



metadata

Protokol o kontrole originality



webprotokol

Kontrolovaná práca

Citácia	Percento*
Porovnanie spracovania 3D dát analytickými metódami a neurónovými sieťami / autor Kocúr Maroš - školiteľ Kohút Miroslav, Ing. - oponent Trebuľa Marek, Ing. - FEI / ÚRK (FEI). - Bratislava, 2023 <i>plagiD: 1789097 typ práce: bakalárska zdroj: STU.Bratislava</i>	1,96%

* Číslo vyjadruje percentuálny podiel textu, ktorý má prekryv s indexom prác korpusu CRZP. Intervaly grafického zvýraznenia prekryvu sú nastavené na [0-20, 21-40, 41-60, 61-80, 81-100].

Zhoda v korpusoch: Korpus CRZP: 1,32% (1418), Internet: 1,49% (170), Wiki: 1,39% (59), Slov-Lex: 0,00% (0)

Informácie o extrahovanom texte dodanom na kontrolu

Dĺžka extrahovaného textu v znakoch: 57965

Celkový počet slov textu: 7565

Počet slov v slovníku (SK, CZ, EN, HU, DE): 4867

Pomer počtu slovníkových slov: 64,3%

Súčet dĺžky slov v slovníku (SK, CZ, EN, HU, DE): 33559

Pomer dĺžky slovníkových slov: 57,9%

Interval	100%-70%	70%-60%	60%-50%	40%-30%	30%-0%
Vplyv na KO*	žiadny	malý	stredný	veľký	zásadný

* Kontrola originality je výrazne ovplyvnená kvalitou dodaného textu. Slovníkový test vyjadruje mieru zhody slov kontrolovanej práce so slovníkom referenčných slov podporovaných jazykov. Nízka zhoda môže byť spôsobená: nepodporovaný jazyk, chyba prevodu PDF alebo úmyselná manipulácia textu. Text práce na vizuálnu kontrolu je na konci protokolu.









Početnosť slov - histogram

Dĺžka slova	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
Indik. odchylka	>>	=	=	>>	=	=	<<	<<	<<	<<	=	=	=	=	=	=	=	=	=	=	=	=	=

* Odchýlky od priemerných hodnôt početnosti slov. Profil početnosti slov je počítaný pre korpus slovenských prác. Značka ">>" indikuje výrazne viac slov danej dĺžky ako priemer a značka "<<" výrazne menej slov danej dĺžky ako priemer. Výrazné odchylky môžu indikovať manipuláciu textu. Je potrebné skontrolovať "plaintext"! Príveľa krátkych slov indikuje vkladanie oddelovačov, alebo znakov netradičného kódovania. Príveľa dlhých slov indikuje vkladanie bielych znakov, prípadne iný jazyk práce.

Práce s nadprahovou hodnotou podobnosti

Dok.	Citácia	Percento*
1	Fúzia údajov zo stereokamery a laserového snímača pre detekciu objektov v 2D priestore / autor Knaperek Patrik Anton - školiteľ Lučan Martin, Ing. - oponent Kohút Miroslav, Ing. - FEI / ÚRK (FEI). - Bratislava, 2022 <i>plagiD: 1750545 typ práce: bakalárska zdroj: STU.Bratislava</i>	1,14%
2	Plánovanie trajektórie pre autonómny monopost Formula Student / autor Mazúr Samuel - školiteľ Lučan Martin, Ing. - oponent Stanko Jaromír, Ing. - FEI / ÚRK (FEI). - Bratislava, 2021 <i>plagiD: 1710808 typ práce: bakalárska zdroj: STU.Bratislava</i>	1,14%

3	Lokalizácia mobilného robota s využitím laserového diaľkomeru / autor Ďemrovski Miroslav - školiteľ Dekan Martin, Ing., PhD. - oponent Sojka Adam, Ing. - FEI / ÚRK (FEI). - Bratislava, 2021 <i>plagID: 1709253 typ práce: bakalárska zdroj: STU.Bratislava</i>	1,14% 
4	Návrh núdzového brzdného systému pre autonómny monopost Formula Student / autor Andocs Kristian - školiteľ Lučan Martin, Ing. - oponent Babinec Andrej, doc., Ing., PhD. - FEI / ÚRK (FEI). - Bratislava, 2021 <i>plagID: 1709500 typ práce: bakalárska zdroj: STU.Bratislava</i>	1,14% 
5	Návrh a overenie metód neuroevolúcie / autor Forintová Jana - školiteľ Sekaj Ivan, prof., Ing., PhD. - oponent Banášová Dominika, Ing. - FEI / ÚRK (FEI). - Bratislava, 2021 <i>plagID: 1709621 typ práce: bakalárska zdroj: STU.Bratislava</i>	1,14% 
163	Vedecká práca z oblasti strojového učenia / autor Adamcová Kristína - školiteľ Pružinský Dominik, Ing. - oponent Marák Pavol, Ing., PhD. - FEI / ÚIM (FEI). - Bratislava, 2023 <i>plagID: 1788671 typ práce: bakalárska zdroj: STU.Bratislava</i>	0,85% 
164	Segmentácia významných oblastí na obrazoch očnej siete / autor Pirhala Matej - školiteľ Goga Jozef, Ing. - oponent Pavlovičová Jarmila, prof., Ing., PhD. - FEI / ÚRK (FEI). - Bratislava, 2020 <i>plagID: 1668441 typ práce: bakalárska zdroj: STU.Bratislava</i>	0,79% 
743	Logistika výroby v knižnici Logistics Library / autor Gonšenica Maroš - školiteľ Duchoň František, prof., Ing., PhD. - oponent Körösi Ladislav, doc., Ing., PhD. - FEI / ÚRK (FEI). - Bratislava, 2023 <i>plagID: 1788970 typ práce: bakalárska zdroj: STU.Bratislava</i>	0,39% 
1114	http://theses.gla.ac.uk/246/1/2008bevanphd.pdf / Stiahnuté: 29.05.2017; Veľkosť: 417,37kB. <i>plagID: 35870101 zdroj: internet/intranet</i>	0,36% 
1647	http://tesi.cab.unipd.it/41544/1/thesis_bertan.pdf / Stiahnuté: 11.10.2017; Veľkosť: 137,43kB. <i>plagID: 38598425 zdroj: internet/intranet</i>	0,26% 

* Číslo vyjadruje percentuálny prekryv testovaného dokumentu len s dokumentom uvedeným v príslušnom riadku.

: Dokument má prekryv s veľkým počtom dokumentov. Zoznam dokumentov je krátený a usporiadaný podľa percenta zostupne. Celkový počet dokumentov je [1647]. V prípade veľkého počtu je často príčinou zhoda v texte, ktorý je predpísaný pre daný typ práce (položky tabuliek, záhlavia, poďakovania). Vo výpise dokumentov sa preferujú dokumenty, ktoré do výsledku prinášajú nový odsek (teda dokumenty ktoré sú plne pokryté podobnosťami iných dokumentov sa v zozname nenachádzajú. Pri prekročení maxima počtu prezentovateľných dokumentov sa v zarážke zobrazuje znak ∞.

Detaily - zistené podobnosti

1. odsek :	spoľahlivosť [100%]
ANOTÁCIA SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE FAKULTA ELEKTROTECHNIKY A INFORMATIKY Študijný program: Autor: Bakalárska práca: Vedúci záverečnej práce: Miesto a rok predloženia práce: Robotika a kybernetika Maroš Kocúr Porovnanie spracovania 3D dát metódami a neurónovými sieťami Ing. Miroslav Kohút Bratislava 2023 analytickým Práca sa zameriava na porovnanie algoritmu RANSAC a rozšírenie algoritmu o ne [«743]u	
2. odsek :	spoľahlivosť [95%]
[164»] kužele. Kľúčové slová: RANSAC, PCL, neurónová sieť ABSTRACTÚ SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY Study Programme: Author: Bachelor's thesis: Supervisor: Place and year of submission: Robotics and cybernetics Maroš Kocúr Comparison of 3D Data Processing by Analytical[«164]	
3. odsek :	spoľahlivosť [68%]
[163»] algoritmy 47 D Návod na spúšťanie algoritmov 48 Zoznam obrázkov a tabuliek Obrázok 2.1 Obrázok 4.1 Obrázok 5.1 Obrázok 5.2 Obrázok 5.3 Obrázok 5.4 Obrázok 6.1 Obrázok 7.1 Obrázok 7.2 Obrázok 7.3 Obrázok 8.1 Obrázok 8.2 Obrázok 8.3 Obrázok C.1 Obrázok C.2 Príklad neurónovej siete [3] Priklad algoritmu v 2D rovine [«163]	
4. odsek :	spoľahlivosť [94%]
[1647»] Dostupné tiež z: https://pointclouds.org/ . 5. RADU BOGDAN RUSU, Steve Cousins (zost.). Point cloud library (pcl): 3D is here. 2011 IEEE international conference on robotics a automation: IEEE. Dostupné tiež z: https://scholar.google.com/citations?view_op=view_citation& [«1647]	
5. odsek :	spoľahlivosť [61%]
[1114»] "VERSION 0.7" << std::endl << "FIELDS x y z rgb" << std :: endl << "SIZE 4 4 4 4" << std :: endl << "TYPE F F F U" << std::endl << "COUNT 1 1 1 1" << std :: endl << "WIDTH 512" << std::endl << "HEIGHT 424" << std :: endl << "VIEWPOINT 0 0 0 1 0 0 0" << std::endl ; myfile << "POINTS 217088" << std::endl << "DATA ascii" << [«1114]	

Plain text dokumentu na kontrolu

Skontroluje extrahovaný text práce na konci protokolu! Plain text (čistý text - extrahovaný text) dokumentu je základem pro textový analyzátor. Tento text může být poškozený úmyslně (vkládáním znaků, používáním neštandardních znakových sad, ...) alebo neúmyslně (napr. při konverzii na PDF nekvalitním programem). Nepoškozený text je čitelný, slova sú správně oddelené, diakritické znaky sú správně, množstvo textu je primeraný rozsahu práce. Při podozření na poškozený text (většího rozsahu), je potřebné práci na kontrolu originality zaslat' opakovaně pod rovnakým CRZPID.

ANOT[743]ÁCIA

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program: Autor: Bakalárska práca:

Vedúci záverečnej práce: Miesto a rok predloženia práce:

Robotika a kybernetika Maroš Kocúr Porovnanie spracovania 3D dát metódami a neurónovými[743]siet'ami Ing. Miroslav Kohút Bratislava 2023 analytickým

Práca sa zameriava na porovnanie algoritmu RANSAC a rozšírenie algoritmu o neurónovú sieť. Algoritmy sa porovnávali na 3D dátach pomocou knižnice Point Cloud Library a rozšírenie bolo realizované pomocou implementácie SPFN (Supervised Fitting of Geometric Primitives to 3D Point Clouds). Výstupy práce sú opisy jednoduchých telies ako sú valce, gule,[164]kužele.

Kľúčové slová: RANSAC, PCL, neurónová sieť

ABSTRACTÚ

SLOVAK UNIVERSITY OF TECHNOLOGY IN BRATISLAVA FACULTY OF ELECTRICAL ENGINEERING AND INFORMATION TECHNOLOGY

Study Programme: Author: Bachelor's thesis:

Supervisor: Place and year of submission:

Robotics and cybernetics Maroš Kocúr Comparison of 3D Data Processing by Analytical[164]Methods and Neural Networks Ing. Miroslav Kohút Bratislava 2023

This thesis focuses on the comparison of the RANSAC algorithm and the extension of the algorithm to a neural network. The algorithms were compared on 3D data using the Point Cloud Library and the extension was implemented using the SPFN (Supervised Fitting of Geometric Primitives to 3D Point Clouds) implementation. The outputs of the work are descriptions of simple solids such as cylinders, spheres, cones.

Keywords: RANSAC, PCL, neural network

Pod'akovanie

S hlbokou vďačnosťou by som sa chcel pod'akovať Ing. Miroslavovi Kohútovi za ochotu, nájdený čas a odborné vedenie počas riešenia bakalárskej práce. Taktiež by som sa chcel pod'akovať Ústavu robotiky a kybernetiky, za nadobudnuté informácie počas bakalárskeho štúdia, ktoré mi pomáhali riešiť problémy s prácou. D'alej by som sa chcel pod'akovať Ing. Michal Tölgyessy, PhD. za poskytnutie h'lbkovej kamery, s ktorou som mal možnosť pracovať.

Obsah

Úvod

10

1 Point Cloud 1.1 General 1.2 Použitie 1.3 Konverzia do 3D povrchu

11 11 11 11

2 Neurónové siete 2.1 História 2.2 Čo je Neural Network (neurónové siete) (NN)

.. 2.3 Vývoj

12 12 12 12

3 Point Cloud Library 3.1 Použitie 3.2 Pred inštaláciou 3.3

Inštalácia PCL 3.4 Nastavovanie parametrov Random sample consensus (RANSAC)

14 14 14 14 15

4 RANSAC 4.1 Overview 4.2 Algoritmus 4.3 RANSAC 3D ...

..... 4.4 Neural guided RANSAC

17 17 17 18 18

5 KINECT 5.1 Ako funguje Kinect 5.2 Výstup z KINECTU 5.2.1 List

potrebných knižnic pre Ubuntu 22.04 LTS 5.3 OpenNI 2.0 5.4 Point cloud (PCD) súbor ..

..... 5.5 Výstup z Kinectu

20 20 20 21 21 21 23

6 Implementovanie RANSAC algoritmu 6.1 PointCloudData 6.2 Zvolenie geometrického modelu

..... 6.3 Definovanie parametrov

24 24 24 24

6.4 Implementovanie RANSAC 24

7 Implementovanie RANSAC algoritmu s použitím neurónovej siete 7.1 ngransac 7.2 ngdsac

cameraloc 7.3 SPFN 7.4 Efektívny RANSAC

.....

26 26 28 29 31

8 Výstupy 8.1 Vyhodnotenie výsledkov PCL 8.2 Vyhodnotenie algoritmu SPFN 8.3

Porovnanie výsledkov

33 33 35 37

Záver

40

Zoznam použitej literatúry

42

Prílohy

44

A Kód na získanie Point Cloud (mrac'no bodov) (PCD) z Kinectu

45

B Programy

46

C Vstupné data pre [163] algoritmy

47

D Návody na spúšťanie algoritmov

48

Zoznam obrázkov a tabuliek

Obrázok 2.1 Obrázok 4.1 Obrázok 5.1 Obrázok 5.2 Obrázok 5.3 Obrázok 5.4 Obrázok 6.1 Obrázok 7.1 Obrázok 7.2

Obrázok 7.3 Obrázok 8.1 Obrázok 8.2 Obrázok 8.3 Obrázok C.1 Obrázok C.2

Príklad neurónovej siete [3] Príklad algoritmu v 2D rovine [«163] Kinect v2 Kinect v2

..... Architektúra softwarového vývoja Príklad výstupný Point Cloud z h'lbkovej kamery [4] Aktualny

vystup z kinectu Grafické znázornenie rovnice 6.3 Výstup Neural guided RANSAC Snímka z h'lbkovej kamery Kinectu a snímka pretrasformovaná na heat mapu Výsledky pokrytia

primitív rôznymi metódami [12] Výstup algoritmu RANSAC v knižnici PCL Odchýlky aproximácie stredu objektu v

prázdnom priestore Odchýlky aproximácie stredu objektu položeného na podložke ... Vstupné mrač'no

bodov so šumom Vstupné mrač'no bodov bez šumu 13 17 20 22 23 25 27

28 31 35 38 39 47 47

Tabul'ka 7.1 Tabul'ka 7.2 Tabul'ka 7.3

Tabul'ka 7.4 Tabul'ka 8.1

Tabul'ka 8.2

Tabul'ka 8.3 Tabul'ka 8.4

Odhad základných matíc Premiestnenie kamery v priestore Presnosť určovania Supervised

Fitting of Geometric Primitives (SPFN) oproti Efektívnemu RANSAC Presnosť určovania SPFN oproti Efektívnemu RANSAC [13]

..... Presnosť vyhodnocovania PCL s premiestnením predmetu v priestore Presnosť vyhodnocovania PCL

na rôznych polomeroch telesa. * označuje teleso položené na rovine, bez * je iba teleso v priestore s rôznymi nastaveniami prahový rozsah (PH) .

..... Presnosť vyhodnocovania SPFN Presnosť vyhodnocovania SPFN na rôznych polomeroch telesa. * označuje

teleso položené na rovine, riadky bez označenia * sú iba telesá v priestore 27 29 30 31 33

34 36

37

8

Zoznam skratiek

NG-RANSAC Neural Guided Random sample consensus

NN Neural Network (neurónové siete)

PCD

Point Cloud (mrac'no bodov)

PH prahový rozsah

RANSAC

Random sample consensus

SPFN

Supervised Fitting of Geometric Primitives

9

Úvod

V tejto práci sme sa zameriavali na implementáciu algoritmu RANSAC a jeho rozšírenie pomocou neurónových sietí. Naše výskumy a experimenty

sa zamerali na analýzu 3D dát získaných z kamery Kinect. Pri týchto dátach sme sa snažili odhadnúť presné vlastnosti scény, no zistili sme, že

existujúce metódy a nástroje nedokázali presne odhadnúť objekty, kvôli zašumením a nepresne naskenovaným dátam z Kinectu.

Preto sme sa rozhodli vytvoriť vlastný dataset, ktorý sme použili na porovnanie dvoch metód: analytickou metódou pomocou knižnice PCL a

metódou SPFN. Knižnica PCL je populárna pre analýzu PCD, avšak algoritmus je zložitejší na používanie. Preto sme sa rozhodli implementovať a

testovať metódu SPFN, ktorá využíva neurónové siete na lepšie spracovanie a porozumenie našim 3D dátam.

V našej práci sme tiež popísali ďalšie dve metódy, ktoré sme testovali, ale neprejavili sa ako vhodné pre naše potreby. Prvá metóda sa zaoberala

hl'adaním podobností medzi dvomi obrázkami zachytenými na rovnakej scéne, pričom bola presunutá kamera. Druhá metóda bola rozšírením

prvej metódy, pričom sme sa snažili odhadnúť presun kamery v priestore. Bohužiaľ, zložitosť vstupného datasetu nám neumožnila úspešne

implementovať a otestovať túto metódu.

V tejto práci sa snažíme demonštrovať, že použitie algoritmu RANSAC a jeho kombinácia s neurónovými sieťami (konkrétne metódou SPFN)

môže priniesť vylepšené výsledky pre analýzu a spracovanie 3D dát. Naše experimenty a porovnania týchto metód poskytujú lepšie porozumenie

a výsledky pre našu konkrétnu doménu a typ dát.

Cieľom tejto práce je teda preskúmať a vyhodnotiť výkonnosť algoritmu RANSAC a jeho rozšírenia prostredníctvom neurónových sietí na

spracovanie 3D dát a porovnať tieto metódy s existujúcimi prístupmi, ako je napríklad analytickou metódou použitím knižnice PCL. Na základe

našich experimentov a výsledkov chceme poskytnúť odporúčania pre budúce výskumy a zlepšenie analýzy 3D dát v podobných oblastiach

aplikácií.

10

1 Point Cloud

1.1 General

Je to súbor bodov v priestore. Body môžu reprezentovať 3D tvary alebo objekty. Každý bod má karteziánske súradnice (X,Y,Z). PCD je generovaný pomocou 3D skenera alebo pomocou software na fotogrametriu, ktorý meria veľa bodov na externom povrchu objektov okolo. My v tomto projekte použijeme Kinect v2. PCD sa používa v 3D modelovaní, metrológii, meranie kvality výrobkov a rôzne vizualizácie. PCD sa často porovnáva s 3D modelmi alebo inými PCD ako registrácia množín bodov. [1]

1.2 Použitie

Pre priemyselnú metrológiu alebo inšpekciu pomocou priemyselnej počítačovej tomografie možno mracno bodov vyrobeného dielu zosúladiť s existujúcim modelom a porovnať, aby sa skontrolovali rozdiely. Geometrické rozmery a tolerancie možno získať aj priamo z PCD. [1]

1.3 Konverzia do 3D povrchu

V geografických informacných systémoch sú PCD jedným zo zdrojov využívaných na tvorbu digitálneho výškového modelu terénu. Používajú sa aj na generovanie 3D modelov mestského prostredia. Drony sa často využívajú na nazbieranie RGB obrázkov ktoré sa neskôr pomocou algoritmu strojového videnia ako je AgiSoft Photoscan, Pix4D alebo DroneDeploy používajú na vytvorenie RGB PCD, kde sa môže požiť vzdialenosť a objemová aproximácia. [1]

11

2 Neurónové siete

2.1 História

V posledných rokoch, najlepšie systémy s aplikáciou umelej inteligencie - ako sú rozpoznávací a reční v mobilných zariadeniach alebo automatické prekladacie majú dobré výsledky v technike nazývajúcej sa hĺbkové učenie. Hĺbkové učenie je v podstate iné meno pre aplikáciu umelej inteligencie s názvom neurónové siete, ktorý sa vyvíja takmer 80 rokov. NN boli prvýkrát navrhnuté v roku 1944, dvoma výskumníkmi z Chicagskej univerzity, ktorí v roku 1952 prešli na MIT a založili oddelenie kognitívnych vied. [2]

2.2 Čo je NN

NN sú myslené na robenie strojového učenia, v ktorom sa počítač učí vykonávať úlohy na základe analyzovania tréningových príkladov, ktoré sú zvyčajne ručne označené. Systém na rozpoznávanie objektov by mohol mať prístup k tisíciam obrázkov označených ako auto, dom, guľa a podobne. NN hľadá vizuálne charakteristiky v obrázku ktoré majú vzájomný vzťah s konkrétnym označením. Voľne modelovaná NN na ľudskom mozgu má tisíce až milióny jednoduchých neurónov, ktoré sú husto prepojené. Väčšina sietí sa dnes organizuje do vrstiev neurónov a tie posielajú údaje vpred, čo znamená že dáta nimi prechádzajú iba v jednom smere. Jeden neurón môže byť pripojený k viacerým neurónom vo vrstvách pod ňou, z ktorej príma dáta a viacej neurónov vo vrstve nad ňou kde spracované dáta posielajú. Ku každému vstupnému prípoju sa prideli číslo známe ako "váha". Ak je sieť aktívna, neurón prijme rôzne dáta z každého vstupu a vynásobí ich priradenou váhou, potom všetky výsledky sčíta ak je výsledne číslo pod hranicou prahovej hodnoty, neurón nepošle žiadne dáta do ďalšej vrstvy. Ak číslo prekročí prahovú hodnotu, tak neurón "vystrelí", čo znamená že pošle súčet vstupov na všetky výstupy.

Keď sa NN trénuje, všetky váhy a prahové hodnoty sú nastavené na náhodnú hodnotu. Tréningové dáta sú posielané do spodnej vstupnej vrstvy a prechádzajú cez nasledujúce vrstvy, násobia sa a sčítavajú sa zložitými cestami, pokiaľ radikálne transformované nedôjdu na vrchnú výstupnú vrstvu. Počas tréningu sa váhy a prahové hodnoty neustále upravujú kým tréningové údaje s podobnými vlastnosťami neprinášajú podobne výstupy. Bližšie popísanie NN je v sekcii s implementáciou, kde je popísané presnejšie používanie algoritmu a funkcie NN. [2]

2.3 Vývoj

Neuronové siete opísané v roku 1944 mali váhy a prahové hodnoty, ale neboli usporiadané do vrstiev a výskumníci nešpecifikovali tréningové mechanizmus. Výskumníci dokázali

12

ukázať, že NN dokáže v princípe vypocítať hocijakú funkciu ako digitálny počítač. Výsledkom bola viac nesurová veda ako počítačová a cieľom bolo poukázať, že ľudský mozog môže považovať za výpočtové zariadenie.[2]

Obr. 2.1: Príklad neurónovej siete [3]

13

3 Point Cloud Library

PCL je samostatný, vysoko škalovaliteľný, otvorený projekt pre 2D a 3D obrázky a spracovávanie PCD. Knižnica je na viac operačných systémov, napísaná v jazyku C++ a Python. Najčastejšie sa používa na operačnom systéme Linux. Existujú balíčky aj pre macOS a Windows vytvorené tretími stranami. My v tomto projekte budeme používať Ubuntu 22.04. LTS a verzia PCL je 1.12.1. Knižnica je vydaná pod BSD licenciami čo znamená, že je voľne použiteľná na komerčné účely a za účelom výskumu.[4]

3.1 Použitie

PCL je open-source knižnica s algoritmami pre PCD spracovanie úloh a spracovanie 3D geometrie, aká sa vyskytuje v trojdimenzionálnom strojovom videní. Knižnica obsahuje algoritmy na filtrovanie, odhady funkcie, rekonštrukcie povrchov, 3D registrácie, prispôbenie modelu, vyhľadávanie objektov a segmentácie. PCL má vlastný formát na ukladanie PCD dát PCD, ale podporuje načítanie a ukladanie dát do rôznych iných formátov.

Algoritmy sa používajú na vnímanie v robotike na filtrovanie zašumených dát, spájanie 3D dát, segmentovanie dôležitých častí v priestore, extrahovanie kľúčových bodov a výpočet deskriptorov na rozpoznávanie objektov vo svete na základe ich geometrického vzhľadu a vytváranie povrchy z PCD a vykresľovanie ich.[5]

3.2 Pred inštaláciou

PCL potrebuje niekoľko knižníc tretích strán na fungovanie, ktoré musia byť nainštalované. Väčšina matematických operácií je implementovaných v Eigen knižnici. Vizualizačný model pre PCD je na základe The Visualization Toolkit(VTK). Knižnica Boost je použitá na zdieľanie pointrov a FLANN knižnica pre rýchle hľadanie v okolí na základe algoritmu k-najbližších. Dá sa ale potrebné nainštalovať OpenNI 2 knižnicu, ktorá zabezpečí komunikáciu s Kinectom v2.[4]

3.3 Inštalácia PCL

PCL je dostupný na mnoho distribúcií Linuxu ako Ubuntu, Debian, Fedora, Gentoo a Arch Linux. PCL na distribúciách Ubuntu a Debian môžeme nainštalovať pomocou.

sudo apt install libpcl-dev Na Windowse sa PCL inštaluje pomocou vcpkg package manažéra vytvoreného Microsoftom.

PS> ./vcpkg install pcl MacOS má Homebrew package manažéra, ktorý podporuje inštaláciu package, ktoré

14

Apple alebo Linux nedokáže nainštalovať. brew install pcl Toto sú odporúčané inštalácie pre PCL na daných operačných systémoch. Na používanie PCL je potrebné v kóde vložiť potrebné knižnice. Po nainštalovaní sa v systéme nastavujú premenné, takže stačí použiť príkaz v C++, #include <pcl<názov_knižnice>.h>. Pre kompiláciu je potrebné vytvoriť súbor CMakeList.txt v ktorom zadefinujeme potrebné premenné aby make vedel nájsť cestu ku knižnici a vedel aké súbory má skompilovať. D'alej je potrebné vytvoriť priečinok s názvom build a v termináli v priečinku build vykonať príkaz cmake "cesta k CMakeList.txt", ktorý vytvorí makefile, ktorý skompiluje potrebné zdrojové kódy a vytvorí binárny súbor pomocou ktorého je možné spustiť projekt. [4]

3.4 Nastavovanie parametrov RANSAC

V kóde treba zadefinovať parametre, ktoré nastavujú algoritmus a odfiltrujú šum. Z pointcloudu sa najprv odfiltrujú body, ktoré ležia ďalej ako sú zadefinované pomocou príkazu. pass.setFilterLimits(0, 1.5) - zo vstupného PCD sa odstránia body ležiace 1.5 metra od kamery. D'alsí krok hľadá rovinu na ktorej sú objekty položené, pomocou algoritmu RANSAC, ktorý ma vstupné parametre matematický model roviny, počet opakovaní algoritmu, váhy normál a prahová vzdialenosť bodov ležiacich v ohraničení. Posledný krok je nájsť zvolený objekt. Algoritmu sa nastavujú nasledovné parametre: seg.setModelType(pcl::SACMODEL_CYLINDER) - matematický opis modelu ktorý algoritmus v PCD vyhľadáva seg.setMethodType(pcl::SAC_RANSAC) - algoritmus hľadania modelu seg.setNormalDistanceWeight(0.15) - nastavenie váh normál seg.setMaxIterations(10000) - počet opakovaní algoritmu seg.setDistanceThreshold(0.05) - prahová hodnota ohraničenia na určenie bodu ležiaceho v množine seg.setRadiusLimits(0.001, 1) - minimálny a maximálny rozmer telesa

15

Po nastavení parametrov a spustení algoritmu výstupom je vektor opisujúci teleso a rovinu. V práci je použitý algoritmus na vyhľadávanie valca, ktorý vracia sedem hodnôt opisujúcich objekt. [6]

• X-ová súradnica bodu ležiaca na osi objektu • Y-ová súradnica bodu ležiaca na osi objektu • Z-ová súradnica bodu ležiaca na osi objektu • X-ová súradnica bodu opisujúca smer osi • Y-ová súradnica bodu opisujúca smer osi • Z-ová súradnica bodu opisujúca smer osi • polomer Zo získaných parametrov pomocou kódu v prílohe cylinder_center.m, je možné vypočítať stred valca.

16

4 RANSAC

4.1 Overview

RANSAC je algoritmus vytvorený Fischlerom a Bollesom, určuje všeobecný prístup k odhadu parametrov, s veľkým podielom outliers v stupnom datasete. Na rozdiel od iných výkonných algoritmov odhadu, ako napríklad M-odhady a metódou najmenších štvorcov s prepojením na strojové učenie. RANSAC bol vytváraný komunitou ľudí používajúcich strojové učenie. RANSAC je vzorkovacia technika ktorá generuje kandidáty na minimálny počet pozorovaní potrebných na zistenie odhadu parametrov ležiacich pod modelom. Narozdiel od ostatných vzorkovacích algoritmov, ktoré používajú čo najviac bodov ako môžu, RANSAC používa najmenší počet bodov ako môže.

4.2 Algoritmus

1. Vybranie minimum náhodných bodov potrebných na určenie parametrov modelu 2. Vypočítanie parametrov pre model 3. Určenie koľko bodov z množiny všetkých bodov leží v preddefinovanom prahovom rozsahu. 4. opakuj kroky 1 až 3, s maximálnym N opakovaniami. 5. Vráti model s najväčším počtom bodov ležiacich v prahovej hodnote. [6]

Obr. 4.1: Příklad algoritmu v 2D rovine Modre sú body z datasetu, pomocou RANSAC algoritmu sa určili 2 body, ktoré majú vo svojom subsete najmenej bodov ležiacich mimo alfy.

17

4.3 RANSAC 3D

Implementácia RANSAC algoritmu na 3D dáta môže byť realizovaná nasledovne:

1. Príprava dát: Nacítaj 3D dáta vo formáte bodového oblaku, ktorý obsahuje 3D súradnice bodov.
2. Výber minimálnej sady: Náhodne vyberie minimálnu sadu bodov z celého bodového oblaku. Táto sada by mala obsahovať minimálny počet bodov potrebný na určenie modelu.
3. Výpočet modelu: Na základe vybranej sady bodov vypočíta model. To znamená, že vykoná výpočet, ktorý určuje parametre modelu, napríklad roviny alebo valca, ktorý sa snaží identifikovať.
4. Vyhodnotenie bodov ležiacich v hranicnej hodnote: Pre všetky body v bodovom oblaku vypočíta vzdialenosť od modelu. Body, ktoré majú vzdialenosť menšiu ako určený prah, považujú sa za bod ležiaci v prahovom rozsahu. Vypočíta počet body ležiace v prahovom rozsahu pre model.
5. Opakovanie krokov 2 až 4: Opakuje kroky výberu minimálnej sady, výpočtu modelu a vyhodnotenia počtu bodov ležiacich v prahovej hodnote určený počet krát alebo do dosiahnutia preddefinovaného kritéria ukončenia.
6. Výber konečného modelu: Vyberie model s najväčším počtom bodov ležiacich v prahovej hodnote ako konečný model.
7. Voliteľné: Ak je potrebné vylepšiť konečný model, je možné použiť všetky body ležiace v prahovej hodnote na presnejší odhad parametrov modelu pomocou metódy najmenších štvorcov.

Tieto kroky poskytujú základnú implementáciu RANSAC algoritmu pre 3D dáta. Je dôležité si uvedomiť, že konkrétna implementácia sa môže líšiť v závislosti od použitých knižníc alebo programovacieho jazyka. [4] [5]

4.4 Neural guided RANSAC

Pri implementácii neurónmi riadeného RANSAC (Neural Guided Random sample consensus (NG-RANSAC)) sa kombinuje algoritmus RANSAC s neurónovou sieťou, ktorá slúži na riadenie výberu minimálnych množín a zlepšenie robustnosti odhadového procesu. Nasleduje hrubý prehľad implementácie NG-RANSAC:

18

1. Príprava dát: Pripravenie datasetu, ktorý sa bude používať na tréning neurónovej siete. Tieto dáta zahrňajú vstupy vo forme korešpondencií medzi obrázkami a očakávané výstupy v podobe binárnych hodnôt, ktoré označujú, či je každá korešpondencia bodom ležiacim v prahovom rozsahu alebo odľahlým bodom.
2. Tréning neurónovej siete: Tréning neurónovej siete na predikciu pravdepodobnosti, že každá korešpondencia je bodom ležiacim v prahovom rozsahu. Vstupy do neurónovej siete by mali byť vlastnosti každej korešpondencie a výstupom by mal byť skalár, ktorý predstavuje predikovanú pravdepodobnosť.

3. Inkorporácia neurónovej siete do RANSAC: Potrebne modifikovať algoritmus RANSAC tak, aby využíval neurónovú sieť na odhad pravdepodobnosti, že každá korešpondencia je bodom ležiacim v prahovom rozsahu. Namiesto náhodného výberu podmnožín korešpondencií pre výpočet hypotéz modelu algoritmus použije odhadnuté pravdepodobnosti na riadenie výberu minimálnych množín.

4. Výpočet konečného modelu: Po výbere minimálnych množín je potrebné použiť tieto množiny na výpočet hypotéz modelu a vyhodnotiť každú hypotézu podľa počtu bodov ležiacich v prahovom rozsahu. Vyberie hypotézu s najväčším počtom bodov ležiacich v prahovom rozsahu ako konečný model.

5. Vylepšenie modelu: Voliteľne môžete vylepšiť konečný model pomocou všetkých inlierov a odhadu parametrov modelu pomocou metódy najmenších štvorcov.

Implementácia neurónmi riadeného RANSAC vyžaduje určitú znalosť strojového učenia a počítačového videnia. Existuje množstvo vedeckých prác a dostupných implementácií s otvoreným zdrojovým kódom, ktoré poskytujú podrobnejšie informácie o implementácii NG-RANSAC.[7]

19

5 KINECT

Kinect je vstupné zariadenie snímajúce pohyb vyrobené Microsoftom. Zariadenie obsahuje RGB kamery a infračervený projektor a detektor ktorý monitoruje hĺbku priestoru na základe štrukturovateľného svetla alebo na základe času trvajúceho svetlu dopadnúť na objekt, vďaka ktorému vie Kinect poskytnúť rekognizáciu gest v reálnom čase. Kinect sa používa hlavne v hernom priemysle, ale používa sa taktiež na komerčné a akademické účely pretože poskytuje mapovanie priestoru a je lacnejší než profesionálne zariadenia.[8]

Obr. 5.1: Kinect v2

5.1 Ako funguje Kinect

Infračervený projektor na kinecte posiela modulované infračervené svetlo ktoré je zachytené senzory. Infračervené svetlo ktoré sa odrazí od bližších objektov má kratší čas letu ako svetlo ktoré sa odrazí od vzdialenejších objektov, takže senzor sníma ako vymodelovaný vzor bol deformovaný z času letu svetla, pixel po pixeli. Čas priletu meranej hĺbky touto metódou môže byť presnejšie vypočítaný v kratšom čase, čo zabezpečí viac snímkov za sekundu. Hneď ako kinect naskenuje hĺbkovú fotografiu, použije metódu zisťovania hrán k vytvoreniu bližších objektov z pozadia fotky.[8]

5.2 Výstup z KINECTU

Na získanie výstupu z Kinectu je potrebné si nainštalovať Open-source knižnicu LibFreenect2, ktorá je určená na získavanie videí z Kinectu verzie 2. Knižnica disponuje

20

vzorovým kódom, ktorý po spustení otvorí prehliadač a okno rozdelí na štyri časti a v nich je možné sledovať výstupy Infra Red kamery, farebnej kamery a depth kamery v reálnom čase. Nato aby sa knižnica mohla používať je potrebné doinštalovať potrebné knižnice a súčasti. Na získanie snímky z Kinectu je potrebné spustiť Python program, ktorý vytvorí point cloud súbor, v ktorom budeme hľadať požadované tvary, alebo modifikovať knižnicu libfreenect2 na ukladanie získaných PCD.

5.2.1 List potrebných knižníc pre Ubuntu 22.04 LTS

- libusb
- TurboJPEG
- OpenGL
- OpenCL (dobrovoľne)
- CUDA (dobrovoľne, iba grafická karta od NVIDIA)
- VAAPI (dobrovoľne)
- OpenNI2

5.3 OpenNI 2.0

Open-source framework vytvorený spoločnosťou PrimeSense pre 3D Natural Interaction senzory od spoločnosti napríklad Asus Xtion, spoločnosť stojí za lincencovaným dizajnom hardwaru a čipov použitých priamo v Kinecte. Druhá verzia OpenNI sa oproti prvej má znížiť zložitosť API tým že zložité funkcie sa presunuli do middlewaru (knižnice na komunikáciu medzi zariadeniami) a aby funkcie na komunikáciu so senzormi boli jednoduchšie na pochopenie a zároveň má podporu pre generáciu Kinectu v2. OpenNI2 poskytuje prácu so surovými dátami z Kinectu. Rôzne vyššie funkcie ako sú spracovanie gest, detekcie a sledovanie pohybov, rozpoznávanie tvarov je potrebné rozšíriť middleware, alebo použiť Microsoft SDK, ale nakoľko táto publikácia je spracúvaná na operačnom systéme Linux je potrebné použiť OpenNI2 SDK, ktoré by malo pracovať aj s inými druhmi kamier.[9]

5.4 Point cloud (PCD) súbor

Pomocou OpenNI2 implementovaného v programovacom jazyku C++ a knižnicou PCL je nadviazaná komunikácia s Kinectom, ktorý streamuje dáta a sú vyobrazené v okne OpenNI2 a po úprave knižnice libfreenect s implementovaným opencv, sa získane dáta ukladajú ako PCD súbor potrebný na vyhládavanie tvarov pomocou RANSAC algoritmu.[9]

21

Obr. 5.2: Architektúra softwarového vývoja

Obr. 5.3: Príklad výstupný Point Cloud z hĺbkovej kamery [4] 22

5.5 Výstup z Kinectu

Kinect má horšie vzorkovanie oproti použitému PCD. Algoritmy mali problém odhadnúť objekty z získanej scény, preto boli zvolené dáta v práci namodelované a taktiež na namodelovaných dátach bol pridaný šum, na čo najpresnejšiu kópiu reálneho sveta.

Obr. 5.4: Aktualný výstup z kinect

23

6 Implementovanie RANSAC algoritmu

6.1 PointCloudData

Pre praktickejšiu prácu s PCD na začiatku odstránia všetky body ktoré sú vzdialenejšie ako 3,5 metra a odhadneme normály povrchu v každom bode. Odfiltrovaný model je uložený v samostatnom PCD súbore. [10]

6.2 Zvolenie geometrického modelu

Vyberieme si geometrický model, čo bude zodpovedať trom náhodným bodom aby sme vedeli zostrojiť rovinu. V blízkosti roviny vo vopred definovanom prahovej hodnote spočítame koľko bodov leží v blízkosti roviny. Vyberieme rovinu ktorá má najväčší počet bodov ležiacich v prahovej hodnote ako najlepší model. Postup sa opakuje pokiaľ neprejde algoritmus presný

pocet interácií zvolených na začiatku. V kóde je zvolený model valca ktorý ma v 3D priestore predpis

$$(y - z)^2 + (z - x)^2 + (x - y)^2 = 3R^2$$

(6.1)

$$(f(x - a/c \cdot z, y - b/c \cdot z) = 0)$$

(6.2)

kde konštanty a,b,c sú zložky normálového vektora , ktorý je kolmý na rovinu a ktorýkoľvek rovnobežný vektor s rovinou. Z toho nám vyplýva, že bod $p=(x,y,z)$ patrí do roviny vektora \vec{n} ak spĺňa rovnicu. [10]

6.3 Definovanie parametrov

V kóde sa na začiatku nastavili požadované hodnoty, čiže hľadaný model v kóde nastavený na valec, vzdialenosť bodov pre vyhodnotenie RANSAC algoritmu prahový rozsah, vplyv normál a nastavili veľkosť hľadaného objektu. Hodnôt treba na začiatku otestovať viacero a vybrať ktoré majú najväčšiu presnosť alebo rýchlosť. Je možné otestovať automatické nastavovanie parametrov, čo by teoreticky mohlo fungovať na základe vypočítania strednej hodnoty vzdialenosti medzi susediacimi bodmi. [10] [6]

6.4 Implementovanie RANSAC

- Hľadanie roviny(1 opakovanie) Najprv sa zvolia 3 body z PCD, z ktorých sa má vytvoriť rovnica roviny nájdením parametrov a,b,c,d. K nim sa dá dopracovať lineárnou algebrou, konkrétne krížovým súčinom dvoch vektorov ktoré generujú ortogonálny vektor. Treba definovať vektory z rovnakého bodu v rovine a vypočítať normálu k nim, ktorá definuje normálu roviny. Pomocou normál normalizujeme náš normálový vektor a získame parametre a,b,c a

24

dopočítame parameter d čo je posun roviny od počiatku.

- Bod k rovine: definovanie prahového rozsahu

D

=

$$|\vec{n} \cdot \vec{w}| / |\vec{n}|$$

=

$$ax_i + by_i + cz_i + a^2 + b^2 + c^2$$

d

(6.3)

D reprezentuje dĺžku priemetu \vec{w} na jednotkový vektor \vec{n} . Pomocou 6.3 je možné vypočítať, či bod so súradnicami x,y,z leží v prahovej hodnote. Vektor \vec{n} reprezentuje normálu k rovine. Čiže vzdialenosť bodu P od roviny reprezentuje absolútny súčin vektorov \vec{n} a \vec{w} .

Obr. 6.1: Grafické znázornenie rovnice 6.3 • Opakovanie Vytvorenie cyklu ktorý sa bude opakovat' počet interácií a porovnávať, či aktuálny model je lepší než doteraz najlepší model a výstupom bude najlepší RANSAC model. [10]

25

7 Implementovanie RANSAC algoritmu s použitím neurónovej siete

NG-RANSAC, je flexibilný algoritmus, ktorý vytvára robustné odhady prispôbením parametrických modelov súborom údajov, ktoré môžu obsahovať šum alebo odľahlé hodnoty. Na určenie pravdepodobnosti výberu každého dátového bodu využíva NG-RANSAC neurónovú sieť. RANSAC potom používa tieto informácie na usmernenie výberu minimálnych množín s cieľom poskytnúť modelové hypotézy. Podobne ako pri konvencínom RANSAC je konečný model definovaný počtom odľahlých hodnôt.

7.1 ngransac

Prvý test bol vykonaný na implementácii NG-RANSAC, vstupom pre test boli dva 2D snímky s rôzneho uhla pohľadu. Výstupnými dátami bol 7.1, ktorý bol konvertovaný do šedej pre rozšírenie údajov na a nájdený model s počtom bodov ležiacich v množine. [7]

Model nájdený pomocou RANSAC:

$$6.418e-7 \ 1.227e-1 \ -3.079e-2$$

$$-7.301e-2 \ -2.326e-2 \ 7.021e-1$$

$$2.113e-2 \ -6.957e-1 \ -2.695e-2$$

Počet bodov ležiacich v množine: 251.

Model nájdený pomocou NG-RANSAC:

$$-3.409e-3 \ 3.847e-2 \ -4.590e-2$$

$$2.083e-2 \ 4.066e-3 \ 7.053e-1$$

$$3.740e-2 \ -7.051e-1 \ 4.446e-2$$

Počet bodov ležiacich v množine: 297.

26

Obr. 7.1: Výstup Neural guided RANSAC

Na 7.1 ľavá dvojica predstavuje model nájdený pomocou RANSAC algoritmu a pravá dvojica model nájdený pomocou NG-RANSAC. NG-RANSAC vyhládaval podobnosti s vyššou presnosťou, čiže na testovacích dátach našiel väčší počet bodov ležiacich v prahovom rozsahu.

Tabuľka 7.1: Odhad základných matíc

Cieľ tréovania %bodov F-skóre Priemerná Stredná

RANSAC

v PH

-

$$21.85 \ 13.84 \ 0.35$$

0.32

USAC Deep F-Mat NG-RANSAC

Priemerná Priemerná

$$21.43 \ 24.61 \ 25.05$$

$$13.90 \ 14.65 \ 14.76$$

$$0.35 \ 0.32 \ 0.32$$

$$0.32 \ 0.29 \ 0.29$$

NG-RANSAC

F-score

24.13 14.72 0.33

0.32

NG-RANSAC %bodov v PH 25.12 14.74

0.32

0.29

27

%Bodov v PH je samostatný tréningový objekt. F-skóre značí body ležiace v PH základnej pravdy.

7.2 ngdsac cameraloc

Táto práca vychádza z hotovej implementácie NN k sekcii 7.1. Pomocou práce sa zist'ovala presnosť určenia podobnosti na 2D dátach použitím RANSAC algoritmu a jeho vylepšenia pomocou NN. Súčasťou práce bolo aj zisťovanie polohy kamery, ktorá mala byť implementovaná. Použitý bol dataset z University of Cambridge cite dac' o, ktorý poskytuje zábery na budovy na univerzite. Dataset však neposkytuje reálnu polohu kamery, aby bol dataset použiteľný pri porovnaní 3D dát potrebných pre túto prácu. Cieľom implementácie bolo vytvoriť dataset z Kinectu z h'lbkovej kamery ktoré sme pre lepšiu čitateľnosť pretransformovali na heat mapu 7.2, s reálnou polohou kamery, ale nebolo možné s vytvoreným datasetom natrénovať NN, pretože NG-RANSAC používala Structure from Motion pipeline, ktorá vytvorila súbor typu .nvm s približne odhadnutou polohou kamery a ďalšími neznámymi dátami, ktoré nebolo možné zreprodukovať, čiže sa nepodarilo natrénovať sieť na 2D dátach. Dáta sa používali na nastavovanie Pythonovskej knižnice tensorflow. [11] [7]

Obr. 7.2: Snímka z h'lbkovej kamery Kinectu a snímka pretransformovaná na heat mapu

Práca porovnáva výsledky diferenciálneho RANSACU (DSAC) a jeho rozšírenie s použitím neurónovej siete (NG-DSAC). Obe siete boli trénované rovnakým spôsobom ale pri NG-DSAC, bola aktivovaná vetva pravdepodobnosti. Diferenciálny RANSAC funguje na pozorovaniach, a nie na základe váh ako klasický RANSAC.

Z tabuľky 7.2 možno vidieť chybu strednej hodnoty odhadu pozície použitia algoritmov. Z výsledkov vyplýva, že diferenciálny RANSAC rozšírený o neurónovú sieť na väčšine datasetu určoval presnejšie polohu kamery.

28

Tabuľka 7.2: Premiestnenie kamery v priestore

DSAC++(VGGNet) DSAC++(ResNet) NG-DSAC++(ResNet)

Great Court

40.3cm

40.3cm

35.0cm

Kings College

17.7cm

13cm

12.6cm

Old Hospital

19.6cm

22.4cm

21.9cm

Shop Facade

5.7cm

5.7cm

5.6cm

St M. Church

12.5cm

9.9cm

9.8cm

7.3 SFPN

Autorský tím sa v tomto článku zameriava najmä na problémy s prístupmi založenými na RANSAC, ktoré sú priemyselným štandardom pre montáž primitív, ale vyžadujú dôkladné ladenie parametrov pre každý vstup a ťažko sa škálujú pre veľké súbory údajov s rôznorodými formami. Riešením týchto problémov, ktoré autori navrhujú, je Supervised Primitive Fitting Network (SPFN), koncová neurónová sieť schopná robustne detegovať premenlivé množstvo primitív v rôznych mierkach bez akejkoľvek ľudskej kontroly. Primitívne povrchy a príslušnosť primitív k vstupným bodom slúžia ako základná pravda pre tréningovanie tejto siete. Návrh využíva modul odhadu modelu na výpočet typu a parametrov primitív namiesto jednoduchého priameho predpovedania primitív. Namiesto toho najprv predpovedá vlastnosti bodov.

Tím preukázal výrazné zlepšenie v porovnaní so špic'kovými metódami s použitím RANSAC a priamym neurónovým predpovedaním pomocou tréningu a hodnotenia navrhovanej metódy s použitím ich nových súborov údajov, 3D modelov mechanických komponentov ANSI.

Tím testoval sieť pri spracovaní vstupných mrac'ien bodov s vysokým rozlíšením rozšírených o šum pri testovaní, hoci sieť bola vycv'čená na mrac'ách bodov s nižším rozlíšením. Tieto testy ukázali, že SPFN funguje efektívne aj za týchto okolností. [12] [7]

29

Tabuľka 7.3: Presnosť určovania SPFN oproti Efektívnemu RANSAC

Metóda

Seg. Primitive Point Primitive

(Mean IoU) Type (%) Normal (°) Axis (°)

Eff.RAN SAC + J Eff.f.RAN SAC + J Eff.f.RAN SAC + J Eff.f.RAN SAC + J + W^ + t^ Eff.f.RAN SAC + N^ + W^ + t^

43.68 56.07 45.9 69.91 60.68 60.56

52.92 43.90 46.99 60.56 92.76 93.13

11.42 6.92 6.87 6.87 6.87 8.15

7.54 2.42 5.85 2.90 6.21 7.02

DP P N (Sec.4.4)

44.05

51.33

-

3.68

SP F N – Lseg SP F N – Lnorm + J SP F N – Lres SP F N – Laxis SP F N ($t^* - > Est.$)

41.61 71.18 72.70 77.31 75.71

92.40 95.44 96.66 96.47 95.95

8.25 6.87 8.74 8.28 8.54

1.70 4.20 1.87 6.27 1.71

SP F N

77.14

96.93

8.66

1.51

V tabuľke 7.3 a 7.4 sú uvedené výsledky algoritmu SPFN v porovnaní s efektívnym RANSAC. Dáta sú testované na identických PCD s rozlíšením 8k bodov riadok 1 v tabuľkách 7.3 a 7.4 a hviezdíčka označuje PCD s vysokým rozlíšením 64k bodov s rušením riadok 2 v tabuľkách 7.3 a 7.4.

Algoritmus SPFN prekonal efektívny RANSAC vo všetkých meraných metrikách. Obe Sk a P prekrytie s prahovou hodnotou $\epsilon = 0.01$ vykazali veľké rezervy, čo ukazuje nato že algoritmus SPFN presnejšie vyhodnocuje primitívne tvary. Efektívny RANSAC testujú aj s vlastnosťami predpovedanými pomocou algoritmu SPFN. Natrénovali SPFN so stratou Lseg a pre každý segment v predpovedanej matici príslušnosti W^* použili efektívny RANSAC na predpovedanie primitívnej vlastnosti riadok 4 v tabuľkách 7.3 a 7.4. Po pridaní Ltype a Lnorm strát pri tréningu za sebou a použili predpovedané primitívne typy t^* a body normál N^* v efektívnom RANSACU riadok 5 a 6 v tabuľke 7.3 a 7.4. Kde vstupne PCD je prvý segment pre efektívny RANSAC s NN, obe Sk a P pokrytie

pre efektívny RANSAC sa podstatne zvýšilo, ale stále nižšie ako SPFN metóda. Predpovedané normály a primitívne typy neurónovou sieťou nezlepšili Sk a P pokrytie v RANSAC.[12]

30

Tabuľka 7.4: Presnosť určovania SPFN oproti Efektívnemu RANSAC [13]

Metóda

Eff.RAN SAC + J Eff.RAN SAC + J Eff.RAN SAC + J Eff.RAN SAC + J $W^* + t^* + N^* + W^* + t^*$

{Sk} Residual Mean \pm Std. 0.072 \pm 0.361 0.067 \pm 0.352 0.080 \pm 0.390 0.029 \pm 0.234 0.036 \pm 0.251 0.054 \pm 0.307

{Sk} Coverage $\epsilon = 0.01$ $\epsilon = 0.02$ 43.42 63.16 56.95 72.74 51.59 67.12 74.32 83.27 65.31 73.69 61.94 70.38

P Coverage $\epsilon = 0.01$ $\epsilon = 0.02$ 65.74 88.63 68.58 92.41 72.11 92.54 78.79 94.58 77.01 92.57 74.80 90.83

DP P N (Sec.4.4)

0.021 \pm 0.158 46.99 71.02 59.74 84.37

SP F N – Lseg SP F N – Lnorm + J SP F N – Lres SP F N – Laxis SP F N ($t^* - > Est.$)

0.029 \pm 0.178 0.022 \pm 0.188 0.017 \pm 0.162 0.019 \pm 0.188 0.013 \pm 0.140

50.04 76.47 79.81 80.80 85.25

62.74 81.49 85.57 86.11 90.13

62.23 83.21 81.32 86.46 86.67

77.74 91.73 91.52 94.43 94.91

SP F N

0.011 \pm 0.131 86.63 91.64 88.31 96.30

Obr. 7.3: Výsledky pokrytia primitív rôznymi metódami [12]

7.4 Efektívny RANSAC

V práci sa zameriavali na jednoduché geometrické tvary ako sú roviny, gule, valce, kužele a torus. Každý tvar má rôzny počet parametrov od troch po sedem. Každý vybraný bod určuje

31

jeden konkrétny parameter tvaru. Na minimalizáciu počtu potrebných bodov použili techniku odhadu približnej normály povrchu, pre každý bod zo vzorky. Tento prístup zabezpečil získať ďalšie parametre zo vzorky prostredníctvom normály ktorá poskytuje orientáciu povrchu. Vďaka tomu je možné odhadnúť každý uvedený tvar pomocou jednej alebo dvoch vzoriek bodov. Napriek tomu v práci použili viac vzoriek, pretože sa to osvedčilo výhodné. Nadbytočné parametre pomohli zvýšiť skóre, čo pomohlo na rýchlejšie overenie tvaru.[13]

32

8 Výstupy

8.1 Vyhodnotenie výsledkov PCL

Po spustení algoritmu v knižnici PCL, dostaneme hodnoty opisujúcich valec a jeho parametre. Algoritmus RANSAC z knižnice PCL vracia bod reprezentujúci bod na osi x y z, body reprezentujúce rotáciu valca a polomer valca. Presnosť vyhodnocovania PCL v tejto práci je zahrnutá na predpovedaní presnosti určenia polomeru valca. Vstupom pre algoritmus bol PCD s rovinou na ktorej ležal valec. Valec bol posúvaný po osiach X a Y ± 3 metre.

Tabuľka 8.1: Presnosť vyhodnocovania PCL s premiestnením predmetu v priestore

Posun po Posun po

Polomer

Chyba

X osi [m] Y osi [m] vyhodnotený PCL [%]

00

98.56

1.44

03

103.56

3.56

0 -3

103.44

3.44

30

103.48

3.48

33

106.02

6.02

3 -3

105.13

5.13

-3 0

103.44

3.44

-3 -3

105.90

5.90

stredná hodnota

chyby [%]

4.21

V štvrtom, piatom a poslednom riadku 8.1 vyšla vysoká chyba odhadu pretože teleso malo veľký šum a algoritmus nevedel presne odhadnúť valec a zachytil aj iné body takže stred valca určil nepresne a taktiež polomer. Chybu odhadu stredu sme získali pomocou nasledovnej rovnici.

$$\text{chyba} = |r_2 - 100 - 100| r_1$$

Kde r_2 je polomer odhadnutý algoritmom a r_1 je reálny polomer telesa.

(8.1)

33

Tabuľka 8.2: Presnosť vyhodnocovania PCL na rôznych polomeroch telesa. * označuje teleso položené na rovine, bez * je iba teleso v priestore s rôznymi nastaveniami PH

Reálny Polomer Polomer Polomer

Stred objektu

Chyba

polomer [cm] (PH 0.5) (PH 0.6) (PH 0.8)

[x y z]

[%]

10 13.85 12.41 7.11 0.0000 0.0002 1.412 30.48 100 11.25 9.40 67.43 -0.0016 -0.0014 1.14 197.60

30 27.69 23.78 27.78 -0.0014 -0.0034 1.4317 11.93 30 29.31 27.56 28.31 0.0242 -0.1120 1.1345 5.25

50 50 100 100

59.08 59.49 112.66 98.56

75.11 66.80 100.78 99.15

61.12 59.96 100.55 100.55

-0.0257 -0.0135 0.8796 -0.0073 0.0008 1.0662 0.0026 0.0084 0.9749 0.0417 0.0082 1.0995

30.20 17.72 5.21 0.95

150 137.18 117.63 123.20 -0.0065 -0.0043 1.0938 16.00 150 134.23 141.45 145.20 0.0002 -0.0003 0.9800 6.47

200 144.24 150.40 177.88 -0.0089 0.0292 0.910 21.24 200 172.88 190.49 196.28 0.0092 -0.0131 0.9704 6.725

stredná hodnota

chyby [%]

29.69

Riadky s * značia vstupné PCD s rovinou pod objektom. V 8.2 možno vidieť presnosť vyhodnotenia rôznych polomerov a stredov telesa (reálny stred telesa je $x=0$, $y=0$, $z=1$), s rôznymi nastaveniami algoritmu. Pri nastavení prahového rozsahu, nižšej hodnote nevedel algoritmus určiť väčšie teleso, pretože počet bodov na teleso ostával stále rovnaký. Naopak pri väčšom rozstupe bodov ležiacich v množine, pri PCD s rovinou, vznikali odchýlky, kvôli šumu ktorý algoritmus zachytil od roviny. Najlepšie výsledky pri rôznych telesách vyšli v stĺpci 3 v 8.2, pretože teleso odhadlo všade s menšou odchýlkou, kvôli strednej hodnote nastavovaných parametrov algoritmu.

34

S použitím PCD z 5.3 výstupom algoritmu so vstupným parametrom na získavanie valca je rovina objektu. PCD bol stiahnutý z internetu z PCL dokumentácie. [6]

Obr. 8.1: Výstup algoritmu RANSAC v knižnici PCL

8.2 Vyhodnotenie algoritmu SPFN

Po natrénovaní siete sme vložili vlastné dáta a zistovali sme presnosť určenia polomeru valca. Kvôli inému typu datasetu sa nám nepodarilo dosiahnuť presnosť ako sa podarilo na natrénovaní datasete, kvôli rôznorodosti dát. Ako bolo spomenuté vyššie dataset je natrénovaný na mechanické komponenty a náš dataset je vytvorený na jednoduché objekty položené na podlažke. Odchýlky taktiež mohlo spôsobiť vyššie rozlíšenie objektov, pretože v práci je napísané, že ocakávajú fixný počet vstupných bodov na objekt. Po testovaní dát sme dostali stred valca s jeho polomerom a rotáciou v osiach. Po vizualizácii dát sme vybrali valec, ktorý najlepšie opisuje z vloženého datasetu. Testovali aj s vyšším počtom Kmax ale nedosiahli lepšie výsledky, ako počtom Kmax, ktorý sme zvolili my pri tréňovaní siete. Kmax reprezentuje

35

maximálny počet primitív.

Tabuľka 8.3: Presnosť vyhodnocovania SPFN

Posun po Posun po

Polomer

Chyba

X osí [m] Y osí [m] vyhodnotený SPFN [cm & %]

00

78.31

21.69

03

103.31

3.31

0 -3

82.41

17.59

30

78.31

21.68

33

91.41

8.59

3 -3

51.56

48.44

-3 0

93.12

6.88

-3 -3

97.22

2.77

stredná hodnota

chyby [%]

16.34

V tabuľke 8.3 vidno chybu vyhodnocovania polomeru odhadovaného telesa s premiestnením v priestore. SPFN mal problém vyhodnotiť polomer objektu presúvaný v priestore. Vznikajú vyššie odchýlky v niektorých riadkoch, kvôli zašumeniu vstupných dát, ktoré mali pod sebou podložku čo mohlo spôsobiť zle vyhodnotenie polomeru.

36

Tabuľka 8.4: Presnosť vyhodnocovania SPFN na rôznych polomeroch telesa. * označuje teleso položené na rovine, riadky bez označenia * sú iba telesá v priestore

Reálny Polomer

Stred objektu

Chyba

polomer [cm] [cm]

[x y z]

[%]

10 41.13 -0.0622 0.0351 1.8132 311.30 10 31.40 0.0550 -0.0145 1.5530 214.00

30 36.10 -0.0232 -0.0106 1.345 20.33 30 41.14 0.0407 -0.0122 1.559 37.13

50 52.12 0.0163 0.0149 1.0346 4.2 50 42.87 0.0015 0.0032 1.0823 14.26

100 98.51 0.0235 -0.0011 1.0677 1.49 100 113.10 -0.0131 0.0218 1.1271 13.10

150 151.28 0.0047 0.0141 1.0606 0.80 150 145.12 0.0322 0.0053 1.2971 3.25

200 201.24 -0.0093 0.0182 1.0258 0.62 200 211.37 -0.0815 0.0906 1.0827 5.68

stredná hodnota

chyby [%]

52.18

V tabuľke 8.4 možno vidieť odhad polomeru a stred rovinného telesa ako v 8.2. Neurónová sieť ako bolo spomenuté vyššie bola natrénovaná na telesách s väčšími rozmermi, takže algoritmus začal lepšie odhadovať polomer až po dosiahnutí väčších rozmerov. Po dosiahnutí rozmeru sa chyba výrazne znížila. Keďže sieť bola trébovaná na jednoduchých telesách v priestore, vložené dáta s rovinou (riadky s *) mali výrazne vyššiu chybu. Chyba sa počítala pomocou 8.1.

8.3 Porovnanie výsledkov

Z nameraných hodnôt z PCL a SPFN, sme zostrojili tabuľky 8.1, 8.2 a 8.3, 8.4, kde porovnávame namerané výsledky. Pri tabuľkách 8.1 a 8.3 vstupnými dátami bol valec s polomerom 1 meter, pričom vstupný PCD obsahoval šum. Tabuľky 8.2 a 8.4 vstupom boli PCD, pričom objekt neobsahoval šum a sledovali sme presnosť vyhodnocovania oboch implementácií. Z odhadnutých stredov objektov sme vypočítali relatívnu odchýlku, ktorú sme graficky znázornili v 8.2 a 8.3. V tabuľke môžeme pozorovať ak sa jednalo o PCD bez roviny pod objektom RANSAC implementovaný o NN odhadoval rozmer telesa presnejšie ako PCL. Implementácia PCL s jedným nastavením algoritmu mal problém v rozdielných veľkostiach

37

objektov. Pre používateľ a ktorý nemá nastudovanú problematiku ohľadom RANSAC, by mohol mať problém s optimálnymi nastaveniami algoritmu. Na druhú stranu nainštalovanie PCL je jednoduchšie a vyžaduje menej výkonný hardware ako SPFN. Po natrénovaní siete vyhodnocovanie vstupných dát pomocou SPFN. Vypočítame si presnosť určovania stredov odchýlok na základe reálneho bodu a bodu vyhodnoteného algoritmom na základe nasledovného vzťahu:

$$\text{odchýlka} = (x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2$$

(8.2)

Kde súradnice s indexom 1 sú súradnice reálneho bodu [0,0,1] metrov, a súradnice s indexom 2 sú súradnice aproximovaného stredy pomocou implementácii. Z vypočítaných odchýlok pre každý test sme zostrojili grafické závislosti.

Obr. 8.2: Odchýlky aproximácie stredy objektu v prázdnom priestore

Na obrázku 8.2 môžeme pozorovať, že implementácia SPFN vedie výrazne lepšie aproximovať polohu objektu v priestore. Vyššiu odchýlku PCL mohla spôsobiť nepoužitá rovina s ktorou algoritmus ráta pri vyhodnocovaní objektov.

38

Obr. 8.3: Odchýlky aproximácie stredy objektu položeného na podložke Pri použití roviny pod objektom, sme pri PCL dosiahli oveľa lepšie výsledky. Odchýlky oproti metóde SPFN mohlo spôsobovať nepresné nastavenie algoritmu, pretože algoritmus bol nastavený na jeden rozmer objektov, po nastavení parametrov algoritmu pre každý rozmer zvlášť, by sme dostali lepšie výsledky, čo by spôsobovalo predĺženie času identifikácie objektu. Nastavovanie pri metóde SPFN nebolo potrebné takže je efektívnejšie na používanie.

39

Záver

Úspešne sa nám podarilo spojiť nami zvolené implementácie na rozpoznávanie objektov s použitím algoritmu RANSAC a jeho rozšírenie o NN. Klasickým algoritmom sme získali jednoduché objekty s odchýlkou 4.21% pri určovaní polomeru a 29.69% pri určovaní, čo zapríčinil šum na vstupných PCD. Narozdiel od implementácii SPFN, ktorá určila polomer objektu so 16.34% odchýlkou a stred objektu s 52.18% odchýlkou. Odchýlky pri SPFN spôsobili menšie objekty ktoré mali >200% odchýlku, čo zvýšilo strednú hodnotu chyby. Na dátach z Kinectu sa nám nepodarilo nájsť žiadane objekty, pretože Kinect mal nižšie rozlíšenie a na výstupnom PCD nebolo dostatok bodov reprezentujúcich objekt, takže nám za kužeľ identifikovalo prevažne kúty v miestnosti, aj s rôznymi nastaveniami RANSAC.

Implementácie s neuronovými sieťami 7.1 a 7.2 spracováva 2D dáta, čo nevyhovovalo zadaniu. Vďaka týmto implementáciám som sa lepšie naučil pracovať s Linuxom a odporúčam používať virtuálne prostredie, pretože pri práci so staršími Python balíkami, je možné že si na novšom operačnom systéme prepíšete systémové premenné, alebo nechcete inštalacným skriptom odinštalujete Python. Tieto implementácie, by sa dali v budúcnosti používať na lokalizáciu objektov na 2D snímkach, určenie lokality odfoteného objektu a na základe podobnosti s vytvoreným datasetom určiť lokalitu budov.

Práca SPFN bola prispôbená na starší hardware, takže som sa v práci naučil pracovať s platformou Docker, ktorá funguje podobne ako virtuálne prostredie, ale narozdiel od prostredia sa tam ľahšie menia python balíky a ľahšie pristupovať k starším verziám pythonu. Naučil som sa, že nie je vhodné ho inštalovať ako root používateľ, pretože pri kompilácii projektu vytvára docker súbor o veľkosti projektu, takže po niekoľkých kompiláciách sa mi zaplnil root priečinok a nevedel som spustiť aplikácie a po reštarte som sa nevedel dostať do operačného systému.

Získal som schopnosť využívať algoritmy a knižnice na spracovanie 3D dát. Venoval som sa dôkladnému štúdiu problematiky týkajúcej sa 3D skenovania a práce s Kinectom. V rámci ďalších projektov je nevyhnutné implementovať softvér, ktorý umožní kalibráciu kamery Kinect, aby sme minimalizovali odchýlky a mohli využiť získané dáta v našich implementáciách.

Pri hodnotení presnosti sme si všimli, že oba algoritmy vykazujú podobné odchýlky. Avšak, po dosiahnutí väčšieho rozmeru objektu výrazne znížila SPFN chybu odhadu, na rozdiel od PCL, ktorý vyžaduje zmenu nastavení parametrov algoritmu pri rôznorodých vstupných dátach, čo je neefektívne. Naopak, pre SPFN by stačilo dodatočné tréningy na menšie objekty, aby vedel presnejšie aproximovať menšie telesá.

40

V konečnom dôsledku majú oba algoritmy veľký potenciál v reálnom svete pre rozpoznávanie objektov. Avšak, PCL disponuje lepšou dokumentáciou, čo zjednodušilo prácu s knižnicou a implementáciu na vlastné dáta. Knižnica PCL je schopná pracovať s bežne používanými súborami, na rozdiel od neuronových sietí, kde je potrebný špecifický formát vstupných dát, čo predstavovalo väčšiu výzvu pri ich simulácii. V odhadoch stredy objektu dosiahol SPFN výrazne lepšie výsledky. Algoritmus si vyberá ďalšie body na základe získaných parametrov, čo prispieva k presnejšiemu odhadu.

41

Zoznam použitej literatúry

1. TECH27.COM (ed.). What are point clouds? tech27. Dostupné tiež z: <https://web.archive.org/web/20220331112105/https://tech27.com/resources/point-clouds/>.

2. HARDESTY, Larry (zost.). Explained: Neural networks: metodický materiál pro autory vysokoškolských kvalifikačných prácí [online]. MIT: MIT News Office, 2014-04-14 [cit. 2022-10-14]. Dostupné z: <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>.

3. VNUK, Peter (ed.). Prehľad: Ako fungujú umelé neuronové siete. [B.r.]. Dostupné tiež z: <https://www.nextech.sk/a/Ako-funguju-umele-neuronove-siete>.

4. Point cloud library. [1647] Dostupné tiež z: <https://pointclouds.org/>. 5. RADU BOGDAN RUSU, Steve Cousins (zost.). Point cloud library (pcl): 3D is here.

2011 IEEE international conference on robotics and automation: IEEE. Dostupné tiež z:

https://scholar.google.com/citations?view_op=view_citation&hl=en&user=U_NEZHQAAAAJ&citation_for_view=U_NEZHQAAAAJ:uX6o8ySG0sC. 6. (PCL), Point Cloud Library. PCL Random Sample Consensus Documentation [online]. Point Cloud Library. [cit. 2023-06-01]. Dostupné z: https://pointclouds.org/documentation/classpcl_1_1_random_sample_consensus.html.

7. BRACHMANN, Eric a ROTHER, Carsten. Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses. In: ICCV. 2019. 8. MICROSOFT. Kinect. Microsoft. Dostupné tiež z: <https://www.microsoft.com/en-us/kinect>. 9. FAIRHEAD, Harry (zost.). OpenNI 2.0 - Another Way To Use Kinect [online]. i-programmer.info, 2012-12-22 [cit. 2022-12-22]. Dostupné z: <https://www.i-programmer.info/news/194-kinect/5241-openni-20-another-wayto-use-kinect.html>.

10. FLORENT POUX, Ph.D. (zost.). 3D Model Fitting for Point Clouds with

RANSAC and Python: A 5-Step Guide to create, detect, and fit linear models for unsupervised 3D Point Cloud binary segmentation: RANSAC implementation from scratch. Towards Data Science. Dostupné tiež z: <https://towardsdatascience.com/3d>

42

model - fitting - for - point - clouds - with - ransac - and - python 2ab87d5fd363. 11. BRACHMANN, Eric a ROTHER, Carsten. Learning less is more - 6D camera localization via 3D surface regression. In: CVPR. 2018. 12. Li, Lingxiao, SUNG, Minhyuk, DUBROVINA, Anastasia, Yi, Li a GUIBAS, Leonidas. Supervised Fitting of Geometric Primitives to 3D Point Clouds. 2018. Dostupné z eprint: arXiv:1811.08988. 13. RUWEN SCHNABEL, Roland Wahl a KLEIN, Reinhard. Efficient RANSAC for point-cloud shape detection. 2007.

43

Prílohy

A Kód na získanie PCD z Kinectu 45 B Programy 46 C Vstupné data pre algoritmy 47 D Návod na spúšťanie algoritmov 48

44

A Kód na získanie PCD z Kinectu

```
if (enable_rgb && enable_depth){ registration ->apply(rgb, depth, &undistorted, &registered); std::cout << pcd_count << " " << registered . height << " " << registered . width << std::endl; std::ofstream myfile; char file_name [15]; sprintf ( file_name, " points %d.pcd", pcd_count++); myfile.open ( file_name ); myfile << [1114]"VERSION 0.7" << std::endl << "FIELDS x y z rgb" << std::endl << "SIZE 4 4 4 4" << std::endl << "TYPE F F F U" << std::endl << "COUNT 1 1 1 1" << std::endl << "WIDTH 512" << std::endl << "HEIGHT 424" << std::endl << "VIEWPOINT 0 0 0 1 0 0 0" << std::endl; myfile << "POINTS 217088" << std::endl << "DATA ascii" << [1114]std::endl; for ( size_t I = 0; I < registered . height; ++I){ for ( size_t J = 0; J < registered . width; ++J){ float X, Y, Z, RGBVALUE; registration ->getPointXYZRGB(&undistorted, &registered, I, J, X, Y, Z, RGBVALUE); myfile << X << " " << Y << " " << Z << " " << RGBVALUE << std::endl; } } myfile . close ();
```

Výpis A.1: Ukážka kinect_pcd Open-source knižnica libfreenect2, ktorá zabezpečuje komunikáciu h'lbkovej kamery s operačným systémom, poskytuje vzorové kódy na získavanie dát z Kinectu. Vzorový kód je potrebné upraviť aby získane body z Kinectu uložil do .pcd súboru. Na vytváranie .pcd súborov z Kinectu existuje mnoho hotových riešení, ale neposkytujú vizualizáciu v c'ase, aby kameru bolo možné vhodne napoľohovať a tak uložiť PCD.

45

B Programy

pcl/ - Priech'ínok s PCL programami cylinder.cpp - Algoritmus na nájdenie valca v 3

matlab/ - Priech'ínok s matlab skriptami cylinder_center.m - Výpočet stredu valca na základe parametrov získaných RANSAC z 3 errors.m - Skript na výpočet relatívnej chyby urč'ovania bodov

python/ - Priech'ínok s python skriptami list.py - skript na vypísanie parametrov výstupného súboru z 7.3 pcd.py - Skript na vytvorenie PCD s vyšším rozlíšením

kinect/ - Príklad používania Kinectu live.cpp - Kód na získavanie PCD z Kinectu s videním v reálnom c'ase z knižnice libfreenect2

46

C Vstupné data pre algoritmy

Obr. C.1: Vstupné mrac'no bodov so šumom pcd/pcd_so_sumom.pcd

· Mrac'no bodov so šumom

Obr. C.2: Vstupné mrac'no bodov bez šumu pcd/pcd_na_velkost_radiusu.pcd

· Mrac'no bodov bez šumu 47

D Návod na spúšťanie algoritmov

Na spustenie algoritmu na nájdenie valca, je potrebné mať nainštalovanú knižnicu 3. V programe je potrebné špecifikovať cestu ku vstupným PCD. Po nainštalovaní je nutné vytvoriť CMakeFiles.txt, aby bolo možné kompilovať .cpp súbor po úpravách algoritmu. Po skompilovaní vytvorí spustiteľný súbor, v terminály pomocou

\$./cylinder <nazov_vstupneho_pcd>

Skripty v Matlabe je potrebné mať nainštalovaný Matlab a pripravené potrebné dáta. Skript cylinder_center.m prijíma dáta z programu spomínaného vyššie a je potrebné mu vložiť získane parametre do premennej "values" na ich urč'enú pozíciu. Program errors.m ma v programe zadefinovaný jeden bod ktorý zodpovedá reálnej polohe objektu. Súradnice je potrebné zapísať do premennej "real_point" body odhadnuté pomocou algoritmu do premennej "approx_point", v ktorej je možné definovať viac bodov. Skript následne vypoc'íta relatívnu chybu a graficky ju znázorni.

Na skripty v pythone je potrebné mať nainštalované potrebné knižnice, s ktorými skript pracuje. Skript list.py výpise obsah celého súboru s príponou .h5, ktorý je možné získať z algoritmu 7.3 pri testovaní vlastných dát. Súbor obsahuje orientáciu normál, vstupne body a taktiež informácie o nájdenom objekte ako je jeho orientácia, polomer a stred.

\$ python3 list.py <nazov_souboru> s .h5 príponou

Skript pcd.py sa používa na získavanie PCD, z Kinectu. Nevýhodou tohto skriptu je že používateľ nevidel výstup z Kinectu v reálnom c'ase, ale rozlíšenie výstupného PCD bolo vyššie.

\$ python3 pcd.py

Program live.cpp je súčasťou knižnice libfreenect2, ktorá komunikuje s Kinectom. Výhodou tohto programu je, že používateľ vidí záznam z kamery v reálnom c'ase a vie si prispôbiť scénu, ktorá následne sa uloží na disk. Súbor je potrebné skompilovať spolu s knižnicou a po skompilovaní sa vytvorí spustiteľný súbor, ktorý na obrazovke otvorí okno, kde možno vidieť záznam z RGB kamery, IR kamery, h'lbkového senzoru a výstupný ??.

Sekcia 7.3 poč'as testovania používa kód od autora témy. Návod na používanie a potrebné príkazy na spustenie sú popísane autorom.

48

metadata: <https://opac.crzp.sk/?fn=detailBiblioForm&sid=08878EB2E7E37CD0999A75C0752B>

webprotokol: <https://www.crzp.sk/eprotokol?pid=6581D068B8B540AD82F1CA7125DE5077>