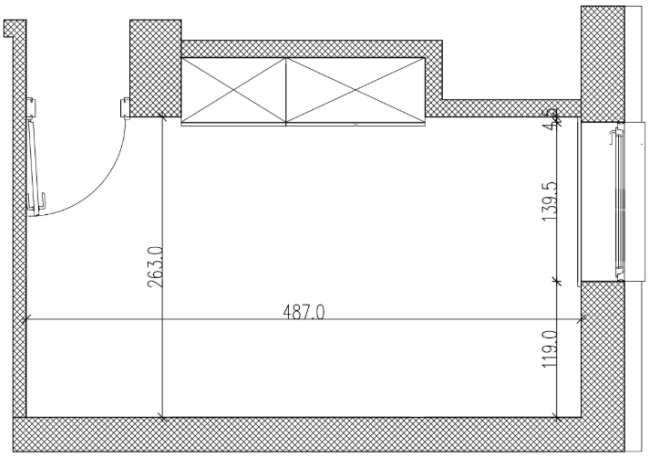
4.2 Rezultati konveksnega skeniranja

V prvem delu smo skenirali celotno sobo tako, da smo "sklenili zanko"oz. združili začetno in končno opazovano točko.

Skenirali smo sobo prikazano na Sliki 4.2. Rezultat skeniranja je bil združen oblak točk, ki smo ga najprej filtrirali. Tako smo zmanjšali debelino slabše poravnanih predelov in hkrati precej zmanjšali gostoto končnega oblaka točk. S tem nismo izgubili natančnosti, saj so bile združene točke redundantne. Kvantitativni podatki meritev so prikazani v tabeli 4.1.

Objekt	Št. oblakov točk	Št. točk po združitvi	Št. točk po filtriranju
Soba	42	12.902.400	10.068.820

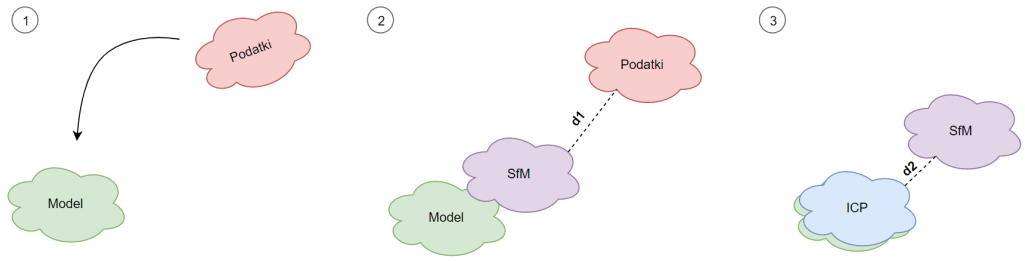
Tabela 4.1: Kvantitativni podatki skeniranja sobe.



Slika 4.2: Načrt skenirane sobe.

Za vrednotenje izboljšave registracije s predhodno grobo poravnavo, smo primerjali oddaljenost zaporednih oblakov točk pred in po grobi poravnavi. Tu smo za vrednotenje upoštevali samo mero translacije; torej za koliko sta poravnana oblaka točk oddaljena med seboj. Mera translacije je tesno povezana s pravilnostjo rotacije, saj napačno rotiran oblak točk negativno vpliva na oceno translacije. Slika 4.3 prikazuje, kako smo pridobili opazovane vrednosti. Pri grobi poravnavi algoritem izračuna razdaljo med oblakoma točk, ki je le ocena dejanske razdalje, na sliki označena z d_1. Algoritem ICP določi rotacijo in translacijo za končno registracijo oblakov točk, za katero predpostavljamo, da je točna. Na sliki je translacija označena z d_2. Razdalja d_2 med grobo poravnanim in končno poravnanim oblakom točk je naše merilo natančnosti. Pove nam, kako blizu (natančno) je algoritem SfM približal oblaka točk. Na ta način lahko ocenimo natančnost algoritma SfM in izračunamo relativno napako. Relativna napaka se ocenjuje glede na ocenjeno začetno oddaljenost oblakov točk. Če algoritem za grobo poravnavo naredi napako velikosti x, bo relativna napaka manjša pri veliki začetni oceni in večja pri manjši začetni oceni razdalje. S tem vrednotimo kako natančen, glede na začetno ocenjeno razdaljo oblakov točk, je algoritem. Visoka relativna napaka bi lahko pomenila, da z grobo poravnavo oblaka točk pravzaprav še bolj

oddaljimo.



Slika 4.3: Prikaz pridobivanja meritev za ocenjevanje natančnosti. Začetno stanje predstavljata neporavnana oblaka točk, ki ju označimo kot "Podatki" in "Model". Podatke želimo poravnati z modelom. Najprej grobo poravnamo oblaka točk. Za poravnavo potrebujemo oceno rotacije in translacije. Mero translacije za grobo poravnavo označimo z d_1. V naslednjem koraku z algoritmom ICP registriramo grobo poravnan oblak točk in model. Tudi tu potrebujemo izračunati rotacijo in translacijo. Mero translacije za končno poravnavo označimo z d_2. Vrednost razdalje d_2 nam predstavlja natančnost grobe poravnave.

Rezultati meritev prikazani v Tabeli 4.2 prikazujejo povprečje obeh meritev skupaj s podatkom o povprečnem številu parov točk med slikama. Upoštevali smo le končne pare točk, kar pomeni, da niso bili izločeni v nobenem koraku potrebnem za delno poravnavo. Iz rezultatov v tabeli lahko razberemo, da je algoritem z grobo poravnavo v povprečju oblaka točk približal na razdaljo 10 cm. To nam omogoča, da algoritem ICP za iskanje korespondenčnih točk pregleduje precej manjše območje in tako hitreje konvergira, hkrati pa z večjo verjetnostjo konvergira v globalni minimum.

Združen in filtriran oblak točk ter rekonstruiran 3D model za sobo je prikazan na Sliki 4.4. Iz rezultatov je razvidno, da sta relativni razliki med dejansko in izmerjeno dolžino krajše stranice in širine okna precej majhni, obe manjši od 3%. Pri višini stropa pride do nekoliko večjega odstopanja,

Objekt	Št. točk v preseku	Razdalja pred poravnavo	Razdalja po poravnavi	Relativna napaka
Soba	9,58	44,2 cm	10,1 cm	22,8%

Tabela 4.2: Kvalitativni podatki skeniranih objektov - vrednotenje izboljšave poravnave s predhodno grobo poravnavo.

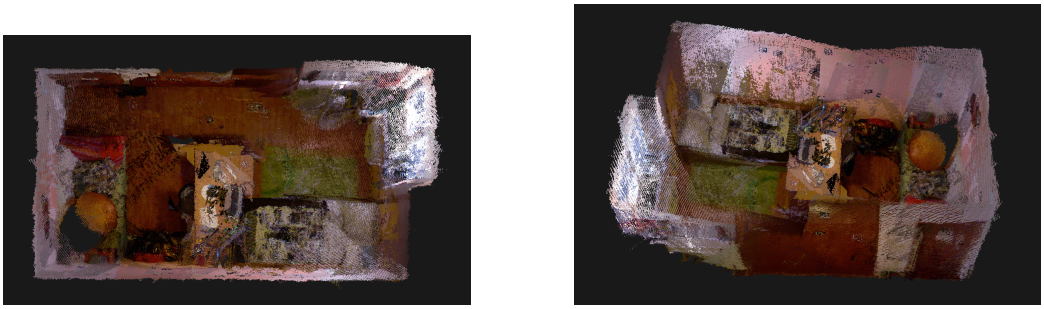
predvsem zaradi monotone geometrije na izmerjenem delu sobe. Največ težav ima algoritem pri poravnavi ravnih površin, kjer izstopa ena glavnih pomankljivosti ICP algoritma. To je opazno pri rekonstrukciji daljše stranice.

Rezultati meritev so zbrani v Tabeli 4.3. Merjene dimenzije pa so označene na Sliki 4.5. Največja napaka je, pričakovano, pri najdaljši dimenziji, saj je soba na obeh daljših dimenzijah (na eni strani stena na drugi omara) geometrijsko zelo monotona. Za uspešno skeniranje je bilo potrebno dodati precej oznak in med izvajanjem programa je bilo razvidno, da je na teh mestih algoritem za grobo poravnavo zelo natančno poravnal oblaka točk, vendar je algoritem ICP navadno našel napačen lokalni minimum.

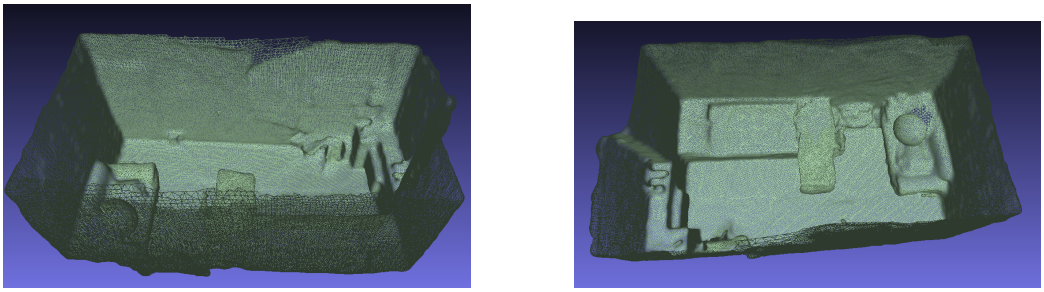
Skeniranje smo izvajali približno v smeri urinega kazalca pri čemer smo prvo sliko zajeli obrnjeni proti oknu ter nato nadaljevali po daljši stranici. Ravno pri prehodu preko daljše stranice je algoritem nekoliko napačno poravnal oblake točk ter tako izgubili razmerje sobe. Posledica tega je tudi nekakšna ukrivljenost sobe.

Dimenzija sobe	Dejanska dolžina	Izmerjena dolžina	Absolutna razlika	Relativna razlika
Krajša	263 cm	257 cm	6 cm	2,28%
Daljša	487 cm	424 cm	63 cm	12,9%
Širina okna	139,5 cm	137 cm	2,5 cm	1,79%
Višina stropa	252 cm	241 cm	11 cm	4,37%

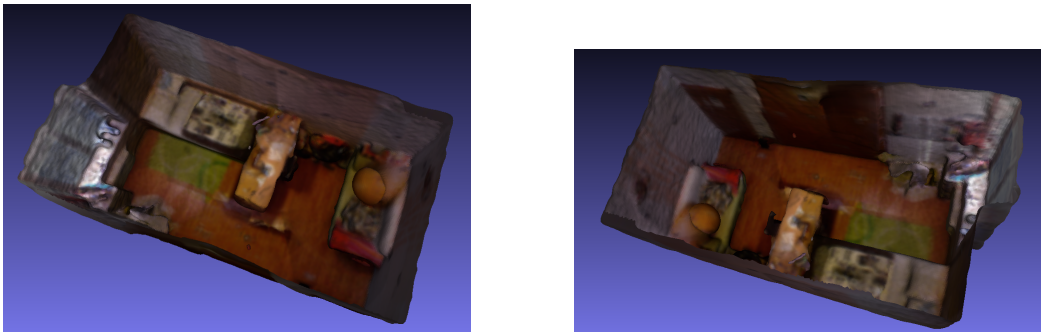
Tabela 4.3: Kvalitativni podatki skeniranja sobe.



(a) Združen oblak točk.



(b) Triangulirana površina (brez dodane teksture).



(c) Rekonstruirana površina (MeshLab).

Slika 4.4: Prikaz združenega oblaka točk, rezultatov triangulacije in končne rekonstrukcije.

Slika 4.5: Prikaz merjenih dimenzij.

Vrednotenje izboljšave registracije z grobo poravnavo

Med izvajanjem meritev je bilo moč opaziti, kako pomembno vlogo v implementaciji ima groba poravnava na podlagi vizualnih oznak. Učinkovitost se je izkazala predvsem pri rekonstrukciji daljše stranice, ker je imel algoritem ICP težavo že pri skoraj povsem poravnanih oblakih točk, saj zaradi geometrije ni uspel določiti pravih značilnih točk. Algoritem ICP za steno prikazano na Sliki 4.6 brez grobe poravnave nikoli ni našel globalnega minimuma.

Celotno površino stene smo pred skeniranjem opremili z vizualnimi oznakami (prikazano na Sliki 4.6), zato je imel algoritem SfM dovolj značilnih točk za delno poravnavo. Rezultati natančnosti poravnave so razvidni iz Tabele 4.2. Iz rezultatov lahko razberemo, da je algoritem za grobo poravnavo oblaka točk v povprečju približal na razdaljo 10cm, kar omogoča mnogo boljše delovanje algoritma ICP.

Potencialni problem bi se lahko pojavil pri določanju 2D-3D korespondenc za črno-bele oznake, saj črna barva absorbira IR svetlobo in tako pogosto senzor Kinect na teh mestih nima podatka za globino. V praksi se je izkazalo, da je imel algoritem podatek o globini za zadostno število točk, da je lahko zadovoljivo natančno ocenil faktor premika.

Iz primera je razvidna uporabnost dodatnega koraka za grobo poravnavo oblakov točk. Ne samo, da zmanjšamo število korakov potrebnih za

Slika 4.6: Dodane oznake na geometrijsko monotono površino.

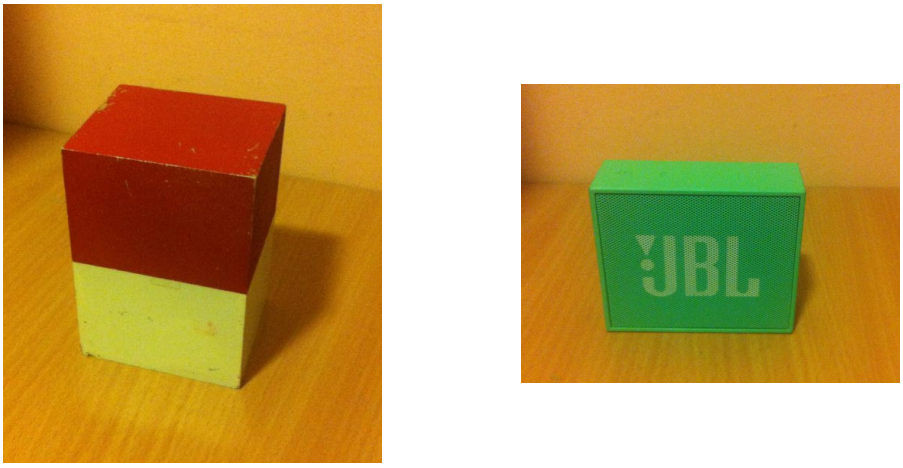
rekonstrukcijo in omogočimo skeniranje s časovnimi premori, ampak skenirani površini dodamo poljubno mnogo vizualih oznak, ki kasneje v obdelavi 3D modela brez teksture pravzaprav niti niso vidne. Če bi želeli enako storiti pri algoritmu ICP, bi z dodajanjem geometrijskih oznak spremenili tudi končno rekonstrukcijo modela.

4.3 Rezultati konkavnega skeniranja

V drugem delu smo skenirali manjša objekta - kvader in pravokotni zvočnik, prikazana na Sliki 4.7. Celoten razvoj naše rešitve je bil usmerjen v skeniranje sobe, vendar bi teoretično moral delovati tudi za skeniranje manjših objektov. Za delovanje je bilo potrebno nekoliko spremeniti algoritem, predvsem prilagoditi nekatere parametre, da smo lahko dosegli zadovoljivo natančnost.

Po filtriranju oblaka točk, je bilo najprej potrebno odstraniti tla (objekt je stal na ravnih tleh). Velike ravne površine, kot so tla ali stene v ozadju, lahko neželjeno vplivajo na delovanje algoritma, saj lahko algoritem ICP poravnavo bolj prilagodi ozadju, kot majhnemu predmetu v ospredju. Zato z odstranitvijo tal (ali stene) poskrbimo, da večjo težo pri poravnavi nosi skenirani predmet. To storimo tako, da stene v ozadju odstranimo s filtrom PassThrough, nato poiščemo največjo množico točk, ki ustreza modelu ravnine in jo odstranimo.

Postavitev okolja za skeniranje je bila precej drugačna od tiste za sobo,



Slika 4.7: Kvader (desno) in zvočnik (levo) uporabljena za ocenjevanje natančnosti skeniranja.

saj je bilo potrebno več priprav. Slika 4.8 prikazuje postavitev za skeniranje kvadra oz. zvočnika. Predmeta nista potrebovala dodatnih oznak, saj je bila podlaga dovolj raznolika.

Slika 4.8: Postavitev okolja za skeniranje kvadra oz. zvočnika.

Algoritem je imel največ težav pri začetni poravnavi, ki ni bila tako natančna kot pri konveksnem skeniranju, vendar dovolj, da je algoritem ICP vedno konvergiral v globalni minimum. Končna registracija oblakov točk je bila venomer pravilna, vendar se napake pri majhnih predmetih (manjših od 10 cm) zelo hitro opazi in le te vplivajo na končni rezultat. Za končno rekonstrukcijo je bilo zato potrebno nekaj več obdelave v programu MashLab.

Najprej je bilo potrebno odstraniti vse točke, ki predstavljajo okolico predmeta. Za boljšo rekonstrukcijo je bilo potrebno odstraniti "dvojne zidove"(ang. double walls); vse stranice objekta, ki so bile preslabo poravnane.

Tabela 4.4 prikazuje kvantitativne rezultate meritev.

Predmet	Št. oblakov točk	Št. točk po združitvi	Št. točk po odstranitvi okolice	Št. točk po filtriranju
Kvader	12	3.686.400	41.467	29.355
Zvočnik	14	4.300.800	23.963	2.709

Tabela 4.4: Kvantitativni podatki skeniranja kvadra in zvočnika.

Na Slikah 4.9 in 4.10 so prikazani končni rezultati rekonstrukcije skeniranih objektov.

Tudi tu smo za vrednotenje izboljšave registracije, s predhodno grobo poravnavo, primerjali razdalji med oblakoma točk pred in po grobi poravnavi, po enakem postopku, kot v prvem delu testiranja. Tabela 4.5 prikazuje povprečje obeh meritev skupaj s podatkom o povprečnem številu parov točk med slikama. Tudi tu smo upoštevali le končne pare točk, kar pomeni, da niso bili izločeni v nobenem koraku potrebnem za delno poravnavo. Iz rezultatov opazimo, da, kjub visoki relativni napaki, z delno poravnavo skoraj za polovico približamo oblaka točk, kar nam omogoča, da algoritem ICP zmanjša okoliš iskanja korespondenčnih točk. Algoritem tako konvergira hitreje in je minimum v katerega konvergira bolj verjetno globalni.

Za vrednotenje natančnosti skeniranja smo opazovana predmeta izmerili po vseh treh dimenzijah in meritve primerjali z merami rekonstruiranega 3D modela. Izmerjene dimenzije so prikazane na Sliki 4.11.

V Tabeli 4.6 so prikazani rezultati meritev. Pri rekonstrukciji kvadra lahko opazimo, da sta dimenziji x in z rekonstruirani izjemno natančno saj sta obe napaki manjši od enega milimetra. Obe dimenziji sta precej natančno rekonstruirani tudi pri zvočniku, sicer z nekoliko večjo napako.

Objekt	Št. točk v preseku	Razdalja pred poravnavo	Razdalja po poravnavi	Relativna napaka
Kvader	62	27,5 cm	13,5 cm	49%
Zvičnik	32	29,3 cm	14,3 cm	48, 8%

Tabela 4.5: Kvalitativni podatki skeniranih objektov - vrednotenje izboljšave poravnave s predhodno grobo poravnavo.

Napake so relativno majhne, saj nobena ni večja od 0,4 cm, a so v veliki meri odvisne od obdelave oblaka točk v programu MeshLab, saj lahko pomotoma odstranimo okoliške točke, ki pripadajo modelu.

Objekt	Dimenzija	Dejanska dolžina	Izmerjena dolžina	Absolutna razlika	Relativna razlika
Kvader	x	7 cm	6,91 cm	0,09 cm	1, 28%
Kvader	y	10 cm	9,73 cm	0,27 cm	2, 7%
Kvader	z	6 cm	5,93 cm	0,07 cm	1, 16%
Zvočnik	x	8,2 cm	8,08 cm	0,12 cm	1, 46%
Zvočnik	y	6,9 cm	6,52 cm	0,38 cm	5, 5%
Zvočnik	z	3 cm	3,22 cm	0,22 cm	7, 33%

Tabela 4.6: Kvalitativni podatki skeniranih objektov - natančnost.

Poleg manjših predmetov z znanimi merami smo rekonstruirali tudi druge objekte različnih velikosti. Kvantitativni rezultati so prikazani v Tabeli 4.7, rezultati rekonstrukcij pa na Sliki 4.12.

V tabeli 4.8 so prikazani tudi kvalitativni podatki za skenirano omarico. Omarico smo izmerili po treh dimenzijah in jih primerjali z dimenzijami rekonstruiranega modela. Rezultati meritev so presenetljivo natančni, predvsem x in y dimenziji, ki sta skoraj enaki dimenzijam izmerjenega objekta. Napaka po dimenziji z je nekoliko večja a relativno še vedno precej majhna. Potrebno je poudariti tudi, da je bila pri rekonstrukciji modela omarice po-

Objekt	Št. oblakov točk	Št. točk v preseku	Razdalja pred poravnavo	Razdalja po poravnavi	Relativna napaka
Ukulele	11	16	22,8 cm	10,9 cm	47,8%
Omarica	17	14	31,06 cm	11,9 cm	38,3%
Roža	12	24	43,45 cm	24,9 cm	57,3%

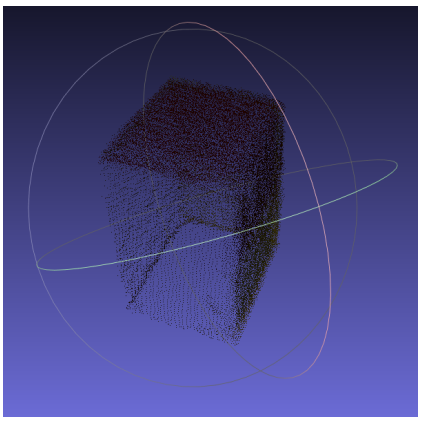
Tabela 4.7: Kvalitativni podatki skeniranih objektov - vrednotenje izboljšave poravnave s predhodno grobo poravnavo.

trebna minimalna obdelava v programu MeshLab, saj poleg okoliških točk ni bilo veliko šuma ali slabo poravnanih oblakov točk. Rezultate je moč pripisati predvsem velikosti in geometriji objekta, prav tako je algoritem za grobo poravnavo zanesljivo približal oblaka točk na dovolj kratko razdaljo.

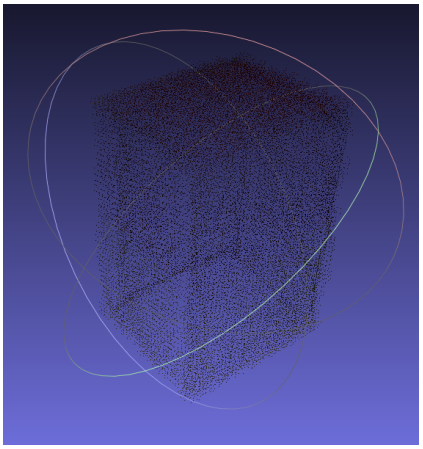
Objekt	Dimenzija	Dejanska dolžina	Izmerjena dolžina	Absolutna razlika	Relativna razlika
Omarica	x	55,5 cm	56,1 cm	0,6 cm	1%
Omarica	y	81,5 cm	80,89 cm	0,61 cm	0,7%
Omarica	z	40 cm	41,05 cm	1,05 cm	2,6%

Tabela 4.8: Kvalitativni podatki skeniranja omarice.

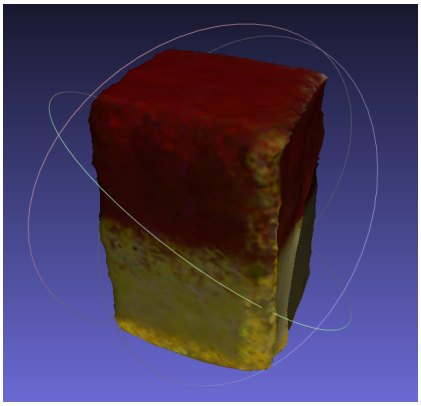
(a) Združeni (registrirani) oblaki točk.



(b) Oblak točk po ročni odstranitvi okoliških točk.

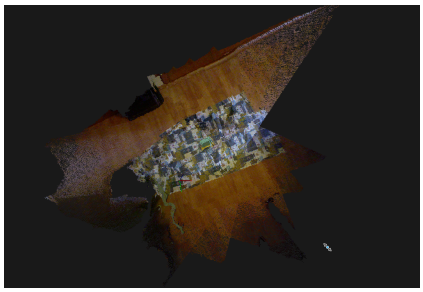


(c) Oblak točk po filtriranju (ang. sampling).

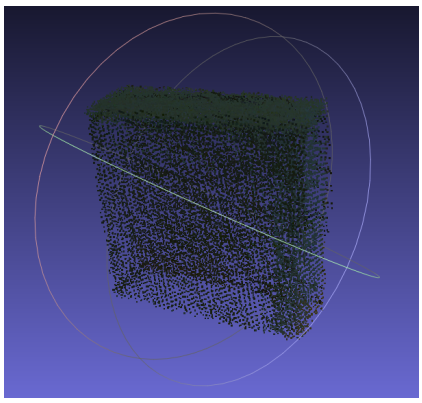


(d) Rekonstrukcija površine.

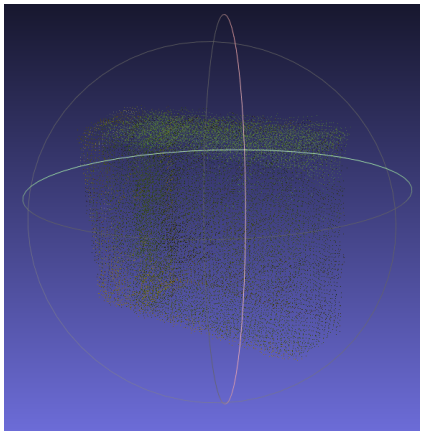
Slika 4.9: Prikaz združenega oblaka točk, rezultatov filtriranja in končne rekonstrukcije za kvader.



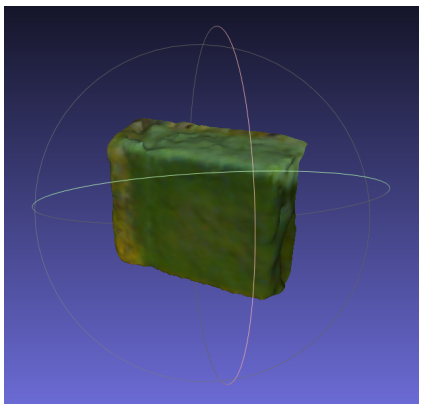
(a) Združeni (registrirani) oblaki točk.



(b) Oblak točk po ročni odstranitvi okoliških točk.

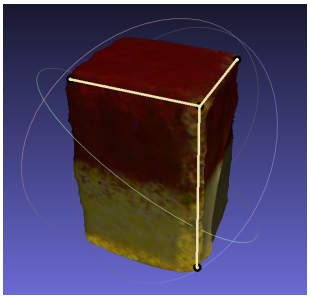


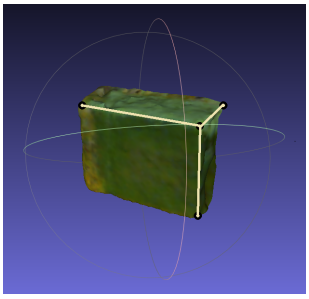
(c) Oblak točk po filtriranju (ang. sampling).



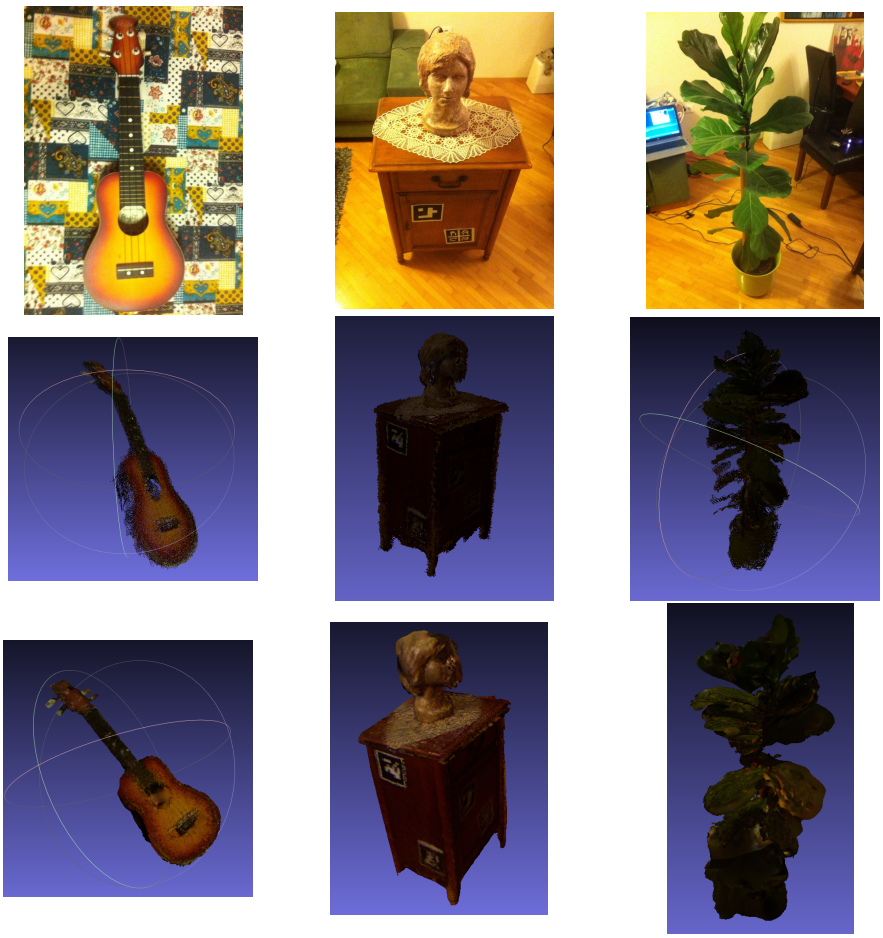
(d) Rekonstrukcija površine

Slika 4.10: Prikaz združenega oblaka točk, rezultatov filtriranja in končne rekonstrukcije za zvočnik.





Slika 4.11: Izmerjene dimenzije 3D rekonstrukcije za kvader in zvočnik.



Slika 4.12: Prikaz skeniranega objekta, združenega in filtriranega oblaka točk ter končne rekonstrukcije za objekte različnih velikosti.

Poglavje 5

Sklepne ugotovitve

Glavni cilj diplomskega dela je bilo izboljšanje algoritma za registracijo oblakov točk na način, da uporablja podatke iz 2D barvne slike. To smo implementirali tako, da smo dva zaporedno zajeta in delno prekrivajoča se oblaka točk pretvorili v 2D barvne slike, na njih poiskali ključne točke, ter ju na podlagi te informacije delno poravnali. Z delno poravnavo oblakov točk smo dosegli, da je algoritem ICP hitreje in uspešneje konvergiral.

Implementirana metoda deluje zelo dobro za skeniranje prostorov in zadovoljivo natančno za skeniranje predmetov. Pri skeniranju prostora je potrebno majhno število zajetih slik, ne-zvezni način skeniranja pa nam omogoča, da zajemamo poljubno velike kadre in s tem zmanjšamo prostor in čas potrebna za rekonstrukcijo. Skeniranje predmetov deluje dobro za večje predmete, pri manjših predmetih so napake relativno večje.

Groba poravnava pred algoritmom ICP znatno približa oblaka točk, kar omogoča, da algoritem ICP hitreje konvergira. Prav tako znatno pomaga pri poravnavi geometrijsko monotonih površin (sten), saj jim lahko dodamo vizualne oznake.

V nadaljevanju bi bilo potrebno algoritmu dodati mehanizem za zaznavo napačne poravnave tako, da bi poravnavo popravil sam, ali pa zahteval ponovno skeniranje. S tem bi algoritem postal bolj robusten.

Algoritem bi poleg robustnosti lahko izboljšali še z dodatno komponentno,

ki bi merila fizične premike senzorja (Arduino s pospeškomerom). S tem bi izboljšali in obogatili informacijo o poziciji kamere. Prav tako bi na senzor Kinect lahko namestili zaslon, ki bi v živo prikazoval rekonstruiran model. Tako bi lahko uporabnik prosto skeniral objekt s pritiskanjem na gumb in opazovanjem modela na zaslonu.

Literatura

[1] Filter PassThrough. http://docs.pointclouds.org/trunk/classpcl_1_1_pass_through.html. Dostopano: 2. 9. 2018.

[2] Knjižnica OpenCV. https://opencv.org/. Dostopano: 2. 9. 2018.

[3] KScan3D. https://lmi3d.com/kscan3d-software. Dostopano: 29. 8. 2018.

[4] MeshLab. http://www.meshlab.net/. Dostopano: 25. 8. 2018.

[5] Microsoft Kinect. https://upload.wikimedia.org/wikipedia/commons/thumb/f/fe/KinectSensor.png/375px-KinectSensor.png. Dostopano: 27. 7. 2018.

[6] PCL. http://pointclouds.org/. Dostopano: 29. 8. 2018.

[7] Primer oznake. https://encrypted-tbn0.gstatic.com/images?q=tbn:ANd9GcS1svcekL4ji5pufSUsT20-2L39RT8Klp7DQ5uJdEABLtgay3Jz. Dostopano: 27. 7. 2018.

[8] Primer rekonstrukcije 3D modela: K-Scann. https://www.heise.de/download/product/kscan3d-88070. Dostopano: 12. 8. 2018.

[9] Proekcija točk. http://gmv.cast.uark.edu/wp-content/uploads/2012/07/KinectIR.png. Dostopano: 27. 7. 2018.

[10] ReconstructMe. http://reconstructme.net/. Dostopano: 29. 8. 2018.

[11] Rekonstrukcija 3D modela. http://zhuoliang.me/meshrecon.html. Dostopano: 12. 8. 2018.

[12] Skanect. https://skanect.occipital.com/. Dostopano: 29. 8. 2018.

[13] Zgradba Microsoft Kinecta. http://gmv.cast.uark.edu/wp-content/uploads/2012/07/KinectIR.png. Dostopano: 27. 7. 2018.

[14] Bruno Albert. Pose tracking of an articulated human body. 09 2013.

[15] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). *Computer vision and image understanding*, 110(3):346–359, 2008.

[16] Paul J Besl and Ramesh C Jain. Three-dimensional object recognition. *ACM Computing Surveys (CSUR)*, 17(1):75–145, 1985.

[17] Paul J Besl and Neil D McKay. Method for registration of 3-d shapes. In *Sensor Fusion IV: Control Paradigms and Data Structures*, volume 1611, pages 586–607. International Society for Optics and Photonics, 1992.

[18] Dmitry Chetverikov, Dmitry Svirko, Dmitry Stepanov, and Pavel Krsek. The trimmed iterative closest point algorithm. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 3, pages 545–548. IEEE, 2002.

[19] Zhaopeng Cui and Ping Tan. Global structure-from-motion by similarity averaging. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 864–872, 2015.

[20] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[21] Stephen D Fox and Damian M Lyons. An approach to stereo-point cloud registration using image homographies. In *Intelligent Robots and Computer Vision XXIX: Algorithms and Techniques*, volume 8301, page 830108. International Society for Optics and Photonics, 2012.

[22] Fengjun Hu, Yanwei Zhao, Wanliang Wang, and Xianping Huang. Discrete point cloud filtering and searching based on vgso algorithm. In *ECMS*, pages 850–856, 2013.

[23] Branko Karan. Calibration of kinect-type rgb-d sensors for robotic applications. *FME Transactions*, 43(1):47–54, 2015.

[24] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

[25] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Epnp: An accurate o (n) solution to the pnp problem. *International journal of computer vision*, 81(2):155, 2009.

[26] Chien-Chou Lin, Yen-Chou Tai, Jhong-Jin Lee, and Yong-Sheng Chen. A novel point cloud registration using 2d image features. *EURASIP Journal on Advances in Signal Processing*, 2017(1):5, 2017.

[27] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[28] Hao Men, Biruk Gebre, and Kishore Pochiraju. Color point cloud registration with 4d icp algorithm. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1511–1516. IEEE, 2011.

[29] Niloy J Mitra and An Nguyen. Estimating surface normals in noisy point cloud data. In *Proceedings of the nineteenth annual symposium on Computational geometry*, pages 322–328. ACM, 2003.

[30] Claudia Raluca Popescu and Adrian Lungu. Real-time 3d reconstruction using a kinect sensor. *Computer Science and Information Technology*, 2(2):95–99, 2014.

[31] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4104–4113, 2016.

[32] Veronica A Thurmond. The point of triangulation. *Journal of nursing scholarship*, 33(3):253–258, 2001.

[33] Thomas Whelan, Michael Kaess, Maurice Fallon, Hordur Johannsson, John Leonard, and John McDonald. Kintinuous: Spatially extended kinectfusion. 2012.

[34] Thomas Whelan, Renato F Salas-Moreno, Ben Glocker, Andrew J Davison, and Stefan Leutenegger. Elasticfusion: Real-time dense slam and light source estimation. *The International Journal of Robotics Research*, 35(14):1697–1716, 2016.

[35] Seung-Hyun Yoon. A surface displaced from a manifold. 4077:677–686, 07 2006.

[36] Zhengyou Zhang. Microsoft kinect sensor and its effect. *IEEE multimedia*, 19(2):4–10, 2012.

[37] Zhengyou Zhang, Rachid Deriche, Olivier Faugeras, and Quang-Tuan Luong. A robust technique for matching two uncalibrated images through the recovery of the unknown epipolar geometry. *Artificial intelligence*, 78(1-2):87–119, 1995.