

**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**

**2023**

**Maroš Kocúr**

**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: prof. Ing. Jarmila Pavlovičová, PhD.  
Konzultant: 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

# SÚHRN

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:	prof. Ing. Jarmila Pavlovičová, PhD.
Konzultant:	Ing. Miroslav Kohút
Miesto a rok predloženia práce:	Bratislava 2023

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 opísat' 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.

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:	prof. Ing. Jarmila Pavlovičová, PhD.
Consultant:	Ing. Miroslav Kohút
Place and year of submission:	Bratislava 2023

Based on the growing need to work with 3D data, there is constant testing and research of new algorithm possibilities. The current work is devoted to the RANSAC algorithm, with the help of which it is possible to mathematically describe various groupings of points. The models can then be used for further processing of 3D data. The aim of the work is to compare the implementation of the analytical method of the RANSAC model with its extension, which also implements neural network models to achieve better results.

Keywords: RANSAC, PCL, neural network

## **Pod'akovanie**

I would like to express a gratitude to my thesis supervisor.

# **Obsah**

<b>Úvod</b>	<b>12</b>
<b>1 PCL</b>	<b>13</b>
1.1 Pouzitie . . . . .	13
1.2 Befor installation . . . . .	13
1.3 Inštalácia PCL . . . . .	13
1.4 Používanie PCL . . . . .	14
1.5 Kompilácia projektu . . . . .	14
1.6 Nastavovanie parametrov Random sample consensus (RANSAC)-u . . . . .	14
<b>2 Point Cloud</b>	<b>15</b>
2.1 General . . . . .	15
2.2 Pouzitie . . . . .	15
2.3 Konverzia do 3D povrchu . . . . .	15
<b>3 Neural Network</b>	<b>16</b>
3.1 History . . . . .	16
3.2 What is it . . . . .	16
3.3 Vývoj . . . . .	16
<b>4 RANSAC</b>	<b>18</b>
4.1 Overview . . . . .	18
4.2 Algoritmus . . . . .	18
4.3 RANSAC 3D . . . . .	19
4.4 Neural guided RANSAC . . . . .	19
<b>5 KINECT</b>	<b>21</b>
5.1 Ako funguje . . . . .	21
5.2 Výstup z KINECTU . . . . .	22
5.2.1 List potrebných Knižnic pre Ubuntu 22.04 LTS . . . . .	22
5.3 OpenNI 2.0 . . . . .	22
5.4 Point cloud (PCD) súbor . . . . .	22
<b>6 Implementovanie RANSAC algoritmu</b>	<b>24</b>
6.1 1. PointCloudData . . . . .	24
6.2 2. Zvolenie geometrického modelu . . . . .	24

6.3	3. Definovanie parametrov . . . . .	24
6.4	4.implementovanie RANSACu . . . . .	25
6.5	5. Vyzualizácia PointCloudu . . . . .	26
<b>7</b>	<b>Implementovanie RANSAC algoritmu s použitím neurónovej sieti</b>	<b>27</b>
7.1	ngransac . . . . .	27
7.2	ngdsac cameraloc . . . . .	28
7.3	SFPN . . . . .	29
<b>8</b>	<b>Vystupy</b>	<b>32</b>
8.1	POZNAMKA . . . . .	32
8.2	Alligment . . . . .	33
	<b>Záver</b>	<b>35</b>
	<b>Zoznam použitej literatúry</b>	<b>36</b>

# Zoznam obrázkov a tabuliek

Obrázok 3.1	Priklad neuronovej siete . . . . .	17
Obrázok 4.1	Priklad algoritmu v 2D rovine . . . . .	18
Obrázok 5.1	Kinect v2 . . . . .	21
Obrázok 5.2	Architektúra softwarového vývoja . . . . .	23
Obrázok 5.3	Výstupný Point Cloud z Kinectu . . . . .	23
Obrázok 6.1	Dajake rovnice co treba prepocitat a prekreslit . . . . .	25
Obrázok 7.1	Výstup Neural guided RANSAC . . . . .	28
Obrázok 7.2	Snímka z hlbkovej kamery Kinectu a snímka pretransformovaná na heat mapu . . . . .	29
Obrázok 8.1	Výstup algoritmu RANSAC v knižnici PCL . . . . .	32
Obrázok 8.2	Aktualny vystup z kinectu . . . . .	33
Obrázok 8.3	Spojene . . . . .	33
Tabuľka 7.1	Odhad základných matíc . . . . .	28
Tabuľka 7.2	Premiestnenie kamery v priestore . . . . .	29
Tabuľka 7.3	Premiestnenie kamery v priestore . . . . .	30
Tabuľka 7.4	Premiestnenie kamery v priestore . . . . .	31

# Zoznam skratiek

**NG-RANSAC** Neural Guided Random sample consensus

**NN** Neural Network

**PCD** Point Cloud

**RANSAC** Random sample consensus

# **Zoznam algoritmov**

# **Zoznam výpisov**

1	Vytvorenie PCD pomocou Kinectu . . . . .	34
---	--	----

# **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.

# 1 PCL

PCL je samostný, vysoko škalovateľný, otvorený projekt pre 2D a 3D obrázky a spracovávanie point cloud. Knižnica je cross platform, napisana 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 PCL je 1.12.1. Knižnica je vydaná pod BSD licenciami čo znamená, že je voľne použiteľna úre komerčné účely a za účelom výskumu.[1]

## 1.1 Pouzitie

PCL je open-source knižnica s algoritmami pre point cloud spracovanie úloh a spracovanie 3D geometrie, aké sa vyskytuje v trojdimenzionálnom strojovom videní. Knižnica má algoritmy na filtrovanie, odhad funkzie, rekonštrukcie povrchov, 3D registracie, prispôsobenie modelu, vyhľadávanie objektov a segmentácie. PCL ma vlastný format na ukladanie point cloud dát - Point Cloud (PCD), ale podporuje načítanie a ukladanie dát do rôznych iných formátov.[2] Algoritmy sa používajú na perception v robotike na filtrovanie zašumiených 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 point clodu a vykrel'ovať ich.

## 1.2 Befor installation

PLC potrebuje niekoľko knižnic tretích strán na fungovanie, ktoré musia byť nainštalované. Väčšina matematických operácií je implementovaných v Eigen knižnici. Vizualizačný model pre 3D point cloudy je na základe VTK. Knižnica Boost je použitá na zdieľanie pointov a FLANN knižnica pre rýchle hľadanie v okolí na základe algoritmu k-nearest. Ďalej sme nainštalovali OpenNI 2 knižnicu, ktorá nám zabezpečuje komunikáciu s Kinectom2

## 1.3 Inštalácia PCL

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

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

Na Windows sa PCL inštaluje pomocou vcpkg package manažéra vytvoreného Microsoftom.

```
PS> ./vcpkg install pcl
```

MacOS ma Homebrew package manažéra ktorý podporuje inštaláciu packagov, ktoré

Apple alebo Linux nedokáže nainštalovať. brew install pcl Toto su odporúčane inštalacie pre PCL na daných operačných systémoch.

## 1.4 Používanie PCL

Na používanie PCL si potrebujeme v našom kóde vložiť potrebné knižnice. Po nainštalovaní sa v našom systéme nastavia premenné, takže stačí nám použiť príkaz v C++, include <pcl/“názovknižnice“.h>.

## 1.5 Kompilácia projektu

Vytvoríme si súbor CMakeList.txt v ktorom zadefinujeme potrebné premenné aby make vedel najst' cestu ku knižnici a vedel aké súbory ma skompilovať. Ďalej vytvoríme priečinok s názvom build, v terminály vjdeme do neho a pomocou príkazu cmake “cesta k CMakeList.txt“, si vytvoríme makefile. Ďalej používame len príkaz make ktorý nam vytvorí súbor pomocov ktorého môžeme spustiť projekt.

## 1.6 Nastavovanie parametrov RANSAC-u

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 pointcloudu sa vyfiltruji iba body ležiace od 0 po 3.5 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 najst' zvolený objekt. Algoritmu sa nastavujú nasledovné parametre:

```
seg.setModelType(pcl::SACMODEL_CYLINDER); - matematický opis modelu ktorý algoritmus v pointcloude vyhľadáva  
seg.setMethodType(pcl::SAC_RANSAC); - algoritmus hľadania modelu  
seg.setNormalDistanceWeight(0.15); - váhy normál  
seg.setMaxIterations(10000); - počet opakovaní algoritmu  
seg.setDistanceThreshold(0.05); - prahová hodnota ohraničenia na určenie inlieru  
seg.setRadiusLimits(0.001, 1); - minimálny a maximálny rozmer telesa
```

Po nastavení parametrov a spustení algoritmu výstupom je vektor opisujúci teleso a rovinu.

**POPISAT ESTE TREBA**

## **2 Point Cloud**

### **2.1 General**

Je to súbor bodov v priestore. Body môžu reprezentovať 3D tvary alebo objekty. Každý bod má karteziánske súradnice (X,Y,Z). Point cloud je generovaný pomocou 3D skenera alebo pomocou softwaru na fotogrametriu, ktorý meria veľa bodov na externom povrchu objektov okolo. My v tomto projekte použijeme Kinect 2 (odsek 6). Point cloud sa používa v 3D modelovaní, metrológii, meranie kvality výrobkov a rôzne vizualizácie. Point cloud sa často zarovnáva s 3D modelmi alebo inými point cloudmi ako registrácia množín bodov. [3]

### **2.2 Pouzitie**

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

### **2.3 Konverzia do 3D povrchu**

V geografických informačných systémoch sú point cloudi jedným zo zdrojov využívaných na tvorbu digitálneho výškového medelu 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, PixčD alebo DroneDeploy používajú na vytvorenie RGB point cloutu, kde sa môže požiť vzdialenosná a objemová approximácia.

# **3 Neural Network**

## **3.1 History**

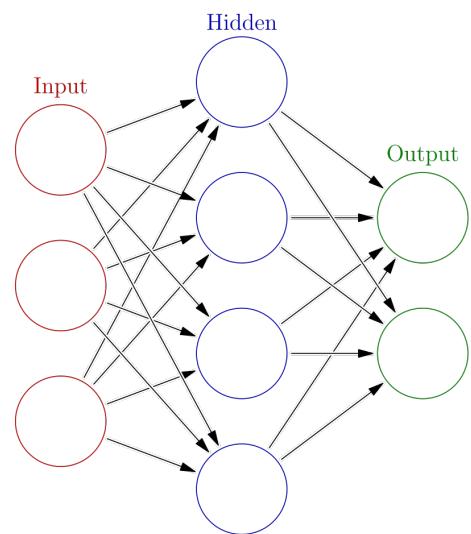
V posledných rokoch, najlepšie systémy s aplikaciou umelej inteligencie - ako sú rozpoznávače reči v mobilných zariadeniach alebo automatické prekladače majú dobré výsledky v technike nazývajúcej sa "deep learning". Deep learning je v podstate iné meno pre aplikaciu umelej inteligencie s názvom Neural network, ktorý sa vyvýja 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.

## **3.2 What is it**

NN sú myšlené na robenie strojového učenia, v ktorom sa počítač učí vykonávať úlohy na základe analyzovania trenováciích 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 siet' hľada vizuálne paterny v obrázku ktoré koré majú vzjomný 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 node a tie posielajú údaje vpred, čo znamená že dátu nimi prechádzaju iba v jednom smere. Jedna noda môže byť pripojená k viacerím nodam vo vrstvách pod ňou z ktorej príjma data a viacej node vo vrstve nad ňou kde spracované dátu posielajú. Ku každemu vstupnému prepoju sa pridelí číslo známe ako "váha". Ak je siet aktívna, noda príjme rôzne dátu z každého vstupu a vynásoby ich pridelenou váhou, potom všetky výsledky sčita ak je výsledne číslo pod threshold hranicou, noda nepošle žiadne dátu do ďalšej vrstvy. Ak číslo prekročuje threshold hodnotu, tak noda "výstrelí", čo znamená že pošle súčet vstupov na všetky výstupy. Keď sa NN trenuje, všetky váhy a thresholdy sú nastavené na náhodnú hodnotu. Trénovacie data sú posielané do spodnej vstupnej vrstvy a prechádzaju cez nasledujúce vrstvy, nasobia sa a sčítavajú sa v zložitými cestami, pokial' radikálne transformované nedojdu na vrchnú vstupnu vrstvu. Počas tréningu sa váhy a prahove hodnoty neustále upravuju kým treningové údaje s podobnými štítkami neprinášajú podobne výstupy. [4]

## **3.3 Vývoj**

Neuronove siete opísane v roku 1944 mali váhy a threshold hodnoty, ale neboli usporiadane do vrstiev a výskumnici nešpecifikovali trénovaci mechanizmus. Výskumnici dokázali ukázať, že NN dokáže v princípe vypočítať hocjakú funkciu ako digitálny počítač. Výsledkom bola viac neurová veda ako počítačová a cieľom bolo poukázať, že ľudslý mozog môžme považovať za výpočtové zariadenie.[4]



Obr. 3.1: Priklad neuronovej siete

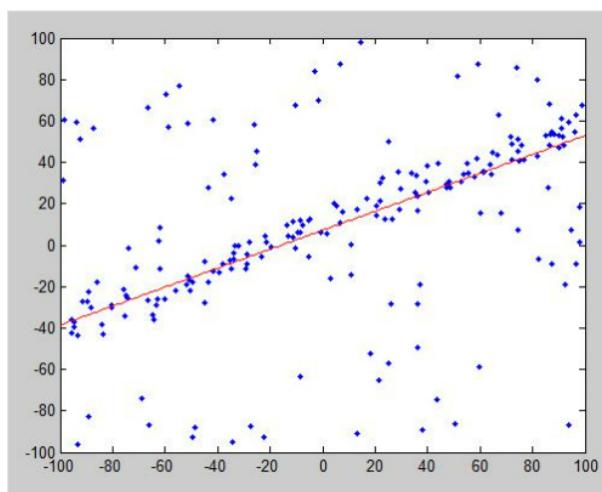
## 4 RANSAC

### 4.1 Overview

RANSAC je algoritmus vytvorený Fischlerom a Bollesom, určuje všeobecný prístup k odhadu parametrov, s veľkym podielom outliers v stupnom datasete. Na rozdiel od iných výkonných algoritmov odhadu, ako napríklad M-estimators a meródov najmenších štvorcov s prepojením na strojové učenie. RANSAC bol vytváraný komunitov ludi používajúci strojové učenie. RANSAC je vzorkovacia technika ktorá generuje kandidáta na minimalny pocet pozorovani potrebnych na zistenie odhadu parametrov ležiacich pod modelom. Naroďal od ostatných vzorkovacích algoritmov, ktoré používaju 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 koľko bodov z množiny všetkých bodov leží s preddefinovanou  $\epsilon$ .
4. Ak zlomok bodov ležiacich v preddefinovanej  $\epsilon$ , presahuje preddefinovaný prah  $\tau$ , prehodnotí parametre modelu použitím všetkých identifikovaných inliers a ukončí.
5. Inak, opakuj kroky 1 až 4, s maximálnym N opakovami.



Obr. 4.1: Priklad algoritmu v 2D rovine

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

## 4.3 RANSAC 3D

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

1. Príprava dát: Načítajte 3D dátá 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 inlierov: 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 dátá. Je dôležité si uvedomiť, že konkrétna implementácia sa môže lísiť v závislosti od použitých knižníc alebo programovacieho jazyka. [1] [2]

## 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 skalar, 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 vyhodnotte 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 snímajuce pohyb, vyrobené Microsoftom. Zariadnei obsahuje RGB kamery a infračervený projektor a detektor ktorý monitoruje hĺbku priestoru na zaklade štrukturovateľného svetla alebo na zaklade času trvajúcemu svetlu dopadnut' na objekt, vďaka ktorému vie kinect poskytnúť rekognizáciu giest v reálnom čase. Kinect sa použiva hlavne v hernom priemysle, ale používa sa taktiež na komerčné a akademické učely pretože poskytuje mapovanie priestoru a je lacnejší než profesionálne zariadenia.



Obr. 5.1: Kinect v2

### 5.1 Ako funguje

Infračervený projektor na kinevte posiela modulované infračervené svetlo ktoré je zachytené sensormy. Infračervené svetlo ktoré sa odrazí od bližších objektov ma kratší čas letu ako svetlo ktoré sa odrazí od vzdialenejších objektov, takže sensor sníma ako vymodulovaný vzor bol deformovaný z času letu svetla, pixel po pixeli. Čas príletu meranej hlbky touto metódov môže byť presnejšie vypočítany v kratšom čase, čo zabezpečí viac snímkov za sekundu. Hned' ako kinect naskenuje hlbkovú fotografiu, použije metódu zist'ovania 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 videi 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 potrebovali sme 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

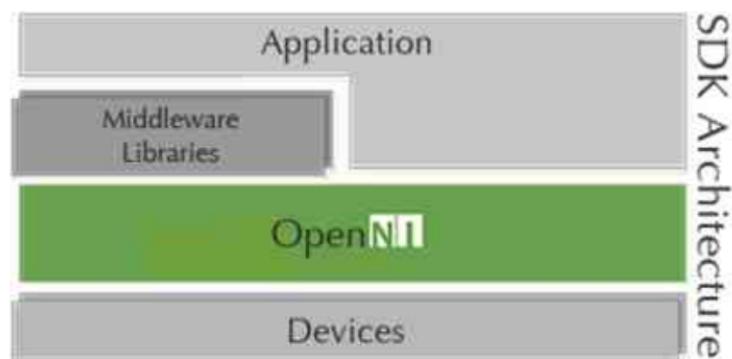
- libusb
- TurboJPEG
- OpenGL
- OpenCL (optional)
- CUDA (optional, NVIDIA only)
- VAAPI (optional)
- OpenNI2

## 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 používych 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 sprcovanie gest, detekcie a sledovanie pohybov, rozpoznávanie tvarov je potrebné rozšíriť middleware, alebo použiť Microsoft SDK, ale napäťko táto publikacia je spracovaná na operačnom systéme Linux je potrebné použiť OpenNI2 SDK, ktoré by malo pracovať aj s inými druhmi kamier.[6]

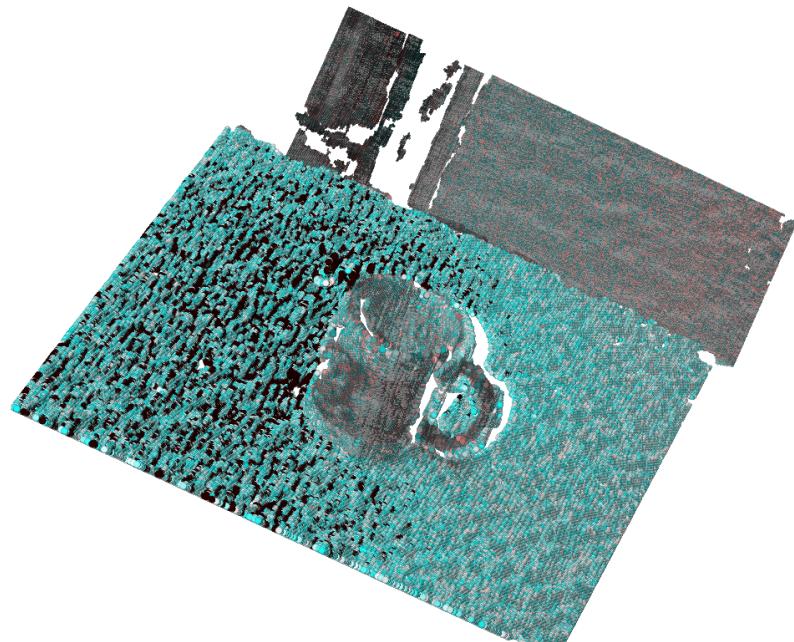
## 5.4 Point cloud (PCD) súbor

Pomocou OpenNI2 implementovaného v programovacom jazyku C++ a knižnicou PCL je nadviazaná komunikácia s Kinectom, ktorý streamuje dátá a sú vyobrazené v okne OpenNI2 a



Obr. 5.2: Architektúra softwarového vývoja

pomocou klávesy S vieme vyhotoviť PCD súbor potrebný na vyhľadavanie tvarou pomocou RANSAC algoritmu.



Obr. 5.3: Výstupný Point Cloud z Kinectu

# 6 Implementavanie RANSAC algoritmu

Kód: BakalarskaPraca/PCLTUTORIAL/cylinder.cpp

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 1. PointCloudData

Pre praktickejšiu prácu s pointcloudom na začiatku odfiltrujeme 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 2. Zvolenie geometrického modelu

Vyberieme si geometricky model, čo bude zodpovedať trom náhodným bodom aby sme vedeli zestrojiť rovinu. V blízkosti roviny vo vopred definovanom thresholde 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 väča 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'vke 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 3. Definovanie parametrov

V kóde sa na začiatku nastavili požadované hodnoty, čiže hľadany model v kóde nastavený na välec, vzdialenosť bodov pre vyhodnotenie RANSACA ako inlier 5cm, vplyv normál na váhu 0.1 a nastavili limit modulu aby bol menší ako 10cm. Hodnoty treba na začiatku preskušať a vybrať ktoré maju najväčšiu presnosť alebo rýchlosť, je možné skúsiť automatické

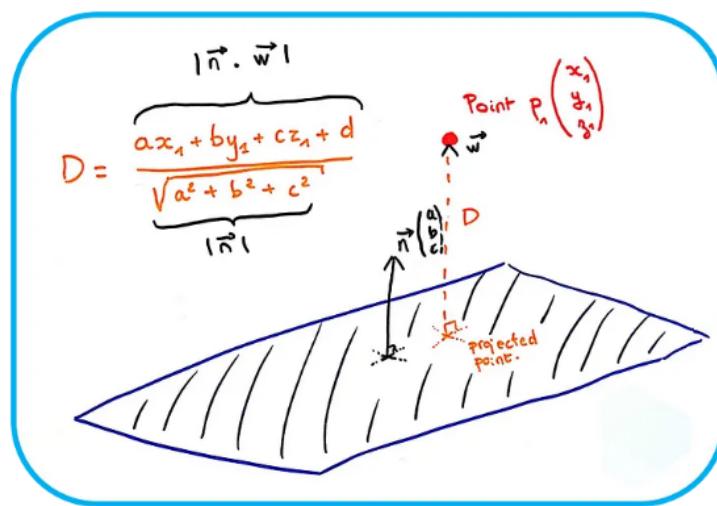
nastavovanie parametrov, čo by teoreticky mohlo fungovať na zaklade vypočítania strednej hodnoty vzdialenosť medzi susediacimi bodmi. [7]

## 6.4 4.implementovanie RANSACu

- Finding a plane (1 opakovanie)

Prv sa zvolia 3 body z PointCloudu, z ktorých sa má vytvoriť rovnica roviny najdením parametrov a,b,c,d. K ním sa dá dopracovať lineárnom algebrou, konkrétnie krížovým súčinom dvoch vektorov ktoré generuju ortogonálny vektor. Treba definovať vektory z rovnakého bodu v rovine a vypočítať normálu k nim, ktorá definuje normalu roviny. Pomocou normal normalizujeme naš normalový vektor a získame parametre a,b,c a dopočítame parameter d čo je posun roviny od originu.

- Bod k rovine: definovanie tresholdu



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

Obr. 6.1: Dajake rovnice co treba prepočítať a prekresliť

Z obrázka 6.1 vyplýva, ak sa vyberie hociktorý bod mimo troch použitých na vytvorenie roviny, získame vzdialosť bodu od roviny. Vzdialosť sa porovnáva s tresholdom a vyhodnotí sa, či je bod inliers.

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

## **6.5 5. Vyzualizácia PointCloudu**

Z predošleho kroku výstupom je set bodov ležiacich v thresholde, ktoré sú zapísané do súboru pre vyhodnotenie výsledku.

# 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 rôzného uhla pohľadu. Výstupnými dátami bol 7.1, ktorý bol prekonvertovaný do sedej pre rozšírenie údajov na a nájdený model s počtom inliers.

Model nájdený pomocou RANSAC:

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

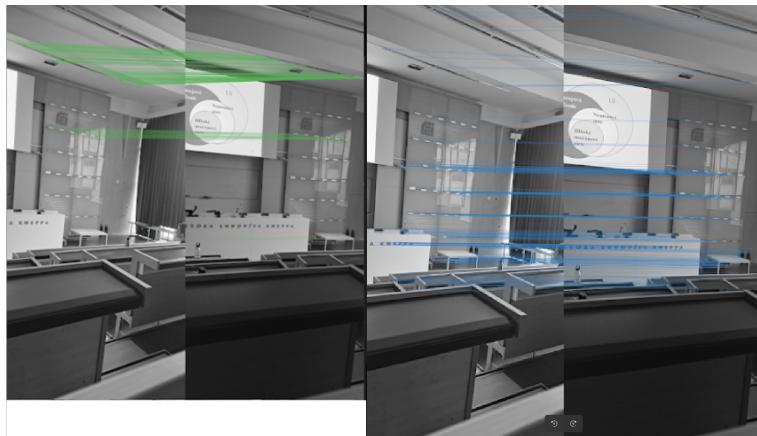
Počet inlierov: 251.

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]



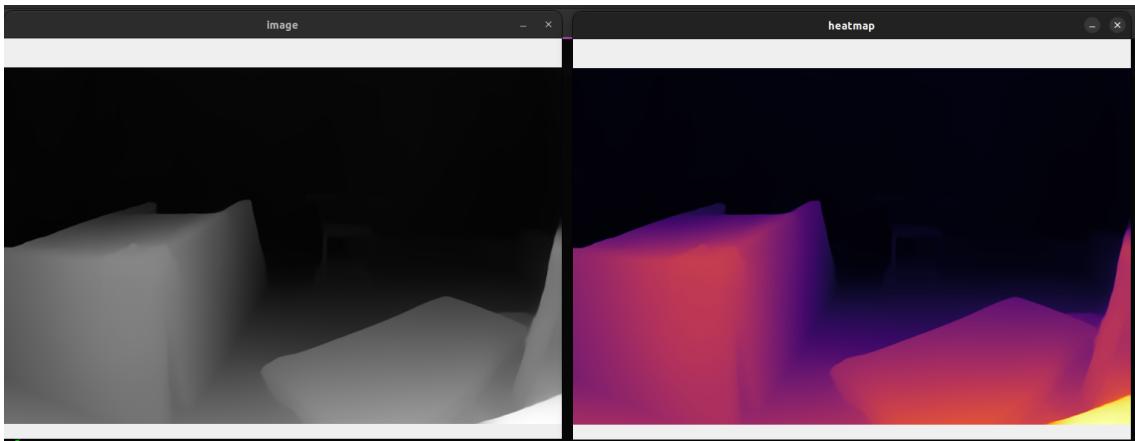
Obr. 7.1: Výstup Neural guided RANSAC

	Trénovací objekt	%Inliers	F-score	Priemerná	Stredná
RANSAC	-	21.85	13.84	0.35	0.32
USAC	-	21.43	13.90	0.35	0.32
Deep F-Mat	Priemerná	24.61	14.65	<b>0.32</b>	<b>0.29</b>
NG-RANSAC	Priemerná	25.05	<b>14.76</b>	<b>0.32</b>	<b>0.29</b>
NG-RANSAC	F-score	24.13	14.72	0.33	0.32
NG-RANSAC	%Inliers	<b>25.12</b>	14.74	<b>0.32</b>	<b>0.29</b>

Tabuľka 7.1: Odhad základných matíc

## 7.2 ngdsac cameralec

Táto práca vychádza z hotovej implementácie Neural Network (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 cieľa, ktorý poskytuje zábery na budovu na univerzite. Dataset však neposkytuje reálnu polohu kamery, aby bol dataset použiteľný pri porovnaní 3D dát potrebných pre túto prácu. Cieľom implementacie bolo vytvoriť dataset z Kinectu z hlbkovej kamery ktoré sme pre lepšiu čitateľnosť pretransformovali na heat mapu 7.2, s reálnou polohou kamery, ale nebolo možné s vytvoreným datasetom natrenovať NN, pretože NN používala Structure from Motion pipeline, ktorá vytvorila súbor typu .nvm s približne odhadnutou polohou kamery a ďalším neznámym dátam, ktoré nebolo možné zreprodukovať, čiže sa nepodarilo natrenovať sieť na 2D dátach. Dáta sa používali na nastavovanie Pythonovskej knižnice tensorflow. [8] [5]



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

	DSAC++(VGGNet)	DSAC++(ResNet)	NG-DSAC++(ResNet)
Great Court	40.3cm	40.3cm	<b>35.0cm</b>
Kings College	17.7cm	13cm	<b>12.6cm</b>
Old Hospital	<b>19.6cm</b>	22.4cm	21.9cm
Shop Facade	5.7cm	5.7cm	<b>5.6cm</b>
St M. Church	12.5cm	9.9cm	<b>9.8cm</b>

Tabuľka 7.2: Premiestnenie kamery v priestore

Práca porovnáva výsledky diferenciálneho RANSACU (DSAC) a jeho rozšírenie s použitím neurónovej sieti (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.

### 7.3 SFPN

Hlavnou téhou štúdie "Supervised Fitting of Geometric Primitives to 3D Point Clouds" je prispôsobovanie geometrických primitív 3D PCD. Táto problematika má zásadný význam na prepojenie digitalizovanými 3D údajmi na nízkej úrovni a štrukturálnymi znalosťami na vysokej úrovni o základných 3D tvaroch, čo umožňuje široké spektrum následných aplikácií pri spracovaní 3D údajov.

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 ľahko 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 povrhy 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.

Výkonnosť SPFN pri spracovaní vstupných mračien bodov s vysokým rozlíšením pri testovaní, hoci siet' bola vycvičená na mračnách bodov s nižším rozlíšením, autori ďalej skúmali pri následných testoch. Tieto testy ukázali, že SPFN funguje efektívne aj za týchto okolností. [9] [5]

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	<b>6.87</b>	5.85
<i>Eff.RANSAC + J* + <math>\hat{W}</math></i>	69.91	60.56	<b>6.87</b>	2.90
<i>Eff.RANSAC + J* + <math>\hat{W} + \hat{t}</math></i>	60.68	92.76	<b>6.87</b>	6.21
<i>Eff.RANSAC + <math>\hat{N} + \hat{W} + \hat{t}</math></i>	60.56	93.13	8.15	7.02
<i>DPPN(Sec.4.4)</i>	44.05	51.33	-	3.68
<i>SPFN – <math>L_{seg}</math></i>	41.61	92.40	8.25	1.70
<i>SPFN – <math>L_{norm} + J^*</math></i>	71.18	95.44	<b>6.87</b>	4.20
<i>SPFN – <math>L_{res}</math></i>	72.70	96.66	8.74	1.87
<i>SPFN – <math>L_{axis}</math></i>	<b>77.31</b>	96.47	8.28	6.27
<i>SPFN(<math>\hat{t} - &gt; Est.</math>)</i>	75.71	95.95	8.54	1.71
<i>SPFN</i>	77.14	<b>96.93</b>	8.66	<b>1.51</b>

Tabuľka 7.3: Premiestnenie kamery v priestore

popisem dacp popisem dacp popisem dacp popisem dacp popisem dacp popisem dacp  
popisem dacp popisem dacp popisem dacp popisem dacp popisem dacp popisem dacp  
popisem dacp popisem dacp popisem dacp popisem dacp popisem dacp popisem dacp  
popisem dacp popisem dacp popisem dacp popisem dacp popisem dacp popisem dacp

popisem dacp

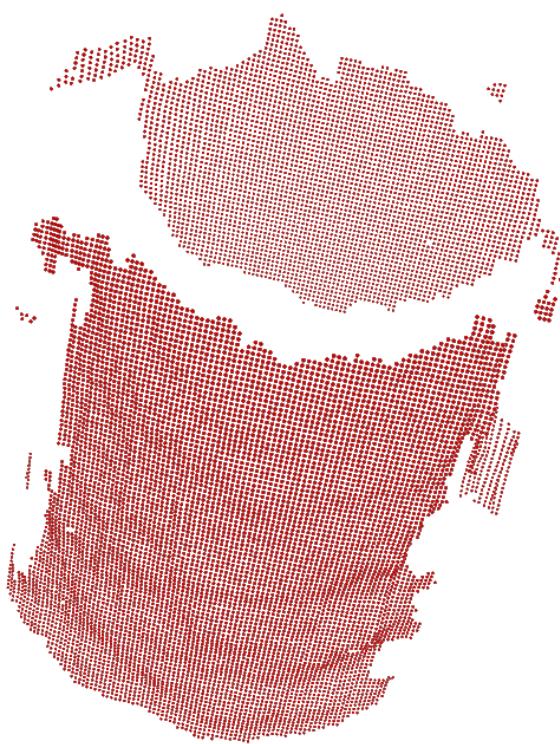
Metóda	$\{S_k\}$ Residual	$\{S_k\}$ Coverage		P Coverage	
	Mean $\pm$ Std.	$\epsilon = 0.01$	$\epsilon = 0.02$	$\epsilon = 0.01$	$\epsilon = 0.02$
<i>Eff.RANSAC + J</i>	0.072 $\pm$ 0.361	43.42	63.16	65.74	88.63
<i>Eff.RANSAC* + J*</i>	0.067 $\pm$ 0.352	56.95	72.74	68.58	92.41
<i>Eff.RANSAC + J*</i>	0.080 $\pm$ 0.390	51.59	67.12	72.11	92.54
<i>Eff.RANSAC + J* + <math>\hat{W}</math></i>	0.029 $\pm$ 0.234	74.32	83.27	78.79	94.58
<i>Eff.RANSAC + J* + <math>\hat{W} + \hat{t}</math></i>	0.036 $\pm$ 0.251	65.31	73.69	77.01	92.57
<i>Eff.RANSAC + <math>\hat{N} + \hat{W} + \hat{t}</math></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 - <math>L_{seg}</math></i>	0.029 $\pm$ 0.178	50.04	62.74	62.23	77.74
<i>SPFN - <math>L_{norm} + J^*</math></i>	0.022 $\pm$ 0.188	76.47	81.49	83.21	91.73
<i>SPFN - <math>L_{res}</math></i>	0.017 $\pm$ 0.162	79.81	85.57	81.32	91.52
<i>SPFN - <math>L_{axis}</math></i>	0.019 $\pm$ 0.188	80.80	86.11	86.46	94.43
<i>SPFN(<math>\hat{t} - &gt; Est.</math>)</i>	0.013 $\pm$ 0.140	85.25	90.13	86.67	94.91
<b>SPFN</b>	<b>0.011<math>\pm</math>0.131</b>	<b>86.63</b>	<b>91.64</b>	<b>88.31</b>	<b>96.30</b>

Tabuľka 7.4: Premiestnenie kamery v priestore

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 každy segment v predpovedanej matici príslušnosti  $\hat{W}$  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 normal  $\hat{N}$  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 normálky a primitívne typy neurónovou siet'ou nezlepšili  $S_k$  a P pokrytie v RANSACU.

## 8 Vystupy

S použitím pointCloudu z obrázka 5.2 výstupom algoritmu so vstupným parametrom na ziskávanie valca je vonkajšia a vnútorna rovina hrnčeka. Point cloud bol stiahnutý z internetu. Z ?? môžno vidieť chybu strednej hodnoty odhadu pozície použitia algoritmov. Z výsledkov

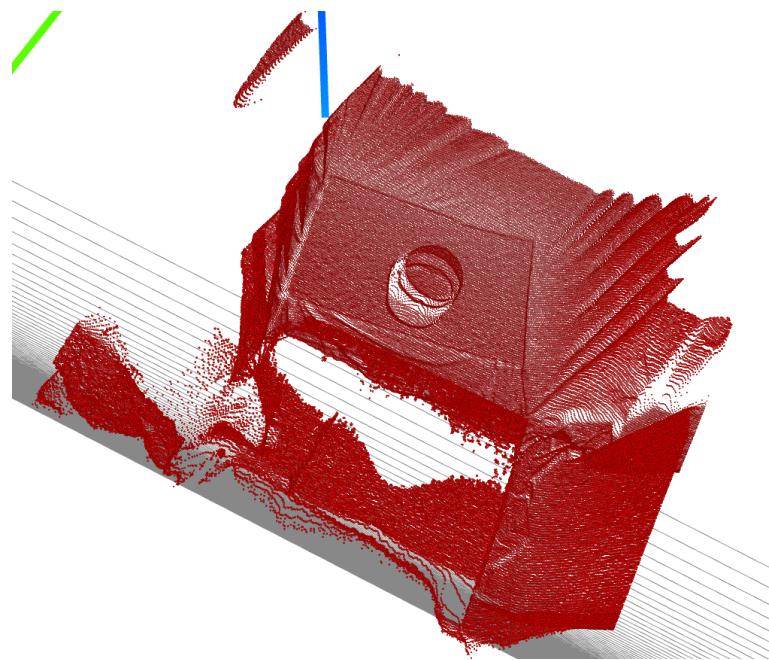


Obr. 8.1: Výstup algoritmu RANSAC v knižnici PCL

výplýva, že diferenciálny RANSAC rozšírený o neurónovú sieť na väčšine datasetu určoval presnosť polohy kamery.

### 8.1 POZNAMKA

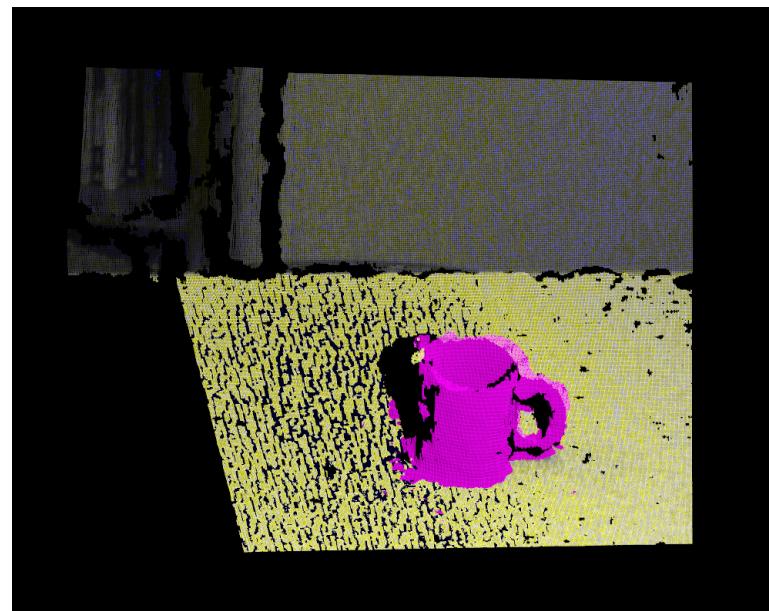
Kinect ma horsie vzorkovanie oproti pouzitemu PCD cize aktuálne pracujem na zdokonalovaní spracovania výstupu Kinectu, pretože na zložitejsie tvary horsie spracuvava nedá sa rozoznať tvar objektu a často krát zpracuje aj okolité predmety do vypočtu co vyhodnocie zle výsledky. Musíme upraviť scenu pretože skor ako nadobu v strede rozozna vlnovitu stenu ako valec.



Obr. 8.2: Aktualny vystup z kinectu

## 8.2 Allignment

vyobrazenie pouziteho pointcloudu a tvaru najdenetho v nom



Obr. 8.3: Spojene

$$R = \begin{vmatrix} 1.000 & -0.024 & 0.00 \\ 0.024 & 1.000 & 0.014 \\ -0.002 & -0.014 & 1.000 \end{vmatrix} T = \begin{vmatrix} -0.001 & -0.012 & 0.001 \end{vmatrix}$$

```

from freenect2 import Device, FrameType
import numpy as np

# Open the default device and capture a color and depth frame.
device = Device()
frames = {}
with device.running():
    for type_, frame in device:
        frames[type_] = frame
    if FrameType.Color in frames and FrameType.Depth in frames:
        break

# Use the factory calibration to undistort the depth frame and register the RGB
# # frame onto it.
rgb, depth = frames[FrameType.Color], frames[FrameType.Depth]
undistorted, registered, big_depth = device.registration.apply(
    rgb, depth, with_big_depth=True)

# Combine the depth and RGB data together into a single point cloud.
with open('output.pcd', 'wb') as fobj:
    device.registration.write_pcd(fobj, undistorted, registered)

with open('output_big.pcd', 'wb') as fobj:
    device.registration.write_big_pcd(fobj, big_depth, rgb)

```

Výpis 1: Vytvorenie PCD pomocou Kinectu

# **Záver**

Conclusion is going to be where?

Here.

# Zoznam použitej literatúry

1. *Point cloud library.* Dostupné tiež z: <https://pointclouds.org/>.
2. 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&hl=en&user=U\\_NEZHQAQAAJ&citation\\_for\\_view=U\\_NEZHQAQAAJ:u-x6o8ySG0sc](https://scholar.google.com/citations?view_op=view_citation&hl=en&user=U_NEZHQAQAAJ&citation_for_view=U_NEZHQAQAAJ:u-x6o8ySG0sc).
3. – SA ZMENI, wikipedia (zost.). *Point cloud.* [https://en.wikipedia.org/wiki/Point\\_cloud](https://en.wikipedia.org/wiki/Point_cloud).
4. 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>.
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 GUIBAS, Leonidas. *Supervised Fitting of Geometric Primitives to 3D Point Clouds.* 2018. Dostupné z eprint: arXiv:1811.08988.