

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-100863-111119

**POROVNANIE SPRACOVANIA 3D DÁT ANALYTICKÝM
METÓDAMI A NEURÓNOVÝMI 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 NEURÓNOVÝMI 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:	Maroš Kocúr
Študijný program:	robotika a kybernetika
Študijný odbor:	kybernetika
Evidenčné číslo:	FEI-100863-111119
ID študenta:	111119
Vedúci práce:	Ing. Miroslav Kohút
Vedúci pracoviska:	prof. Ing. Jarmila Pavlovičová, PhD.
Miesto vypracovania:	Ústav robotiky a kybernetiky

Názov práce: **Porovnanie spracovania 3D dát analytickými metódami a neurónovými sieťami**

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. Popíšte a analyzujte základnú problematiku práce s 3D dátami a neurónovými sieťami.
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 vyhodnoťte výsledky.

Termín 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 sieťami
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 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.

Kľúčové slová: RANSAC, PCL, neurónová sieť

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

Úvod	1
1 PCL	2
1.1 Použitie	2
1.2 Befor installation	2
1.3 Inštalácia PCL	2
1.4 Používanie PCL	3
1.5 Kompilácia projektu	3
2 Point Cloud	4
2.1 General	4
2.2 Použitie	4
2.3 Konverzia do 3D povrchu	4
3 Neural Network	5
3.1 History	5
3.2 What is it	5
3.3 Vývoj	5
4 RANSAC	6
4.1 Overview	6
4.2 Algoritmus	6
4.3 RANSAC 3D	7
4.4 Neural guided RANSAC	7
5 KINECT	8
5.1 Ako funguje	8
5.2 Výstup z KINECTU	8
5.2.1 List potrebných Knižnic pre Ubuntu 22.04 LTS	8
5.3 OpenNI 2.0	9
5.4 PCD súbor	9
6 Implementovanie RANSAC algoritmu	10
6.1 1. PointCloudData	10
6.2 2. Zvolenie geometrického modelu	11
6.3 3. Definovanie parametrov	11

6.4	4.implementovanie RANSACu	11
6.5	5. Vyzualizácia PointCloudu	13
7	Implementovanie RANSAC algoritmu s použitím NN	14
8	Vystupy	15
8.1	POZNAMKA	15
9	Recitácia	16
10	Kod = neaktualne	17
	Záver	18
	Zoznam použitej literatúry	19

Zoznam obrázkov a tabuliek

Obrázok 3.1	Priklad neuronovej siete	6
Obrázok 4.1	Priklad algoritmu v 2D rovine	7
Obrázok 5.1	Architektúra softwarového vývoja	9
Obrázok 5.2	Výstupný Point Cloud z Kinectu	10
Obrázok 6.1	Dajake rovnice co treba prepocitat a prekreslit	12
Obrázok 8.1	Výstup algoritmu RANSAC v knižnici PCL	15
Obrázok 8.2	Aktualny vystup z kinectu	16

Zoznam algoritmov

Zoznam výpisov

1	Vytvorenie PCD pomocou Kinectu	17
---	--	----

Úvod

V tejto práci sa zameriame na vyhľadavanie rôznych tvarov v Point Cloude, ktorý si sami naskenujeme pomocou Kinect v2. Porovname algoritmus RANSAC do ktorého implementujeme neuronové siete a porovname efektívnosť a presnosť tohto algoritmu. Taktiež sa naučíme pracovať s Point Cloud Library a hardwarom Kinect v2 na operačnom systéme Ubuntu 22.04 LTS.

1 PCL

PCL je samostný, vysoko škaloateľný, otvorený projekt pre 2D a 3D obrázky a spracovávanie point cloud. Knižnica je cross platform, napisana v jazyku C++ a Pythone. 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 použiteľná na úroveň komerčné účely a za účelom výskumu.[1]

1.1 Použitie

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, odhady funkcie, rekonštrukcie povrchov, 3D registrácie, prispôsobenie modelu, vyhľadávanie objektov a segmentácie. PCL má vlastný formát na ukladanie point cloud dát - PCD, ale podporuje načítanie a ukladanie dát do rôznych iných formátov.[2] Algoritmy sa používajú na perception v robotike na filtrovanie zašumených dát, spájanie 3D dát, segmentovanie dôležitých častí v priestore, extrahovanie kľúčových bodov a výpočet deskriptorov na rozpoznávanie objektov vo svete na základe ich geometrického vzhľadu a vytváranie povrchov z point cloudu a vykresľovanie ich.

1.2 Befor installation

PCL potrebuje niekoľko knižníc 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 pointerov 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 distribúciách Ubuntu a Debian môžeme nainštalovať pomocou.

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

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

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

MacOS má Homebrew package manažéra ktorý podporuje inštaláciu 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 nastaví 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 vojdeme 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.

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 veľ'a bodov na externom povrchu objektov okolo. My v tomto projekte použijeme Kinect 2 (odsek 6). Point cloud sa používa v 3D modelovaní, metroológii, 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 Použitie

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

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 cloudu, kde sa môže požiť vzdialenostná a objemová aproximácia.

3 Neural Network

3.1 History

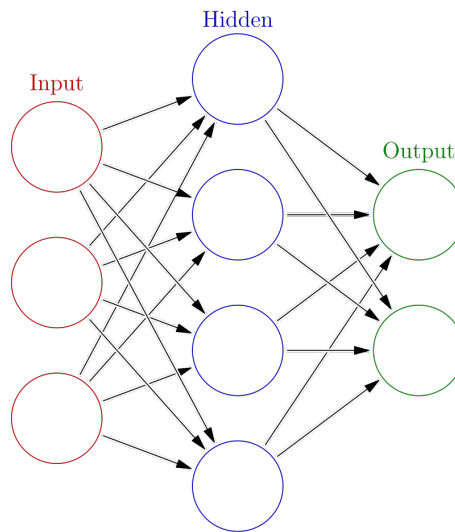
V posledných rokoch, najlepšie systémy s aplikáciou umelej inteligencie - ako sú rozpoznávače reči v mobilných zariadeniach alebo automatické prekladače majú dobré výsledky v technike nazývajúcej sa "deep learning". Deep learning je v podstate iné meno pre aplikáciu umelej inteligencie s názvom Neural network, ktorý sa vyvíja takmer 80 rokov. NN boli prvýkrát navrhnuté v roku 1944, dvoma výskumníkmi z Chicagskej univerzity, ktorí v roku 1952 prešli na MIT a založili oddelenie kognitívnych vied.

3.2 What is it

NN sú myslené na robenie strojového učenia, v ktorom sa počítač učí vykonávať úlohy na základe analyzovania trenovacích 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 sieť hľadá vizuálne patery v obrázku ktoré majú vzajomný vzťah s konkrétnym označením. Voľne modelovaná NN na ľudskom mozgu má tisíce až milióny jednoduchých node, ktoré sú husto prepojené. Väčšina siete sa dnes organizuje do vrstiev node a tie posielajú údaje vpred, čo znamená že dáta nimi prechádzajú iba v jednom smere. Jedna noda môže byť pripojená k viacerým nodam vo vrstvách pod ňou z ktorej prijíma dáta a viacej node vo vrstve nad ňou kde spracované dáta posielajú. Ku každému vstupnému prepoju sa pridá číslo známe ako "váha". Ak je sieť aktívna, noda prijíma rôzne dáta z každého vstupu a vynásobí ich pridelenou váhou, potom všetky výsledky sčíta ak je výsledné číslo pod threshold hranicou, noda nepošle žiadne dáta do ďalšej vrstvy. Ak číslo prekročí threshold hodnotu, tak noda "výstrelí", čo znamená že pošle súčet vstupov na všetky výstupy. Keď sa NN trénuje, všetky váhy a thresholdy sú nastavené na náhodnú hodnotu. Trénovacie dáta sú posielané do spodnej vstupnej vrstvy a prechádzajú cez nasledujúce vrstvy, násobia sa a sčítavajú sa v zložitými cestami, pokiaľ radikálne transformované nedojdu na vrchnú výstupnú vrstvu. Počas tréningu sa váhy a prahové hodnoty neustále upravujú kým tréningové údaje s podobnými štítkami neprinášajú podobne výstupy.

3.3 Vývoj

Neuronové siete opísané v roku 1944 mali váhy a threshold hodnoty, ale neboli usporiadané do vrstiev a výskumníci nešpecifikovali trénovací mechanizmus. Výskumníci dokázali ukázať, že NN dokáže v princípe vypočítať hocikakú funkciu ako digitálny počítač. Výsledkom bola viac neurová veda ako počítačová a cieľom bolo poukázať, že ľudský mozog môžeme považovať za výpočtové zariadenie.[4]



Obr. 3.1: Příklad neuronovej siete

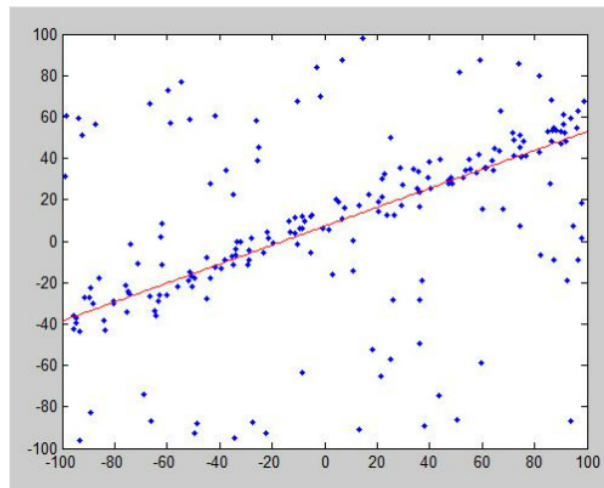
4 RANSAC

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-estimators a meródov najmenších štvorcov s prepojením na strojové učenie. RANSAC bol vytváraný komunitou ľudí používajúci strojové učenie. RANSAC je vzorkovacia technika ktorá generuje kandidáta na minimalny počet pozorovani potrebných na zistenie odhadu parametrov ležiacich pod modelom. Na rozdiel 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 koľko 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 inliers a ukončí.
5. Inak, opakuj kroky 1 až 4, s maximálnym N opakovaniami.



Obr. 4.1: Příklad algoritmu v 2D rovine

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

4.3 RANSAC 3D

Podobne ako pri opise vyššie RANSAC 3D funguje s výberom náhodných troch bodov na ktorých zostaví rovinu a spočíta body ležiace v rovine a body ležiace mimo roviny. Algoritmus sa opakuje n-opakovaní a výstupom je najlepší model.

4.4 Neural guided RANSAC

5 KINECT

Kinect je vstupné zariadenie snímajúce pohyb, vyrobené Microsoftom. Zariadenie obsahuje RGB kamery a infračervený projektor a detektor ktorý monitoruje hĺbku priestoru na základe štrukturovateľného svetla alebo na základe času trvajúceho svetlu dopadnúť na objekt, vďaka ktorému vie kinect poskytnúť rekognizáciu miest 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.

5.1 Ako funguje

Infračervený projektor na kinecte posiela modulované infračervené svetlo ktoré je zachytené sensormi. Infračervené svetlo ktoré sa odrazí od bližších objektov má 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 priletu meranej hĺbky touto metódou môže byť presnejšie vypočítaný v kratšom čase, čo zabezpečí viac snímkov za sekundu. Hneď ako kinect naskenuje hĺbkovú fotografiu, použije metódu zisťovania hrán k 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 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.

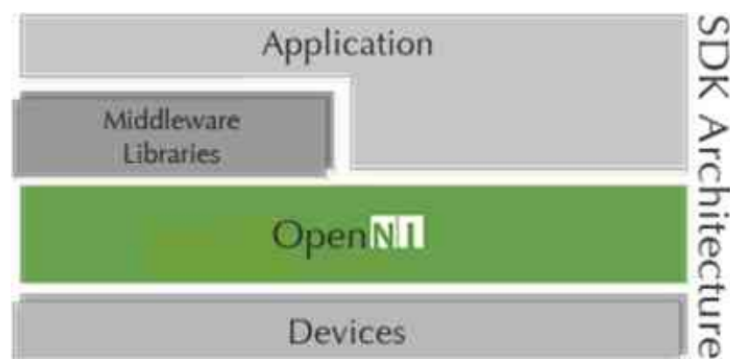
5.2.1 List potrebných Knižníc pre Ubuntu 22.04 LTS

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

- OpenNI2

5.3 OpenNI 2.0

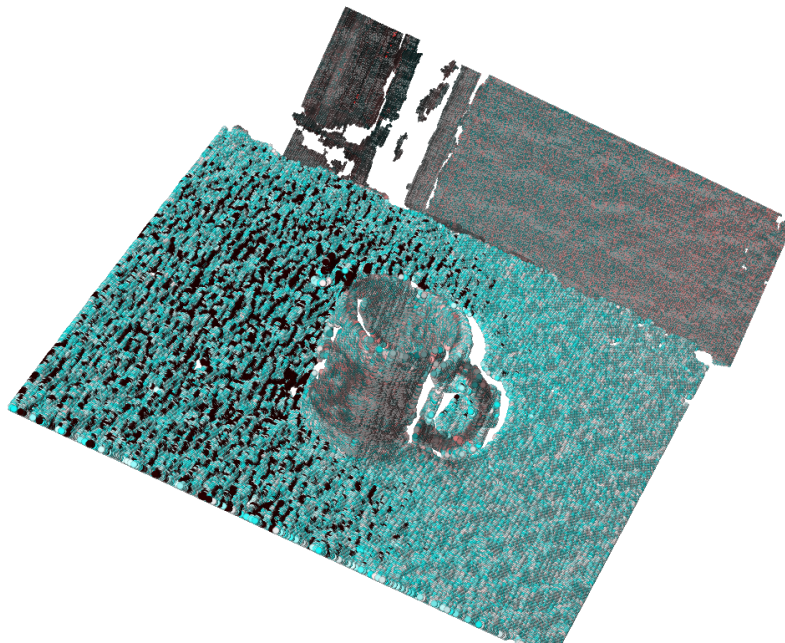
Open-source framework vytvorený spoločnosťou PrimeSense pre 3D Natural Interaction senzory od spoločnosti napríklad Asus Xtion, spoločnosť stojí za licencovaným dizajnom hardwaru a čipov použitých priamo v Kinecte. Druhá verzia OpenNI sa oproti prvej ma znížiť zložitosť API tým že zložité funkcie sa 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 nakoľko táto publikácia je spracovaná na operačnom systéme Linux je potrebné použiť OpenNI2 SDK, ktoré by malo pracovať aj s inými druhmi kamier.[5]



Obr. 5.1: Architektúra softwarového vývoja

5.4 PCD súbor

Pomocou OpenNI2 implementovaného v programovacom jazyku C++ a knižnicou PCL je nadviazaná komunikácia s Kinectom, ktorý streamuje dáta a sú vyobrazené v okne OpenNI2 a pomocou klávesy S vieme vyhotoviť PCD súbor potrebný na vyhľadávanie tvarou pomocou RANSAC algoritmu.



Obr. 5.2: Výstupný Point Cloud z Kinectu

6 Implementovanie RANSAC algoritmu

OTAZKA NA KONZULTANTA TREBA POPISOVAT MATEMATICKY KAZDY JEDEN MODEL ?

Kód: `BakalarskaPraca/PCLTUTORIAL/cylinder.cpp`

Budeme postupovat pomocou krokov:

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

[6]

6.1 1. PointCloudData

Pre praktickejšiu prácu s pointcloudom na začiatku odfiltrujeme všetky body ktoré su vzdialenejšie ako 1,5 metra a odhadneme normaly povrchu v každom bode. Odfiltrovaný model

je uložený v samostatnom PCD súbore.

6.2 2. Zvolenie geometrického modelu

Vyberieme si geometricky model, čo bude zodpovedať trom náhodným bodom aby sme vedeli zostrojiť rovinu. V blízkosti roviny vo vopred definovanom thresholde spočítame koľko bodov leží v blízkosti roviny. Vyberieme rovinu ktorá ma najväčši počet inlierov ako najlepší model. Postup sa opakuje pokiaľ neprejde algoritmus presný počet interácii zvolených na začiatku. V kóde je zvolený model válcu ktorý ma v 3D priestore predpis

$$f(x - a/c * z, y - b/c * z) = 0 \quad (6.1)$$

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

6.3 3. Definovanie parametrov

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

6.4 4.implementovanie RANSACu

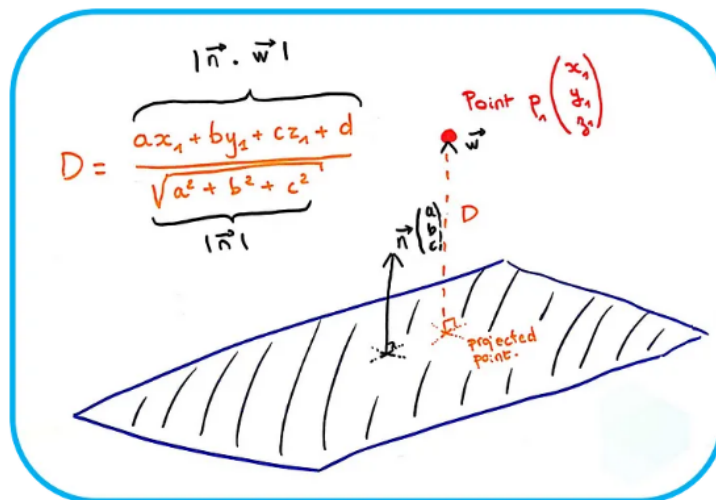
- Finding a plane (1 opakovanie)

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

- Bod k rovine: definovanie thresholdu

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

- Opakovanie Vytvorenie cyklu ktorý sa bude opakovať počet interácii a porovnávať, či



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 prepocitat a prekreslit

aktuálny model je lepší než doteraz najlepší model a výstupom bude najlepší fitting RAN-SAC model.

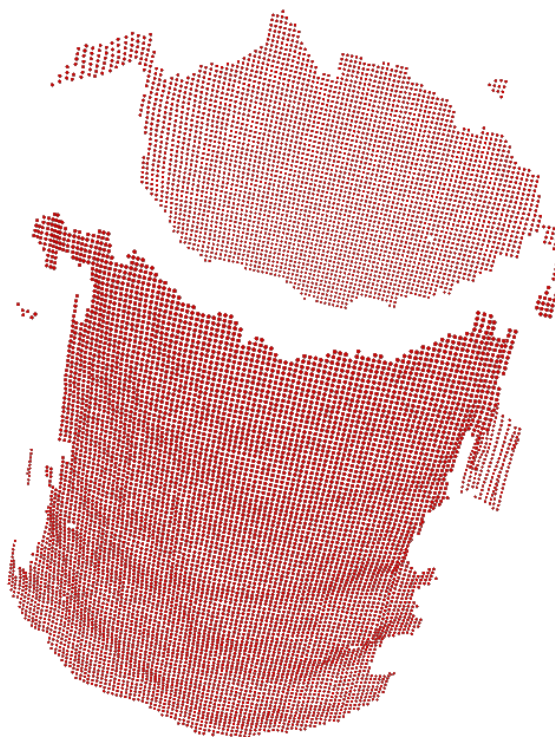
6.5 5. Vyzualizácia PointCloudu

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

7 Implementovanie RANSAC algoritmu s použitím NN

8 Vystupy

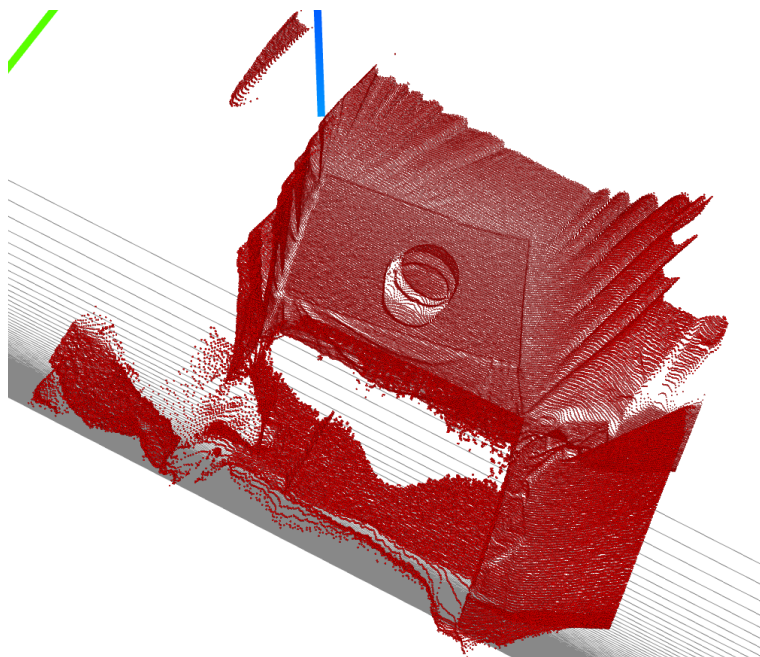
S použitím pointCloudu z obrázka 5.2 výstupom algoritmu so vstupným paramtrom na získavanie valca je vonkajšia a vnútorná rovina hrnčeka. Point cloud bol stiahnutý z internetu.



Obr. 8.1: Výstup algoritmu RANSAC v knižnici PCL

8.1 POZNAMKA

Kinect má horšie vzorkovanie oproti použitému PCD čiže aktuálne pracujem na zdokonalovaní spracovania výstupu Kinectu, pretože na zložitejšie tvary horšie spracováva neda sa rozoznať tvar objektu a často krát zapracuje aj okolité predmety do výpočtu čo vyhodnotí zlé výsledky. Musím upraviť scenu pretože skor ako nadobu v strede rozozná vlnovitou stenu ako valec.



Obr. 8.2: Aktualny vystup z kinect

9 Recitácia

Citujem všetky zdroje v **bibliography.bib**, [4, 2, 1, 5, 3, 6].
Good luck.

10 Kod = neaktualne

```
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_NEZHQA AAAAJ&citation_for_view=U_NEZHQA AAAAJ:u-x6o8ySG0sC.
3. – SA ZMENI, wikipedia (zost.). *Point cloud*. <https://en.wikipedia.org>. Dostupné tiež z: 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. 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>.
6. 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>.