VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE GEOMETRIE V MRAČNECH BODŮ

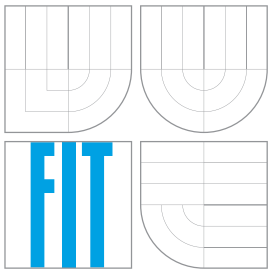
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PAVOL ELDES

BRNO 2014

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE GEOMETRIE V MRAČNECH BODŮ

BASIC SHAPES DETECTION IN POINT CLOUDS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

PAVOL ELDES

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MICHAL ŠPANĚL, Ph.D.

BRNO 2014

Abstrakt

Tato práce se zabývá detekcí jednoduché geometrie v mračnech bodů se speciálním důrazem na vlivy normál bodů na kvalitu a rychlost produkovaných výsledků. Hlavním produktem je pak aplikace demonstrující detekce ploch, koulí a válců. Na detekci je využit RANSAC algoritmus, který je v téhle práci modifikován na práci s více modelama současně. Další modifikací je úprava detekčních modelů pro přísnější výběr kandidátních útvarů.

Abstract

This thesis deals with the topic of simple geometric shapes detection from point-clouds with a special interest in the influence of point normals on speed and quality of produced results. Its main product is an application that demonstrates detection of planes, spheres and cylinders. Detection is done using the RANSAC paradigm that is modified in this thesis to allow it to work with multiple models simultaneously. Another modification presented in this thesis focuses on enforcing stricter candidate shapes selection conditions for detection models.

Klíčová slova

mračna bodů, normály bodů, detekce útvarů, jednoduché útvary, RANSAC, PCL

Keywords

point clouds, point normals, shape detection, simple shapes, RANSAC, PCL

Citace

Pavol Eldes: Detekce geometrie v mračnech bodů, bakalářská práce, Brno, FIT VUT v Brně, 2014

Detekce geometrie v mračnech bodů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Michala Španěla, Ph.D.. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Pavol Eldes
20. května 2014

Poděkování

Rád by som sa poďakoval vedúcemu práce Ing. Michalovi Španělovi, Ph.D. za jeho vedenie počas tvorby tohto diela a mojim priateľom a blízkym za morálnu podporu v časoch nezdaru.

Obsah

1 Úvod **4**

2 Teória detekcie útvarov z mračien bodov **5**
2.1 Mračná bodov 5
2.2 Metódy na detekciu útvarov v point-cloudoch 6
2.3 RANSAC 8

3 Návrh detektora jednoduchej geometrie **10**
3.1 Základná štruktúra 10
3.2 Výpočet normál 11
3.3 Modely detekovaných útvarov 12
3.4 RANSAC algoritmus 14
3.5 Vstupy a výstupy 16

4 Implementácia **18**
4.1 Použité nástroje 18
4.2 Základná štruktúra 19
4.3 Výpočet normál 20
4.4 Modely detekovaných útvarov 21
4.5 RANSAC algoritmus 23
4.6 Vstupy a výstupy 24

5 Experimenty **27**
5.1 Popis testovacích dát a systému 27
5.2 Normály 29
5.3 Detekcia 32

6 Záver **37**

Seznam obrázků

2.1 Kinect pre Xbox One. Prevzaté z [7] 5
2.2 Príklad detekcie útvarov scény. Prevzaté z [27] 6
2.3 Spôsob práce RANSAC algoritmu. Prevzaté z [6] 8

5.1 Scéna 1 28
5.2 Scéna 2 28
5.3 Scéna 3 29
5.4 Scéna 4 29
5.5 Scéna 5 30
5.6 Kontaktný bod gule a plochy – Scéna 1 - Krok 1,0 - Šum 0,50 31
5.7 Miesto kontaktu plochy, valca a gule – Scéna 1 - Krok 1,0 - Šum 0,00 32
5.8 Výsledok detekcie – Scéna 5 - O/M/M - 1,0/0,0 - Doba detekcie: 11,13 s . . 35
5.9 Výsledok detekcie – Scéna 5 - O/O/O - 1,0/0,0 - Doba detekcie: 74,01 s . . . 35
5.10 Výsledok detekcie – Scéna 5 - O/M/M - 0,5/0,5 - Doba detekcie: 207,29 s . . 36
5.11 Výsledok detekcie – Scéna 5 - O/O/O - 0,5/0,5 - Doba detekcie: 622,29 s . . 36

Seznam tabulek

5.1 Vlastnosti scén 27
5.2 Úspešnosť výpočtu hodnôt a orientácie normál pre dané parametre 31
5.3 Výsledky rýchlosti a kvality detekcie v daných situáciách 33
5.4 Pomer unikátnych detekovaných útvarov ku korektne detekovaným útvarom 34

Kapitola 1

Úvod

Zrak je najdôležitejším z ľudských zmyslov, pretože pomocou neho získavame najviac informácií. Ruka v ruke s ním idú schopnosti rozlišovať, členiť a inak pracovať s vizuálnymi informáciami, ktoré sú pre nás úplnou samozrejmosťou. Je teda prirodzené, že sa snažíme túto schopnosť preniesť aj do sféry výpočtovej techniky, kde je táto schopnosť rovnako želanou, no nie natoľko samozrejmou. Toto je práve doména odvetvia výpočtovej techniky nazývajúcej sa počítačové videnie.

Jednou z oblastí, ktorou sa zaoberá odvetvie počítačoveho videnia je detekcia určitých útvarov a príznakov z obrazových dát, ktoré môžu byť v rôznych formách – napríklad aj vo forme mračien bodov. Toto je vo všeobecnosti často žiadaná funkcia, pretože takáto informácia má široké využitie v rôznych aplikačných oblastiach, od automatického spracovania dát po ovládanie zariadení.

Táto práca sa zaoberá konkrétnou oblasťou z tohto komplexného celku, detekciou jednoduchej geometrie v mračnách bodov. Cieľom práce je navrhnúť, implementovať a otestovať detektor jednoduchej geometrie. Tento bude prezentovať spôsob detekcie viacerých modelov pomocou metódy RANSAC so špeciálnym zameraním na vplyvy normál bodov na kvalitu a rýchlosť výpočtu. Za základ tejto práce poslúžil článok *Efficient RANSAC for Point-Cloud Shape Detection* [11].

Text práce je logicky členený podľa etáp, ktorými sa počas tvorby prechádzalo. V kapitole 2 nájdete teoretický základ potrebný na orientáciu v tejto problematike a pochopenie princípov, na základe ktorých je postavená táto práca. V kapitole 3 sa nachádza návrh základných prvkov aplikácie a v kapitole 4 nájdete jej implementáciu spolu s použitými nástrojmi. Kapitola 5 obsahuje popis používaných testovacích dát, experimenty ktoré boli vykonané s touto aplikáciou na týchto dátach a zhodnotenie ich výsledkov. Posledná kapitola 6 potom obsahuje zhodnotenie výsledkov práce ako celku a návrh možných smerov pokračovania v tejto oblasti.

Kapitola 2

Teória detekcie útvarov z mračien bodov

V tejto kapitole sa nachádza prehľad základných konceptov, na ktorých je postavená táto práca. V prvej sekcii 2.1 je stručný popis point cloudových dát, ich vznik a využitie. V ďalšej sekcii 2.2 nájdete stručný prehľad všeobecných prístupov k detekcii v point cloudoch, ktoré sa využívajú v dnešnej dobe a v sekcii 2.3 je podrobnejšie popísaná metóda RANSAC, ktorá je použitá v tejto práci.

2.1 Mračná bodov

Keďže je táto práca zasadená v oblasti mračien bodov, bolo by vhodné ako prvé aspoň v krátkosti vysvetliť, čo vlastne tieto mračná bodov sú. Mračná bodov alebo point cloudy sú množinou bodov popísaných v n-rozmernom priestore [24]. V 3D priestore sú to zvyčajne body, popisujúce povrch objektov, na ktorých sa nachádzajú. Point cloudy majú oproti tradičným 2D obrazovým dátam niekoľko výhod rovnako ako aj nevýhod. Medzi výhody zaiste patrí jednoznačnosť polohy v priestore, naopak nevýhodou je ich obrovská početnosť aj v jednoduchých scénach. Hoci sú point cloudy používané v množstve situácií, zriedkakedy sú používané priamo vo svojom základnom tvare a bývajú prvotne prevedené napr. na polygoniálne modely.

Obrázek 2.1: Kinect pre Xbox One. Prevzaté z [7]

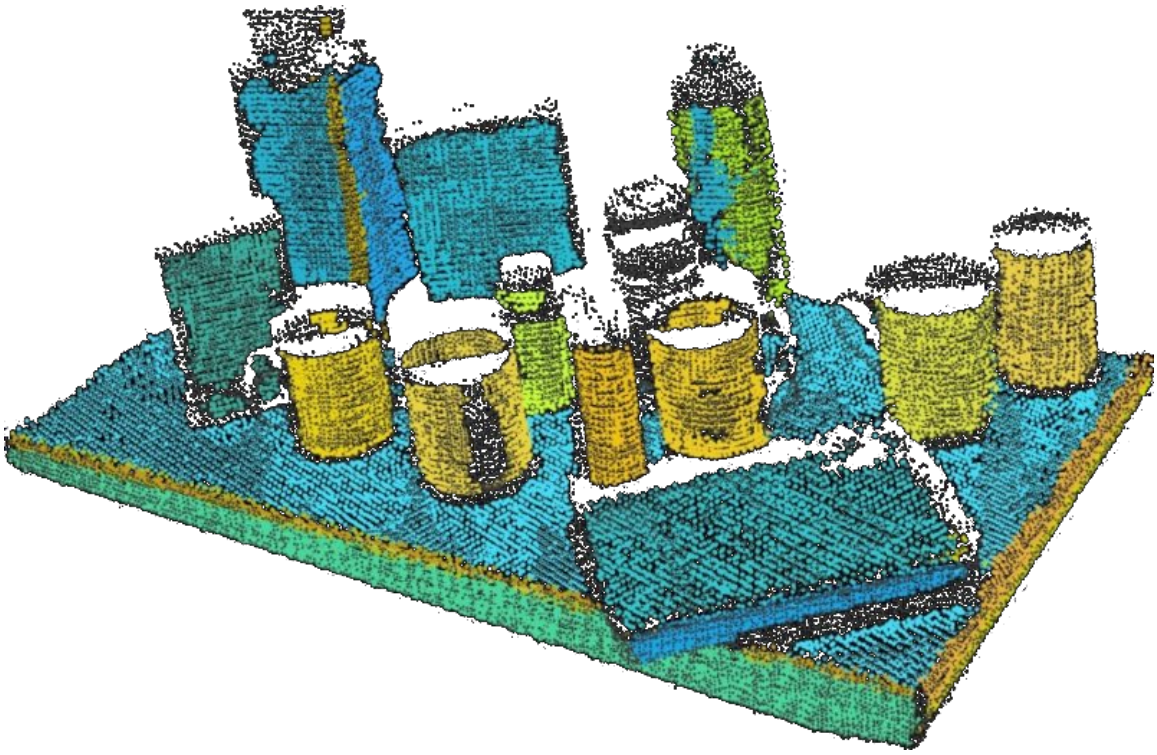
.

Point cloudy sú získavané pomocou zariadení nazývaných 3D skenery. Tieto zariadenia sú pomerne rôznorodé a dajú sa deliť do rôznych kategórií podľa technológie, na základe ktorej sú postavené. Ešte donedávna boli tieto zariadenia príliš nákladnými na bežné použitie a tak sa využívali na prácu v produkčnej sfére. S príchodom lacných zariadení typu Microsoft Kinect, ktorý bol vyvinutý primárne na účely prirodzeného ovládania počítačových hier pre Xbox 360, sa toto odvetvie sprístupnilo širšiemu okoliu a rozšírila sa tak komunita venujúca sa práve tejto oblasti [4].

2.2 Metódy na detekciu útvarov v point-cloudoch

Funkcia detekcie útvarov je pomerne základnou v rôznych odvetviach výpočtovej techniky, no hlavne tých, súvisiacich s počítačovou geometriou. Existuje teda množstvo rozličných detekčných metód, ktoré sú závislé na type úlohy, aplikačnej oblasti a zložitosti hľadaných útvarov. Nakoľko však ešte donedávna práca s point cloudmi nebola až tak rozšírená, väčšina z týchto existujúcich metód pracuje nad 2D obrazovými dátami a oveľa menej je schopných pracovať priamo nad 3D point cloudmi [3].

Na základné delenie týchto metód by sa dala použiť práve zložitosť útvarov, ktoré chceme detekovať.



Obrázek 2.2: Príklad detekcie útvarov scény. Prevzaté z [27]

.

Detekcia objektov

Zložitejšie útvary sa vo väčšine prípadov buď vôbec nedajú popísať analyticky alebo sa popisujú len veľmi ťažko. Takéto metódy zväčša pracujú na základe detekcie objektov, kto-

rých reprezentáciu sa snažia zjednodušiť, ale zachovať pritom všetky podstatné vlastnosti. Na toto zjednodušenie slúžia rôzne deskriptory, ktoré popisujú najdôležitejšie informácie spoločné pre objekty rovnakého typu. Proces detekcie potom väčšinou obsahuje krok strojového učenia, v ktorom sa detektor z testovacej množiny naučí ako vyzerajú jednotlivé vyhľadávané typy útvarov. Príkladom tohto prístupu je práca *Implicit shape models for object detection in 3D point clouds* [14]. Tieto prístupy sú odlišné od prístupu, na základe ktorého je postavená táto práca a aj keď stoja za zmienku, ďalej nebudú spomínané.

Metódy v reverznom inžinierstve

V prípade jednoduchých útvarov, ktoré sú centrom tejto práce je situácia trochu odlišná. V oblasti reverzného inžinierstva je hlavnou úlohou reprezentovať čo možno najväčšiu časť point cloudu jednoduchými útvarmi, ktoré sú potom použité na vytvorenie kompletného modelu celej scény v CAD aplikácii. Tieto prístupy však väčšinou vyžadujú istý typ spojitostnej informácie a nedosahujú dobré výsledky v prítomnosti väčšieho množstva outlierov, t.j. bodov, ktoré nepatria nijakému z hľadaných jednoduchých útvarov. Taktiež sú tu často používané postupy, ktoré zahŕňajú oddelený segmentačný krok alebo nejaký typ algoritmu, ktorý pracuje na základe rozširovania svojho okolia z počiatočného bodu, ak sú splnené podmienky na jeho rast. Približne týmto spôsobom funguje aj algoritmus z práce *Segmentation of point clouds using smoothness constraint* [8]. Až po prvotnej segmentácii point cloudu na menšie časti prichádza krok, v ktorom sa každej časti zvlášť priradzuje jednoduchý objekt, ktorý ju vhodne popisuje. Podobne by sa dal využiť aj algoritmus z článku *Sharp Feature Detection in Point Clouds* [15], ktorý sa sústredí na vyhľadávanie ostrých čŕt. Tie by sa dali využiť na rozdelenie point cloudu na menšie časti tak, že tieto ostré črty by predstavovali hrany jednotlivých menších oblastí. Tieto oblasti by potom ako v predchádzajúcom prípade boli priradené jednoduchým útvarom, ktoré by ich vhodne reprezentovali. Ani jedna z týchto metód však nevyhľadáva priamo jednotlivé geometrické primitíva.

Metódy v počítačovom videní

Oblasť počítačového videnia obsahuje niekoľko metodológií, ktoré sa zaoberajú extrakciou jednoduchých útvarov, pričom medzi najznámejšie patria Houghova transformácia a RANSAC. Obe tieto metódy dokážu úspešne nájsť jednoduché geometrické útvary či už v 2D alebo 3D dátach. Aj keď pracujú odlišnými spôsobmi, ich základné vlastnosti sú rovnaké. Obe dokážu spoľahlivo nájsť útvary aj pri prítomnosti veľkého množstva outlierov a určitého množstva šumu. Ich efektivita a značné hardwarové nároky sú však ich najväčšími slabinami a bez dodatočných optimalizácií nie sú tieto algoritmy veľmi použiteľné. Pre oba algoritmy ale existuje množstvo optimalizačných techník, ktoré ich robia zaujímavými voľbami na riešenie určitých typov problémov [11].

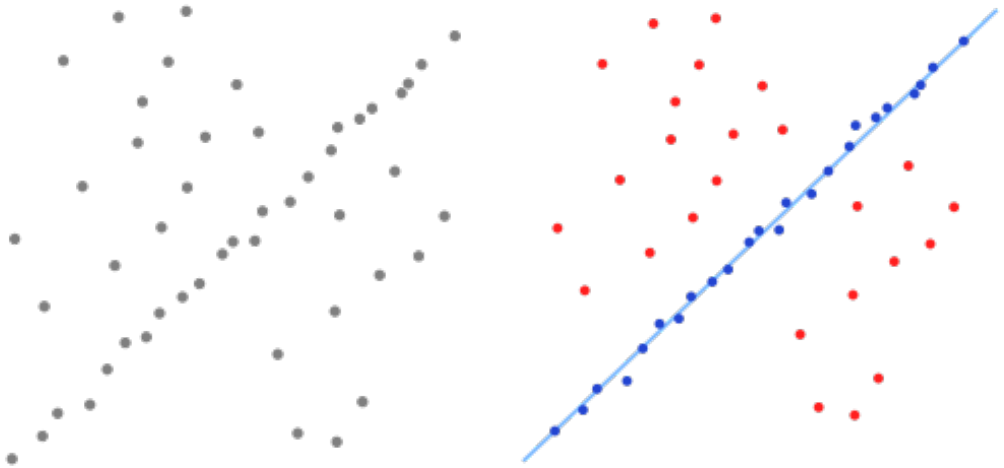
Základná myšlienka Houghovej transformácie je jednoduchá. V prvom kroku sa z dátového prostredia mapuje každý bod pre každý parametrizovaný útvar do parametrického prostredia, kde sú popísané všetky možné varianty tohto útvaru, ktoré obsahujú originálny bod, pomocou hlasov. Útvary sú potom vybrané tak, že sa vyberú tie vektory parametrov, ktoré získali najviac hlasov. Problémom je, že pri požadovanej presnosti veľkosť Houghovho parametrického prostredia rastie exponenciálne s počtom parametrov [17]. Aj keď existuje množstvo metód na zníženie pamäťových nárokov tohto algoritmu, jeho doménou ostáva hlavne sféra 2D, kde je počet parametrov hľadaných útvarov nízky.

Ďalej sa dostávame k metóde RANSAC, ktorá je použitá v tejto práci, a preto bude popísaná detailnejšie v ďalšej podkapitole.

2.3 RANSAC

RANSAC je skratkou pre „RANdom SAmple Consensus“, čo je iteratívna metóda, ktorá sa snaží získať parametre daného modelu z predložených dát. Jej základná verzia bola prezentovaná už v roku 1981 v článku *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography* [1]. Táto metóda je nedeterministická, čo znamená, že produkuje dobre výsledky len s istou pravdepodobnosťou. Keďže je to metóda iteratívna, šanca, že bude nájdený dobrý výsledok, sa zvyšuje s počtom iterácií algoritmu.

Dáta, s ktorými pracuje, sa delia na dve základné skupiny – inliery a outliery. Inliery sú dáta, ktoré patria hľadanému útvaru alebo sa nachádzaju v jeho blízkom okolí. Toto okolie tvorí hranicu tolerancie chýb, ktoré sa v dátach môžu vyskytnúť vplyvom negatívnych efektov ako napr. šum. Outliery sú potom všetky ostatné body, ktoré nemožno reprezentovať pomocou hľadaného modelu. Úlohou RANSACu je potom rozlíšiť pre každý bod, do ktorej skupiny patrí. Tento princíp je vidieť aj na obrázku 2.3



Obrázek 2.3: Spôsob práce RANSAC algoritmu. Prevzaté z [6]

.

Spôsob výpočtu

Pri výpočte sa postupuje tak, že z daných dát sa vyberajú minimálne množiny. Minimálna množina je množinou, ktorá obsahuje presne taký počet bodov, aby sa z nich dali zostrojiť parametre modelu, ktoré ho unikátne identifikujú. Tieto kandidátne útvary sú následne testované voči všetkým bodom v datasete, čím sa zistí, koľko bodov celkovo je dobre aproximovaných týmito kandidátnymi útvarmi. Každý útvar má svoje skóre, ktorým je práve počet bodov, ktoré dobre aproximuje. V priebehu výpočtu sa udržiava informácia o najlepšom kandidátovi a akonáhle pravdepodobnosť, že sa podarí nájsť lepšieho kandidáta klesne pod určitý prah, algoritmus sa ukončí a najlepší kandidát je extrahovaný. Algoritmus potom začína znova na zvyšku dát [28].

Ukončenie algoritmu je jeho podstatnou súčasťou, keďže ovplyvňuje rýchlosť výpočtu a kvalitu výsledku. V praxi je zvyčajne potrebné previesť pravdepodobnostný prah, po ktorom bude algoritmus ukončený, na počet kandidátov, ktorých je potrebné ešte vygenerovať.

Majme point cloud $\mathcal{P}$ veľkosti N a útvar ψ veľkosti n. Nech k je veľkosť minimálnej množiny tohto útvaru. Predpokladajme, že hocakých k bodov útvaru ψ vedie k nájdeniu dobrého kandidátneho útvaru. Pravdepodobnosť, že sa nám podarí zdetekovať útvar v jednom kroku, je potom:

$$P(n) = \binom{n}{k} \Big/ \binom{N}{k} \approx \left(\frac{n}{N}\right)^k \tag{2.1}$$

Pravdepodobnosť úspešnej detekcie po tom ako bolo vygenerovaných s kandidátnych útvarov sa rovná komplementu ich za sebou idúcich neúspechov:

$$P(n, s) = 1 - (1 - P(n))^s \tag{2.2}$$

Riešením rovnice pre s získame počet kandidátov T potrebných na detekovanie útvarov veľkosti n s pravdepodobnosťou $P(n, T) \geq p_t$:

$$T \geq \frac{\ln(1 - p_t)}{\ln(1 - P(n))} \tag{2.3}$$

Pre malé $P(n)$ môže byť logaritmus v menovateli aproximovaný jeho Tayloorovým rozvojom $\ln(1 - P(n)) = -P(n) + O(P(n)^2)$ čo dáva výsledok:

$$T \approx \frac{-\ln(1 - p_t)}{P(n)} \tag{2.4}$$

Týmto sme získali želaný počet kandidátov, ktorých je potrebné vygenerovať na ukončenie výpočtu s istou pravdepodobnosťou [11]. Zo vzorca môžeme jasne vidieť, že počet potrebných kandidátov je závislý od veľkosti kandidátneho útvaru, čo znamená, že prakticky bude nutné reevaluovať túto hodnotu zakaždým, keď bude nájdený lepší kandidátny útvar.

Hlavné vlastnosti

Medzi hlavné výhody RANSACu patrí jeho konceptuálna jednoduchosť, ktorá je prínosom pri jeho modifikácii a implementácii a ďalej potom jeho schopnosť poradiť si s dátami pozostávajúcimi z vyše 50 % outlierov, hoci pri týchto hodnotách už nie je v základnej forme ideálnym. Jeho negatívnou stránkou je vysoká náročnosť na výpočtové prostriedky, ktorá je najvýraznejšia, ak nie sú použité žiadne optimalizácie.

Existujúce verzie

Od vydania RANSACu bolo vytvorených niekoľko jeho modifikácií s cieľom vylepšiť jeho vlastnosti. Modifikácia MLESAC [13] sa oproti pôvodnej verzii líši v tom, že skóre kandidátneho útvaru nie je vypočítané z jeho kardinality, ale z maximálnej vierohodnosti, ktorú dosahujú jeho inliery. Táto optimalizácia má za následok vylepšenie v oblasti robustnosti algoritmu, no neprináša nijaké vylepšenie rýchlosti výpočtu. O vylepšenie tohto aspektu sa snaží modifikácia R-RANSAC [5]. Tá upravuje vyhodnocovanie kandidátnych útvarov pri ich generovaní tak, že prvotne útvar testuje voči podmnožine dát nad ktorými pracuje a pokračuje vo vyhodnocovaní skóre iba v prípade, že kandidátny útvar vyhovuje tejto podmnožine. V opačnom prípade je kandidát okamžite zavrhnutý ako nevhodný. Z podstaty tejto metódy sa dá usúdiť, že bude pracovať dobre v situáciách, keď sa v datasete, resp. v oblasti blízkej aktuálne skúmanému útvaru, nachádza väčšie množstvo inlierov.

Kapitola 3

Návrh detektora jednoduchej geometrie

Táto kapitola obsahuje popis návrhu aplikácie na detekciu jednoduchej geometrie, ktorá je základným produktom tejto práce. Nasledujúce sekcie sú zoradené logicky na časti potrebné pre aplikáciu. V sekcii 3.1 je popis kostry programu a blokov, z ktorých je zložená. Ďalšia sekcia 3.2 obsahuje návrh spôsobu výpočtu normál bodov spolu s ich korektnou orientáciou. Sekcia 3.3 sa zaoberá návrhom modelov, ktoré sú dostupné na detekciu vo výslednej aplikácii a sekcia 3.4 obsahuje návrh práce RANSAC algoritmu a zmien nad ním prevedených spolu s ich zdôvodnením. Posledná sekcia 3.5 ponúka náhľad do rozhrania programu.

3.1 Základná štruktúra

Návrh základnej štruktúry je dôležitou časťou tvorby každej aplikácie. Táto aplikácia sa v tomto ohľade snaží o maximálne využitie už existujúcej kostry, ktorá je upravená podľa potreby. Za existujúcu kostru je považovaný článok *Efficient RANSAC for Point-Cloud Shape Detection* [11].

Celková štruktúra programu bola rozložená na pracovné bloky, ktoré poskytujú potrebnú funkcionalitu. Tieto bloky sú v hlavnej časti programu integrované dokopy tak, aby tvorili jeden funkčný celok. Z pohľadu pracovných blokov sa teda kostra programu skladá z nasledovných častí:

- výpočet normál
- modely detekovaných útvarov
- RANSAC algoritmus
- vstupy a výstupy

Bližší popis týchto blokov, ktoré tvoria dokopy celkovú funkčnosť programu, sa nachádza v ďalších sekciách tejto kapitoly, pričom každému bloku je venovaná samostatná sekcia.

Z pohľadu pracovného procesu je štruktúra aplikácie navrhnutá pomerne jednoducho. Prvotnou fázou aplikácie je spracovanie vstupných parametrov, ktoré je potrebné pri každej trochu zložitejšej aplikácii. V ďalšej fáze dochádza k výpočtu normál bodov dodaných ako vstup. Táto je do istej miery voliteľná, nakoľko je možné ju preskočiť za predpokladu, že vstupné dáta už obsahujú predpočítané normály. Po dokončení tejto fázy sú vstupné

dáta pripravené na použitie. Ďalej nasleduje tvorba zvolených modelov a nastavenie ich parametrov, ktoré potom spolu s pripravenými dátami vstupujú do samotného výpočtu. V tejto časti prebieha vlastná detekcia zvolených útvarov podľa zadaných parametrov. Po jej dokončení sú potom výsledky zobrazené a voliteľne uložené.

3.2 Výpočet normál

Táto časť predstavuje prvý krok výpočtu, ktorý zabezpečuje úpravu dát v prípade potreby. Ako bolo v predchádzajúcej sekcii uvedené je možné ho vynechať za predpokladu, že vstupné dáta už obsahujú vypočítané normály.

Normály bodov sú pre detekciu navrhnutým spôsobom nutné z dôvodu redukcie počtu bodov minimálnych množín detekovaných útvarov. Sú počítané ako normály dotykových plôch bodov, ktoré sú aproximáciou lokálneho povrchu vytvoreného z okolitých bodov. Spôsob výpočtu použitý v tejto práci, ktorý je ďalej popisovaný je čiastočnou modifikáciou výpočtu z článku *Surface Reconstruction from Unorganized Points* [2].

Výpočet hodnôt

Každá dotyková plocha $Tp(\mathbf{x}_i)$ patriaca bodu $\mathbf{x}_i$ zo vstupných dát je reprezentovaná ako stred $\mathbf{o}_i$ spolu s jednotkovým normálovým vektorom $\hat{\mathbf{n}}_i$. Znamienková vzdialenosť náhodného bodu $\mathbf{p} \in \mathbb{R}^3$ od plochy $Tp(\mathbf{x}_i)$ je definovaná ako $\text{dist}_i(\mathbf{p}) = (\mathbf{p} - \mathbf{o_i}) \cdot \hat{\mathbf{n}}_i$. Stred a normála plochy $Tp(\mathbf{x}_i)$ sa určia tak, že sa vytvorí množina $Nbhd(\mathbf{x_i})$ z k bodov vstupných dát X, ktoré sú najbližšie bodu $\mathbf{x}_i$. Táto množina sa nazýva k-okolie bodu $\mathbf{x}_i$. Toto k môže byť fixované číslo alebo môže byť určené napr. ako počet bodov v guli o polomere r. V tejto práci je použité k fixnej veľkosti, ktorá je zadaná ako vstupný parameter. Stred a normála sú vypočítané tak, že plocha $\text{dist}_i(\mathbf{p}) = 0$ je najlepšia plocha vypočítaná pomocou metódy najmenších štvorcov pre množinu $Nbhd(\mathbf{x_i})$ t.j. stred $\mathbf{o}_i$ je zvolený ako ťažisko $Nbhd(\mathbf{x_i})$ a normála $\hat{\mathbf{n}}_i$ je určená pomocou metódy PCA [25].

Vo všeobecnosti však neexistuje metóda, pomocou ktorej by sa dala určiť orientácia normály a výsledná orientácia po vypočítaní normály metódou PCA ostáva náhodná a nekonzistentná naprieč celým vstupným point cloudom. Ak by bol vstupným datasetom point cloud z jedného snímku, resp. z viacerých snímkov získaných z rovnakého miesta, a bol by k nemu dostupný počiatočný bod skenera, riešením by bolo natočenie vypočítaných normál smerom k tomuto počiatočnému bodu. Keďže ale táto práca počíta s možnosťou použitia globálnych point cloudov, pre ktoré takýto spoločný bod nie je dostupný z bližšie nešpecifikovaného dôvodu, toto riešenie nemôže byť použité. Čo je horšie, pre takýto prípad neexistuje nijaké triviálne riešenie. Vo vyššie spomínanom článku [2] je prezentovaný spôsob na konzistentné natočenie normál pomocou propagácie orientácie normály, definovaný ako problém optimalizácie grafu.

Natočenie normál

Ak platí predpoklad, že dva body $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}$ sú si geometricky blízke, potom v ideálnom prípade, keď sú dáta navzorkované dostatočne husto a pomyselný povrch, ktorý tvoria, je hladký, dotykové plochy korešpondujúce týmto bodom $Tp(\mathbf{x}_i) = (\mathbf{o_i}, \hat{\mathbf{n}}_i)$ a $Tp(\mathbf{x}_j) = (\mathbf{o_j}, \hat{\mathbf{n}}_j)$ sú takmer rovnobežné t.j. $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j \approx \pm 1$. Ak majú plochy rovnakú orientáciu potom $\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j \approx +1$ v opačnom prípade by jedna z nich mala byť otočená podľa druhej. Aby tento predpoklad platil globálne, mal by platiť pre všetky „dostatočne blízke“ body datasetu. Riešením je

použitie aproximačného algoritmu nad grafom vytvoreným tak, že graf obsahuje jeden uzol N_i ako zástupcu každej dotykovej plochy $Tp(\mathbf{x}_i)$.

Bod vytvárania hrán tohto grafu je momentom, kedy sa riešenie použité v tejto práci viac nezhoduje s originálnym riešením. Keďže vstupný point cloud nemusí pozostávať z jedného spojitého konvexného útvaru, nie je prvotne vytváraná minimálna kostra grafu pre celý dataset. Hrany grafu sú vytvárané iba nasledovne – do grafu je pridaná hrana (i, j) ak $\mathbf{o_i}$ je v k-okolí $\mathbf{o_j}$ alebo $\mathbf{o_j}$ je v k-okolí $\mathbf{o_i}$. Preferovaný smer propagácie orientácie normály je od $Tp(\mathbf{x}_i)$ ku $Tp(\mathbf{x}_j)$, ak sú dotykové plochy takmer rovnobežné. Na docielenie tohto efektu je každej hrane priradená váha $1-|\hat{\mathbf{n}}_i \cdot \hat{\mathbf{n}}_j|$, ktorá má tie pozitívne vlastnosti, že je nezáporná a malej hodnoty, ak sú príslušné dotykové plochy takmer rovnobežné. Posledným krokom pred samotnou propagáciou je zvolenie koreňa, od ktorého sa začne propagovať a zvolenie orientácie jeho normály. Táto práca používa ako koreň bod s najväčšou z hodnotou a orientácia normály je nastavená smerom k $+z$ osi. Propagácia potom prebieha tak, že je prehľadávaná smerom do hĺbky minimálna kostra grafu.

Keďže prvotne nebola nad celým datasetom vytvorená minimálna kostra, tento môže byť fragmentovaný do niekoľkých zhlukov, z ktorých každý má vlastný graf. Aby prebehla orientácia normály nad celým datasetom, je propagácia prevedená postupne nad každým fragmentom, pričom každý z fragmentov má zvolený koreň podľa vyššie uvedeného princípu.

3.3 Modely detekovaných útvarov

Každý RANSACom detekovaný útvar potrebuje model, v ktorom sa definujú jeho parametre a spôsob ako sa s ním pracuje. Táto práca podporuje tri typy útvarov:

- plocha
- valec
- guľa

Hoci sú tieto útvary na prvý pohľad značne rozdielne, z pohľadu využitia v RANSACu majú spoločné črty. Každý z útvarov má minimálnu množinu určitej veľkosti, ktorá sa dá použiť na výpočet parametrov modelu, ktorých hodnoty unikátne definujú konkrétny kandidátny útvar. Ďalšou spoločnou vlastnosťou je fakt, že všetky modely sú testované voči dvom podmienkam – maximálnej euklidovskej vzdialenosti a uhlovej odchýlke. Obe tieto hodnoty sú vo vytvorenej aplikácii zadávateľné pomocou vstupných parametrov. Pre modely samotné je však najdôležitejšie navrhnúť, v akom tvare budú ich parametre vyjadrené a akým spôsobom budú počítané.

Plocha

Je prvým modelom a jej minimálna množina pozostáva z troch bodov. Tieto body sú použité na výpočet parametrov v takzvanom Hessiánskom normálovom tvare – $\hat{\mathbf{n}} \cdot \mathbf{x} = -p$. Tento tvar sa dá získať aj z klasického parametrického vyjadrenia roviny v priestore prevodom pomocou jednoduchých vzorcov [16]. $\hat{\mathbf{n}} = (n_x, n_y, n_z)$ je jednotkový normálový vektor a $\mathbf{x}$ je bod patriaci tejto ploche. Konštanta $-p$ z tohto vzorca vyjadruje vzdialenosť plochy od stredu súradnicovej sústavy. Znamienko konštanty p ďalej určuje, na ktorej strane plochy sa tento stred nachádza. Ak je p kladné, stred sa nachádza v polrovine, kde smeruje normálový vektor tejto plochy. Prakticky sa parametre vypočítajú tak, že sa z bodov minimálnej

množiny vytvoria dva vektory patriace do tejto roviny. Nad týmito vektormi sa vykoná vektorový súčin, ktorý určí normálový vektor tejto plochy a tento sa ešte normalizuje, aby bol jednotkový. Jeho smer je nastavený konzistentne so smermi normál jednotlivých bodov minimálnej množiny. Následne sa ešte dopočíta Hessiánsky komponent p. Plocha je akceptovaná ako kandidátny útvar iba v prípade, že jej normála sa neodchyľuje od normál bodov, z ktorých bola vytvorená, o viac ako je povolený prah.

Valec

Je ďalším z podporovaných modelov. Jeho minimálna množina sa skladá z dvoch bodov. Parametrami tohto útvaru sú rotačná os, bod na tejto osi a jeho polomer. Pre každé dve priamky v priestore

$$\begin{aligned} \mathbf{L_1} : P(s) &= P_0 + s(P_1 - P_0) = P_0 + s\mathbf{u} \\ \mathbf{L_2} : Q(t) &= Q_0 + t(Q_1 - Q_0) = Q_0 + t\mathbf{v} \end{aligned} \tag{3.1}$$

platí, že majú od seba najmenšiu vzdialenosť v unikátnych bodoch $P_C = P(s_C)$ a $Q_C = Q(t_C)$, pre ktoré je $\mathbf{w}(s_C, t_C)$ unikátnym minimom $\mathbf{w}(s,t) = P(s) - Q(t)$. Ak naviac $\mathbf{L_1}$ a $\mathbf{L_2}$ nie sú rovnobežné a navzájom sa nepretínajú, potom segment $P_C Q_C$ spájajúci tieto body je unikátne kolmý na obe priamky zároveň. Z toho vyplýva, že pre $\mathbf{w}_C = \mathbf{w}(s_C, t_C)$:

$$\begin{aligned} \mathbf{u} \cdot \mathbf{w}_C &= 0 \\ \mathbf{v} \cdot \mathbf{w}_C &= 0 \end{aligned} \tag{3.2}$$

Tieto dve rovnice je možné vyriešiť substituovaním $\mathbf{w}_C = P(s_C) - Q(t_C) = \mathbf{w}_0 + s_C\mathbf{u} - t_C\mathbf{v}$, kde $\mathbf{w}_0 = P_0 - Q_0$. Tým dostaneme sústavu:

$$\begin{aligned} (\mathbf{u} \cdot \mathbf{u})s_C - (\mathbf{u} \cdot \mathbf{v})t_C &= -\mathbf{u} \cdot \mathbf{w}_0 \\ (\mathbf{v} \cdot \mathbf{u})s_C - (\mathbf{v} \cdot \mathbf{v})t_C &= -\mathbf{v} \cdot \mathbf{w}_0 \end{aligned} \tag{3.3}$$

Nahradením $a = \mathbf{u} \cdot \mathbf{u}$, $b = \mathbf{u} \cdot \mathbf{v}$, $c = \mathbf{v} \cdot \mathbf{v}$, $d = \mathbf{u} \cdot \mathbf{w}_0$, $e = \mathbf{v} \cdot \mathbf{w}_0$ sa s_C a t_C vypočíta z rovníc

$$s_C = \frac{be - cd}{ac - b^2} \quad \text{a} \quad t_C = \frac{ae - bd}{ac - b^2} \tag{3.4}$$

ak $ac - b^2$ je nenulový. Keďže $ac - b^2 = |\mathbf{u}|^2|\mathbf{v}|^2 - (|\mathbf{u}||\mathbf{v}|\cos\theta)^2 = (|\mathbf{u}||\mathbf{v}|\sin\theta)^2 \geq 0$, jediná možnosť, ktorá ostáva je prípad, keď $ac - b^2 = 0$. Vtedy sú priamky rovnobežné a vzdialenosť medzi nimi je konštantná. V tomto prípade sú s_C a t_C vypočítané tak, že jeden parameter je fixovaný a druhý sa získa výpočtom hociktorej z rovníc. Ak teda $s_C = 0$, potom $t_C = d/b = e/c$ [12]. Z týchto koeficientov sa dá ďalej vypočítať rotačná os a bod na nej ležiaci. Posledný parameter, ktorým je polomer, je vypočítaný ako vzdialenosť prvého z bodov minimálnej množiny k získanej osi. Keďže normály bodov minimálnej množiny budú na rotačnú os valca vždy kolmé, nie je potrebné počítať ich deviáciu voči povrchovým normálam valca v daných bodoch a kontrolovať ju voči zadanému prahu. Dostačujúce je skontrolovať, či tieto normály smerujú k osi alebo od nej. Posledným krokom je kontrola voči voliteľnému minimálnemu a maximálnemu možnému polomeru.

Guľa

Je posledným z podporovaných útvarov. Jej parametrami sú stred a polomer, ktoré sú získané z minimálnej množiny štyroch bodov. Štyri body tvoria v priestore unikátnu guľu za predpokladu, že neležia všetky v jednej rovine. Ak by ležali, v tomto prípade by z nich bolo možné vytvoriť nekonečné množstvo gúľ, ak by sa všetky nachádzali na jednej kružnici alebo by neexistovala nijaká guľa popísateľná pomocou týchto bodov. Guľu prechádzajúcu štyrmi nekoplanárnymi bodmi je možné vypočítať riešením determinantu

$$\begin{vmatrix} x^2+y^2+z^2 & x & y & z & 1 \\ x_1^2+y_1^2+z_1^2 & x_1 & y_1 & z_1 & 1 \\ x_2^2+y_2^2+z_2^2 & x_2 & y_2 & z_2 & 1 \\ x_3^2+y_3^2+z_3^2 & x_3 & y_3 & z_3 & 1 \\ x_4^2+y_4^2+z_4^2 & x_4 & y_4 & z_4 & 1 \end{vmatrix} = 0 \tag{3.5}$$

Ten je možné vyriešiť výpočtom kofaktorov jeho prvého riadku. Determinant je teda možné vyjadriť ako rovnicu $(x^2+y^2+z^2)M_{11}-xM_{12}+yM_{13}-zM_{14}+M_{15}=0$. Keďže $(x^2+y^2+z^2)=r^2$ je možné ju zjednodušiť na tvar

$$r^2 - x\frac{M_{12}}{M_{11}} + y\frac{M_{13}}{M_{11}} - z\frac{M_{14}}{M_{11}} + \frac{M_{15}}{M_{11}} = 0 \tag{3.6}$$

Všeobecná rovnica gule s polomerom r_0 a stredom so súradnicami (x_0, y_0, z_0) je $(x-x_0)^2+(y-y_0)^2+(z-z_0)^2-r_0^2=0$. Expandovaním a preusporiadaním vznikne rovnica

$$r^2 - 2xx_0 - 2yy_0 - 2zz_0 + x_0^2 + y_0^2 + z_0^2 - r_0^2 = 0 \tag{3.7}$$

Zrovnaním korešpondujúcich elementov rovníc 3.6 a 3.7 vznikne sústava rovníc

$$\begin{aligned} x_0 &= +0,5\frac{M_{12}}{M_{11}} \\ y_0 &= -0,5\frac{M_{13}}{M_{11}} \\ z_0 &= +0,5\frac{M_{14}}{M_{11}} \\ r_0^2 &= x_0^2 + y_0^2 + z_0^2 - \frac{M_{15}}{M_{11}} \end{aligned} \tag{3.8}$$

Pri bližšom skúmaní je vidieť, že pre $M_{11}=0$ neexistuje správne riešenie. To znamená, že v tomto prípade body neležia na jednej guli [10]. Dosadením do rovníc sú nájdené hodnoty parametrov a teda príslušný kandidátny útvar. Ďalším krokom je kontrola deviácií normál bodov minimálnej množiny voči povrchovým normálam výsledného kandidáta v príslušných bodoch či vyhovujú zadanému prahu. Ak kandidát vyhovie, poslednou kontrolou je vyhovenie minimálnemu a maximálnemu možnému polomeru, ktoré sú voliteľnými prahmi.

3.4 RANSAC algoritmus

Tento blok je hnacou silou celého programu, keďže zabezpečuje samotnú detekciu zvolených útvarov. Pre bližšie pochopenie rozhodnutí, ktoré sprevádzali návrh tohto bloku, sa v tejto sekcii nachádzajú aj odvôvodnenia, ktoré k týmto rozhodnutiam viedli a porovnanie s inými riešeniami, najmä tým z referenčného článku.

Nakoľko cieľom tejto práce nie je demonštrácia existujúcich optimalizačných techník detekcie jednoduchej geometrie s využitím metódy RANSAC, optimalizačné techniky popisované v referenčnom článku [11] sú z väčšej časti opomenuté a použitý je iba čistý základ. To znamená, že táto práca používa tradičný globálny sampling minimálnych množín čo má vplyv na výber varianty RANSAC algoritmu. R-RANSAC použitý v referenčnom článku dobre komplementuje prezentovaný lokalizovaný sampling, vďaka jeho predtestu. Práve to je však dôvod, prečo pre túto prácu nie je vhodný. S globálnym samplingom by totiž s veľkou pravdepodobnosťou dochádzalo k veľkému počtu zamietnutých kandidátov, ktorí by inak vyhovovali, čo by malo negatívny dopad na vlastnosti detekcie. Pre túto prácu je nakoniec zvolený ako základ pôvodný RANSAC algoritmus z roku 1981 [1]. Jeho úprava spočíva v pridaní možnosti detekovať naraz viacero útvarov.

Klasický prístup a jeho problémy

Tradične je RANSAC algoritmom jednomodelovým. Na detekciu viacerých útvarov je teda potrebné spustiť RANSAC pre jednotlivé hľadané útvary osobitne a rozhodnúť, ktorý výsledok je najviac vyhovujúci. Týmto výsledkom potom zvyčajne bude model, ktorý dobre aproximuje najviac bodov point cloudu, rovnako ako pri behu algoritmu a hľadaní najlepšieho kandidátneho útvaru pre konkrétny model. Tento prístup má však potenciál byť veľmi neefektívny. Ak by sa celý point cloud skladal z jedného detekovaného útvaru, pričom by nebolo možné nijakú jeho časť aproximovať iným útvarom, nemusel by byť nikdy nájdený ani len jeden vyhovujúci kandidátny útvar pre zvyšné detekované modely a takýto program by sa dostal do nekonečného cyklu. Tomuto problému je možné pomerne jednoducho zabrániť pridaním hranice maximálneho počtu vygenerovaných nevyhovujúcich kandidátnych útvarov, po ktorých algoritmus skončí bez výsledku. Problémom, ktorý naopak pretrváva je dĺžka výpočtu. Hoci útvar, z ktorého je vstupný point cloud zložený, bude detekovaný takmer okamžite, ďalšie útvary budú mať kandidátov, ktorí budú aproximovať v lepšom prípade len zlomkovú časť tohto vstupu. Ako už bolo v predchádzajúcej kapitole popísané, algoritmus sa ukončí po istom počte vygenerovaných kandidátnych útvarov, kedy bude pravdepodobnosť úspešnej detekcie dostatočne vysoká. 2.1 Táto je však závislá na skóre dovtedy najlepšieho nájdeného kandidáta, ktorým je zvyčajne počet bodov, ktoré sú ním dobre reprezentované. Počet kandidátov potrebných na úspešnú detekciu ostatných útvarov bude preto niekoľkonásobne väčší, čo výrazne zpomalí celkovú detekciu.

Navrhovaná modifikácia

Modifikácia RANSACu, ktorá je tu prezentovaná, rieši hľadanie viacerých útvarov odlišným spôsobom. Algoritmus pracuje so všetkými typmi útvarov naraz a to tak, že v jednej iterácii vygeneruje kandidátny útvar pre každý z hľadaných modelov. Každý útvar vybraný na detekciu si uchováva vlastné informácie o najlepšom kandidátovi zo svojej kategórie a na globálnej úrovni algoritmu je udržiavaná len informácia o tom, ktorá kategória má momentálne kandidáta s najlepším skóre. V prípade, že táto kategória dosiahne potrebný počet vygenerovaných kandidátov na svoje ukončenie, celý algoritmus je ukončený a nie je potrebné generovať ďalších kandidátov zo zvyšných kategórií. Na druhej strane, ak niektorý z hľadaných útvarov vygeneroval dostatok kandidátov na úspešné ukončenie detekcie no nie je globálne najlepším, ďalšie útvary z jeho kategórie nie sú viac generované, pretože šanca na nájdenie lepšieho kandidáta je už malá a pokračovať by bolo neefektívne. Algoritmus ako celok ale pokračuje ďalej so zvyškom útvarov, ktoré ešte stále môžu vygenerovať lepšieho kandidáta. Toto riešenie bude v najhoršom prípade rovnako pomalé ako sekvenčné, pretože

bude musieť vygenerovať dostatočný počet kandidátov na ukončenie pre všetky útvary. Keďže ale detekcia hľadá modely od najväčšieho smerom k menším, v praxi by mal byť výsledok detekcie vyprodukovaný rýchlejšie. Otázka kvality vyprodukovaného výsledku je o niečo zložitejšia. Po nájdení najlepšieho globálneho kandidáta sa vo všeobecnosti výpočet ukončí predtým ako sa nájdu riešenia pre každú kategóriu. Mohlo by teda dôjsť ku stavu, že niektorý z útvarov, ktorých detekcia bola ukončená skôr ako vyprodukovali potrebný počet kandidátov na ukončenie s určenou pravdepodobnosťou, by bol eventuálne našiel lepšieho globálneho kandidáta. Toto by de-facto znamenalo, že by detekovaný útvar bol horším ako ten, ktorý by bol vyprodukovaný v rovnakej situácii sekvenčným RANSACom. Nakoľko je však RANSAC nedeterministický a táto situácia môže nastať aj v prípade jeho sekvenčného použitia, hoci s menšou pravdepodobnosťou, nie je dôvod na obavu, že dôjde k značnému poklesu kvality výsledku v prospech rýchlosti výpočtu. K tomuto pohľadu prispieva aj fakt, že sa bežne používa na podmienku ukončenia pomerne vysoký prah napr. 99 %. Ak existuje takáto vysoká pravdepodobnosť, že nejaký útvar sa vo vstupných dátach nachádza, je nepravdepodobné, že by sa jednalo o chybnú klasifikáciu. Najpravdepodobnejším vysvetlením tohto javu by snáď mohlo byť, že vo vstupných dátach existuje ešte ďalší objekt s o málo lepším skóre iného typu a teda momentálne detekovaný objekt by bol pravdepodobne pri sekvenčnom RANSACu výsledkom ďalšieho behu na zvyšnom vstupe. Predchádzajúce tvrdenie je však len špekuláciou a neopiera sa o nijaké konkrétne teoretické modely alebo reálne výsledky.

3.5 Vstupy a výstupy

Aby bol program použiteľný a detekcia nastaviteľnou, je potrebné dodať jej parametre, ktorými sa bude riadiť. Z pracovných blokov programu je vstupno-výstupný blok jediný, ktorý je priamo dostupný zvonka programu. Vstupy teda odrážajú predpokladané požiadavky užívateľa na nastavenie detekcie. Rovnako je potrebné nejakým spôsobom prezentovať výsledky užívateľovi. So zreteľom na úlohu aplikácie sa ako najvhodnejší spôsob javí grafická reprezentácia, vstupné rozhranie však bude pre jednoduchosť implementované v konzole.

Z vyššie popísaných sekcií vyplývajú približne tieto skupiny parametrov:

- všeobecné
- počítanie normál
- detekcia
- vizualizácia

Program má navrhnuté dva základné módy, ktoré odrážajú predpokladané potreby užívateľa – výpočtový a vizualizačný. Všeobecná skupina parametrov je spoločná pre oba programové módy a obsahuje parameter na prepínanie medzi týmito módmi. Keďže oba módy budú pracovať so vstupom, ďalším parametrom, ktorý bude slúžiť obom je vstupný súbor. Každý program by mal tiež obsahovať parameter slúžiaci na zobrazenie pomocníka a tento program túto konvenciu dodržuje. Ďalšou dobrou konvenciou, ktorá je dodržaná v tejto aplikácii, je možnosť meniť úroveň vypisovaných kontrolných hlášok.

Vizualizačný mód

Je jednoduchším z programových módov a umožní zobraziť zadaný vstup bez nutnosti priebehu detekcie. Toto je vhodným doplnkom, nakoľko detekcia útvarov môže potenciálne trvať dlhší čas a užívateľ bude s najväčšou pravdepodobnosťou chcieť zobraziť výsledky aspoň v niekoľkých prípadoch viac než raz.

Výpočtový mód

Je zaujímavejším a značne zložitejším, pretože zastrešuje parametre riadiace časti detekcie. Parametre zahrnuté v tejto skupine reprezentujú možnosti nastavenia jej jednotlivých etáp. Pre počítanie normál je samozrejmým parametrom veľkosť použitého k-okolia. Keďže však táto časť výpočtu nemusí byť využitá, je rovnako potrebné, aby existoval parameter, ktorý toto umožní nastaviť. Pre detekciu samotnú existuje hneď niekoľko očividných parametrov. Za základný by sa dal považovať výber modelov, ktoré chceme detekovať. Rovnako dôležité sú aj parametre nastavujúce maximálnu povolenú euklidovskú a uhlovú odchýlku. Ďalším veľmi podstatným parametrom je spôsob ukončenia detekcie. Pre užívateľské pohodlie je tu navrhnutá voľba medzi detekciou konkrétneho počtu útvarov a detekovaním všetkých útvarov väčších ako želaný počet bodov. Táto voľba samozrejme musí mať aj párový parameter, ktorý bude obsahovať hodnotu pre voľbu predošlého parametra. Za nepovinné by sa dali považovať voľby minimálneho a maximálneho možného polomeru, ktorý bude použitý u modelov majúcich polomer a bude tak limitovať veľkosť vyhľadávaných kandidátnych útvarov. Výsledky detekcie môžu byť ďalej použité na iné účely a tak ďalším parametrom je možnosť uložiť ich do výstupného súboru. Minimálne pre vizualizačný mód tohto programu je to zaiste vhodným komplementom. Výstupný program tejto práce navrhuje prezentáciu výsledkov aj graficky. Na jasnú reprezentáciu jednotlivých detekovaných útvarov je preto použité farebné rozlíšenie. Spôsob tohto farebného rozlíšenia je tiež vhodným voliteľným vstupným parameterom.

Potreby vizualizácie

Použiteľná vizualizácia výsledkov musí taktiež spĺňať isté požiadavky. S ohľadom na rozsah aplikácie, ktorá je produktom tejto práce, sú však tieto považované za iba základného charakteru. Keďže program pracuje s 3D dátami, je potrebné, aby vizualizácia obsahovala možnosť posuvov, rotácií, približovania a odďalovania výsledku a nakoľko vizuálne mení vstup, je navrhnuté zobrazenie východiskového stavu povedľa výsledku.

Kapitola 4

Implementácia

Táto kapitola obsahuje detaily implementácie výsledného programu. Program bol implementovaný v jazyku C++. Mnoho funkcií, potrebných v tomto programe, sa podarilo nájsť v použitých knižniciach, no v nie úplne konečnom tvare. Takmer vždy bolo nutné upraviť isté časti, aby odrážali návrh prezentovaný v kapitole 3. Ak bola niektorá časť kompletne prebraná bez akejkoľvek úpravy, je tento fakt uvedený v komentári zdrojového kódu. Rovnako tak sú v komentároch v krátkosti zhrnuté vykonané zmeny. Implementácia by sa tak dala charakterizovať skôr ako modifikácia existujúcej funkcionality pre potreby aplikácie a takým spôsobom je aj popísaná.

V sekciách, venujúcich sa implementácii jednotlivých častí programu, nie sú uvádzané detaily ako sú implementované funkcie z použitých knižníc, ktoré nie sú absolútne nevyhnutné. Je možné predpokladať, že použitá funkcia vyhovuje návrhu práce. Podrobne popisované sú len jednotlivé modifikácie funkcií, ktoré sú produktom tejto práce. V prípade záujmu sa bližšie informácie nachádzajú v dokumentáciách príslušných knižníc.

V sekcii 4.1 sa nachádza prehľad nástrojov použitých pri implementácii a sekcia 4.2 obsahuje popis štruktúry pracovného procesu programu a jeho, oproti návrhu rozšíreným, možnostiam. Ďalšie sekcie 4.3, 4.4 a 4.5 obsahujú popis implementácie hlavných pracovných blokov aplikácie. V záverečnej sekcii 4.6 je potom popis vstupno-výstupného rozhrania programu.

4.1 Použité nástroje

Pri tvorbe aplikácie bolo využitých niekoľko nástrojov, predovšetkým knižníc. Hlavnou z nich bola *Point Cloud Library (PCL)* [21]. Podobne ako referenčný článok [11] tvoril teoretický základ, PCL tvorila programový základ. Knižnica je značne komplexná a obsahuje množstvo funkcií často používaných v oblasti počítačového videnia. Veľa navrhnutých častí tohto programu sa nachádza v nejakej forme implementovaných v tejto knižnici, čoho bolo maximálne využité. Pochádzali hlavne z modulov *common*, *features*, *io*, *sample consensus*, *search* a *visualization*. Táto knižnica sama využíva množstvo ďalších knižníc, na ktorých základe je postavená. Niektoré z nich boli okrem poskytovania podpory pre PCL použité aj priamo pri vytváraní programového kódu.

Prvou takou je *Boost* [22]. Aj táto knižnica je veľmi komplexnou, no nezaoberá sa čisto počítačovým videním. Je všeobecnejšia a obsahuje funkcie potrebné pri programovaní vo viacerých oblastiach informatiky. V tejto práci boli využité hlavne jej časti *Graph* a *Program Options*.

Ďalšou priamo použitou knižnicou je *Eigen* [23], ktorá obsahuje matematické funkcie zaoberajúce sa lineárnou algebrou. Hoci bola použitá priamo, bola použitá iba v situáciách súvisiacich s PCL.

Knižnice, ktoré boli použité iba skryto prostredníctvom PCL nasledujú ďalej. Prvou z nich je *Visualization Toolkit (VTK)* [18]. Táto knižnica obsahuje nástroje na prácu s 3D grafikou, spracovanie obrazu a vizualizáciu. V PCL je použitá ako podpora pre modul *visualization*. Ďalšou je *Fast Library for Approximate Nearest Neighbors (FLANN)* [20], ktorá obsahuje kolekciu funkcií implementujúcich rôzne typy hľadania k-najbližších susedov. Je využitá v rôznych moduloch PCL napr. v module *search*. Poslednou je voliteľná knižnica *OpenNI* [19], ktorá obsahuje open-source ovládače na použitie rôznych 3D skenerov ako napr. MS Kinect. Táto je použitá PCL modulom *io*.

Konkrétne funkcie z jednotlivých modulov použitých knižníc, ktoré boli využité v tomto programe, budú spomentuté v im príslušiacich sekciách.

Táto aplikácia používa tieto verzie jednotlivých knižníc:

- PCL 1.7.1
- Boost 1.53.0
- Eigen 3.2.0 spolu s unsupported súčasťami
- VTK 5.8.0
- FLANN 1.8.4
- OpenNI 1.5.4

V súvislosti s knižnicou OpenNI stojí za zmienku, že po akvizícii spoločnosti PrimeSense, ktorá bola zakladateľom organizácie OpenNI, spoločnosťou Apple, bola webová stránka tejto organizácie OpenNI.org zrušená dňa 23.4.2014. Jej materiály však boli uchované rôznymi spoločnosťami, ktoré ich používali [26].

Každú z týchto knižníc je možné nájsť na stiahnutie v použitej alebo vyššej verzii na webových stránkach v príslušných citáciách.

4.2 Základná štruktúra

Základná štruktúra pracovného procesu výsledného programu sa riadi návrhom v tejto práci, ale má pre každú sekciu programu dostupných viac volieb. S ohľadom na použité nástroje a implementáciu navrhovaných blokov tak, že sú z veľkej časti kompatibilné s použitými knižnicami, existuje vo výslednej aplikácii možnosť zvoliť si, ktoré súčasti budú pri detekcii použité. Je možné použitie akejkoľvek kombinácie normál, modelov a detekčného RANSAC algoritmu, teda existuje osem rôznych kombinácií. Toto rozšírenie volieb je veľmi výhodné z pohľadu testovania vplyvov normál na detekciu, čo je jedným z hlavných cieľov tejto práce. Rozšírenie dostupných volieb sa odráža aj v náraste počtu vstupných parametrov aplikácie, ktorých popis sa nachádza sekcii 4.6 tejto kapitoly.

Pre výpočet normál, ktoré sú už navrhnuté ako voliteľné, neexistuje žiadne rozšírenie nad rámec návrhu a u modelov toto predstavuje iba jednoduchý výber medzi originálnymi a modifikovanými modelmi pred ich vytváraním. V prípade výberu RANSAC algoritmu sa objavujú menšie zmeny spôsobu práce, ktoré musia existovať z podstaty modifikácii algoritmu, ktoré sú nad originálnou verziou vykonané. Keďže originálna verzia RANSACu je

jednomodelová, na detekciu viacerých modelov je nutné vykonať detekciu každého z útvarov zvlášť a potom vybrať ten najlepší, čo u modifikovanej verzie nie je nutné.

4.3 Výpočet normál

Táto časť vychádza z triedy `pcl::NormalEstimation<PointInT, PointOutT>` modulu *features*. Tá dokáže vypočítať normály, pre daný vstup a veľkosť okolia, pomocou metódy najmenších štvorcov. Orientácia normál však prebieha pomocou natočenia smerom k danému bodu, ktorým je väčšinou bod akvizície dát. Toto, ako je v návrhu spomínané, nevyhovuje tejto práci a je nutné pozmeniť. `normal_estimation<PointInT, PointOutT>` je trieda vytvorená dedením z pôvodnej triedy za účelom globálne zabezpečiť správne natočenie normál. Okrem zmeny spôsobu natočenia pracuje nová trieda rovnako ako originálna implementovaná v knižnici PCL. Všetky možnosti ako zvolenie metódy, podľa ktorej sa bude hľadať k najbližších susedov, sú dostupné aj v novej triede.

Výmena metódy na otáčanie normál

Prvým krokom k implementovaniu správnej orientácie je zmena metódy, ktorá je volaná za účelom zabezpečenia natočenia normál, `pcl::flipNormalTowardsViewpoint()`. Tá je v novej triede nahradená metódou `propagate_normal_orientation()`. K tejto zmene dochádza v metóde `computeFeature(PointCloudOut &output)`, v ktorej sa uskutočňuje vypočítanie hodnôt normál. Nová metóda na natáčanie normál potrebuje k svojej práci stredy dotykových plôch, z ktorých bola hodnota normály vypočítaná. Tieto sú preto vo volajúcej metóde ukladané do dočasného point cloudu, ktorý je potom metóde na úpravu smeru natočenia poskytnutý.

Implementácia novej metódy otáčania normál

Ďalším krokom je implementácia novej metódy na nastavenie orientácie normál. Na to je použitá knižnica *Boost – Graph*. Keďže sa normála propaguje grafom, je potrebné najskôr tento graf vytvoriť. Ako prvá sa vyberie metóda hľadania k-najbližších susedov, podľa nastavení dodaných zvonka triedy. Keďže propagácia je prevádzaná na rovnakých dátach ako výpočet hodnôt normál, body blízko seba už majú tieto normály ovplyvnené zväčša rovnakými bodmi, pretože ich k-okolia z výpočtu hodnoty pozostávali z veľkého množstva spoločných bodov. Kôli pamäťovej optimalizácii je veľkosť použitého k-okolia pri otáčaní normál iba 10% zadanej veľkosti, čo by ale nemalo mať kôli vyššie popísanému dôvodu negatívny dopad na výsledok pri rozumnej veľkosti zadaného k-okolia. Pri veľkých datasetoch a/alebo okoliach však pamäťová náročnosť môže byť stále pomerne vysoká, čo vyplýva z podstaty vytvárania hrán grafu, ktorá je popísaná v návrhu. Graf samotný zastrešuje štruktúra `adjacency_list<...>`, ktorá je značne nastaviteľná pomocou jej šablónových parametrov. Po vytvorení grafu sa zvolí koreňový bod propagácie a jeho orientácia systémom popísaným v návrhu. Ako ďalší nasleduje výpočet minimálnej kostry grafu metódou `prim_minimum_spanning_tree()`. Keďže teoreticky nemusí byť celý graf vzájomne prepojený, vytvorí sa len kostra dosiahnuteľná z koreňového bodu. Táto sa potom prechádza pomocou metódy `undirected_dfs()`, v ktorej dochádza k samotnému otáčaniu normál. Po dokončení tohto fragmentu sa pristúpi k ďalšiemu a proces sa opakuje kým všetky fragmenty neprešli procesom otáčania normál.

Zaujímavosť z procesu implementácie

Zaujímavosťou môže byť, že počas implementácie bol potencionálne odhalený nezdokumentovaný problém v knižnici *Boost – Graph*. Funkcia `prim_minimum_spanning_tree()` má vo svojom popise oznámené, že na grafoch obsahujúcich násobné hrany neprodukuje správne výsledky. Graf v tejto práci túto podmienku spĺňa, t.j. neobsahuje násobné hrany. Problém však nastáva pri výskyte grafových sľučiek, kedy vo vnútri algoritmu vzniká chyba segmentácie, anglicky segmentation fault. Po dodatočnom odstránení výskytu sľučiek algoritmus pracuje bez problémov. Ak by aj bola grafová sľučka považovaná za násobnú hranu, čo grafová teória popiera, takáto forma „neprodukovania správnych výsledkov" je prinajmenšom podozrivá pri knižnici akou je *Boost*. Výskyt tohto problému bol počas vytvárania programu jednou z najnepríjemnejších vecí ktorá sa vyskytla, nakoľko jeho odhalenie bolo komplikované a podstatne zdržalo práce na vytváraní programu. Táto chyba bude nahlásená vývojárom knižnice za predpokladu, že sa vo voľnom čase podarí zreprodukovať túto chybu na minimálnom programe, čím bude potvrdené, že chyba nevznikla z iného dôvodu, napr. nesprávnym použitím funkcie.

4.4 Modely detekovaných útvarov

Modely pre RANSAC algoritmus sa tiež nachádzajú v knižnici PCL, no rovnako ako u počítania normál, nie sú úplne kompatibilné s návrhom tejto práce. Popis ich úprav nasleduje na ďalších riadkoch.

Časti potrebujúce úpravu

Zmeny na všetkých modeloch by sa dali označiť a popísať skupinovo, nakoľko každý model obsahuje tieto zmeny, no ich konkrétna implementácia sa líši podľa typu útvaru, na ktorom je prevedená. Prvou zmenou je rozdelenie kontroly maximálnej euklidovskej a uhlovej odchýlky. Modely pracujúce s normálami, relevantné pre túto prácu, sú v knižnici PCL implementované tak, že obsahujú premennú `normal_distance_weight_`, ktorá je váhou určujúcou podiel uhlovej deviácie na hodnote celkovej odchýlky. Hodnota $1 -$ `normal_distance_weight_` je potom váha euklidovskej odchýlky na hodnote odchýlky celkovej. [1] Vypočítané čiastkové odchýlky sú vynásobené svojimi váhami a sčítané, čím sa získa hodnota celkovej odchýlky, ktorá je testovaná voči danému prahu `threshold`. To v praxi znamená nasledovný druh problému, ktorý je najlepšie prezentovať na príklade: ak je napr. želaná maximálna deviácia uhla $30°$ a euklidovskej vzdialenosti $0,30\,m$ a každá z týchto dvoch odchýlok je rovnocenná, t.j. váha `normal_distance_weight_` $= 0,5$, je nutné si vybrať z týchto možností: buď bude v krajnej situácii, keď je jedna z odchýlok nulová možné zarátať bod s až dvojnásobnou veľkosťou ostávajúcej maximálnej odchýlky ako inlier alebo nebude možné zarátať ako inlier bod, ktorý má jednu odchýlku polovičnej veľkosti napr. odchýlku normály o veľkosti uhla $15°$ a zároveň ostávajúcu odchýlku väčšiu ako polovica želaného maxima, t.j. napr. euklidovskú odchýlku z intervalu $(15 - 30\rangle\, m$. Tieto fakticky nie sú jediné dve možnosti, ale prezentujú dobre problém tohto prístupu – nie je možné zaručiť, že bod zakaždým vyhovuje obom želaným podmienkam súčasne a kontrolovať tieto podmienky oddelene. Tieto odchýlky majú medzi sebou závislosť nepriamej úmernosti. Keďže

[1] Toto nie je úplna pravda, nakoľko výpočet v PCL v tomto vzorci pracuje aj so zakrivením predpokladaného povrchu v danom bode. Toto ale nie je podstatné pre prezentáciu podstaty problému so spôsobom výpočtu oproti návrhu tejto práce a teda môže byť opomenuté.

táto práca pripúšťa ako inliery iba body spĺňajúce obe podmienky naraz, je nutné tieto prahy oddeliť.

Implementácia potrebných zmien

Z týchto dôvodov sú preto pre každý útvar vytvorené nové triedy, ktoré dedia beznormálové verzie svojich náprotivkov z PCL. Pre cylinder, ktorý takúto verziu v PCL nemá, je tento efekt docielený dedením základnej triedy spoločnej pre všetky RANSAC modely PCL a čistým zkopírovaním metód, ktoré je potrebné prevziať. Nové triedy sa snažia v maximálnej možnej miere zachovať štruktúru existujúcu v PCL pokiaľ to nenarúša návrh tejto aplikácie. V taktomto stave by však nové triedy ešte neobsahovali normály. Pomocná trieda, ktorá túto funkcionalitu pridáva pre modely v PCL sa nazýva `pcl::SampleConsensusModelFromNormals<PointT, PointNT>` Delením kontroly prahov vzniká potreba upraviť túto triedu. Vzniká nová trieda `sacmodel_from_normals<PointT, PointNT>`, v ktorej sú zabezpečené potrebné úpravy. Tie pozostávajú z pridania dátovej štruktúry, ktorá udržiava vypočítanú uhlovú odchýlku zvlášť a pridania metód pracujúcich s touto štruktúrou. Keďže táto nová dátová štruktúra je ekvivalentom už existujúcej dátovej štruktúry, ktorá uchováva euklidovské/celkové odchýlky, aj metódy, ktoré je potrebné pridať, sú tie, ktoré majú ekvivalent pracujúci na pôvodnej štruktúre. Jedná sa napríkald o metódu `getAngularDistancesToModel()`, ktorá má ekvivalent v metóde `getDistancesToModel()`. Okrem nich je potrebné upraviť aj metódy, v ktorých dochádza k naplneniu pôvodnej štruktúry hodnotami celkovej odchýlky a zabezpečiť ukladanie jednotlivých odchýlok do im príslušiacich štruktúr. Táto pomocná trieda je ďalšou, ktorú dedí každá z novovytvorených tried modelov. Okrem pridania novej štruktúry a metód je práve táto trieda, po zdedení pôvodnej, zodpovedná za váhu `normal_distance_weight_`. Tá v nových modeloch nie je viac použitá ako váha, ale ako plnohodnotný prah maximálnej uhlovej deviácie, čo znamená, že predtým spoločný prah `threshold` v nových triedach slúži len na kontrolu euklidovskej odchýlky. Aby tieto modely kontrolovali každý prah osobitne, je potrebné pozmeniť testovacie podmienky v každej z metód, kde sa nachádzajú. Jedná sa napr. o metódy `countWithinDistance()`, `selectWithinDistance` a `doSamplesVerifyModel()`, ale tieto nie sú jediné.

Ďalšou modifikáciou pre každý z modelov je zmena spôsobu kontroly uhlovej odchýlky a pridanie tejto kontroly do metódy zodpovednej za počítanie parametrov pre kandidátne útvary. Modely z PCL kontrolujú uhlovú odchýlku do uhla 180°. Kontrola týmto spôsobom znamená, že bod je považovaný za patriaci útvaru bez ohľadu na to, či má jeho normála korektnú orientáciu alebo presne opačnú. Toto správanie je nevyhovujúce a je v nových triedach modelov patrične upravené. Rovnako tak je pre každý model do funkcie `computeModelCoefficients()` pridaná kontrola, či normály bodov minimálnej množiny, z ktorých sú parametre modelu počítané, spĺňajú podmienku maximálnej povolenej uhlovej deviácie v korektnom smere, ktorá pôvodným triedam modelov úplne chýba. Ak nie je táto podmienka splnená, kandidátny útvar je okamžite zavrhnutý ako nevhodný. Kontrola euklidovskej odchýlky nie je nutná, pretože parametre sú počítané tak, aby body minimálnej množiny ležali na povrchu útvaru definovaného vypočítanými parametrami. Výnimkou je cylinder, kde by mala byť pridaná aj táto kontrola, no z dôvodu presnejšieho testovania efektov normál bodov na rýchlosť a presnosť detekcie nebola, pretože nie je použitá ani v pôvodnej triede z knižnice PCL. Čo sa týka kontroly uhlovej odchýlky v ostatných metódach, v ktorých k nej dochádza, je aj tu pozmenená tak, aby naďalej neboli akceptované body, s odchýlkami normál presne opačnými, t.j. kontrola uhlovej odchýlky je prevádzaná

na celých 360°. Medzi metódy, ktorých sa táto zmena týka patria napr. rovnaké metódy v ktorých bolo nutné rozdeľovať kontroly jednotlivých prahov, ktoré sú spomenuté vyššie v tejto sekcii.

Výnimka kompatibility

Takto upravené modely sú ďalej použité na detekciu v RANSACu. Sú kompatibilné s pôvodným RANSACom implementovaným v knižnici PCL takmer úplne. Jednou výnimkou je metóda prevádzajúca optimalizáciu parametrov, ktorú nebolo možné upraviť tak, aby pracovala presne pôvodným spôsobom. Po rozdelení jednotlivých prahov táto metóda optimalizuje parametre iba na úrovni euklidovskej odchýlky, pretože nová štruktúra, ktorá obsahuje uhlové deviácie nie je navrhnutá v PCL a teda táto metóda v pôvodnej verzii RANSACu takúto štruktúru nepozná. Metóda ale nie je v programe tejto práce použitá, takže problém je čisto teoretický.

4.5 RANSAC algoritmus

Aj pre zvolenú verziu RANSAC algoritmu existuje v knižnici PCL jeho implementácia. Konkrétne sa jedná o triedu `pcl::RandomSampleConsensus<PointT>`, ktorá je implementáciou pôvodnej verzie algoritmu a teda nie je schopná pracovať s viacerými modelmi súčasne. Je teda vytvorená nová trieda `ransac<PointT>`, v ktorej je táto schopnosť implementovaná.

Prvým krokom je vytvorenie dátovej štruktúry, ktorá udržiava informácie o tom, s ktorými modelmi sa pracuje. Táto musí vždy obsahovať minimálne jeden model. Keďže sa pracuje s viacerými modelmi, je tiež potrebné vytvoriť metódy, ktoré umožnia pridávať a odoberať modely a metódu, ktorá kontroluje, či všetky modely pracujú nad rovnakým vstupom. Táto kontrolná metóda je potrebná kôli návrhu štruktúry v knižnici PCL, z ktorej sa implementačne vychádza. Vstup je totiž v PCL zadávaný modelom a toto teda platí aj pre upravené modely. Snaha detekovať naraz rôzne modely nad rôznymi vstupmi je záležitosťou paralelizácie a nie multimodelovej detekcie a preto je potrebné kontrolovať, či všetky zadané modely pracujú nad tým istým vstupom.

Posledným krokom je úprava metódy `computeModel()`, ktorá je zodpovedná za detekciu samotnú. Tá, ako je už v návrhu spomínané, je upravená aby fungovala tak, že pre každý model existujú jemu vlastné štruktúry, ktoré udržiavajú informácie o potrebných aspektoch detekcie, akými sú napr. parametre najlepšieho dovtedajšieho kandidáta tohto modelu, jeho skóre a počet kandidátov, ktorých je potrebné vygenerovať, aby bola detekcia úspešná so želanou pravdepodobnosťou. Medzi jednotlivými modelmi je potom udržiavaná iba informácia, ktorý z nich má globálne najlepšieho kandidáta, čo je implementované jednoducho – pomocou ukazovateľa. Na konci detekcie sú potom výsledné hodnoty najlepšieho útvaru zkopírované do príslušných štruktúr odkiaľ je možné zdedenými metódami tieto informácie získať zvonka RANSACu. Prahy, ako napr. vrchný limit počtu kandidátov, ktorí môžu byť vygenerovaní alebo maximálny počet vygenerovaných nevyhovujúcich kandidátov, ktorý je nastavený na desaťnásobok limitu predošlého, už existujú aj v základnej verzii algoritmu. Implementačne teda nie je na úprave tejto metódy nič obzvlášť zaujímave, čo by stálo za zmienku.

4.6 Vstupy a výstupy

Časti programu prístupné užívateľovi sa nachádzajú vo vstupno-výstupnom bloku. Základom vstupnej časti je ovládanie programu pomocou konzolového rozhrania. Implementované je pomocou knižnice *Boost – Program Options*, ktorá podstatne zjednodušuje spracovanie parametrov tohto rozhrania, ktoré pozostáva z týchto parametrov:

- Všeobecné:

  `-h [ --help ]`
  Výpis pomocníka.

  `-v [ --verbosity ] arg (=3)`
  Nastavenie úrovne vypisovaných kontrolných správ v súlade s knižnicou PCL.

  `-m [ --mode ] arg (=computation)`
  Nastavenie programového módu. Možnosti:
  1.) computation - Plne pracujúca verzia.
  2.) visualization - Iba grafické zobrazenie vstupu.

  `-i [ --input_file ] arg`
  Vstupný point cloud súbor typu PCD.

- Programový mód computation:

  Výpočet normál:

  `-n [ --neighborhood_size ] arg (=100)`
  Veľkosť okolia použitého pri počítaní normál. Platné hodnoty $\langle 0-\text{INT\_MAX}\rangle$.

  `--original_normals arg (=0)`
  Použije normály zo vstupného súboru. Ak nijaké neexistujú, sú vypočítané a parameter je ignorovaný.

  Detekcia:

  `-o [ --output_file ] arg`
  Súbor na uloženie výsledného point cloudu vo formáte PCD.

  `-s [ --shape ] arg`
  Útvary dostupné na detekciu:
  1.) plane
  2.) sphere
  3.) cylinder
  Môžu byť vybrané viaceré útvary.

  `--end_mode arg (=maximum)`
  Výber spôsobu ukončenia algoritmu. Možnosti:
  1.) maximum - Po detekcii určeného počtu útvarov.
  2.) smallest - Po detekcii všetkých útvarov väčších ako stanovená veľkosť.
  Hodnota pre tento parameter je nastavená v parametre `end_value`.

  `-l [ --end_value ] arg`
  Hodnota pre vybranú `end_mode` možnosť.

`--modified_ransac arg (=1)`
: Výber medzi originálnym pcl::RANSAC algoritmom, ktorý pracuje s jedným modelom naraz a jeho modifikovanou verziou, ktorá pracuje s viacerými modelmi súčasne.

`--modified_models arg (=1)`
: Výber medzi originálnymi modelmi pcl::SacModel(s) želaného útvaru a ich modifikovanou verziou.

`--random_seed arg (=0)`
: Nastavenie počiatočnej hodnoty náhodného generátora čísel pre modely na momentálny čas.

`-e [ --euclidean_threshold ] arg`
: Maximálna povolená euklidovská odchýlka od povrchu útvaru.

`-a [ --angle_threshold ] arg`
: Relatívna váha $\langle 0, 0-1, 0\rangle$ uhlovej odchýlky $\langle 0-\pi/2\rangle$ normály bodu od normály modelu pri použití originálnych modelov a maximálna povolená uhlová odchýlka normály bodu od normály modelu v rozsahu $\langle 0, 0-1, 0\rangle$ pre uhly rozsahu $\langle 0-\pi/2\rangle$ pri použití modifikovaných modelov.

`--min_radius arg (=-1.7976931348623157e+308)`
: Minimálny povolený polomer útvaru pre všetky útvary, ktoré majú polomer.

`--max_radius arg (=1.7976931348623157e+308)`
: Maximálny povolený polomer útvaru pre všetky útvary, ktoré majú polomer.

Vizualizácia:

`--color_mode arg (=random)`
: Farebná schéma použitá pre extrahované útvary. Možnosti:
1.) random - Náhodné farby
2.) typized - Farby podľa typu modelu.

Väčšina týchto parametrov nepotrebuje podrobnejšie vysvetlenie, no niektoré súvislosti nie sú z čistého zoznamu parametrov viditeľné a ich popis je predmetom ďalších riadkov.

Kontrolné výpisy programu sú implementované systémom použitým v knižnici PCL z dôvodu zachovania použitej štruktúry. Aj keď by použitie iného typu výpisov nemalo byť problémom, neexistuje nijaký dôvod, prečo by mal byť vytváraný nový systém výpisov a nepoužiť už existujúci. Nastavenie úrovne výpisu kontrolných správ je preto pre tento systém.

Vstupné a výstupné súbory majú uvedený používaný formát PCD. Tento formát je natívnym formátom knižnice PCL a bol vyvinutý špeciálne pre prácu s point cloud dátami [9]. Keďže je implementácia programu založená na tejto knižnici, použitie tohto formátu je logickou voľbou.

Všeobecné parametre sú, ako už je v návrhu uvedené, spoločné pre oba programové módy. Program vo vizualizačnom móde má jediný povinný parameter, ktorým je vstupný súbor. Použitie parametrov, ktoré nie sú určené pre tento mód, vyvolá chybu. Pre výpočtový mód programu existujú dodatočné parametre. Okrem povinnosti zadať vstupný súbor je v tomto móde povinné zadať maximálnu povolenú euklidovskú a uhlovú odchýlku a aspoň jeden útvar na detekciu. Množstvo parametrov dostupných pre tento mód má svoju východiskovú hodnotu, ktorá môže byť v prípade potreby zmenená a niektoré parametre sú

čisto voliteľné (ktoré parametre patria do ktorej kategórie je zrejmé z vyššie dostupného zoznamu).

Grafická prezentácia výsledkov, ktorá spadá do výstupnej časti, je implementovaná pomocou triedy `pcl::PCLVisualizer` modulu *visualization*. V programe sú vytvorené dve funkcie, `simpleVis()` a `doubleVis()`, ktoré nastavujú parametre objektu tejto triedy za účelom modifikácie spôsobu zobrazenia. Prvá z funkcií je využitá na zobrazenie vstupného súboru vo vizualizačnom móde programu, druhá potom slúži na prezentáciu výsledku detekcie spolu s pôvodným vstupom, čo implikuje hlavný rozdiel medzi týmito funkciami. Pretože vo vizualizačnom móde nie je potrebné zobrazovať dva vstupy, nevzniká tu potreba deliť okno zobrazenia na dve časti.

Kapitola 5

Experimenty

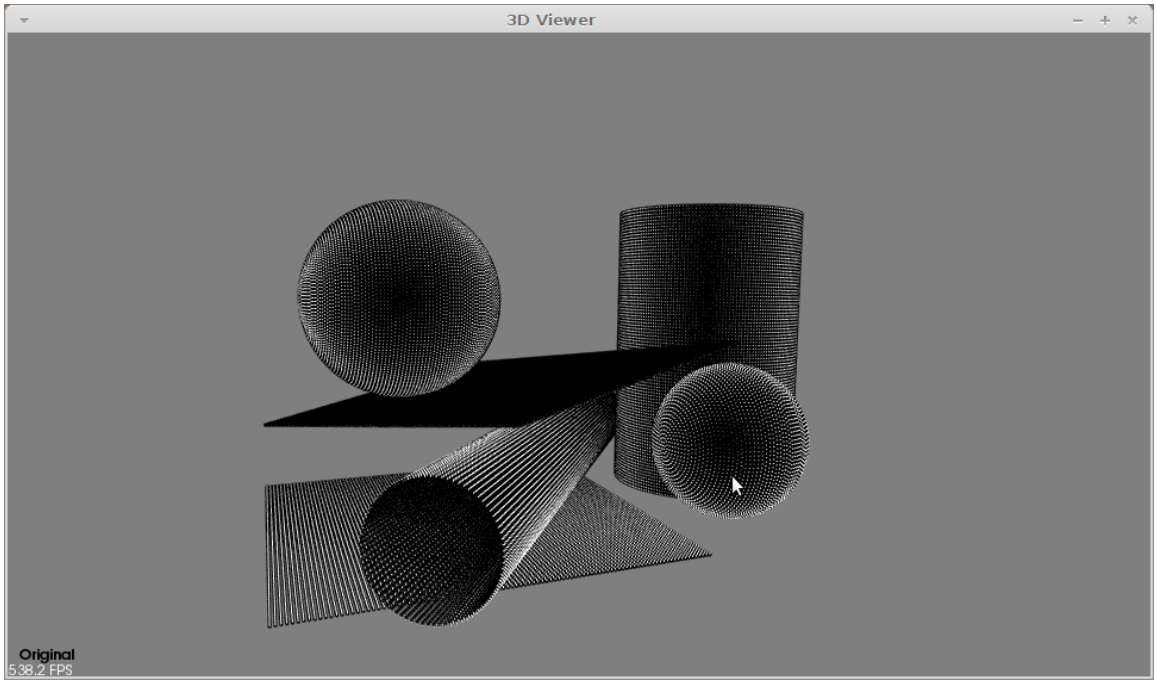
V tejto kapitole sú prezentované testy vykonané na vytvorenom programe a ich výsledky. Sekcia 5.1 obsahuje podrobnosti o testovacích dátach a systéme, na ktorom boli prevedené testy. V sekcii 5.2 sa nachádzajú testy týkajúce sa výpočtu hodnôt a orientácie normál a sekcia 5.3 obsahuje testy rýchlosti a kvality detekcie.

5.1 Popis testovacích dát a systému

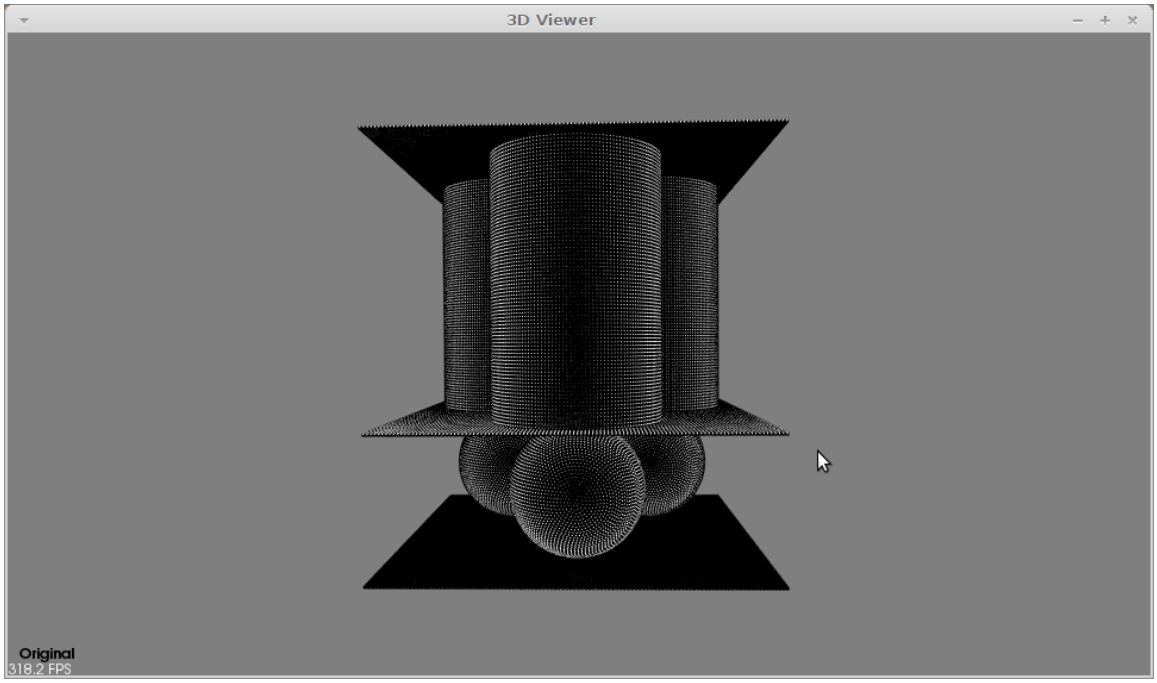
Testovacie dáta pozostávajú zo synteticky vygenerovaných scén zložených z útvarov detekovateľných v tejto aplikácii. Jedná sa o päť unikátnych scén s dvomi rôznymi veľkosťami kroku (v metroch) medzi susediacimi bodmi pre každú scénu a tromi hodnotami gaussovského šumu (so smerodajnou odchýlkou σ) pre každý krok. Celkovo je teda na testy dostupných 30 scén. Tieto scény majú aj ďalšie vlastnosti, ktorými sú počet modelov, z ktorých je scéna zložená, spôsob usporiadania modelov scény a veľkosti najväčšieho, najmenšieho a priemerného útvaru (v počte bodov). Popis všetkých vlastností sa nachádza v tabuľke 5.1. Jednotlivé scény je možné vidieť na obrázkoch 5.1, 5.2, 5.3, 5.4 a 5.5.

	Modely						
				Veľkosť			
Scéna č.	**Počet**	**Prekryv**	**Štruktúra**	**Kroku**	**Min.**	**Max.**	**Priem.**
1	6	Nie	Náhodná	0,5	20163	68237	44529
				1,0	5048	17108	11165
2	9	Nie	Usporiadaná	0,5	20163	50554	39779
				1,0	5048	12717	9991
3	9	Áno	Náhodná	0,5	10207	121204	42999
				1,0	2544	30301	10787
4	12	Nie	Usporiadaná	0,5	4800	40200	11087
				1,0	1250	10100	2821
5	12	Áno	Náhodná	0,5	4680	75551	32025
				1,0	1228	19026	8062

Tabulka 5.1: Vlastnosti scén

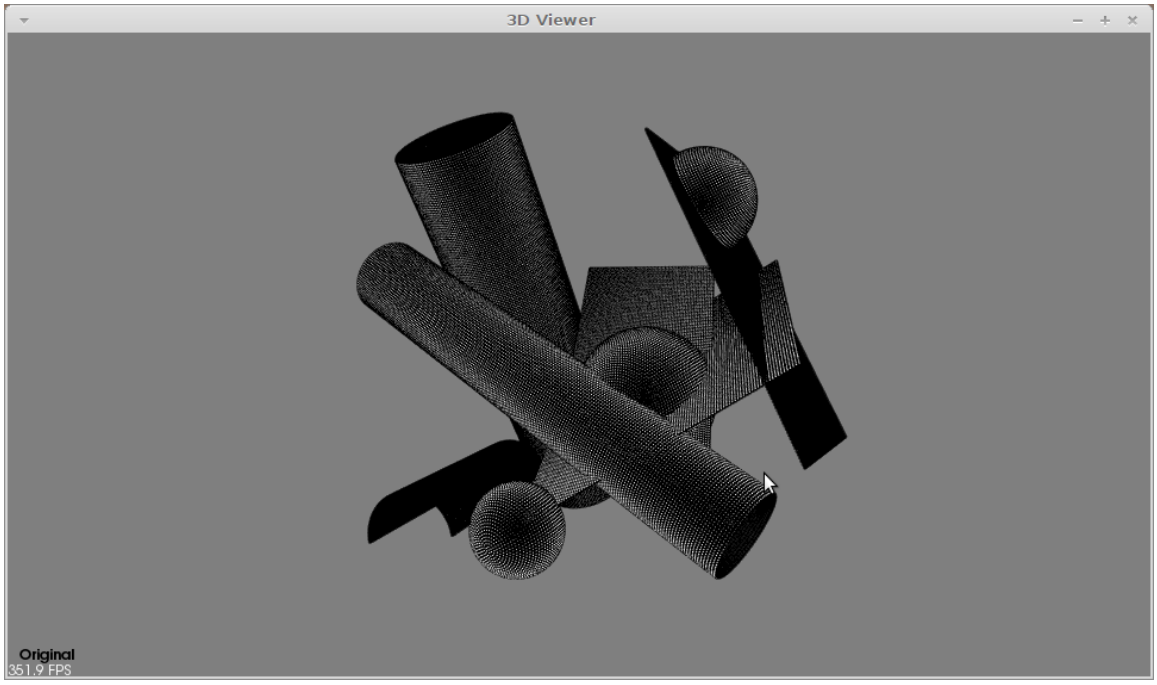


Obrázek 5.1: Scéna 1

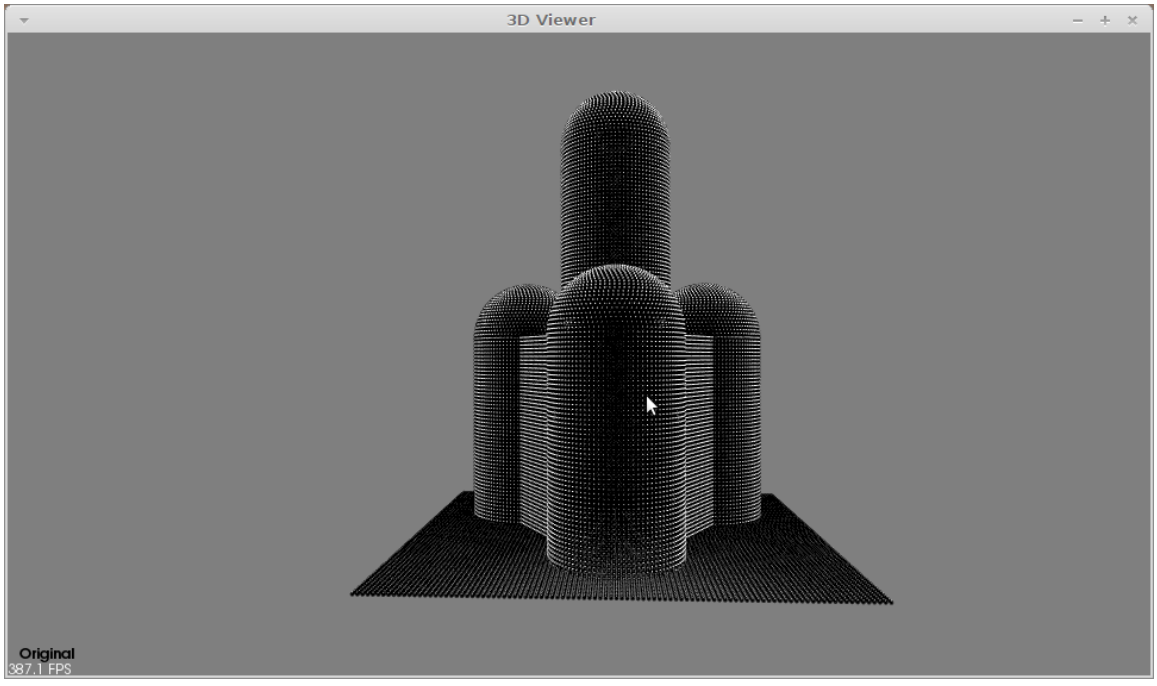


Obrázek 5.2: Scéna 2

Testovacím systémom je notebook Sony VAIO VPC-F11M1E s procesorom Intel Core i5 520M, grafickou kartou NVIDIA GeForce GT 330M, 4GB RAM a operačným systémom Linux Mint 16 "petra"verzia MATE 64-bit.



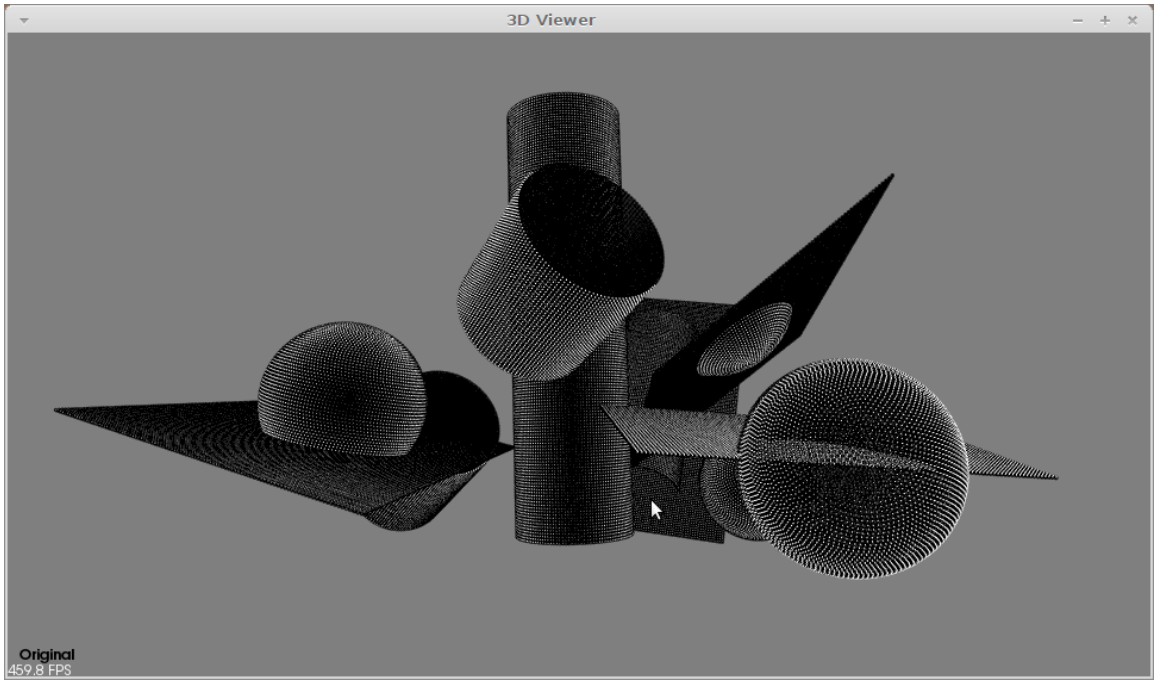
Obrázek 5.3: Scéna 3



Obrázek 5.4: Scéna 4

5.2 Normály

Táto sekcia sa sústredí na testovanie úspešnosti výpočtu hodnoty a propagácie orientácie normál. Normály sú vypočítané pre každú scénu, jej nastavenie kroku medzi susediacimi bodmi a gaussovského šumu σ a päť rôznych veľkostí použitého k-okolia (v počte bodov).



Obrázek 5.5: Scéna 5

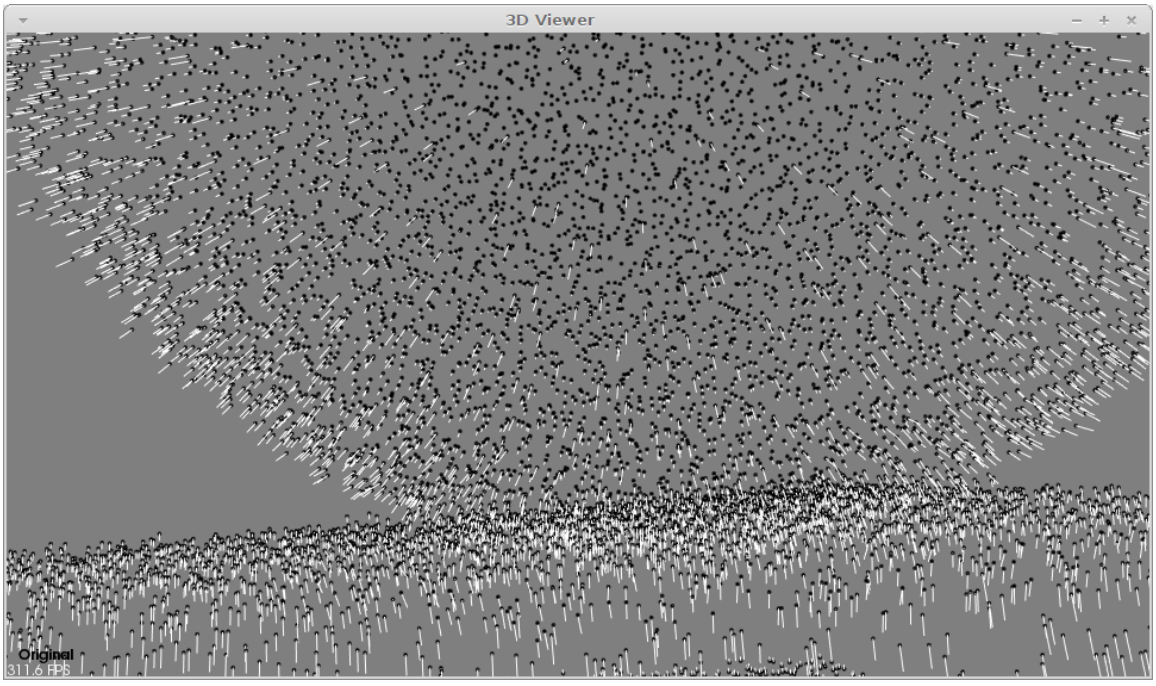
Výsledné hodnoty sú porovnané s referenčnými, pričom normála je považovaná za korektnú ak sa od referenčnej neodchyľuje o viac než 30°. Za inverznú je považovaná vtedy, ak sa od inverzie referenčnej normály neodchyľuje o viac než 30°. Všetky ostatné normály sú považované za nesprávne. Výsledky testov je možné nájsť v tabuľke 5.2. Každá bunka je priemerom hodnôt získaných z jednotlivých scén.

Z testov je jasne vidieť, že pre veľké hodnoty šumu a malé hodnoty k-okolia je už samotný výpočet hodnôt normál veľmi nepresný. Toto je očakávaný výsledok, keďže pri tomto stave sú pomocné dotykové plochy, z ktorých sú normály počítané, získavané z malého počtu veľmi nepresných dát. Pri vyššom počte bodov k-okolia majú tieto nepresnosti vyššiu šancu vzájomne sa vyrušiť ak sú body tohto okolia z toho istého modelu ako bod, ktorého normála je momentálne počítaná. V opačnom prípade môže mať väčšie okolie skôr negatívny efekt. Vo výsledkoch je zachytený pozitívny efekt väčšieho okolia, ktorý je očakávaný s ohľadom na testovací dataset. Pri nižších hodnotách šumu nie je s nepresnosťou dát problém a preto má aj veľkosť zvoleného k-okolia z tohto hľadiska menší dopad na výpočet, čo je rovnako viditeľné vo výsledkoch.

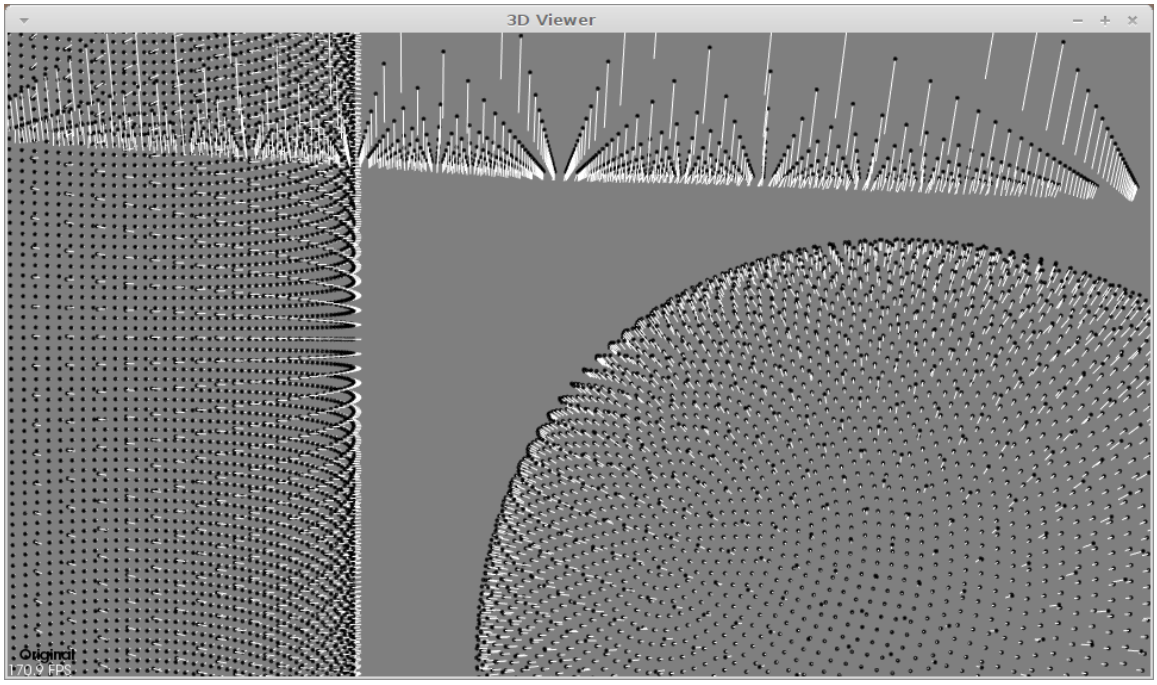
Z pohľadu propagácie orientácie normál je vidieť, že jej úspešnosť je približne $50 - 60\,\%$ pri dobre vypočítaných hodnotách a pomer medzi inverznými a korektnými normálami sa výraznejšie nemení pre hocijakú situáciu. Metóda preto nie je z tohto pohľadu veľmi úspešnou. Tento fakt je prisúdený vzájomnému ovlyvňovaniu sa medzi útvarmi v miestach ich kontaktu počas propagácie orientácie. Na obrázku 5.6 a 5.7 sa dá pozorovať ako k tomuto ovlyvneniu dôjde. Zaujímavým faktom je, že aj keď nie je metóda v porovnaní s referenčnými dátami úspešná, pre jednotlivé útvary sú vo väčšine prípadov normály, hoci možno inverzné, ale z väčšej časti konzistentné vrámci konkrétneho útvaru, čo je tiež zachytené na predošlých obrázkoch.

Korektné Inverzné Chybné		Veľkosť okolia k				
		20	40	60	80	100
Krok	**Šum**					
0,5	1,00	24,95 % 18,17 % 56,88 %	48,68 % 29,50 % 21,82 %	54,56 % 36,91 % 8,53 %	44,21 % 49,50 % 6,29 %	43,71 % 50,50 % 5,79 %
	0,75	33,60 % 23,81 % 42,59 %	55,96 % 35,22 % 8,82 %	60,47 % 34,33 % 5,20 %	51,59 % 43,66 % 4,75 %	51,12 % 44,32 % 4,56 %
	0,50	49,12 % 33,97 % 16,91 %	52,90 % 42,89 % 4,21 %	48,34 % 47,85 % 3,81 %	49,71 % 46,64 % 3,65 %	52,25 % 44,17 % 3,58 %
1,0	0,50	56,84 % 38,53 % 4,63 %	59,20 % 36,87 % 3,93 %	47,87 % 48,32 % 3,81 %	64,04 % 32,15 % 3,81 %	62,40 % 33,73 % 3,87 %
	0,25	57,88 % 38,80 % 3,32 %	53,11 % 43,68 % 3,21 %	60,13 % 36,56 % 3,31 %	63,69 % 32,87 % 3,44 %	62,66 % 33,77 % 3,57 %
	0,00	58,23 % 38,61 % 3,16 %	56,45 % 40,58 % 2,97 %	48,09 % 48,91 % 3,00 %	46,53 % 50,04 % 3,43 %	56,40 % 40,04 % 3,56 %

Tabulka 5.2: Úspešnosť výpočtu hodnôt a orientácie normál pre dané parametre



Obrázek 5.6: Kontaktný bod gule a plochy – Scéna 1 - Krok 1,0 - Šum 0,50



Obrázek 5.7: Miesto kontaktu plochy, valca a gule – Scéna 1 - Krok 1,0 - Šum 0,00

5.3 Detekcia

V tejto sekcii sa nachádza rozbor testov rýchlosti a kvality detekcie medzi jednotlivými kombináciami použitých pracovných blokov. Detekcia je vykonaná pre každú kombináciu scény, nastavenia veľkosti kroku medzi susediacimi bodmi, gaussovského šumu σ a kombináciu rôznych pracovných blokov. Veľkosť použitého k-okolia, pre kombinácie využívajúce vypočítané normály je 100 bodov. Detekcia prebieha pre všetky podporované útvary a modely majú nastavenú počiatočnú hodnotu náhodného generátora čísel na momentálny čas. Veľkosť maximálnej povolenej euklidovskej odchýlky je totožná s hodnotou veľkosti σ šumu konkrétnej scény a maximálna povolená uhlová odchýlka je 30°. Pri použití pôvodných modelov sú tieto prahy prepočítané tak, aby bolo možné rátať ako inlier bod spĺňajúci obe podmienky súčasne a váha každej z odchýlok je 50 %. Taktiež je nastavený maximálny povolený polomer modelu na veľkosť $50\,m$. Algoritmus končí po nájdení určeného počtu modelov. Ten je daný počtom modelov, z ktorých je daná scéna zložená.

Doba výpočtu detekcie je súčtom systémového a užívateľského času zo štandardného unixovského nástroja time. Model je považovaný za korektný ak aspoň 80 % jeho bodov patrí jednému modelu scény totožného typu útvaru alebo ak je v ňom obsiahnutých aspoň 80 % bodov modelu scény totožného typu útvaru. Ak detekovaný model dobre aproximuje viacero modelov scény súčasne, je počítaný medzi korektné modely príslušný počet krát. Detekovaný unikátny model je taký, z ktorého je zložená testovacia scéna a zároveň preň existuje detekovaný model. Výsledky testov sa nachádzajú v tabuľke 5.3. V každej bunke sa nachádzajú priemerné hodnoty zo všetkých scén daného nastavenia.

Z výsledkov testov sa dá vyvodiť niekoľko záverov.

Porovnaním originálnej sekvenčnej verzie RANSACu a jeho multimodelovej modifikácie je vo veľkej väčšine prípadov vidieť zlepšenie v rýchlosti dosiahnutého výsledku pri zachovaní kvality detekcie. Toto je značne výraznejšie pri použití originálnych modelov než pri

Čas behu(s) **Korektné** **Unikátne**		**Normály (O – Originálna / M – Modifikovaná)** **Modely (O – Originálne / M – Modifikované)** **RANSAC (O – Originálny / M – Modifikovaný)**							
Krok	**Šum**	O O O	O O M	O M O	O M M	M O O	M O M	M M O	M M M
0,5	1,00	534,8 8,4 7,8	165,7 8,2 7,4	115,7 8,8 6,8	99,1 8,6 6,8	861,5 9,0 7,8	338,7 8,4 7,2	313,5 7,8 4,2	329,6 7,8 4,4
	0,75	494,5 9,2 8,4	141,2 8,0 7,2	119,5 8,8 6,6	99,6 8,8 7,0	871,0 8,2 7,2	256,9 8,2 6,6	293,2 8,4 4,8	267,8 8,2 4,6
	0,50	580,2 9,4 8,6	171,3 9,0 7,6	112,9 8,4 6,8	106,8 9,2 7,0	752,2 9,0 7,6	256,3 9,0 7,6	409,4 8,2 4,6	392,9 8,4 5,0
1,0	0,50	142,4 9,2 8,4	39,8 9,4 8,2	31,2 9,0 6,8	26,4 9,2 7,2	160,3 9,4 8,4	52,3 9,0 7,8	75,9 8,2 5,0	71,9 8,2 5,2
	0,25	135,3 9,6 8,6	43,5 9,2 8,4	31,0 9,4 7,6	29,8 9,4 7,8	149,7 9,4 8,6	52,3 9,4 8,0	56,7 9,2 5,8	76,1 8,8 5,8
	0,00	189,6 9,4 9,4	24,7 9,4 9,4	53,5 9,6 9,6	9,9 9,6 9,6	747,7 9,6 7,8	518,6 9,6 7,6	254,4 9,4 5,2	203,3 9,4 5,2

Tabulka 5.3: Výsledky rýchlosti a kvality detekcie v daných situáciách

ich modifikovanej verzii. Takéto správanie je očakávané, keďže originálne modely musia vyhodnotiť skóre pre väčší počet kandidátov, nakoľko skoro všetci kandidáti sú korektnými vďaka nižším restrikciám. Z pohľadu RANSACu sa teda jedná o jasnú optimalizáciu.

Z pohľadu modelov je situácia trochu zložitejšia. Na strane výpočtu normál je vidieť nižšiu kvalitu výsledkov a z toho aj fakt, že ich kvalita má väčší dopad na modifikované modely, ktoré kladú na informáciu o normále väčší dôraz.

Pri použití originálnych normál je vidieť jasné zlepšenie v oblasti rýchlosti detekcie pri minimálnom až nulovom znížení kvality vo všetkých prípadoch. Tento výsledok ukazuje jasný pozitívny vplyv použitia normál bodov na rýchlosť a kvalitu detekcie útvarov z point cloudov.

Pri použití vypočítaných normál s propagáciou orientácie sú očividné problémy s ich kvalitou. Výraznejší pokles oproti originálnym modelom je, v tejto situácii, vidieť v počte nájdených unikátnych modelov, čo je možné overiť kontrolou dosiahnutého pomeru unikátnych a korektných modelov k priemeru v tabuľke 5.4. Problém je takisto poznať aj v čase výpočtu, ktorý je v niekoľkých prípadoch dokonca vyšší ako pri originálnych modeloch. Celkový počet korektných detekovaných modelov ale nezaznamenal voči originálnym modelom až tak výrazný pokles. Skombinovaním týchto zistení so zisteniami z predošlej sekcie vzniká zaujímavý záver: keďže modifikované modely akceptujú iba kandidátne útvary s korektnou orientáciou normál, pri výpočte normál implementovaným spôsobom, kde je orientácia normál vrámci modelu konzistentná, no môže byť inverzná, sa takéto modely

stávajú nedetekovateľnými. To má za následok výrazne nižšie hodnoty unikátnych detekovaných útvarov a takisto vyšší čas výpočtu. Čas výpočtu je negatívne ovplyvnený dvomi spôsobmi. Prvým je potreba vygenerovať v priemere vyšší počet kandidátov na nájdenie takého, ktorý je akceptovateľný. Druhým je fakt, že útvary, ktoré by s korektnou orientáciou normál mali oveľa vyššie skóre, ostávajú nedetekovateľné vo vstupnom point cloude a zvyšujú počet potrebných korektných kandidátov na dosiahnutie úspešnej detekcie so želanou pravdepodobnosťou.

Riešením tohto problému, ktorý by zachoval prospešné vlastnosti, by mohlo byť zvoľnenie podmienok prijatia kandidátneho útvaru u modifikovaných modelov tak, že kandidátne útvary s inverznými normálami by boli akceptovateľné. Novou podmienkou validity by bola iba konzistentná orientácia normál vrámci kandidáta bez ohľadu na smer. Spolu s modifikovaným výpočtom normál, ktorý vo väčšine prípadov konzistenciu normál vrámci jednotlivých modelov dosahuje, sú modely s čisto inverznými normálami znovu detekovateľné, čím sa odstránia negatívne efekty, ktoré sprevádzajú použitie vypočítaných normál.

Unikátne / Korektné		**Scéna č.**				
Krok	**Šum**	1	2	3	4	5
0,5	1,00	0,8444	0,8551	0,7222	0,6833	0,8090
	0,75	0,8333	0,8310	0,7222	0,6724	0,8000
	0,50	0,8125	0,9014	0,6667	0,7083	0,8000
1,0	0,50	0,8333	0,9583	0,6806	0,7200	0,8022
	0,25	0,8750	0,9306	0,6944	0,7976	0,8021
	0,00	0,8750	0,8333	0,8143	0,8936	0,7917

Tabulka 5.4: Pomer unikátnych detekovaných útvarov ku korektne detekovaným útvarom

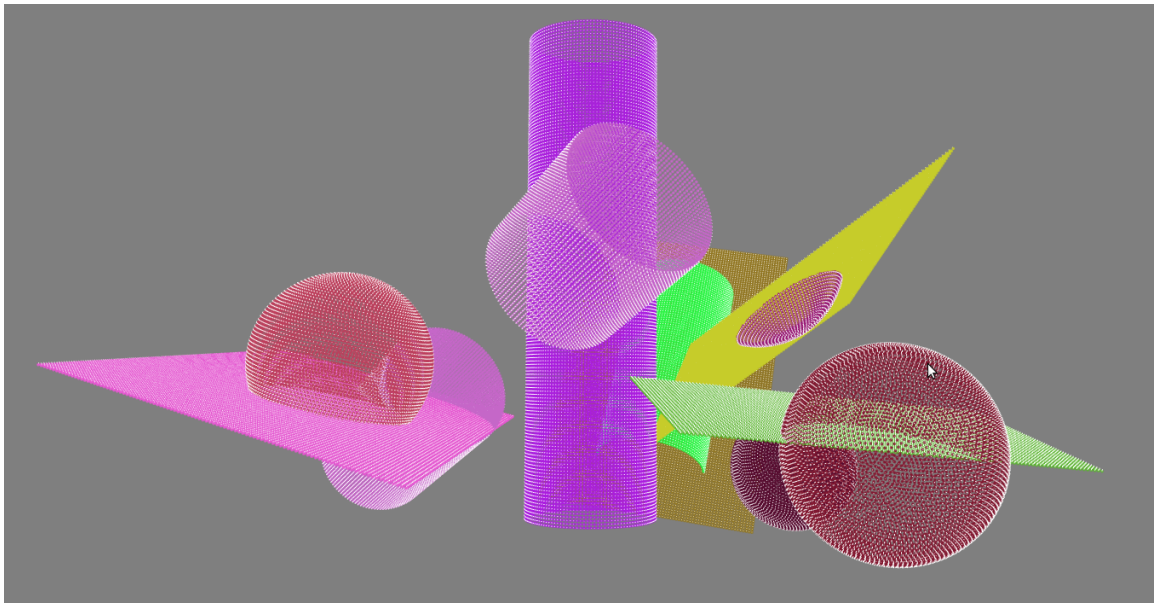
Pre vizuálne porovnanie dosiahnutých výsledkov je na ďalších stránkach uvedených niekoľko príkladov výsledkov detekcie. Nastavenia jednotlivých scén sú nasledujúce:

Obrázok 5.8
Scéna č.5
Originálne normály — Modifikované modely — Modifikovaný RANSAC
Veľkosť kroku = 1,0 — Veľkosť šumu = 0,0

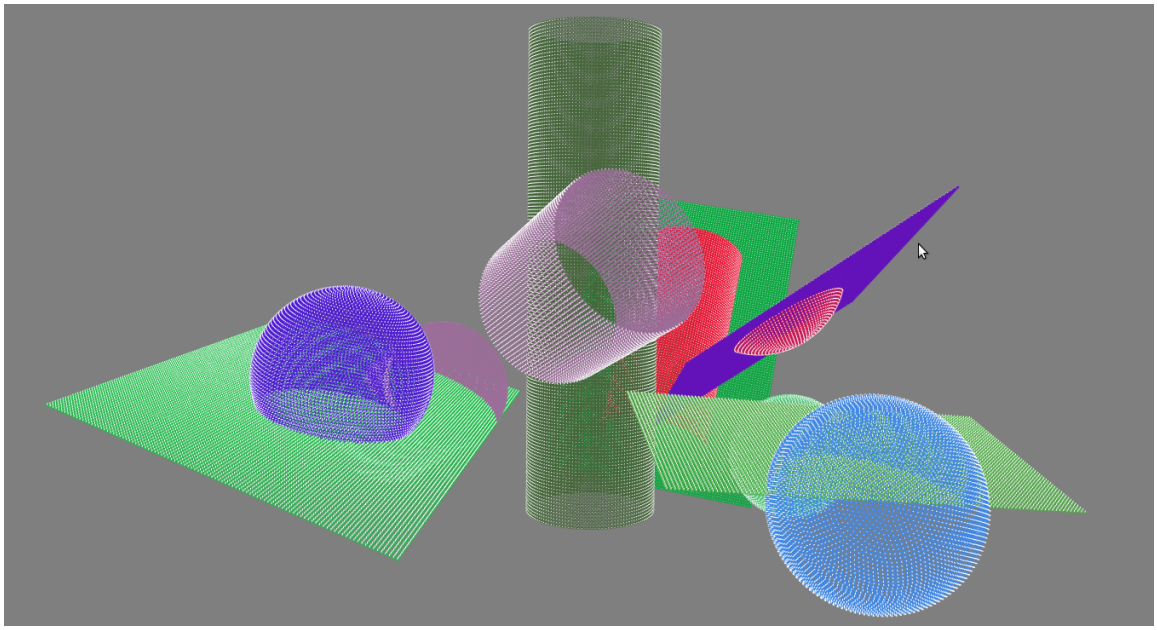
Obrázok 5.9
Scéna č.5
Originálne normály — Originálne modely — Originálny RANSAC
Veľkosť kroku = 1,0 — Veľkosť šumu = 0,0

Obrázok 5.10
Scéna č.5
Originálne normály — Modifikované modely — Modifikovaný RANSAC
Veľkosť kroku = 0,5 — Veľkosť šumu = 0,5

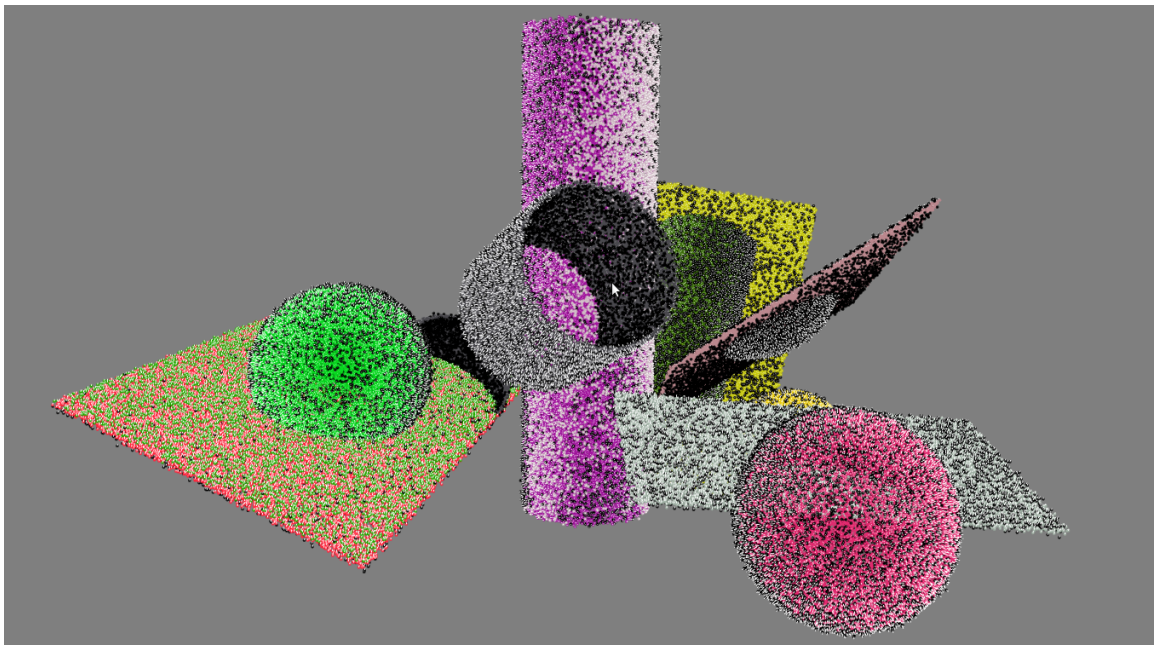
Obrázok 5.11
Scéna č.5
Originálne normály — Originálne modely — Originálny RANSAC
Veľkosť kroku = 0,5 — Veľkosť šumu = 0,5



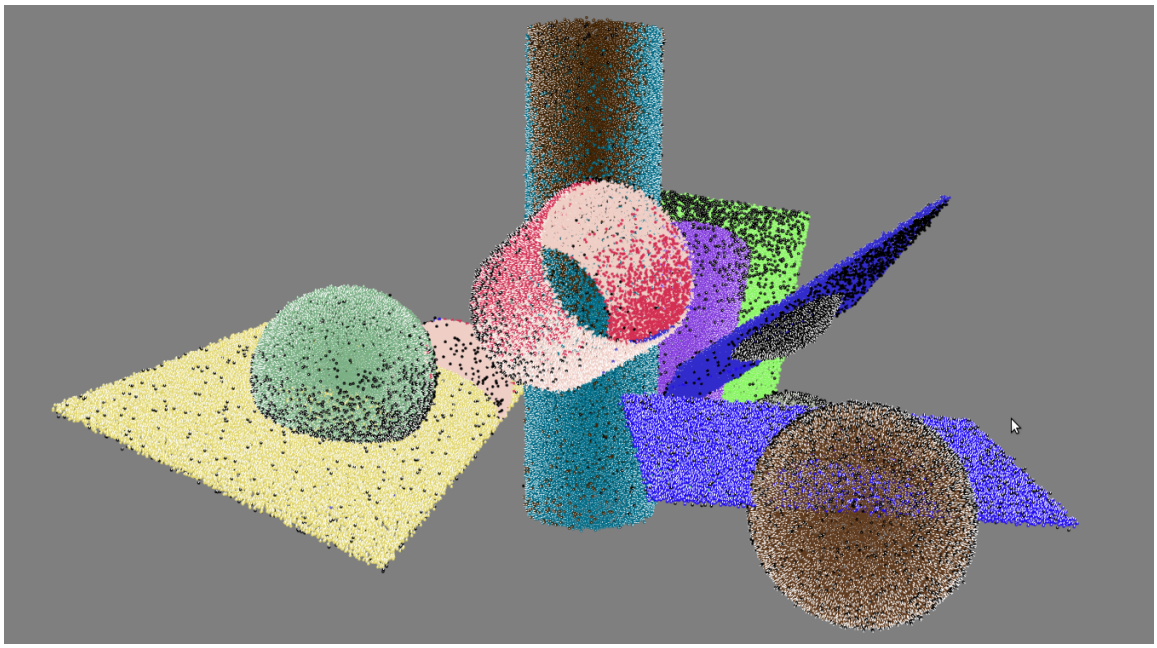
Obrázek 5.8: Výsledok detekcie – Scéna 5 - O/M/M - 1,0/0,0 - Doba detekcie: 11,13 s



Obrázek 5.9: Výsledok detekcie – Scéna 5 - O/O/O - 1,0/0,0 - Doba detekcie: 74,01 s



Obrázek 5.10: Výsledok detekcie – Scéna 5 - O/M/M - 0,5/0,5 - Doba detekcie: 207,29 s



Obrázek 5.11: Výsledok detekcie – Scéna 5 - O/O/O - 0,5/0,5 - Doba detekcie: 622,29 s

Kapitola 6

Záver

Cieľom tejto práce bolo naštudovať problematiku detekcie jednoduchej geometrie z point cloudov a navrhnúť, implementovať a otestovať program, ktorý by bol schopný takúto detekciu prevádzať. Tento cieľ sa podarilo splniť.

Detektor je založený na pôvodnej verzii RANSAC algoritmu, ktorý bol v tejto práci upravený tak, aby dokázal pracovať s viacerými modelmi súčasne. Ďalšia úprava sa týkala modelov použitých na detekciu pomocou RANSACu. Tým bola ako hlavná modifikácia pridaná striktnejšia kontrola kandidátnych útvarov na základe orientácie normál bodov. Keďže normály bodov nie sú vždy dodané spolu so vstupom, hlavne pri globálnych point cloud dátach, v tejto práci bola takisto prezentovaná metóda na ich výpočet a získanie ich korektnej orientácie. Testovaním výslednej aplikácie bol potom demonštrovaný prínos jednotlivých modifikácii pre rýchlosť a kvalitu výsledkov celkovej detekcie.

Existuje množstvo možných smerov ďalšieho vývoja v tejto oblasti. Prezentované modifikácie by sa napríklad mohli skombinovať s už existujúcimi optimalizačnými technikami, za účelom zistenia ich praktického použitia v aplikáciach pracujúcich s reálnymi dátami. Odlišným smerom vývoja by mohlo byť kombinovanie jednoduchých útvarov do zložitejších pomocou ich vzájomnej pozície za účelom detekcie zložitejších útvarov.

Literatura

[1] Fischler, M. A.; Bolles, R. C.: Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*, ročník 24, č. 6, Červen 1981: s. 381–395, ISSN 0001-0782, doi:10.1145/358669.358692, http://doi.acm.org/10.1145/358669.358692.

[2] Hoppe, H.; DeRose, T.; Duchamp, T.; aj.: Surface Reconstruction from Unorganized Points. *SIGGRAPH Comput. Graph.*, ročník 26, č. 2, Červenec 1992: s. 71–78, ISSN 0097-8930, doi:10.1145/142920.134011, http://doi.acm.org/10.1145/142920.134011.

[3] Huang, J.; You, S.: Detecting Objects in Scene Point Cloud: A Combinational Approach. In *3D Vision, 2013 International Conference on*, Washington, DC, USA: IEEE Computer Society, Červen 2013, s. 175–182, doi:10.1109/3DV.2013.31, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=6599074.

[4] Khoshelham, K.; Elberink, S. O.: Accuracy and Resolution of Kinect Depth Data for Indoor Mapping Applications. *Sensors*, ročník 12, č. 2, 2012: s. 1437–1454, ISSN 1424-8220, doi:10.3390/s120201437, http://www.mdpi.com/1424-8220/12/2/1437.

[5] Matas, J.; Chum, O.: Randomized RANSAC with $T_{d,d}$ test. *Image and Vision Computing*, ročník 22, č. 10, 2004: s. 837–842, ISSN 0262-8856, doi:10.1016/j.imavis.2004.02.009, British Machine Vision Computing 2002, http://www.sciencedirect.com/science/article/pii/S0262885604000514.

[6] Msm: RANSAC — Wikipedia, The Free Encyclopedia [online]. [cit. 2014-5-19], http://en.wikipedia.org/w/index.php?title=RANSAC&oldid=603279731.

[7] Noorkhanuk85: Kinect 2.0 for the Xbox One — Wikipedia, The Free Encyclopedia [online]. 2013-12-1 [cit. 2014-5-19], http://en.wikipedia.org/w/index.php?title=Kinect&oldid=606268059.

[8] Rabbani, T.; van den Heuvel, F.; Vosselman, G.: Segmentation of point clouds using smoothness constraint. In *Proceedings of the ISPRS Commission V Symposium 'Image Engineering and Vision Metrology'*, Září 2006, http://www.isprs.org/proceedings/XXXVI/part5/paper/RABB_639.pdf.

[9] Radu B. Rusu: The PCD (Point Cloud Data) file format [online]. [cit. 2014-5-18], http://pointclouds.org/documentation/tutorials/pcd_file_format.php.

[10] Schmitt, S. R.: Center and Radius of a Sphere from Four Points [online]. 2005 [cit. 2014-5-15], http://www.abecedarical.com/zenosamples/zs_sphere4pts.html.

[11] Schnabel, R.; Wahl, R.; Klein, R.: Efficient RANSAC for Point-Cloud Shape Detection. *Computer Graphics Forum*, ročník 26, č. 2, Červen 2007: s. 214–226, http://onlinelibrary.wiley.com/doi/10.1111/j.1467-8659.2007.01016.x/abstract.

[12] Sunday, D.: Distance between 3D Lines & Segments [online]. 2012 [cit. 2014-5-14], http://geomalgorithms.com/a07-_distance.html.

[13] Torr, P. H. S.; Zisserman, A.: MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Comput. Vis. Image Underst.*, ročník 78, č. 1, Duben 2000: s. 138–156, ISSN 1077-3142, doi:10.1006/cviu.1999.0832, http://dx.doi.org/10.1006/cviu.1999.0832.

[14] Velizhev, A.; Shapovalov, R.; Schindler, K.: Implicit shape models for object detection in 3D point clouds. In *ISPRS Congress*, Červenec 2012, s. 179–184, http://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/I-3/179/2012/isprsannals-I-3-179-2012.pdf.

[15] Weber, C.; Hahmann, S.; Hagen, H.: Sharp Feature Detection in Point Clouds. In *Proceedings of the 2010 Shape Modeling International Conference*, SMI '10, Washington, DC, USA: IEEE Computer Society, 2010, ISBN 978-0-7695-4072-6, s. 175–186, doi:10.1109/SMI.2010.32, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=5521460.

[16] Weisstein, E. W.: Hessian Normal Form — MathWorld, A Wolfram Web Resource [online]. http://mathworld.wolfram.com/HessianNormalForm.html, 2014-5-13 [cit. 2014-5-14].

[17] Woodford, O.; Pham, M.-T.; Maki, A.; aj.: Demisting the Hough Transform for 3D Shape Recognition and Registration. In *Proceedings of the British Machine Vision Conference*, BMVA Press, 2011, ISBN 1-901725-43-X, s. 32.1–32.11, doi:10.5244/C.25.32, http://www.bmva.org/bmvc/2011/proceedings/paper32/.

[18] WWW stránky: Visualization Toolkit (VTK) [online]. 2011-9-14 [cit. 2014-5-16], http://www.vtk.org/.

[19] WWW stránky: OpenNI [online]. 2012-5-7 [cit. 2014-5-16], https://github.com/OpenNI/OpenNI.

[20] WWW stránky: FLANN – Fast Library for Approximate Nearest Neighbors [online]. 2013-1-15 [cit. 2014-5-16], http://www.cs.ubc.ca/research/flann/.

[21] WWW stránky: Point Cloud Library (PCL) [online]. 2013-10-7 [cit. 2014-5-16], http://pointclouds.org/.

[22] WWW stránky: Boost C++ Libraries [online]. 2013-2-4 [cit. 2014-5-16], http://www.boost.org/.

[23] WWW stránky: Eigen [online]. 2013-7-24 [cit. 2014-5-16], http://eigen.tuxfamily.org/.

[24] WWW stránky: Point cloud — Wikipedia, The Free Encyclopedia [online]. 2014-2-18 [cit. 2014-5-11], http://en.wikipedia.org/w/index.php?title=Point_cloud&oldid=596051400.

[25] WWW stránky: Principal component analysis — Wikipedia, The Free Encyclopedia [online]. 2014-5-13 [cit. 2014-5-14], http://en.wikipedia.org/w/index.php?title=Principal_component_analysis&oldid=608451980.

[26] WWW stránky: OpenNI — Wikipedia, The Free Encyclopedia [online]. 2014-5-16 [cit. 2014-5-16], http://en.wikipedia.org/w/index.php?title=OpenNI&oldid=608777903.

[27] WWW stránky: Point Cloud Library (PCL) — Module sample_consensus [online]. [cit. 2014-5-19], http://docs.pointclouds.org/trunk/a02954.html.

[28] Zuliani, M.: RANSAC for Dummies. With examples using the RANSAC toolbox for Matlab™& Octave and more. . . , Srpen 2012, http://vision.ece.ucsb.edu/~zuliani/Research/RANSAC/docs/RANSAC4Dummies.pdf.