VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV AUTOMATIZACE A INFORMATIKY
INSTITUTE OF AUTOMATION AND COMPUTER SCIENCE

IDENTIFIKACE 3D OBJEKTŮ PRO ROBOTICKÉ APLIKACE

3D OBJECT IDENTIFICATION FOR ROBOTIC GRASPING

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. JAROSLAV HUJŇÁK**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. RADOMIL MATOUŠEK, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav automatizace a informatiky
Student:	**Bc. Jaroslav Hujňák**
Studijní program:	Strojní inženýrství
Studijní obor:	Aplikovaná informatika a řízení
Vedoucí práce:	**doc. Ing. Radomil Matoušek, Ph.D.**
Akademický rok:	2019/20

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Identifikace 3D objektů pro robotické aplikace

Stručná charakteristika problematiky úkolu:

Jde o řešení velmi důležité a netriviální úlohy identifikace 3D objektů pro robotické aplikace, jako je bin–picking. V kontextu zadání budou realizovány nejméně dva přístupy pro identifikaci 3D objektu typu sféra. Jedna z metod bude využívat aparát CGA (Clifford's Geometric Algebra). Matematické postupy vyhledání shody modelu dílu s nasnímanými daty jsou stále předmětem výzkumu. Úloha bude řešena na reálném hardware.

Cíle diplomové práce:

1. Rešerše přístupů robotického 3D vidění pro aplikaci bin picking.
2. Popis užité technologie 3D snímání (Phoxi 3D, Photoneo).
3. Identifikace sféry pomocí dvou metod, z nichž jedna bude CGA.
4. Programová implementace a její popis vhodný k dalšímu užití.
5. Vyhodnocení výsledků (přesnost, opakovatelnost, rozsah vstupních dat, rychlost aj.) a kvalitní prezentace.

Seznam doporučené literatury:

PAJDLA, Tomáš. Global optimization in camera and robot calibration: Globální optimalizace v kalibraci kamer a robotů. V Praze: České vysoké učení technické, [2017]. ISBN 978-80-01-06114-5.

KOLÍBAL, Zdeněk. Roboty a robotizované výrobní technologie. Brno: Vysoké učení technické v Brně - nakladatelství VUTIUM, 2016. ISBN 978-80-214-4828-5.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2019/20

V Brně, dne

L. S.

doc. Ing. Radomil Matoušek, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

Abstrakt

Tato diplomová práce se zabývá popisem přístupu robotického 3D vidění pro aplikaci bin picking. Práce se zaměřuje na identifikaci sfér v pointcloudech nasnímaných 3D skenerem a otestování nové metody založené na konformní geometrické algebře (CGA). Rychlost, přesnost a škálovatelnost této metody je porovnána s tradiční metodou založenou na deskriptorech. Testováním bylo prokázáno, že CGA dosahuje pro testované pointcloudy obdobné přesnosti jako metoda založená na deskriptorech, ale za významně kratší čas. Přístup pomocí metody CGA se jeví slibně pro budoucí použití v robotickém 3D vidění pro identifikaci a lokalizaci sfér.

Abstract

This thesis focuses on robotic 3D vision for application in Bin Picking. The new method based on Conformal Geometric Algebra (CGA) is proposed and tested for identification of spheres in Pointclouds created with 3D scanner. The speed, precision and scalability of this method is compared to traditional descriptors based method. It is proved that CGA maintains the same precision as the traditional method in much shorter time. The CGA based approach seems promising for the use in the future of robotic 3D vision for identification and localization of spheres.

Klíčová slova

bin picking, konformní geometrická algebra, RANSAC, detekce sfér, deskriptor, pointcloud, robotické vidění, inlier, podvzorkování, přiřazení šablon, metoda nejmenších čtverců

Keywords

Bin Picking, Conformal Geometric Algebra, RANSAC, sphere detection, descriptor, pointcloud, robotic vision, Inlier, downsampling, template alignment, Least Square Method

Bibliografická citace

HUJŇÁK, Jaroslav. *Identifikace 3D objektů pro robotické aplikace* [online]. Brno, 2020 [cit. 2020-06-26]. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/121521. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství. Vedoucí práce doc. Ing. Radomil Matoušek, Ph.D.

Poděkování

Děkuji svému vedoucímu, doc. Ing. Radomilu Matouškovi, Ph.D, za možnost pracovat na zajímavé diplomové práci zkoumající nové přístupy identifikace objektu. Dále bych chtěl poděkovat ústavu za poskytnuté prostory a potřebné vybavení a panu Ing. Romanu Parákovi za cenné rady. Za podporu při studiu a trpělivost při psaní diplomové práce bych chtěl poděkovat své rodině, zejména svému bratrovi Ing. Ondřeji Hujňákovi.

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením doc. Ing. Radomila Matouška, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....

Jaroslav Hujňák

V Brně dne 26.6.2020

Obsah

1 Úvod **15**

2 Robotické vidění a bin picking **17**
2.1 Robotické vidění 17
2.2 Bin picking 18

3 Snímání povrchu **19**
3.1 Typy 3D skenerů 19
3.1.1 3D skenery na principu strukturovaného světla 20
3.2 PhoXi 3D Scanner 21

4 Identifikace pomocí deskriptorů **23**
4.1 Pointcloud 23
4.2 Počáteční úprava dat 24
4.2.1 Odstraňování zbytečných bodů 24
4.2.2 Podvzorkování 24
4.3 K-D tree 24
4.4 Normály v pointcloudu 26
4.5 Výběr vhodného měřítka 28
4.6 Deskriptory 29
4.6.1 Point Feature Histograms (PFH) 29
4.6.2 Fast Point Feature Histograms (FPFH) 31
4.7 Párování deskriptorů a shlukování korespondencí 32

5 Identifikace pomocí geometrické algebry CGA **35**
5.1 Základy geometrických algeber 35
5.1.1 Základní algebraické prvky geometrických algeber 35
5.1.2 Součiny v geometrických algebrách 36
5.1.3 Involuce a dualita 37
5.2 CGA - Conformal Geometry algebra 38
5.2.1 Reprezentace objektů 38
5.2.2 Vzdálenost v CGA 40
5.3 Clifford library 40
5.4 Aplikace CGA k identifikaci sfér 40
5.4.1 Metoda nejmenších čtverců 41
5.4.2 RANSAC 42

6 Praktická část **45**

6.1 Aplikace Global registration 45
6.1.1 Potřebné předchystání aplikace Global registration 47
6.1.2 CloudCompare 47
6.1.3 Vytvoření šablon 47
6.1.4 Implementace Global registration 49
6.2 Aplikace CGA 53
6.2.1 Vytvoření objektů v CGA 53
6.2.2 RANSAC 54

7 Výsledky práce **61**
7.1 Zhodnocení aplikací 67

8 Závěr **69**

Literatura **71**

Kapitola 1

Úvod

Robotické stroje hrají ve výrobě stále důležitější roli. Zvyšování produkce, nedostatek pracovních sil a snaha eliminovat lidské chyby si žádá automatizaci řady výrobních procesů. Snahou výrobců je automatizovat zejména rutinní nebo pro člověka namáhavé činnosti. Ne všechny činnosti, které jsou snadným úkolem pro člověka, lze snadno automatizovat. Právě jednou z takovýchto činností - bin pickingem - se zabývá tato diplomová práce.

Bin picking je automatizační úloha, která provádí přesun předmětu mezi zásobníkem a výrobním strojem. Přesouvání, srovnávání a třídění objektů je součástí našeho každodenního života, aniž bychom si to pořádně uvědomovali. Při těchto úkonech člověk automaticky zvolí vhodnou sílu díky nabytým zkušenostem a rozhodovacím schopnostem.

Přestože jsou roboti vhodnými pomocníky při aplikacích, které se neustále opakují, bin picking pro ně představuje náročný problém. Je při něm dbán důraz na přesnost v proměnném prostředí. Robot musí lokalizovat hledaný objekt, který má libovolnou orientaci, v prostředí, které se při každém cyklu mění. To vyžaduje nalezení rovnováhy mezi robotickou obratností, strojovým viděním, softwarem, výpočetním výkonem, který zvládne pracovat s daty v reálném čase, a řešením pro uchopení a extrahování dílu.

Cílem této diplomové práce je popsat dva přístupy robotického vidění pro aplikaci bin picking, konkrétně přístup pomocí deskriptorů a pomocí CGA. Cílem je také vytvořit aplikaci pro identifikaci sfér, porovnat zvolené přístupy a zhodnotit výhody jejich využití. Vytvořená aplikace musí být schopna rozeznat několik objektů na základě dat z kamery s hloubkovým vjemem. Aplikace by měla daný objekt lokalizovat, rozeznat ho a určit jeho polohu a orientaci.

Pro softwarové řešení identifikace sfér je využit program CloudCompare a knihovny Clifford a Open3d. Program CloudCompare je použit na úpravu snímků z kamery, potřebnou pro jejich další zpracování. Knihovna Open3d obsahuje metody, algoritmy a funkce sloužící k řešení daného problému. Práce s geometrickými algebrami, v této práci konkrétně konformní geometrické algebry (CGA), je umožněna díky knihovně Clifford. Knihovna obsahuje základní funkce geometrických algeber jako například geometrický či vnější součin. Pro práci s reálnými daty byla zvolena kamera PhoXi 3D M od firmy Photoneo.

Proces popsaný pomocí deskriptorů se skládá ze zpracování snímku kamery ve formátu velké množiny bodů snímaného povrchu (pointcloud), výpočtu deskriptorů popisujících geometrii menších oblastí bodů a jejich přiřazení k předem připraveným šablonám.

Identifikace sféry pomocí CGA využívá metodu RANSAC. Z pointcloudu jsou náhodně vybrány 4 body potřebné k popisu objektu, objekt je následně v algebře vytvořen a poté je vypočítán počet inlierů a outlierů. Při dostatečném počtu inlierů, tedy bodů příslušných

k objektu, je vytvořený objekt uložen do kandidátů. Nejvhodnější kandidát je následně označen jako nalezený objekt.

Teoretická část diplomové práce se zabývá popisem prostředků využitých k realizaci identifikace, metodami a algoritmy využitými při práci s pointcloudy. Dále je zde podrobně popsán zdrojový kód aplikací vytvořených pro identifikaci sfér. V závěru práce je zhodnoceny obě použité metody, tj. metoda CGA a metoda založená na deskriptorech.

Kapitola 2

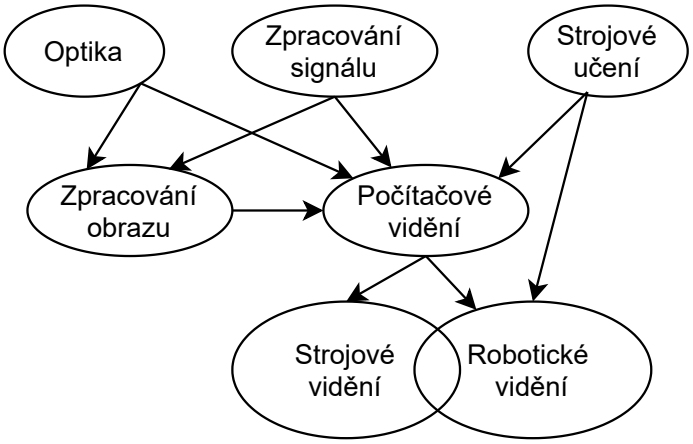
Robotické vidění a bin picking

2.1 Robotické vidění

Robotické vidění je sadou algoritmů, které umožňují robotickým komponentám zaznamenávat a analyzovat obrazové informace. Jedná se o kombinaci počítačových algoritmů, kamer, senzorů a dalších hardwarových komponent, díky kterým je robot či stroj schopný zpracovat vizuální data okolního prostředí. To umožňuje robotovi plnit náročné úkoly, ke kterým je potřeba vizuální porozumění. Systémy obsahující 2D kameru dokáží detekovat objekt na rovném podkladu, uchopit jej a přesunout na jiné místo. Pro složitější případy je nutné využití některé metody 3D skenování, například stereofonní kamery.

Robotické vidění bývá často spojováno se strojovým viděním. Oba termíny si jsou velmi blízké, ale existují určité rozdíly, ve kterých se liší. Strojové vidění nachází uplatnění zejména v průmyslovém odvětví. Mezi významné aplikace strojového vidění patří automatická kontrola, navádění robota a řízení procesů. I přesto, že se využívají některé metody strojového vidění pro navádění robota, mluvíme přesněji při navádění robota o robotickém vidění. Robotické vidění je na rozdíl od strojového vidění primárně určeno pro implementaci v robotickém světě.

Dalšími pojmy se kterými bývá robotické vidění často spojeno jsou počítačové vidění, strojové učení, zpracování signálů a procesů.



Obrázek 2.1: Strom technických pojmů [19]

2.2 Bin picking

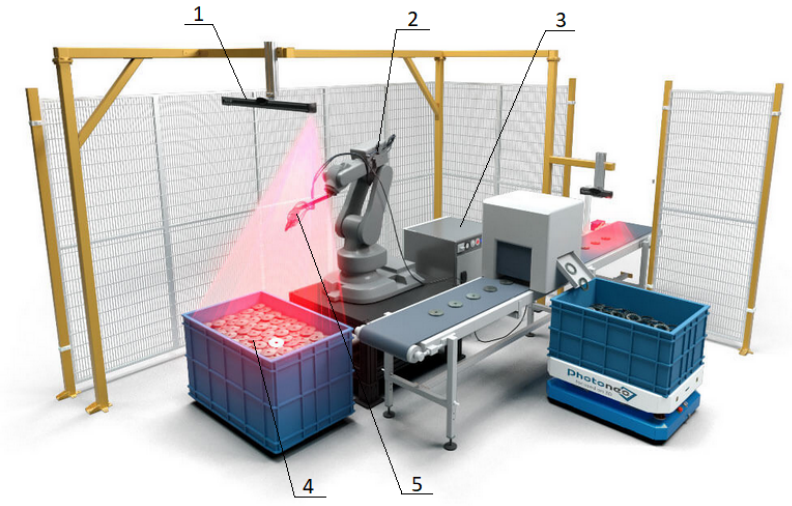
Bin picking je technologie, při které robot spolupracuje s 3D průmyslovou kamerou. Rameno robota vybírá nahodile navršené díly z bedny a podle předem stanoveného algoritmu je nyní již organizovaně předává k dalšímu zpracování. Předměty mohou být variabilní, tedy mít různé velikosti, povrchy i tvary. Při předávání dílu k dalšímu zpracování je třeba dát důraz na to, aby při přesunu nedošlo k porušení okolních předmětů. Bin picking umožňuje robotovi vykonávat operace přemisťování bez nutnosti zdlouhavého programování.

Technologie bin picking se využívá především pro monotónní úkoly, které jsou pro člověka nudné a často i nebezpečné. Přemisťované předměty mohou být těžké, ostré či mít příliš vysokou teplotu pro manipulaci. Konkrétně se jedná například o zásobování stroje materiálem, třídění několika různých předmětů či přípravu materiálu pro montáž finální sestavy. Bin picking technologie se aplikuje v mnoha odvětvích potravinářského, chemického nebo výrobního průmyslu [6].

Pro funkčnost technologie bin picking je nutné mít více správně zvolených a naprogramovaných zařízení, která mezi sebou komunikují.

Jedná se o:

- robota,
- kamerový systém (popř. další senzory) s řídicí jednotkou (většinou na bázi PC),
- efektor,
- periferní zařízení,
- systém osvětlení,
- bezpečnostní prvky,
- komunikační rozhraní.



1. kamera
2. robot
3. komunikační zařízení
4. bedna s předměty
5. efektor

Obrázek 2.2: Schéma bin pickingu [1]

Kapitola 3

Snímání povrchu

Pro identifikaci objektu ve scéně je potřeba nejprve získat 3D informace o této scéně. Právě tomu se věnují 3D skenery. 3D skenování je proces analýzy objektů reálného světa nebo prostředí pro sběr dat o jeho tvaru i vzhledu (např. barvě). Získaná data pak slouží k vytvoření digitálních 3D modelů. Ve většině případů je princip 3D skenování založený na snímání velkého počtu jednotlivých bodů na povrchu objektu, tj. pointcloudu. Poté využitím těchto bodů zrekonstruuje prostorový počítačový model použitím vhodné polygonové sítě. K získání těchto bodů se využívá mnoho různých technologií: kamery, rentgeny, magnetické mikrotomografy, lasery, dotykové snímače. Od použitých technologií se pak i nazývají jednotlivé metody skenování tzn. například rentgenové, ultrazvukové, laserové, optické nebo mechanické 3D skenery. Nejčastěji se používají nedestruktivních metody, které snímanou součástku nezničí. Každá metoda má však svoje omezení, výhody a nevýhody a liší se i cenou potřebného vybavení. Výstupy, tj. 3D počítačové modely, mají mnohé využití od filmových efektů a počítačových her přes průmyslový design, ortotiku a protetiku, reverzní inženýrství až po kontrolu kvality nebo dokumentaci kulturních památek [9].

3.1 Typy 3D skenerů

3D skenery lze kategorizovat na základě různých parametrů, tj. např. podle skenovací technologie, nasazení či rozměrů. V obrázku 3.1 jsem zvolil rozdělení podle metody skenovací technologie, které jsem seskupil do dvou hlavních tříd v závislosti na tom, jestli metoda vyžaduje přímý kontakt se skenovaným objektem. Na pravé straně grafu je pak uvedeno, zda je technologie schopna skutečného 3D snímání, nebo je hloubka detekována pouze v jedné ose (tzv. 2,5D snímání).

Protože byl v této práci použit bezkontaktní optický skener fungující na principu strukturovaného světla, je na obrázku zvýrazněn a dále v práci je mu věnována zvýšená pozornost.

Měření doby letu světla (Time-of-Flight, ToF) počítá vzdálenosti snímaných bodů z doby letu světla mezi emitorem, daným bodem na objektu a odrazem do detektoru.[15] Emitor může vysílat úzký paprsek, kde dojde k změření hloubkových dat v jednom čase pouze v jednom bodě (LIDAR) nebo provést měření více bodů najednou (Kinect v2 [18])

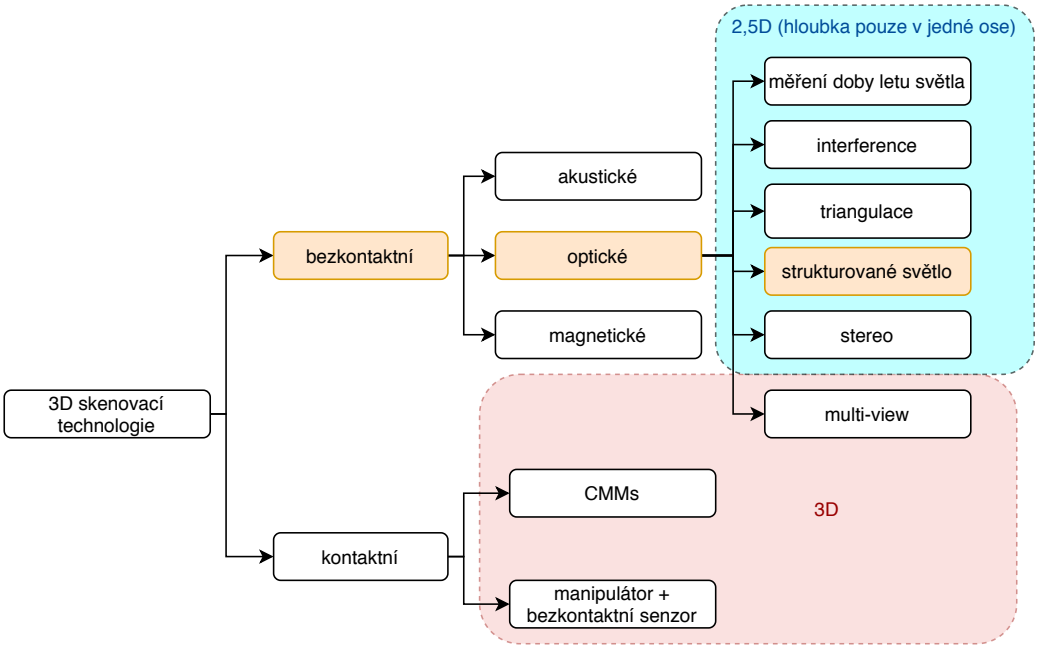
Interference využívá dvou vlnění - referenčního a měřícího. Jejich vzájemnou interakci zaznamenává senzor skrze polarizační dělič a z naměřených hodnot vypočítá vzdálenost dvou bodů. Díky kalibraci lze vytvořit i hloubkovou mapu s absolutními hodnotami.

Triangulace využívá úhlu odrazu k dopočtení vzdálenosti objektu. Na objekt je vyslán laserový paprsek nebo strukturovaná síť, která se od objektu odráží do senzorické matice. Podle přijatého signálu lze ze vzdálenosti od emitoru a úhlu vyslaného paprsku dopočítat vzdálenost bodu (v případě sítě množiny bodů). Vzhledem k malé snímací ploše a nutnosti odrazu paprsku na senzorickou matici je pro vytvoření modelu potřeba využít více snímků v multi-view kombinaci.

Strukturované světlo využívá speciálního vzoru promítaného na skenovaný objekt. Z deformace vzoru nasnímaného 2D senzorem lze rekonstruovat hloubková data. Tento princip je detailněji rozebrán v kapitole 3.1.1.

Stereo skener používá dva snímače umístěné v definované vzdálenosti a používá shodný princip jako lidské oči. Složením perspektivních obrazů z obou snímačů lze pomocí úhlové paralaxy dopočítat vzdálenost od skeneru. [9]
Pro úspěšné měření je ale nutná vhodná textura skenovaného objektu, proto může být skener doplněn aktivním projektorem umělé textury. [15]

Multi-view skenování je založeno na principu snímání objektu z vícero úhlů a následného složení jednotlivých dat do výsledného pointcloudu. Díky kombinaci pohledů z více úhlů získáváme hloubková data ve více osách a lze tedy vytvořit plný 3D model i při použití 2,5D skenerů.



Obrázek 3.1: Rozdělení 3D skenerů

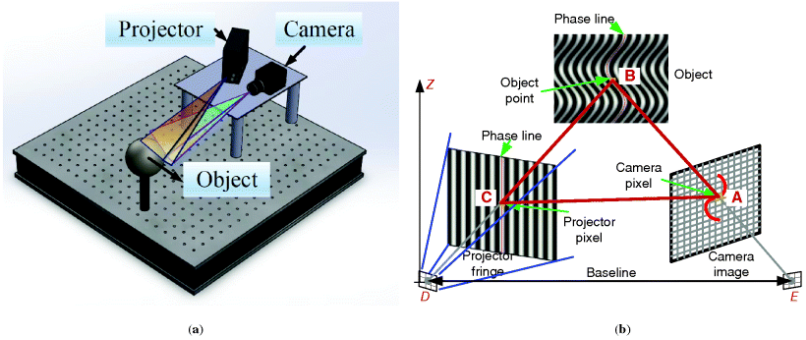
3.1.1 3D skenery na principu strukturovaného světla

3D skenery snímají najednou celou scénu, což umožňuje zpracovávat i pohybující se předměty. Jsou tvořeny dvojicí projektor a kamera, znázorněné na obrázku 3.2, kde projektor

promítá strukturovaný vzor na skenovaný objekt a kamera zachytí 2D obraz objektu s deformovaným vzorem. Interní algoritmus skeneru dopočítá z deformace zaznamenaného vzoru hloubková data obrazu.

Vzhledem k dostupnosti projektorů i kamerových snímačů a poměrně vysoké přesnosti této metody jsou skenery na principu strukturovaného světla často využívány v mnoha odvětvích jako je strojírenství, kontrola kvality či zábavní průmysl. Další výhodou těchto skenerů je dosažení vysokého rozlišení. Toho lze docílit buď použitím jemnějšího vzoru a odpovídajícího senzoru nebo kombinací více snímků s různými vzory [15].

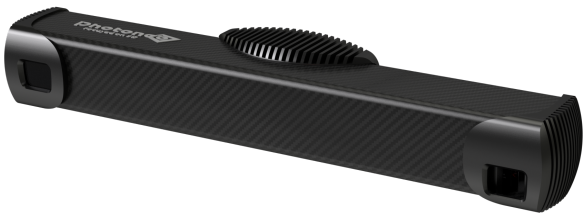
Použitý vzor hraje klíčovou roli má na výkon skeneru vyšší vliv než použitý kamerový snímač. Tento vzor může být generován pseudonáhodně, častěji se však používají komplexní precizně zkonstruované vzory. Tyto mohou obsahovat: binární (černobílé) vzory, vzory v odstínech šedi, sinusoidy a jejich různé kombinace. Vzory jsou většinou promítány bílým světlem, lze však využít i promítání barevných mřížek, teček, čtverečků a kombinace s komplikovanými vzory [12].



Obrázek 3.2: Schéma 3D skenerů na principu strukturovaného světla [8]

3.2 PhoXi 3D Scanner

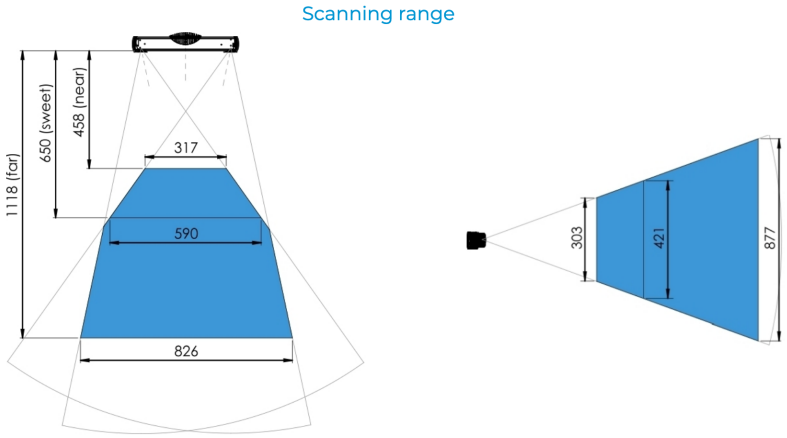
Pro zachycení 3D scény byl využit fakultní 3D skener PhoXi 3D Scanner M od slovenské firmy Photoneo. Tento průmyslový vysoce výkonný skener je vyvíjen pro strojové vidění se zaměřením právě na bin picking a obsahuje integrovaný GPU čip umožňující rychlé zpracování 3D obrazu přímo ve skeneru [16].



Obrázek 3.3: 3D skener PhoXi M [4]

Tento skener je založen na principu strukturovaného světla (viz kapitolu 3.1.1) a pro promítání vzoru používá laserový paprsek vysílající červené viditelné světlo o vlnové délce 637 nm průměrnou silou 4,32 mW. Použití laserového paprsku je firmou označováno jako "paralelní strukturované světlo"a zpracování senzorem vyvinutým touto společností umožňuje rozpoznávání 3D tvarů v rámci každého nasnímaného rámce, což je důležité pro pohybující se scénu [5, 15].

Použitý skener PhoXi M obsahuje senzor s vysokým rozlišením 3.2 megapixelu a každý pixel je skenerem převeden na 3D bod v pointcloudu (pro porovnání Kinect v2 IR senzor má rozlišení pouze 512 × 424 px a snímá tedy poincloudy s 217 088 body [18]). Vzhledem ke snímané ploše 590 × 421 mm při optimální vzdálenosti 650 mm je rozlišení výsledného pointcloudu zhruba 0, 3 mm (viz obrázek 3.4).



Obrázek 3.4: Snímací plocha skeneru PhoXi M [4]

Skener je připojen k počítači pomocí ethernetového rozhraní a pro jeho ovládání je vyžadována ovládací aplikace PhoXi Control. Tato aplikace umožňuje naskenovat scénu a zobrazit její 3D model se zvýrazněním textury, hloubky nebo normál. Aplikace pracuje primárně s Photoneo RAW formátem, ale umožňuje naskenovanou scénu vyexportovat i ve formátech Stanford's PLY, Leica's PTX, tiff obrázcích nebo do txt souboru. Z naskenované scény je schopna ukládat pointcloudy 3D bodů, mapu normálových vektorů, mapu vzdáleností od senzoru a texturu v odstínech šedi. Skener nemá RGB senzor a není tedy schopen zachytit barevnou texturu snímané scény.

Firma Photoneo dodává hotové řešení i pro aplikaci bin picking umožňující rychlé nasazení v praxi a použití s velkým počtem robotů. Vzhledem k tomu, že se tato práce zabývá novým přístupem ke strojovému vidění pro tuto aplikaci, nebyla tato možnost využita.

Kapitola 4

Identifikace pomocí deskriptorů

Nejběžnějším přístupem k rozeznání objektů v pointcloudu je metoda založená na deskriptorech. V této práci je využita knihovna Open3d, která obsahuje algoritmy potřebné k identifikaci pomocí deskriptorů. Knihovna PCL - Point Cloud Library, která je obvykle využívána pro práci s pointcloudy, je z důvodu její neaktuálnosti a použití programovacího jazyka Python pro tvorbu aplikací v této diplomové práci nevhodná.

K popsání metod použitých k identifikaci objektů pomocí deskriptorů je potřeba nejprve se seznámit se základními pojmy, jako jsou např. pointcloud, podvzorkování.

4.1 Pointcloud

Množinu bodů, které se nachází v souřadnicovém systému, označujeme jako mračno bodů neboli pointcloud. V prostorovém souřadnicovém systému jsou body definované souřadnicemi x, y, z reprezentující vnější povrch objektu. K těmto souřadnicím se často přidává informace o barvě a intenzitě. Pointcloud můžeme vytvořit pomocí 3D skenerů, senzorů či ho vymodelovat výpočetní technikou.

V současnosti existuje několik různých formátů, které umožňují reprezentaci pointcloudů. Mezi nejznámější patří PLY, STL a OBJ [23]:

- **OBJ** je formát, který byl poprvé vyvinut technologií Wavefront a byl přijat širokou škálou 3D grafických aplikací. Jedná se o jednoduchý datový formát, který představuje pouze 3D geometrii, normály, barvu a texturu.
- **PLY** je formát známý jako polygonový formát souboru nebo Stanfordův trojúhelníkový formát. Byl vytvořen na základě OBJ a vytvořen pro ukládání 3D dat. Jedná se o formát souboru schopný reprezentovat barvy, průhlednost, povrchové normály, texturu, souřadnice a hodnoty spolehlivosti. Existují dvě verze tohoto souboru, jedna v ASCII a druhá binární.
- **STL** je formát popisující pouze geometrii povrchu trojrozměrného objektu bez jakékoliv reprezentace barvy, textury nebo jiných atributů modelu. Soubory STL jsou obvykle vytvářeny programem pro návrh (CAD) jako koncový produkt procesu 3D modelování. Jedná se o nejčastěji používaný formát souborů pro 3D tisk.

Všechny výše uvedené formáty byly vyvinuty v době, kdy ještě snímací zařízení nebyla na takové úrovni jako dnes, a proto byl vyvinut formát **PCD**.

Ten umožňuje:

- uchovávat a upravovat organizované pointcloudy,
- uchovávat hodnoty v různých datových typech,
- ukládat N-dimenzionální histogramy pro deskriptory.

4.2 Počáteční úprava dat

Kamera PhoXi 3D M, použitá pro získání pointcloudů, snímá více než 3,2 miliony 3D bodů a zpracování takového množství bodů klade vysoký nárok na početní výkon. Provedení jednoduché operace pro každý bod v pointcloudu by odpovídalo časové složitosti $O(n)$, kde n je počet bodů. Porovnává-li se každý bod s k sousedními body v jeho oblasti sousedství, časová složitost narůstá na $O(kn)$. Výpočty potřebné k získání informací by tedy byly zbytečně zdlouhavé a v reálné praxi by nenašly uplatnění. Proto se pro urychlení výpočtů odstraní body, které jsou šumem či příliš vzdálené od ostatních a následně se získají klíčové body. Klíčovými body rozumíme body dostatečně charakteristické k reprezentaci celé oblasti původních bodů. Jednou z možností, jak tyto body získat, je podvzorkování [13].

4.2.1 Odstraňování zbytečných bodů

Pro odstranění zbytečných bodů jsou vhodné k využití dvě funkce knihovny Open3d. Jedná se o funkce `statistical_oulier_removal` a `radius_oulier_removal`, díky kterým se zbavíme bodů, které jsou nadbytečné [24].

`Statistical_outlier_removal` je funkce, která odstraňuje body vzdálenější od sousedů v porovnáním s průměrem pointcloudu. Má 2 vstupní parametry: `nb_neighbors` a `std_ratio`, tedy počet sousedů pro výpočet průměrné vzdálenosti a práh na základě standardní odchylky průměrných vzdáleností pointcloudu.

`Radius_outlier_removal` je funkce, která odstraňuje body s nedostatečným počtem sousedů ve sféře vytvořené okolo nich. Opět má dva vstupní parametry: `nb_points` a `radius`, tedy počet bodů, které vytvořená sféra obsahuje a její poloměr.

4.2.2 Podvzorkování

Při podvzorkování se původní body překryjí mřížkou nových bodů s pravidelným vzorkováním, které by měly respektovat u snímaného objektu jeho původní tvar. Nejprve se pointcloud rozdělí do mnoha oblastí tvaru krychle (voxelů) s požadovanou délkou hrany. V každé krychli se vypočítá centroid z obsažených bodů. Tento centroid představuje nový bod. Jedná se o obdobu snížení rozlišení obrázku.

4.3 K-D tree

K-D strom (v angličtině K-D tree) je datová struktura pro ukládání bodů ve vícedimenzionálním prostoru, kde K označuje kardinalitu daného prostoru a D reprezentuje dimenzi prostoru. Můžeme tedy specifikovat pro konkrétní prostor odpovídající strom (3-D strom pro 3-D prostor). Jedná se o druh binárního stromu, tedy každý uzel, který není list, má maximálně dva potomky. Každá úroveň stromu rozděluje prostor podle dané dimenze. Souřadnice jsou rozděleny podle velikosti - menší se umístí do levého substromu, větší do pravého.

Za bod dělení se volí medián souřadnic v příslušném podintervalu vzniklém rozdělením předchozího intervalu. K-D stromy představují užitečný nástroj pro hledání nejbližších sousedů a hledání sousedů s určitou vzdáleností [10].

Jako příklad vezměme 3-dimenzionální prostor. V kořenovém uzlu rozdělíme potomky na základě první dimenze (tedy na základě souřadnice X). Na druhé úrovni na základě druhé dimenze (souřadnice Y), na třetí podle třetí dimenze (souřadnice Z). Na čtvrté úrovni by bylo rozdělení provedeno podle první dimenze a celý proces by se opakoval. Příklad vytvoření K-D stromu je uveden dále.

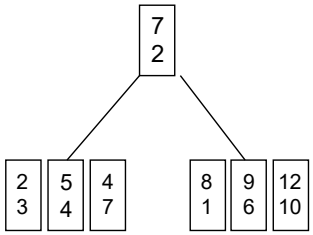
Příklad 4.1. Vytvoření K-D stromu
Mějme následující 2D body:

$$\begin{bmatrix} 5 \\ 4 \end{bmatrix} \begin{bmatrix} 2 \\ 3 \end{bmatrix} \begin{bmatrix} 12 \\ 10 \end{bmatrix} \begin{bmatrix} 8 \\ 1 \end{bmatrix} \begin{bmatrix} 9 \\ 6 \end{bmatrix} \begin{bmatrix} 7 \\ 2 \end{bmatrix} \begin{bmatrix} 4 \\ 7 \end{bmatrix}$$

Srovnáme body podle první dimenze a ostatní dimenze zanedbáme:

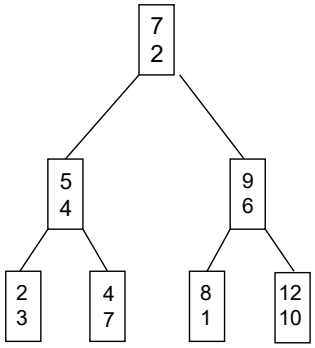
$$\begin{bmatrix} 2 \\ 3 \end{bmatrix} \begin{bmatrix} 4 \\ 7 \end{bmatrix} \begin{bmatrix} 5 \\ 4 \end{bmatrix} \begin{bmatrix} 7 \\ 2 \end{bmatrix} \begin{bmatrix} 8 \\ 1 \end{bmatrix} \begin{bmatrix} 9 \\ 6 \end{bmatrix} \begin{bmatrix} 12 \\ 10 \end{bmatrix}$$

Vybereme medián $\begin{bmatrix} 7 \\ 2 \end{bmatrix}$, rozdělíme interval na podintervaly a prvky v podintervalech seřadíme podle druhé dimenze.



Obrázek 4.1: První rozdělení 2-D stromu

Obdobným způsobem pokračujeme dále, dokud nerozdělíme všechny prvky.



Obrázek 4.2: Finální rozdělení 2-D stromu

4.4 Normály v pointcloudu

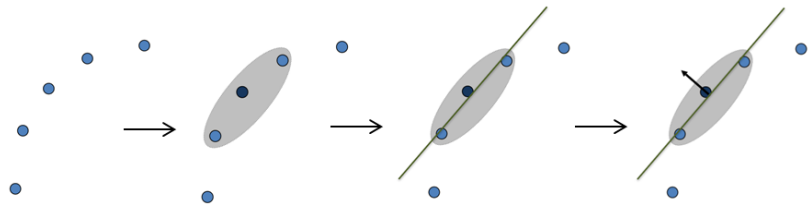
Pro porovnávání dvou různých modelů (hledaný objekt s pointcloudem) je nejprve potřeba získat popis povrchu. Jedna z nejjednodušších a nejpoužívanějších metod k takovému popisu je odhad normál v jednotlivých bodech.

Normála daného $n-1$ dimenzionálního podprostoru v n-dimenzionálním prostoru je přímka kolmá na daný podprostor. Vektor určující směr normály se nazývá normálový vektor. Když známe povrch objektu, lehce dokážeme určit směr normály v daném bodě. V případě pointcloudů ale povrch není přímo známý, jelikož je tvořen množstvím bodů [2].

Existují dvě možnosti jak určit normály pointcloudu:

- získat povrch z pointcloudu a příslušné normály dopočítat z nově získaného povrchu,
- odvodit normály pomocí aproximačních metod.

Určení normál se nejčastěji provádí pomocí aproximačních metod, jelikož rekonstrukce plochy vyžaduje čas, který se snažíme minimalizovat. Jedna z aproximačních metod je proložit plochu body a u té poté určit normálu.



Obrázek 4.3: Určení normály v pointcloudu

K nalezení plochy tvořené body je použita metoda nejmenších čtverců následovně:

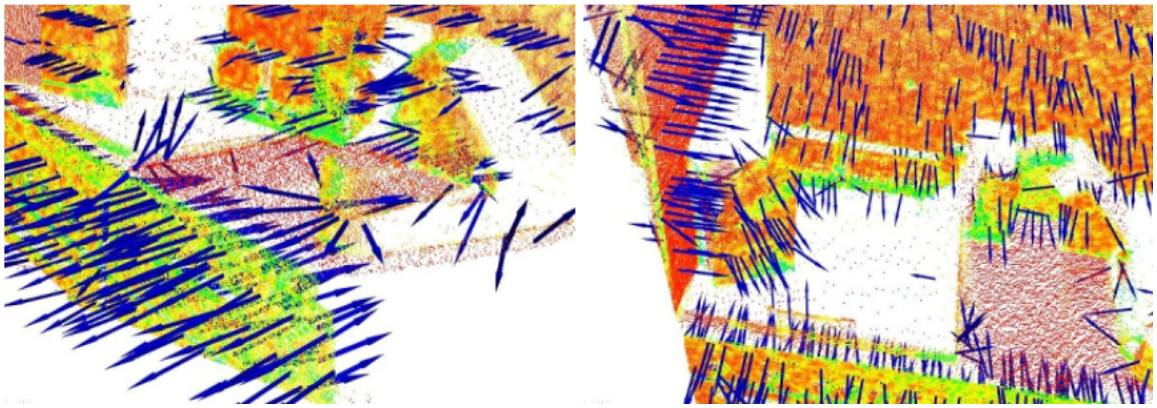
1. Pro každý bod $\mathbf{x}$ z pointcloudu vyber k nejbližších členů, či všechny body ve sféře s poloměrem r:
$$\{\mathbf{x}_i | \, \|\mathbf{x}_i - \mathbf{x}\| < r\}$$
2. Najdi plochu Π, která minimalizuje součet kvadrátů vzdáleností:
$$\min \sum_{i=1}^{n} \text{dist}\,(\mathbf{x}_i, \Pi)^2$$

Řešení nalezení normály povrchu je tedy převedeno na analýzu vlastních vektorů a vlastních čísel (PCA - Principal Component Analysis) kovariační matice vytvořené z nejbližších bodů dotazovaného bodu. Pro každý bod $\boldsymbol{p}_i$ je sestavena kovariační matice $\mathcal{C}$ následujícím způsobem:

$$\mathcal{C} = \frac{1}{k} \sum_{i=1}^{k} \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}}) \cdot (\boldsymbol{p}_i - \overline{\boldsymbol{p}})^T, \; \mathcal{C} \cdot \vec{\mathrm{v}}_j = \lambda_j \cdot \vec{\mathrm{v}}_j, \; j \in \{0, 1, 2\} \tag{4.1}$$

kde k je počet příslušných sousedů $\boldsymbol{p}_i$, $\overline{\boldsymbol{p}}$ představuje 3D centroid nejbližších sousedů, λ_j je j-té vlastní číslo kovariační matice a $\vec{\mathrm{v}}_j$ je j-tý vlastní vektor [21].

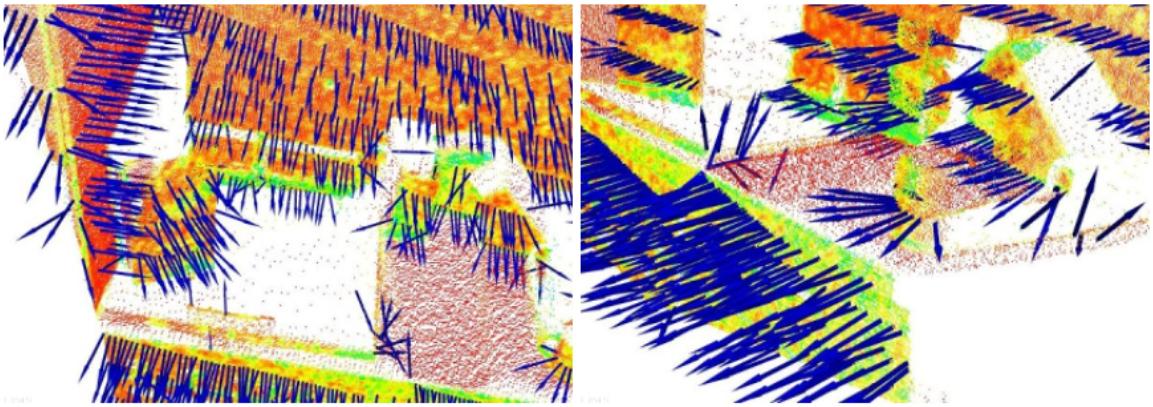
Jelikož není obecně používaný žádný matematický postup jak vyřešit znaménko normály, orientace vypočítaná pomocí PCA je nejednoznačná a není konzistentně orientována v celém pointcloudu.



Obrázek 4.4: Nekonzistentní orientace normálových vektorů [21]

Řešení tohoto problému je triviální, pokud známe bod kamery, ze kterého byla scéna snímána. K tomu, aby byly všechny normály $\vec{n}_i$ orientovány souhlasně vůči pozorovacímu bodu v_p, je potřeba splnit podmínku:

$$\vec{\boldsymbol{n}}_i \cdot (\mathsf{v}_p - \boldsymbol{p}_i) > 0 \tag{4.2}$$

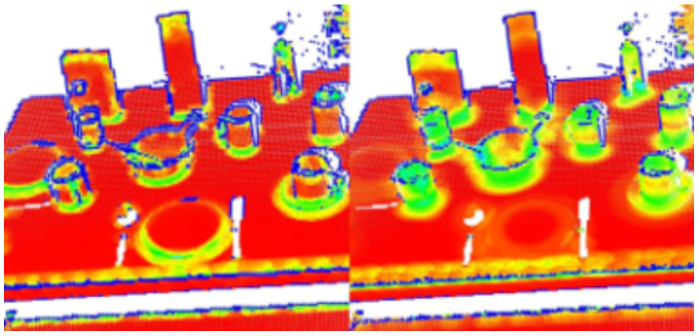


Obrázek 4.5: Konzistentní orientace normálových vektorů [21]

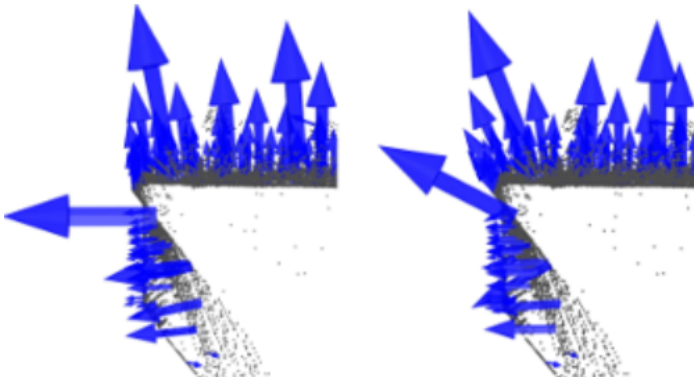
4.5 Výběr vhodného měřítka

Jak bylo řečeno v kapitole 4.4, plošná normála bodu v pointcloudu je vypočtena z jeho okolních bodů (tzv. k-neighborhood). Specifika problému odhadu nejbližších sousedících bodů vedou k otázce výběru vhodného měřítka vzorkování datové množiny pointcloudu. Je tedy třeba určit, jaký počet sousedů k či jaký poloměr r by měl být zvolen při určování množiny nejbližších sousedů.

Tento problém je nesmírně důležitý a představuje omezující faktor v automatickém odhadu (tj. bez uživatelského nastavení) bodově interpretovaných prvků. Pro ilustraci je na obrázku 4.6 uvedeno srovnání dvou případů s různě zvoleným měřítkem, malé měřítko vůči velkému (malé r či k vůči velkému r či k). Levá část obrázku zobrazuje vhodně zvolené měřítko, takže vypočtené normály jsou přibližně kolmé pro dva ploché povrchy a několik malých hran viditelných po celém stole. Pravá část obrázku znázorňuje stav, kdy bylo nastaveno příliš velké měřítko. Oblast sousedství je tedy širší a tvořena větším krycím povrchem okolo sousedících bodů. Důsledkem nastavením nevhodného, zde příliš velkého, měřítka je zkreslení odhadovaných bodových oblastí reprezentující povrch. Z toho vyplývá pootočení povrchových normál na hranách a okrajích ploch, a tedy potlačení jemných detailů, viz obrázek 4.7.



Obrázek 4.6: Porovnání vytvořených povrchů s různým měřítkem vzorkování [2]



Obrázek 4.7: Porovnání povrchových normál s různým měřítkem vzorkování [2]

Jak lze vidět, velikost měřítka musí být zvolena na úrovni detailů, které jsou pro danou aplikaci potřeba. Pro zjednodušení, jestli je v pointcloudu nějaké důležité zakřivení, hrana či detail, který má být na snímaném objektu viditelný, musíme zvolit menší měřítko. V opačném případě je naopak vhodné zvolit měřítko větší.

4.6 Deskriptory

Deskriptor je datový soubor, který nese informaci o geometrii okolí dotazovaného bodu. Díky deskriptorům je tedy možné přesně popsat konkrétní bod. Jedním z deskriptorů jsou povrchové normály. Ty nesou informaci o geometrii okolí díky výpočtu zahrnujícím k zvolených nejbližších bodů či body v okolí s poloměrem r. Povrchové normály však nejsou dostatečné k identifikaci objektu v obraze.

K tomu aby byly deskriptory optimálně stanoveny, musí splnit následující kritéria:

- Odolnost vůči transformacím - přesněji k hmotným transformacím. Jelikož se nemění vzdálenost mezi body, deskriptory se nemění při změně rotace či při translaci.
- Odolnost vůči šumu - chyby měření, které způsobují šum, by neměly mít velký vliv na odhad deskriptorů.
- Neměnnost vůči změně vzorkování - při změně vzorkování by se deskriptory neměly změnit vůbec či jen minimálně.

Deskriptory tedy slouží k jednoznačné identifikaci bodu napříč pointcloudy nezávisle na šumu, změně vzorkování či translaci. Rozlišujeme lokální a globální deskriptory. Lokální deskriptory popisují pouze povrchy v okolí zvoleného bodu. Globální deskriptory jsou určeny pro popis celého objektu. Metody založené na lokálních deskriptorech dosahují lepších výsledků rozpoznávání v přeplněných scénách než metody založené na globálních deskriptorech. Jejich využití se ukázalo lepší i pro rozpoznání ve 2D i 3D obraze.

Mezi nejběžnější metody využívající deskriptory patří:

- Signature of Histograms of OrienTations (SHOT),
- Point Feature Histogram (PFH),
- Fast Point Feature Histograms (FPFH).

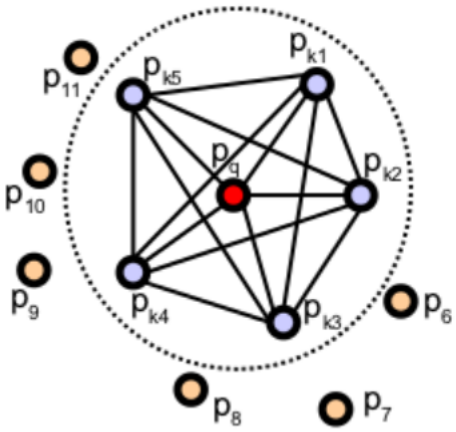
V této diplomové práci je použita metoda FPFH, která vychází z metody PFH.

4.6.1 Point Feature Histograms (PFH)

Cílem metody PFH je co nejpřesněji popsat geometrické vlastnosti sousedících bodů zobecněním středního zakřivení kolem bodu pomocí mnohodimenzionálního histogramu hodnot. Tento vysoce dimenzionální hyperprostor poskytuje deskriptorům informativní popis daných prvků a velmi dobře zvládá různé vzorkovací hustoty bodů nebo různé hladiny šumu v okolí bodu.

Reprezentace deskriptorů PFH je založena na vztazích mezi body v k-okolí a na k nim náležejícím odhadnutým normálám. Jinak řečeno, metoda se snaží najít co nejlepší variantu vzorkovaného povrchu tak, že zohledňuje všechny interakce mezi směry odhadnutých normál povrchu. Výsledný hyperprostor je závislý na kvalitě odhadnutých normál povrchu v každém bodě.

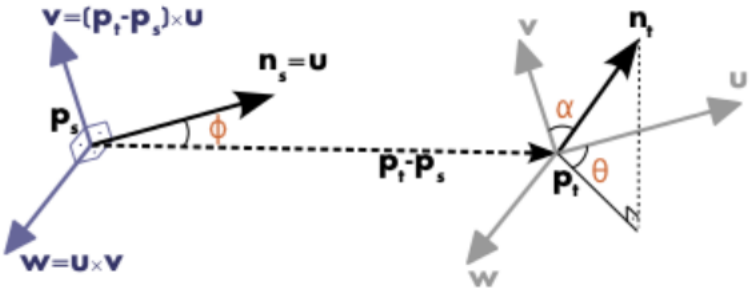
Na obrázku č. 4.8 je znázorněn diagram ovlivněné oblasti výpočtu PFH pro počáteční bod p_q (znázorněn uprostřed a označený červenou barvou). Kružnice na obrázku znázorňuje 3D sféru s poloměrem r a se všemi k sousedy (body jejichž vzdálenost od počátečního bodu je menší než poloměr r) propojenými ve společnou síť. Finální PFH deskriptor je vypočten jako histogram vztahů mezi všemi páry modře znázorněných bodů. Jeho výpočetní složitost tak odpovídá $O(k^2)$, kde k je počet sousedů ve výpočtu.



Obrázek 4.8: Diagram oblasti výpočtu PFH deskriptorů [21]

Pro výpočet relativního rozdílu mezi body p_s a p_t a jejich normálami $\vec{n}_s$ a $\vec{n}_t$ definujeme fixní Darbouxův rámec (fixní souřadnice vektorů $\vec{u}, \vec{v}, \vec{w}$) pro jeden ze dvou bodů. Pokud zvolíme p_s za zdrojový bod a p_t jako cílový, za p_s dosadíme ten bod, který svírá nejmenší úhel mezi svojí normálou a čarou spojující body p_s a p_t (viz Obr. 4.9). Základ Darbouxového rámce můžeme v bodě p_s definovat jako:

$$\begin{aligned}\vec{u} &= \overrightarrow{n_s} \\ \vec{v} &= \vec{u} \times \frac{(p_t - p_s)}{\|p_t - p_s\|_2} \\ \vec{w} &= \vec{u} \times \vec{v}\end{aligned}$$



Obrázek 4.9: Grafická reprezentace Darbouxového rámce a úhlových PFH deskriptorů dvojice bodů p_s a p_t [21]

Použitím Darbouxového rámce **uvw** definujeme rozdíl dvou normál $\vec{n}_s$ a $\vec{n}_t$ jako množinu úhlových deskriptorů následovně:

$$\begin{aligned}\alpha &= \vec{v} \cdot \overrightarrow{n_t} \\ \phi &= \vec{u} \cdot \frac{(p_t - p_s)}{d} \\ \theta &= \arctan(\vec{w} \cdot \overrightarrow{n_t}, \vec{u} \cdot \overrightarrow{n_t})\end{aligned}$$

kde d je Euklidovská vzdálenost mezi dvěma body p_s a p_t ($d = ||p_t - p_s||_2$). Čtveřice α, ϕ, θ, d je vypočítána pro každou dvojici bodů v k-sousedství. Tato čtveřice představuje zredukování 12 hodnot dvou bodů a jejich normál (souřadnic x, y, z a popis normály) na 4 hodnoty [21].

Pro vytvoření výsledné reprezentace PFH pro daný bod je množina všech čtveřic vložena do histogramu. Proces pokračuje rozdělením rozsahu hodnot každého deskriptoru do b subintervalů a počítá se počet výskytu hodnot daných bodů v těchto intervalech. Protože 3 hodnoty z každé čtveřice jsou hodnoty úhlů mezi normálami, lze jejich hodnoty snadno normalizovat do stejného intervalu na jednotkové kružnici. Čtvrtá složka **d** pro datové množiny 2.5D nemá díky zvětšující se vzdálenosti sousedních bodů od pozorovacího bodu v obvyklých případech robotiky velký význam. Bylo otestováno, že zanedbání složky **d** pro skeny s velkou lokální hustotou bodů je pro výpočet prospěšné (urychluje ho).

4.6.2 Fast Point Feature Histograms (FPFH)

Teoretická výpočetní složitost metody PFH pro daný pointcloud s n body je $O(nk^2)$, kde k je počet sousedů pro každý bod p v pointcloudu [20]. Pro real-time aplikace nebo aplikace blížící se real-time aplikacím představuje výpočet PFH v hustých oblastech sousedících bodů jedno z hlavních úskalí.

Zjednodušená verze metody PFH , tj. metoda Fast Point Feature Histograms, má teoretickou výpočetní složitost $O(nk)$ při zachování popisných vlastností PFH. Pro zjednodušení výpočtu postupujeme následovně:

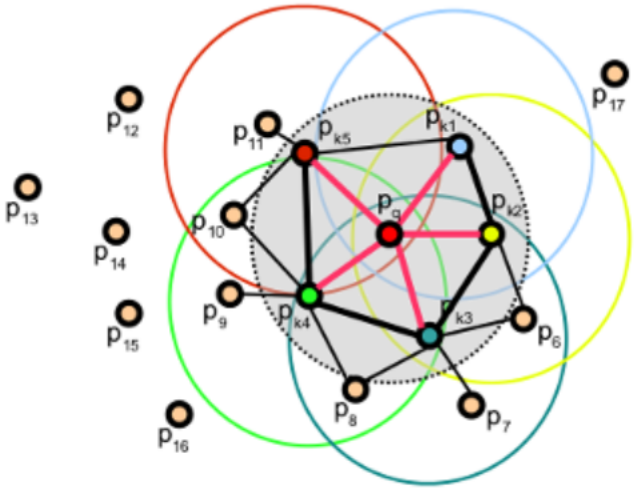
- V prvním kroku se pro každý bod p_q vypočítá trojice úhlových hodnot α, ϕ, θ mezi tímto bodem a jeho sousedními body stejným způsobem, jako je popsáno v kapitole 4.6.1. Tento výpočet se nazývá Simplified Point Feature Histograms (SPFH).

- V druhém kroku je pro každý bod znovu určena oblast sousedství a sousední hodnoty SPFH jsou použity k vážení finálního histogramu daných bodů p_q, který se nazývá Fast Point Feature Histogram, následovně:

$$FPFH(\boldsymbol{p}_q) = SPFH(\boldsymbol{p}_q) + \frac{1}{k}\sum_{i=1}^{k}\frac{1}{\omega_k} \cdot SPFH(\boldsymbol{p}_k) \tag{4.3}$$

  kde váha ω_k reprezentuje vzdálenost mezi tázaným bodem p_q a sousedním bodem p_k ve zvolené metrice [21].

Výsledné hodnoty jsou závislé na hodnotách sousedních bodů a ty zase na hodnotách ve svém okolí. Na obrázku 4.10 je červenými čarami vyznačen vztah mezi bodem p_q a jeho sousedními body (díky vypočítaným SPFH hodnotám). Dodatečná spojení získaná díky vylepšení PFH deskriptorů jsou vyznačeny černými čarami. Páry, které se počítají dvakrát, jsou vyznačeny tlustou černou čárou.



Obrázek 4.10: Zobrazení všech vztahů mezi bodem p_q a jeho sousedními body v metodě FPFH [20]

Hlavní rozdíly mezi PFH a FPFH [3]:

- Metoda FPFH nepropojuje v oblasti sousedství p_q každý bod s každým, a tedy postrádá párové hodnoty, které mohou přispět k zachycení přesnější geometrie v okolí bodu p_q.
- Metoda PFH modeluje přesněji povrch v okolí bodu p_q, zatímco metoda FPFH do svého výpočtu zahrnuje i páry tvořené s body vně oblasti sousedství bodu p_q, tedy zahrnuje body ve vzdálenosti větší než r a menší než $2r$.
- FPFH díky opakovanému přepočítávání vah hodnot diagramu kombinuje hodnoty SPFH a může tak znovu získat některé hodnoty soudních párů.
- Celková složitost FPFH je nižší, je tedy vhodné ji použít pro real-time aplikace.
- Výsledný histogram je vytvořený z d různých histogramů, každý pro jednu dimenzi daného deskriptoru, které jsou následně spojeny dohromady.

4.7 Párování deskriptorů a shlukování korespondencí

Pro správnou transformaci objektu je třeba na základě vypočtených lokálních 3D deskriptorů pro scénu nalézt možné dvojice odpovídajících si bodů. U každého klíčového bodu v pointcloudu musíme spárovat deskriptory hledáním shod s deskriptory vypočítanými pro model. Pro párování se využívá datových struktur jako jsou K-D strom (viz kapitola 4.3), pomocí kterých se vykonává vyhledávání nejbližších sousedů získáváním Euklidovské vzdálenosti mezi deskriptory a nalezení dvojic bodů s nejmenšími k vzdálenostmi. Parametr k je definovaný jako vstup, většinou se používá $k > 1$ a ovlivňuje výsledný počet korespondencí (větší k představuje větší počet korespondencí). Jelikož se ve scéně může nacházet více hledaných objektů (např při bin pickingu více krabic), musí být každý deskriptor

ze scény porovnán se všemi deskriptory modelů v databázi. Výsledkem porovnávání deskriptorů je seznam všech korespondencí mezi klíčovými body ve scéně a modely v databázi.

Všechny korespondence nezaručují, že se daný objekt skutečně na pozici nachází. Vezmeme-li v úvahu například krabici se dvěma klíčovými body umístěnými v opačných rozích se známou vzdáleností mezi těmito body, mohou se v pointcloudu objevit dva klíčové body, které odpovídají popisu hledaných bodů, avšak jejich vzdálenost je větší. Pokud nebereme v úvahu deformace, tak zjistíme, že tyto body nenáleží hledanému objektu, ale týkají se více objektů.

K identifikaci objektu se využívá metoda **shlukování korespondencí**, která seskupuje geometricky stabilní korespondence a ostatní vyřadí. Rotace a translace jsou povolené, ale jiné transformace nesplňují kritéria (nejsou povolené). Pro získání pozice objektu je minimální počet korespondencí tři, shluky s menším počtem korespondencí tedy ignorujeme [13].

Kapitola 5

Identifikace pomocí geometrické algebry CGA

Netradiční způsob pro identifikaci sfér v pointcloudu je pomocí geometrické algebry. Geometrická algebra představuje matematický aparát umožňující jednoduše počítat translace a rotace v 3D prostoru. V této diplomové práci byla k identifikaci sfér v pointcloudu použita konformní geometrická algebra (CGA).

5.1 Základy geometrických algeber

5.1.1 Základní algebraické prvky geometrických algeber

Zatímco bázové vektory $e_1, e_2, ..., e_n$ jsou základními algebraickými prvky $n-$dimenzionální vektorové algebry, u $n-$dimenzionální geometrické algebry tvoří pouze část algebraických prvků. Základním algebraickým prvkem geometrických algeber jsou blady. $N-$dimenzionální geometrická algebra se skládá z bladů stupně $0, 1, 2, ..., n$, kde skalár je 0-blade (blade stupně 0) a 1-blady jsou bázové vektory $e_1, e_2, ..., e_n$. 2-blady $e_i \wedge e_j$ jsou tvořené složením dvou 1-bladů. Obdobně to platí pro blady vyšších stupňů. Každá geometrická algebra obsahuje pouze jeden blade nejvyššího stupně $n, I = e_1 \wedge e_2 ... \wedge e_n$, ten se nazývá pseudoskalár. Lineární kombinace $k-$bladů se nazývá $k-$vektor. Například je $e_2 \wedge e_3 + e_1 \wedge e_2$ 2-vektor (bivektor). Lineární kombinací bladů s rozdílnými stupni vzniká multivektor [14].

Blade	**Stupeň**
1	0
e_1	1
e_2	1
e_3	1
$e_1 \wedge e_2$	2
$e_1 \wedge e_3$	2
$e_2 \wedge e_3$	2
$e_1 \wedge e_2 \wedge e_3$	3

Tabulka 5.1: Seznam všech bladů v 3D Euklidovské geometrické algebře

5.1.2 Součiny v geometrických algebrách

V geometrických algebrách se nachází 3 různé součiny - geometrický, vnitřní a vnější. Vnější součin se využívá ke konstrukci a k průsečíkům objektů, vnitřní součin nalezne uplatnění ve výpočtu úhlů a vzdáleností a geometrický součin obecně popisuje transformace.

Zápis	**Význam**
AB	Geometrický součin A a B
$A \wedge B$	Vnější součin A a B
$A \cdot B$	Vnitřní součin A a B

Tabulka 5.2: Součiny v geometrických algebrách

Vnější součin

Vnější součin v geometrické algebře je antikomutativní, distributivní a asociativní.

Vlastnost	**Význam**
Antikomutativita	$u \wedge v = -(v \wedge u)$
Distributivita	$u \wedge (v + w) = u \wedge v + u \wedge w$
Asociativita	$u \wedge (v \wedge w) = (u \wedge v) \wedge w$

Tabulka 5.3: Vlastnosti vnějšího součinu

Vnější součin dvou stejných vektorů je 0:

$$u \wedge u = -(u \wedge u) = 0$$

Příklad 5.1. Úprava výrazu v geometrické algebře
Zjednodušme výraz

$$c = (e_1 + e_2) \wedge (e_1 - e_2)$$

na základě vlastnosti distributivity, můžeme výraz převést na:

$$c = (e_1 \wedge e_1) - (e_1 \wedge e_2) + (e_2 \wedge e_1) - (e_2 \wedge e_2)$$

jelikož $u \wedge u = 0$ a vnější součin je antikomutativní:

$$c = -2(e_1 \wedge e_2)$$

Vnitřní součin

Vnitřní součin je komutativní částí geometrického součinu. Jedná se o rozšíření skalárního součinu z vektorů na obecné blady. Vnitřní součin dvou kolmých vektorů je roven 0, například $e_1 \cdot e_2 = 0$. Vnitřní součin dvou stejných vektorů v geometrické algebře je roven 1. V geometrické algebře není vnitřní součin, na rozdíl od skalárního, definovaný pouze na vektory. Vnitřní součin snižuje stupeň bladu. Pokud máme 2-blade a 1-blade, jejich vnitřní součin bude blade stupně 1.

Vnitřní součin geometrické algebry se používá při výpočtu vzdáleností a velikosti úhlů, protože obsahuje metrické informace.

Geometrický součin

Geometrický součin je dán jako součet vnitřního a vnějšího součinu. Využívá se především pro práci s transformacemi. Pro vektory u a v je geometrický součin uv definován jako:

$$uv = u \cdot v + u \wedge v \tag{5.1}$$

Vyjádření vnějšího a vnitřního součinu pomocí součinu geometrického:

$$u \cdot v = \frac{1}{2}(uv + vu)$$

$$u \wedge v = \frac{1}{2}(uv - vu)$$

Příklad 5.2. Ukázka práce se součiny v geometrické algebře

$$u = (e_1 \wedge e_2), v = ((e_1 + e_2) \wedge e_3)$$

$$\begin{aligned}
uv &= (e_1 \wedge e_2)((e_1 + e_2) \wedge e_3) \\
uv &= (e_1 \wedge e_2)(e_1 \wedge e_3 + e_2 \wedge e_3) \\
uv &= (\underbrace{e_1 \cdot e_2}_{0} + e_1 \wedge e_2)(\underbrace{e_1 \cdot e_3}_{0} + e_1 \wedge e_3 + \underbrace{e_2 \cdot e_3}_{0} + e_2 \wedge e_3) \\
uv &= e_1 e_2 (e_1 e_3 + e_2 e_3) \\
uv &= e_1 e_2 e_1 e_3 + e_1 \underbrace{e_2 e_2}_{1} e_3 \\
uv &= -e_2 e_1 e_1 e_3 + e_1 e_3 \\
uv &= -e_2 e_3 + e_1 e_3 \\
uv &= -(e_2 \wedge e_3) + e_1 \wedge e_3
\end{aligned}$$

5.1.3 Involuce a dualita

Důležitým pojmem v geometrických algebrách je involuce. Involucí se rozumí takové operace, které mapují prvek sám na sebe, je-li tato operace použita dvakrát. Jedním z takovýchto prvků je inverze, ta je obecně popsána v algebře vztahem:

$$A^{-1}A = AA^{-1} = 1$$

Přičemž platí $I^{-1} = -I$.

Dalším takovým prvkem je duální prvek. Duálním prvkem rozumíme takový prvek, který vznikl jeho podělením pseudoskalárem. Duální prvek značíme *.

Příklad 5.3. Použití involuce a duality

$$\begin{aligned}
(e_2 \wedge (e_1 + e_3))^* &= (e_2 \wedge (e_1 + e_3))(e_1 e_2 e_3)^{-1} \\
(e_2 \wedge (e_1 + e_3))^* &= (e_2 \wedge (e_1 + e_3))(-e_1 e_2 e_3) \\
(e_2 \wedge (e_1 + e_3))^* &= -(e_2(e_1 + e_3))e_1 e_2 e_3 \\
(e_2 \wedge (e_1 + e_3))^* &= -e_2 e_1 e_1 e_2 e_3 - e_2 e_3 e_1 e_2 e_3 \\
(e_2 \wedge (e_1 + e_3))^* &= -e_3 - e_3 e_1 e_2 e_2 e_3 = -e_3 + e_1
\end{aligned}$$

Duality prvku se využívá k popisu objektů v geometrických algebrách. V Geometrických algebrách se objekty dají popsat dvěma možnými způsoby, vnitřní (IPNS) a vnější (OPNS) reprezentací. Mezi oběma způsoby platí vztah duality:

$$A^*_{OPNS} = A_{IPNS}$$

5.2 CGA - Conformal Geometry algebra

CGA, aneb konformní geometrická algebra, je rozšíření 3D geometrické algebry do 5D prostoru. Hlavní výhodou CGA je reprezentace bodů, sfér a ploch pomocí stejné algebraické struktury a díky tomu je možné problémy z 3D prostoru formulovat jednodušeji a více intuitivně.

CGA používá tři euklidovské bázové vektory e_1, e_2, e_3 a dva další bázové vektory e_-, e_+. Přičemž platí : $e_-^2 = -1$, $e_+^2 = 1$, $e_+ \cdot e_- = 0$

Pomocí bázových vektorů e_-, e_+ se vyjádří nové bázové vektory e_0, e_∞. Tyto bázové vektory představují počátek a nekonečno. Vztah mezi novými a starými bázovými vektory:

$$e_0 = \frac{1}{2}(e_- + e_+), \qquad e_\infty = e_- + e_+$$

Pro nové bázové vektory e_0, e_∞ platí:

$$e_0^2 = e_\infty^2 = 0, \qquad e_\infty \cdot e_0 = -1$$

5.2.1 Reprezentace objektů

Reprezentace bodu

Pro reprezentaci bodu v CGA prostoru je původní bod z 3D prostoru rozšířen lineární kombinací bázových vektorů e_1, e_2, e_3, e_∞ a e_0 podle následující rovnice:

$$P = x_1e_1 + x_2e_2 + x_3e_3 + \frac{1}{2}(x_1e_1 + x_2e_2 + x_3e_3)^2e_\infty + e_0 = \mathbf{x} + \frac{1}{2}\mathbf{x}^2e_\infty + e_0, \tag{5.2}$$

kde $\mathbf{x}^2$ je skalární součin.

$$\mathbf{x}^2 = x_1^2 + x_2^2 + x_3^2$$

Reprezentace sféry

Sféra může být reprezentována 2 různými způsoby:

- Pomocí středu P a poloměru r

$$S = P - \frac{1}{2}r^2e_\infty \tag{5.3}$$

  nebo užitím vzorce 5.2.

$$S = \mathbf{x} + \frac{1}{2}(\mathbf{x}^2 - r^2)e_\infty + e_0 \tag{5.4}$$

- Pomocí 4 bodů náležících sféře.

$$S^* = P_1 \wedge P_2 \wedge P_3 \wedge P_4 \tag{5.5}$$

Jak je vidět ze vzorce 5.3, sféra s nulovým poloměrem odpovídá v CGA bodu.

Reprezentace plochy

Plocha může být reprezentována 2 různými způsoby:

- Pomocí normálového vektoru z 3D Euklidovského prostoru $\mathbf{n}$ a vzdálenosti od počátku d.

$$\pi = \mathbf{n} + de_\infty \tag{5.6}$$

- Pomocí 3 bodů náležících ploše a bodu v nekonečnu.

$$\pi^* = P_1 \wedge P_2 \wedge P_3 \wedge e_\infty \tag{5.7}$$

Jak je vidět ze vzorce 5.7, plocha odpovídá sféře s nekonečným poloměrem.

Reprezentace kružnice

Kružnice může být reprezentována 2 různými způsoby:

- Pomocí průsečíku 2 sfér.

$$Z = S_1 \wedge S_2 \tag{5.8}$$

- Pomocí 3 bodů náležících kružnici.

$$Z^* = P_1 \wedge P_2 \wedge P_3 \tag{5.9}$$

Reprezentace přímky

Přímka může být reprezentována 2 různými způsoby:

- Pomocí průsečíku 2 ploch.

$$L = \pi_1 \wedge \pi_2 \tag{5.10}$$

- Pomocí 2 bodů náležících přímce a bodu v nekonečnu.

$$L^* = P_1 \wedge P_2 \wedge e_\infty \tag{5.11}$$

Objekt	**IPNS reprezentace**	**OPNS reprezentace**
Bod	$P = \mathbf{x} + \frac{1}{2}\mathbf{x}^2 e_\infty + e_0$	
Sféra	$S = P - \frac{1}{2}r^2 e_\infty$	$S^* = P_1 \wedge P_2 \wedge P_3 \wedge P_4$
Plocha	$\pi = \mathbf{n} + de_\infty$	$\pi^* = P_1 \wedge P_2 \wedge P_3 \wedge e_\infty$
Kružnice	$Z = S_1 \wedge S_2$	$Z^* = P_1 \wedge P_2 \wedge P_3$
Přímka	$L = \pi_1 \wedge \pi_2$	$L^* = P_1 \wedge P_2 \wedge e_\infty$

Tabulka 5.4: Objekty v CGA

$U \cdot V$	Plocha	Sféra	Bod
Plocha	Úhel mezi plochami	Euklidovská vzdálenost od středu	Euklidovská vzdálenost
Sféra	Euklidovská vzdálenost od středu	Vzdálenost	Vzdálenost
Bod	Euklidovská vzdálenost od středu	Vzdálenost	Euklidovská vzdálenost

Tabulka 5.5: Aplikace vnitřního součinu

5.2.2 Vzdálenost v CGA

V CGA jsou body, plochy a sféry reprezentovány vektory. Vnitřní součin těchto objektů je skalár a představuje vzdálenost. Vnitřní součin slouží například k:

- měření Euklidovské vzdálenosti dvou bodů,
- určení vzdálenosti bodu od plochy,
- určení, zda se bod nachází uvnitř či vně sféry.

Příklad 5.4. Výpočet vzdálenosti dvou bodů v CGA

$$\begin{aligned} P \cdot S &= \left(\mathbf{p} + \tfrac{1}{2}\mathbf{p}^2 e_\infty + e_0\right) \cdot \left(\mathbf{s} + \tfrac{1}{2}\mathbf{s}^2 e_\infty + e_0\right) \\ &= \mathbf{p} \cdot \mathbf{s} + \tfrac{1}{2}\mathbf{s}^2 \underbrace{\mathbf{p} \cdot e_\infty}_{0} + \underbrace{\mathbf{p} \cdot e_0}_{0} \\ &+ \tfrac{1}{2}\mathbf{p}^2 \underbrace{e_\infty \cdot \mathbf{s}}_{0} + \tfrac{1}{4}\mathbf{p}^2\mathbf{s}^2 \underbrace{e_\infty^2}_{0} + \tfrac{1}{2}\mathbf{p}^2 \underbrace{e_\infty \cdot e_0}_{-1} \\ &+ \underbrace{e_0 \cdot \mathbf{s}}_{0} + \tfrac{1}{2}\mathbf{s}^2 \underbrace{e_0 \cdot e_\infty}_{-1} + \underbrace{e_0^2}_{0} \\ &= \mathbf{p} \cdot \mathbf{s} - \tfrac{1}{2}\mathbf{p}^2 - \tfrac{1}{2}\mathbf{s}^2 = -\tfrac{1}{2}(\mathbf{p}^2 - \mathbf{s}^2) \end{aligned}$$

5.3 Clifford library

Pro možnost pracovat s geometrickými algebrami v Pythonu byl zvolen modul Clifford. Modul Clifford obsahuje řadu funkcí, které lze využít pro identifikaci objektu (konkrétně sféry) z pointcloudu [7].

5.4 Aplikace CGA k identifikaci sfér

Jednou z výhod CGA je stejná algebraická struktura pro přímky, sféry a plochy. Díky tomu je možné použít metodu nejmenších čtverců na jakýkoliv pointcloud (i subpointcloud) a dostaneme řešení odpovídající jedné z těchto možností. Pro hledání sfér je tedy CGA ideální. Objekty jiných tvarů (např. čtverec) tato algebra není schopná identifikovat [14].

5.4.1 Metoda nejmenších čtverců

Metoda nejmenších čtverců minimalizuje sumu kvadrátu vzdáleností (v CGA tedy vnitřních součinů) mezi všemi body a uvažovanou sférou/plochou.

$$\min \sum_{i=1}^{n} (P_i \cdot X^*)^2 \tag{5.12}$$

Pro řešení je rovnice 5.12 převedena do bilineární formy:

$$\min(x^t B x), \tag{5.13}$$

kde

$$x = (x_1, x_2, x_3, x_4, x_5)$$

a matice 5×5

$$B = \begin{pmatrix} b_{1,1} & b_{1,2} & b_{1,3} & b_{1,4} & b_{1,5} \\ b_{2,1} & b_{2,2} & b_{2,3} & b_{2,4} & b_{2,5} \\ b_{3,1} & b_{3,2} & b_{3,3} & b_{3,4} & b_{3,5} \\ b_{4,1} & b_{4,2} & b_{4,3} & b_{4,4} & b_{4,5} \\ b_{5,1} & b_{5,2} & b_{5,3} & b_{5,4} & b_{5,5} \end{pmatrix}$$

má vstupy

$$b_{j,k} = \sum_{i=1}^{n} w_{i,j} w_{i,k}, \quad w_{i,k} = \begin{cases} p_{i,k} & \text{když } k \in \{1,2,3\} \\ -1 & \text{když } k = 4 \\ -\frac{1}{2}\mathbf{p}_i^2 & \text{když } k = 5 \end{cases}$$

Je možné dokázat, že řešení minimalizačního problému je dáno vlastním vektorem matice B, který odpovídá nejmenšímu vlastnímu číslu. Jelikož je matice B symetrická, pak pro hledání vlastních vektorů a čísel můžeme použít singulární rozklad matice:

$$\mathbf{B} = \mathbf{U}\mathbf{\Sigma}\mathbf{U}^T$$

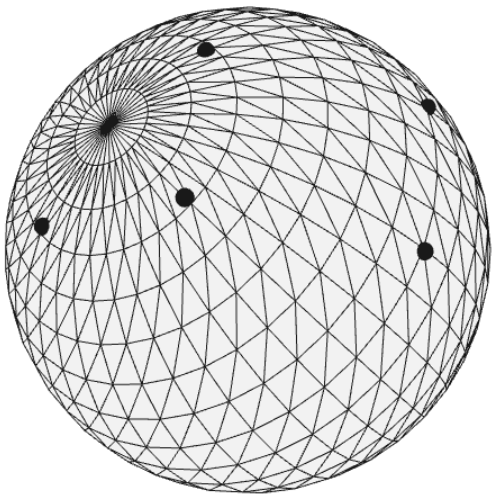
kde $\mathbf{U}$ je matice, ve které každý sloupec představuje vlastní vektor a matice $\mathbf{\Sigma}$ je diagonální matice, kde každé číslo na diagonále představuje vlastní číslo příslušného vlastního vektoru.

Příklad 5.5. Ukázka použití metody nejmenších čtverců v aplikaci

```
P1=[1, 0, 0]
P2=[1, 1, 0]
P3=[0, 0, 1]
P4=[0, 1, 1]
P5=[-1, 0, 1]
points = [P1, P2, P3, P4, P5]
shape = (len(points), 5)
X = empty(shape)
for i in range(len(points)):
    X[i, 0] = points[i][0]
    X[i, 1] = points[i][1]
    X[i, 2] = points[i][2]
    X[i, 3] = -1
    X[i, 4] = -0.5 * (X[i, 0] ** 2 + X[i, 1] ** 2 + X[i, 2] ** 2)
```

```
B = dot(X.transpose(), X)
eigenvalue, eigenvector = linalg.eigh(B)
minValueIndice = argmin(eigenvalue)
solution = np.empty(shape = (5,1))
norm = eigenvector.transpose()[minValueIndice, 4]
for i in range(5):
    solution[i] = eigenvector.transpose()[minValueIndice, i] / norm
solution[3] = eigenvector.transpose()[minValueIndice, 4] / norm
solution[4] = eigenvector.transpose()[minValueIndice, 3] / norm
```

Dostáváme řešení ve tvaru $-(0.5e_1) + (0.5e_2) - (0.5e_3) + (1.0e_0) - (1.0e_\infty)$, které odpovídá sféře s poloměrem $\sqrt{2.75}$ a středem $[-0.5, 0.5, -0.5]$.



Obrázek 5.1: Metoda nejmenších čtverců

5.4.2 RANSAC

RANdom SAmple Consenus (RANSAC), v překladu shoda náhodných vzorků, je iterativní nedeterministická metoda. To znamená, že výsledek metody je akceptovatelný jen při určitém množství iterací a nemusíme vždy obdržet stejný výsledek, či vůbec nalézt optimální řešení.

RANSAC pracuje na principu náhodného výběru minimálního počtu vzorků pro popis daného objektu a zjistí, kolik bodů přísluší tomuto objektu (inliers) a kolik ne (outliers).

Postup identifikace objektu dle metody RANSAC:

1. Náhodně vyber minimální počet bodů potřebných k vygenerování objektu.
2. Spočítej počet inlierů a outlierů.
3. Pokud má aktuální model více inlierů než dosud nejlepší, stává se sám nejlepším modelem.

Abychom mohli RANSAC použít, potřebujeme kromě dat a modelu znát i následující 3 parametry:

- Chybovou toleranci E_T, která se využívá k určení, zda-li bod patří do skupiny inlierů.
- Počet iterací k.
- Práh t, který odpovídá počtu inlierů, u kterých můžeme prohlásit, že odpovídají hledanému modelu.

Cyklus hledání je ukončen, pokud má kandidát více než t inlierů nebo po k iteracích. Po poslední iteraci je model upřesněn pomocí metody nejmenších čtverců [11], [17].

Kapitola 6

Praktická část

Diplomová práce se zaměřuje na identifikaci sfér v pointcloudu obecnou metodou využívající deskriptory a metodou na základě geometrické algebry CGA. Obě metody jsou na závěr porovnány z hlediska jejich vhodnosti použití.

Metoda identifikace pomocí deskriptorů využívá knihovnu Open3d. Tato knihovna obsahuje mnoho funkcí a algoritmů umožnující práci s pointcloudy. V diplomové práci jsou použity metody knihovny k vytvoření aplikace Global registration, která přiřazuje šablonu na konkrétní místo v pointcloudu.

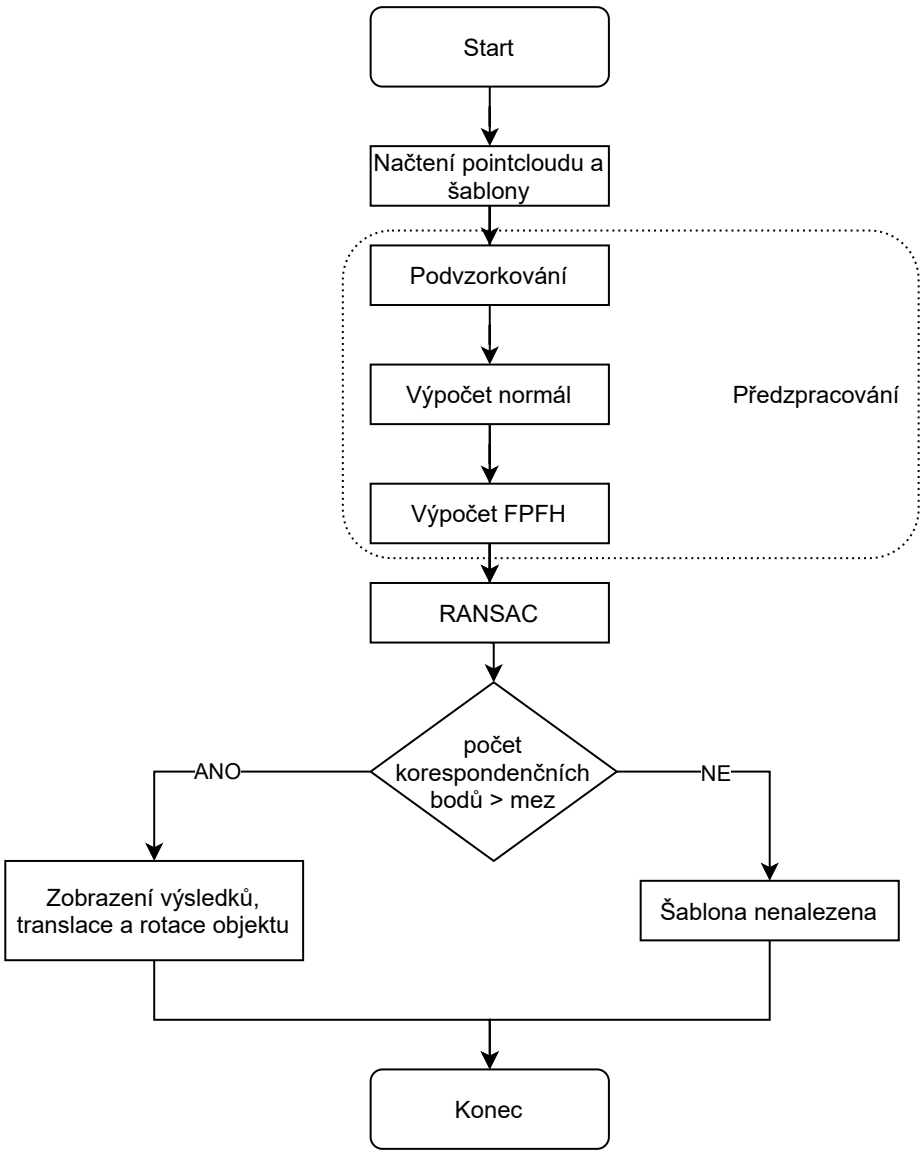
K metodě identifikace využívající geometrickou algebru CGA je použita knihovna Clifford, která obsahuje základní funkce geometrické algebry, společně s metodou RANSAC.

6.1 Aplikace Global registration

Aplikace Global registration se snaží zasadit pomocí algoritmů předem známou šablonu povrchu tělesa do snímku z kamery, a dopočítat tak správnou polohu a orientaci hledaného objektu. Výsledkem navrženého algoritmu je matice translace a rotace hledaného objektu. Díky výstupům z aplikace Global registration lze navigovat robotické rameno k uchopení daného objektu, tedy realizovat úlohu bin picking.

Přesnost aplikace Global Registration je závislá na přesnosti kamery a na zvolených parametrech pro výpočet. Parametry musí být zvoleny tak, aby byl nalezen kompromis mezi časem a přesností výsledků identifikace objektu. Pro real-time aplikace by při požadavku na vysokou přesnost byl čas příliš velký a aplikace by tedy nebyla použitelná.

Jak je vidět na vývojovém diagramu 6.1, aplikace Global registration byla vytvořena pro detekci jednoho objektu v pointcloudu. Principiálně je snadné ji přeformulovat na detekci objektů v pointcloudu na základě předpřipravených vhodných šablon a to jejím spouštěním v cyklu. Cyklus by při nesplnění podmínky pro množství korespondenčních bodů (či fitness funkce) načetl další šablonu a prošel celý program znovu.



Obrázek 6.1: Schéma Global registration

6.1.1 Potřebné předchystání aplikace Global registration

Pro správné fungování aplikace Global registration je nutné mít předpřipravené šablony. Šablony musí být ve stejném formátu, v jakém jsou snímky z kamery, tedy formát pointcloud. Šablony musí obsahovat reprezentativní část hledaného objektu, přičemž by šablona měla být:

- Unikátní - aby nebylo více možností jak danou šablonu do pointcloudu uložit.
- Věrohodná - rozměry šablony by měly odpovídat rozměrům hledaného tělesa.
- Dobře pozorovatelná - pro snímání z mnoha úhlů natočení.

Pro aplikaci v této diplomové práci byly zvoleny jednoduché objekty - sféry a kvádry. K vytvoření šablony byly využity pointcloudy nasnímané z kamery PhoXi 3D M. Z pointcloudu stačilo vybrat daný objekt a pomocí translace a rotace ho posunout tak, aby střed hledaného objektu byl v souřadnicích [0,0,0] (více v kapitole 6.1.3).

Získat pointcloudy je také možné získat převedením .STL souboru, který je buď dodán nebo vytvořen z modelu, do pointcloudového souboru. Této konverze se dá docílit pomocí metody Sample points v programu CloudCompare. Tato cesta byla vytvořené šablon byla také experimentálně ověřena.

6.1.2 CloudCompare

CloudCompare je program umožňující práci s pointcloudy. Poskytuje základní nástroje pro manuální úpravu a vykreslování pointcloudů. Dále také náročnější procesní algoritmy jako například:

- registrace,
- segmentace,
- projekce.

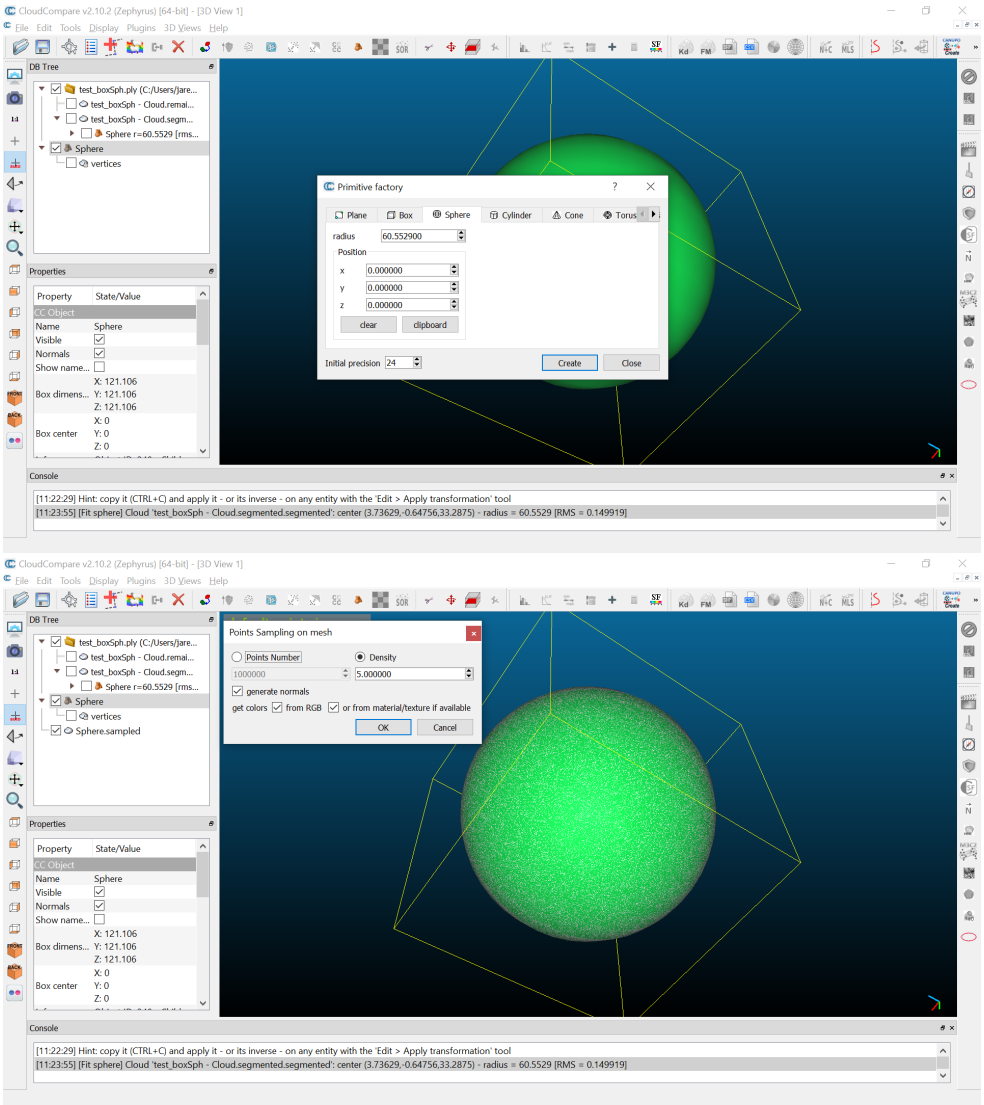
Program CloudCompare byl použit k vytvoření šablon.

6.1.3 Vytvoření šablon

Šablony byly vytvořeny různými způsoby díky své specifičnosti. Jak již bylo zmíněno v kapitole 6.1.1, obecný postup by byl vytvořit .STL soubor (tj. výstup CAD modelu) a následně jeho reprezentativní část pokrýt množstvím bodů. Je více než vhodné myslet již při vytváření CAD modelu na to, aby byl střed souřadného systému umístěn ve středu tělesa. Tím se docílí toho, že střed skutečného tělesa se zachová i pro jeho reprezentativní část a díky tomu je možné použít aplikaci bin picking bez dalších konverzí. Vytvoření šablony sféry je znázorněno na obrázku 6.2.

Vytvoření šablony sféry

Prvním hledaným objektem byla sféra. Jelikož u sfér nezáleží na rotaci, byla v programu CloudCompare vytvořena sféra s požadovaným poloměrem na nulových souřadnicích. Tato sféra byla následně pokryta body pomocí funkce Sample points. Bylo odzkoušeno, že aplikace Global registration je schopná tuto sféru umístit do nasnímaného pointcloudu a šablonu není nutné upravovat pouze na reprezentativní část. Výsledná šablona sféry je zobrazena na obrázku 6.3.

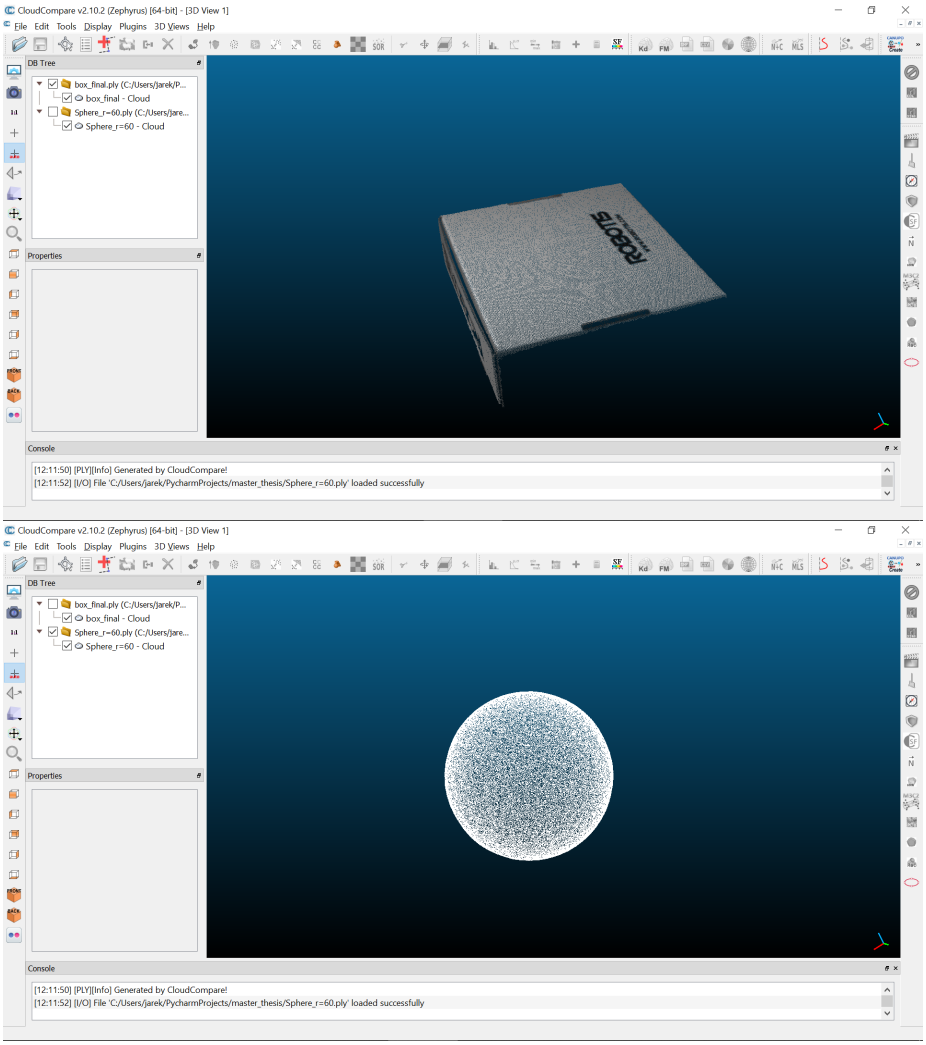


Obrázek 6.2: Vytváření šablon

Vytvoření šablony krabice

Pro vytvoření šablony krabice byly naměřeny její základní parametry, ale pro vytvoření přesného CAD modelu by bylo třeba znát i její zářezy a další její specifika. Pomocí funkcí programu CloudCompare byl vytvořen kvádr naměřených délek a výřez z pointcloudu byl zarovnán s tímto kvádrem. K zarovnání byla použita funkce Align programu CloudCompare.

Vytvořená šablony jsou zobrazeny na obrázku 6.3



Obrázek 6.3: Šablony hledaných objektů

6.1.4 Implementace Global registration

Aplikace Global registration, která vykonává přiřazení šablony do daného pointcloudu, je vytvořena na základě algoritmů knihovny Open3d a používá se k robotickému vidění a lokalizaci objektu. Výstupem z aplikace je matice rotace a vektor translace známého středu hledaného objektu a počet korespondenčních bodů obou pointcloudů.

Aplikaci Global registration je možné dělit do souvisejících celků na předzpracování, výpočet deskriptorů a přiřazení šablony do pointcloudu.

Předzpracování a výpočet deskriptorů

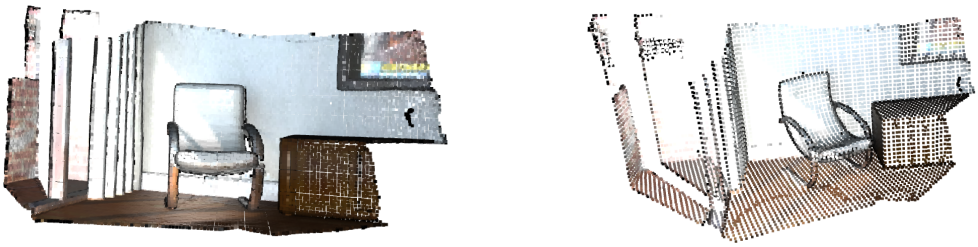
Tato část aplikace se zabývá předzpracováním pointcloudů, aby bylo možné přiřadit šablonu do pointcloudu a získat výsledek v real-time čase.

Funkce prepare_dataset načte dvojici pointcloudů, přičemž jeden je zdroj (source) a jeden šablona (target), vykreslí je v aktuálním stavu a předá je na zpracování další funkci `preprocess_point_cloud`. Výstupem je trojice parametrů každého pointcloudu, a to sice: pointcloud, podvzorkovaný pointcloud a příslušné FPFH deskriptory pointcloudu.

```
def prepare_dataset(voxel_size):
    source = o3d.io.read_point_cloud("test_boxSph - Cloud.pcd")
    target = o3d.io.read_point_cloud("box_final.ply")
    draw_registration_result(source, target, np.identity(4), 0)
    source_down, source_fpfh = preprocess_point_cloud(source, voxel_size)
    target_down, target_fpfh = preprocess_point_cloud(target, voxel_size)
    return source, target, source_down, target_down, source_fpfh, target_fpfh
```

Funkce `preprocess_point_cloud` podvzorkuje pointcloud, vypočítá normály a dále FPFH deskriptory. Pro výpočet normál je použita funkce `estimate_normals` z knihovny Open3d. Tato funkce využívá K-D tree (viz kapitola 4.3) pro určení nejbližších sousedů. Na výpočet deskriptorů FPFH je použita funkce `compute_fpfh_feature`, která z povrchových normál vypočítá FPFH deskriptory. Velikosti parametrů byly získány empiricky.

```
def preprocess_point_cloud(pcd, voxel_size):

    pcd_down = pcd.voxel_down_sample(voxel_size)
    radius_normal = voxel_size * 4
    pcd_down.estimate_normals(
        o3d.geometry.KDTreeSearchParamHybrid(radius=radius_normal, max_nn=50))
    radius_feature = voxel_size * 50
    pcd_fpfh = o3d.registration.compute_fpfh_feature(
        pcd_down,
        o3d.geometry.KDTreeSearchParamHybrid(radius=radius_feature,
                                             max_nn=150))
    return pcd_down, pcd_fpfh
```



Obrázek 6.4: Ukázka příkladu pointcloudu a jeho podvzorkování

Přiřazování šablony

Když jsou pointcloudy předzpracované, je možné spustit vlastní metodu Global registration, která slouží k registrace šablony (odtud vyplývá název registration).

K provedení Global registration je použita metoda RANSAC. V každé iteraci je vybrán určitý počet náhodných bodů. Jejich odpovídající body cílového pointcloudu jsou nalezeny prohledáním nejbližších sousedů v 33-dimenzionálním FPFH prostoru. Snahou je co nejrychleji odstranit body, které nesplňují požadovaná kritéria, a tím urychlit metodu RANSAC. Důležitým parametrem je `RANSACConvergenceCriteria`, který obsahuje maximum iterací RANSAC a maximální počet ověřovacích kroků. Čím vyšší je počet iterací a ověřovacích kroků, tím přesnější je výsledek z metody RANSAC.

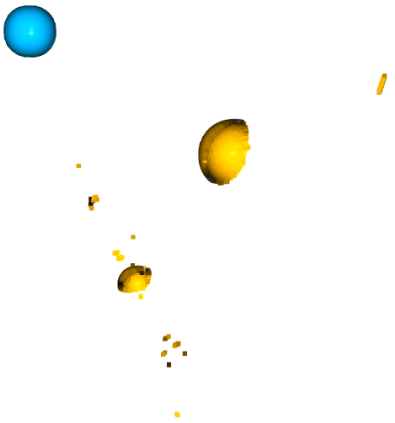
Knihovna Open3D používá na odstranění nevhodných bodů následující kritéria:

- `CorrespondenceCheckerBasedOnDistance` - zkontroluje, zda pointcloudy jsou dostatečně blízko (méně než nastavená mez).
- `CorrespondenceCheckerBasedOnEdgeLength` - zkontroluje, zda délky libovolných dvou hran vykreslených ze zdrojového a cílového pointcloudu jsou dostatečně podobné.

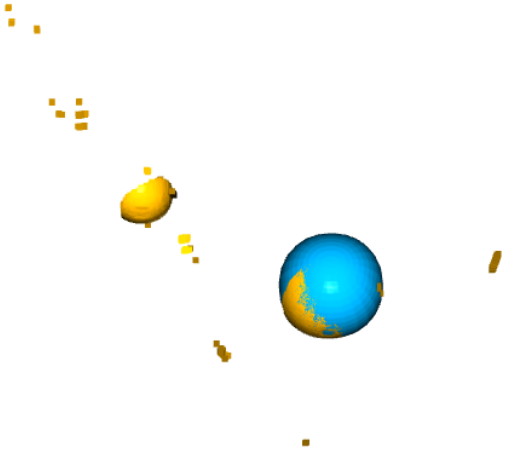
Pokud není na konci programu Global registration dostatečný počet korespondenčních bodů, program vyhodnotí, že nebyla šablona nalezena.

```
def execute_global_registration(source_down, target_down, source_fpfh,
                                target_fpfh, voxel_size):
    distance_threshold = voxel_size * 150
    result = o3d.registration.registration_ransac_based_on_feature_matching
        (
        source_down, target_down, source_fpfh,
        target_fpfh, distance_threshold,
        o3d.registration.TransformationEstimationPointToPoint(False), 4, [
            o3d.registration.CorrespondenceCheckerBasedOnEdgeLength(0.9),
            o3d.registration.CorrespondenceCheckerBasedOnDistance(
                distance_threshold)
        ], o3d.registration.RANSACConvergenceCriteria(900000000, 100000)
        )
    if len(np.asarray(result.correspondence_set)) < corr_treshold:
        print("TEMPLATE NOT FOUND IN SOURCE")
        result = 0
    return result
```

Na obrázcích 6.5 a 6.6 je zobrazen stav hledání sféry v pointcloudu.



Obrázek 6.5: Počáteční stav hledání sféry s poloměrem 60



Obrázek 6.6: Konečný stav hledání sféry s poloměrem 60

Výstup z programu Global registration pro nalezení sféry je reprezentován transformační maticí.

```
Transformation is:
[[ 8.85236923e-01 -8.64273176e-02 -4.57040379e-01  2.92691012e+02]
 [ 4.29688244e-01 -2.24302710e-01  8.74674972e-01 -7.37891072e+02]
 [-1.78111207e-01 -9.70679459e-01 -1.61424239e-01  3.11252645e+02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  1.00000000e+00]]
registration::RegistrationResult with fitness=8.110437e-01,
inlier_rmse=1.632298e+00, and correspondence_set size of 6683
```

6.2 Aplikace CGA

Identifikace objektu pomocí geometrické algebry CGA je hlavní náplní této diplomové práce. Při realizaci aplikace CGA použito teoretických znalostí popsaných v kapitole 5 a knihovny Clifford.

6.2.1 Vytvoření objektů v CGA

Aby bylo možné identifikovat objekt pomocí algebry CGA, byl vytvořen v aplikaci vzor sféry. Sféra se v geometrické algebře dá interpretovat 2 různými způsoby (viz kapitola 5.2.1). Pro vytvoření sféry byl naprogramován převod bodu z 3D souřadnic do prostoru CGA a naopak.

Na převod bodu z 3D souřadnic do prostoru CGA byl aplikován vztah 5.2, kde bázové vektory e_1, e_2, e_3, e_0 a e_∞ jsou reprezentovány výrazy $e1$, $e2$, $e3$, $e0$ a $einf$ a p představuje bod v 3d prostoru, tedy $p = [x, y, z]$. Převod z CGA zpět do Euklidovského 3D prostoru zahrnuje pouze prvky u $e1$, $e2$, $e3$ a uloží je zpět.

```
@staticmethod
    def from3D(p):
        return (
            (p[0] * e1) + (p[1] * e2) + (p[2] * e3)
            + 0.5 * (p[0]**2  + p[1]**2  + p[2] **2 ) * einf
            + eo
        )

@staticmethod
    def to3D(S):
        return ([ S[e1], S[e2], S[e3] ])
```

Na základě nadefinovaní bodu byla naprogramována reprezentace sféry. Při vytvoření sféry byly využity vztahy uvedené v teoretické části, a to sice 5.3 a 5.4. První metoda vytvoří sféru ze středu a jejího poloměru, druhá ze 4 bodů z prostoru či pointcloudu převedených do CGA. Všechny tyto možnosti jsou využity v aplikaci.

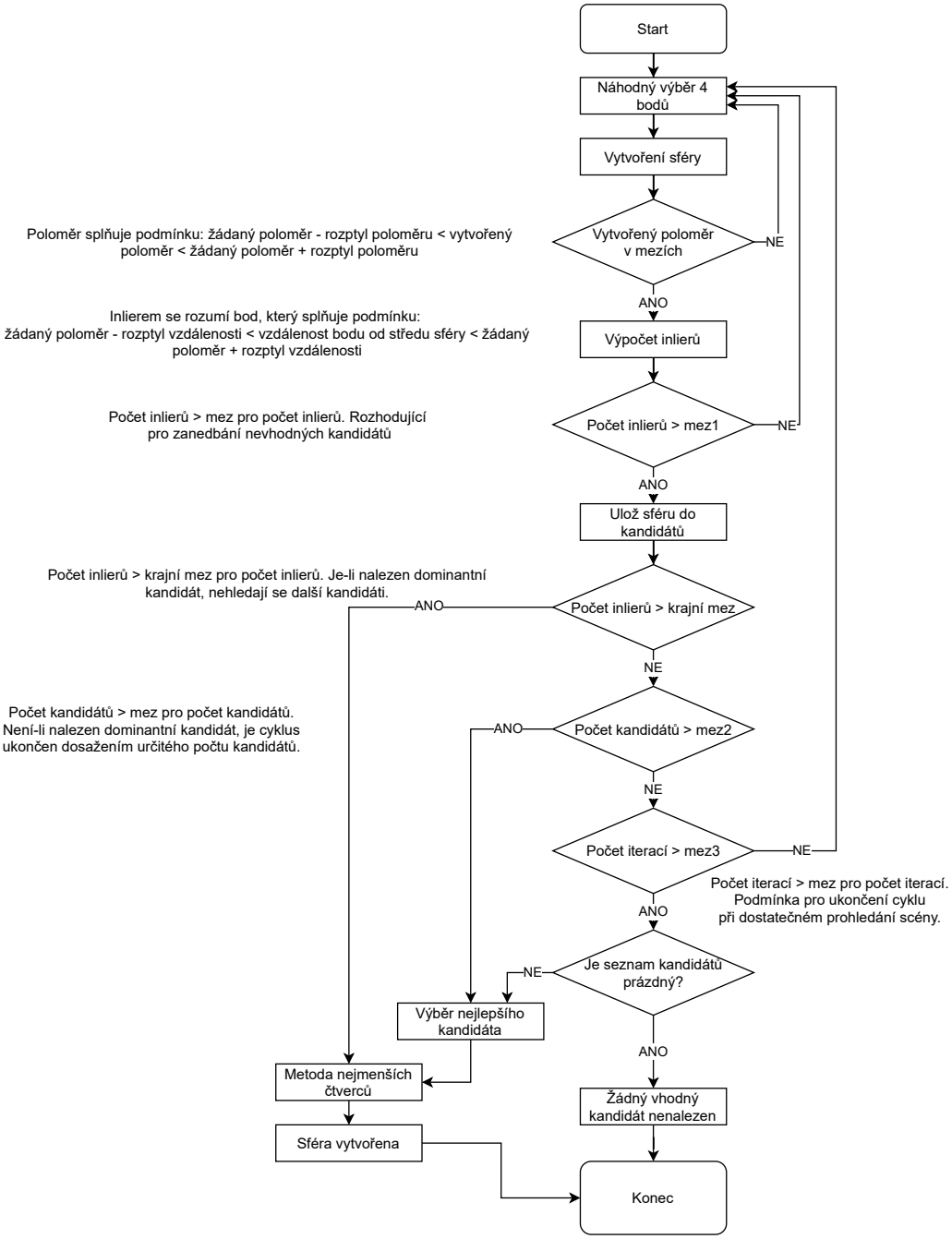
```
    @staticmethod
    def fromCenter(center, radius):
        return (
            (Point.from3D(center) - 0.5 * (radius ** 2) * einf)
            )

    @staticmethod
    def fromPoints(p, q, r, s):
        return (
            (Point.from3D(p) ^ Point.from3D(q) ^
                Point.from3D(r) ^ Point.from3D(s))
        )
```

Pro nalezení výsledků byla naprogramována metoda isInlier, která metoda určuje, jestli je bod dostatečně blízko požadovanému poloměru. Metoda se využívá pro výpočet příslušných bodů. Konkrétně tedy bod, který je v určitém předem určeném rozptylu od poloměru, je připsán do seznamu příslušných bodů. Díky dostatečnému počtu těchto bodů je možné určit, zda byla sféra nalezena.

```
@staticmethod
def isInlier3D(sphere, width, point):
    center = Sphere.center(sphere)
    sphere_radius = Sphere.radius(sphere)
    dist = np.sqrt(
        (center[e1] - point[0]) ** 2
        + (center[e2] - point[1]) ** 2
        + (center[e3] - point[2]) ** 2
    )
    if (
        (dist > (sphere_radius - width))
        and (dist < (sphere_radius + width))
    ):
        return True
    else:
        return False
```

6.2.2 RANSAC

S využitím funkcí popsaných v předchozí kapitole byla naprogramována samotná RANSAC. Cyklický průběh této metody je znázorněn na vývojovém diagramu 6.7.



Obrázek 6.7: Vývojový diagram cyklu RANSAC

Metoda RANSAC je dle zmíněného vývojového diagramu implementovaná ve funkci *findSphere*. Stav výpočtu je uchováván v následujících klíčových proměnných:

pointsCnt uchovává celkový počet bodů v pointcloudu pro výběr náhodného indexu,

candidates obsahuje seznam nalezených sfér s dostatečným počtem inlierů,

candidInliers body příslušící aktuálně nalezené sféře.

Výpočet byl dále urychlen díky uchovávání spočtených inlierů jednotlivých kadnidátů `candidInliers` v seznamu `indexlist` a inliery tak nemusí být počítány opakovaně.

Pro přístup k počtu bodů v pointcloudu byla použita funkce numpy.asarray() z knihovny Open3. Tím dostaneme matici 3×n, kde n je počet bodů a 3 je dimenze systému. Pro počet bodů pointlcoudu tedy platí:

```
pointsCnt = int(np.asarray(pointcloud.points).size /_spaceDimensions)
```

Celý cyklus RANSAC se vykonává, pokud není splněna jedna z podmínek:

- Počet kandidátů není dostatečně velký pro ukončení cyklu.
- Počet iterací cyklu byl dostatečně velký pro určení výsledku.
- Byl nalezen dominantní kandidát.

Pro reprezentaci sféry jsou v CGA 4 body dostatečným počtem bodů. Cyklus RANSAC tedy na začátku náhodně vybere 4 body a snaží se z nich vytvořit sféru.

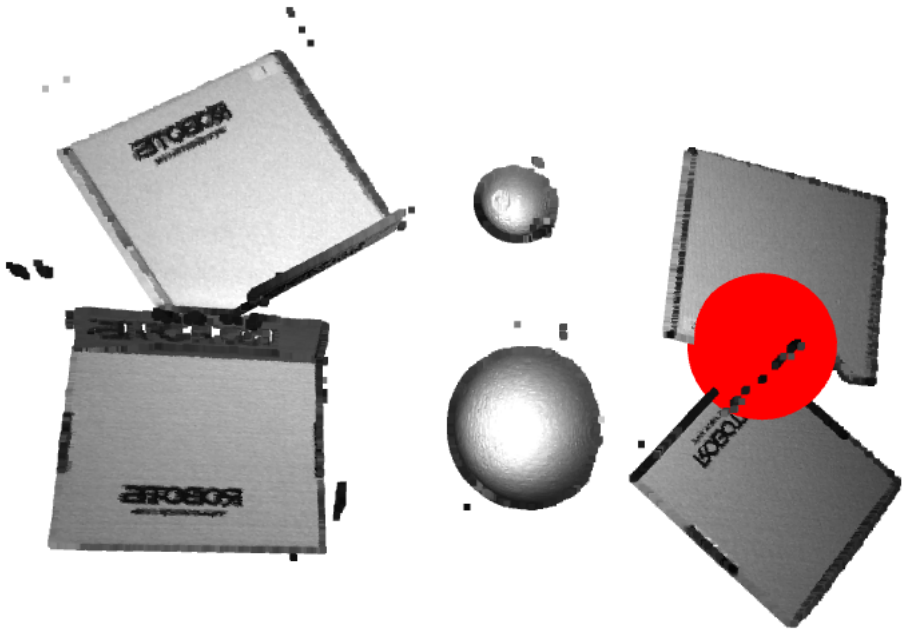
```
# Vyber náhodné 4 indexy z pointcloudu
  ran = [None,None,None,None]
  for i in range(len(ran)):
      tmp = randint(0, pointsCnt - 1)
      # Zkontroluj, zda náhodně vybraný bod již nebyl vybrán v
                  této iteraci
      while tmp in ran:
          tmp = randint(0, pointsCnt - 1)
      ran[i] = tmp
  # Vytvoř z náhodných bodů sféru v CGA
  sphere = Sphere.fromPointcloud(pointcloud, ran[0], ran[1],
                  ran[2], ran[3])
```

Nyní dochází k ověření, zda vytvořená sféra odpovídá hledanému poloměru (se zadaným rozptylem). Pokud není podmínka splněna, je iterace ukončena a cyklus jde znovu od začátku.

```
tstRadius = Sphere.radius(sphere)
if ((tstRadius > targetRadius * (1 + radiusTolerance))
    or (tstRadius < targetRadius * (1 - radiusTolerance))):
    iterCnt += 1
    continue
```

Následně se spočítají inliery pro vytvořenou sféru, která prošla dosavadní cyklus, a tyto body se uloží do listu `inlier_list`. Zároveň se spočítá počet takovýchto bodů. Jejich počet je důležitý pro vynechání nevhodně vytvořených sfér, viz obrázek 6.8.

```
# Spočítej počet příslušných bodů s vytvořenou sférou
   inlierCnt = 0
   inlier_list = []
   for i in range(pointsCnt - 1):
       if Sphere.isInlier3D(sphere, wallWidth,
                            pointcloud.points[i]):
           inlierCnt += 1
           inlier_list.append(i)
```



Obrázek 6.8: Nevhodně vytvořená sféra s nízkým počtem inlierů

Pokud je nalezený příslušný počet bodů dostatečný, je sféra uložena do kandidátů a následně se zjišťuje, jestli je dostatečně kvalitním reprezentantem hledané sféry.

```
            if inlierCnt > _inlierTreshold:
                candidates.append(sphere)
                candidinliers.append(inlierCnt)
                indexlist.append(inlier_list)
                if inlierCnt > _dominantTreshold:
                    sphFound=True
```

Při dosažení maximálního nastaveného počtu iterací bez nalezení sféry je cyklus RANSAC ukončen s výsledkem "sféra nebyla nalezena". To ovšem nemusí znamenat, že se sféra v pointcloudu skutečně nenachází, jelikož metoda RANSAC je založena na náhodném výběru bodů. Může se tedy stát, že body reprezentující sféru nebudou v celém průběhu metody RANSAC vybrány. V tomto případě je možné spustit celý cyklus RANSAC znovu.

Po zjištění vhodných kandidátů s příslušnými počty inlierů, je zapotřebí vybrat kandidáta s největším počtem takovýchto bodů. Tento kandidát je následně přepočítán pomocí metody nejmenších čtverců (`Sphere.fit`), aby sféra vytvořená z těchto bodů odpovídala co nejpřesněji hledané sféře. Program ověřuje, jestli se správně vypočítala nová sféra a pokud ano, prohlásí ji za výsledek. Pokud je nově vytvořená sféra nepřesná (tzn. poloměry sféry vytvořené pomocí metody nejmenších čtverců a sféry z listu candidates se výrazně liší), je vybrán další nejlepší výsledek a přepočítání je zopakováno.

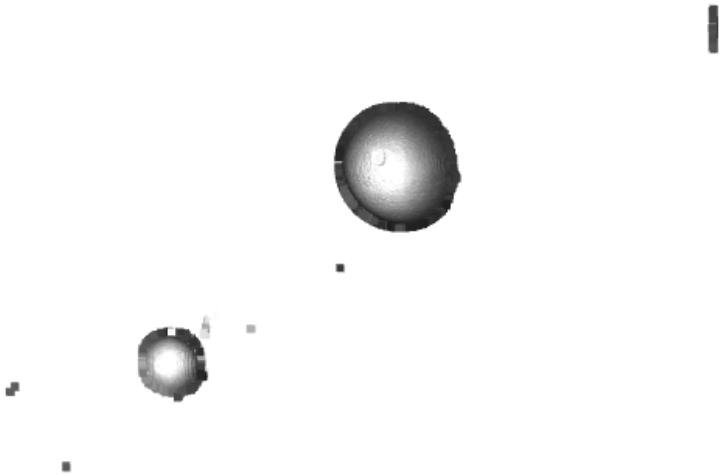
```
    # vrať nejlepšího kandidáta u kterého se podaří vytvořit vhodnou sféru
    for i in range(len(rankedCandidates)):
        sphere = Sphere.fit(
            pointcloud, indexlist[Sphere.find_max_list(indexlist)]
                                )
        radiusOrig = Sphere.radius(candidates[i])
        radiusFit  = Sphere.radius3(sphere)

        # zkontroluj, jestli je sféra správně vytvořená
        if radiusOrig/radiusFit < 1.02 and radiusOrig/radiusFit > 0.98:
            return sphere
```

Jako výsledek program zobrazí nalezený poloměr sféry, její střed, příslušný počet inlierů a pro určení efektivity i časový interval, který byl potřebný k dosažení výsledku. Pro porovnání jsou vypisovány i hodnoty získané přímo z listu candidates. Jak je vidět na obrázku 6.9, výsledky se liší pouze nepatrně.

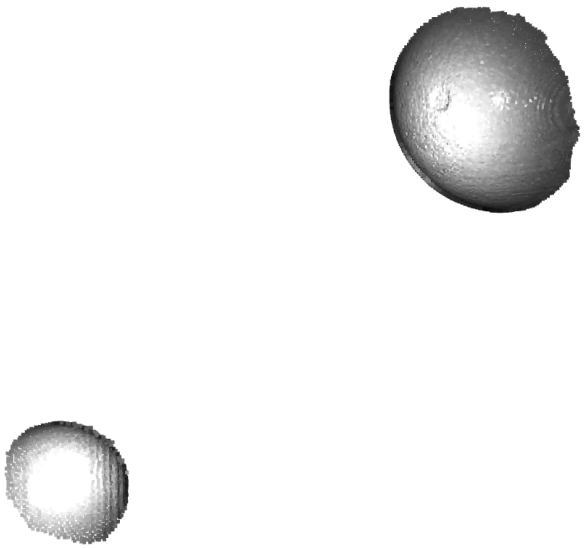
```
Number of inliners: 5758
Original radius 59.96157623845457 , original center [107.66819894931196, 131.557551947322, 821.8501321971097]
LMS radius 60.59462291368141 , LMS center [107.6315832297, 131.57279627805542, 822.7404890366554]
Sphere found succesfully!
==================================
Time elapsed 0:00:12.352413
```

Obrázek 6.9: Porovnání výsledků cyklu

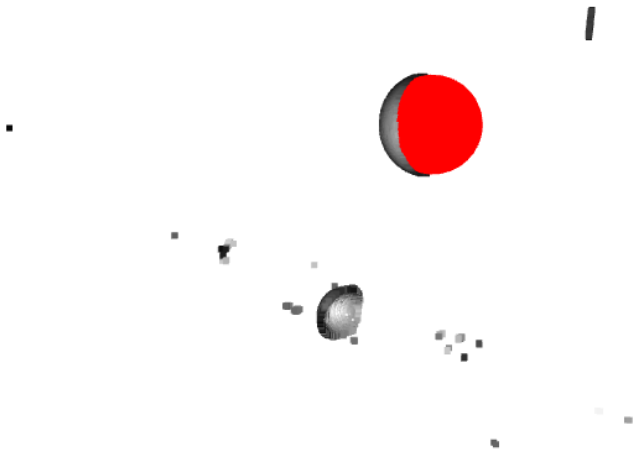
Na její vykreslení je následně použita knihovna Open3d. Na následujících obrázcích 6.10, 6.11 a 6.12 jsou graficky zobrazeny dosažené výsledky.



Obrázek 6.10: Počáteční stav pointcloudu



Obrázek 6.11: Upravený pointcloudu pro rychlejší výpočty

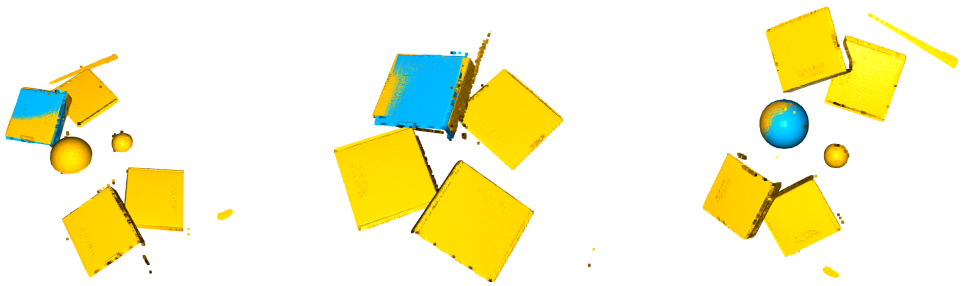


Obrázek 6.12: Finální stav pointcloudu s nalezenou sférou

Kapitola 7

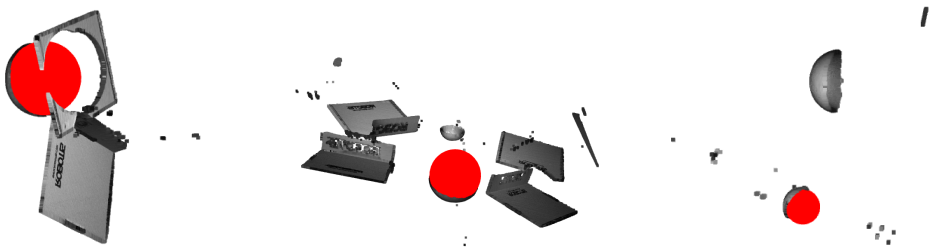
Výsledky práce

Než se mohlo přistoupit k samotnému porovnávání metod, bylo potřeba ověřit jejich správnou funkčnost. Toho bylo docíleno testováním obou metod na zvolených vzorových datech získaných z kamery PhoXi 3D M. U aplikace Global registration (popsané v kapitole 6.1) byly empiricky určeny hodnoty potřebných parametrů pro její běh.



Obrázek 7.1: Ověření funkčnosti aplikace Global registration

Na obrázku 7.1 je ukázána funkčnost aplikace Global registration ve 3 různých případech. Žlutě je označen samotný pointcloud a modře nalezený objekt. Obdobným způsobem je ukázána funkčnost aplikace CGA na obrázku 7.2, kde je červeně označena nalezená sféra.



Obrázek 7.2: Ověření funkčnosti aplikace CGA

Porovnání metod bylo provedeno hledáním sfér s poloměrem 60.5 mm pomocí obou metod ve 3 různých pointcloudech. V každém měření byl zjištěn čas, za který byla sféra nalezena a počet bodů pointcloudu odpovídající nalezené sféře. Čas byl měřen od počátku běhu programu po vypočítání výsledků. Vykreslení sfér už do měřeného času nebylo zahrnuto. Počet bodů odpovídajících nalezené sféře přímo odpovídá přesnosti hledání. Čím více bodů pointcloudu obsahuje vytvořená sféra, tím přesněji je sféra nalezena. Testování bylo provedeno na přenosném počítači s čtyřjádrovým Intel Core i5 procesorem, 8 GB RAM a operačním systémem Windows.

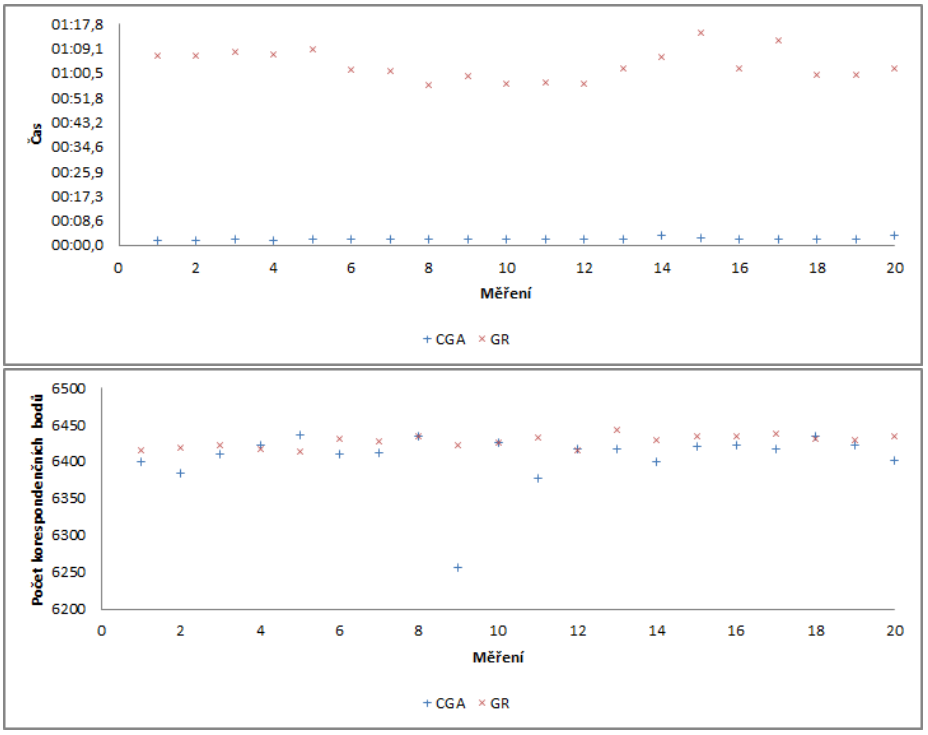
Test 1

První testovací scénář obsahoval vyhledání sféry v pointcloudu tvořeném 8 240 body, který obsahoval 2 rozdílné sféry. Pointcloud je znázorněn na obrázku 7.3. Test byl opakován celkem dvacetkrát pro každou metodu, aby byl minimalizován vliv vnějších vlivů a náhody u metody RANSAC. Naměřené hodnoty jsou uvedeny v tabulce 7.1. Jak je z grafů 7.4 zřejmé, CGA na těchto datech dosahuje při obdobné přesnosti výrazně lepších časů než metoda Global registration (GR).

Obrázek 7.3: Pointcloud test_sphere - Cloud.pcd

Měření	CGA-čas	CGA-počet bodů	GR-čas	GR-počet bodů
1	00:01,8	6400	01:06,5	6415
2	00:01,8	6385	01:06,5	6420
3	00:01,9	6410	01:08,1	6422
4	00:01,8	6422	01:07,3	6418
5	00:02,3	6436	01:08,8	6414
6	00:02,3	6410	01:01,5	6431
7	00:02,2	6412	01:01,4	6427
8	00:01,9	6435	00:56,3	6435
9	00:02,2	6257	00:59,3	6423
10	00:02,3	6426	00:56,8	6426
11	00:02,2	6377	00:57,4	6433
12	00:02,2	6417	00:57,0	6415
13	00:02,2	6418	01:02,2	6443
14	00:03,5	6401	01:06,4	6429
15	00:02,5	6421	01:14,8	6435
16	00:02,1	6423	01:02,2	6434
17	00:02,1	6418	01:12,0	6439
18	00:02,1	6434	00:59,9	6432
19	00:02,2	6422	00:59,8	6429
20	00:03,3	6402	01:02,4	6435

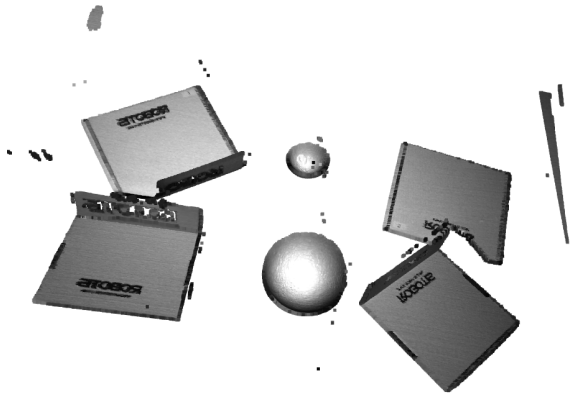
Tabulka 7.1: Hledání sféry v pointcloudu test_sphere - Cloud.pcd



Obrázek 7.4: Grafy testování pointcloudu test_sphere - Cloud.pcd

Test 2

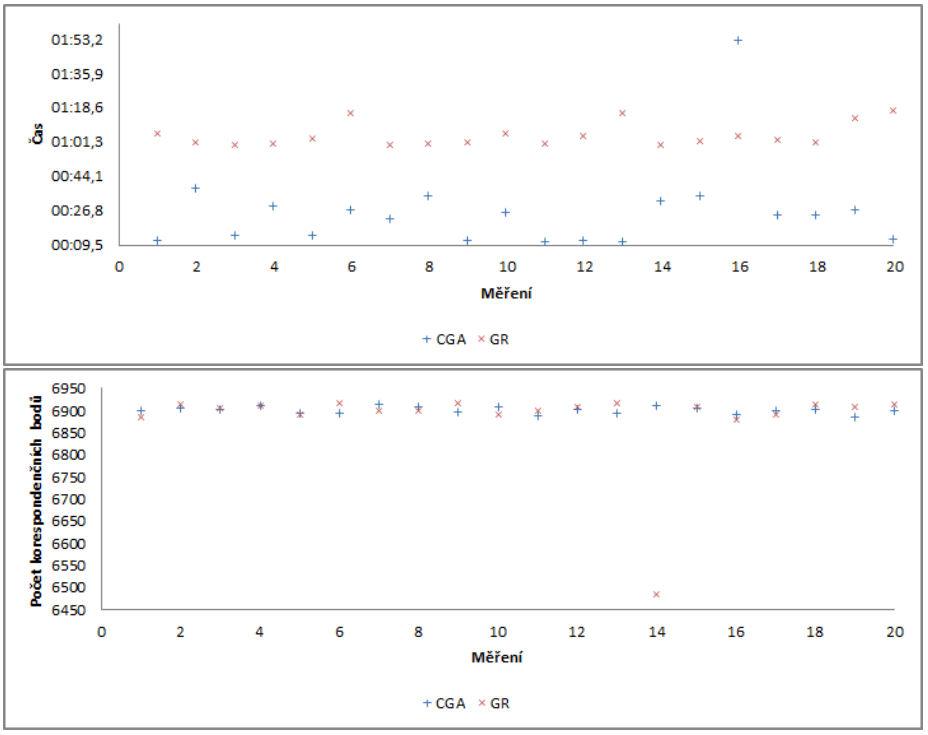
Další porovnávaný pointcloud obsahoval 4 stejné krabice a 2 sféry s různými poloměry. Tento pointcloud byl tvořen 68 082 body, tedy řádově vyšším počtem bodů než v prvním testu. Jak ukazují výsledky měření v tabulce 7.2, i pro tato data dosahuje metoda CGA lepších časů než Global registration. Výjimkou byl jeden případ, kdy metoda Global registration nalezla sféru rychleji, což bylo způsobeno nevhodně vybranými body při průchodu cyklu. Jelikož jsou tyto body vybírány náhodně, je nemožné se této situaci úplně vyhnout.



Obrázek 7.5: Pointcloud test_boxSph - Cloud.pcd

Měření	CGA-čas	CGA-počet bodů	GR-čas	GR-počet bodů
1	00:11,6	6900	01:05,9	6884
2	00:38,5	6906	01:01,3	6914
3	00:14,6	6902	01:00,2	6906
4	00:28,9	6910	01:00,5	6910
5	00:14,7	6894	01:03,6	6890
6	00:27,0	6892	01:16,1	6917
7	00:23,0	6912	01:00,0	6900
8	00:34,3	6909	01:01,0	6900
9	00:11,8	6896	01:01,1	6916
10	00:26,2	6908	01:06,1	6891
11	00:11,3	6888	01:00,9	6900
12	00:12,0	6901	01:04,8	6909
13	00:11,2	6894	01:15,9	6916
14	00:31,6	6910	01:00,2	6483
15	00:34,1	6905	01:01,7	6908
16	01:52,7	6890	01:04,7	6878
17	00:24,8	6900	01:02,6	6890
18	00:24,5	6902	01:01,3	6912
19	00:27,4	6885	01:13,3	6907
20	00:12,3	6900	01:17,7	6912

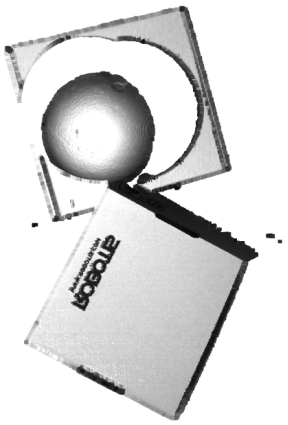
Tabulka 7.2: Hledání sféry v pointcloudu test_boxSph - Cloud.pcd



Obrázek 7.6: Grafy testování pointcloudu test_boxSph - Cloud.pcd

Test 3

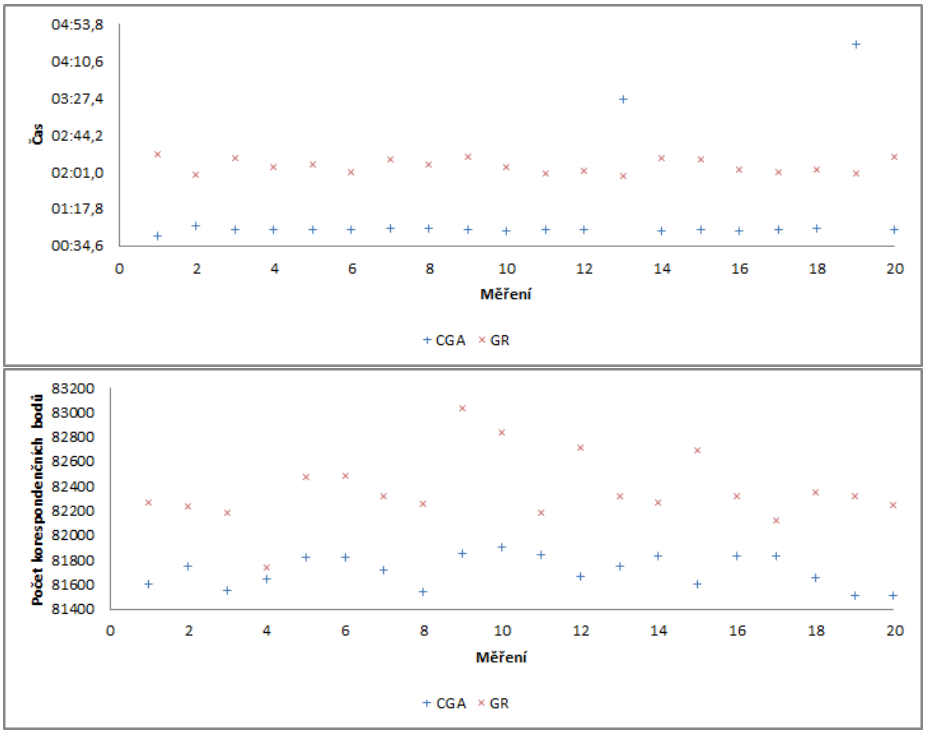
Poslední test se zaměřil na hledání sféry v pointcloudu s vysokým rozlišením 1 169 549 bodů, který obsahoval dvě krabice a hledaná sféra byla umístěna na jedné z nich. Výsledky pomocí aplikace Global registration jsou přesnější (mají větší počet korespondenčních bodů), avšak v 18 z 20 měření bylo těchto výsledků docíleno ve více než dvojnásobném čase. Kvůli velkému počtu bodů počítá program CGA v tomto případě zhruba 40 vteřin počet příslušných bodů. Tento časový údaj bylo možné zjistit díky výpisu z aplikace CGA, který informuje uživatele o počátku počítání inlierů. Jak je vidět na měření 13 a 19, pokud nebylo dostatečného výsledku dosaženo hned při prvním průchodu, byl čas hledání delší než při použití metody Global registration.



Obrázek 7.7: Pointcloud test_spherebox1.ply

Měření	CGA-čas	CGA-počet bodů	GR-čas	GR-počet bodů
1	00:46,3	81608	02:22,4	82270
2	00:57,9	81751	01:57,1	82236
3	00:53,1	81555	02:17,1	82188
4	00:53,0	81651	02:06,4	81741
5	00:53,1	81823	02:10,3	82473
6	00:53,8	81826	02:01,3	82482
7	00:55,1	81719	02:16,4	82323
8	00:55,3	81539	02:10,4	82261
9	00:53,5	81858	02:19,5	83035
10	00:52,8	81903	02:06,2	82835
11	00:53,8	81844	01:58,8	82181
12	00:53,1	81668	02:01,8	82717
13	03:25,5	81745	01:55,7	82319
14	00:52,7	81834	02:17,1	82267
15	00:54,2	81607	02:15,2	82694
16	00:52,8	81838	02:04,2	82317
17	00:53,6	81831	02:00,8	82128
18	00:55,2	81660	02:04,1	82357
19	04:30,6	81507	01:59,5	82323
20	00:53,9	81514	02:18,2	82249

Tabulka 7.3: Hledání sféry v pointcloudu test_spherebox1.ply



Obrázek 7.8: Grafy testování pointcloudu test_spherebox1.ply

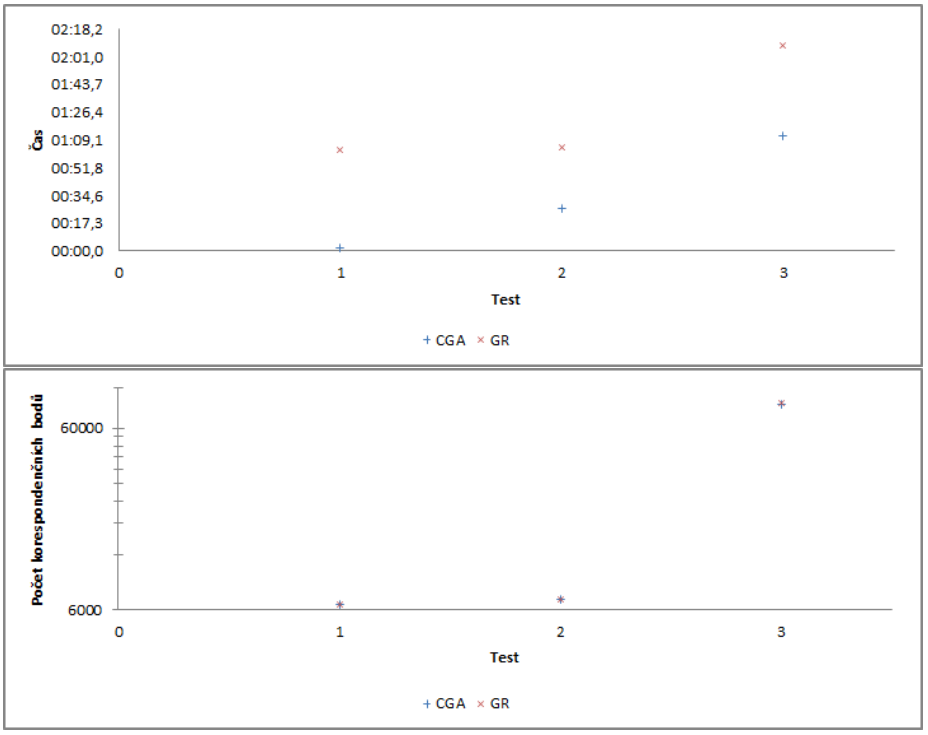
7.1 Zhodnocení aplikací

Ve všech třech testech dosáhla aplikace CGA obdobně přesných výsledků jako aplikace Global registration s rozdílem menším než 1%. V prvním testu dosáhla aplikace Global registration v průměru přesnějších výsledků o 0,4%, v druhém se ukázala přesnější aplikace CGA o 0,2% a ve třetím byla přesnější znovu Global registration, a to o 0,8%. Jak je vidět, přesnost obou aplikací je téměř totožná. Aplikace CGA této přesnosti dosahovala ve výrazně kratších časech. V prvním testu byl průměrný čas potřebný k nalezení sféry o 96,3% kratší než při použití aplikace Global registration. V druhém testu už byla aplikace CGA rychlejší v průměru pouze o 59% a ve třetím již jen o 44%.

Průměrné výsledky testů jsou znázorněny na grafu 7.9, kde první graf popisuje průměrnou dobu potřebnou k nalezení sféry pro jednotlivé testy a druhý graf znázorňuje průměrnou přesnost nalezené sféry dle počtu korespondenčních bodů (inlierů). Vzhledem k rychle rostoucímu počtu bodů v testovacích pointcloudech rostl stejným způsobem i počet korespondenčních bodů, pro zobrazení výsledků všech testů do jednoho grafu bylo zvoleno logaritmické měřítko.

Testy prokázaly, že s rostoucím počtem bodů v jednotlivých pointcloudech doba potřebná k nalezení sféry roste pro aplikaci CGA rychleji než pro aplikaci Global registration. Proto pro pointcloudy s řádově větším počtem bodů než testovanými bude časová úspora aplikace CGA klesat.

Časovou analýzou průběhu metody CGA jsem dospěl k závěru, že se zvyšujícím se počtem bodů neúměrně roste čas ověřování, zda jednotlivé body jsou či nejsou inlierem. Vzhledem ke skutečnosti, že toto ověření provede pro každý bod pointcloudu při každém testovaném kandidátu, stává se toto hlavním úskalím této aplikace.



Obrázek 7.9: Graf porovnání průměrných výsledků testů

Z testování vyplývá, že aplikace CGA je vhodná na hledání sfér v běžných pointcloudech a je schopná najít sféry různých poloměrů bez nutnosti vytvářet šablonu. Díky možnosti nastavit rozmezí pro poloměr je možné nalézt objekty, které se sféře pouze podobají.

Kapitola 8

Závěr

Cílem této diplomové práce bylo popsat přístup robotického 3D vidění pro aplikaci bin picking a vytvořit dvě aplikace pro identifikaci sfér, přičemž důraz byl kladen na použití konformní geometrické algebry (CGA). Vytvořené aplikace jsou schopny rozeznat vybraný objekt na základě dat z kamery s hloubkovým vjemem a umí hledaný objekt lokalizovat. Testování aplikací dokázalo, že cílů bylo úspěšně dosaženo a implementace metody CGA je využitelná pro hledání sfér v pointcloudech.

Bin picking, jakožto spolupráce robota s 3D průmyslovou kamerou pro výběr objektů z obecného zásobníku, je v diplomové práci dostatečně popsán. Jsou rozebrány principy používaných 3D skenerů včetně jejich taxonomie. Blíže je rozebrán princip strukturovaného světla, který je využíván fakultním 3D skenerem PhoXi 3D. Je definován pojem konformní geometrické algebry a její využití pro identifikaci sfér v pointcloudu metodou RANSAC. Tato metoda je porovnána s tradiční metodou založenou na deskriptorech vypočítaných pomocí struktury 3-D stromu. Obě metody byly využity v praktické části diplomové práce.

Metody CGA a pomocí deskriptorů byly testovány na třech různých pointcloudech nasnímaných kamerou PhoXi 3D obsahující hledanou sféru v množině odlišných objektů. Pointcloudy měly zvyšující se rozlišení pro porovnání škálovatelnosti metod a obsahovaly různý počet objektů pro simulaci odlišných podmínek. Z důvodu použití metody RANSAC bylo provedeno v rámci každého testu 20 měření, aby se eliminoval vliv náhody. Bylo dokázáno, že v rámci testovaných pointcloudů je metoda CGA výrazně rychlejší při obdobné přesnosti, nicméně se zvyšující se komplexitou pointcloudu její rychlost klesá. Pokles rychlosti u metody CGA je výraznější než u metody založené na deskriptorech, proto je u této metody důležité mít vhodně předzpracovaná data. Aplikaci CGA lze využít pro identifikaci sfér v pointcloudu bez použití předem připravených šablon. Není tedy nutné dopředu znát přesné rozměry hledané sféry. Přes použití metody RANSAC silně závisející na náhodě předčila metoda CGA očekávané výsledky. Tato metoda se ukázala jako efektivní pro budoucí využití v průmyslových aplikacích a robotice.

Jelikož CGA obsahuje pouze sféry, plochy a body, není tato algebra vhodná pro popis obecného objektu. V reálném použití je tedy vhodné ji kombinovat s jinými metodami. Další výzkum může směřovat na nalezení náhrady metody RANSAC, závislé na náhodném výběru bodu z pointcloudu, a vyvinout metodu předzpracování pointcloudu, na kterou nebyla tato práce zaměřena.

Literatura

[1] *Automated Bin Picking* [online]. Photoneo [cit. 2020-10-05]. Dostupné z: https://www.photoneo.com/bin-picking/.

[2] *Estimating Surface Normals in a PointCloud* [online]. Point Cloud Library [cit. 2020-06-15]. Dostupné z: https://pcl.readthedocs.io/projects/tutorials/en/latest/normal_estimation.html.

[3] *Fast Point Feature Histograms (FPFH) descriptors* [online]. Point Cloud Library [cit. 2020-06-15]. Dostupné z: https://pcl.readthedocs.io/projects/tutorials/en/latest/fpfh_estimation.html.

[4] *PhoXi 3D Scanner M* [online]. Photoneo [cit. 2020-05-23]. Dostupné z: https://www.photoneo.com/products/phoxi-scan-m/.

[5] *PhoXi 3D scanners family* [online]. Photoneo [cit. 2020-05-23]. Dostupné z: https://wiki.photoneo.com/index.php/PhoXi_3D_scanners_family.

[6] Anandan, T. M. *Robotic Bin Picking – The Holy Grail in Sight* [online]. 2016 [cit. 2020-10-05]. Dostupné z: https://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Insights/Robotic-Bin-Picking-The-Holy-Grail-in-Sight/content_id/6002.

[7] Arsenovic, A., Hadfield, H., Wieser, E., Kern, R. a The Pygae Team. *Pygae/clifford v1.3.0dev2*. Zenodo, březen 2020. DOI: 10.5281/zenodo.3724677. Dostupné z: https://doi.org/10.5281/zenodo.3724677.

[8] Bell, T., Li, B. a Zhang, S. Structured Light Techniques and Applications. Wiley Online Library. Feb 2016. DOI: 10.1002/047134608X.W8298. Dostupné z: https://onlinelibrary.wiley.com/doi/full/10.1002/047134608X.W8298.

[9] Čermák, J. *Metody 3D skenování objektů*. Brno, CZ, 2015. Bakalářská práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=103850.

[10] Day, J. *Using k-d trees to efficiently calculate nearest neighbors in 3D vector space*. Feb 2018. Dostupné z: https://blog.krum.io/k-d-trees/.

[11] Fischler, M. A. a Bolles, R. C. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Commun. ACM*. New York, NY, USA: Association for Computing Machinery. červen 1981, sv. 24, č. 6, s. 381–395. DOI: 10.1145/358669.358692. ISSN 0001-0782. Dostupné z: https://doi.org/10.1145/358669.358692.

[12] Geng, J. Structured-light 3D surface imaging: a tutorial. *Advances in Optics and Photonics*. OSA. Jun 2011, sv. 3, č. 2, s. 128–160. DOI: 10.1364/AOP.3.000128. Dostupné z: http://aop.osa.org/abstract.cfm?URI=aop-3-2-128.

[13] Herák, P. *Rozpoznání objektů a jejich polohy s využitím 3D modelů objektů*. Ostrava, CZ, 2017. Diplomová práce. Vysoká škola báňská - Technická univerzita Ostrava. Fakulta elektrotechniky a informatiky. Dostupné z: http://hdl.handle.net/10084/119001.

[14] Hildenbrand, D. *Foundations of Geometric Algebra Computing*. Springer, 2013. ISBN 978-3-642-31794-1.

[15] Kovacovsky, T. Parameters of 3D sensing techniques in a nutshell. Photoneo. 2017. Dostupné z: https://www.photoneo.com/technology/.

[16] Kratky, A. PhoXi 3D Scanner in a Bin Picking Application. Photoneo. 2018. Dostupné z: https://www.photoneo.com/technology/.

[17] Labaj, T. *Detekce křivek v obraze*. Brno, CZ, 2009. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství. Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=116823.

[18] Lachat, E., Macher, H., Landes, T. a Grussenmeyer, P. Assessment and Calibration of a RGB-D Camera (Kinect v2 Sensor) Towards a Potential Use for Close-Range 3D Modeling. *Remote Sensing*. Říjen 2015, sv. 7, s. 13070–13097.

[19] Owen Hill, A. *Robot Vision vs Computer Vision: What's the Difference?* [online]. 2016 [cit. 2020-10-05]. Dostupné z: https://blog.robotiq.com/robot-vision-vs-computer-vision-whats-the-difference.

[20] Rusu, R. B., Blodow, N. a Beetz, M. Fast Point Feature Histograms (FPFH) for 3D registration. In: *2009 IEEE International Conference on Robotics and Automation*. 2009, s. 3212–3217.

[21] Rusu, R. B. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments*. Mnichov, GE, 2009. Dissertation. Institut für Informatik der Technischen Universität München. Dostupné z: http://mediatum.ub.tum.de/doc/800632/document.pdf.

[22] Sveier, A., Kleppe, A., Tingelstad, L. a Egeland, O. Object Detection in Point Clouds Using Conformal Geometric Algebra. *Advances in Applied Clifford Algebras*. Únor 2017. DOI: 10.1007/s00006-017-0759-1.

[23] Thomson, C. *Common 3D point cloud file formats & solving interoperability issues* [online]. [cit. 2020-10-05]. Dostupné z: https://info.vercator.com/blog/what-are-the-most-common-3d-point-cloud-file-formats-and-how-to-solve-interoperability-issues.

[24] Zhou, Q.-Y., Park, J. a Koltun, V. Open3D: A Modern Library for 3D Data Processing. *ArXiv:1801.09847*. 2018.