

**SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY**

Evidenčné číslo: FEI-100863-111119

**POROVNANIE SPRACOVANIA 3D DÁT ANALYTICKÝM
METÓDAMI A NEURÓNOM VÝMI SIEŤAMI**
BAKALÁRSKA PRÁCA

Študijný program: Robotika a kybernetika

Názov študijného odboru: kybernetika

Školiace pracovisko: Ústav robotiky a kybernetiky

Vedúci záverečnej práce: Ing. Miroslav Kohút

Bratislava 2023

Maroš Kocúr



ZADANIE BAKALÁRSKEJ PRÁCE

Autor práce:
Študijný program:
Študijný odbor:
Evidenčné číslo:
ID študenta:
Vedúci práce:
Vedúci pracoviska:
Miesto vypracovania:

Maroš Kocúr
robotika a kybernetika
kybernetika
FEI-100863-111119
111119
Ing. Miroslav Kohút
prof. Ing. Jarmila Pavlovičová, PhD.
Ústav robotiky a kybernetiky

Názov práce:

Porovnanie spracovania 3D dát analytickými metódami a neurónovými sietami

Jazyk, v ktorom sa práca vypracuje:

slovenský jazyk

Špecifikácia zadania:

Na základe rastúcej potreby práce s 3D dátami dochádza k neustálemu testovaniu a výskumu nových možností algoritmov. Súčasná práca sa venuje algoritmu RANSAC pomocou ktorého je možné matematicky opísať rôzne zoskupenia bodov. Modely je následne možné použiť pri ďalšom spracovaní 3D dát. Cieľom práce je porovnať analytickú metódu implementácie RANSAC modelu s jeho rozšírením, ktoré implementuje aj modely neurónových sietí na dosiahnutie lepších výsledkov.

Úlohy:

1. Popište a analyzujte základnú problematiku práce s 3D dátami a neurónovými sietami.
2. Analyzujte algoritmus RANSAC a možnosti jeho rozšírenia pomocou NN modelu.
3. Implementujte vami vybrané algoritmy.
4. Otestujte implementované algoritmy na zvolených dátach.
5. Zdokumentujte a vyhodnotte výsledky.

Termin odovzdania práce:

02. 06. 2023

Dátum schválenia zadania práce:

12/08/2022

Zadanie práce schválil:

doc. Ing. Eva Miklovičová, PhD.
garantka študijného programu

ANOTÁCIA

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Študijný program:	Robotika a kybernetika
Autor:	Maroš Kocúr
Bakalárska práca:	Porovnanie spracovania 3D dát analytickým metódami a neurónovými sietami
Vedúci záverečnej práce:	Ing. Miroslav Kohút
Miesto a rok predloženia práce:	Bratislava 2023

Práca sa zameriava na porovnanie algoritmu RANSAC a rozšírenie algoritmu o neurónovú siet'. 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, kužeľe.

Kľúčové slová: RANSAC, PCL, neurónová siet'

ABSTRACT

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

Study Programme:	Robotics and cybernetics
Author:	Maroš Kocúr
Bachelor's thesis:	Comparison of 3D Data Processing by Analytical Methods and Neural Networks
Supervisor:	Ing. Miroslav Kohút
Place and year of submission:	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 vd'ačnosťou by som sa chcel pod'akovat' Ing. Miroslavovi Kohútovi za ochotu, nájdený čas a odborné vedenie počas riešenia bakalárskej práce. Taktiež by som sa chcel pod'akovat' Ústavu robotiky a kybernetiky, za nadobudnúte informácie počas bakalárskeho štúdia, ktoré mi pomahali riešiť problémy s prácou. Ďalej by som sa chcel pod'akovat' Ing. Michal Tölgessy, PhD. za poskytnutie hľbkovej kamery, s ktorou som mal možnosť pracovať.

Obsah

Úvod	10
1 Point Cloud	11
1.1 General	11
1.2 Použitie	11
1.3 Konverzia do 3D povrchu	11
2 Neurónové siete	12
2.1 História	12
2.2 Čo je Neural Network (neurónové siete) (NN)	12
2.3 Vývoj	12
3 PCL	14
3.1 Použitie	14
3.2 Pred inštaláciou	14
3.3 Inštalácia 3	14
3.4 Nastavovanie parametrov Random sample consensus (RANSAC)	15
4 RANSAC	17
4.1 Overview	17
4.2 Algoritmus	17
4.3 RANSAC 3D	18
4.4 Neural guided RANSAC	18
5 KINECT	20
5.1 Ako funguje Kinect	20
5.2 Výstup z KINECTU	21
5.2.1 List potrebných Knižnic pre Ubuntu 22.04 LTS	21
5.3 OpenNI 2.0	21
5.4 Point cloud (PCD) súbor	22
5.5 Výstup z Kinectu	23
6 Implementovanie RANSAC algoritmu	24
6.1 PointCloudData	24
6.2 Zvolenie geometrického modelu	24
6.3 Definovanie parametrov	24

6.4	Implementovanie RANSACu	25
6.5	Vyhodnotenie 3	25
7	Implementovanie RANSAC algoritmu s použitím neurónovej sieti	29
7.1	ngransac	29
7.2	ngdsac cameraloc	30
7.3	SFPN	32
7.4	Efektívny RANSAC	34
7.5	Naše výstupy	34
8	Výstupy	37
Záver		38
Zoznam použitej literatúry		39
Prílohy		41
A	Kód na získanie Point Cloud (mračno bodov) (PCD) z Kinectu	42
B	Matlab scripty	43
C	Vstupné data pre algoritmy	44
D	Návody na spušťanie algoritmov	45

Zoznam obrázkov a tabuliek

Obrázok 2.1	Príklad neurónovej siete	13
Obrázok 4.1	Priklad algoritmu v 2D rovine	17
Obrázok 5.1	Kinect v2	20
Obrázok 5.2	Architektúra softwerového vývoja	22
Obrázok 5.3	Príklad výstupný Point Cloud z hĺbkovej kamery	22
Obrázok 5.4	Aktualny vystup z kinectu	23
Obrázok 6.1	Grafické znázornenie rovnice 6.3	25
Obrázok 6.2	Výstup algoritmu RANSAC v knižnici PCL	28
Obrázok 7.1	Výstup Neural guided RANSAC	30
Obrázok 7.2	Snímka z hĺbkovej kamery Kinectu a snímka pretransformovaná na heat mapu	31
Obrázok 7.3	Výsledky pokrytie primitív rôznymi metódami	34
Obrázok 8.1	Relatívny error implementácie	37
Obrázok C.1	Vstupné mračno bodov so šumom	44
Obrázok C.2	Vstupné mračno bodov bez šumu	44
Tabuľka 6.1	Presnosť vyhodnocovania 3 s premiestnením predmetu v priestore .	26
Tabuľka 6.2	Presnosť vyhodnocovania 3 na rôznych polomeroch telesa. * označuje teleso položené na rovine, bez * je iba teleso v priestore	27
Tabuľka 7.1	Odhad základných matíc	30
Tabuľka 7.2	Premiestnenie kamery v priestore	31
Tabuľka 7.3	Presnosť určovania Supervised Fitting of Geometric Primitives (SPFN) oproti Efektívnomu RANSAC	32
Tabuľka 7.4	Presnosť určovania SPFN oproti Efektívnomu RANSAC [10]	33
Tabuľka 7.5	Presnosť vyhodnocovania SPFN	35
Tabuľka 7.6	Presnosť vyhodnocovania SPFN na rôznych polomeroch telesa. * označuje teleso položené na rovine, bez * je iba teleso v priestore	36

Zoznam skratiek

NG-RANSAC	Neural Guided Random sample consensus
NN	Neural Network (neurónové siete)
PCD	Point Cloud (mračno bodov)
RANSAC	Random sample consensus
SPFN	Supervised Fitting of Geometric Primitives

Úvod

Práca porovnáva získavanie jednoduchých objektov z mračna bodov, naskenovaných Kinectom v2. Porovnáva sa analytický spôsob pomocou algoritmu RANSAC a jeho implementácia s neurónovou siet'ou. Na získanie objektu pomocou analytickej metódy bola zvolená knižnica PointCloudLibrary 3 a na implementáciu pomocou neurónovej siete bol vybraný algoritmus SPFN. Porovnávala sa presnosť určenia polomeru telesa, získaných z oboch implementácií. Testovanie prebehlo na viacerých objektoch s rôznymi polomermi. V práci boli testované aj 2 iné typy RANSAC implementované s neurónovou siet'ou, len pri testovaní nevyhovovali. V práci je popísaný postup inštalácie knižníc, Kinectu, práca s RANSAC, ako aj nastavovanie parametrov.

1 Point Cloud

1.1 General

Je to súbor bodov v priestore. Body môžu reprezentovať 3D tvary alebo objekty. Každý bod má karteziánsky 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 zarovnáva s 3D modelmi alebo inými PCD ako registrácia množín bodov. [1]

1.2 Použitie

Pre priemyselnú metrológiu alebo inšpekcii pomocou priemyselnej počítačovej tomografie možno mračno bodov vyrobeného dielu zosúladit s existujúcim modelom a porovnať, aby sa skontrolovali rozdiely. Geometrické rozmery a tolerancie možno získať aj priamo z PCD.

1.3 Konverzia do 3D povrchu

V geografických informačných 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 vzdialenosná a objemová approximácia.

2 Neurónové siete

2.1 História

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

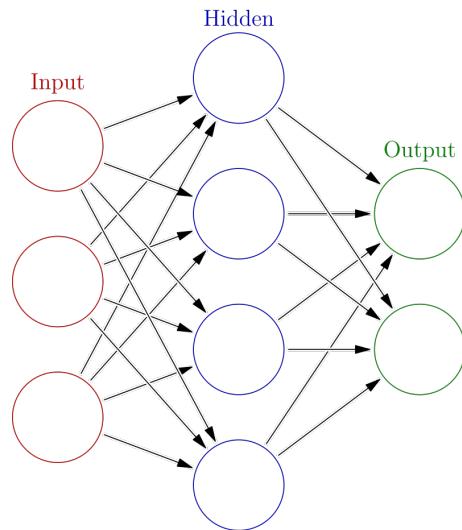
2.2 Čo je NN

NN sú myšlené na robenie strojového učenia, v ktorom sa počítač učí vykonávať úlohy na základe analyzovania trénovacích príkladov, ktoré sú zvyčajne ručne označené. Systém na rozpoznávanie objektov by mohol mať prístup k tisíciam obrázkov označených ako auto, dom, gul'a a podobne. NN siet' hľadá vizuálne charakteristiky v obrázku ktoré ktoré majú vzájomný vzťah s konkrétnym označením. Voľne modelovaná NN na ľudskom mozgu má tisíce až milióny jednoduchých node, ktoré sú husto prepojené. Väčšina sieti sa dnes organizuje do vrstiev neurónov a tie posielajú údaje vpred, čo znamená že dátu nimi prechádzajú iba v jednom smere. Jeden neurón môže byť pripojená k viacerým neurónov vo vrstvách pod ňou, z ktorej príma dátu a viacej neurónov vo vrstve nad ňou kde spracované dátu posielala. Ku každému vstupnému prípoju sa pridelí číslo známe ako "váha". Ak je siet' aktívna, neurón príjme rôzne dátu z každého vstupu a vynásobí ich pridelenou váhou, potom všetky výsledky scítá ak je výsledne číslo pod threshold hranicou, neurón nepošle žiadne dátu do ďalšej vrstvy. Ak číslo prekračuje threshold hodnotu, tak neurón "vystrelí", čo znamená že pošle súčet vstupov na všetky výstupy. Keď sa NN trénuje, všetky váhy a thresholdy sú nastavené na náhodnú hodnotu. Trénovacie dátu sú posielané do spodnej vstupnej vrstvy a prechádzajú cez nasledujúce vrstvy, násobia sa a scítavajú sa v zložitými cestami, pokiaľ radikálne transformované nedôjdu na vrchnú výstupnú vrstvu. Počas tréningu sa váhy a prahové hodnoty neustále upravujú kým tréningové údaje s podobnými štítkami neprinášajú podobne výstupy. [2]

2.3 Vývoj

Neuronové siete opísané v roku 1944 mali váhy a threshold hodnoty, ale neboli usporiadané do vrstiev a výskumníci nešpecifikovali trénovacie mechanizmus. Výskumníci dokázali ukázať, že NN dokáže v princípe vypočítať hocjakú funkciu ako digitálny počítač. Výsledkom bola viac nesurová veda ako počítačová a cieľom bolo poukázať, že ľudský mozog

môžme považovať za výpočtové zariadenie.[2]



Obr. 2.1: Príklad neurónovej siete

3 PCL

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

3.1 Použitie

3 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, prispôsobenie modelu, vyhľadávanie objektov a segmentácie. 3 ma 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 vzhladu a vytvárať povrhy z PCD a vykrel'ovať ich.[4]

3.2 Pred inštaláciou

3 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. Ďalej je potrebné nainštalovať OpenNI 2 knižnicu, ktorá zabezpečuje komunikáciu s Kinectom v2.[3]

3.3 Inštalácia 3

3 je dostupný na mnoho distribúcií Linuxu ako Ubuntu, Debian, Fedora, Gentoo a Arch Linux. 3 na distribúciach Ubuntu a Debian môžeme nainštalovať pomocou.

```
sudo apt install libpcl-dev
```

Na Windows sa 3 inštaluje pomocou vpckg package manažéra vytvoreného Microsoftom.

```
PS> ./vcpkgs install pcl
```

MacOS má Homebrew package manažéra, ktorý podporuje inštaláciu packagov, ktoré

Apple alebo Linux nedokáže nainštalovať.

```
brew install pcl
```

Toto sú odporúčané inštalácie pre 3 na daných operačných systémoch. [3]

Na používanie 3 je potrebné v kóde vložiť potrebné knižnice. Po nainštalovaní sa v systéme nastavia premenné, takže stačí použiť príkaz v C++, #include <pcl“názov_knižnice“.h>.

Pre kompliaciu je potrebné vytvoriť súbor CMakeList.txt v ktorom zadefinujeme potrebné premenné aby make vedel nájsť cestu ku knižnici a vedel aké súbory má skompilovať. Ďalej je potrebné vytvoriť priečinok s názvom build a v terminály v priečinku build executnut’ príkaz cmake “cesta k CMakeList.txt“, ktorý vytvorí makefile, ktorý skompliluje potrebné zdrojové kódy a vytvorí binárny súbor pomocou ktorého je možné spustiť projekt.

3.4 Nastavovanie parametrov RANSAC

V kóde treba zadefinovať parametre, ktoré nastavia algoritmus a odfiltrujú šum. Z pointcloudu sa najprv odfiltruju body, ktoré ležia d’alej ako sú zadefinované pomocou príkazu. **pass.setFilterLimits(0, 1.5)** - zo vstupného PCD sa odfiltrujú iba body ležiace od 0 po 1.5 metra od kamery.

Ďalší 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

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.

- 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 **cylinder_center.m**, je možné vypočítať stred valca.

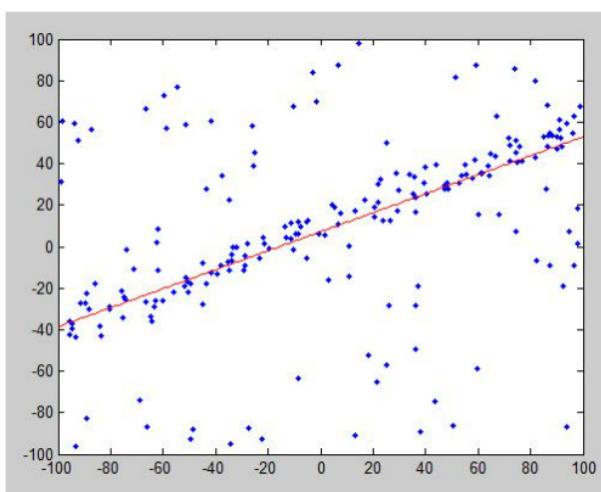
4 RANSAC

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-odhadu a metódou najmenších štvorcov s prepojením na strojové učenie. RANSAC bol vytváraný komunitov ludi používajúci strojové učenie. RANSAC je vzorkovacia technika ktorá generuje kandidáta na minimálny počet pozorovaní potrebných na zistenie odhadu parametrov ležiacich pod modelom. Naroďal 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. Vyriešenie parametrov pre model
3. Určenie kol'ko bodov z množiny všetkých bodov leží s preddefinovanou ϵ .
4. Ak zlomok bodov ležiacich v preddefinovanej ϵ , presahuje preddefinovaný prah τ , prehodnotí parametre modelu použitím všetkých identifikovaných inlierov(Body ležiace v prahovej hodnote) a ukonči.
5. Inak, opakuj kroky 1 až 4, s maximálnym N opakovaniami.



Obr. 4.1: Priklad algoritmu v 2D rovine

Modre sú body z datasetu, pomocou RANSAC algoritmu sa určili 2 body, ktoré majú vo svojom subseze najmenej bodov ležiacich mimo alfy.

4.3 RANSAC 3D

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

1. Príprava dát: Načítajte 3D dátu vo formáte bodového oblaku, ktorý obsahuje 3D súradnice bodov.
2. Výber minimálnej sady: Náhodne vyberte minimálnu sadu bodov z celého bodového oblaku. Táto sada by mala obsahovať minimálny počet bodov potrebný na určenie vášho modelu.
3. Výpočet modelu: Na základe vybranej sady bodov vypočítajte model. To znamená, že vykonáte výpočet, ktorý určuje parametre modelu, napríklad roviny alebo valca, ktorý sa snažíte identifikovať.
4. Vyhodnotenie bodov ležiacich v hraničnej hodnote: Pre všetky body v bodovom oblaku vypočítajte vzdialenosť od modelu. Body, ktoré majú vzdialenosť menšiu ako určený prah, považujte za inliérov. Vypočítajte počet inliérov pre váš model.
5. Opakovanie krokov 2 až 4: Opakujte kroky výberu minimálnej sady, výpočtu modelu a vyhodnotenia inlierov určený počet krát alebo do dosiahnutia preddefinovaného kritéria ukončenia.
6. Výber konečného modelu: Vyberte model s najväčším počtom inliérov ako konečný model.
7. Voliteľné: Ak chcete vylepšiť konečný model, môžete použiť všetkých inlierov na presnejší odhad parametrov vášho modelu pomocou napríklad metódy najmenších štvorcov.

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

4.4 Neural guided RANSAC

Pri implementácii neurónmi riadeného RANSAC (Neural Guided Random sample consensus (NG-RANSAC)) sa kombinuje metóda RANSAC s neurónovou siet'ou, ktorá slúži na

riadenie výberu minimálnych množín a zlepšenie robustnosti odhadového procesu. Nasleduje hrubý prehľad implementácie NG-RANSAC:

1. Príprava dát: Pripravte dataset, ktorý sa bude používať na trénovanie neurónovej siete. Tieto dáta zahŕňajú vstupy vo forme korešpondencií medzi obrázkami a očakávané výstupy v podobe binárnych hodnôt, ktoré označujú, či je každá korešpondencia inliérom alebo odľahlým bodom.
2. Trénovanie neurónovej siete: Trénujte neurónovú sieť na predikciu pravdepodobnosti, že každá korešpondencia je inliérom. 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: Modifikujte algoritmus RANSAC tak, aby využíval neurónovú sieť na odhad pravdepodobnosti, že každá korešpondencia je inliérom. Namiesto náhodného výberu podmnožín korešpondencií pre výpočet hypotéz modelu použite predikované pravdepodobnosti na riadenie výberu minimálnych množín.
4. Výpočet konečného modelu: Po výbere minimálnych množín použite tieto množiny na výpočet hypotéz modelu a vyhodnoťte každú hypotézu podľa počtu inliérov. Vyberte hypotézu s najväčším počtom inliérov ako konečný model.
5. Vylepšenie modelu: Voliteľne môžete vylepšiť konečný model pomocou všetkých inliérov 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.[5]

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 trvajúcemu svetlu dopadnúť na objekt, vďaka ktorému vie Kinect poskytnúť rekognizáciu giest 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.



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 ma 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 príletu meranej hĺbky touto metódou môže byť presnejšie vypočítaný v kratšom čase, čo zabezpečí viac snímkov za sekundu. Hned' ako kinect naskenuje hĺbkovú fotografiu, použije metódu zistovania hrán k vytýčeniu bližších objektov z pozadia fotky.

5.2 Výstup z KINECTU

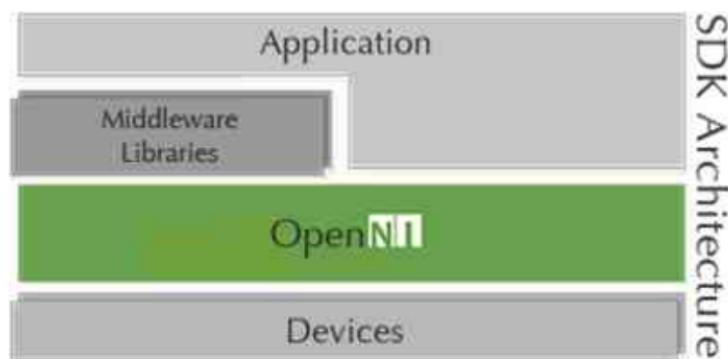
Na získanie výstupu z Kinectu je potrebné si nainštalovať Open-source knižnicu LibFreenect2, ktorá je určená na získavanie videí z Kinectu verzie 2. Knižnica disponuje vzorovým kódom, ktorý po spustení zapne Viewer a okno rozdelí na štyri časti a v nich uvidíme výstupy Infra Red kamery, farebnej kamery a depth kamery. Nato aby sme knižnicu mohli používať sme potrebovali doinštalovať potrebné knižnice a súčasti. Na získanie snímky z Kinectu sme použili Python script, ktorý nám vytvorí point cloud súbor, v ktorom budeme hľadať požadované tvary.

5.2.1 List potrebných Knižnic 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

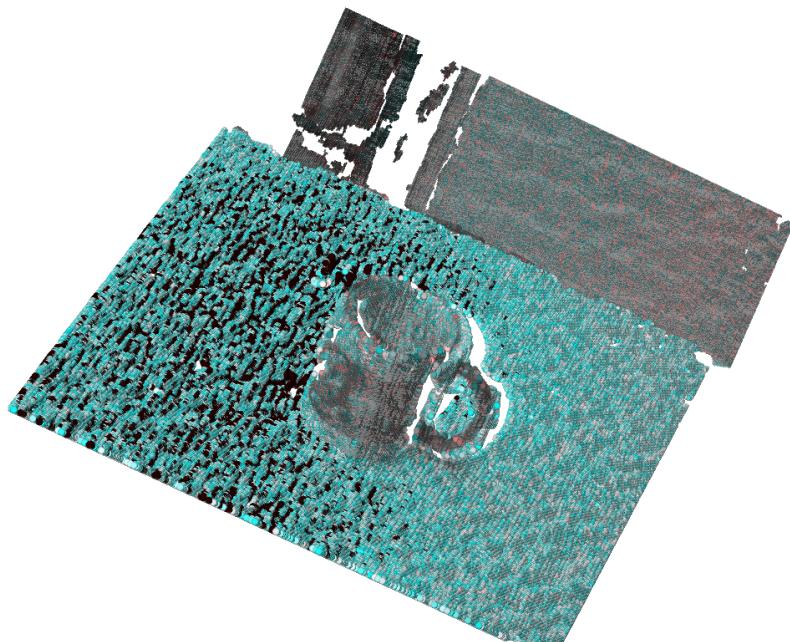
Open-source framework vytvorený spoločnosťou PrimeSense pre 3D Natural Interaction senzory od spoločnosti napríklad Asus Xtion, spoločnosť stojí za lincencovaným dizajnom hardwaru a čipov použitých priamo v Kinecte. Druhá verzia OpenNI sa oproti prvej ma znížiť zložitosť API tým že zložité funkcie sa presunuli do middlewaru a aby funkcie na komunikáciu so senzormi boli jednoduchšie na pochopenie a zároveň ma 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 napokialko táto publikácia je spracúvaná na operačnom systéme Linux je potrebné použiť OpenNI2 SDK, ktoré by malo pracovať aj s inými druhmi kamier.[6]



Obr. 5.2: Architektúra softwerového vývoja

5.4 Point cloud (PCD) súbor

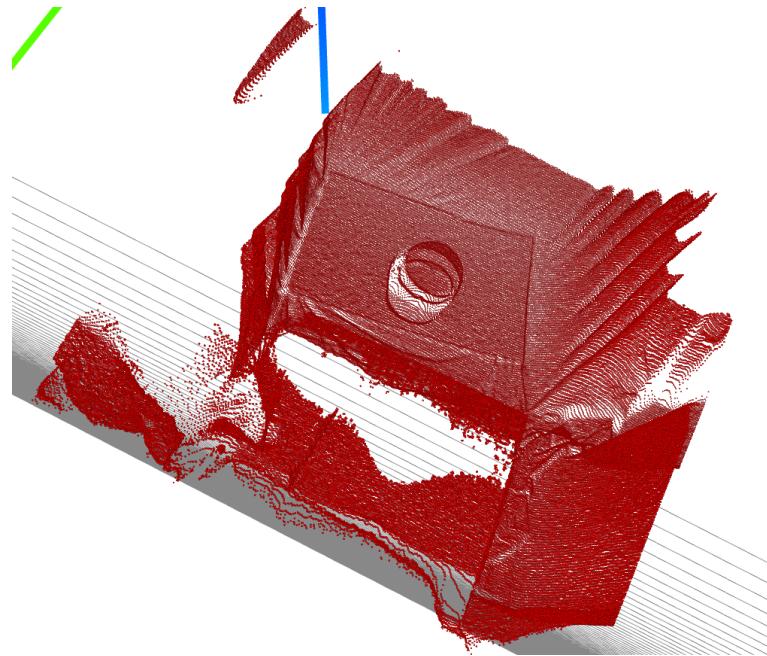
Pomocou OpenNI2 implementovaného v programovacom jazyku C++ a knižnicou PCL je nadviazaná komunikácia s Kinectom, ktorý streamuje dátu a sú vyobrazené v okne OpenNI2 a po úprave knižnice libfreenect s implementovaným opencv, sa získane dátu ukladajú ako PCD súbor potrebný na vyhľadávanie tvarou pomocou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



Obr. 5.3: Príklad výstupný Point Cloud z hĺbkovej kamery

5.5 Výstup z Kinectu

Kinect ma horšie vzorkovanie oproti použitému PCD. Algoritmy mali problém odhadnuť objekty z získanej scény, preto boli zvolené dátá v prací namodelované a taktiež na namodelovaných dátach bol pridaný šum, na co najpresnejšiu kópiu reálneho sveta.



Obr. 5.4: Aktualny výstup z kinectu

6 Implementovanie RANSAC algoritmu

Budeme postupovať pomocou krokov:

1. PointCloudData .pcd súbor
2. Zvolenie geometrického modelu
3. Definovanie parametrov
4. implementovanie RANSACu
5. Point Cloud Binary Segmentation

6.1 PointCloudData

Pre praktickejšiu prácu s PCD na začiatku odstránia všetky body ktoré sú vzdialenejšie ako 1,5 metra a odhadneme normaly povrchu v každom bode. Odfiltrovaný model je uložený v samostatnom PCD súbore. [7]

6.2 Zvolenie geometrického modelu

Vyberieme si geometricky model, čo bude zodpovedať trom náhodným bodom aby sme vedeli zstrojíť rovinu. V blízkosti roviny vo vopred definovanom prahovej hodnote(threshold) spočítame kol'ko bodov leží v blízkosti roviny. Vyberieme rovinu ktorá ma najväčší počet inlierov ako najlepší model. Postup sa opakuje pokiaľ neprejde algoritmus presný počet interácií zvolených na začiatku. V kóde je zvolený model valca ktorý ma v 3D priestore predpis

$$(y - z)^2 + (z - x)^2 + (x - y)^2 = 3R^2 \quad (6.1)$$

$$(f(x - a/c * z, y - b/c * z) = 0) \quad (6.2)$$

kde konštanty a,b,c sú zložky normálového vektora n' , ktorý je kolmý na rovinu a ktorý kol'vek rovnobežný vektor s rovinou. Z toho nám vyplýva, že bod $p=(x,y,z)$ patrí do roviny vektora n' ak splňa rovnicu. [7]

6.3 Definovanie parametrov

V kóde sa na začiatku nastavili požadované hodnoty, čiže hľadaný model v kóde nastavený na valec, vzdialenosť bodov pre vyhodnotenie RANSACA ako inlier 5cm, vplyv normál na váhu 0.1 a nastavili limit modulu aby bol menší ako 10cm. Hodnôt treba na začiatku otestovať viacero a vybrať ktoré majú najväčšiu presnosť alebo rýchlosť. je možné skúsiť automatické nastavovanie parametrov, čo by teoreticky mohlo fungovať na základe vypočítania strednej hodnoty vzdialenosť medzi susediacimi bodmi. [7]

6.4 Implementovanie RANSACu

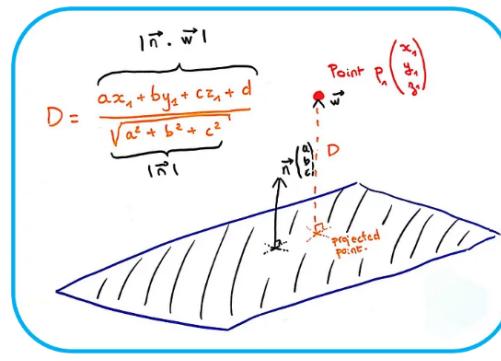
- Hľadanie roviny(1 opakovanie)

Najprv sa zvolia 3 body z PCD, z ktorých sa má vytvoriť rovnica roviny nájdením parametrov a,b,c,d. K ním sa dá dopracovať lineárnow algebrou, konkrétnie krížovým súčinom dvoch vektorov ktoré generujú ortogonálny vektor. Treba definovať vektory z rovnakého bodu v rovine a vypočítat' normálu k nim, ktorá definuje normálu roviny. Pomocou normál normalizujeme náš normálový vektor a získame parametre a,b,c a dopočítame parameter d čo je posun roviny od počiatku.

- Bod k rovine: definovanie thresholdu

$$D = \frac{|\vec{n} \cdot \vec{w}|}{|\vec{n}|} = \frac{ax_i + by_i + cz_i + d}{\sqrt{a^2 + b^2 + c^2}} \quad (6.3)$$

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



A normal n defines the plane, and we can see what the distance point-to-plane we want to compute looks like. © F. Poux

Obr. 6.1: Grafické znázornenie rovnice 6.3

- Opakovanie Vytvorenie cyklu ktorý sa bude opakovat' počet interacii a porovnávať, či aktuálny model je lepší než doteraz najlepší model a výstupom bude najlepší fitting RANSAC model.

[7]

6.5 Vyhodnotenie 3

Z predošlého kroku výstupom je set bodov opisujúcich valec a jeho parametre. Algoritmus RANSAC z knižnice 3 vracia bod reprezentujúci bod na osi x y z, body reprezentujúce rotáciu

valca a polomer valca. Presnosť vyhodnocovania 3 v tejto práci je zahrnutá na predpovedanie 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 6.1: Presnosť vyhodnocovania 3 s premiestnením predmetu v priestore

Posun po X osi [m]	Posun po Y osi [m]	Polomer vyhodnotený PCL	Chyba [cm & %]
0	0	98.56	1.44
0	3	103.56	3.56
0	-3	103.44	3.44
3	0	103.48	3.48
3	3	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

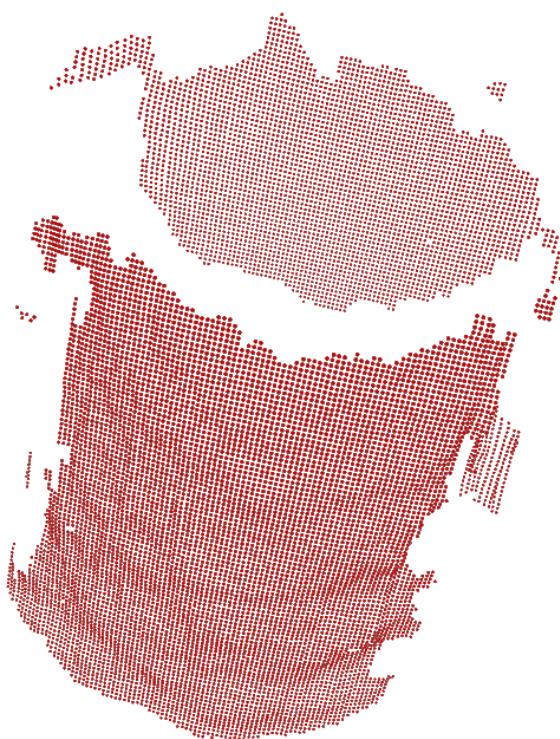
V druhom riadku 6.1 vyšla vysoká chyba odhadu pretože teleso malo veľký šum a algoritmus nevedel presne odhadnúť valec a zachytil aj iné body takže stred valca určil nepresne a taktiež polomer.

Tabuľka 6.2: Presnosť vyhodnocovania 3 na rôznych polomeroch telesa. * označuje teleso položené na rovine, bez * je iba teleso v priestore

Reálny polomer [cm]	Polomer inliers 0.5	Polomer inliers 0.6	Polomer inliers 0.8	Stred objektu [x y z]	Chyba [%]
10	13.85	12.41	7.11	0.0000 0.0002 1.412	30.48
10*	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	59.08	75.11	61.12	-0.0257 -0.0135 0.8796	30.20
50*	59.49	66.80	59.96	-0.0073 0.0008 1.0662	17.72
100	112.66	100.78	100.55	0.0026 0.0084 0.9749	5.21
100*	98.56	99.15	100.55	0.0417 0.0082 1.0995	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					

V 6.2 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 inliers. Pri nižšej hodnote nevedel algoritmus určiť väčšie telesa, pretože počet bodov na teleso ostával stále rovnaký. Naopak pri väčšom rozostupe inlierov, pri PCD s rovinou, vznikali odchýlky, kvôli šumu ktorý algoritmus zachytil od roviny. Najlepšie výsledky pri rôznych telesách vyšli v stĺpci 3 v 6.2, pretože telesa odhadlo všade s menšou odchýlkou, ale nevznikli veľké odchýlky ako v ostatných stĺpcach.

S použitím PCD z 5.3 výstupom algoritmu so vstupným parametrom na získavanie valca je vonkajšia a vnútorná rovina hrnčeka. PCD bol stiahnutý z internetu z 3 dokumentácie.



Obr. 6.2: Výstup algoritmu RANSAC v knižnici PCL

7 Implementovanie RANSAC algoritmu s použitím neurónovej sieti

NG-RANSAC, známy aj ako NG-RANSAC, je flexibilná metóda, ktorá vytvára robustné odhady prispôsobením parametrických modelov súborom údajov, ktoré môžu obsahovať šum alebo odľahlé hodnoty. Táto aplikácia Súbor riedko zarovnaných korešpondencií medzi dvoma obrazmi a epipolárnymi geometriami, ako je základná alebo esenciálna matica, môže byť prispôsobený NG-RANSAC. 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 konvenčnom 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 rôznym uhla pohľadu. Výstupnými dátami bol 7.1, ktorý bol konvertovaný do šedej pre rozšírenie údajov na a nájdený model s počtom inliers.

Model nájdený pomocou RANSAC:

$$\begin{vmatrix} 6.418e^{-7} & -7.301e^{-2} & 2.113e^{-2} \\ 1.227e^{-1} & -2.326e^{-2} & -6.957e^{-1} \\ -3.079e^{-2} & 7.021e^{-1} & -2.695e^{-2} \end{vmatrix}$$

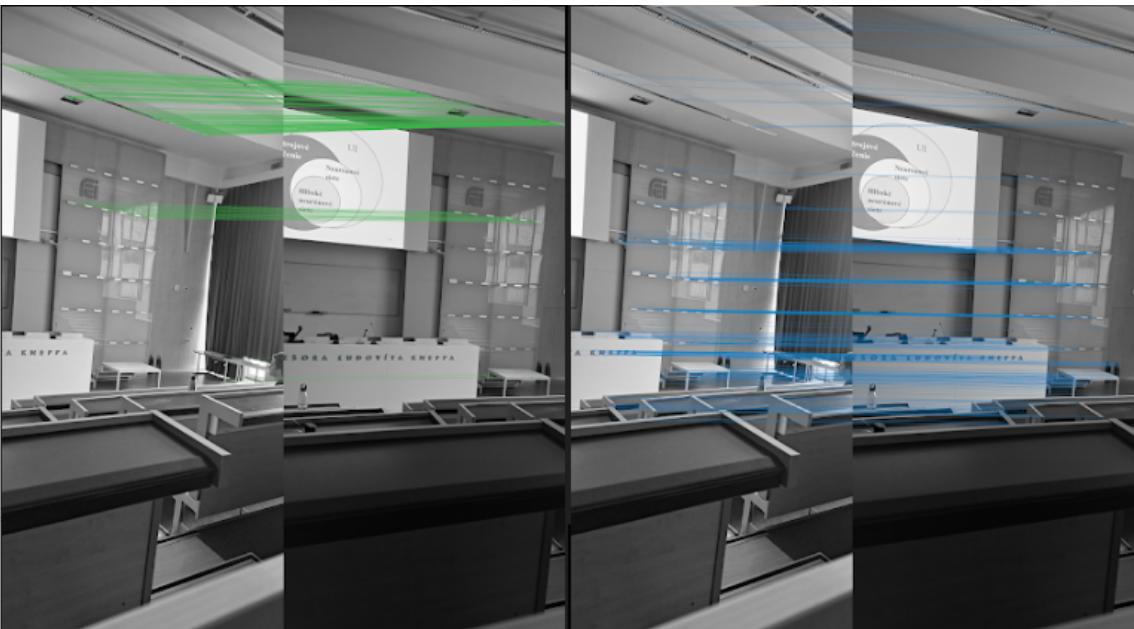
Počet inlierov: 251.

Model nájdený pomocou NG-RANSAC:

$$\begin{vmatrix} -3.409e^{-3} & 2.083e^{-2} & 3.740e^{-2} \\ 3.847e^{-2} & 4.066e^{-3} & -7.051e^{-1} \\ -4.590e^{-2} & 7.053e^{-1} & 4.446e^{-2} \end{vmatrix}$$

Počet inlierov: 297.

Na 7.1 ľavá dvojica predstavuje model nájdený pomocou RANSAC algoritmu a pravá dvojica model nájdený pomocou NG-RANSAC. NG-RANSAC vyhľadal podobnosti s vyššou presnosťou, čiže na testovacích dátach našiel väčší počet inliers. [5]



Obr. 7.1: Výstup Neural guided RANSAC

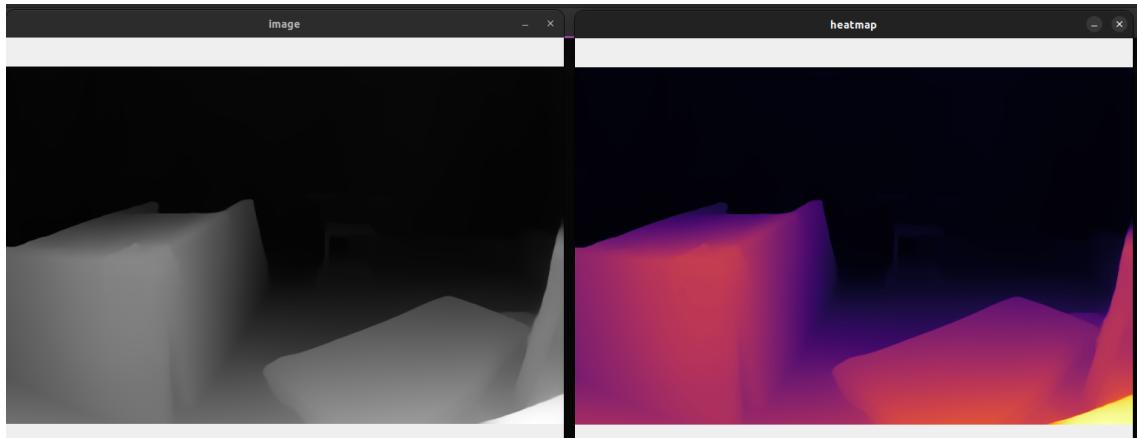
Tabuľka 7.1: Odhad základných matíc

Trénovací objekt	%Inliers	F-score	Priemerná	Stredná
RANSAC	-	21.85	13.84	0.35
USAC	-	21.43	13.90	0.35
Deep F-Mat	Priemerná	24.61	14.65	0.32
NG-RANSAC	Priemerná	25.05	14.76	0.32
NG-RANSAC	F-score	24.13	14.72	0.33
NG-RANSAC	%Inliers	25.12	14.74	0.32

7.2 ngdsac cameralloc

Táto práca vychádza z hotovej implementácie NN ku RANSAC-u. Pomocou práce sa zistovala presnosť určenia podobnosti na 2D dátach použitím RANSAC algoritmu a jeho vylepšenia pomocou NN. Súčasťou práce bolo aj zistovanie polohy kamery, ktorá mala byť implementovaná. Použitý bol dataset z University of Cambridge **cite dačo**, 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 hĺbkovej kamery ktoré sme pre lepšiu čitateľnosť pretransformovali na heat mapu 7.2, s reálnou polohou kamery, ale nebolo možné s

vytvoreným datasetom natrénoval' NN, pretože NN 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 natrénoval' siet' na 2D dátach. Dáta sa používali na nastavovanie Pythonovskej knižnice tensorflow. [8] [5]



Obr. 7.2: Snímka z hĺbkovej kamery Kinectu a snímka pretransformovaná na heat mapu

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

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 môžno vidieť chybu strednej hodnoty odhadu pozície použitia algoritmov. Z výsledkov výplýva, že diferenciálny RANSAC rozšírený o neurónovú siet' na väčšine datasetu určoval presnejšie polohu kamery.

7.3 SFPN

Autorský tím sa v tomto článku zameriava najmä na problémy s prístupmi založenými na RANSAC, ktoré sú priemyselným štandardom pre fitovanie primitív, ale vyžadujú dôkladné ladenie parametrov pre každý vstup a t'ažko sa škálujú pre veľké súbory údajov s rôznorodými formami.

Riešením týchto problémov, ktoré autori navrhujú, je Supervised Primitive Fitting Network (SPFN), koncová neurónová siet' 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 trénovanie 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 špičkovými metódami RANSAC a priamym neurónovým predpovedaním pomocou tréningu a hodnotenia navrhovanej metódy s použitím ich nových súborov údajov, 3D modelov mechanických komponentov ANSI.

Tím testoval 7.3 pri spracovaní vstupných mračien bodov s vysokým rozlíšením rozšírených o šum pri testovaní, hoci siet' bola vycvičená na mračnách bodov s nižším rozlíšením. Tieto testy ukázali, že 7.3 funguje efektívne aj za týchto okolností. [9] [5]

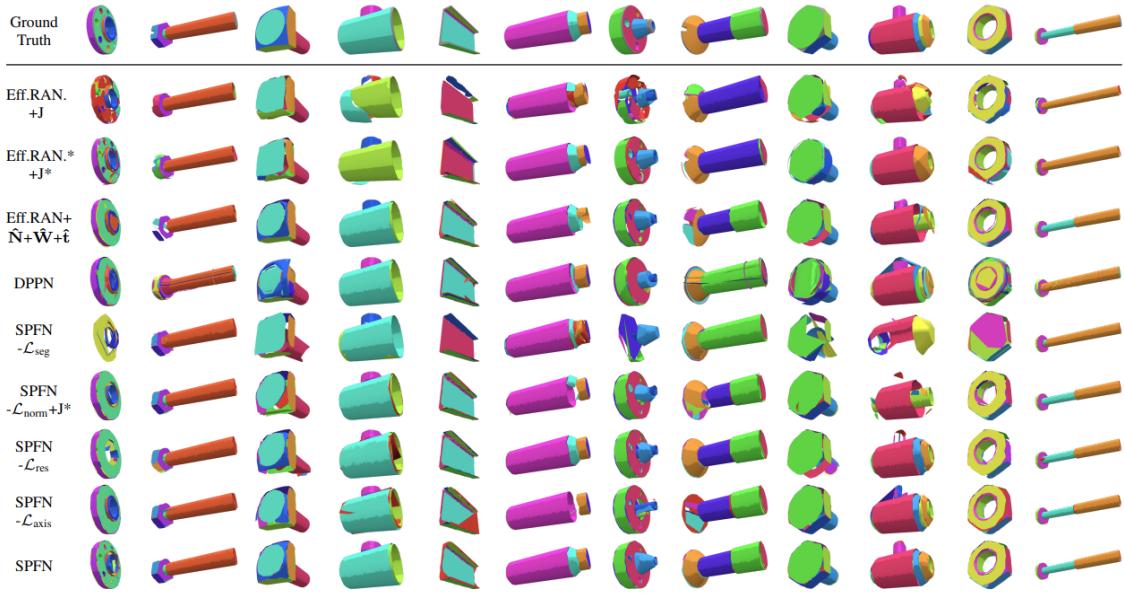
Tabuľka 7.3: Presnosť určovania SPFN oproti Efektívnomu RANSAC

Metóda	Seg. (Mean IoU)	Primitive Type (%)	Point Normal (°)	Primitive Axis (°)
<i>Eff.RANSAC + J</i>	43.68	52.92	11.42	7.54
<i>Eff.RANSAC* + J*</i>	56.07	43.90	6.92	2.42
<i>Eff.RANSAC + J*</i>	45.9	46.99	6.87	5.85
<i>Eff.RANSAC + J* + \hat{W}</i>	69.91	60.56	6.87	2.90
<i>Eff.RANSAC + J* + $\hat{W} + \hat{t}$</i>	60.68	92.76	6.87	6.21
<i>Eff.RANSAC + $\hat{N} + \hat{W} + \hat{t}$</i>	60.56	93.13	8.15	7.02
<i>DPPN(Sec.4.4)</i>	44.05	51.33	-	3.68
<i>SPFN - L_{seg}</i>	41.61	92.40	8.25	1.70
<i>SPFN - $L_{norm} + J^*$</i>	71.18	95.44	6.87	4.20
<i>SPFN - L_{res}</i>	72.70	96.66	8.74	1.87
<i>SPFN - L_{axis}</i>	77.31	96.47	8.28	6.27
<i>SPFN($\hat{t} - > Est.$)</i>	75.71	95.95	8.54	1.71
<i>SPFN</i>	77.14	96.93	8.66	1.51

Tabuľka 7.4: Presnosť určovania SPFN oproti Efektívnomu RANSAC [10]

Metóda	$\{S_k\}$ Residual	$\{S_k\}$ Coverage		P Coverage	
	Mean \pm Std.	$\epsilon = 0.01$	$\epsilon = 0.02$	$\epsilon = 0.01$	$\epsilon = 0.02$
<i>Eff.RANSAC + J</i>	0.072 \pm 0.361	43.42	63.16	65.74	88.63
<i>Eff.RANSAC* + J*</i>	0.067 \pm 0.352	56.95	72.74	68.58	92.41
<i>Eff.RANSAC + J*</i>	0.080 \pm 0.390	51.59	67.12	72.11	92.54
<i>Eff.RANSAC + J* + \hat{W}</i>	0.029 \pm 0.234	74.32	83.27	78.79	94.58
<i>Eff.RANSAC + J* + $\hat{W} + \hat{t}$</i>	0.036 \pm 0.251	65.31	73.69	77.01	92.57
<i>Eff.RANSAC + $\hat{N} + \hat{W} + \hat{t}$</i>	0.054 \pm 0.307	61.94	70.38	74.80	90.83
<i>DPPN(Sec.4.4)</i>	0.021 \pm 0.158	46.99	71.02	59.74	84.37
<i>SPFN – L_{seg}</i>	0.029 \pm 0.178	50.04	62.74	62.23	77.74
<i>SPFN – $L_{norm} + J^*$</i>	0.022 \pm 0.188	76.47	81.49	83.21	91.73
<i>SPFN – L_{res}</i>	0.017 \pm 0.162	79.81	85.57	81.32	91.52
<i>SPFN – L_{axis}</i>	0.019 \pm 0.188	80.80	86.11	86.46	94.43
<i>SPFN($\hat{t} - > Est.$)</i>	0.013 \pm 0.140	85.25	90.13	86.67	94.91
SPFN	0.011\pm0.131	86.63	91.64	88.31	96.30

V tabuľke 7.3 a 7.4 sú uvedene výsledky algoritmu SPFN v porovnaní s efektívnym RANSAC. Dáta sú testované na identických PCD s rozlišením 8k bodov riadok 1 v tabuľkách 7.3 a 7.4 a hviezdička 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 S_k a P prekrytie s thresholdom $\epsilon = 0.01$ vykázali 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 L_{seg} a pre každý segment v predpovedanej matici prislušnosti \hat{W} použili efektívny RANSAC na predpovedanie primitívnej vlastnosti riadok 4 v tabuľkách 7.3 a 7.4. Po pridaní L_{type} a L_{norm} strát pri trénovaní za sebou a použili predpovedané primitívne typy \hat{t} a body normál \hat{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 S_k a P pokrytie pre efektívny RANSAC sa podstatne zvýšilo, ale stále nižšie ako SPFN metoda. Predpovedané normály a primitívne typy neurónovou siet'ou nezlepšili S_k a P pokrytie v RANSAC.



Obr. 7.3: Výsledky pokrycia primitív rôznymi metódami

7.4 Efektívny RANSAC

V práci sa zameriavali na jednoduché geometrické tvary ako sú roviny, gule, valce, kužeľe a torus. Každý tvar ma rôzny počet parametrov od troch po sedem. Každý vybraný bod určuje 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.[10]

7.5 Naše výstupy

Po natrénovaní siete sme vložili vlastné dátá 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 nás dataset je vytvorený na jednoduché objekty položené na podložke. Odchýlky taktiež mohlo spôsobiť vyššie rozlíšenie objektov, pretože v práci je napísane, že očakávajú 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 K_{max} ale nedosiahli lepšie výsledky, ako počtom K_{max} , ktorý sme zvolili my pri trénovaní siete. K_{max} reprezentuje

maximálny počet primitív.

Tabuľka 7.5: Presnosť vyhodnocovania SPFN

Posun po X osi [m]	Posun po Y osi [m]	Polomer vyhodnotený SPFN	Chyba [cm & %]
0	0	78.31	21.69
0	3	103.31	3.31
0	-3	82.41	17.59
3	0	78.31	21.68
3	3	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 7.5 vidno chybu vyhodnocovania polomeru odhadovaného telesa s premiestnením v priestore. 7.3 mal problém vyhodnotiť polomer objektu presúvaný v priestore. Vznikajú vyššie odchýlky v niektorých riadkoch, kvôli zašumieniu vstupných dat, ktoré mali pod sebou podložku co mohlo spôsobiť zle vyhodnotenie polomeru.

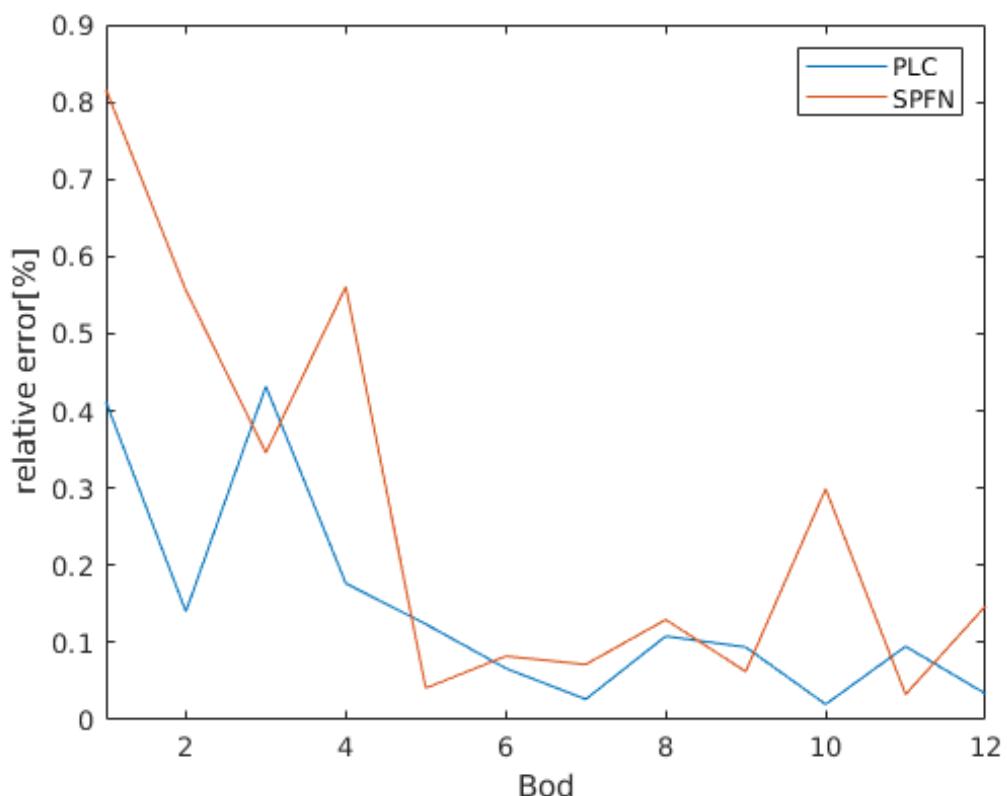
Tabuľka 7.6: Presnosť vyhodnocovania SPFN na rôznych polomeroch telesa. * označuje teleso položené na rovine, bez * je iba teleso v priestore

Reálny polomer [cm]	Polomer [cm]	Stred objektu [x y z]	Chyba [%]
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 7.6 možno vidieť odhad polomeru a stredu rovnakého telesa ako v 6.2. Neurónová siet' ako bolo spomenuté vyššie bola natrénovana na telesách s väčšími rozmermi, takže algoritmus začal lepšie odhadovať polomer až po dosiahnutí, väčších rozmerov. Po dosiahnutí rozmeru sa chyba výrazne znížila. Ked'že siet' bola trénovaná na jednoduchých telesách v priestore, vložené dátá s rovinou (riadky s *) mali výrazne vyššiu chybu.

8 Výstupy

Z nameraných hodnôt z 3 a 7.3, sme zostrojili tabuľky 6.1, 6.2 a 7.5, 7.6, kde porovnávame namerané výsledky. Pri tabuľkách 6.1 a 7.5 vstupnými dátami bol valec s polomerom 1 meter, pričom vstupný PCD obsahoval šum. Tabuľky 6.2 a 7.6 vstupom boli PCD bez sumu, pričom objekt neobsahoval šum a sledovali sme presnosť vyhodnocovania oboch implementácií. Z odhadnutých stredov objektov sme vypočítali relatívnu odchýlku, ktorú sme graficky znázornili v 8.1. V tabuľke možme pozorovať ak sa jednalo o PCD bez roviny pod objektom RANSAC implementovaný o NN odhadoval rozmer telesa presnejšie ako 3. Implementácia 3 s jedným nastavením algoritmu mal problém v rozdielnych veľkostach objektov. Pre používateľa ktorý nemá naštudovanú problematiku ohľadom RANSAC, by mohol mať problém s optimálnymi nastaveniami algoritmu. Na druhu stranu nainštalovanie 3 je jednoduchšie a vyžaduje menej výkonný hardware ako 7.3. Po natrénovaní siete vyhodnocovanie vstupných dat pomocou 7.3.



Obr. 8.1: Relatívny error implementácií

Záver

Úspešne sa nám podarilo spojazdníť nami zvolené implementácie na rozpoznávanie objektov s použitím algoritmu RANSAC a jeho rozšírenie o NN. Klasickým algoritmom sme získali jednoduché objekty s odchýlkou 20%, čo zapríčinil šum na vstupných PCD. Na dátach z Kinectu sa nám neporilo nájsť žiadane objekty, pretože Kinect mal nižšie rozlíšenie a na výstupnom PCD nebolo dostať bodov reprezentujúcich objekt, takže nám za kužeľ identifikovalo prevažne kúty, aj s rôznymi nastaveniami RANSAC.

Implementácie s neurónovými sietami 7.1 a 7.2 spracováva 2D dátu, čo nevyhovovalo zadaniu. Vďaka týmto implementáciám som sa lepšie naučil pracovať s Linuxom a odporúčam používať virtuálne environment, pretože pri práci so staršími Python baličkami, je možné že si na novšom operačnom systéme prepíšete systémové premenné, alebo nechcene inštalačným skriptom odinstalujete Python.

Práca 7.3 bola prispôsobená na starší hardware, takže som sa v prací naučil pracovať s platformou Docker, ktorá funguje podobne ako virtuálny environment, ale naroziel od environment sa tam ľahšie menia python baličky a ľahšie pristupovať k starším verziám pythonu. Naučil som sa, že nie je vhodné ho inštalovať ako root používateľ, pretože pri komplikácii projektu vytvára docker súbor o veľkosti projektu, takže po niekoľkých komplikáciách sa mi zaplnil root priečinok a nevedel som spúšťať aplikácie a po reštarte som sa nevedel dostať do operačného systému.

V presnosti vyhodnocovania možno pozorovať, že oba algoritmy majú podobne odchýlky, ale 7.3 po dosiahnutí väčšieho rozmeru telesa výrazne znížilo chybu odhadu, kde pri 6.2 pri rôznorodosti vstupných dát treba zmeniť nastavenia siete, čo je neefektívne. Na druhu stranu 7.3, by stačilo dotrénovať na menšie rozmery telies, aby vedelo presnejšie approximovať menšie telesá. V konečnom dôsledku oba algoritmy majú skvele využitie v reálnom svete na rozpoznávanie, ale 3 má lepšiu dokumentáciu, tak práca s knižnicou bola jednoduchšia a ľahšie sa implementovať na vlastné dátu, pretože knižnica pracuje s bežne používanými súbormi, oproti riešeniam s neurónovou sietou, kde siet požadovala vstupne dátu v konkrétnom tvare, čo bolo zložitejšie napodobniť.

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 kvalifikačních prací* [online]. MIT: MIT News Office, 2014-04-14 [cit. 2014-04-14]. Dostupné z : <https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414>.
3. *Point cloud library.* Dostupné tiež z: <https://pointclouds.org/>.
4. 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_NEZHQAQAAJ&citation_for_view=U_NEZHQAQAAJ:u-x6o8ySG0sC.
5. BRACHMANN, Eric a ROTHER, Carsten. Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses. In: *ICCV*. 2019.
6. FAIRHEAD, Harry (zost.). *OpenNI 2.0 - Another Way To Use Kinect* [online]. i-programmer.info, 2012-12-22 [cit. 2012-12-22]. Dostupné z : <https://www.i-programmer.info/news/194-kinect/5241-openni-20-another-way-to-use-kinect.html>.
7. 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-model-fitting-for-point-clouds-with-ransac-and-python-2ab87d5fd363>.
8. BRACHMANN, Eric a ROTHER, Carsten. Learning less is more - 6D camera localization via 3D surface regression. In: *CVPR*. 2018.
9. LI, Lingxiao, SUNG, Minhyuk, DUBROVINA, Anastasia, YI, Li a GUIAS, Leonidas. *Supervised Fitting of Geometric Primitives to 3D Point Clouds.* 2018. Dostupné z eprint: arXiv:1811.08988.

10. RUWEN SCHNABEL, Roland Wahl a KLEIN, Reinhard. *Efficient RANSAC for point-cloud shape detection*. Computer graphics forum, 2007.

Prílohy

A	Kód na získanie PCD z Kinectu	42
B	Matlab scripty	43
C	Vstupné data pre algoritmy	44
D	Návody na spušťanie algoritmov	45

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 << "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" << 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 hĺbkovej 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 čase, aby kameru bolo možné vhodne napolohovať a tak uložiť PCD.

B Matlab scripty

pcl/

- Priečinok s PCL programami
cylinder.cpp
 - Algoritmus na nájdenie valca v 3

matlab/

- Priečinok s matlab skriptami
cylinder_center.m
 - Výpočet stredu valca na základe parametrov získaných RANSAC z 3
- Skript na výpočet relativnej chyby určovania bodov
errors.m
 - Skript na výpočet relativnej chyby určovania bodov

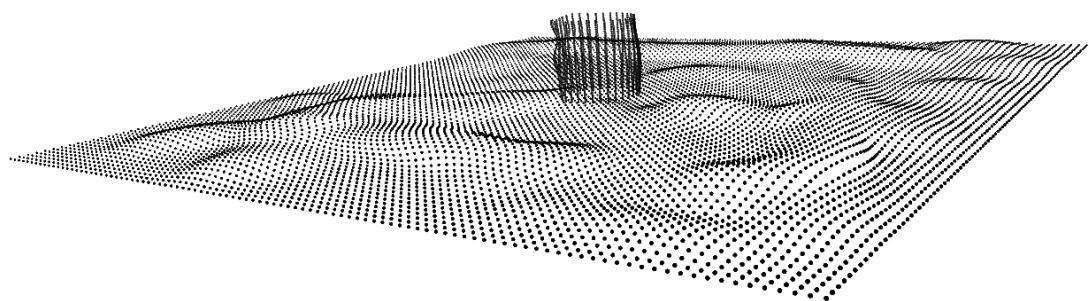
python/

- Priečinok s python skriptami
list.py
 - skript na vypísanie parametrov výstupného súboru z 7.3
- Skript na vytvorenie PCD s vyšším rozlíšením
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 čase z knižnice libfreenect2

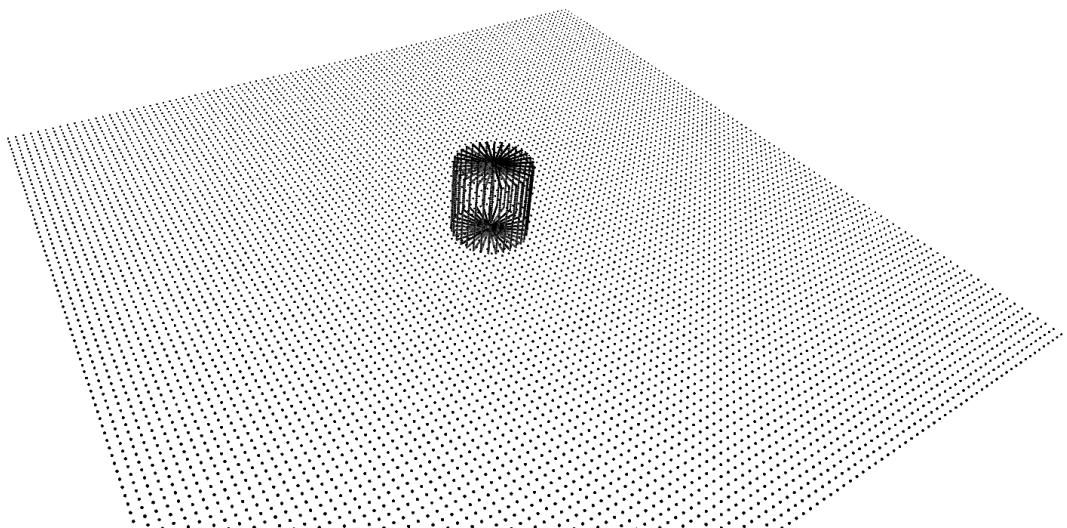
C Vstupné data pre algoritmy



Obr. C.1: Vstupné mračno bodov so šumom

pcd/pcd_so_sumom.pcd

- Mračno bodov so šumom



Obr. C.2: Vstupné mračno bodov bez šumu

pcd/pcd_na_velkost_radiusu.pcd

- Mračno bodov bez šumu

D Návody na spušťanie algoritmov

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

```
$ ./cylinder <nazov_vstupneho_pcd>
```

Skripty v Matlabe je potrebne mat nainštalovaný Matlab a pripravene potrebne dátu. Skript cylinder_center.m prijíma dátu z programu spominaného vyssie a je potrebne mu vložiť získane parametre do premennej "values" na ich určenú pozíciu. Skript errors.m ma v programe zadefinovaný jeden bod ktorý zodpovedá reálnej polohe objektu. Súradnice je potrebne zapísat' do premennej "real_pointä body odhadnuté pomocou algoritmu do premennej approx_point", v ktorej je možne definovať viac bodov. Skript následne vypočíta relatívnu chybu a graficky ju znázorni.

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

```
$ python3 list.py <nazov suboru s .h5 priponou>
```

Skript pcd.py sa používa na získavanie PCD, z Kinectu. Nevýhodou tohto skriptu je že používateľ nevidel výstup z Kinectu v reálnom čase, 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 vidi záznam z kamery v reálnom čase a vie si prispôsobiť scénu, ktorá následne sa uloží na disk. Súbor je potrebne skompilovať spolu s knižnicou a po skompilovaní sa vytvori spustiteľný súbor, ktorý na obrazovke otvorí okno, kde možno vidieť záznam z RGB kamery, IR kamery, hĺbkového senzoru a výstupný ??.

Sekcia 7.3 počas testovania používa kód od autora témy. Návod na používanie a potrebne príkazy na spustenie sú popísane autorom.