VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

REKONSTRUKCE POVRCHU Z MRAČNA BODŮ

SURFACE RECONSTRUCTION FROM POINT CLOUDS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. STANISLAV KNOT**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MICHAL ŠPANĚL, Ph.D.**
SUPERVISOR

BRNO 2017

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2016/2017

Zadání diplomové práce

Řešitel: **Knot Stanislav, Bc.**
Obor: Počítačová grafika a multimédia
Téma: **Rekonstrukce povrchu z mračna bodů**
Surface Reconstruction from Point Clouds
Kategorie: Počítačová grafika

Pokyny:

1. Prostudujte základy počítačového vidění. Zaměřte se na problematiku zpracování tzv. mračna bodů (point clouds), které vznikají při 3D rekonstrukci z obrazu.
2. Prostudujte dostupné materiály na téma rekonstrukce povrchu z mračna bodů. Seznamte se s existujícími nástroji (např. knihovna PCL).
3. Vyberte vhodné metody a navrhněte nástroj pro rekonstrukci povrchu z mračna bodů.
4. Experimentujte s vaší implementací a případně navrhněte vlastní modifikace metod.
5. Porovnejte dosažené výsledky a diskutujte možnosti budoucího vývoje.
6. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Španěl Michal, Ing., Ph.D.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 24. května 2017

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Tato diplomová práce se zabývá problematikou zpracování mračen bodů sejmutých senzorem Kinect z jediné pozice. V rámci práce byla navržena aplikace, která za použití vybraných metod dokáže spojit takto získaná mračna a rekonstruovat povrch místnosti. Návrh pro zarovnávání překrývajících se snímků je založen na výpočtu klíčových bodů a jejich FPFH histogramů, ze kterých je vypočten odhad vzájemné korespondence. Tento odhad je poté doladěn a je provedeno filtrování redundantních bodů. Takto zarovnané a upravené mračno je rekonstruováno metodou Greedy Projection Triangulation. Veškeré výpočty probíhají offline. Výstupem aplikace je jak polygonální model, tak obrázek pro tvorbu textury. Za předpokladu správně nastavených parametrů jsou výsledky v přijatelné kvalitě pro tvorbu virtuálních prohlídek a jejich vizualizaci.

Abstract

This diploma thesis deals with the processing of point clouds captured by the Kinect sensor from single position. As part of this thesis an application was designed, which is able to register and reconstruct surface using selected methods. The registration of overlapping frames is based on computation of key points and their FPFH histograms from which the estimation of correspondence is computed. This estimation is then refined and redundant point filtering is performed. Surface is reconstructed from the registered and modified point cloud using Greedy Projection Triangulation. All computations are performed offline. The output of this application is textured polygonial model and an image for texture creation. With assumption of correctly set parameters the results are in a good quality for creation of virtual tours and visualization.

Klíčová slova

mračno bodů, zarovnávání mračen, Kinect, rekonstrukce povrchu

Keywords

point cloud, cloud registration, Kinect, surface reconstruction

Citace

KNOT, Stanislav. *Rekonstrukce povrchu z mračna bodů*. Brno, 2017. 59 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Španěl Michal.

Rekonstrukce povrchu z mračna bodů

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana doktora Španěla. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Stanislav Knot
17. května 2017

Poděkování

Děkuji Silvě za motivaci k dokončení této práce a Marušce za pomoc se snímáním pokojů. Dále bych chtěl poděkovat Ing. Michalu Španělovi, Ph.D. za rady při zpracování technické zprávy.

Obsah

Kapitola 1

Úvod

Cílem této práce je návrh a implementace nástroje používajícího metody pro rekonstrukci povrchu z mračna bodů za použití knihovny PCL. Jako zdroj mračna bodů mohou sloužit různé hloubkové snímače. Jedním z cenově dostupných a relativně přesných snímačů je Kinect od společnosti Microsoft. Tento fakt spolu s tím, že v mé bakalářské práci [9] jsem se věnoval detekci gest právě pomocí senzoru Kinect, zapříčinil, že jsem se rozhodl rozšířit své zkušenosti s tímto zařízením a používat jej jako hlavní vstup pro výslednou aplikaci.

Pro větší teoretické rozpětí práce jsem se rozhodl navrhnout postup, jakým spojit více nasnímaných překrývajících se mračen v jedno sjednocené, a to pak dále zpracovávat.

Oproti klasickému zarovnávání snímků algoritmem IPC jsem navrhl vylepšený postup. Za pomocí vypočtených FPFH histogramů [22] klíčových bodů dvojice snímků je proveden odhad vzájemné korespondence a ten je pak pouze doladěn. Při spojení vznikají duplicitní body, pro jejichž odstranění jsem navrhl použití filtru MLS. Takto upravené mračno je rekonstruováno metodou Greedy Projection Triangulation [16].

Tento výstup je vhodným kandidátem pro použití v architektuře, herním průmyslu a v neposlední řadě může sloužit jako zdroj dat pro pohyb autonomních robotů v interiéru. S rostoucí popularitou virtuální reality lze nalézt širší příklady využití 3D rekonstrukce vnitřních prostor. Vhodně nasnímaný a rekonstruovaný povrch může také sloužit jako vstup pro tisknutí 3D modelů.

Snímání mračen senzorem Kinect je omezeno na vnitřní prostory jeho technickým provedením. Z tohoto důvodu jsem se věnoval snímání, spojení a rekonstrukci povrchu místností. Není však problém aplikaci použít s daty získanými jiným typem senzoru či synteticky.

Výsledná aplikace dokáže úspěšně zarovnat n snímků za předpokladu správně nastavených parametrů použitých metod. Výstup je vhodný spíše pro vizualizaci.

Za účelem informovat čtenáře o mračnech bodů a postupech jejich zpracovávání včetně metod pro rekonstrukci polygonálního povrchu vzniká kapitola zabývající se teoretickým podkladem (kapitola 2). V praktické části práce (kapitola 4) získá čtenář přehled o tom, jaké konkrétní postupy jsem použil v návrhu já, jaké implementační prostředky byly použity k dosažení požadovaného výstupu. Na závěr jsou v této práci uvedeny experimenty s hotovou implementací, jejich výsledky a zhodnocení (kapitola 6).

Tato práce nijak nenavazuje na semestrální projekt.

Kapitola 2

Zpracování 3D dat ve formě mračna bodů

V této kapitole je popsáno mračno bodů, metody jeho ukládání a zpracovávání a teoretický základ těchto metod. Tyto metody lze roztřídit do základních skupin a těmi jsou filtrování a vyhlazení mračna bodů, spojení mračen bodů a rekonstrukce povrchu.

2.1 Mračno bodů a jeho získávání

Tak jako obyčejné kamery snímají obraz reálného světa ve dvou rozměrech, kterými jsou výška a šířka, hloubkové snímače přidávají třetí rozměr a to hloubku. Snímek pořízený obyčejnou kamerou je tedy matice o předem známých rozměrech, určených rozlišením snímacího čipu, kde každý pixel, prvek této matice, obsahuje informaci o barvě. U trojrozměrné kamery (hloubkového snímače) je to podobně. Je dána matice, která je naplněna informacemi o hloubce každého „pixelu“ tedy bodu v prostoru. Některé hloubkové snímače umožňují, mimo snímání hloubky, také snímání barevné informace. Takovéto snímače jsou pak označovány jako RGBD[1] snímače. Mezi tyto RGBD snímače patří i Kinect, který je vybrán pro tuto práci jako hlavní zdroj dat. Na obrázku 2.1 je zobrazeno ukázkové mračno bodů bez barevné informace.

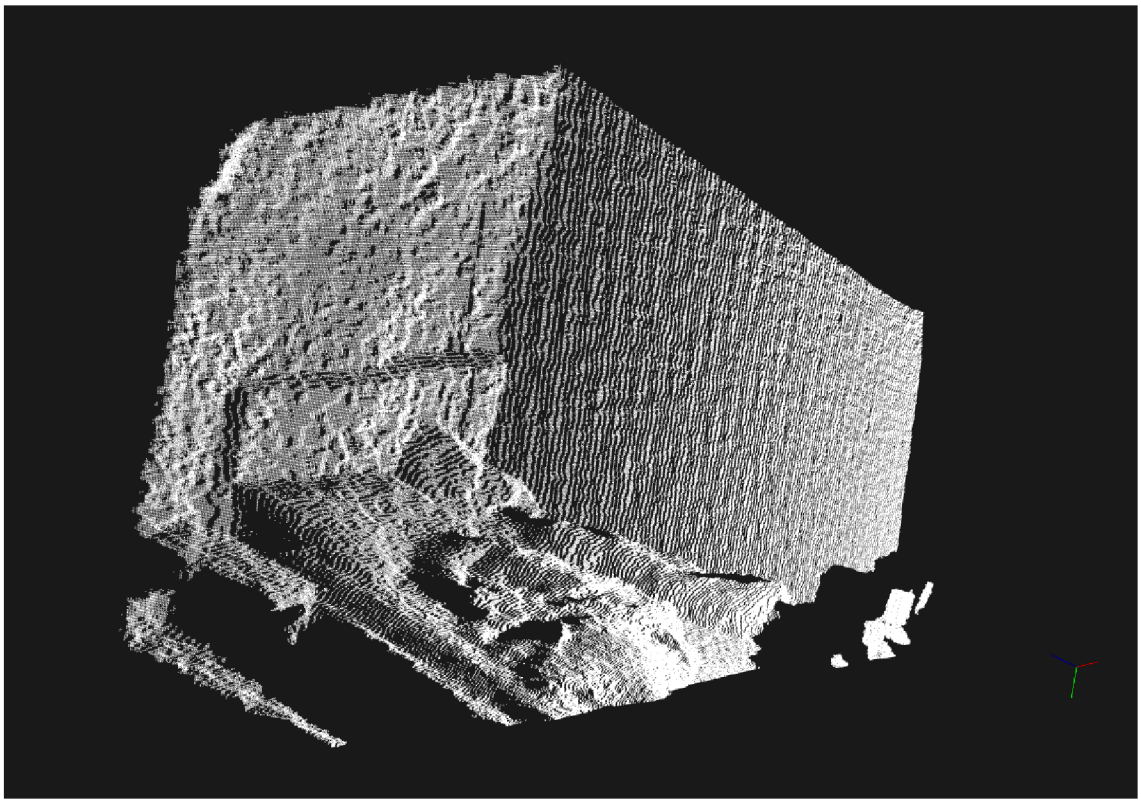
Jelikož se čočka RGB kamery nenachází fyzicky na stejném místě jako čočka infra kamery, je nutné provést mapování barevných dat na hloubková. Toto mapování bývá také nazýváno UV mapování a je provedeno následujícím výpočtem [19], kde f je ohnisková vzdálenost, x, y a z jsou data z hloubkové kamery. Souřadnice u a v ukazují do „textury“ tedy RGB snímku. Někdy je barva označována jako čtvrtý rozměr.

$$u = f \cdot \frac{x}{z} \tag{2.1}$$

$$v = f \cdot \frac{y}{z} \tag{2.2}$$

Jednou ze základních vlastností mračna bodů je uspořádanost (organizovanost). Mračno může být uspořádané nebo chaotické (neorganizované). Mračno je organizované, když je v paměti uloženo tak, jak je nasnímáno. Tedy například u senzoru Kinect organizované mračno znamená, že je v paměti uloženo jako matice 640x480 položek. Výhodou takto uloženého mračna je, že jsou na něm mnohem efektivněji prováděny operace týkající se prohledávání nejbližšího okolí vybraného bodu. Jakákoliv operace nad organizovaným mračnem,

[1] z anglického Red Green Blue Depth, tedy červená, zelená, modrá a hloubka



Obrázek 2.1: Ukázkové mračno bez barevné informace.

která odstraňuje body z mračna (včetně NaN hodnot) nebo přidává body, ruší vlastnost organizovanosti.

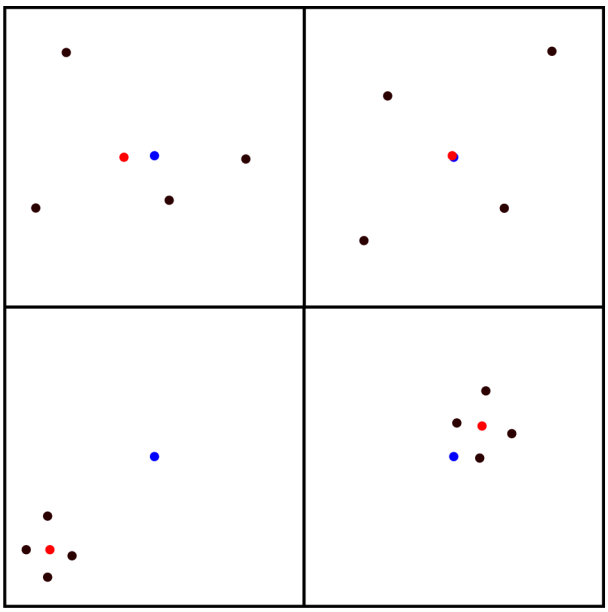
2.2 Filtrování a výpočet normálových vektorů

Množství algoritmů zpracovávající mračna bodů je výpočetně náročná, proto je vhodné mračna podvzorkovat a tím zmenšit počet jejich bodů. Výpočet normálových vektorů je důležitý pro získání informace o povrchu mračna bodů. Tato sekce vzniká za účelem teoreticky popsat algoritmy, které jsou posléze někdy i vícenásobně použity v algoritmech vyšší úrovně aplikace a je dobré znát teoretický základ.

2.2.1 Podvzorkování mřížkovým filtrem

Mřížkový filtr (anglicky Voxel Grid Filter) funguje tak, že prostor (3D) s původním mračnem je proložen kvádry, kdy každý kvádr obsahuje několik bodů z původního mračna [24]. Kolik bodů kvádr obsahuje záleží na hustotě mračna a na zvolené velikosti hrany kvádru. Ve většině případů chceme filtrovat ve všech třech rozměrech rovnoměrně, proto se volí všechny hrany kvádru stejně velké a jedná se tak o krychli v prostoru. Z původních bodů je vypočten jejich střed a posléze jsou odstraněny. Do krychle je pak umístěn jeden bod se souřadnicemi vypočteného středu. Z implementačního hlediska by bylo rychlejší nový bod umístit do středu krychle, ale výpočet středu shluku bodů umožňuje přesnější aproximaci podvzorkovaného povrchu. Nevýhodou je, že nevzniká mračno s uniformní hustotou

bodů. K demonstraci tohoto filtru můžeme nahlédnout na obrázek 2.2, kde je funkčnost znázorněna pro dvourozměrný prostor.



Obrázek 2.2: Černé body jsou původní body mračna. Modré jsou středy čtverců a červené jsou středy shluků v prostoru ohraničeném čtvercem.

2.2.2 Výpočet normálových vektorů

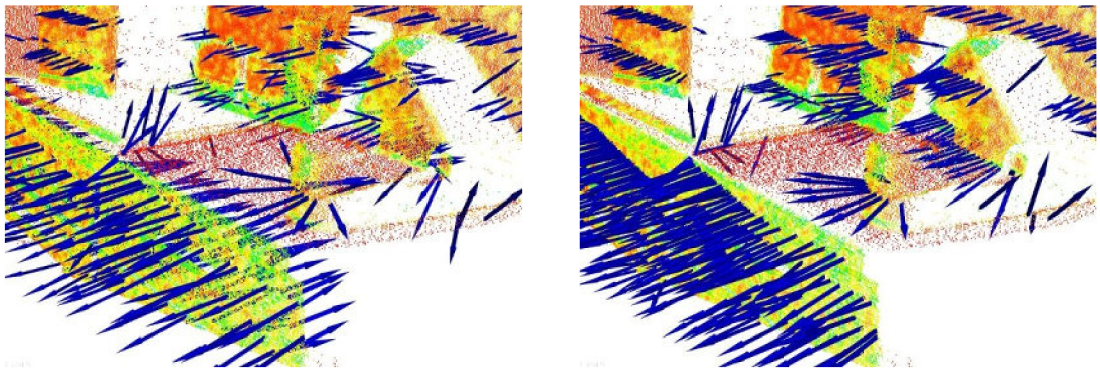
Pro algoritmy, které pracují s povrchem, je nutné získat základní informace o mračnu. Jednou z takových základních informací je směr normálových vektorů. Normála je definována jako vektor kolmý na rovinu (v trojrozměrném prostoru). Mračno je však definováno pouze body, takže se musí provést odhad směru normálových vektorů pro všechny body. Možnost, jak provést výpočet normály je, že pro uvažovaný bod, kterému se má odhadnout normála, je vybráno okolí s daným poloměrem a těmito body je proložena plocha. Z této plochy je pak odvozen normálový vektor. Druhou možností je odhad přímo z bodů. Tento postup zahrnuje analýzu vlastního vektoru a vlastního čísla kovarianční matice pro nejbližší okolí zkoumaného bodu. Podle zdroje [1] je pro bod p_i kovarianční matice sestrojena jako:

$$\mathcal{C} = \frac{1}{k}\sum_{i=1}^{k} \cdot (p_i - \bar{p}) \cdot (p_i - \bar{p})^T, \mathcal{C} \cdot \vec{v_j} = \lambda_j \cdot \vec{v_j}, \, j \in \{0, 1, 2\} \tag{2.3}$$

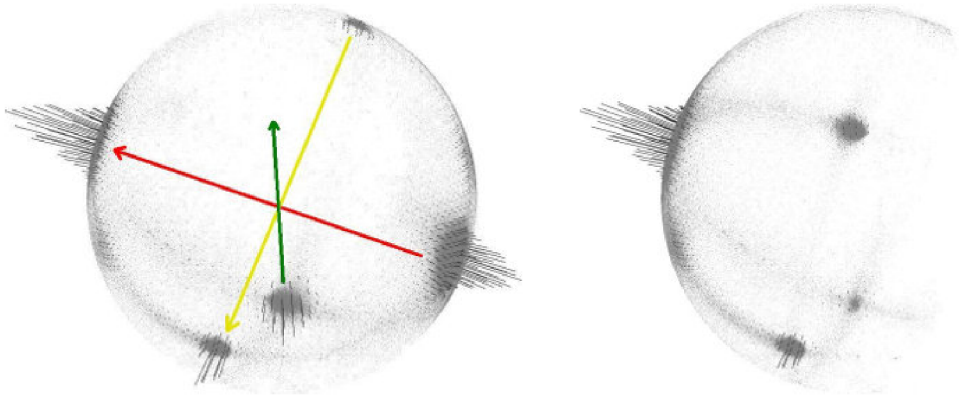
Symbol k určuje počet bodů ve vybraném okolí bodu p_i, $\bar{p}$ je střed všech bodů ve vybraném okolí, λ_j je j-té vlastní číslo kovarianční matice a $\vec{v_j}$ je j-tý vlastní vektor [1].

Je zřejmé, že pro stejnou množinu bodů může normála mít dvě orientace. To by vedlo k nekonzistenci orientace vypočtených normál. Díky tomu, že pro mračno známe bod, odkud je sejmuto, jsou všechny normály přeorientovány vzhledem k bodu sejmutí.

Na obrázku 2.5 vidíme výpočet normál na reálném příkladu.



Obrázek 2.3: Vlevo ilustrace normál s nekonzistentní orientací. Vpravo s konzistentní orientací. Převzato ze zdroje [1].



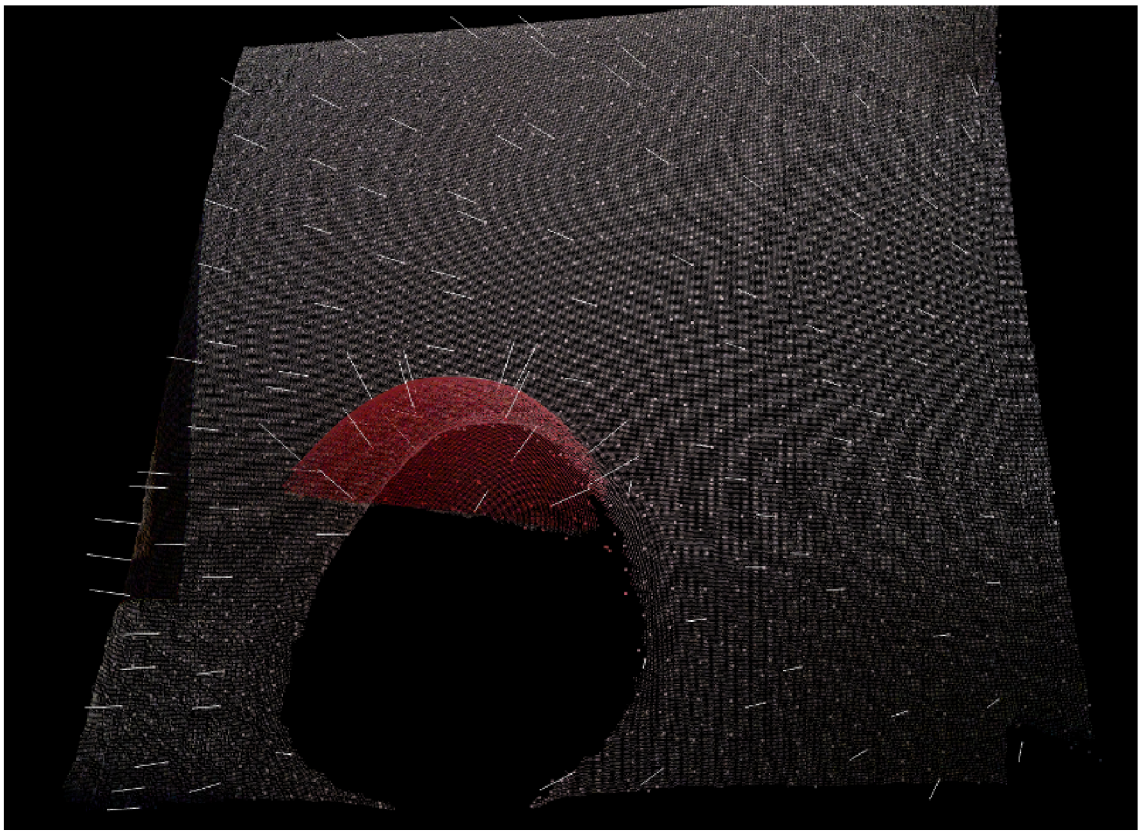
Obrázek 2.4: Vlevo Gaussova plocha s hustotou normál s nekonzistentní orientací, vpravo s konzistentní orientací. Převzato z [1].

2.3 Metody vyhlazení mračna

S rozvojem technologií v odvětví počítačové grafiky vznikají algoritmy, které používají k reprezentaci povrchu namísto polygonů pouze body. Výhoda zobrazování modelu pomocí mračna bodů spočívá v možnosti zobrazit velice jemné detaily modelu. Senzory, poskytující tato data však pracují s omezeným rozlišením a vstupní data jsou často zašuměná. Například Kinect dokáže určovat hloubkovou informaci pouze diskrétně, což se projevuje zvláště ve větších vzdálenostech objektů od senzoru „schodkovitostí“. To můžeme vidět na obrázku 4.8 vlevo. Tyto artefakty vznikají nejvíce při snímání ploch pod úhlem.

Proto je nutné mračno zpracovat tak, aby bylo dobře reprezentovatelné. Za tímto účelem vzniká několik algoritmů, které vyhlazují a někdy i převzorkovávají vstupní data. V sekci Rychlého bilaterálního filtru (sekce 2.3.1) je popsán postup pro organizovaná mračna. Často jsou však data upravena tak, že jsou odstraněny nežádoucí hodnoty nebo jsou naopak přidány další body spojením mračen po registraci a mračno tak ztrácí vlastnost organizovanosti. Proto je zde popsána i metoda MLS[2][3] (sekce 2.3.2).

[2]Moving Least Squares



Obrázek 2.5: Odhadnuté normálové vektory bodů na snímku ze senzoru Kinect.

2.3.1 Rychlý bilaterální filtr

Bilaterální filtr je nelineárním filtrem, který zanechává ostré hrany a může být použit jak ve 2D, tak v 3D. Konkrétní implementace rychlého bilaterálního filtru využívá pohledu na data jako na signál. Filtr je vyjádřen ve vyšší dimenzi a je tak možné ho popsat jako konvoluci, která je následována dvěma jednoduchými nelinearitami v rozšířeném prostoru. Veškeré informace jsem čerpal ze zdroje [18].

Filtr nahrazuje každý pixel v 2D prostoru respektive bod v 3D prostoru váženým průměrem svého okolí. Váha každého sousedního bodu se zmenšuje s rostoucí vzdáleností v prostoru $\mathcal{S}$ a se vzdáleností v ose intenzity $\mathcal{R}$. Jako úbytková funkce je použita Gaussova funkce G_σ. Pro černobílý obrázek I je funkce bilaterálního filtru definována jako

$$I^{bf}_{\mathbf{p}} = \frac{1}{W^{bf}_{\mathbf{p}}} \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) I_{\mathbf{q}} \tag{2.4}$$

kde

$$W^{bf}_{\mathbf{p}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) \tag{2.5}$$

Parametr σ_s definuje velikost okolí použitého k filtrování. Parametr σ_r pak definuje jakou mírou je zmenšena váha sousedních pixelů. W^{bf} normalizuje součet vah.

Aby filtr mohl být převeden na konvoluci, je definována homogenní intenzita, která nám umožňuje získat homogenní komponentu $W_{\mathbf{p}}^{bf}$ po konvoluci. Z rovnice 2.4 vidíme, že nelinearita vychází z dělení členem W^{bf} a ze závislosti na intenzitě pixelu z $G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|)$.

Řešením, jak získat homogenní intenzitu, je vynásobení obou stran rovnice 2.4 členem $W_{\mathbf{p}}^{bf}$. Pak můžeme rovnice 2.4 a 2.5 přepsat pomocí dvourozměrných vektorů na:

$$\begin{pmatrix} W_{\mathbf{p}}^{bf} I_{\mathbf{p}}^{bf} \\ W_{\mathbf{p}}^{bf} \end{pmatrix} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) \begin{pmatrix} I_{\mathbf{q}} \\ 1 \end{pmatrix} \tag{2.6}$$

Aby byla zachována taková vlastnost bilaterálního filtru, že má vážený přírůstek, definujeme funkci W, jejíž hodnota je vždy 1. Pak:

$$\begin{pmatrix} W_{\mathbf{p}}^{bf} I_{\mathbf{p}}^{bf} \\ W_{\mathbf{p}}^{bf} \end{pmatrix} = \sum_{\mathbf{q} \in \mathcal{S}} G_{\sigma_s}(\|\mathbf{p} - \mathbf{q}\|) G_{\sigma_r}(|I_{\mathbf{p}} - I_{\mathbf{q}}|) \begin{pmatrix} W_{\mathbf{q}} I_{\mathbf{q}} \\ W_{\mathbf{q}} \end{pmatrix} \tag{2.7}$$

Přiřazením dvojice $(W_{\mathbf{q}} I_{\mathbf{q}}, W_{\mathbf{q}})$ každému pixelu $\mathbf{q}$ vyjadřujeme filtrované pixely jako lineární kombinaci jejich okolí. Dělení je pak prováděno až v závěrečné fázi. Dvojice (WI, W) je pak homogenní intenzitou

Ačkoliv je rovnice 2.7 lineární kombinací, nereprezentuje lineární filtr, protože je stále závislá na aktuální hodnotě pixelu. Provedením dalších úprav dostáváme lineární a nelineární složku bilaterálního filtru. Lineární složka je vyjádřena jako

$$(w^{bf} i^{bf}, w^{bf}) = g_{\sigma s, \sigma_r} \otimes (wi, w) \tag{2.8}$$

a nelineární jako

$$I_{\mathbf{p}}^{bf} = \frac{w^{bf}(\mathbf{p}, I_{\mathbf{p}}) i^{bf}(\mathbf{p}, I_{\mathbf{p}})}{w^{bf}(\mathbf{p}, I_{\mathbf{p}})} \tag{2.9}$$

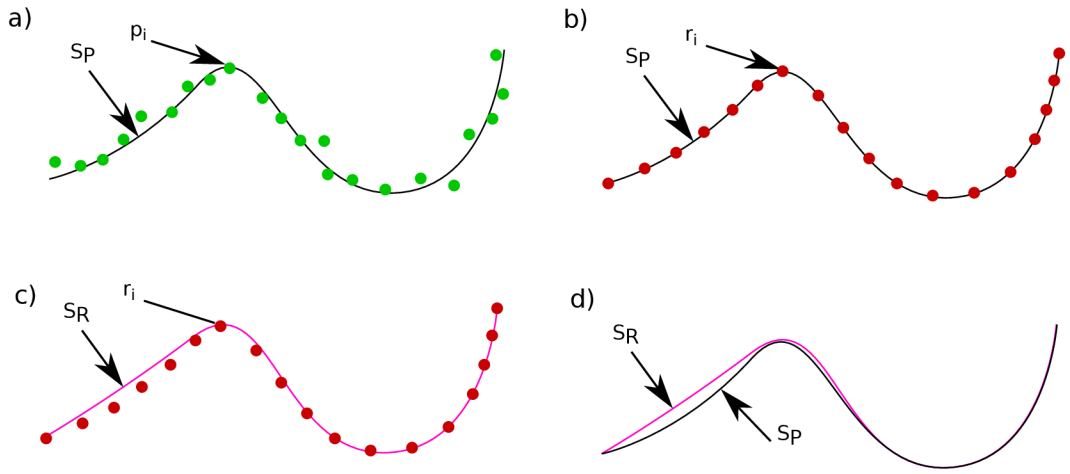
kde $g_{\sigma s, \sigma_r}, i$ a w jsou zobrazeními v relaci $\mathcal{S} \times \mathcal{R}$.

Jak bylo zmiňováno, nelineární složka je ve skutečnosti tvořena dvěma operacemi. Funkce $w^{bf} i^{bf}$ a w^{bf} jsou vyhodnoceny v bodě $(\mathbf{p}, I_{\mathbf{p}})$. Druhou nelinearitou je dělení.

Chování filtru tedy ovlivňujeme parametry σ_s a σ_r. Důležitou poznámkou je, že okraje filtrovaných dat jsou doplněny nulami. Při nevhodně zvolených parametrech pak mohou vznikat nežádoucí artefakty. Výhodou tohoto filtru je rychlost výpočtu. Díky tomu lze použít pro aplikace, které běží v reálném čase.

2.3.2 Moving Least Squares

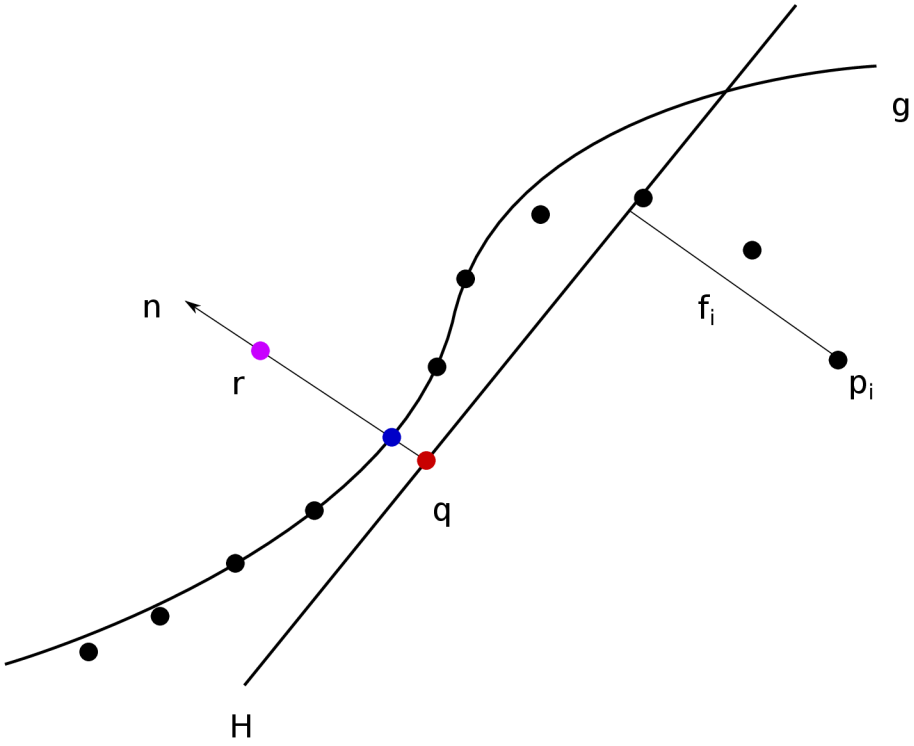
Mějme nasnímaný povrch S_P tvořen body $P = \{p_i\}$. Povrch MLS je pak vytvořen nahrazením bodů P pozměněnou množinou bodů $R = \{r_i\}$, která definuje povrch S_R jež aproximuje S_P. Na obrázku 2.6 je znázorněn tento postup 2D prostoru. Informace jsem čerpal ze zdrojů [3] a [10].



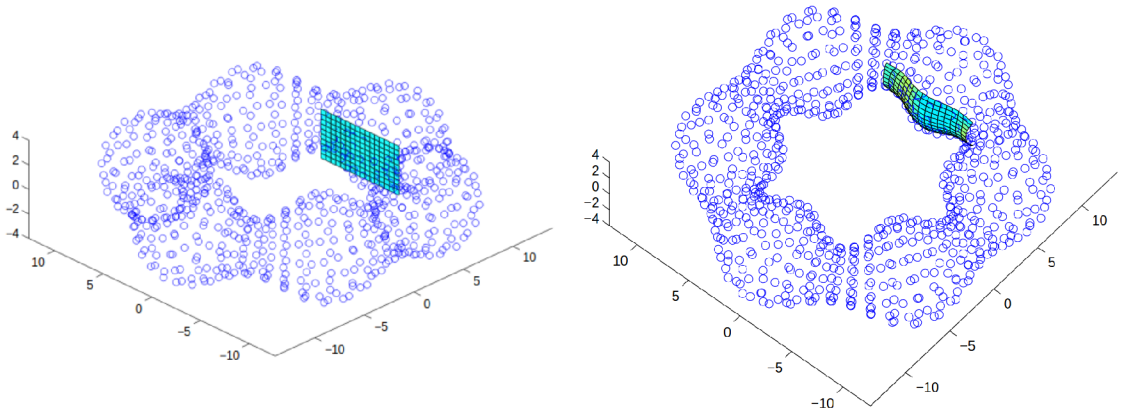
Obrázek 2.6: Znázornění vyhlazení pomocí algoritmu MLS. Popis: a) Povrch aproximovaný body, které mohou obsahovat šum nebo být redundantní. b) Upravená množina bodů R, které aproximují původní povrch S_P. c) Proložení množiny bodů R vyhlazenou křivkou S_R. d) Rozdíl mezi nevyhlazenou S_P a vyhlazenou křivkou S_R.

Základní myšlenkou algoritmu MLS v 3D prostoru je tedy projekce bodů na povrch. Poté je definován MLS povrch a původní body jsou na něj promítnuty. Na obrázku 2.7 je tato projekce znázorněna. Na obrázku 2.8 je znázorněn případ promítání bodů na rovinu v 3D prostoru.

Výhodou tohoto algoritmu je použitelnost na mračnu bodů, které nemá organizovanou (uspořádanou) strukturu. Nevýhodou je vyšší časová náročnost.



Obrázek 2.7: Nejdříve si určíme referenční plochu (doménu) H pro fialový bod r. Projekce bodu r na H nám umožní získat původní polohu bodu r na H, tj. červený bod q. Poté přejdeme k polynomiální aproximaci díky níž získáme křivku g. Tento krok zahrnuje proložení výšek f_i vhodným polynomem. Výšky f_i získáme spuštěním kolmice z bodu p_i na H. Váha každého bodu p_i je funkcí vzdálenosti p_i od q. Následná projekce bodu r na polynom g je výsledek MLS projekce.



Obrázek 2.8: Ilustrace promítání bodů na rovinu v 3D prostoru. Převzato z [10].

2.4 Spojování existujících mračen na základě klíčových bodů

Vypočítat FPFH hodnoty (sekce 2.4.2) pro množinu všech bodů v mračnu je výpočetně neefektivní. Proto se ve vstupních datech hledají takzvané klíčové body, které jsou z hlediska zpracování obrazových dat nějak zajímavé. Základními vlastnostmi klíčových bodů je to, že takové body jsou výrazné a také, že jsou zjistitelné z různých bodů pohledu. Mezi algoritmy, jak získat tyto klíčové body v 3D prostoru patří například ISS[3] nebo NARF[4]. Existují i metody odvozené z algoritmů určené pro 2D data. Mezi zástupce patří například popsaný algoritmus SIFT 2.4.1 nebo dále Harris [25].

2.4.1 Detekce klíčových bodů pomocí SIFT

Algoritmus určený pro detekci klíčových bodů nezávisle na souřadnicích pozorovacího bodu, osvětlení a šumu ve vstupní scéně. Metoda je původně určena pro dvourozměrná data, knihovna PCL však obsahuje adaptaci pro tři rozměry. Detekce klíčových bodů se skládá ze čtyř základních kroků. Informace jsem čerpal ze zdroje [12].

- Vyhledání extrému v scale-space (spojitá funkce měřítka [4]) - výpočet probíhá přes všechny velikosti a lokace ve vstupních datech. To je implementováno za použití rozdílu Gaussiánů. Scale-space obrázku je definována jako funkce $L(x, y, \sigma)$. Tato funkce je vytvořena konvolucí Gaussiánu

  $$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/2\sigma^2} \tag{2.10}$$

  a vstupního obrázku $I(x, y)$. Tedy:

  $$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.11}$$

  Pro detekci lokálních extrémů definujeme funkci

  $$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = L(x, y, k\sigma) - L(x, y\sigma), \tag{2.12}$$

  kde σ je míra rozostření a k je násobicí faktor.

  Rozdíl Gaussiánů je vybrán protože dobře aproximuje Laplacián Gausiánů, ale je mnohem výpočetně efektivnější.

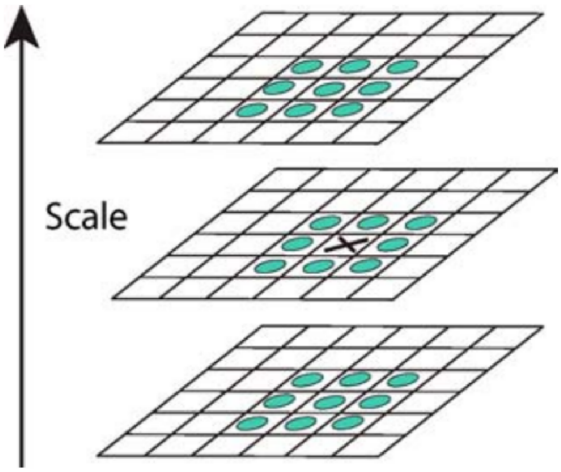
  Pro získání lokálních extrémů funkce $D(x, y, \sigma)$ je každý zkoumaný bod porovnán se svým osmi-okolím v aktuální vrstvě a s devíti body ve vrstvě velikosti nad a pod. Bod je pak vybrán jako lokální extrém pokud je menší (minimum) nebo větší (maximum) než všechny body v okolí. To vidíme na obrázku 2.9.

- Určení přesné pozice klíčového bodu - když máme takto vybrány lokální extrémy, je vhodné některé body zamítnout kvůli nízkému kontrastu, který vede k vyšší náchylnosti k šumu.

- Přiřazení orientace - přiřazením konzistentní orientace každému bodu získáváme invarianci vůči rotaci zpracovávaného obrázku.

- Přiřazení lokálního deskriptoru - předešlými kroky jsme získali invarianci vůči lokaci, velikosti a orientaci klíčových bodů. Dalším krokem je výpočet deskriptoru pro lokální oblast tak, aby byl co nejvíce rozlišitelné a zároveň si udrželi zmíněné invariance. Zavedení deskriptoru přináší nezávislost na změně pozice bodu pozorování a osvětlení.

[3] Intrinsic Shape Signatures [27]
[4] Normal Aligned Radial Feature [26]



Obrázek 2.9: Porovnání bodu s jeho okolím. Převzato z [12].

2.4.2 Vylepšený výpočet histogramu vlastností bodu FPFH

Později popsaný algoritmus ICP (sekce 2.4.3) spojuje dvě mračna statické scény pomocí iterativní translace a rotace snímků a následného ohodnocení, zda se snímky shodují v překrývající se části více nebo méně. Problémem je, pokud jsou snímky v první iteraci posunuty o velký úhel nebo vzdálenost od sebe. Algoritmus ICP má pak v některých případech tendenci ukončit porovnávání po spadnutí do lokálního minima. Proto je vhodné provést lepší prvotní odhad korespondence dvou snímků. K vytvoření takového odhadu slouží takzvané „featury“[5] mračna. Pro svou práci jsem se rozhodl použít histogram vlastností nazvaný FPFH[6]. V této sekci je detailně popsán výpočet histogramu těchto vlastností. Informace jsem čerpal ze zdroje [22].

FPFH zjednodušuje výpočet klíčových vlastností mračna nazvaných PFH, popsaných ve zdroji [20]. Hlavním vylepšením oproti původnímu algoritmu PFH je ukládání a znovupoužití již vypočtených hodnot (cachování), uspořádání mračna a změna množiny bodů, ze kterých je vlastnost vypočítána. Poslední uvedené vylepšení metody snižuje výpočetní složitost z $O(k^2)$ na $O(k)$, což zlepšuje možnost použití této metody reálném čase.

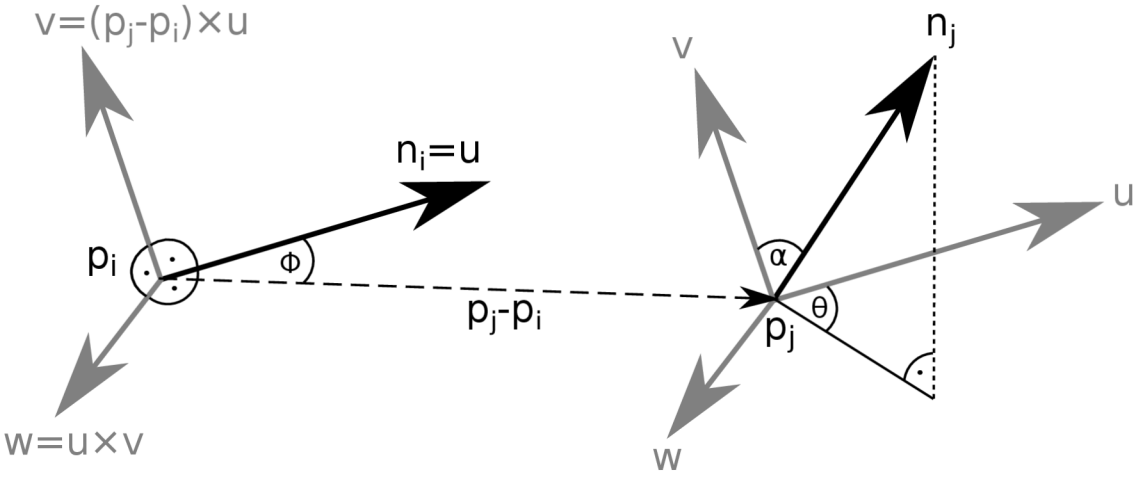
Vstupem základní verze algoritmu jsou souřadnice bodů a normály odhadnuté pomocí postupu uvedeného v sekci 2.2.2. Výpočet vlastnosti bodu p je pak proveden tak, že se uvažuje jeho okolí udané poloměrem koule r. Pro každou dvojici bodů p_i a p_j, kde $i \neq j$, z k-okolí bodu p a jejich normály n_i a n_j definujeme Darbouxův uvn rámec, kde $u = n_i$, $v = (p_j - p_i) \times u$, $w = u \times v$. Je důležité, že úhel mezi normálou a křivkou spojující body p_i a p_j je menší v bodě p_i. Úhlové odchylky jsou pak definovány jako

$$\alpha = v \cdot n_j \tag{2.13}$$

$$\phi = u \cdot \frac{(p_j - p_i)}{||p_j - p_i||} \tag{2.14}$$

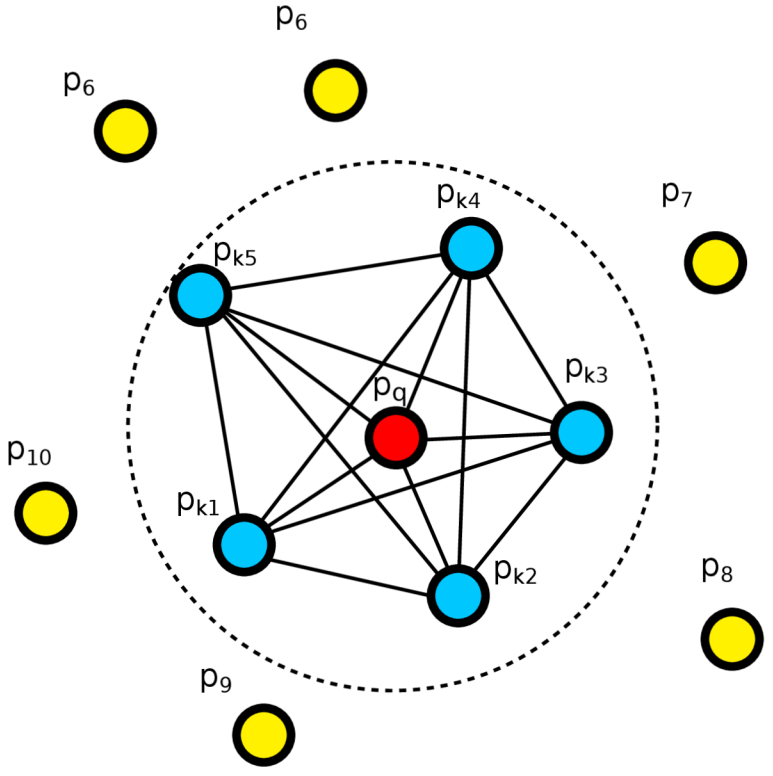
$$\theta = arctan(w_j \cdot n_j, u \cdot n_j) \tag{2.15}$$

[5] Česky - význačná vlastnost, hlavní rys
[6] Fast Point Feature Histogram



Obrázek 2.10: Konstrukce Darbouxova rámce mezi body p_i a p_j.

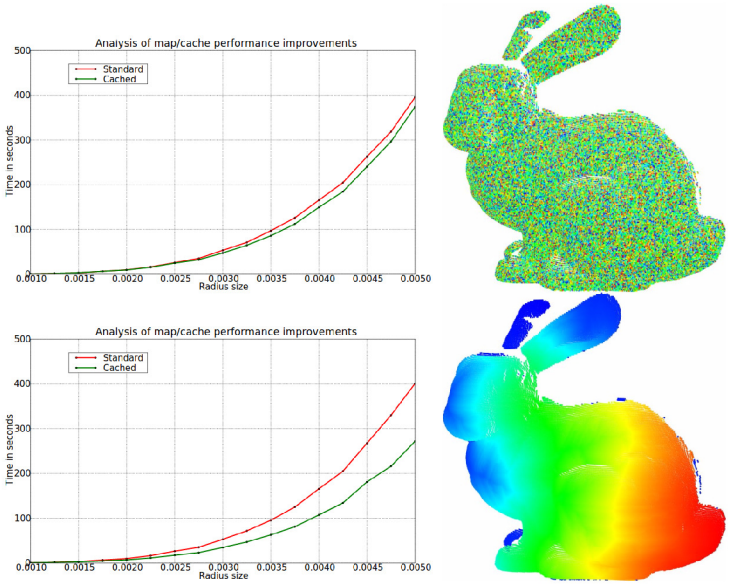
Na obrázku 2.11 je znázorněn bod zájmu p_q. Čárkovanou kružnicí, jejímž středem je bod zájmu p_q, je ohraničeno jeho k-okolí ve kterém leží body p_{kn}, které jsou uvažovány při výpočtu vlastností. Body p_n jsou neuvažované body mračna pro bod p_q s daným poloměrem okolí. Vztahy mezi dvojicemi bodů jsou určeny vytvořením úplného grafu, kde uzly jsou body v dané kružnici pro 2D respektive kouli pro 3D.



Obrázek 2.11: Graf pro bod zájmu p_q. Varianta PFH.

Z příkladu, kdy uvažujeme jako bod zájmu bod p_{k2}, je zřejmé, že původní bod p_q leží taktéž v jeho sousedství, proto se vypočtené hodnoty ukládají do cache, což urychluje výpočet tím, že není potřeba tuto hodnotu vypočítat znova.

Dalším výrazným urychlením výpočtu je reorganizace mračna. Pokud je mračno neuspořádané, body mají náhodné indexy a náhodný přístup do paměti výpočet zpomaluje. Prostým uspořádáním bodů je výpočetní čas redukován na 75% původní hodnoty. Rozdíl mezi neuspořádaným a uspořádaným mračnem můžeme vidět na obrázku 2.12



Obrázek 2.12: Nahoře neuspořádané indexy bodů mračna, dole uspořádané. Převzato z [22].

Zásadní zrychlení algoritmu spočívá v tom, že se nevypočítávají všechny hodnoty, ale pouze ty mezi bodem zájmu a jeho sousedy. Takto vypočtené hodnoty nazýváme SPFH[7]. Ve druhém kroku se znovu určí body v sousedství a jejich SPFH slouží k určení váhy vypočtené hodnoty pro výsledný histogram bodu p_q nazvaný FPFH. Výsledný histogram je tedy určen takto:

$$FPFH(p_q) = SPFH(p_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_k}\cdot SPFH(p_k) \tag{2.16}$$

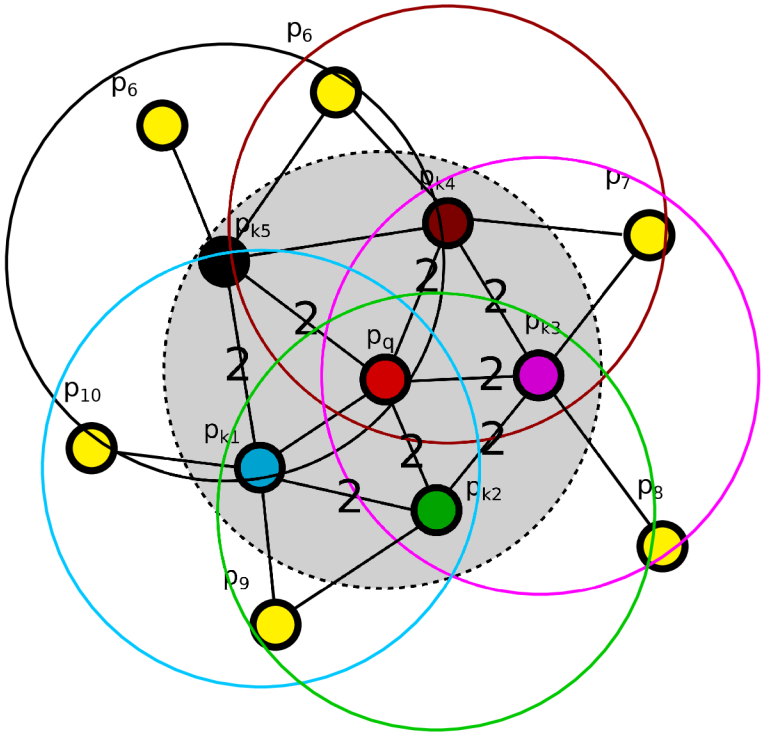
Kde symbol ω_k je ohodnocení vztahu bodů p_q a p_k. Může se použít například jejich vzdálenost v prostoru.

Na obrázku 2.13 je zobrazeno, jak vypadá výsledný graf pro výpočet FPFH hodnoty. Hrana ohodnocená číslem 2 značí, že se hrana podílí na výpočtu hodnoty dvakrát.

Rozdíly mezi PFH a FPFH jsou tedy následující:

- PFH může počítat s body v maximální vzdáleností r, kdežto FPFH až $2r$.
- FPFH není konstruována jako úplný graf, tudíž může některé výpočty mezi dvojicemi vynechat.
- FPFH používá metriku pro vážení vztahu dvou bodů.

[7]Simplified Point Feature Histogram



Obrázek 2.13: Graf pro bod zájmu p_q. Varianta FPFH.

Výsledný histogram by měl mít teoreticky q^d položek, kde q je počet intervalů rozsahu hodnot featur a d je počet vybraných featur. V implementaci knihovny PCL je hodnota stanovena jako $q = 11$ a počet featur je dán jako $d = 3$. Můžeme si všimnout, že původně jsou featury čtyři a to α, ϕ, θ a vzdálenost bodů. Bylo zjištěno, že zanedbáním vzdálenosti postup nijak neutrpí v přesnosti, proto je tato hodnota vypuštěna. Neupravený výstup by měl tedy $11^3 = 1331$ hodnot. Proto se používá dekorelace hodnot, kdy se vytvoří d histogramů, kde každý odpovídá jedné featuře a tyto histogramy jsou spojeny v jeden. V knihovně PCL je pro tento účel připraven datový typ `pcl::FPFHSignature33`.

Důležitým poznatkem je, že parametr poloměru vyhledávání musí být větší než poloměr, který je použit pro výpočet normál (2.2.2).

2.4.3 Algoritmus iterativního nejbližšího bodu ICP

Algoritmus ICP[8] nachází široké využití v oblasti rekonstrukce jak 2D tak 3D statických dat. Dalším příkladem jeho využití je lokalizace autonomního robota v prostoru reálného světa. Algoritmus využívá optimistického předpokladu, že dvě mračna jsou blízko sebe a snaží se najít nejlepší korespondenci překrývajících se částí mračen. Jedno mračno vždy slouží jako cíl a druhé jako zdroj. Mračno označené jako cíl je neměnné. Pomocí aplikace translace a rotace na mračno označené jako zdroj se snažíme nalézt co nejlepší korespondenci.

ICP ze vstupních mračen stanoví Γ korespondencí mezi zdrojovým a cílovým mračnem. Používá rotaci $\mathbf{R}$ a translaci $\mathbf{T}$ k vytvoření odhadové matice pro inicializační krok. Poté je počítána chyba jako součet čtverců rozdílů mezi Γ rozsahovými hodnotami shodných

[8]Iterative Closest Point

vzorků. Funkce je definována jako:

$$C(\mathbf{R},\mathbf{T}) = \sum_{j=1}^{\Gamma}(p_{0,j} - p_{1,j}(\mathbf{R},\mathbf{T}))^2 \tag{2.17}$$

Kde $p_0 \in \mathbf{P}_0$ a $p_1 \in \mathbf{P}_1$. Mračno $\mathbf{P}_0$ je cílové, $\mathbf{P}_1$ zdrojové. $p_{1,j}(\mathbf{R},\mathbf{T})$ jsou rozsahové hodnoty získány ze zdrojového mračna $\mathbf{P}_1$ po transformaci definované maticemi $(\mathbf{R},\mathbf{T})$ [13].

Nevýhodou této metody je, že může lehce spadnout do lokálního minima, při nevhodných vstupních datech, jak můžeme vidět na obrázku 4.3.

2.5 Metody rekonstrukce povrchu

Po získání a upravení vstupních dat přichází na řadu samotná rekonstrukce povrchu. Existuje několik dostupných metod. Metody rekonstrukce povrchu neboli polygonizace mračna bodů lze rozdělit na tři skupiny a to podle toho, jakým způsobem zpracovávají vstupní data a následně tvoří rekonstruovaný povrch. Jedná se o tyto skupiny:

- metody rostoucí oblasti - Greedy Projection Triangulation, Marching Cubes
- geometrické metody - Alpha Shapes, Delaunayho triangulace
- algebraické metody - Poissonova metoda, Grid Projection.

Nejjednodušší případ vytvoření povrchu z mračna bodů je, když má mračno organizovanou strukturu. Jednoduše tak dokážeme nalézt sousední body a spojit je ve výslednou polygonální reprezentaci. Dále jsou popsány netriviální metody polygonizace mračna bodů, které jsem použil při provádění experimentů v implementační části. Důležitým rozdělením je taktéž metoda zpracování vstupních dat. Metody můžeme rozdělit podle toho, zda zachovávají původní strukturu vstupního mračna, či nikoliv. Důležitou vlastností každé metody je, zda je předpokládaným výstupem manifold[9], což je uzavřený povrch. Je zřejmé, že jediný snímek ze senzoru Kinect nemůže vést k výstupu s takovou vlastností. Po sjednocení vhodných mračen už to však možné je.

2.5.1 Greedy Projection Reconstruction

Metoda je vhodná k polygonizaci neorganizovaného mračna. Díky svým vlastnostem je také vhodná k inkrementální polygonizaci již hotových polygonů a přírůstku v podobě dalších bodů. Metoda je velmi rychlá a je vhodným kandidátem pro použití v reálném čase. Veškeré informace jsem čerpal ze zdroje [16].

Tato metoda pracuje v několika krocích. Prvním krokem je nalezení nejbližších sousedů aktuálně vybraného bodu. Jelikož je zpracováváno neorganizované mračno, je toto netriviálním problémem. Možností, jak získat nejbližší okolí bodu v d-rozměrném prostoru je použití kD stromu.

Tedy pro každý bod p_i je nalezeno jeho k-okolí tvořeno k nejbližšími body, které jsou v 3D prostoru vymezeny koulí o průměru $r = \mu \cdot d_0$. Tato koule je díky uživatelem stanovenému parametru μ adaptivní k hustotě mračna v lokálním prostoru. Hodnota d_0 je vzdálenost p_0 ke svému nejbližšímu sousedu. Takto nalezené nejbližší body jsou promítnuty na rovinu, která je přibližně tangenciální k povrchu tvořenému okolními body bodu p. Poté

[9] varieta

jsou body ořezány podle viditelnosti a kritéria vzdálenosti. Díky tomu nenastává jev, kdy by výsledný povrch protínal sám sebe. Následně jsou body spojeny hranami s p a s následnými body. Takto jsou vytvářeny trojúhelníky, které jsou omezeny parametry zadanými uživatelem. Výhodou tohoto přístupu je to, že zachovává všechny původní body a žádné body nejsou interpolovány.

Při rekonstrukci povrchu touto metodou jsou bodům přidělovány stavy. Mezi ně patří

- free - počáteční stav všech bodů, jsou definovány jako body bez incidentních trojúhelníků
- fringe - body, které ještě nebyly vybrány jako referenční
- boundary - bod, který byl již vybrán, ale některé trojúhelníky byly zahozeny kvůli nesplnění kritéria velikosti úhlů
- complete - bod je takto označen, pokud jsou určeny všechny jeho trojúhelníky

Nevýhodou je zpracování uzavřených povrchů, kde jsou počáteční body označeny jako zpracované a nedojde tak k napojení tvořící uzavřený povrch. To je znázorněno na obrázku 2.14.

Obrázek 2.14: Viditelné oblasti, kde algoritmus Greedy Projection Triangulation nenapojil hraniční oblasti.

Výhodami tohoto algoritmu jsou:

- adaptivnost k hustotě bodů - metoda nepředpokládá hladké nebo uniformě vzorkované mračno bodů na vstupu

- real-time blízké zpracování - díky použití rychlého kD stromu a znovupoužití již vypočtených hran je metoda velice rychlá
- podpora označení bodů - metoda respektuje kritérium pro zastavení lokální triangulace
- paměťová efektivnost - optimalizace, díky níž nemusí být v paměti uloženy všechny hrany již polygonizovaného povrchu
- inkrementální růst - po spojení již rekonstruovaného povrchu a nového mračna se rekonstruuje pouze přírůstek bez nutnosti rekonstrukce celého povrchu od začátku
- výpočet normály plochy - metoda používá váženého průměru normál okolních bodů pro výpočet skutečné normály plochy

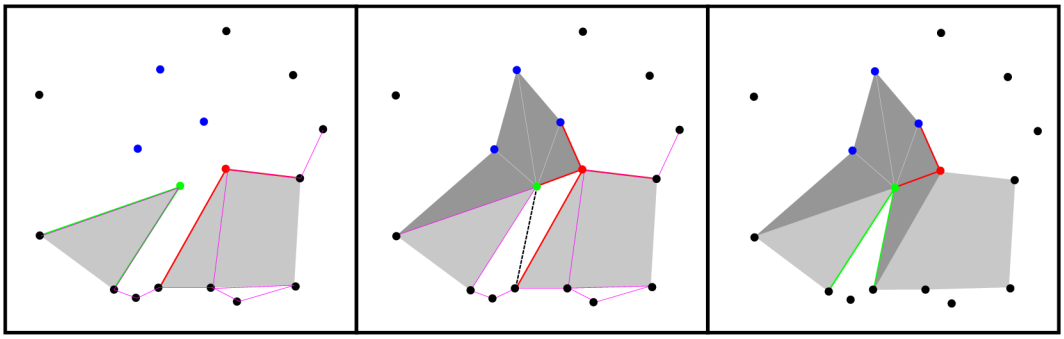
Výpočet normál pro bod p z mračna bodů P je proveden z jeho k okolí pomocí kovariančí matice C.

$$C = \sum_{i=1}^{k} \xi_i \cdot (p_i - \overline{p})^T \cdot (p_i - \overline{p}), \overline{p} = \frac{1}{k} \cdot \sum_{k}^{i=1} p_i \tag{2.18}$$

Kde $i = 1 \dots k$ a $p \in P$. Symbol ξ reprezentuje váhu bodu p_i jako $\xi_i = exp(-\frac{d_i^2}{\mu^2})$, kde μ je průměrná vzdálenost bodů p_i od bodu p.

Jakmile je vytvořen nový trojúhelník, každé dva sousední body, které jsou připojeny k okrajovému bodu, tvoříc hrany, jsou uloženy. Takto vzniká redundance, která je potřebná v případech, kdy je v sousedství pouze jeden bod. Výhodou je zrychlení výběru hran, protože jsou hledány lokálně.

Nevýhodou je, že v speciálních případech musíme vyřešit dotyk více front v jednom bodě. Tento problém je vyřešen spojením přídavných bodů, dokud každý bod nemá pouze dvě přední hrany. To je popsáno na obrázku 2.15.



Obrázek 2.15: Dvě fronty triangulace se dotýkají v jediném (červeném) bodě, což vede k incidentu hran v takovém bodě. To je řešeno přidáním trojúhelníku navíc (čárkovaná čára). Vizte obrázek vpravo. Zelený bod je aktuálním bodem triangulace. Červený a modré body jsou jeho k okolím. Zelené hrany jsou hranami zeleného bodu. Podobně červené hrany náleží červenému bodu. Světle šedé trojúhelníky jsou původní trojúhelníky spojené s červeným a zeleným bodem. Bez korekce by přibyly pouze tmavé trojúhelníky. Vznikly by tak čtyři hrany spojené v červeném bodě (obrázek uprostřed).

2.5.2 Grid Projection Triangulation

Tento algoritmus používá k rekonstrukci takzvaných krajních povrchů. Jedná se o implicitní typ povrchu definovaného extrémem skalárního pole a zároveň omezeného směrem tohoto pole. Metoda je vhodná pro polygonizaci jak neuzavřených povrchů, tak i uzavřených povrchů, např. pro tisk modelů pomocí 3D tiskáren, právě proto, že zaručuje splnění uzavřenosti (vodotěsnosti). Veškeré informace jsem čerpal ze zdroje [11].

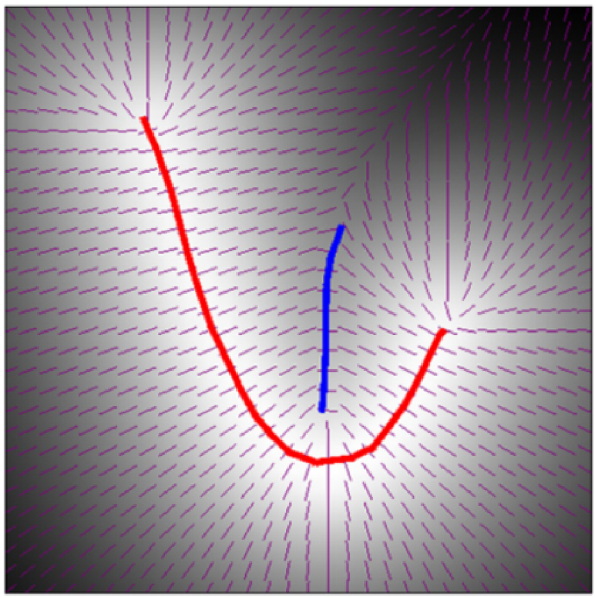
Na rozdíl od iso povrchů může být krajní povrch neuzavřený či neorientovatelný. Může také obsahovat neuzavřené hrany.

Krajní povrch je definován jako

$$S = \{x | x \in arglocalmin_{y \in l(x,n(x))} s(y)\} \tag{2.19}$$

kde $s : \mathbb{R}^d \to \mathbb{R}$, $n : \mathbb{R}^d \to \mathbb{RP}^{d-1}$ a $l(x, n(x))$ značí přímku skrze x se směrnicí $n(x)$.

S je pak tvořeno všemi body z x, jejichž skalární funkce je minimální podél směru. To je znázorněno v 2D na obrázku 2.16.



Obrázek 2.16: Skalární funkce s je znázorněna stupni šedi. Vyšší hodnoty jsou znázorněny světleji. Normálová funkce n je znázorněna krátkými úsečkami. Krajní povrch s je vyjádřen jako červená a modrá křivka. Převzato z [11].

Klíčovou vlastností krajního povrchu je jeho spojitá struktura. Můžeme uvažovat kritický povrch tvořený body z s omezenými $l(x, n(x))$. Tyto body mohou být lokálními maximem, minimem nebo inflexním bodem s. Z toho vyplývá, že krajní povrch je podmnožinou kritického povrchu.

Pokud uvažujeme $\nabla s(x)$ jako gradient vektoru s, je kritický povrch definován jako skalární funkce $g(x) = \vec{n}(x) \cdot \nabla s(x)$ pro x, kde $g(x) = 0$ (nulová množina).

Za předpokladu, že s je hladká funkce se spojitou druhou derivací, krajní povrch je ohraničen na kritickém povrchu inflexními body s, omezenými přímkami $l(x, n(x))$. Tyto inflexní body formují varietu (uzavřený povrch) v nižší dimenzi než konečný povrch (pro

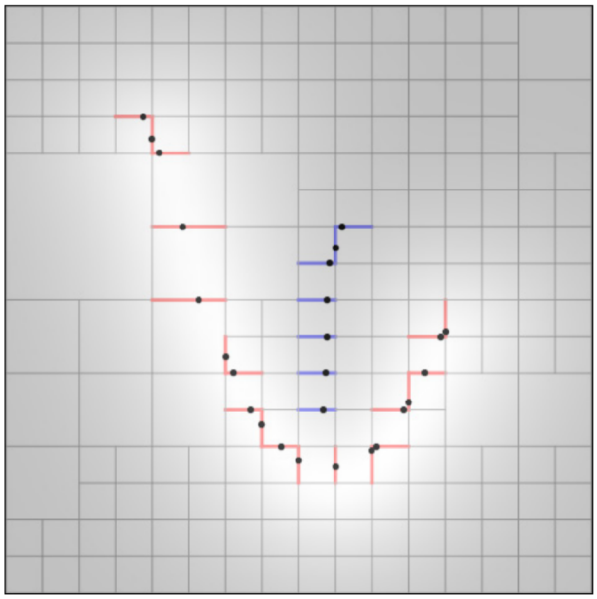
2D jsou to body, pro 3D křivky), jako průsečíky kritického povrchu a nulové množiny druhé derivace s omezené n.

Krátce tedy - konečný povrch je $(d-1)$ varieta s neuzavřenými hranicemi a hladkou skalární funkcí s z $\mathbb{R}^d$. Hranice leží buď na nespojistech n nebo na inflexních bodech s omezenými přímkami $l(x, n(x))$.

Algoritmus pak vypočítá polygonální aproximaci krajního povrchu ve dvou krocích. Prvním krokem je identifikace hran mřížky, které protíná kritický povrch. Tyto hrany jsou nazvány *kritické hrany*. Podmnožina kritických hran, proťatých krajním povrchem, nazýváme krajní hrany. Poté je vytvořen polygonální kritický povrch, který protíná zjištěné kritické hrany.

Na obrázku 2.17 jsou znázorněny kritické hrany s viditelnou strukturou čtvercového stromu (quadtree). Na obrázku 2.18 je vytvořena aproximace polygonálního kritického povrchu. Tyto obrázky znázorňují situaci pro dvourozměrný prostor.

Z uvedeného principu je zřejmé, že metoda nezachovává původní mračno bodů.

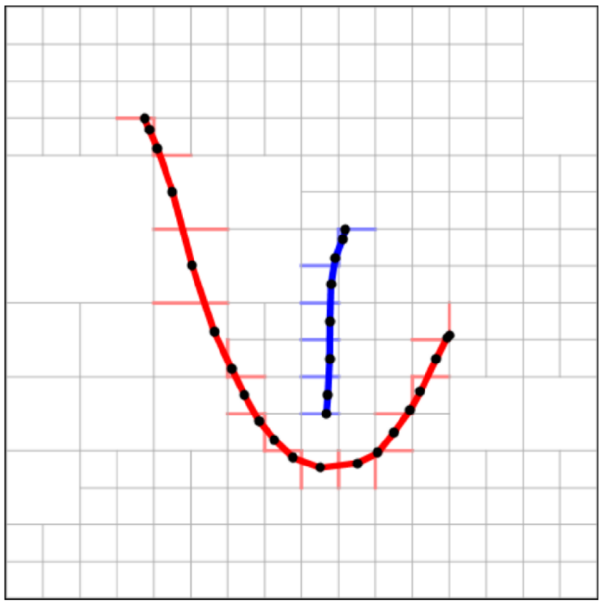


Obrázek 2.17: Znázornění kritických hran mřížky proťatých kritickým povrchem. Převzato z [11].

2.5.3 Poissonova metoda

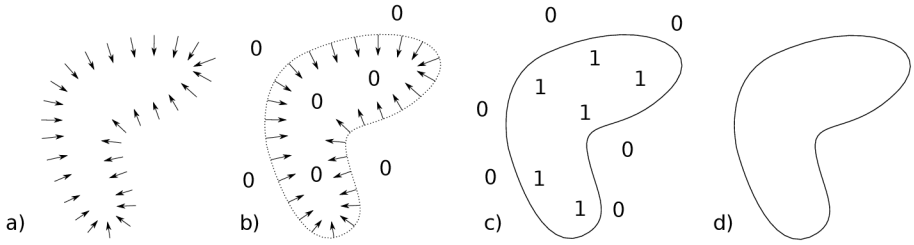
Rozšířenou metodou v rekonstrukci povrchu je Poissonova metoda. Metoda zkoumá povrch poskytnutého mračna a rekonstrukci provádí nad celou množinou dat najednou. Metoda je vhodná pro polygonizaci uzavřených povrchů. Pro neuzavřené povrchy vhodná není, z důvodu vzniku artefaktů. Veškeré informace jsem čerpal ze zdroje [8], kde také můžou být nalezeny veškerá odvození a hlubší teoretický popis metody.

Klíčovou myšlenkou tohoto postupu je uvažování integrálního vztahu mezi orientovanými body modelu a indikační funkcí modelu. Konkrétně gradient indikační funkce je vektorové pole, které nabývá nulových hodnot téměř všude, jelikož indikační funkce je téměř všude konstantní, kromě bodů blízko povrchu, kde je hodnota rovna invertované normále



Obrázek 2.18: Aproximovaný polygonální povrch. Převzato z [11].

povrchu. Díky tomu mohou být orientované body brány jako vzorky gradientu indikační funkce. Na obrázku 2.19 je toto znázorněno.



Obrázek 2.19: a) orientované body $\vec{V}$, b) indikátor gradientu $\nabla\chi_M$, c) indikační funkce χ_M, d) povrch ∂M.

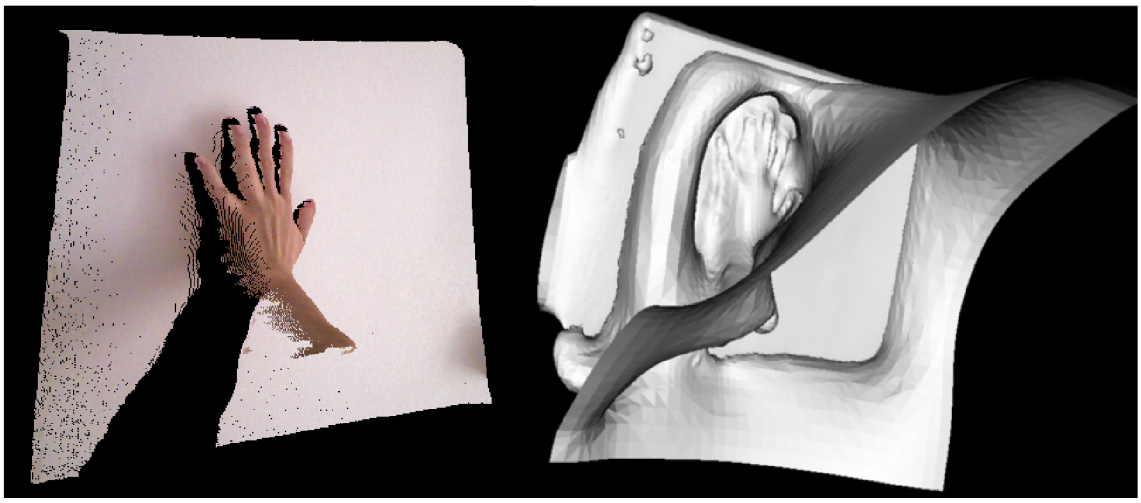
Problém výpočtu indikační funkce je redukován na obrácení směru normál. Pokud tak provedeme, je problém transformován na standardní Poissonův problém, tj. výpočet skalární funkce χ, jejíž Laplacian (divergence gradientu) je roven divergenci vektorového pole $\vec{V}$,

$$\triangle\chi \equiv \nabla \cdot \nabla\chi = \nabla \cdot \vec{V} \tag{2.20}$$

Díky tomu, že je rekonstrukce definována jako Poissonův problém, vzniká výhoda oproti ostatním polygonizačním metodám a to taková, že je celý povrch uvažován najednou. V důsledku toho je touto metodou vytvořen hladký povrch, který dobře aproximuje zašuměný vstup. Ale vzhledem k tomu, že ideální RBF[10] funkce jsou podporovány globálně a nerozkládají se pomocí parciálních derivací, Poissonův problém připouští hierarchii lokálně podporovaných funkcí. Díky tomu je řešení redukováno na dobře určený řídký lineární systém.

[10]Radial Basis Function

Mnohdy je hodnota implicitní funkce omezena pouze blízko bodů a v důsledku toho může rekonstruovaný povrch obsahovat rušivé plochy vzdálené od těchto bodů. Tento jev nastává, protože gradient implicitní funkce je omezen ve všech bodech. Gradient je omezen na nulu pro vzdálené body. Jev může být oslaben přidáním pomocných bodů, které nepatří k povrchu. Znázornění tohoto jevu je na obrázku 2.20.



Obrázek 2.20: Vlevo původní mračno bodů, vpravo mračno rekonstruované Poissonovou metodou. Mimo hladce rekonstruovaný model ruky jsou zde jasně vidět rušivé plochy.

Prvním krokem implementace je vybrání prostoru funkcí, ve kterém se problém diskretizuje. Nejpříhodnějším přístupem je začít s 3D mřížkou. Ale tato struktura je příliš uniformní a pro detailní rekonstrukci povrchu nepraktická. Protože přesná reprezentace funkce je nutná pouze blízko rekonstruovaného povrchu, je možné použít adaptivní oktalový strom. Konkrétně je použita pozice bodu k definování oktalového stromu $\mathcal{O}$ a přiřadíme funkci F_o každému uzlu $o \in \mathcal{O}$ tak, aby byly splněny tyto požadavky:

- vektorové pole $\vec{V}$ může být přesně a efektivně reprezentováno jako součet funkcí F_o
- matice reprezentující Poissonův problém vyjádřena jako F_o může být efektivně vyřešena
- reprezentace indikační funkce jako suma F_o může být přesně a efektivně vyhodnocena poblíž povrchu modelu

Množina bodů S a maximální hloubkou stromu D definuje strom $\mathcal{O}$ jako minimální oktalový strom, s takovou vlastností, že veškeré vzorky z S jsou v uzlech s hloubkou D. Pro každý uzel $o \in \mathcal{O}$ definujeme funkci F_o ve středu uzlu o a zvětšenou o velikost o.

$$F_o(q) \equiv \left(\frac{q - o.c}{o.w} \right) \frac{1}{o.w^3}, \tag{2.21}$$

kde $o.c$ a $o.w$ jsou střed a šířka uzlu o.

Výsledný aproximovaný povrch $\partial\tilde{M}$ získáme vybráním isohodnoty a extrahováním isopovrchu z vypočtené indikační funkce.

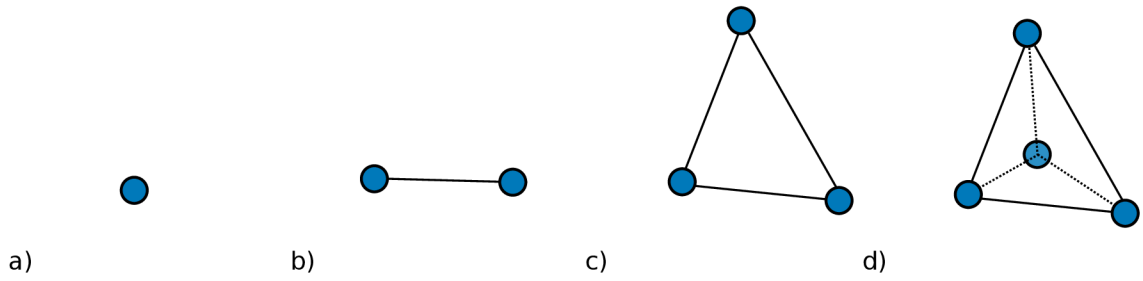
$$\partial\tilde{M} \equiv \left\{q \in \mathbb{R}^3 \,|\, \tilde{\chi}(q) = \gamma\right\}, \gamma = \frac{1}{|S|}\sum_{s\in S}\tilde{\chi}(s.p) \tag{2.22}$$

Rekonstrukce povrchu je pak provedena adaptací metody Marching Cubes[11] na oktalový strom.

2.5.4 Delaunayho triangulace

Tato metoda je zástupcem rekonstrukce využívající geometrického přístupu. Metoda je vhodná pro polygonizaci uzavřených povrchů. Experimenty jsem s ní neprováděl, je však vhodné uvést alespoň základní myšlenku tohoto algoritmu, jež se řadí do skupiny polygonizačních metod zkoumajících geometrii povrchu.

Delaunayho triangulace je založena na takzvaných simplexech, což jsou nejmenší nevyplněné objekty v dané dimenzi. Každý n-simplex se skládá z $n+1$ simplexů z $n-1$ prostoru. Simplexy až pro tři dimenze jsou znázorněny na obrázku 2.21. Důležitou vlast-

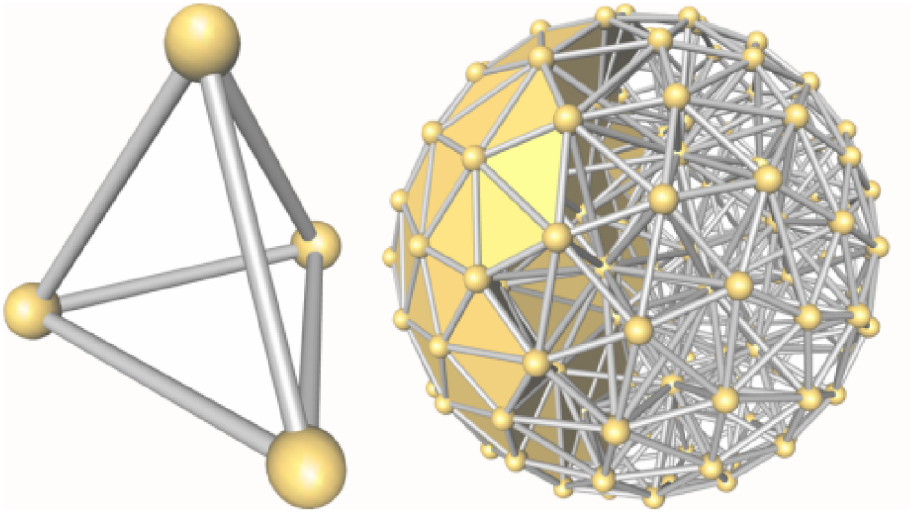


Obrázek 2.21: a) 0 simplex - bod b) 1 simplex - úsečka c) 2 simplex - trojúhelník d) 3 simplex - čtyřstěn

ností Delaunayovy triangulace je to, že koule (uvažujeme-li 3D prostor) kolem simplexu neobsahuje jiné body než ty, které tvoří uvažovaný simplex.

Existuje několik postupů, jak vytvořit polygonální povrch ze vstupního mračna [17]. Příkladem může být takový postup, kdy se zvolí inicializační simplex. Postupně jsou pak brány nejbližší body ze vstupního mračna a simplex je takto rozšiřován - změněn na reprezentaci povrchu, dokud nejsou zpracovány všechny vstupní body a tak tedy vznikne konvexní obálka. Simplexy tedy mohou v 3D prostoru sdílet bod, hranu nebo stěnu. Jak může vypadat tato triangulace vidíme na obrázku 2.22.

[11] http://paulbourke.net/geometry/polygonise/



Obrázek 2.22: Postup vytvoření konvexní obálky mračna bodů pomocí Delaunayho triangulace. Vlevo vidíme simplex v 3D prostoru. Vpravo je částečně polygonizované mračno bodů. Obrázek je převzat z dokumentace knihovny CGAL [7].

2.6 Knihovna PCL

Implementační část této práce je založena na knihovně PCL[12], proto je v této sekci čtenáři přiblížena. Jedná se o opensource knihovnu, která vznikla jako součást knihovny ROS[13]. V roce 2010 se oddělila jako samostatný projekt a je nadále vyvíjena. Knihovna se zaměřuje na zpracování 2D a 3D dat. Projekt je podporován jak komunitou, tak množstvím společností a univerzit, které ji následně používají při výzkumu. Knihovna je licencována BSD licencí, takže mimo výzkum je také určena pro komerční využití.

Knihovna je velmi dobře navržena, co se struktury týče a je také kvalitně dokumentována. Na oficiálních stránkách[14] projektu je mnoho návodů, které ukazují filozofii používání knihovny a některé algoritmy jsou zde detailně popsány. Ve většině případů je v dokumentaci algoritmu odkaz na teoretický podklad, podle kterého byl algoritmus implementován. Knihovna obsahuje několik modulů pro počítačové zpracování obrazu (rozpoznávání, segmentace, vizualizace, ...) [23].

V současné době je stabilní vydání ve verzi 1.8. Vývoj je stále aktivní, nicméně je zaměřen spíše na opravu chyb v existujících implementacích metod.

Mezi informace, které struktura mračna v knihovně PCL obsahuje, patří [19]:

- width - šířka mračna
- height - výška mračna (v případě neorganizovaného mračna se rovná jedné)
- points - matice bodů, obsahující informace o jejich hloubce, případně barvě
- is_dense - informace o tom, zda mračno může obsahovat hodnoty NaN a Inf
- sensor_origin_ - specifikuje změnu polohového vektoru snímače

[12] Point Cloud Library
[13] http://wiki.ros.org/Client%20Libraries
[14] http://pointclouds.org/

- sensor_orientation_ - specifikuje změnu v orientaci snímače

Mračna bodů mohou být uložena do souboru. Rozšířeným formátem pro ukládání mračen je formát `PCD`. Existují další formáty jako `PLY` nebo `VTK`. Soubory lze ukládat buďto v `ASCII` formátu čitelném člověkem nebo v binární podobě. Níže je popsán formát souboru typu `PCD`.

Pole začínající klíčovým slovem jsou povinná a je nutné dodržet jejich pořadí. Význam jednotlivých klíčových slov je uveden v následujícím výčtu [21].

- VERSION - specifikuje verzi PCD formátu, ve které je soubor uložen.
- FIELDS - výčet polí, které jsou v souboru zapsány. Tato položka může nabývat například hodnoty `x y z` pro uložení mračna bodů bez barevné informace. Pro uložení včetně RGB hodnot bodů je použita hodnota `x y z rgb`
- SIZE - velikost každé hodnoty v bytech.
- TYPE - určuje, jaký datový typ mohou hodnoty nabývat
  - I - datový typ `integer`
  - U - datový typ `unsigned integer`
  - F - datový typ `float`
- COUNT - počet prvků položky. Například souřadnice `x` má pouze jeden prvek, ale histogram použitý v algoritmu FPFH má 33 prvků, tudíž pro uložení tohoto histogramu by bylo třeba hodnotu upravit.
- WIDHT - stejný význam jako v struktuře mračna bodů - šířka ukládaného mračna
- HEIGHT - stejný význam jako v struktuře mračna bodů - výška ukládaného mračna (opět platí, že výška neorganizovaného mračna je rovna jedné)
- VIEWPOINT - souřadnice bodu, ze kterého je mračno sejmuto
- POINTS - určuje počet bodů v mračnu
- DATA - nabývá hodnot `ascii` nebo `binary` v závislosti na tom, zda má být soubor uložen v textové nebo v binární podobě.
- `data` - hodnoty vybraných polí jednotlivých bodů

Můžeme si všimnout, že hodnoty RGB polí jsou zakódovány do jednoho `float` záznamu. Kódování a dekódování je provedeno prostým bitovým posunem [2]. Hodnoty x, y a z jsou udány v metrech.

```
// pack r/g/b into rgb
uint8_t r = 255, g = 0, b = 0;    // Example: Red color
uint32_t rgb = ((uint32_t)r << 16 | (uint32_t)g << 8
    | (uint32_t)b);
p.rgb = *reinterpret_cast<float*>(&rgb);
```

Obrázek 2.24: Kódování hodnot RGB polí do jedné `float` hodnoty.

```
# .PCD v.7 - Point Cloud Data file format
VERSION .7
FIELDS x y z~rgb
SIZE 4 4 4 4
TYPE F F F F
COUNT 1 1 1 1
WIDTH 3
HEIGHT 1
VIEWPOINT 0 0 0 1 0 0 0
POINTS 3
DATA ascii
0.93773 0.33763 0 4.2108e+06
0.90805 0.35641 0 4.2108e+06
0.81915 0.32 0 4.2108e+06
```

Obrázek 2.23: Ukázkový PCD soubor zobrazující hlavičku pro ukládání dat a data tří bodů.

```
PointXYZRGB p;
// unpack rgb into r/g/b
uint32_t rgb = *reinterpret_cast<int*>(&p.rgb);
uint8_t r = (rgb >> 16) & 0x0000ff;
uint8_t g = (rgb >> 8)  & 0x0000ff;
uint8_t b = (rgb)       & 0x0000ff;
```

Obrázek 2.25: Dekódování hodnot RGB polí z jedné `float` hodnoty.

Kapitola 3

Existující řešení pro 3D skenování místností

V oblasti rekonstrukce povrchu z mračna bodů existuje několik řešení. Mezi nejzdařilejší pak patří Kinect Fusion přímo od společnosti Microsoft. Použitá knihovna PCL má vlastní implementaci Kinect Fusion (někdy zkráceně KinFu), která je opensource a je dostupná v repozitáři[1]. Tato řešení umožňují rekonstrukci povrchu a to v téměř reálném čase díky tomu, že k výpočtům používají grafickou kartu. Mezi komerční řešení pak patří například Skanect[2]. Tato aplikace umožňuje rekonstrukci povrchu včetně textury a to offline (opak reálného času). Skanect je poskytován ve dvou licencích. Verze zdarma je různě omezena. Pořizovací cena plné verze je 129$. Dalším opensource řešením je knihovna RGBD-SLAM[3]. Tato knihovna dokáže spojovat snímaná mračna taktéž téměř v reálném čase, ale už neposkytuje rekonstrukci povrchu.

Na obrázku 3.1 vidíme jak vypadá rekonstruovaný povrch Kinectem nasnímané místnosti pomocí aplikace Kinfu[4]. Podobně na obrázku 3.2 je scéna rekonstruovaná programem Skanect.

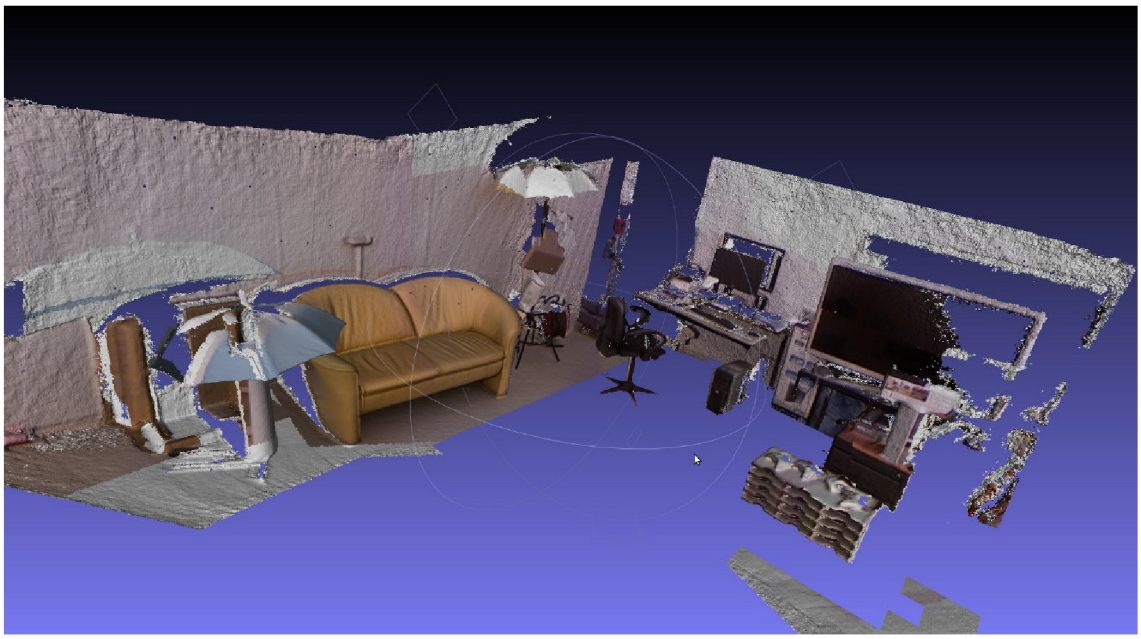
Aplikace Skanect poskytuje robustní práci s texturami a materiálem výsledného polygonálního modelu. Nevýhodou je použití čistého algoritmu ICP. To je však jen má domněnka. Soudím z toho, že je scéna nahrávána plynulým pohybem senzoru. Projekt není opensource, takže nelze říct s jistotou, jakých metod je použito k zarovnání snímků.

Porovnání aplikací Skanect a Kinfu nalezneme v tabulce 3.1.

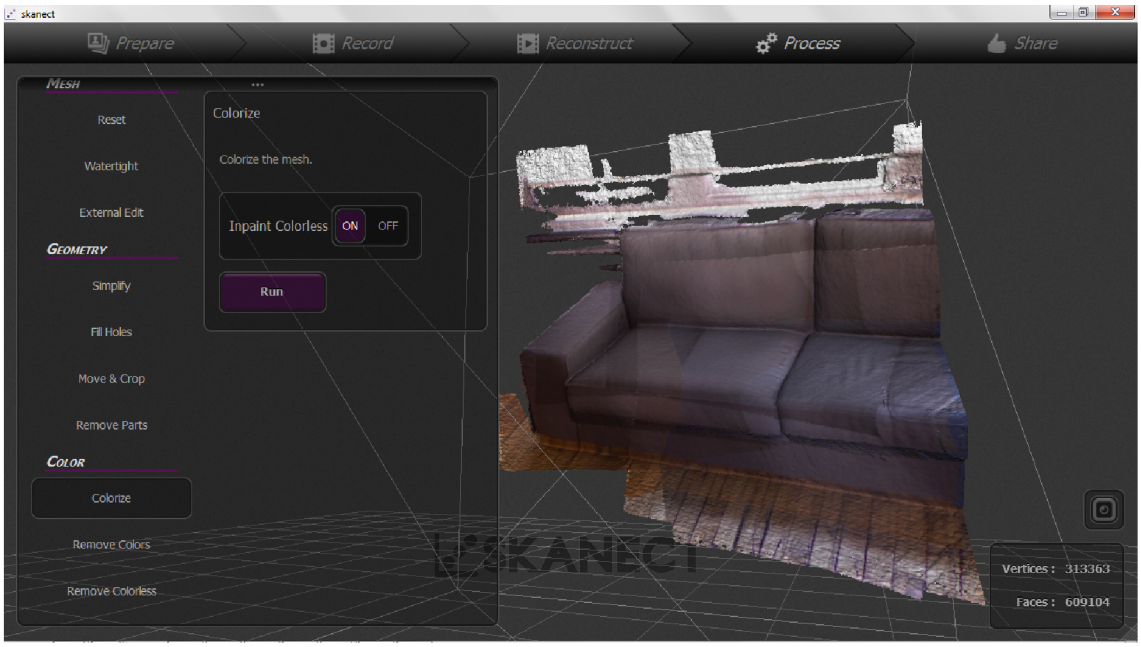
Aplikace	Realtime	Metoda zarovnání	Podpora GPU	Textura	Cena
Skanect	ne	ICP	ano	ano	129$
Kinfu	ano	camera tracking	ano	implicitně ne, lze	zdarma

Tabulka 3.1: Porovnání aplikací Skanect a Kinfu.

[1] https://github.com/PointCloudLibrary/pcl/tree/master/gpu/kinfu
[2] http://skanect.occipital.com/download/
[3] https://openslam.org/rgbdslam.html
[4] http://pointclouds.org/documentation/tutorials/using_kinfu_large_scale.php



Obrázek 3.1: Výstup aplikace Kinfu. Obrázek je převzat z dokumentace knihovny PCL.

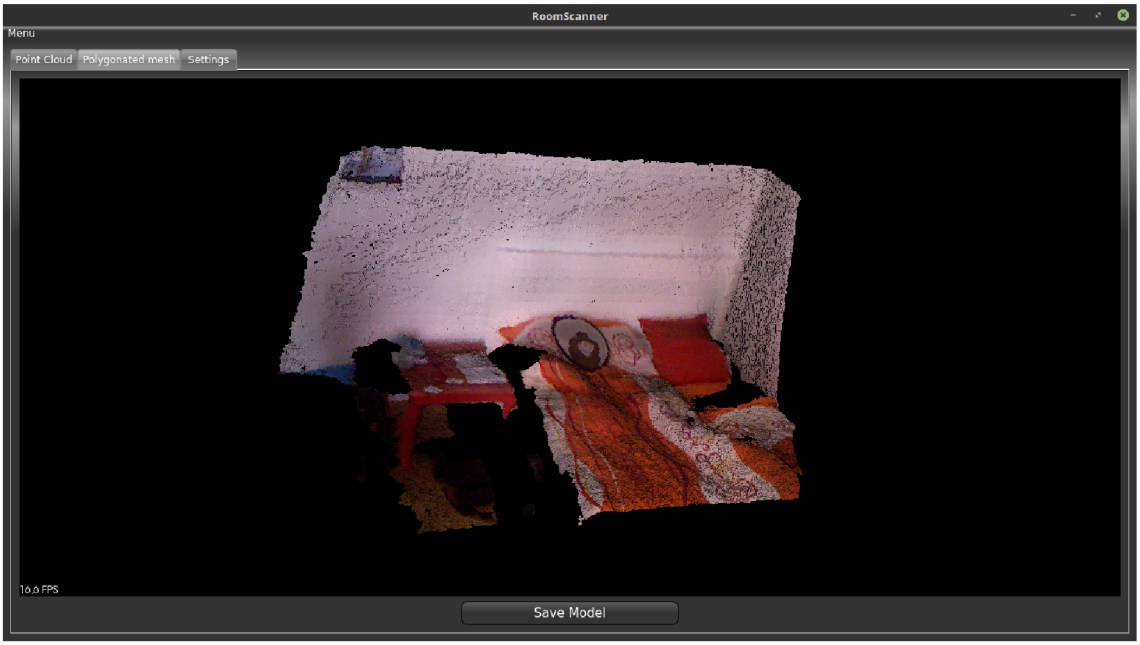


Obrázek 3.2: Výstup aplikace Skanect. Obrázek je převzat z http://jordanpelovitz.com/?p=1180.

Kapitola 4

Návrh skeneru místností s využitím MS Kinect

V této kapitole je popsán návrh schématu aplikace tak, aby byla schopná spojovat sejmutá data a následně rekonstruovat polygonální povrch to vše v grafickém rozhraní. Výsledek tohoto návrhu je možno vidět na obrázku 4.1. Aplikace nese název RoomScanner.



Obrázek 4.1: Výsledný vzhled implementace navržené aplikace.

4.1 Návrh aplikace

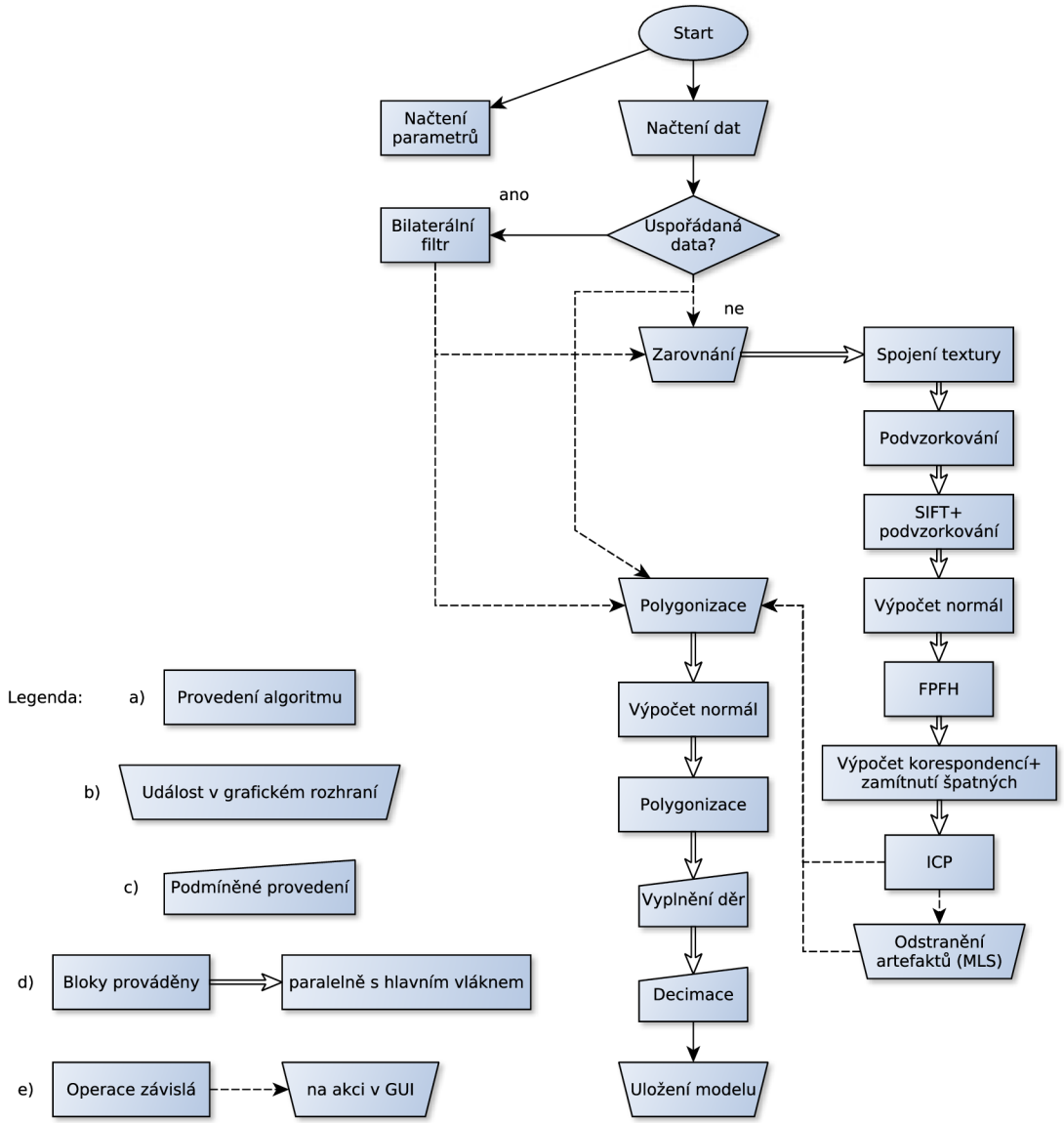
Rozhodl jsem se pro implementaci aplikace s grafickým prostředím. Základní požadavky na funkcionalitu této aplikace jsou:

- získání dat ze senzoru Kinect nebo ze souboru

- zarovnání a spojení těchto snímků
- offline rekonstrukce polygonálního povrchu
- obarvený hladký povrch na výstupu
- obrázek použitelný pro tvorbu textury vysoké kvality

To vše za takového omezení, že data jsou sejmutá z jednoho bodu (senzorem se nepohybuje, pouze rotuje). Vstupní data mohou být získána buďto senzorem Kinect a nebo nahrána ze souborů. Pokud jsou tato data ze senzoru Kinect (jsou uspořádaná), je na ně aplikován bilaterální filtr (popsán v sekci 2.3.1). Pokud jsou nahrána zašuměná data ze souboru, lze na ně volitelně aplikovat filtr MLS (popsán v sekci 2.3.2). Poté už může být spuštěno zarovnávání nebo polygonizace povrchu. Zavrovnávání může být zřejmě spuštěno pouze pokud máme dva a více vstupních snímků. Výsledek i vstupní data si lze libovolně prohlížet v grafickém prostředí aplikace. Veškeré parametry algoritmů lze nastavit v grafickém prostředí nebo přítomností konfiguračního souboru ve složce, kde je umístěn sestavený program. Celý diagram aplikace můžeme vidět na obrázku 4.2.

Výstupem aplikace by měl být rekonstruovaný hladký povrch uložen ve formátu `obj` a textura ve formátu `jpg`. Od původního návrhu se aplikace liší v tom, že jsem zavrhl segmentaci modelu na jednotlivé části, jelikož toto může být jednoduše provedeno v různých aplikacích (Blender, 3DS Max, Cinema4D,...), které importují model vystupující z této aplikace. Naopak co oproti původnímu návrhu přibylo je exportování textury do souboru.



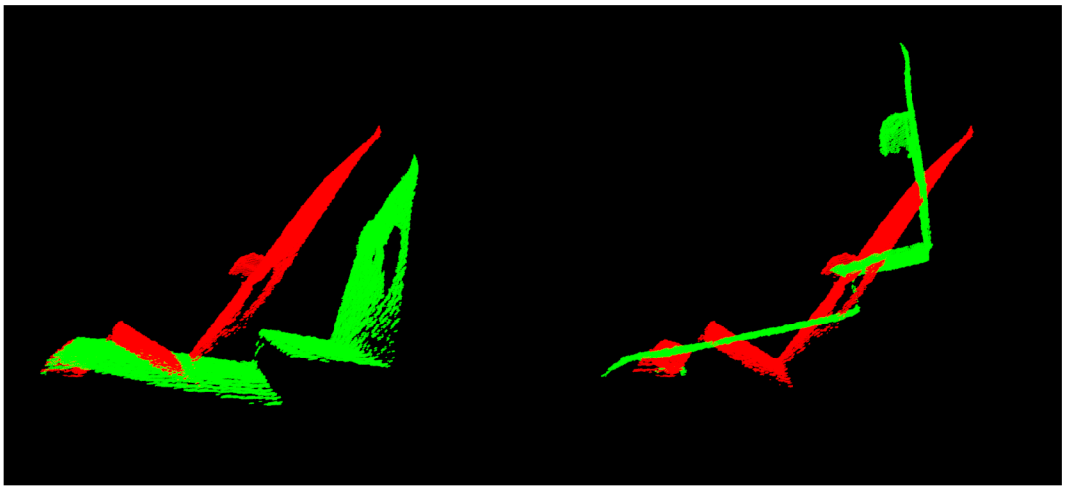
Obrázek 4.2: Zjednodušený diagram celé aplikace. Objasnění legendy: a) Algoritmus je bezprostředně proveden. b) Uživatel klikl na tlačítko. c) Pokud je v nastavení povoleno použití daného algoritmu, použije se, jinak se blok přeskočí. d) Operace je náročná na výpočet, proto je spuštěna v novém vláknu a hlavní vlákno zobrazuje načítací obrazovku. d) Uživatel klikl na tlačítko. Provede se operace v závislosti na tom, které tlačítko to bylo.

4.2 Vylepšení zarovnání n snímků

Mnohé implementace nástrojů pro skenování místností pracují na plné frekvenci snímače (např. Kinect pracuje na frekvenci 30Hz tedy 30 snímků za vteřinu). Scéna je pak snímána plynulým pohybem. Takto sejmutá data jsou postupně zpracovávána algoritmem ICP (sekce 2.4.3). Výhodou tohoto přístupu je, že dva následné snímky obsahují velké množství spo-

lečných bodů a je vysoce pravděpodobné, že se snímky zarovnají správně. Nevýhodou je velké množství takto zpracovávaných snímků a tím i delší doba zpracování.

Při nedostatečném překrytí zarovnávaných snímků může dojít k tomu, že algoritmus ICP zarovnávání ukončí v lokálním minimu. Tento případ je zobrazen na obrázku 4.3.



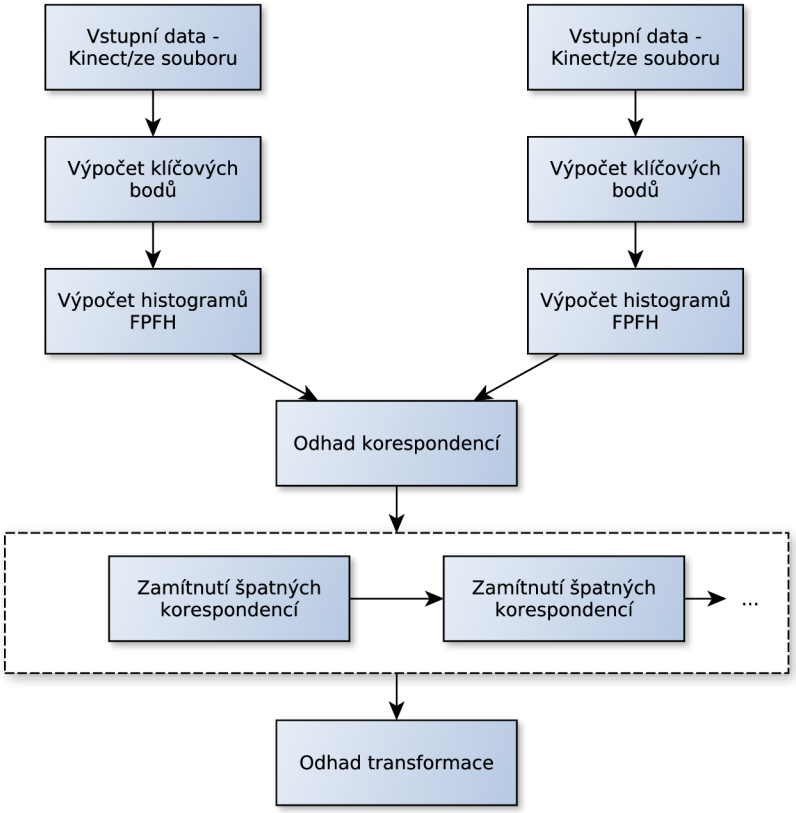
Obrázek 4.3: Vlevo jsou původní mračna. Vpravo je pak případ, kdy algoritmus ICP uvázne v lokálním minimu a zarovnání není provedeno správně.

Abych se tomuto vyhnul, rozhodl jsem navrhnout jiný přístup a to takový, že snímků je sejmuto jen zlomek. Z těchto snímků jsou extrahovány speciální body zájmu, takzvané klíčové body. V praxi se často jako klíčové body používá původní mračno bodů, které je hrubě podvzorkováno. Zdi místnosti však nedisponují přílišnou změnou hloubkové informace, ale mohou obsahovat náhlou změnu barevné informace (např. zavřené dveře, pověšený obraz, ...). Proto jsem jako klíčové body použil podvzorkované vstupní mračno sjednocené s výstupem algoritmu SIFT (sekce 2.4.1). Takto získané klíčové body zaručují dostatečný popis vstupních dat.

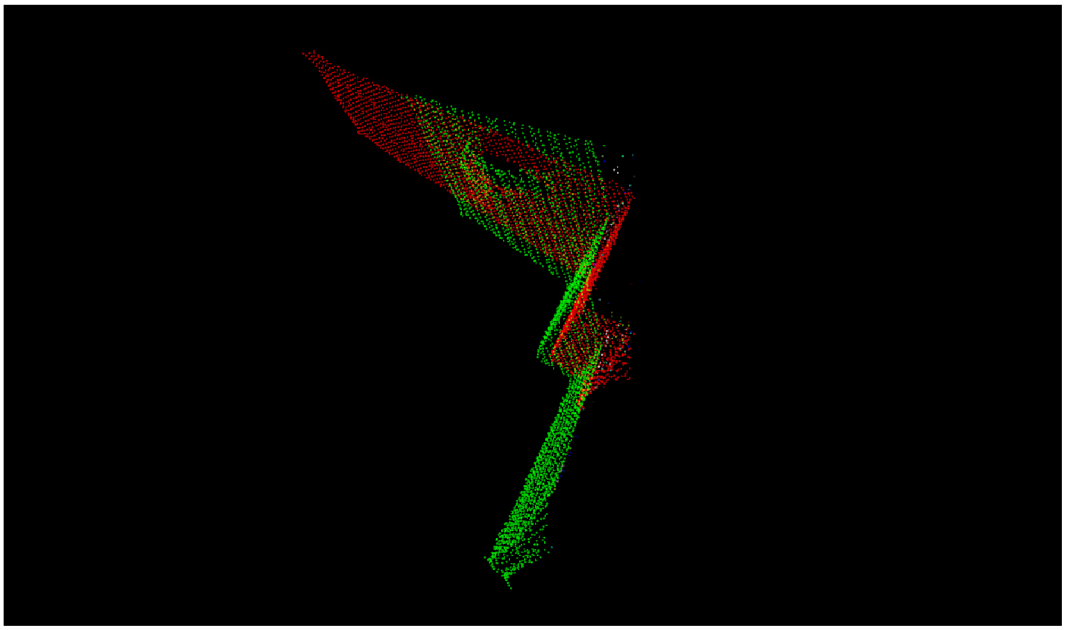
Pro klíčové body jsou následně vypočteny histogramy vlastností FPFH (sekce 2.4.2). Toto je provedeno jak pro referenční snímek, tak pro snímek, který se má zarovnat. Z histogramů FPFH těchto snímků je zjištěna množina jejich možných korespondencí. Z této množiny zamítneme korespondence se špatným ohodnocením (velká vzdálenost, ...). Zbude nám množina korespondencí s dobrým ohodnocením a z této množiny je proveden odhad transformace mezi těmito korespondencemi. Totožná transformace je použita na zarovnávaný snímek.

Takto získáme odhad zarovnání snímku. Tento odhad však nemusí být přesný, jak můžeme vidět na obrázku 4.5. Postup zpracování je zobrazen na obrázku 4.4. Pro přesné konečné zarovnání je použit algoritmus ICP, vizte obrázek 4.6.

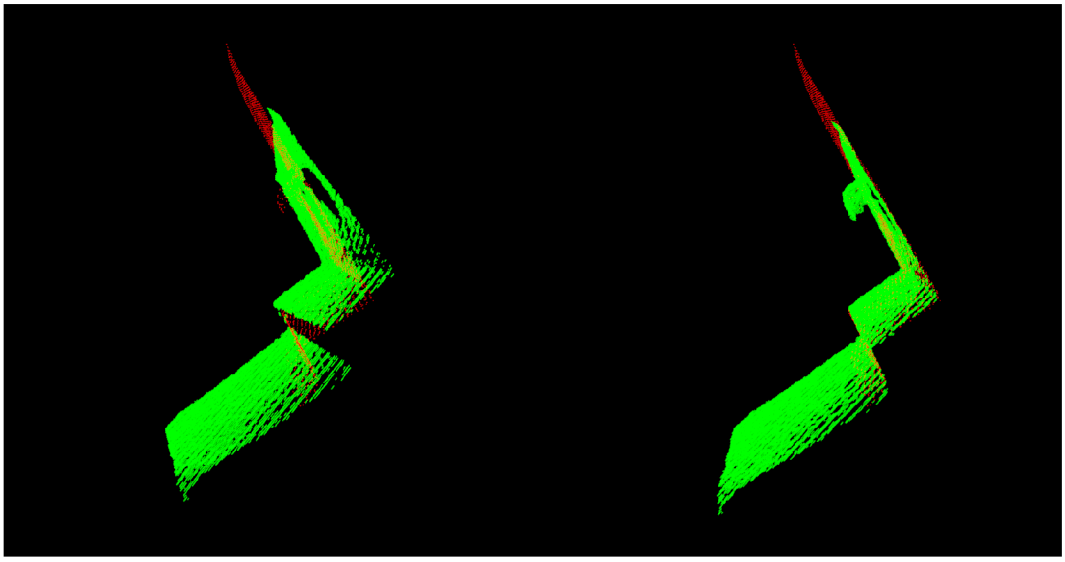
Tento přístup získává výhodu toho, že pokud jsou odhadnuté korespondence dobré a jsou blízko sebe, algoritmus ICP konverguje ve výsledné zarovnání rychle. Je však velice důležité správné nastavení parametrů pro výpočet korespondencí pomocí FPFH a také je důležité mít vhodně zvolené parametry pro detekci klíčových bodů, ze kterých jsou tyto FPFH histogramy vypočteny. Správné zarovnání je tedy silně závislé na vstupních datech, ale i na správném nastavení parametrů.



Obrázek 4.4: Zobrazení postupu pro odhad transformace mezi dvěma snímky. Můžeme použít několik ohodnocovacích metod pro zamítnutí nesprávných korespondencí, což je zde znázorněno jako čárkovaný blok.



Obrázek 4.5: Odhad provedený pomocí FPFH histogramů bodů vstupních snímků. Je vidět, že odhad je lepší než za použití čistého ICP algoritmu, ale není přesný.

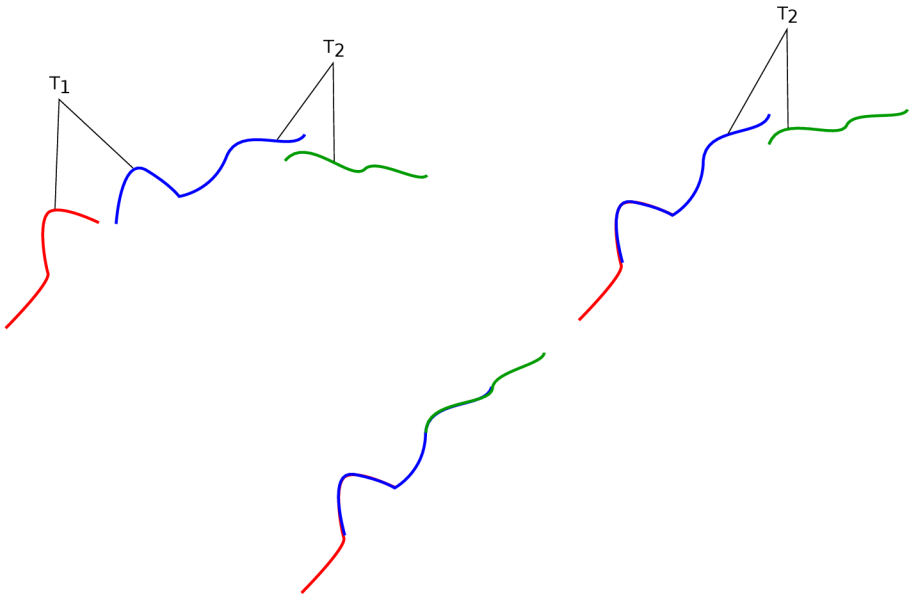


Obrázek 4.6: Vlevo vidíme vstup upravený pomocí odhadu počáteční pozice z FPFH histogramů mračen. Vpravo pak vidíme finální zarovnání algoritmem ICP.

Takto je tedy vypočtena transformace mezi dvěma snímky. Pro spojení n snímků je tento postup opakován pro každou dvojici. Je nutno uvažovat to, že $n+2$-tý snímek musí být navíc transformován transformací vypočtenou z n a $n+1$-tého snímku. Toho je dosaženo tak, že je vytvořena globální transformace, jež má jako inicializační hodnotu matici identity. Tato globální hodnota je pak násobena vypočtenou transformací a zarovnávaný snímek následující dvojice je touto transformací pozměněn. Je nutno dodat, že celková transformace je tvořena dvěma maticemi. První z nich je získána odhadem FPFH histogramů a druhá z doladění zarovnání pomocí ICP. Zjednodušený případ vidíme na obrázku 4.7.

Jelikož výpočet transformace dvou snímků je operací binární, nabízí se několik postupů, jak přistoupit k implementaci. Jedním postupem je pro každou dvojici vytvořit vlákno a výpočty provádět paralelně. Pak je ale složitější delegace transformací. Proto jsem zvolil sekvenční metodu, kdy se transformace jednoduše postupně skládají.

Nejužším místem zarovnání dvou snímků je právě algoritmus ICP. Značným urychlením by bylo provádění výpočtu těchto transformací na grafické kartě (CUDA, implementace např. projekt[1]), což by mohlo vést k real-time zpracování vstupu. Výslednou aplikaci jsem však částečně vyvíjel na počítači bez grafické karty, proto jsou veškeré výpočty prováděny na procesoru.



Obrázek 4.7: Zarovnání n snímků. Nahoře vlevo vidíme vstupní data (zjednodušený případ), kde je vypočtena transformace T_1. Tato transformace uvažuje červenou křivku jako cíl a modrá křivka je zarovnávána. Transformace T_2 je pak vypočtena z modré a zelené křivky. Pro výslednou transformaci musí být na zelenou křivku aplikována transformace T_1 (případ vpravo nahoře) i T_2 (případ dole).

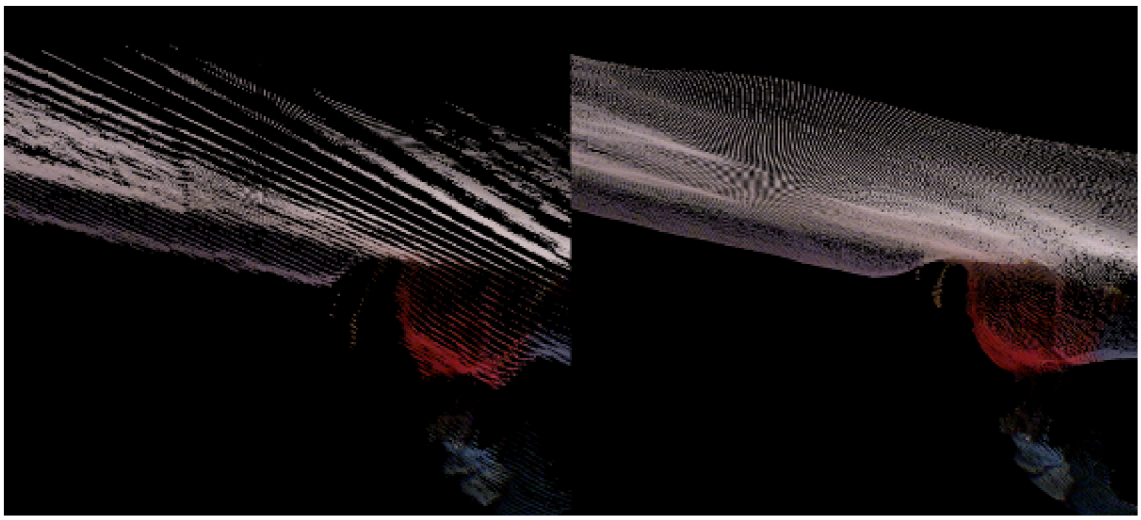
[1] https://github.com/mp3guy/ICPCUDA

4.3 Hybridní rekonstruování povrchu

V sekci 2.5 je popsáno několik metod rekonstrukce povrchu. Některé z nich jen jednoduše vytváří povrch spojováním bodů a některé se snaží co nejlépe aproximovat zašuměný povrch. To však může vést k vyšší časové náročnosti na výpočet. Proto jsem se rozhodl navrhnout „hybridní“ rekonstrukci povrchu. Návrh spočívá v tom, že upravujeme vstupní mračno bodů tak, aby co nejlépe aproximovalo povrch a zároveň bylo možno použít jednodušší metody pro vytvoření polygonálního povrchu.

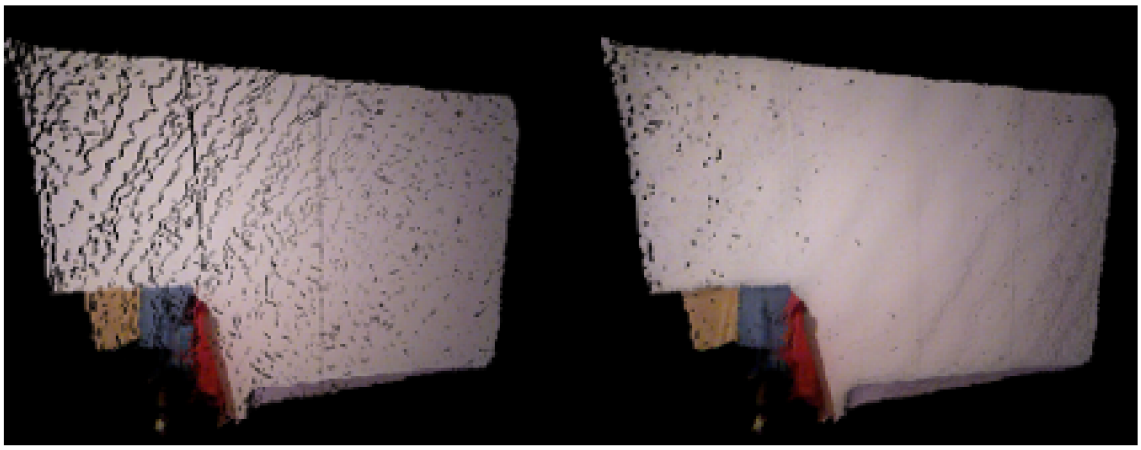
Jako nejvhodnější kandidáti na použití se ukázaly metody bilaterálního filtru (sekce 2.3.1) a MLS (sekce 2.3.2) pro vyhlazení vstupních dat. Vytvoření polygonálního povrchu pak používá algoritmus Greedy Projection Triangulation (sekce 2.5.1). Nicméně experimenty jsem prováděl i s jinými metodami polygonizace. Tyto metody jsem ponechal implementovány v aplikaci a uživatel si může zvolit, v závislosti na druhu vstupních dat, kterou metodu použít.

Na obrázku 4.8 vlevo vidíme, jak vypadá původní snímek ze senzoru Kinect. Vpravo je pak tento smímek vyhlazen bilaterálním filtrem. Na každý snímek je aplikován algoritmus Greedy Projection Triangulation se stejnými parametry. Na obrázku 4.9 vidíme znatelný rozdíl v kvalitě rekonstruovaného povrchu.

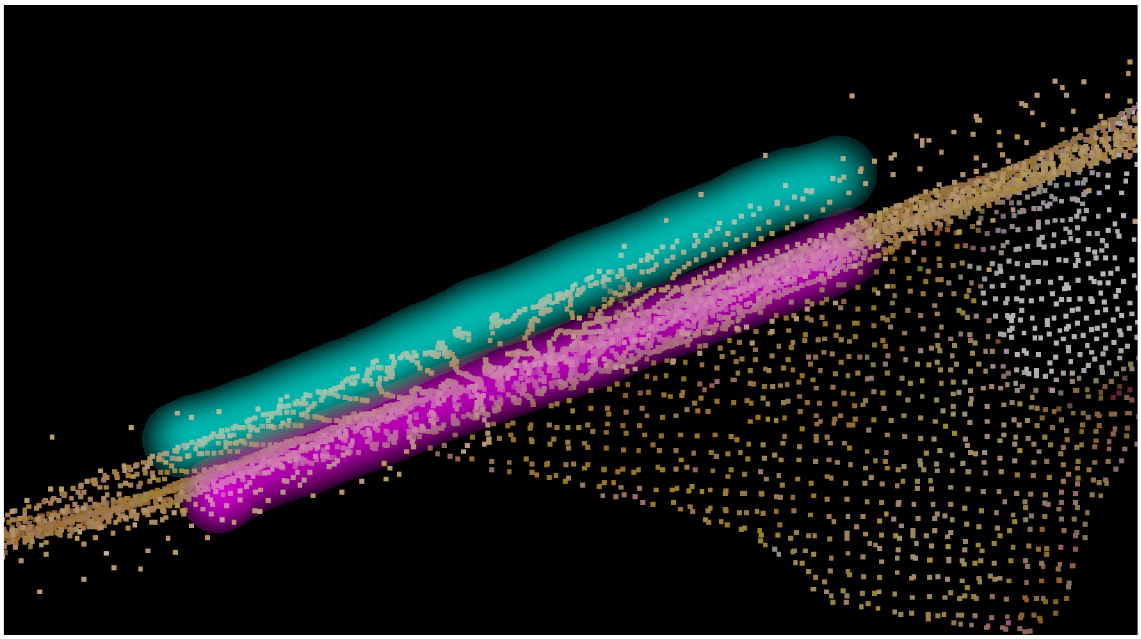


Obrázek 4.8: Vyhlazení snímku bilaterálním filtrem.

Problém, který plyne z principu bilaterálního filtru je, že body nezachovávají původní pozici. To není problém u rekonstrukce povrchu jediného snímku. Problém nastává po zarovnání několika snímků. Mezi snímky vznikají artefakty, které se snaží algoritmus rekonstrukce povrchu chybně spojit. Tyto artefakty jsou zobrazeny na obrázku 4.10. Rekonstruovaný povrch pak trpí těmito artefakty, jak je znázorněno na obrázku 4.11. Abychom jich zbavili, je mračno vyhlazeno filtrem MLS. Rekonstruovaný povrch takto upraveného mračna je zobrazen na obrázku 4.12.



Obrázek 4.9: Vlevo polygonizace nevyhlazeného snímku. Vpravo polygonizace snímku vyhlazeného bilaterálním filtrem.



Obrázek 4.10: Artefakty vznikající po spojení snímků vyhlazených bilaterálním filtrem. Zelenomodře označeny body jednoho mračna, růžově druhého mračna.

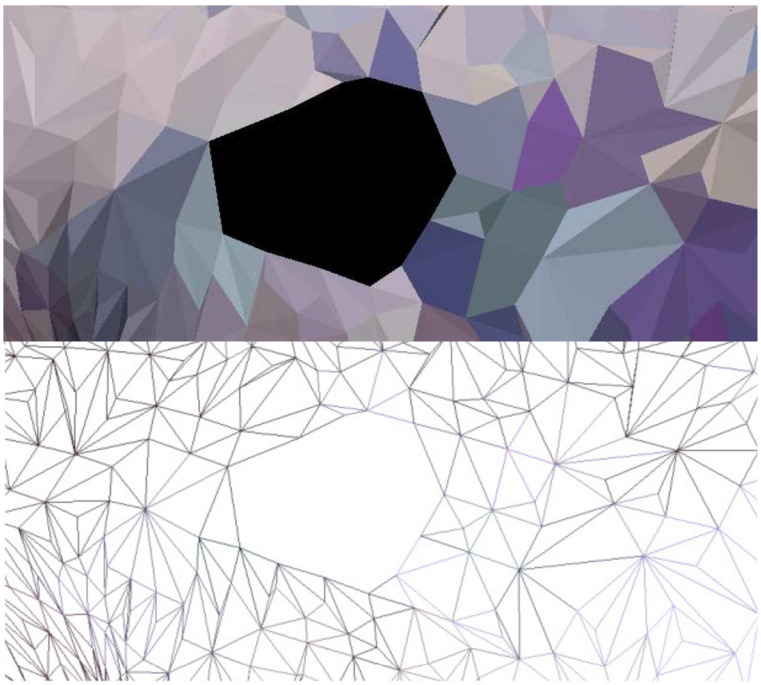
Obrázek 4.11: Povrch rekonstruovaný z mračna obsahujícího artefakty.

Obrázek 4.12: Povrch z mračna, které bylo po zarovnání vyhlazeno filtrem MLS. Rozmazání je způsobeno podvzorkováním.

4.4 Úpravy rekonstruovaného povrchu

Při nedostatečné hustotě bodů mračna může rekonstruovaný povrch obsahovat díry, což je uzavřená kružnice několika hran, čímž je porušena topologie disku. Příklad díry v povrchu vidíme na obrázku 4.13.

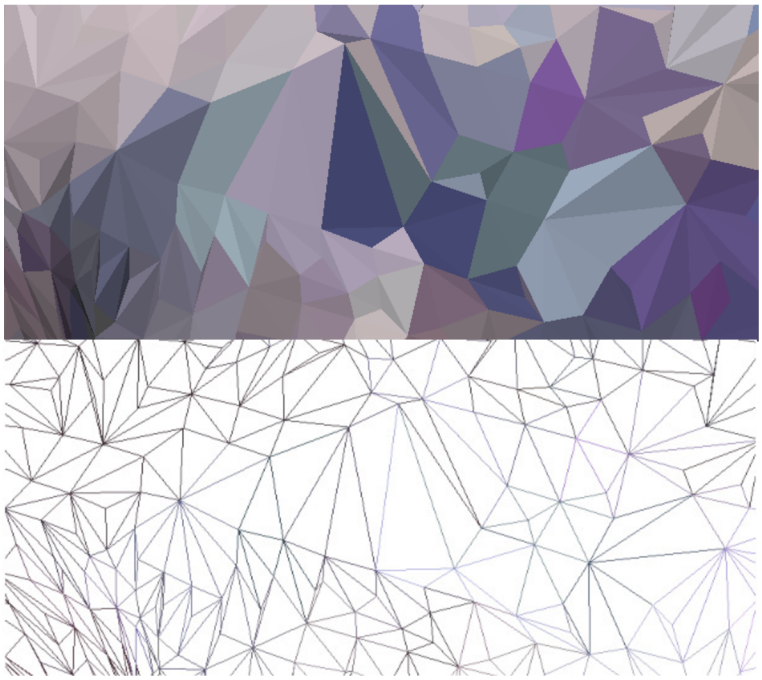
Pro odstranění tohoto jevu jsem se rozhodl použít algoritmus pro detekci a zaplnění děr [14]. Na obrázku 4.14 je pak vidět povrch po aplikování tohoto algoritmu.



Obrázek 4.13: Díra v povrchu vzniklá po rekonstrukci. Ve spodní části je znázorněna mřížka tohoto povrchu.

Po provedení rekonstrukčního algoritmu obsahuje povrch redundantní trojúhelníky. Například rovná zeď má stejnou hustotu bodů jako nerovné povrchy a po rekonstrukci tak vznikají nadbytečné trojúhelníky. Proto jsem navrhl použití algoritmu pro decimaci rekonstruovaného povrchu [15]. Nevýhodou tohoto algoritmu je, že se zcela ztrácí barevná informace povrchu. Rozdíl mezi původním a decimovaným povrchem vidíme na obrázku 4.15.

Jelikož obě tyto metody mění výslednou topologii rekonstruovaného povrchu, je v programu jejich použití volitelné pro případ, kdy uživatel chce získat povrch přímo vystupující z polygonizační metody (pro demonstraci či jiné účely).



Obrázek 4.14: Stejný povrch po detekci a zalátání díry. Ve spodní části je znázorněna mřížka tohoto povrchu.

4.5 Vytvoření textury

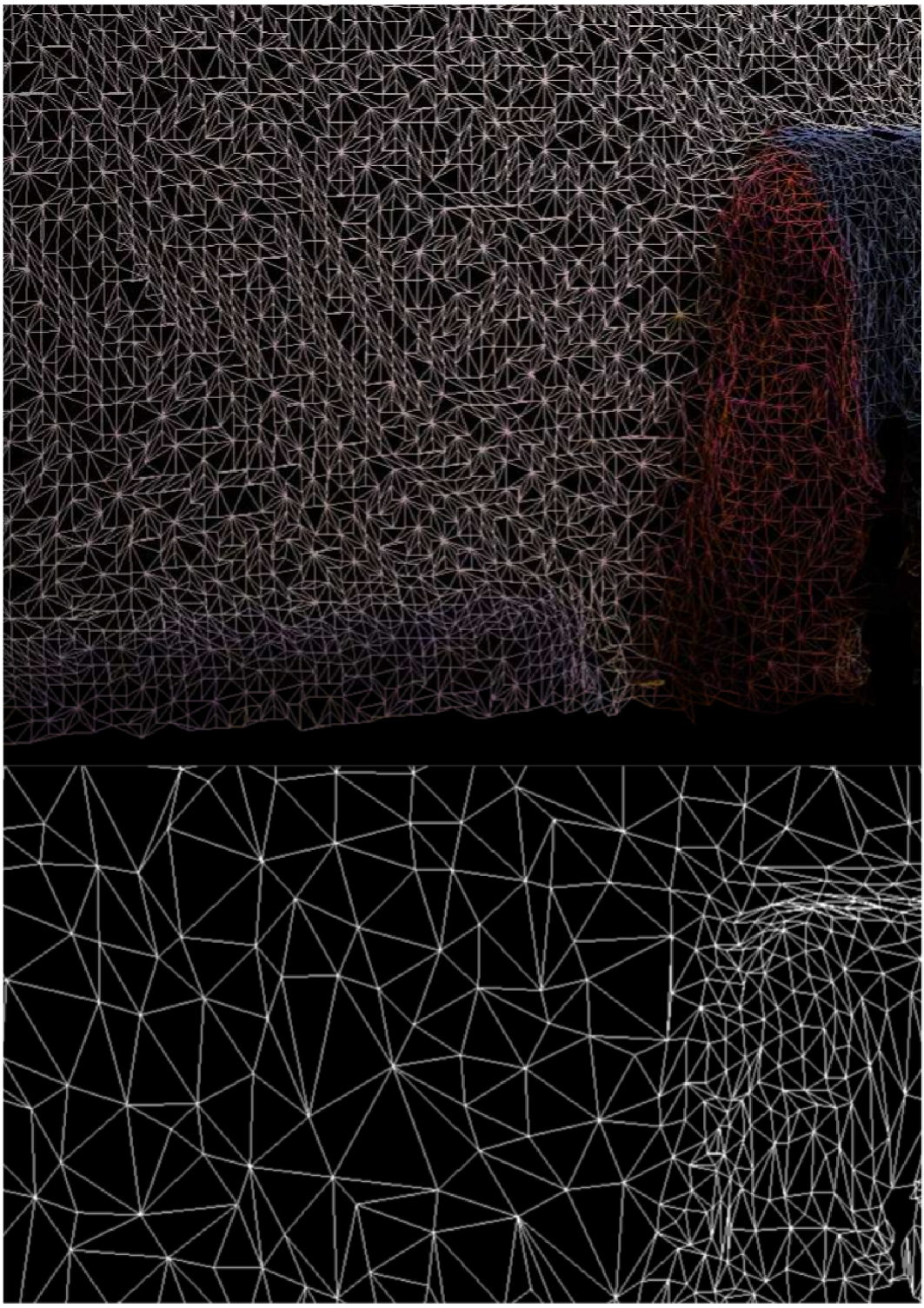
V případě, že nechceme upravit počet trojúhelníků výsledné sítě, je uložena polygonální reprezentace, kdy každý vrchol má RGB hodnoty. Barvy trojúhelníků jsou pak interpolovány.

Jak bylo popsáno v návrhu zjednodušení rekonstruovaného povrchu (sekce 4.4), ztrácí se barevná informace trojúhelníků. Jelikož hodnoty RGB jsou uloženy přímo v bodech, ze kterých je povrch rekonstruován, trpí kvalita barevné informace i při jakémkoliv podvzorkování dat. Proto jsem navrhl postup pro vytvoření obrázku, který může sloužit k vytvoření textury.

Tento návrh předpokládá na vstupu uspořádané mračno. Z jeho definice je zřejmé, že můžeme uložit matici pouze s barevnou informací, tedy obrázek. Tento obrázek je uložen před jakoukoliv operací, která by porušila uspořádanost mračna nebo kvalitu obrázku. Pro neuspořádaná mračna je tedy tento princip nevhodný, jelikož nemůžeme získat matici barevných bodů.

Postup při zarovnávání snímků jsem navrhl takový, že je z každého vstupního mračna exportován obrázek a všechny takovéto obrázky jsou spojeny v jeden kumulovaný za použití takzvaného „sešívání“. Tento postup spojí překrývající se obrázková data v celek. Tento princip je znám především z tvorby panoramat.

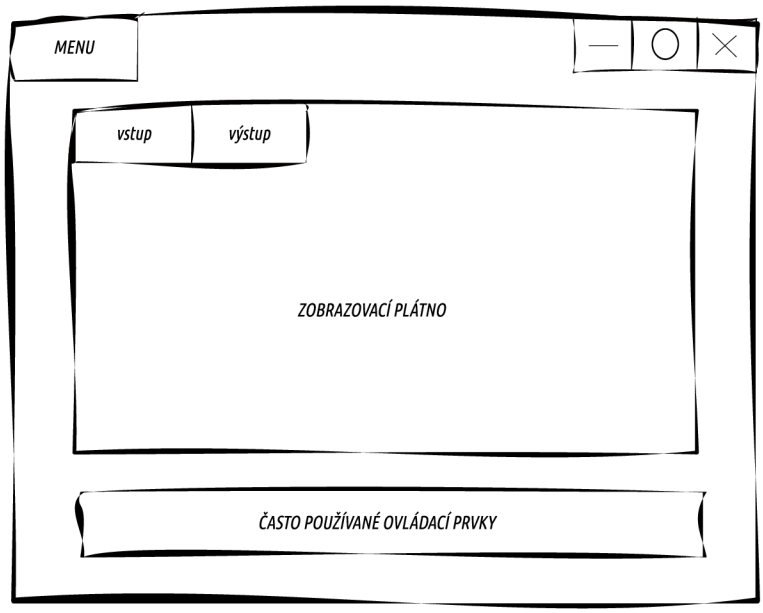
Z takto vytvořeného obrázku se přepočtem UV souřadnic vytvoří textura.



Obrázek 4.15: Decimace povrchu. Úbytek trojúhelníků je výrazný v jednolitých oblastech.

4.6 Grafické prostředí

K návrhu grafického prostředí jsem použil takzvaný mock-up. Ilustraci můžeme vidět na obrázku 4.16. Zde jsem rozvrh ovládací prvky aplikace tak, že podstatnou část okna zabírá zobrazovací plátno. Pod tímto plátnem se nachází často používaná tlačítka. Aplikace má dvě záložky. První slouží k zobrazení vstupních dat. Druhá záložka pak měla sloužit k manipulaci s již polygonizovanými daty. Oproti návrhu byla implementace rozšířena o další záložku, která obsahuje nastavení parametrů algoritmů. Aplikace dále obsahuje menu, kde se nacházejí méně často používané funkce.



Obrázek 4.16: Mock-up návrhu grafického prostředí aplikace.

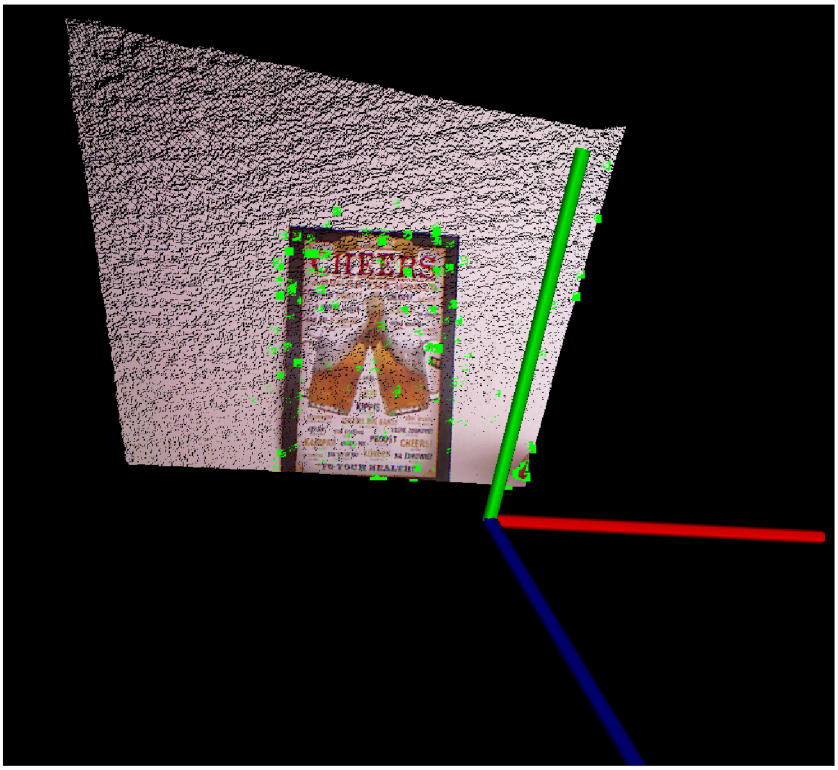
Jelikož jsou některé algoritmy výpočetně náročné a jejich provedení zabere nějaký čas, jsou tyto operace doprovázeny tématickou načítací obrazovkou v podobě rotující krychle měnící polygonizované hrany. Generátor načítacích obrázků je na adrese [2].

Může nastat situace, že by uživatel chtěl takovou operaci zrušit. Například pokud data nekonvergují správně. Navrhl jsem proto načítací obrazovku bude klikatelnou. Po kliknutí na ni bude taková operace s varováním zrušena.

Mezi náročné algoritmy patří například ICP. Aby byl uživateli znázorněn aktuální stav zarovnávání snímků, navrhl jsem, že se po každé iteraci zarovnání bude obnovovat zobrazení zarovnávané dvojice snímků.

Mezi další návrhy v grafickém prostředí patří volitelné zobrazení os souřadnicového systému a klíčových bodů algoritmu SIFT pro lepší nastavení parametrů uživatelem. Jak vypadá plátno aplikace se zapnutím těchto voleb vidíme na obrázku 4.17.

[2] https://loading.io/



Obrázek 4.17: Zapnutí voleb pro zobrazení klíčových bodů snímku (zelené body) a os souřadnicového systému.

Kapitola 5

Implementace

V této kapitole se čtenář dozví, jaké implementační metody a prostředky byly použity k realizaci návrhu popsaného v kapitole 4.

5.1 Použité programovací prostředky

Jádro navržené aplikace používá množství funkcí z knihovny PCL, která je napsána v jazyce C++. Proto jsem tento jazyk zvolil pro implementaci také. Celá aplikace byla vyvíjena na operačním sytému Linux a to na verzi Linux Mint 18 a Fedora 25.

Knihovna PCL vnitřně používá několik knihoven, jejichž přítomnost je nutná k úspěšnému sestavení implementované aplikace. Níže můžeme vidět výčet všech použitých knihoven, jejich verzí a stručný popis, k čemu byly použity.

- * boost 1.6 - výlučný přístup, multithreading, parsování json souborů
- * Eigen 3.0 - matematické struktury (matice, vektory, . . .)
- * FLANN 1.7.1 - vyhledání nejbližších sousedů
- * VTK 6.3 nebo VTK 7.1 - vizualizace vstupní a výstupních dat, operace s rekonstruovaným povrchem
- OpenNI 1.3
- PCL 1.8
- OpenCV 3.2
- Qt 5.7

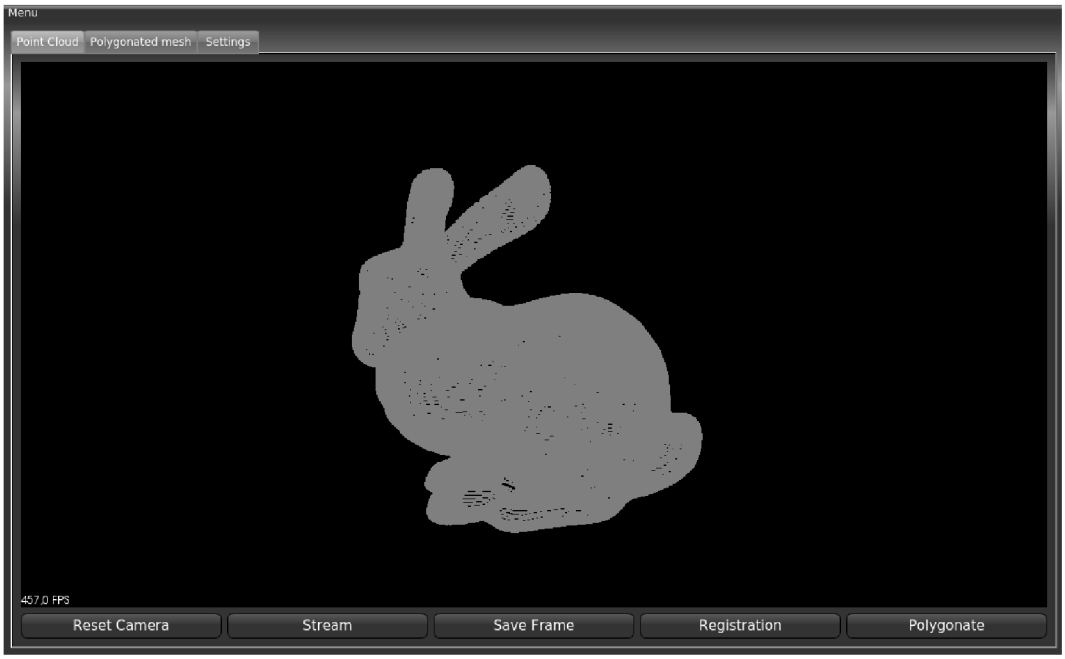
Knihovny označeny symbolem „*“ jsou povinné pro sestavení knihovny PCL. Knihovna VTK[1] je volitelná avšak bez ní se nesestaví modul pro zobrazení dat. Knihovna OpenNI slouží v kombinaci s ovladači senzoru Kinect k získání vstupních dat. Pro spojování textur (sekce 4.5) je použita knihovna OpenCV.

Grafické prostředí aplikace (4.6) bylo implementováno ve frameworku Qt. Jeho výsledná podoba je zobrazena na obrázcích 5.1 a 5.2. Aktuální dění a debugovací informace jsou vypisovány v příkazovém řádku. Většina ovládání aplikace probíhá pomocí myši. Promítací

[1] Visualization ToolKit

plátno je instancí objektu knihovny PCL a obsahuje několik akčních kláves. Jejich výčet je vypsán po stisknutí klávesy „h“ (help). Já jsem chování rozšířil o stisknutí mezerníku, kterým je uložen aktuální snímek ze senzoru.

Vzhled grafického prostředí je vytvořen pomocí aplikace Qt Creator a je popsán jazykem CSS a xml.



Obrázek 5.1: Hlavní obrazovka aplikace s načteným mračnem bodů.

Pro implementaci jsem použil verzovacího systému git. Kvůli rozsahu používaných knihoven jsem zvolil takový postup implementace, že jsem daný algoritmus (například zarovnávání snímků) implementoval ve zvláštním projektu a po úspěšném odladění jsem danou část zakomponoval do celku aplikace. To vede k tomu, že je aplikace přehledně rozdělena do několika souborů se zdrojovým kódem. V repozitáři jsem pak udržoval jen zdrojové kódy hlavní aplikace.

Rozdělení do několika zdrojových souborů přináší otázku, jak bude probíhat překlad do výsledné podoby aplikace. Vývojové prostředí Qt obsahuje nástroj `qmake`. Já jsem se rozhodl pro použití nástroje `cmake` a to kvůli zkoušenostem s ním. Výhodou je flexibilní generování návodu pro překlad (Makefile) na různých operačních systémech a automatická detekce všech použitých knihoven.

Prvním krokem implementace bylo získání a zobrazení dat ze senzoru Kinect. Implementaci jsem zvolil jednoduchou a to takovou, že data z Kinectu jsou kopírována do předem připravené vyhovující struktury mračna bodů. Při spuštění aplikace a úspěšné detekci připojeného senzoru je spuštěn časovač, na nějž je připojen slot. Tento slot slouží ke zkopírování dat z Kinectu do dočasné struktury a následnému překopírování z mračna, se kterým se pracuje dále. Knihovna PCL sice obsahuje metodu k překopírování, ale barevná složka trpí rozkmitem, proto jsem toto implementoval jinak. Volitelně jsou pak vypočteny SIFT klíčové body a vstupní data jsou překreslena. Jelikož překopírování z dočasné struktury není atomická operace, je nutno tuto sekci označit za kritickou a uzavřít ji výlučným přístupem.

Obrázek 5.2: Obrazovka s nastavením parametrů.

Díky dobrému návrhu knihovny PCL pak volání jejích funkcí probíhá obdobně pro všechny algoritmy. Filozofie knihovny v kombinaci mnou navrhnutého načítání parametrů je zobrazena na algoritmu 1.

Algoritmus 1 Popis volání knihovních funkcí

```
1: ziskej instanci nastaveni
2: vytvor mracno<typBodu> vystup
3: vytvor objekt pcl::algoritmus<typBodu> alg
4: alg.nastavVstup(vstup)
5: alg.nastavParametr1(nastaveni.hodnota1)
   ...
6: alg.nastavParametrN(nastaveni.hodnotaN)
7: alg.spustVypocet(vystup)
```

Jak bylo zmíněno, rozsah použitých knihoven postupem rostl a tudíž i čas sestavení aplikace přestával být zanedbatelným. Proto jsem se rozhodl pro načítání všech parametrů použitých metod ze souboru, později i z grafického prostředí. To se ukázalo jako velice efektivní, jelikož jsem nemusel aplikaci překládat po každé změně tzv. „hardcoded“ hodnoty (napevno daná hodnota přímo ve zdrojovém kódu). Parsování hodnot z `json` souboru je implementováno pomocí knihovny boost.

Při nepřítomnosti konfiguračního souboru jsou načteny defaultní hodnoty z hlavičkového souboru `parameters.h`. Z této třídy je vytvořena instance pomocí návrhového vzoru jedináček. To zaručuje, že je v celé aplikaci pouze jedna instance, a nemůže tak dojít k tomu, že se používají špatné hodnoty nastavení.

Další zajímavou konstrukcí, která stojí za popis, je spouštění načítací obrazovky. Při výpočetně náročné operaci je vytvořeno nové vlákno, ve kterém je tato operace zpracová-

vána. Hlavní vlákno slouží pouze pro obnovování grafického prostředí. Načítací obrazovka je implementována pomocí vytvoření nového okna aplikace, ve kterém je dokola přehráván gif soubor s načítací krychlí. Po dokončení časově náročné operace je toto okno zavřeno, což značí konec výpočtu. Navrhl jsem i zrušení výpočtu kliknutím na tuto načítací obrazovku.

Pro lepší přehled aktuálního postupu zarovnávání snímků jsem implementoval postupné zobrazování v grafickém prostředí. To je implementováno tak, že po dokončení iterace je vyslán signál, kterému naslouchá slot hlavní aplikace a po přijmutí je překreslena zobrazovací obrazovka. Překresluje se pouze hrubě podvzorkované mračno pro představu, jak snímky konvergují. V případě, že by se překreslovalo originální mračno, by výpočet transformací trval dlouho a zbytečně by se tak zvyšoval celkový čas zarovnávání snímků.

Další optimalizací pro zrychlení výpočtů je pozastavení vstupního „streamu“ dat při výpočetně náročných operacích. Získávání dat totiž běží ve vlastním vlákně a je zbytečné, pokud je zobrazováno něco jiného. Stream vstupních dat je vypínán i při kliknutí na jinou záložku aplikace (rekonstruovaný povrch, nastavení).

Standardně aplikace počítá na vstupu s mračnem, které obsahuje barevnou informaci. Rozhodl jsem se však i pro implementaci načítání bezbarvých mračen ze souboru, jelikož množství ukázkových dat dostupných na internetu tuto informaci postrádá. To je implementováno tak, že je alokována struktura barevného mračna, do které jsou nakopírovány souřadnice bodů bezbarvého vstupního mračna. Všem bodům je pak přiřazena šedá barva. Je také nutno pamatovat na alfa kanál. Standardně je jeho hodnota nulová, tudíž by body byly průhledné a nešly by zobrazit.

Co se týká vytvoření textury pro výsledný rekonstruovaný model, zvolil jsem jednoduchou metodu. Jelikož se předpokládá, že se jedná o skeny z jedné pozice snímače, lze vytvořit z barevných složek mračen takzvané panorama. Je však nutné se zbavit zkreslení. Toho jsem dosáhl použitím objektu typu `Stitcher` z knihovny OpenCV. V konstruktoru tohoto objektu je předán parametr (`cv::Stitcher::SCANS`), který udává, že je požadováno spojení obrázků bez zkreslení.

5.2 Problémy při řešení

Během implementace jsem narazil na několik problémů, které nepřívětivě zapůsobily na časovou náročnost realizace celé aplikace. Hned z počátku implementace bylo výzvou zprovoznění ovladačů pro senzor Kinect pro operační systém Linux. Naštěstí je na internetu k nalezení spousta návodů jak získávat data ze senzoru. Úspěšný jsem však byl pouze na systému Linux Mint 18 (neúspěch na Fedora 25).

Další problém, který k vyřešení spotřeboval podstatnou část celkového času potřebnému k dokončení aplikace, bylo použití algoritmu pro výpočet FPFH vlastností bodů mračna bodů. Použil jsem zdrojové kódy knihovny PCL z nejnovější revize a sestavil ji s přepínačem pro debugování. To byla však chyba, protože knihovna se musí pro správnou funkčnost sestavit jako `release`. Po překonání těchto překážek už nastávaly jen menší problémy.

Mezi tyto problémy patřila například chyba v implementaci algoritmu Marching Cubes knihovny PCL. Tuto metodu jsem chtěl použít jako zástupce metod rostoucí oblasti. Avšak v současné době je v implementaci chybné alokování paměti, které způsobí neoprávněný přístup do paměti. Dalším problémem v knihovních funkcích je zobrazování polygonálního modelu, kdy se některé polygony chovají jakoby měly obrácenou normálu a tedy nejsou vidět, když by měly.

Nepodařilo se mi implementovat ukončení časově náročné operace, protože knihovna boost z bezpečnostních důvodů neobsahuje prostředek k tvrdému ukončení běžícího vlákna.

Kapitola 6

Experimenty

S hotovou implementací aplikace jsem provedl sadu experimentů. Obecný postup pro provádění experimentu je nasnímání místnosti, nastavení vhodných parametrů a vizuální kontrola dosaženého výsledku. V této kapitole jsou podrobně popsány dva experimenty.

Během těchto experimentů jsem zjistil, že při použití algoritmu MLS je lepší jej používat několikanásobně, vždy s různým poloměrem vyhledávání nejbližších sousedů. S malým poloměrem je odstraněno dostatek redundantních bodů, ale nemusí dojít k odstranění artefaktů popsaných na obrázku 4.10. Po prvním průchodu může být dle potřeby nastaven větší poloměr vyhledávání a mohou tak být odstraněny zmíněné artefakty a to v menším čase, než by bylo za použití jednoho průchodu s velkým poloměrem vyhledávání sousedních bodů. Níže je zobrazena tabulka 6.1 s výsledky.

V obou případech vznikla mřížka reprezentující povrch a obrázek vhodný pro tvorbu textury. Avšak mapování textury v externím programu (například Blender) je nutno provádět ručně, protože se mi již nezdařilo implementovat mapování UV souřadnic.

Veškeré experimenty byly prováděny na počítači Lenovo Thinkpad T440s (procesor Intel®Core^{TM}i7-4600U, 12GB RAM).

	Pokoj 1		Pokoj 2	
Počet snímků	6		8	
Počet bodů	219614		109204	
Čas zarovnání [s]	347		238	
Čas vyhlazení [s]	432	180 + 86	192	80 + 40
Poloměr pro vyhlazení [cm]	15	10 poté 15	15	10 poté 15
Čas rekonstrukce [s]	3,1		2,2	

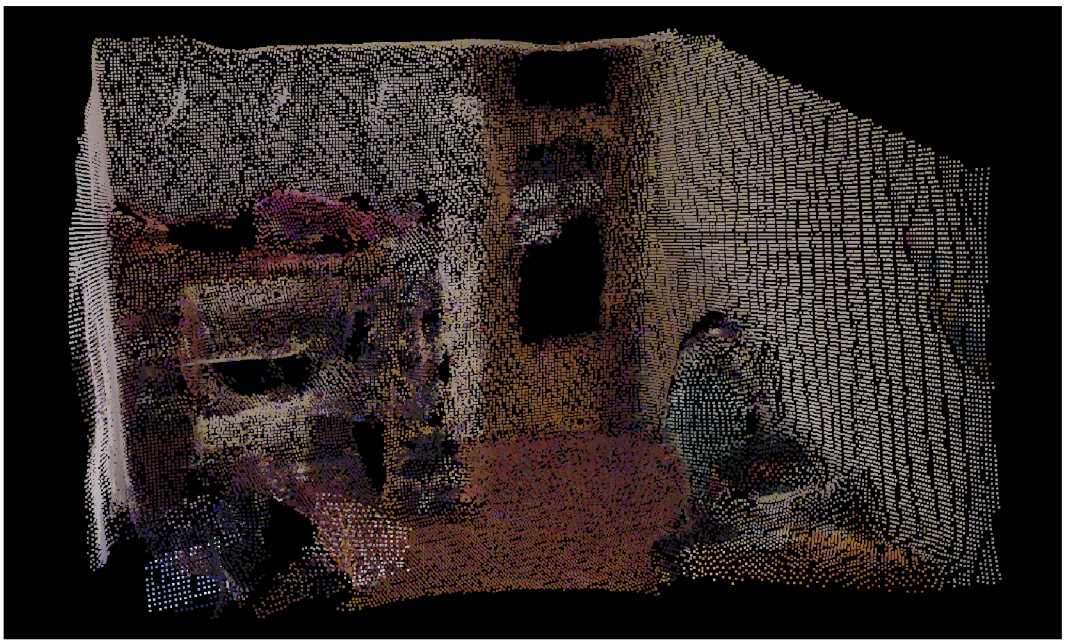
Tabulka 6.1: Výsledky experimentů

6.1 Pokoj 1

Prvním experimentem je nasnímání pokoje. Pokoj obsahuje mnoho barevně rozličných oblastí, což vede k detekci dostatečného množství klíčových bodů. Místnost také obsahuje dostatek hloubkových změn, které jsou důležité pro správné zarovnání algoritmem ICP. Skenování probíhalo z jednoho konce pokoje, což zapříčinilo větší vzdálenost snímaných ploch. Na obrázku 6.1 vidíme spojená mračna a na obrázku 6.2 vidíme výsledek rekonstrukce povrchu. Zarovnání konvergovalo dobře i když trvalo déle. Díky větší vzdálenosti

povrchu od senzoru vzniká větší šum a větší vzdálenosti mezi body, což se projevuje na kvalitě rekonstruovaného povrchu (na obrázku je pouze interpolovaná barva mezi vrcholy).

I přes nízkou hustotu bodů vzniká dobrý model avšak ne tak kvalitní jako když je povrch nasnímán z menší vzdálenosti. Zobrazení drobných detailů by měla řešit textura aplikovaná na jednoduchý povrch.



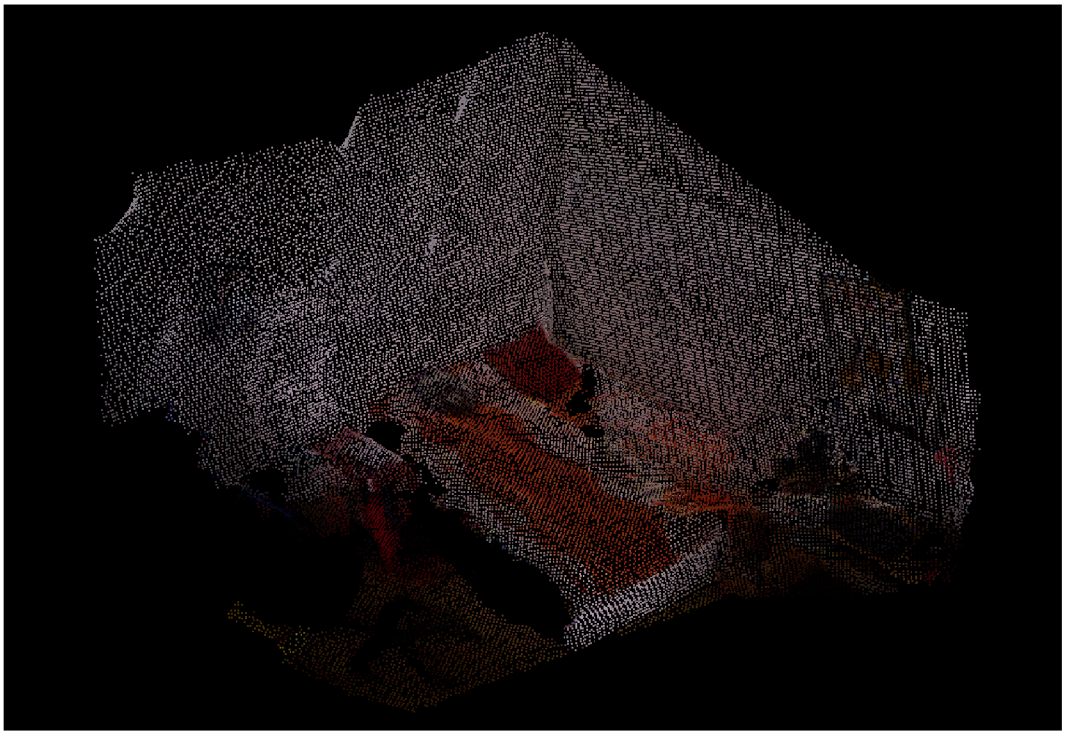
Obrázek 6.1: Zobrazení mračen po zarovnání.

Obrázek 6.2: Rekonstruované mračno zobrazené na obrázku 6.1.

6.2 Pokoj 2

Druhý experiment byl prováděn v jiném pokoji. Oproti prvnímu experimentu jsem nasnímal objekt ze středu pokoje, tedy blíže snímané ploše. Výsledky jsou zobrazeny na obrázcích 6.3 a 6.4. Tento postup je lepší zejména kvůli šumu, který je vyhlazován. Vzdálenější malé detaily jsou potom ztraceny použitím bilaterálního filtru.

Z naměřených časů je zřejmé, že tento postup při skenování je lepší z hlediska časové náročnosti. Zarovnávání snímků proběhlo rychleji, přestože se zarovnávalo více snímků. Výsledky dosáhly lepší kvality než v prvním experimentu právě díky menší vzdálenosti senzoru od snímané plochy.

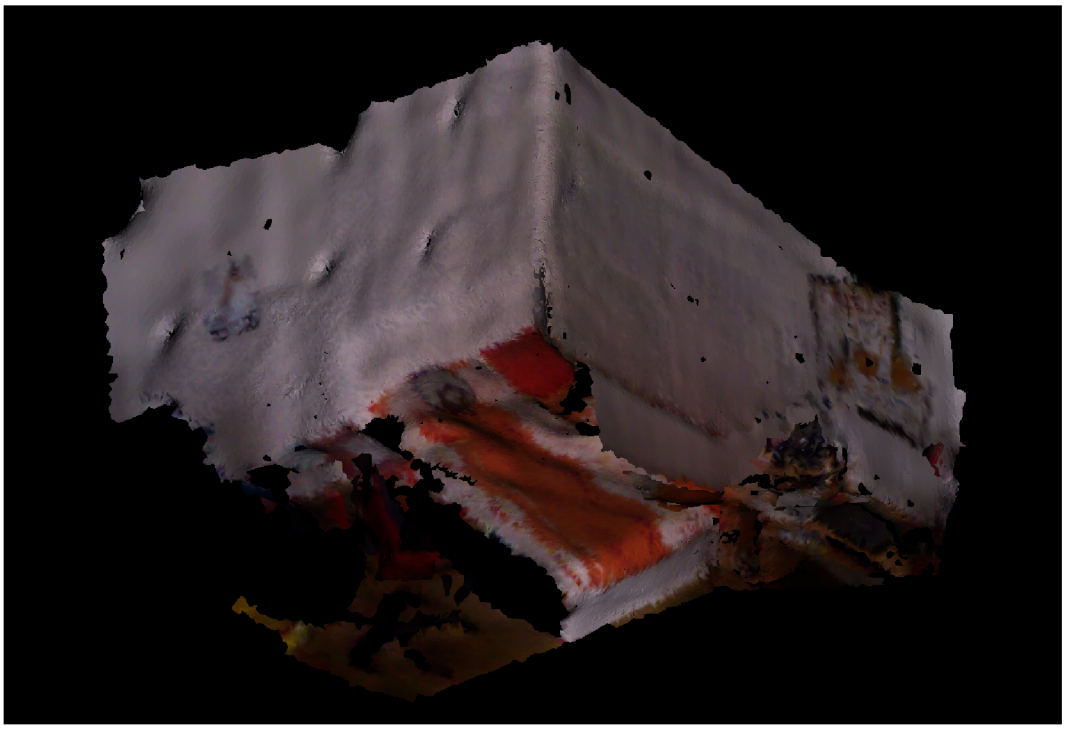


Obrázek 6.3: Zobrazení mračen po zarovnání.

6.3 Rekonstrukce referenčních dat

Předchozí experimenty měly na vstupu data z Kinectu. Pro demonstrační účely jsem se rozhodl provést i experiment na již zarovnaných datech, která jsou nasnímána přesným senzorem nebo jsou tato data uměle vytvořena. Jako vstup jsem použil mračno bodů reprezentující standfordského králíčka. Výsledky můžeme vidět na obrázku 6.5. Z výsledků lze soudit, že metoda Greedy Projection Triangulation je vhodná pro neuzavřené povrchy. Pokud je nasnímána místnost pouze částečně (první snímek nenavazuje na poslední), je tato metoda nejvhodnějším kandidátem pro použití.

Metoda Grid Projection je vhodná pro uzavřené i neuzavřené povrchy. Je však nutné předpokládat velice dlouhou výpočetní dobu. Pokud není aplikace omezena výpočetní dobou, tato metoda se jeví jako vhodná k použití.



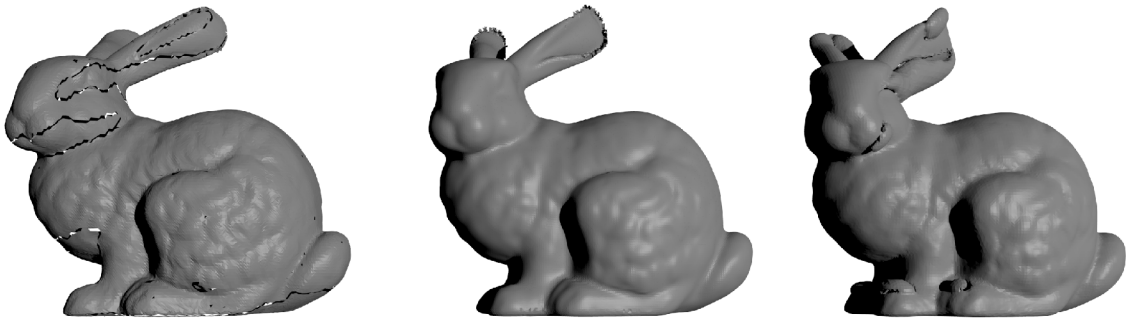
Obrázek 6.4: Rekonstruované mračno zobrazené na obrázku 6.3.

Poslední použitá metoda je Poissonova. Tato metoda je určena výhradně pro uzavřené povrchy, jinak vznikají artefakty. Časová náročnost metod pro tento model je znázorněna v tabulce 6.2.

Automatizované porovnávání chyb rekonstrukce povrchu lze provádět například detekcí děr nebo testem na uzavřenost (vodotěsnost) rekonstruovaného povrchu. Vhodným způsobem jak testovat kvalitu oproti referenčním datům je výpočet Hausdorffovy vzdálenosti mezi dvěma povrchy [6]. K tomu může být použit například program Metro [5].

Metoda	Čas rekonstrukce [s]	Počet vytvořených polygonů
Greedy Projection	2	69288
Grid Projection	74	81848
Poissonova	5,5	194198

Tabulka 6.2: Výsledky použitých rekonstrukčních metod.



Obrázek 6.5: Vlevo mračno rekonstruované pomocí metody Greedy Triangulation Projection. Můžeme si všimnout, že z použitých metod má nejjemnější detaily, ale obsahuje artefakty nespojených oblastí. Prostřední králíček je rekonstruován metodou Grid Projection. Model je velice kvalitní a obsahuje malé množství artefaktů (kolem uší), ale metoda je velice časově náročná. Poslední použitou metodou je Poissonova rekonstrukce (králíček vpravo). Metoda zvládá jemné detaily v závislosti nastavení hloubky oktalového stromu. Je velmi rychlá, ale vznikají artefakty.

6.4 Zhodnocení experimentů

Zarovnání snímků získaných ze senzoru Kinect je silně závislé na nastavení parametrů metod. Pokud jsou tyto parametry nastaveny správně, je proveden úspěšně hrubý odhad korespondence. Tento odhad je doladěn algoritmem ICP. Zpracování dvojice snímků na uvedené sestavě probíhá v řádu minut. Při zarovnávání většího počtu snímků toto může být zdlouhavé a bylo by vhodnější k výpočtům využít grafickou kartu.

Provedení rekonstrukce povrchu je rychlé, ale kvalita detailů je ovlivněna vzdáleností senzoru od snímaného povrchu. Rekonstruovaná vhodně nasnímaná vstupní data jsou pak vhodná k vizualizaci. Po provedení decimace modelu nastává rapidní úbytek trojúhelníků bez větší ztráty detailů. To v kombinaci s aplikováním textury, vytvořené z výstupního obrázku, dává prostor k využití v aplikacích virtuální reality či her.

Kapitola 7

Závěr

Hlavním přínosem této práce oproti existujícím řešením je myšlenka nepoužití samotného algoritmu ICP pro zarovnávání dat. Namísto toho jsem navrhl výpočet odhadu posunu mezi dvěma snímky a to za pomocí FPFH histogramů bodů mračna. Algoritmus ICP je použit jen pro jemnější zarovnání.

Dalším přínosem je vyhlazení spojených mračen, které mohou obsahovat artefakty. Takto upravené mračno lze rekonstruovat jednoduchou metodou. Mřížku rekonstruovaného modelu lze opravit a zjednodušit.

Veškeré operace jsou spouštěny v grafickém prostředí implementované aplikace a uživateli se výsledek zobrazí jakmile je dostupný.

Přínosem mohlo být i generování textury s vysokým rozlišením pro takto vytvořený model. Bohužel se mi nezdařilo implementovat výpočet UV souřadnic tak, aby po načtení modelu a aplikování textury v externím programu byl připraven hotový výsledek. V aktuální fázi je stav takový, že se texturování musí provést ručně z vytvořeného obrázku.

To dává prostor pro další vývoj. Spolu s tímto bych jako příští postup zvažoval adaptivní nastavování parametrů metod pro lepší práci se zarovnáváním snímků popřípadě nabídnout uživateli z několika předvoleb (místnost s nábytkem, velikost místnosti, ...). Pro rychlejší zpracování dat by také bylo dobré použít výpočtů na grafické kartě pro náročné operace. Zajímavou myšlenkou je detekce materiálů z nasnímané textury. Po segmentaci dat a přiřazení fotorealistických materiálů by mohl být výstup aplikace kvalitním materiálem pro další práci.

Literatura

[1] *Estimating Surface Normals in a PointCloud*. Point Cloud Library Documentation, [Online; navštíveno 18.03.2017].
URL http://pointclouds.org/documentation/tutorials/normal_estimation.php#normal-estimation

[2] *pcl::PointXYZRGB Struct Reference*. Point Cloud Library Documentation, [Online; navštíveno 18.03.2017].
URL
http://docs.pointclouds.org/1.7.0/structpcl_1_1_point_x_y_z_r_g_b.html

[3] Alexa, M.; Behr, J.; Cohen-Or, D.; aj.: *Computing and Rendering Point Set Surfaces*. Technická zpráva, [Online; navštíveno 01.04.2017].
URL http://www.sci.utah.edu/~shachar/Publications/crpss.pdf

[4] Cagaš, P.: Extrakce lokálních obrazových deskriptorů na GPU.

[5] Cignoni, P.; Rocchini, C.; Scopigno, R.: Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum*, ročník 17, č. 2, 1998: s. 167–174, ISSN 1467-8659, doi:10.1111/1467-8659.00236.
URL http://vcg.isti.cnr.it/publications/papers/metro.pdf

[6] Guthe, M.; Borodin, P.; Klein, R.: *Fast and accurate Hausdorff distance calculation between meshes*. Technická zpráva, University of Bonn, Institute of Computer Science II, [Online; navštíveno 15.05.2017].
URL http://www.mathematik.uni-marburg.de/~guthe/Publications/guthe-2005-fast.pdf

[7] Jamin, C.; Pion, S.; Teillaud, M.: *CGAL 4.9.1 - 3D Triangulations*. Technická zpráva, [Online; navštíveno 14.04.2017].
URL https://doc.cgal.org/latest/Triangulation_3/

[8] Kazhdan, M.; Bolitho1, M.; Hoppe, H.: *Poisson Surface Reconstruction*. Technická zpráva, [Online; navštíveno 22.04.2017].
URL http://hhoppe.com/poissonrecon.pdf

[9] Knot, S.: *Ovládání počítače pomocí senzoru Kinect*. [Online; navštíveno 18.03.2017].
URL https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=119182

[10] Levin, D.: *Mesh-Independent Surface Interpolation*. Technická zpráva, Tel Aviv University, [Online; navštíveno 01.04.2017].
URL http://www.math.tau.ac.il/~levin/mls03cor.pdf

[11] Li, R.; Liu, L.; Phan, L.; aj.: *Polygonizing Extremal Surfaces with Manifold Guarantees*. Technická zpráva, Washington University, [Online; navštíveno 07.04.2017].
URL http://www.cs.wustl.edu/~taoju/research/paper_es_short.pdf

[12] Lowe, D. G.: *Distinctive Image Features from Scale-Invariant Keypoints*. Technická zpráva, [Online; navštíveno 16.04.2017].
URL https://www.cse.unr.edu/~bebis/CS491Y/Papers/Lowe04.pdf

[13] Marani, R.; Reno, V.; Nitti, M.; aj.: *A Modified Iterative Closest Point Algorithm for 3D Point Cloud Registration*. Technická zpráva, Institute of Intelligent Systems for Automation, Italian National Research Council, 2016, [Online; navštíveno 01.04.2017].
URL http://onlinelibrary.wiley.com/doi/10.1111/mice.12184/pdf

[14] Martin, K.; Schroeder, W.; Lorensen, B.: *vtkFillHolesFilter Class Reference*. Technická zpráva, [Online; navštíveno 15.05.2017].
URL http://www.vtk.org/doc/nightly/html/classvtkFillHolesFilter.html

[15] Martin, K.; Schroeder, W.; Lorensen, B.; aj.: *vtkQuadricDecimation Class Reference*. Technická zpráva, [Online; navštíveno 15.05.2017].
URL http://www.vtk.org/doc/nightly/html/classvtkQuadricDecimation.html

[16] Marton, Z. C.; Rusu, R. B.; Beetz, M.: On Fast Surface Reconstruction Methods for Large and Noisy Datasets. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Kobe, Japan, Květen 12-17 2009.

[17] Maur, P.: *Delaunay Triangulation in 3D*. Technická zpráva, University of West Bohemia in Pilsen, 2002, [Online; navštíveno 14.04.2017].
URL https://www.kiv.zcu.cz/site/documents/verejne/vyzkum/publikace/technicke-zpravy/2002/tr-2002-02.pdf

[18] Paris, S.; Durand, F.: *A Fast Approximation of the Bilateral Filterusing a Signal Processing Approach*. Technická zpráva, Massachusetts Institute of Technology Computer Science and Artificial Intelligence Laboratory, [Online; navštíveno 30.03.2017].
URL http://people.csail.mit.edu/sparis/publi/2006/eccv/Paris_06_Fast_Approximation.pdf

[19] Rusu, R. B.: *Getting Started / Basic Structures*. Point Cloud Library Documentation, [Online; navštíveno 18.03.2017].
URL http://pointclouds.org/documentation/tutorials/basic_structures.php#basic-structures

[20] Rusu, R. B.: Semantic 3D Object Maps for Everyday Manipulationin Human Living Environments.

[21] Rusu, R. B.: *The PCD (Point Cloud Data) file format*. [Online; navštíveno 18.03.2017].
URL http://pointclouds.org/documentation/tutorials/pcd_file_format.php

[22] Rusu, R. B.; Blodow, N.; Beetz, M.: *Fast Point Feature Histograms (FPFH) for 3D Registration*. Technická zpráva, Intelligent Autonomous Systems, Technische

Universität München, [Online; navštíveno 18.03.2017].
URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.332.3710&rep=rep1&type=pdf

[23] Rusu, R. B.; Cousins, S.: 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[24] Rusu, R. B.; Steder, B.: *pcl::VoxelGrid< PointT > Class Template Reference*. Technická zpráva, [Online; navštíveno 10.05.2017].
URL
http://docs.pointclouds.org/trunk/classpcl_1_1_voxel_grid.html#details

[25] Sipiran, I.; Bustos, B.: Harris 3D: A Robust Extension of the Harris Operator for Interest Point Detection on 3D Meshes. *Vis. Comput.*, ročník 27, č. 11, Listopad 2011: s. 963–976, ISSN 0178-2789, doi:10.1007/s00371-011-0610-y.
URL http://dx.doi.org/10.1007/s00371-011-0610-y

[26] Steder, B.; Rusu, R. B.; Konolige, K.; aj.: *NARF: 3D Range Image Features for Object Recognition*. Technická zpráva, [Online; navštíveno 01.04.2017].
URL http://ais.informatik.uni-freiburg.de/publications/papers/steder10irosws.pdf

[27] Zhong, Y.: *Intrinsic shape signatures: A shape descriptor for 3D object recognition*. Technická zpráva, Carnegie Mellon University, [Online; navštíveno 01.04.2017].
URL https://goo.gl/ddZtw1

Přílohy

Příloha A

Sestavení a spuštění aplikace

Pro překlad aplikace je nutné mít nainstalované uvedené knihovny:

- boost 1.6
- Eigen 3.0
- FLANN 1.7.1
- VTK 6.3 nebo VTK 7.1
- OpenNI 1.3
- PCL 1.8
- OpenCV 3.2
- Qt 5.7

Navíc je nutné mít nainstalován program `cmake` v minimální verzi 2.6.

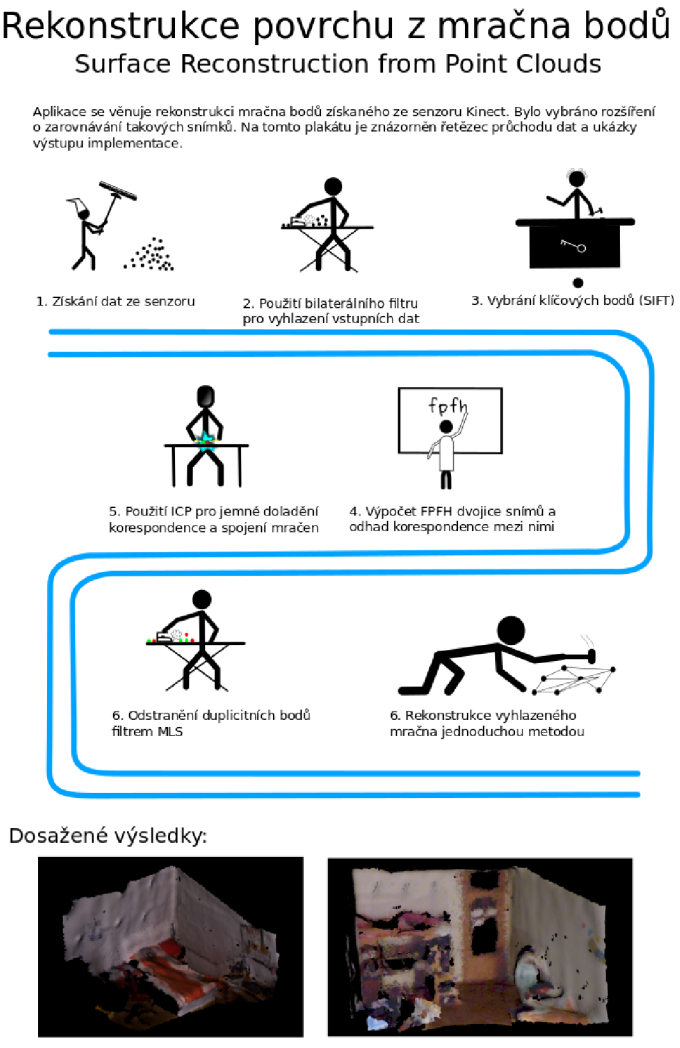
První možností, jak program sestavit, je použití dvou příkazů ve složce `build`. Prvním příkazem je `cmake ../src`. Tento příkaz vyhledá všechny požadované knihovny a vytvoří soubor Makefile. Poté stačí spustit příkaz `make` a po úspěšném sestavení se vytvoří spustitelný soubor `RoomScanner`.

Druhou možností sestavení je otevření projektu (soubor RoomScanner.pro) v aplikaci Qt Creator a po správném nastavení sestavovacích kroků (opět `cmake` a `make`) lze aplikaci sestavit a spustit přímo z tohoto vývojového prostředí.

Spuštění aplikace z příkazové řádky probíhá bez parametrů a to příkazem `./RoomScanner` ve složce `build`.

Příloha B

Plakát



Obrázek B.1: Plakát.