UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

REKONŠTRUKCIA POVRCHOV S POMOCOU ZARIADENIA KINECT

BAKALÁRSKA PRÁCA

2013

Michal Moravčík

UNIVERZITA KOMENSKÉHO V BRATISLAVE
FAKULTA MATEMATIKY, FYZIKY A INFORMATIKY

REKONŠTRUKCIA POVRCHOV S POMOCOU ZARIADENIA KINECT

BAKALÁRSKA PRÁCA

Študijný program:	Informatika
Študijný odbor:	2508 Informatika
Školiace pracovisko:	Katedra informatiky
Vedúci bakalárskej práce:	Mgr. Martin Samuelčík, PhD.

Bratislava, 2013
Michal Moravčík

Univerzita Komenského v Bratislave
Fakulta matematiky, fyziky a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Michal Moravčík
Študijný program: informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 9.2.1. informatika
Typ záverečnej práce: bakalárska
Jazyk záverečnej práce: slovenský

Názov: Rekonštrukcia povrchov s pomocou zariadenia Kinect

Cieľ: Študent má v práca za úlohu preskúmať možnosti rekonštrukcie povrchov a objektov s použitím zariadenia Microsoft Kinect. Sústredí sa na rekonštrukciu objektov s menšími rozmermi a na použitie už existujúcich knižníc. Výsledkom bude aj aplikácia, ktorá bude riadiť systém skenovania, zobrazovať čiastkové a aj finálne výsledky a výsledný 3D model uloží do externého súboru. Ak to bude možné, výsledný model sa v aplikácii aj otextúruje.

Anotácia: Zariadenie Microsoft Kinect je primárne určené na detekciu ľudských postáv v obraze kamery. Jedným z medzivýsledkov tejto detekcie je ale aj vytvorenie hĺbkovej mapy pre obraz z kamery, t.j. pre každý pixel máme určenú aj jeho približnú hĺbku. Tieto dáta nám definujú približnú rekonštrukciu scény pred kamerou Kinectu. Existuje niekoľko projektov, ktoré sa snažia použiť tieto dáta na rekonštrukciu povrchov scény vo forme množiny trojuholníkov. Cieľom práce bude hlavne preskúmanie takýchto projektov, ich dostupnosť, porovnanie, testovanie ich výsledkov. Druhou časťou bude vytvorenie aplikácie s pomocou jedného z testovaných projektov, ktorá bude rekonštruovať scénu s čo najmenším zásahom používatela.

Vedúci: Mgr. Martin Samuelčík, PhD.
Katedra: FMFI.KAI - Katedra aplikovanej informatiky
Vedúci katedry: doc. PhDr. Ján Rybár, PhD.

Dátum zadania: 19.09.2012

Dátum schválenia: 24.10.2012 doc. RNDr. Daniel Olejár, PhD.
garant študijného programu

.....
študent

.....
vedúci práce

Pod'akovanie

Chcel by som sa pod'akovať môjmu vedúcemu Mgr. Martin Samuelčíkovi, PhD. za výber témy, jeho pomoc, a usmerňovanie.

Abstrakt

Cieľom tejto práce je preskúmanie možností využitia zariadenia Microsoft Kinect, ktorý umožnuje 3D snímanie reálneho sveta za pomoci hĺbkového senzoru, na rekonštrukciu povrchov a objektov. Popisuje techniky využívané pri zarovnávaní skenov scény zíkaných z rôznych uhlov a pri samotnej rekonštrukcii povrchu ktorý tvorí trojrozmerný model danej skenovanej scény. Ďalej implementujeme aplikáciu,ktorá za pomoci zariadenia Microsoft Kinect automaticky rekonštruuje reálnu scénu.

KĽÚČOVÉ SLOVÁ: Microsoft Kinect, registrácia obrazov, rekonštrukcia povrchov, 3D modelovanie

Abstract

The main purpose of this work is to explore the possibilities of using Microsoft Kinect device, which provides a way of 3D scanning of real world with his depth sensor, for the reconstruction of surfaces and objects. It describes techniques used in registering scans acquired from different angles and in reconstruction of a surface which defines a 3D model of the scanned scene. Additionally we implement application that automatically reconstructs a real scene using Microsoft Kinect device.

KEYWORDS: Microsoft Kinect, image registration, surface reconstruction, 3D modeling

OBSAH

Úvod

Ručné modelovanie 3D objektov je Často pracné a časovo náročné. Zjednodušením tohto procesu je možné urýchliť tvorbu modelov a projektov na nich závislých. Zariadenie Microsoft Kinect je vhodnou platformou pre tuto prácu aj vďaka jeho rozšírenosti a dostupnosti.

Trojrozmerné modely objektov sa využívajú v mnohých priemyselných odvetviach vrátane filmového a herného priemyslu, pri návrhu prototypov zariadení alebo pri prezentácii konceptov, dizajnov a návrhov. Naviac so zvyšujúcou sa dostupnosťou 3D tlače sa zvyšuje dopyt po najrôznejších modeloch a spôsoboch ich tvorby. Jedným spôsobom je rekonštrukcia existujúcich objektov pomocou 3D skeneru - zariadenia schopného snímať scénu v troch rozmeroch - šírka, výška a hĺbka. Jedným takýmto zariadenim je aj Kinect, ktorý je zároveň cenovo dostupný a rozšírený, keďže bol pôvodne určený ako ovládač k hernej konzole.

Témou tejto práce je aj vytvorenie aplikácie ktora umožní rýchlo a intuitívne vytvárať 3D modely menších objektov ako sú napríklad plyšové zvieratká, rôzne modely z hliny, kuchynské potreby alebo aj elektornika. Princípom bude použitie mračien bodov ktoré produkuje Microsoft Kinect pomocou svojho hĺbkového senzora. Tieto mračná bodov vytvárajú hĺbkový obraz priestoru ktorý následne budeme spracovávať. Aby sme umožnili vytváranie kompletných modelov pokúsime sa vytvoriť vhodný proces na automatické spojenie záberov z rôznych uhlov, aby bolo možné napríklad obísť Kinectom okolo objektu a vymodelovať ho z každej strany. Tento proces by mal byť prívetivý a jednoduchý a vyžadovať čo najmenší zásah používateľa.

Kapitola 1

Základné pojmy

V tejto kapitole opíšeme zariadenie Microsoft Kinect, jeho vlastnosti a uvedieme definície a pojmy ktoré budeme v práci používať.

1.1 Microsoft Kinect

Zariadenie Microsoft Kinect (ďalej len Kinect) bolo vyvinuté ako doplnkový senzor pre hernú konzolu Xbox 360 a je primárne určené na detekciu ľudských postáv a ich pohyb v obraze kamery. Jedným z medzivýsledkov tejto detekcie je aj vytvorenie hĺbkovej mapy pre obraz z kamery, teda pre každý pixel máme určenú aj jeho približnú hĺbku.

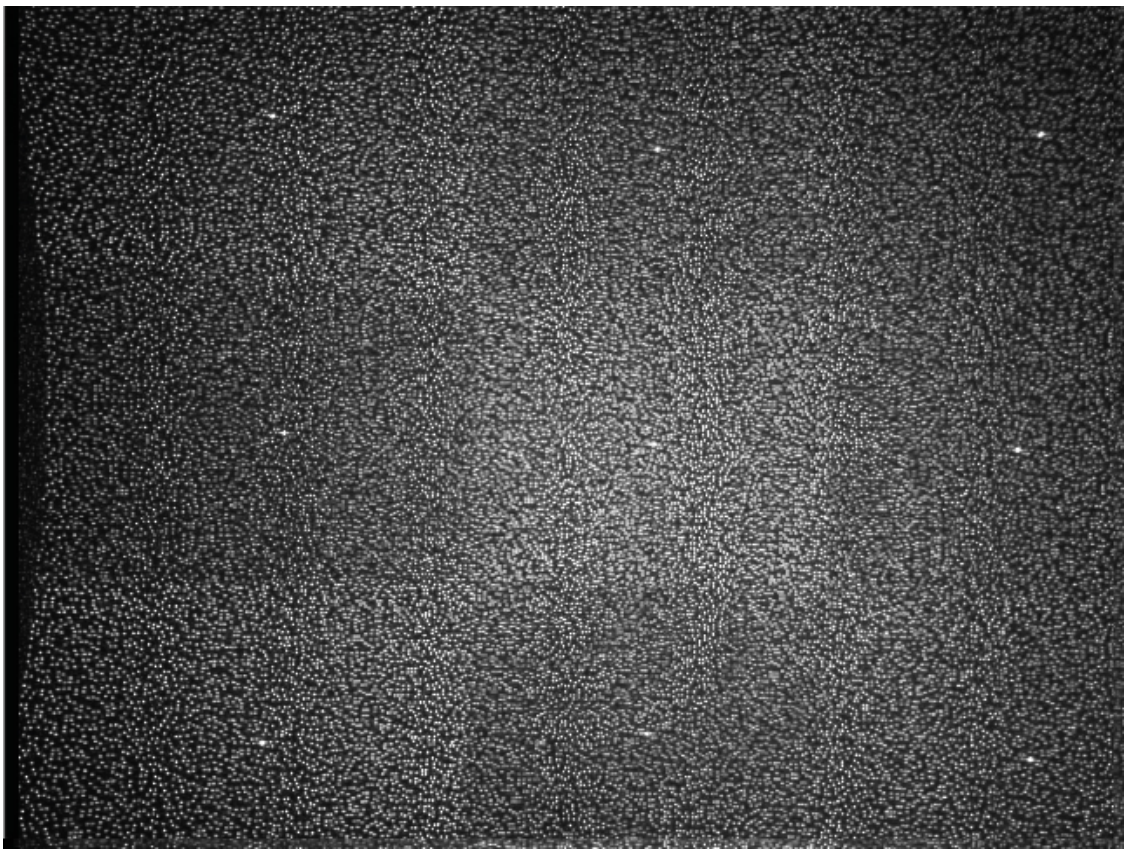
Špecifikácia

Kinect obsahuje RGB (Bayer) kameru s rozlíšením 640 x 480 pixelov pri frekvencií 30Hz, infračervený emitor a CMOS infračervený senzor s rozlíšením 640 x 480 pixelov a 2048 stupňami senzitivity. Okrem vizuálnych senzorov má aj pole štyroch mikrofónov pre lokalizáciu zdroja zvuku a akcelerometer.

Obrázok 1.1: Zariadenie Microsoft Kinect

Hĺbkový senzor

Hĺbkový senzor Kinectu funguje na princípe štrukturovaného svetla. Emitor vypúšťa známy pseudonáhodný vzor a jeho odraz je zachytený senzorom, ktorý ho s daným vzorom porovná. Vzor má 9 hlavných ostrých bodov.



Obrázok 1.2: Vzor vysielaný Kinectom. Svetlejšie body majú vyššiu intenzitu. Zdroj[2, p. 12]

Obmedzenia

Zorné pole senzora je 58° horizontálne 45° vertikálne a 70° diagonálne. S novými drivermi Kinect for Windows a Prime sense sensor je merateľná vzdialenosť v rozmedzí od 0.4 m do 4 m. V najnižšej vzdialenosti teda zo zorného poľa a rozlíšenia vyplýva rozlišovacia schopnosť $\sim 1mm$. Hĺbkový senzor pracuje so svetlom o vlnovej dĺžke 830 nm a preto sa v slnkom osvetlených miestnostiach obajvuje signifikantný šum v hĺbkových údajoch. Viac o vnútornom fungovaní Kinectu popisuje kniha Hacking Kinect[2, p. 12].

Kinect for Windows SDK

Kinect for Windows je developerský toolkit od spoločnosti Microsoft pre tvorbu aplikácii využívajúcich Kinect. Obsahuje spôsob na získavania údajov z Kinectu, sledovanie kostry

a spracovávanie zvuku z jeho mikrofónového poľa. Pre ciele tejto práce bol relevantný len prístup k informáciam z kamery a hĺbkového senzora, preto sa na získavanie a spracúvanie údajov vybrala knižnica PCL ktorá obsahuje aj algoritmy relevantné k povrchovej rekonštrukcii(viac o PCL sa nachádza v kapitole Implemetácia). Počas písania tejto práce bola vydaná verzia 1.7[3] tohto SDK ktorá pridala KinectFusion[4] do toolkitu. KinectFusion poskytuje spôsob na rekonštrukciu povrchu ktorý je cieľom tejto práce. V kapitole výsledky stručne porovnáme KinectFusion s našou implementáciou .

1.2 Point cloud

Point cloud alebo tiež mračno bodov je množina reprezentujúca n-rozmerné dáta. Prvkami tejto množiny sú body definované n súradnicami. Pre účely tejto práce budeme používať implementácie s troma priestorovými súradnicami (ozn. X,Y,Z) ktoré môžu byť doplnené troma súradnicami pre zložky farby (ozn. R,G,B) , alebo povrchovou normálou (ozn. N). Pointcloudy môžu byť priamo získané zo skenovacích zariadení ako je Kinect.

1.3 Transformácia

V tejto práci budeme pod pojmom transformácia pointcloudu myslieť pevnú transformáciu teda zmenu polohy jeho bodov so zachovaním všetkých vzdialeností medzi nimi. Matica T reprezentujúca takúto transformáciu vyzerá nasledovne :

$$T = \begin{pmatrix} Xx & Yx & Zx & Tx \\ Xy & Yy & Zy & Ty \\ Xz & Yz & Zz & Tz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

kde $Xx,Xy,Xz,Yx,Yy,Yz,Zx,Zy,Zz$ tvorí maticu rotácie a (Tx,Ty,Tz) vektor posunu. Viac o tomto type transformácie je možné nájsť na stránke vysvetľujúcej transformácie v OpenGL[5].

1.4 Voxel

Voxel je objemová častica reprezentujúca hodnoty ako napríklad farbu, priehľadnosť, materiál alebo povrchové normály v pravidelnej mriežke v trojrozmernom prostredí. Je trojrozmernou obdobou pixelu. Voxely budeme používať na rozdelenie scény, čo využívajú niektoré neskôr spomínané algoritmy.

1.5 Povrchová normála

Povrchová normála povrchu S v bode $p \in S$ je nenulový vektor $\bar{n}$ kolmý na povrch S v bode p. Normály sú neoddeliteľnou súčasťou vektorových polí používaných rekonštrukčnými algoritmami.

Kapitola 2

Rekonštrukcia povrchov

V tejto kapitole vysvetlíme algoritmy používané pri vytváraní modelu z pointcloudu, ich vlastnosti, výhody a prípadne nevýhody.

2.1 Registrácia

Prvým krokom k modelu je registrácia snímkov skenera. Snímky treba zarovnať do správnej polohy tak aby sa ich prienik prekrýval. Postupným pridávaním a zarovnávaním záberov z rôznych polôh tak môžeme získať snímku celého objektu. V tejto kapitole sa budeme venovať technikám potrebným pre správny výpočet transformáčnej matice tohto zarovnania.

2.1.1 Odstraňovanie outlierov

Outlier je bod, ktorý je signifikantne vzdialený od väčšiny. V našom prípade sa jedná hlavne o šum, okraje scény a okraje objektov. Odstránením týchto bodov sa zvyšuje presnosť pri registrácii a objekt sa vyhladzuje, čím sa zjednodušuje odhad povrchových normál. Nevýhodou ich odstránenia je možná strata jemných detailov.

Radius outlier removal

Jedným spôsobom ako odstrániť outlierov je zadať požiadavku na počet bodov v nejakom okolí bodu. Body ktoré nespĺňajú túto požiadavku sú označené a po prejdení všetkých bodov v množine odstránené.

Statistical outlier removal

Ďalším spôsobom je odstránenie bodov, ktorých priemerná vzdialenosť od K najbližších bodov d_p je vyššia alebo nižšia od priemernej vzdialenosti μ aspoň o nejaký násobok smerodajnej odchylky celku δ. Point cloud P^* s odstránenými outliermi z pôvodného point cloudu

P dostaneme nasledovne:

$$P^* = \{p \in P | (\mu - \alpha\delta) \leq d_p \leq (\mu + \alpha\delta)\}$$

Narozdiel od radius outlier removal metódy netreba vyberať vhodný polomer okolia podľa hustoty bodov.

2.1.2 Korešpondencie

Ak sú známe (aspoň tri) korešpondujúce dvojice bodov problém správneho zarovnania snímkov je triviálny. Pri skenovaní objektov z rôznych polôh je však nejasné ktoré body spolu korešpondujú a zároveň snímky nemusia obsahovať rovnaké body. Nájsť korešpondujúce body je možné odhadom ak sa snímky nelíšia signifikantne alebo pomocou rôznych čŕt bodov. Tieto črty môžu byť napríklad normály, farba alebo PFH - point feature histogram ktorého podrobný popis je zhrnutý v práci *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments* [8, p. 50-57], ktorý vyjadruje blízke okolie bodu. Po vybratí metódy sa môžu hľadať dvojice napríklad bucket sortom.

2.1.3 Podvzorkovanie

Pri zarovnávaní dvoch pointcloudov A, B je $|A| \times |B|$ možných dvojíc korešpondujúcich bodov na zváženie. Vo veľkých pointcloudoch pochádzajúcich zo skenovacieho procesu reálnych objektov v našom prípade býva rádovo 10^4 až 10^5 bodov. Keďže na správne zarovnanie potrebujeme len 3 korešpondujúce dvojice môžeme pointcloudy podvzorkovať a výrazne tak urýchliť registračný proces.

Podvzorkovanie voxelovou mriežkou

Pomocou 3D mriežky voxelov je možné znížiť počet bodov v pointcloude a vyrovnať jeho hustotu. Body prislúchajúce rovnakému voxelu v mriežke sú odstránené a nahradené jedným so súradnicami ich ťažiska. Nevýhodou tohoto prístupu je vytváranie chyby až do $\frac{1}{2}$ dĺžky najdlhšej uhlopriečky voxelu.

Podvzorkovanie vyhadzovaním bodov

Ďalšou možnosťou ako znížiť počet bodov v pointcloude je prostým vyhadzovaním bodov. Spomenieme tri spôsoby ako tieto body vyberať:

1. Vyhodiť každý n-tý bod pointcloudu

2. Náhodne vybrať a vyhodiť body

3. Vyhodiť body patriace rovnakému voxelu okrem bodu, ktorý leží najbližšie k stredu voxelu.

Pre spôsob 1 je potreba pointcloud najskôr vhodne zoradiť aby bolo odstránenie rovnomerné napríklad zlúčením ich súradníc. V prípade 2 je výsledok nedeterministický a pre malé pointcloudy nevhodný.V treťom prípade sa budú vytvárané rovnobežné segmenty, čo môže vytvárať lokálne minimá pre registračný proces.

2.1.4 RANSAC

RANSAC(Random sample and consensus) je gradientová iteratívna metóda pre odhad modelu z množiny, ktorá obsahuje outlierov. RANSAC metóda pozostáva z dvoch krokov:

1. Vyber náhodnú minimálnu podmnožinu G (ktorá ešte vybratá nebola) celku S ktorá postačuje na určenie parametrov modelu(teda ak chceme modelovať čiaru treba dva rôzne body)

2. Nájdi množinu CS_i (consensus) bodov ktoré tomuto modelu zodpovedajú(s niakou povolenou chybovou vzdialenosťou)

Výstupom RANSACu je množina CS_n s najviac prvkami. Je ľahko vidieť, že $|CS_i| \leq |CS_{i+1}|$a teda postupnou iteráciou sa model blíži k najlepšiemu možnému. Nevýhodou tohto postupu je jeho časová zložitosť pre nájdenie optimálneho modelu $O\binom{|S|}{|G|}$. Výhodou je verzatilita (náhodný prístup výberu nie je citlivý na typ vstupu) a možnosť ukončiť hľadanie a zobrať najlepší doteraz nájdený model. Pre detailnejší popis a ukážkovú implementáciu v Matlabe odporúčame dokument od Marca Zulianiho[6]

2.1.5 Iterative closest point

ICP

Iterative closest point(ICP) je iteratívna zostupná metóda pre nájdenie optimálnej transformácie medzi dvoma neorganizovanými pointcloudmi minimalizáciou euklidovskéj vzdialenosti ich prekryvu. ICP berie páry p_i, q_i najbližších bodov zarovnávaných pointcloudov P a Q ako korešpondencie a predpokladá, že každy bod korešponduje k nejakému inému. Chyba medzi pointcloudmi je vyjadrená ako

$$e = \sum_{i=1}^{n} ||p_i^T - q_i||^2$$

kde p_i^T je bod p_i po transformovaní pointcloudu P transformáciou T . Z týchto korešpondencii potom ICP odhadne transformáciu (napríklad pomocou SVD - Singular value dekompozíciou), aplikuje ju na daný pointcloud a tento proces opakuje kým chyba e neklesne pod požadovanú hodnotu. Keďže ICP ako korešpondencie berie blízke body potrebuje aby na začiatku boli pointcloudy pomerne dobre zarovnané.

Color ICP

Color ICP je varianta ICP algoritmu, ktorá pri výpočte chyby používa eukidovskú vzdialenosť zložiek farby bodov. Týmto postupom je možné registrovať páry pointcloudov, ktoré nemajú výrazné črty (napríklad steny budov alebo podlaha) alebo sú príliš pravidelné (napríklad guľa). Nevýhodou tohto postupu je náročnosť nastavenia kritéii konvergencie, keďže sa pridali 3 ďalšie rozmery, nekonzistentonosť farieb pri skenovaní objektov a odlesky na povrchu objektu. Tento registračný algoritmus je skúmaný v papieri *Pair-wise Registration of 3D/Color Data Sets with ICP*[7].

2.2 Rekonštrukcia

Rekonštrukcia je proces pri ktorom sa z bodov vytvorí povrch najčastejšie tvorený trojuholníkmi alebo viacuholníkmi. Spojením a vyplnením viacerých takýchto mnohouholníkov hranami vzniká aproximovaný povrch daného objektu. V neusporiadanom pointcloude je treba zistiť ktoré body pospájať do takéhoto útvaru, prípadne aj pridať body pre hladší povrch.

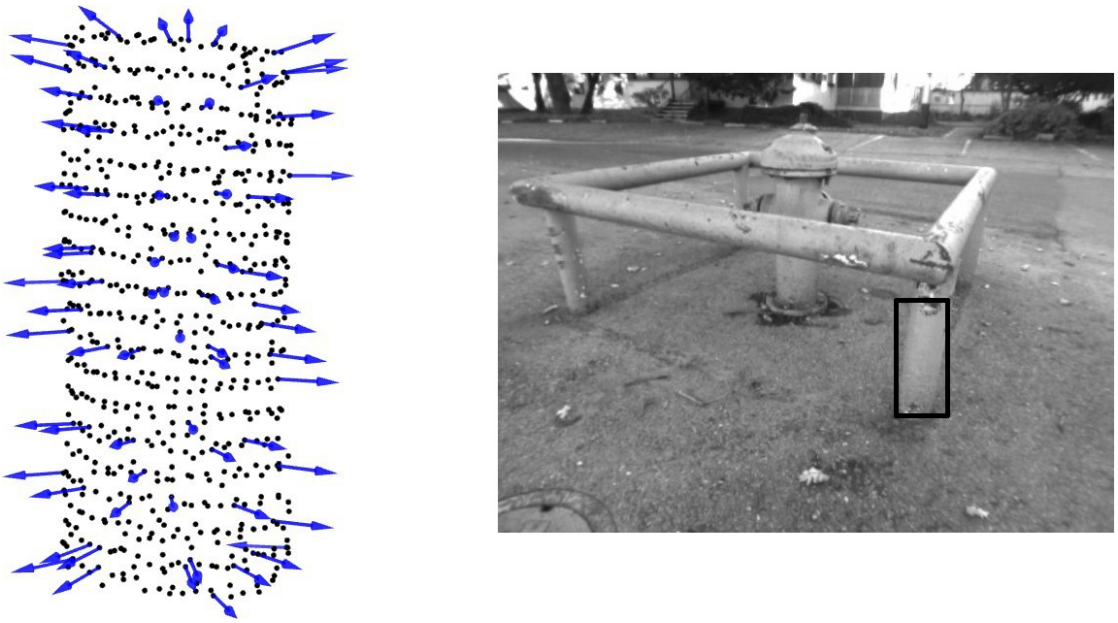
2.2.1 Odhad Povrchových normál

Keďže sa snímaním povrchu kamerou s obmedzeným rozlíšením stráca informácia je povrch aproximovaný pointcloudom P nejednoznačný, preto treba normály v bodoch odhadnúť podľa ich k najbližších bodov, množinu týchto bodov označíme P^k.
Normála $\bar{n}$ v bode p je aproximovaná vlastným vektorom prisluchajúcim k najmenšiemu vlastnému číslu λ_0 kovariantnej matice C:

$$C = \frac{1}{k}\sum_{i=1}^{k}(p_i - \bar{p})(p_i - \bar{p})^T$$

kde $\bar{p} = \frac{1}{k}\sum_{i=1}^{k} p_i$ je ťažisko P^k , $p_i \in P^k$ [8, p. 45-46]

Poznámka Keďže je matica C symetrická jej vlastné čísla a vektory sú realne, teda pri implementácii netreba riešiť komplexné čísla.



Obrázok 2.1: Naznačenie normál pre označený objekt na fotke. Šípky predstavujú smer normály. Zdroj[8, p. 200]

2.2.2 Nadvzorkovanie

Pod nadvzorkovaním rozumieme proces pri ktorom zvyšujeme hustotu, teda rozlíšenie pointcloudu. Cieľom tohto procesu je vyhladenie povrchu ktorý z pointcloudu vznikne rekonštrukčným procesom. Toto vyhladenie nemusí vždy zlepšiť kvalitu alebo realistickosť reprezentácie keďže vyhladzuje aj ostré hrany (napríklad hrany stola) ktoré sú chcené teda tu je riziko straty detailov.

Moving least squares

Moving least squares(MLS) je metódou ktorá má za úlohu vytvoriť spojitý povrch z diskrétnych dát akými je pointcloud. Toto dosahuje odhadom rádu m polynómu charakterizújuceho povrch v bode $x \in P$ minimalizáciou chyby

$$e = \sum_{i=1}^{I} (p(x_i) - f(x_i))^2 \theta(||x - x_i||)$$

kde θ je váhová funkcia a $x_i : i = 1..I$ sú najbližšie body k x. Rád m je potom hodnota kde $p^*(x)p^* \in \Phi_m$ minimalizuje túto chybu oproti ostatným $p \in \Phi_m$. Pointcloud je potom premietnutý na povrch určený týmto procesom a body ktoré sú vzdialené sú zahodené. Po tomto je možné k povrchu pridať ďalšie body nasledovnými metódami:

- Voxel grid dilatation - vytvorí sa mriežka voxelov, ktorá sa dilatuje aby zodpovedala MLS povrchu a pridajú sa body najbližsie k povrchu. Výhodou je, že zaplňuje diery a

pridané body sú uniformne roztriedené a teda je zmena hustoty konštantná.

- Random uniform density - body sú pridávané na MLS povrch náhodne pričom je ale zachovávaná hustota.
- Sample local plane - rovina v okolí bodu bude vzorkovaná v nejakom jeho okolí.

Obrázok 2.2: Rekonštrukcia pomocou greedy triangulácie(popísaná nižšie). Trojuholníky majú medzi sebou veľké uhly.

Obrázok 2.3: Rekonštrukcia pomocou greedy triangulácie(popísaná nižšie) s použitím MLS - random uniform density pre vyhladenie. Vyhladzovacím procesom sa uhly medzi trojuholníkmi otupili a povrch výrazne zjemnel.

Bilateral upsampling

Tento postup vyžaduje usporiadaný pointcloud P napríklad podľa X, Y súradníc s farbou. Podľa farby potom dopĺňa informáciu o hĺbke (ktorá v niektorých bodoch nemusí byť definovaná) nasledovne:

$$S_p = \frac{1}{k_p} \sum_{q_d \in \Omega} S_{q_d} f(||p_d - q_d||) g(||I_p - I_q||))$$

kde S je pointcloud hĺbky, I je farebný obraz a S_p pointcloud s rozlíšením I a doplnenou hĺbkou. Tento postup je vhodný ak sa pri skenovaní nezhoduje rozlíšenie hĺbkového senzora s rozlíšením kamery. Viac o tomto postupe popisuje práca *Joint bilateral upsampling*[9]

2.2.3 Polygon reconstruction

Greedy triangulácia

Tento postup vytvára na základe normál bodov pointcloudu lokálne trojuholníky v nejakom okolí bodu p nasledovne:

1. Pre bod p vyber k nespracovaných bodov z jeho okolia s polomerom $r = \mu d_0$ kde d_0 je vzdialenosť určená podľa lokálnej hustoty a μ konštanta.

2. Tieto body sú projektované na rovinu kolmú na ich normály a body ktoré sú pod touto rovinou (t.j. sú medzi ťažiskom objektu a rovinou) sú zahodené, ostatné sú s bodom p a najbližším susedom, takže tvoria trojuholníky.

3. Za bod p vyber ďalší nespracovaný bod.

Výhodou greedy triangulácie projekcie je najmä jej rýchlosť, avšak pre dobré výsledky vyžaduje nečlenitý povrch s dostatočne hustými prechodmi. Podrobnejší opis poskytuje práca *On Fast Surface Reconstruction Methods for Large and Noisy Point Clouds*[10]

Poisson

Poisson rekonštrukcia má za cieľ vybudovať vodotesný (watertight teda bez dier) aproximovaný povrch objektu z pointcloudu. Využíva na to izoplochu objektu (jeho kontúru) a vektorové pole $\vec{V}$ čo umožňuje vyššiu presnosť ako len s použitím bodov. Spojitá, uzavretá izoplocha je upravená vektorovým poľom s danou hĺbkou D a tak vzniká chcený povrch. Tento prístup je škálovateľný hĺbkou D od ktorej závisí rozlíšenie výsledného objektu. Čo sa týka časovej a pamäťovej zložitosti je porovnateľný s inými prístupmi s kvalitatívne podobným výsledkom a obe zložitosti rastiú približne lineárne (faktor $x4$) s narastajúcou hĺbkou. Kompletné vysvetlenie a vlastnosti Poisson rekonštrukcie možno nájsť v papieri Poisson surface reconstruction[11].

Marching cubes

Algoritmus marching cubes rozdelí scénu do mriežky voxelov s jednotnou hranou, čím vznikne množina kociek. Algoritmus má preddefinovanú množinu konfigurácii mnohouholníkov v jednej kocke. Do tejto mriežky sa projektuje pointcloud a marching cubes prechádza kocky a určí ich konfigurácie na základe susedných kociek, normál a polôh bodov v nej. [13, 12]

Kapitola 3

Implementácia

Táto kapitola popisuje spôsob implementácie výslednéj aplikácie, použité knižnice a prostredie. Taktiež opíšeme spôsob použitia programu a jeho nastavenia.

3.1 Prostredie a knižnice

Jazyk

Keďže je časová zložitosť mnohých algoritmov vysoká bol zvolený jazyk C++ ktorý sa sa vyznačuje rýchlosťou a keďže je objektovo orientovaný dovoľuje zostrojovať komplexné štruktúry. Ďalej je jedným z najrozšírenejších čo prináša aplikácii multiplatformovosť.

Point cloud library

Point cloud library (PCL) je najdôležitejšou knižnicou projektu. Staviame na jeho implementáciach algoritmov spomínaných v predošlej kapitole. PCL[14] je veľký kolaboračný otvorený projekt zaoberajúci sa spracovávaním pointcloudov, 2D a 3D obrazov. Obsahuje optimalizované algoritmy pre filtráciu, registráciu, segmentáciu[15], rekonštrukciu povrchov a odhad čŕt. Knižnica PCL bola vybraná kvôli jej všestrannosti vďaka ktorej netreba konvertovať údaje do rôznych formátov a natívnej podpore OpenNI 3D interfaceu[16]. Vďaka OpenNI interfaceu je aplikácia využívať na získanie údajov nielen Kinect ale aj iné skenovacie zariadenia, medzi inými ASUS Xtion alebo SICK LMS400[17] ktoré sa od Kinectu líšia iba kozmeticky (ASUS Xtion) alebo aj základným princípom (napríklad SICK LMS400 je vysokoprecízny 2D skener[18]). Pre optimalizácoiu paralelizovateľných algoritmov používa knižnicu OpenMP[21] ktorá dodáva API pre paralelné programovanie so zdieľanou pamäťou. Projekt PCL je financovaný mnohými spoločnosťami z ktorých môžme spomenúť napríklad Google, Toyota, nVidia[19] a bol ocenený hlavnou cenou Open Source Software World Challenge 2011[22].

Visualization toolkit

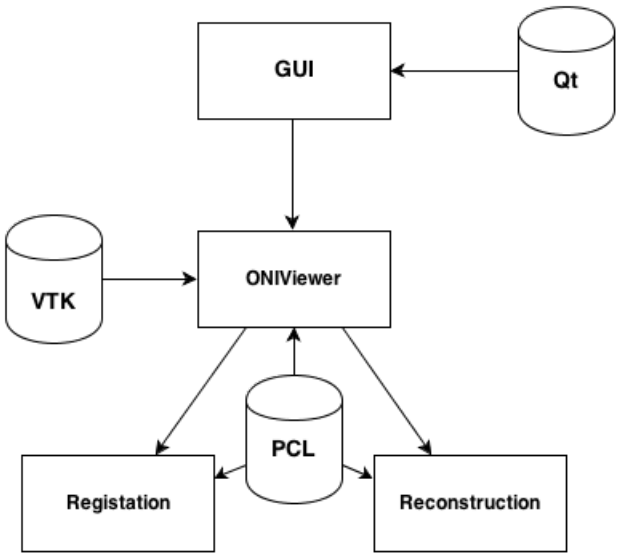
Na zobrazovanie výsledkov sa využíva otvorená knižnica Visualization toolkit(VTK). PCL je na nej závislé keďže pri niektorých algoritmoch priamo využíva ich implementácie vo VTK. Ďalej obsahuje metódy pre ukladanie obrázkov,objektov alebo pointcloudov do súborov v štandardizovaných formátoch akými sú Stanford PLY[23], STL[24] a Wavefront OBJ[25].

Qt

Interface aplikácie je riešený pomocou Qt[26] frameworku ktorý poskytuje widget toolkit na tvorbu GUI. Súčasťou Qt je Qt Creator ktorým je možné vytvárať rozloženie grafického prostredia intuitívnym sposobom.

3.2 Popis tried aplikácie

Aplikácia je tvorená štyrmi triedami: GUI, ONIViewer, Registration a Reconstruction ktorých relácie sú nasledovné:



Obrázok 3.1: Relačný diagram aplikácie.

GUI

Trieda GUI má na starosti interface aplikácie a poskytuje užívateľovi jednoducho nastavovať parametre rekonštrukcie a slúži aj na kontrolu kamery. Tvorí ju XML súbor rozloženia prvkov interfaceu a objekt MainWindow predstavujúci samotné okno programu.

ONIViewer

Táto trieda má za úlohu manažment zobrazovania kamery, poskytuje výsledky snímania triedam Reconstruction a Registration a zobrazuje výsledky ich činnosti. Na zobrazenie využíva dve okná vizualizácie - jedno pre polygónové objekty a druhé pre rýchle zobrazovanie pointcloudov. Prijíma pokyny od GUI.

Registration

Obsahuje metódy pre registráciu pointcloudov získaných z Kinectu pomocou OpenNI a vyprodukuje ich zarovnané spojenie nasledovnými krokmi:

1. Z každého pointcloudu odstráň outlierov. Outlierov odstraňujeme algoritmom Statistical outlier removal popísaným v časti 2.1.1
2. Podvzorkuj pointcloud vyhodením náhodných bodov a pomocou ICP algoritmu implementovanom v PCL ktorý je podporený RANSACom vypočítaj správnu transformáciu k predošlému pointcloudu.
3. Uprav výsledok zložený zo všetkých predošlích vypočítanou transformáciou a pridaj k nemu terajší pointcloud.

Tento proces sa spolieha na malé odchylky medzi jednotlivými snímkami, preto treba snímať skenovacím zariadením plynulo a s malými zmenami v polohe medzi dvoma vstupmi.

Reconstruction

Táto trieda spracúva pointcloudy a vytvára ich povrch. Rovnako ako Registration využiva algoritmy popísané v predošlej kapitole. Postup tejto rekonštrukcie je takýto:

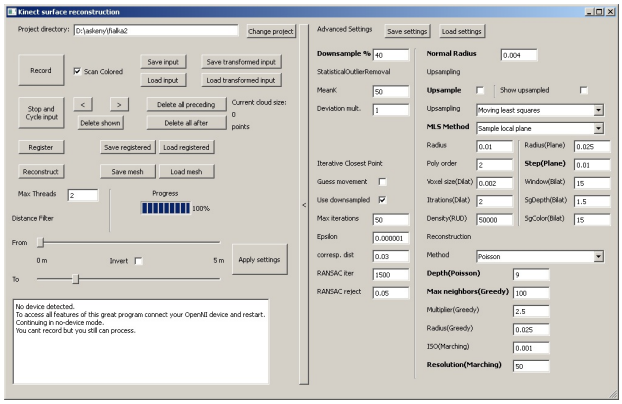
1. Nadvzorkuj vstupný pointcloud vybraným algoritmom pre hladší povrch.
2. Vypočítaj povrchové normály.
3. Zostroj povrch za pomoci vypočítaných normál.

3.3 Ovládanie

GUI

Interface programu sa delí na dve časti - hlavné ovládanie a pokročilé nastavenia ktoré sa dajú rozvynúť stlačením príslušného tlačidla na boku.

V základnéj časti sú dva posuvníky pre obmedzenie vstupu kamery podľa hĺbky s možnosťou tento výber invertovať využitie tohto obmedzenia popíšeme neskôr v časti zaoberajúcou sa skenovacími postupmi. Z nastavení sa v tejto časti nachádza ešte možnosť vypnúť farby aby boli výraznejšie viditeľné body tmavej farby a nastavenie maximálneho počtu vlákien ktoré bude program využívať v časovo zložitých algoritmoch ako je napríklad výpočet normál. Na ovládanie celého procesu slúžia tlačidlá *Record* ktoré spustí nahrávanie dát zo skeneru, *Register* zarovná vstup do jedného pointcloudu a *Reconstruct* z neho vytvorí polygónový model. Ku každému z týchto krokov prislúcha aj dvojica tlačidiel Save a Load ktorá ich výsledky uloží prípadne nažíta z projektovéj zložky. Po registrácii je možné uložiť aj vstup transformovaný vypočítanými transformáciami. Nahraný vstup je možné prechádzať automaticky tlačidlom *Stop and cycle input* alebo ručne pomocou $<$ a $>$ a mazať nechcené obrazy po jednom *Delete shown* alebo množstevne *Delet all preceding*, *Delete all after*. Na sledovanie postupu operácii slúži indikátor priebehu, ukazovateľ veľkosti zobrazeného pointcloudu a textové okno dávajúce používateľovi rôzne informácie.
V druhej časti sa nachádza možnosť uloženia a načítania nastavení a samotné nastavenia rekonštrukčných algoritmov. Program je prednastavený na pomerne rýchle základné nastavenie podávajúce dobré výsledky.



Obrázok 3.2: Interface programu s rozbalenými nastaveniami napravo.

Nastavenia

Tu si popíšeme účinky jednotlivých nastavení na proces spracovania. Vďaka tímto nastaveniam program môže slúžiť aj na porovnávanie dôležitosti jednotlivých nastavení na kvalitu a výkon algoritmov. V interfacei sú nastavenia s najvyššou váhou označené hrubo.

Ľavý stĺpec - registrácia

Downsample % - koľko percent bodov má ostať po podvzorkovaní pred registráciou

MeanK - priemer vzdialenosti od koľkých najbližších bodov má zohľadňovať algoritmus odstránenia outlierov

Deviation mult. - Násobok smerodajnej odchylky α. Body ktoré sa budú od priemeru líšiť o viac ako α násobok odchylky budú považované za outlierov a odstránené
Guess movement - Odhadni pri registrácii na základe predošlého pohybu pohyb ďalší. Môže urýchliť a zlepšiť registráciu pri plynulých pohyboch, pri neplynulých môže pôsobiť kontraproduktívne.
Use downsampled - Výsledok registrácie bude pozostávať z podvzorkovaných snímok. Pri vysokej frekvencii snímok ktoré dodáva Kinect je výsledok dostatočne hustý aj bez odstránených bodov.
Max iterations - nastaví hranicu iterácii pre ICP algoritmus
Epsilon - nastaví kritérium konvergencie pre ICP, ak je chyba ICP menšia ako táto hodnota ICP skončí
corresp. dist - ako približne sú vzdialené korešpondencie snímok v metroch, pre úspešné nájdenie transformácie ak sa skenujú vzdialené veľké objekty je vhodné túto hodnotu zvôčšiť
RANSAC iter - nastaví hranicu iterácii pre RANSAC algoritmus
RANSAC reject - maximálna vzdialenosť bodu od modelu aby bol do neho prijatý v metroch

Pravý stĺpec - rekonštrukcia

Normal radius - určí polomer okolia pre výpočet normál
Upsample - vypína a zapína nadvzorkovanie
Show upsampled - určuje či sa po zostrojení povrchu má zobraziť nadvzorkovaný pointcloud z ktorého bol vytvorený, nadvzorkované pointcloudy sú značne veľké(milióny bodov) a ich zobrazovanie zaťažuje procesor
Upsampling - vyberie algoritmus pre nadvzorkovanie
MLS Method - vyberie metódu pri MLS algoritme
Radius - nastaví polomer z ktorého sa vyberajú najbližší susedia
Poly order - stupeň polynómu pre MLS viac ako 2 nie je pre vhodný pre ľudí keďže závisí od neviditeľných vlastností ako popisuje [27, p.6-7]
Voxel size(Dilat) - hrana voxelu pre metódu dilatácie voxelovej mriežky
Iterations (Dilat) - počet iterácii dilatácie voxelovej mriežky
Density(RUD) - hustota pre MLS Random uniform density
Radius(Plane) - okolie bodu pre MLS Sample local plane
Step(Plane) - krok MLS Sample local plane v metroch, menšie kroky znamenajú hustejšie nadvzorkovanie
Window(Bilat) - nastaví rozmer okolia pre bilaterálne nadvzorkovanie
SgDepth(Bilat) - smerodajná odchylka σ_depth pre bilaterálne nadvzorkovanie
SgColor(Bilat) - smerodajná odchylka σ_color pre bilaterálne nadvzorkovanie
Method - vyberie metódu rekonštrukcie povrchu

Depth(Poisson) - nastaví hĺbku vektorového poľa pre poisson rekonštrukciu
Max.neighbors(Greedy) - určuje koľko maximálne susedov sa spojí v jednom cykle greedy triangulácie
Multiplier(Greedy) - nastaví váhu pre zmenu
Radius(Greedy) - nastaví polomerovú konštantu μ pre greedy triangluláciu
ISO(Marching) - nastaví minimálnu potrebnú hodnotu skalára voxelu aby bol považovaný za voxel nachádzajúci sa v povrchu
Resolution(Marching) - nastaví hranu mriežky pre Marching cubes algoritmus, udávané v počte kociek

3.4 Skenovanie Kinectom

Na nasnímanie scény Kinectom môžme využiť dva prístupy: hýbať Kinectom okolo objektu alebo hýbať snímanou scénou. V prvom prípade predstavuje požiadavka na minimálnu vzdialenosť značnú nepríjemnosť pri ručnom skenovaní keďže je treba udržovať stály odstup a dostatočne konzistentný záber skenovaného objektu, tento prístup je vhodný najmä pri väčších objektoch a scénach s ktorými je ťažké hýbať. Druhý pripad je realizovateľný vyfiltrovaním získaného obrazu tak, že sa zachytáva len cielený objekt. Pri tomto postupe je vhodné využiť slidery v programe tak aby pozadie zaniklo. Po tom čo dosiahneme záber len na daný objekt ho môžeme postupne otáčať a získať tak dostatok údajov na jeho rekonštrukciu. Jednoduchým spôsobom ako ním hýbať je umiestniť ho na pravidelný symetrický rotačný disk ktorý registrácii nebude spôsobovať veľké problémy.

Obrázok 3.3: Rotačný disk umiestnený na valcoch papiera.

Narozdiel od disku skenované objekty by mali byť nesymetrické aby sa ľahko registrovali. Problémovými sú napríklad guľové objekty(napr. pomaranč) alebo objekty valcovité (fľaše, misky poháre). Spôsob ako takéto objekty naskenovať je pridať k nim do scény pomocné ľahko zarovnateľné objekty ktoré budú slúžiť ako pomôcka pre ICP. Medzi takéto

objekty patria napríklad rôzne krabičky. Tieto pomocné objekty môžu aj väčšie ako cielený objekt.

Kapitola 4

Výsledky

V tejto kapitole zhrnieme výsledkyprogramu, popíšeme jeho správanie a určíme slabé miesta jeho fungovania.

4.1 Rýchlosť a pamäťová náročnosť

Registrácia

Rýchlosť spracovania závisí najmä od spracovávaných dát. S prednastavenými nastaveniami trvá registrácia stovky 20 000-30 000 bodových pointcloudov približne jednu minútu. S väčšími pochopiteľne registrácia spomaľuje a to výrazne preto je dobré aby sa v registračnom procese používali tieto veľké pointcloudy podvzorkované na 10-20 tisíc bodov.
Pamäťovo registrácia vyžaduje dostatok pamäte na uloženie vstupu čo činí pri náročnejších vstupoch zhruba 200MB.

Rekonštrukcia

Rekonštrukcia prebieha vo fázach nadvzorkovanie, výpočet normál a rekonštrukcia. Najnáročnejšia forma nadvzorkovania je MLS s použitím Sample local plane trvá pri najväčšom odporúčanom vstupe okolo 2 minút. Najväčší odporúčaný vstup je taký ktorého nadvzorkovaním nevznikne pointcloud s viac ako štyrmi miliónmi bodov pretože veľkým pointcloudom je zdĺhavé určiť normály. Určovanie normál trvá zhruba 5 minút (pointcloud o 2 miliónoch bodov). Prednastavená metóda rekonštrukcie je Poisson o hĺbke 9 a časovo zaberá 2 až 3 minúty. Spolu teda rekonštrukcia trvá okolo desiatich minút.
Oproti registrácii je rekonštrukcia výrazne náročnejšia na pamäť a v prípadoch spomínaných vyšši si žiada do 1GB dostupnej pamäte.

Uložené súbory

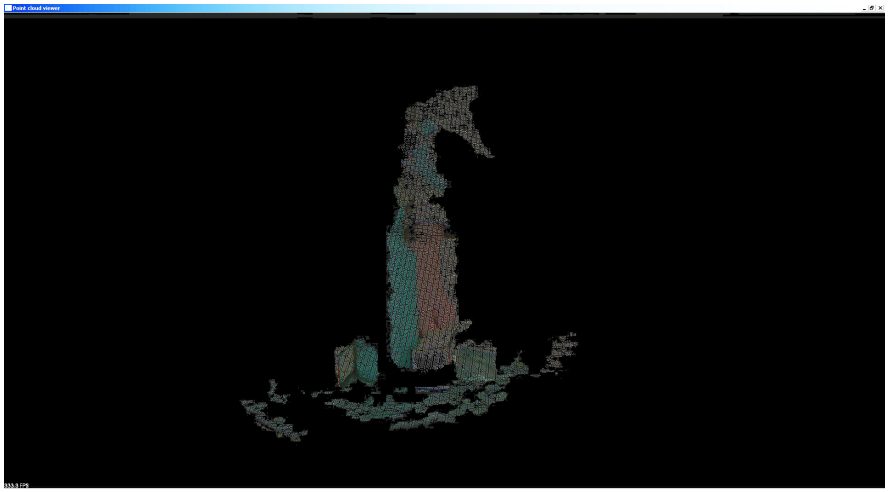
Súbory vstupu sú ukladané v binárnej forme pre čo najnižšiu veľkosť a mávajú do 2MB. Pointcloud vzniknutý z registrácie je ich podmnožinou a pri rozumnom počte záberov neprekračuje 30MB. Modely bývajú menšie od registrovaného cloudu.

Poznámka Časové údaje sú produktom počítaču s dvojjadrovým procesorom *AMD Athlon 64 x2 5600+* a operačným systémom *Microsoft Windows 7*

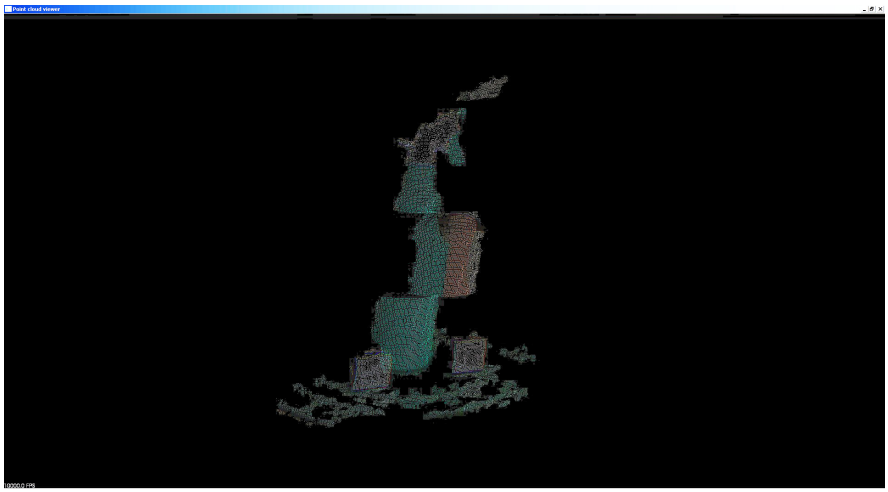
4.2 Slabiny

Šum a chyby Kinectu

Ako sme už spomenuli kinect je citlivý na slnečné svetlo preto dosahuje výsledky v uzavretých priestoroch. Okrem toho Kinect môže občas poskytne zlé údaje. Je možné, že je to chybou konkrétneho zariadenia, avšak presnú príčinu tohto javu sa určiť nepodarilo. Údaje môžu prísť narušené troma spôsobmi: S chybnou farbou čo znamená, že na správny hĺbkový obraz je premietnutá farba z inej časti scény, s chybou v súradniciach - kus scény je výrazne posunutý alebo ich kombináciou.



Obrázok 4.1: Chyba farby. Na fľaši sa projektuje ruka ktorá bola zhruba 30cm za ňou. Súradnicovo je objekt vporiadku teda ide pravdepodobne o chybu spojenia obrazu kamery a hĺbkového senzora.



Obrázok 4.2: Chyba v súradniciach. Stredná časť fľaše je posunutá, farba sedí podľa súradníc X a Y teda ide o chybu hĺbkového senzora.

Registrácia

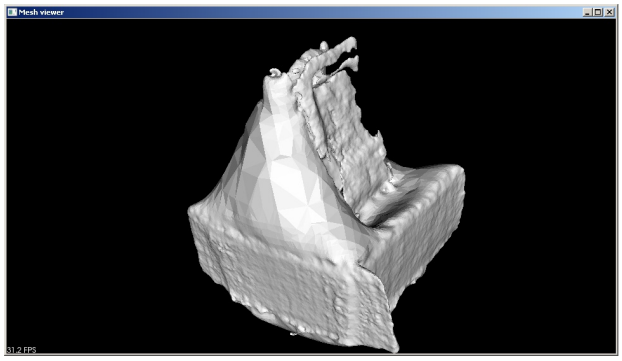
Registrácia je najväčšou slabinou programu. Pri postupnom zarovnávaní snímkov sa akumuluje chyba keďže dokonalé zarovnanie je prakticky len veľmi ťažko dosiahnuteľné. Pre dobrý výsledok registrácie je vhodné dodržiavať nasledovné:

1. Nevytvárať zbytočne veľa záberov - optimum sa ukázalo 60 záberov pre celkový 360° sken
2. Objekty menšie ako 30 cm potrebujú pomoc - pridaním krabice alebo iného pomocného objektu do skenovania sa výrazne zlepší výsledok
3. Nevytvárať rovnaké zábery mnohonásobne

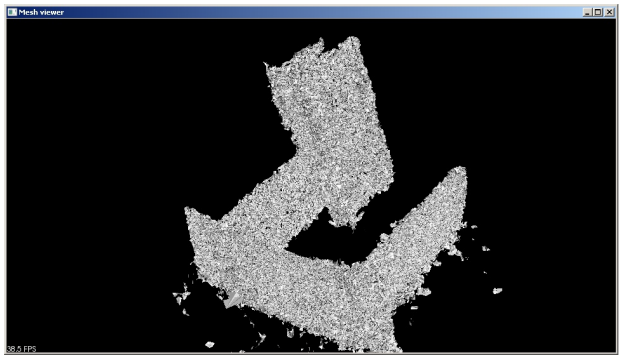
Rekonštrukcia

Každá z rekonštrukčných metód má svoje slabé miesta. Aj keď sa poisson rekonštrukcia ukázala najkvalitnejšou má problémy z dátami, ktoré obsahujú roztrúsené objekty. Keďže sa ich snaží pospájať aby bol model vodotesný niekedy ich spojí zle.

Obrázok 4.3: Výsledok registrácie bezdrôtového teplomeru. Vrchná stena pomocnej krabice nebola nasnímaná



Obrázok 4.4: Poisson rekonštrukcia bezdrôtového teplomeru. Metóda poisson spojila krabicu s teplomerom zle.

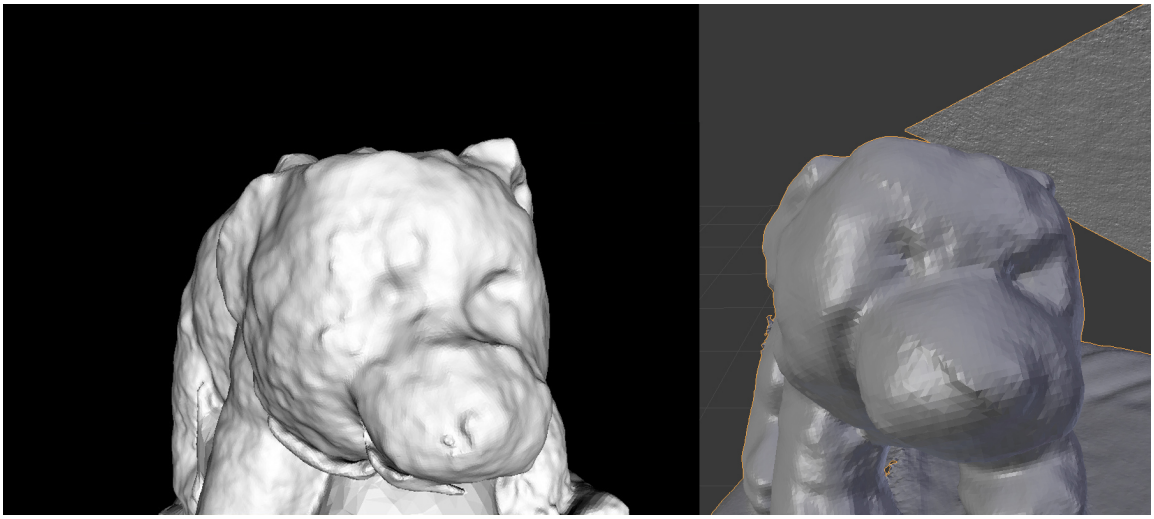


Obrázok 4.5: Rekonštrukcia pomocou greedy triangulácie. Teplomer aj krabica zostávajú oddelené.

Greedy triangulácia problém s nespojitým pointcloudom nemá, avšak je kvalitatívne už na prvý pohľad horšia.
A napokon metóda Marching cubes potrebuje byť špeciálne nastavená pre rôzne pointcloudy.

4.3 Porovnanie s KinectFusion

V prvej kapitole sme spomenuli rekonštrukčnú pipeline naimplementovanú v toolkite od Microsoftu - KinectFusion. KinectFusion využíva výkon grafickej karty a pri skenovaní zarovnáva už polygónové objekty povrchu a to v reálnom čase. Tento proces je bližšie popísaný na stránke jeho návodu[4]. Nevýhodou tohto prístupu je, že rozlíšenie obmedzuje pamäť grafickej karty a pre zvýšenie rozlíšenia je teda potrebné nadvzorkovať výstup alebo pridať hardware. Obrázok nižšie porovnáva objekt vytvorený KinectFusionom a našou implementáciou.



Obrázok 4.6: Porovnanie rekonštrukcie plyšového medveďa. Vľavo je výsledok pomocou našej implementácie bez vyhladenia, vpravo pomocou KinectFusion s najvyšším rozlíšením

4.4 Výsledné modely

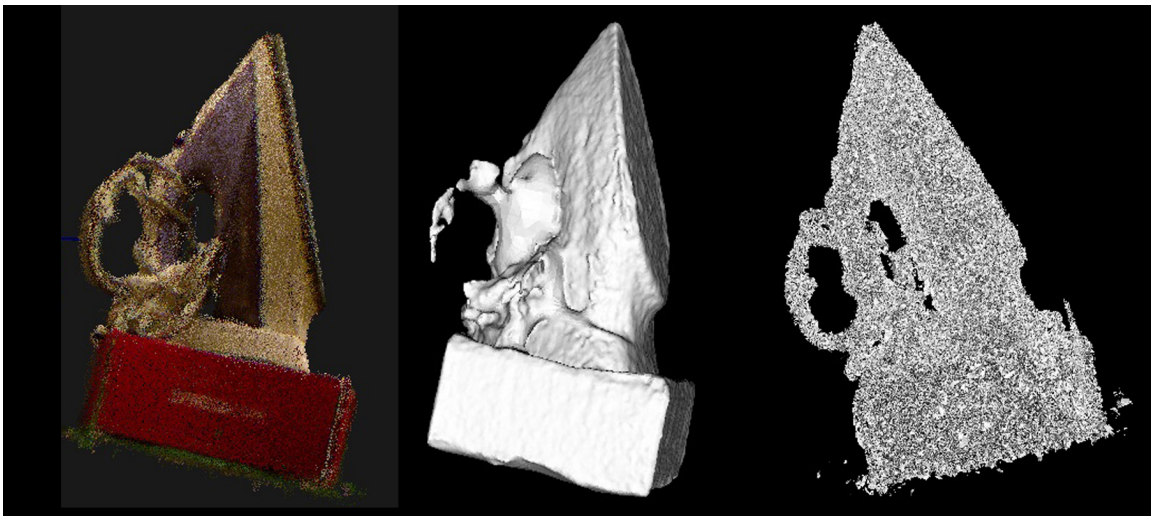
Žehlička

Na ukážku ako objekt na rekonštrukciu sme vybrali žehličku. Zaujímavé časti sú jej prívodová šnúra ktorá je tenká a jej rúčka, ktorá v objekte vytvára dieru.

Obrázok 4.7: Fotka žehličky.

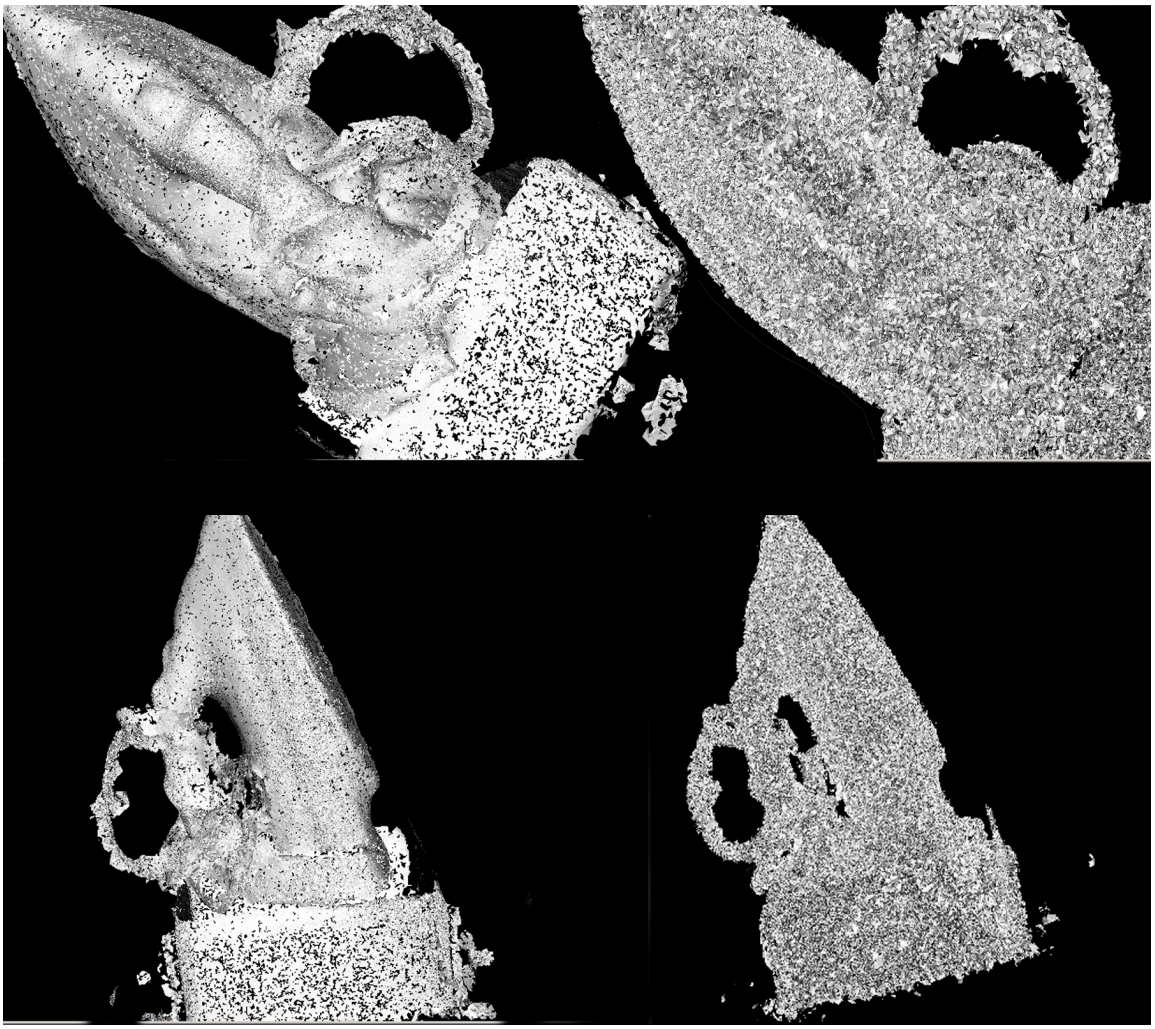
Obrázok 4.8: Naľavo sa nachádza výsledok registrácie žehličky, vpravo jej poisson rekonštrukcia.

Obrázok 4.9: Naľavo sa nachádza výsledok registrácie žehličky, vpravo jej poisson rekonštrukcia.



Obrázok 4.10: Vľavo je výsledok registrácie. V strede sa nachádza model vytvorený Poisson rekonštrukciou a vpravo model ktorý vznikol pomocou greedy triangulácie.

Na Poisson modeli je vidieť ako sa diera tvorená rúčkou nežiadúco zaplnila. Naviac prívodová šnúra v modeli chýba. Pri greedy triangulácii je rúčka aj prívodná šnúra vyobrazená správne, avšak kvalita tohto modelu je viditeľne nižsia. Pre skvalitnenie výsledku greedy triangulácie vyhladíme vstup pomocou algoritmu MLS s random uniform density.



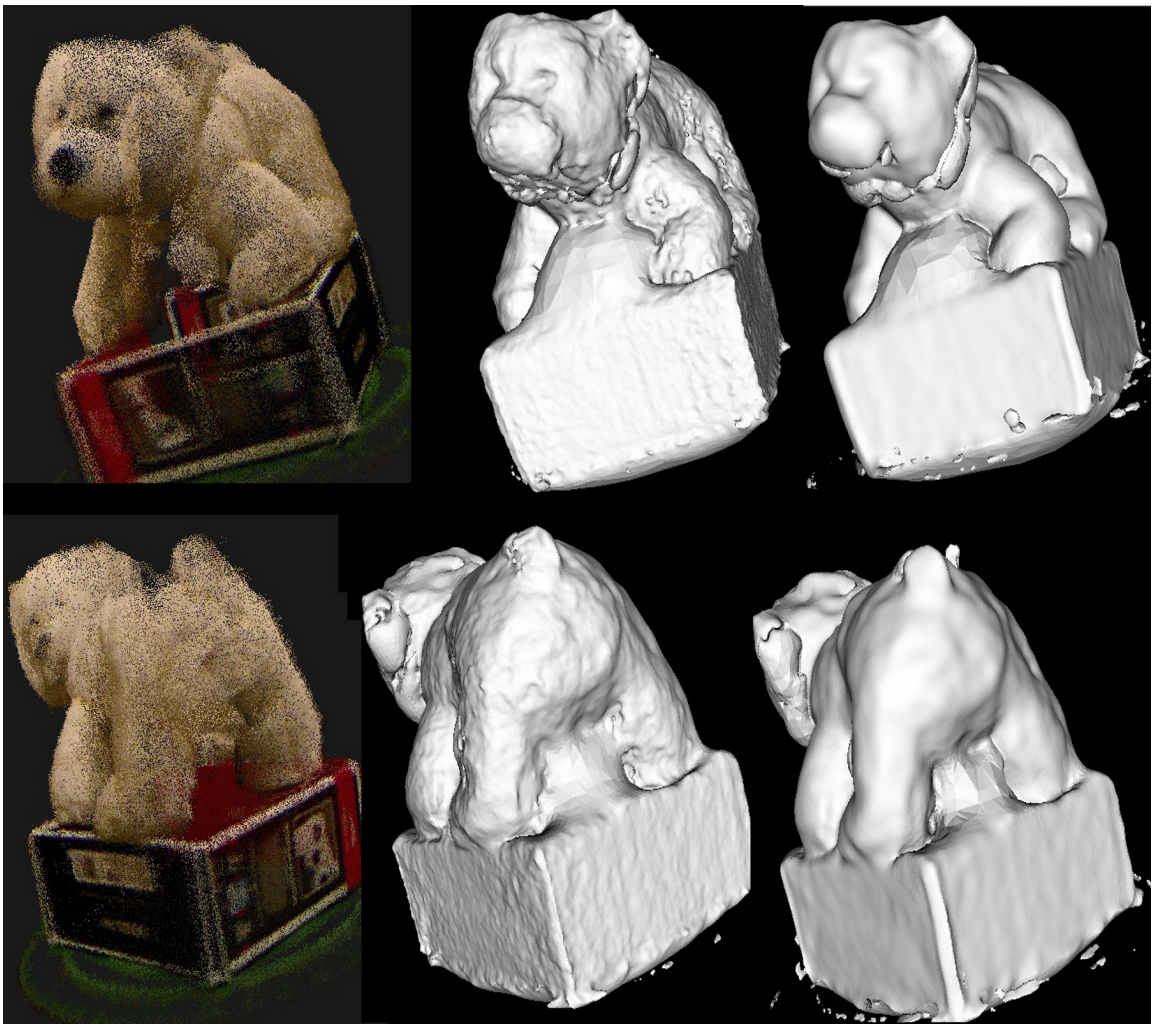
Obrázok 4.11: Naľavo sú rekonšrukcie greedy trianguláciou s vyhladením pomocou MLS - random uniform density, napravo bez nej.

Vyhladením vstupného pointcloudu sa kvalita výrazne zlepšila. Uhly medzi susednými trojuholníkami sú tupšie a teda je povrch hladší.

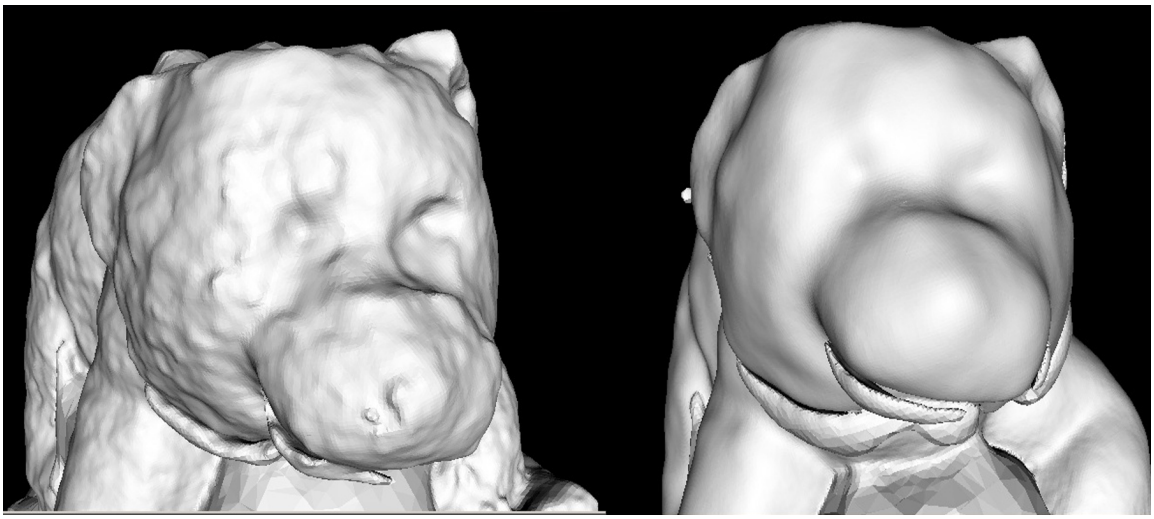
Plyšový medveď

Ďalším rekonštruovaným objektom je plyšový medveď ktorý má zhruba 0.5 cm srsť. Ukážeme na ňom vplyv vyhladenia na poisson rokonštrukciu.

Obrázok 4.12: Fotka medveďa.

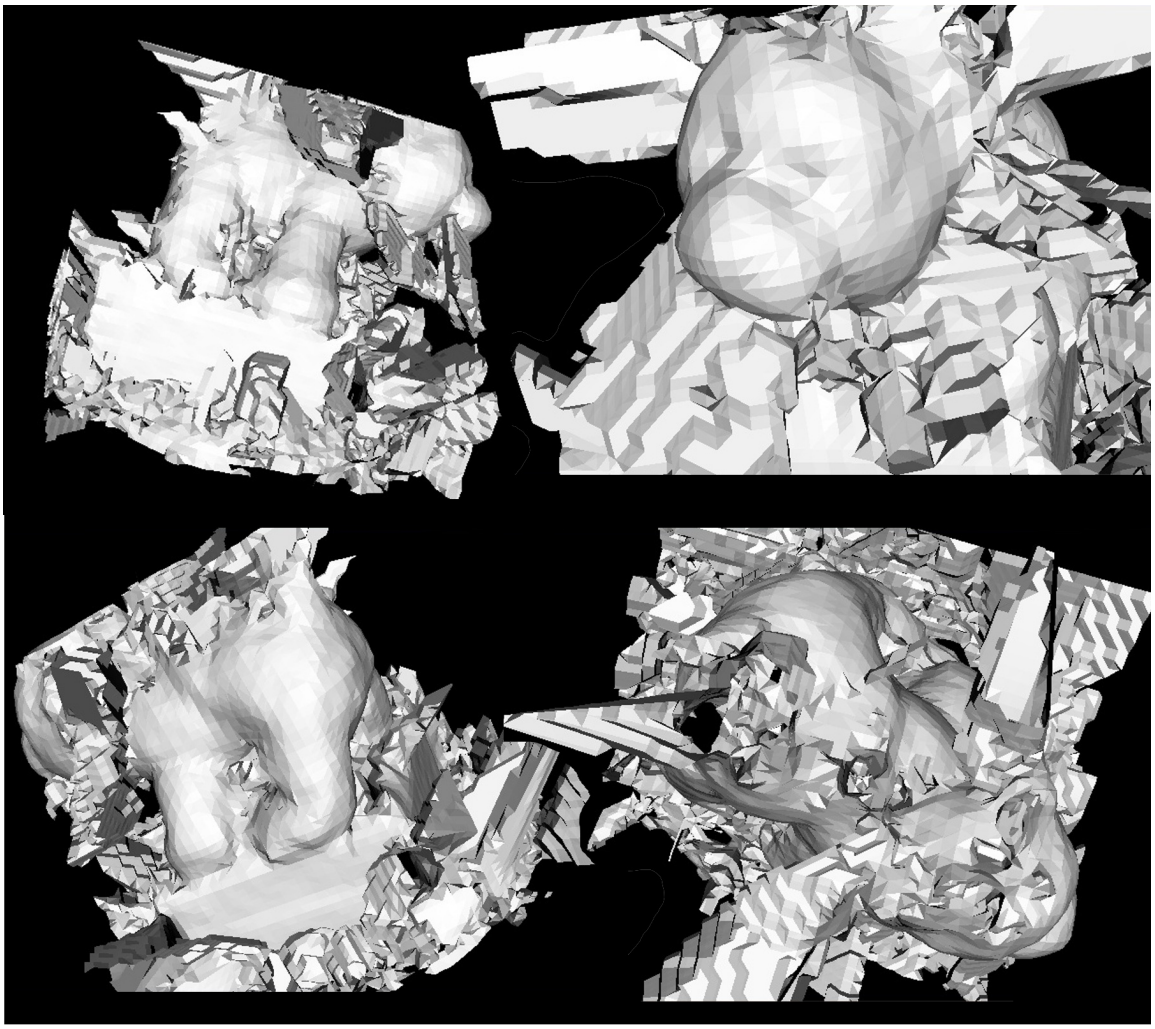


Obrázok 4.13: Napravo výsledok registrácie, v strede poisson rekonštrukcia, vľavo poisson rekonštrukcia s MLS - random uniform density vyhladením. Všimnime si, že poisson rekonštrukcia spojila medveďa s podložkou keďže vrchná stena podložky nebol nasnímaná.



Obrázok 4.14: Detail hlavy. Napravo model vytvorený poisson rekonštrukciou, vľavo s použitím MLS - random uniform density. Všimnime si, že bez vyhladenia vidno detaily ako sú oči ale po vyhladení strácame výraznosť týchto detailov.

Z výsledkov vidíme ako môže vyhladenie v správnej miere pomôcť. Ďalej si ukážeme výsledok Marching cubes algoritmu na sken medveďa. Keďže bol medveď oskenovaný z boku jeho chrbát a podložka nemajú vrchnú stranu presne definovanú. Toto algoritmu Marching cubes robí problém a vytvára z nedozavretých častí plochy rovnobežné s najbližším povrchom.



Obrázok 4.15: Rekonštrukcia pomocou Marching cubes s použitím nadvzorkovania MLS - random uniform density. Je jasne vidieť vytvorený šum tohto postupu. Na pravom spodnom modeli je možné vidieť ako sa šum tvorí a šíri z dier v modeli na medveďovom chrbte.

Záver

Práca vymenúva a popisuje pojmy algoritmy používane pri registrácii rekonštrukcii povrchov obbjektov z dát získaných 3D skenermi. Zameriava sa na konkrétny skener Microsoft Kinect a vysvetľuje jeho princípy a nedostatky.

Naimplementovali sme aplikáciu využívajúcu viacero pístupov, ktorá spracúva tieto údaje do modelov ktoré môžu byť následne uložené do bežného formátu. Táto aplikácia úspešne rekonštruuje objekty scény a poskytuje možnosť spájať zábery rovnakej scény do jedného celku. Táto aplikácia môže tiež slúžiť na porovnávanie výkonu jednotlivých algoritmov a vplyvu ich nastavenia na výstup. Automatické zarovnanie snímkov sa nám nepodarilo spraviť dokonale a pri úplnom 360° skene vyžaduje od užívateľa použitie správnych skenovacích techník, pri tvorení čiastočných modelov je však pomerne prívetivá. Výsledné modely splnili očakávania a ich kvalita zodpovedá kvalite vstupu, pričom je v aplikácii možnosť ich vyhladiť ktorá s čiastočným spomalením výrazne zvyšuje kvalitu.

V pokračovaní v tejto práci by bolo vhodné ukladanie výsledkov do viacerich formátov, ktoré podporujú aj textúry a samotné automatické otextúrovanie pomocou kamery zariadenia. Ďalej by bolo vhodné hlbšie preskúmať nové možnosti toolkitu Kinect for Windows a prípadne ho využiť na zlepšenie a zrýchlenie procesu.

Literatúra

[1] Kinect for Windows Sensor http://msdn.microsoft.com/en-us/library/hh855355.aspx

[2] Jeff Kramer, *Hacking kinect* (ISBN 1430238674)

[3] The latest Kinect for Windows SDK is here, (http://blogs.msdn.com/b/kinectforwindows/archive/2013/03/18/the-latest-kinect-for-windows-sdk-is-here.aspx)

[4] KinectFusion, (http://msdn.microsoft.com/en-us/library/dn188670.aspx)

[5] OpenGL transformation, (http://www.songho.ca/opengl/gl_transform.html)

[6] Marco Zuliani, *RANSAC for Dummies* (http://vision.ece.ucsb.edu/∼zuliani/Research/RANSAC/doc

[7] Lounis DOUADI, Marie-José ALDON, André CROSNIER, *Pair-wise Registration of 3D/Color Data Sets with ICP* (http://hal-lirmm.ccsd.cnrs.fr/docs/00/12/86/88/PDF/PID266727.pdf)

[8] Radu Bogdan Rusu, *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*, files.rbrusu.com/publications/RusuPhDThesis.pdf

[9] Johannes Kopf, Michael F. Cohen, Dani Lischinski, Matt Uyttendaele, *Joint bilateral upsampling* (http://johanneskopf.de/publications/jbu/)

[10] Zoltan Csaba Marton, Radu Bogdan Rusu, Michael Beetz, *On Fast Surface Reconstruction Methods for Large and Noisy Point Clouds* (http://ias.cs.tum.edu/_media/spezial/bib/marton09icra.pdf)

[11] Michael Kazhdan , Matthew Bolitho and Hugues Hoppe, *Poisson Surface Reconstruction* Symposium on Geometry Processing 2006, 61-70.

[12] James Sharman,*The Marching Cubes Algorithm*(http://www.exaflop.org/docs/marchcubes/)

[13] Hugues Hoppe Tony DeRose Tom Duchamp John McDonald Werner Stuetzle *Surface Reconstruction from Unorganized Points* (http://graphics.stanford.edu/courses/cs468-03-fall/Papers/Hoppe_SurfaceReconstruction.pdf)

[14] Point Cloud Library (http://pointclouds.org/)

[15] Allan Jepson *Image Segmentation* (http://www.cs.toronto.edu/∼jepson/csc2503/segmentation.pdf)

[16] Open Natural Interaction framework (http://www.openni.org/)

[17] PCL Module IO documentation (http://docs.pointclouds.org/trunk/group__io.html)

[18] SICK LMS 4xx series information (http://www.robotsinsearch.com/sites/default/files/produts/literature/LMS4xx/Product

[19] about PCL (http://pointclouds.org/about/)

[20] Visualization toolkit (http://www.vtk.org/)

[21] OpenMP (http://openmp.org/wp/)

[22] PCL wins the 2011 OSS World Challenge Grand Prize (http://pointclouds.org/news/2011/11/23/pcl-ossaward-2011/)

[23] Polygon File Format (http://paulbourke.net/dataformats/ply/)

[24] STereoLithography format (http://www.ennex.com/∼fabbers/StL.asp)

[25] Wavefront .obj format (http://www.martinreddy.net/gfx/3d/OBJ.spec)

[26] Stránka projektu Qt (http://qt.digia.com/About-us/)

[27] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, Claudio T. Silva *Computing and Rendering Point Set Surfaces* (http://www.sci.utah.edu/∼shachar/Publications/crpss.pdf)