SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-xxxx-xxxx

POROVNANIE SPRACOVANIA 3D DÁT ANALYTICKÝM METÓDAMI A NEURÓNOVÝMI SIEŤAMI

BAKALÁRSKA PRÁCA

2022 **Maroš Kocúr**

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Evidenčné číslo: FEI-xxxx-xxxx

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 2022 **Maroš Kocúr**

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 2022

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 2022

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

Zoznam obrázkov a tabuliek

Zoznam skratiek

NN	Neural Network
PCL	Point Cloud Library
RANSAC	Random sample consensus

Zoznam algoritmov

Zoznam výpisov

Úvod

V tejto prácy sa zameriame na vyhľadavanie rôzných tvarou v Point Cloude, ktorý si sami naskenujeme pomocou Kinect v2. Porovname algoritmus RANSAC do ktorého implementujeme neuronove siete a porovname efektivitu 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 škalovateľný, 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 tretímy stranami. My v tomto projekte budeme používať Ubuntu 22.04. LTS a verzia PCL je 1.12.1. Knižnica ja vydana pod BSD licenciami čo znamená, že je voľne použiteľna úre komerčné účely a za účelom výskumu.

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, odhady funkzie, rekonštrukcie povrchov, 3D registracie, prispôsobenie modelu, vyhľadavanie objektov a segmentácie. PCL ma vlastný format na ukladanie point cloud dát - PCD, ale podporuje načítanie a ukladanie dát do rôznych iných formátov. Algoritmy sa používajú na perception v robotike na filtrovanie zašumených dát, spájanie 3D dát, segmentovanie dôležitých Častí v priestore, extrahovanie kľučových bodovv a výpoČet deskriptorov na rozpoznávanie objektov vo svete na základe ich geometrického vzhľadu a vytvárať povrchy z point cloudu a vykreľovať ich.

1.2 Befor installation

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

1.3 Inštalácia PCL

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

sudo apt install libpcl-dev

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

PS> .lomka 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 stači nám použiť príkaz v C++, include <pcl/“názov$_k nižnice_1.h >$.

1.5 Kompilácia projektu

Vytvoríme si súbor CMakeList.txt v ktorom zadefinujeme potrebné premenné aby make vedel najsť 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 vytvorime 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 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.

2.2 Pouzitie

Pre priemyselnú metrológiu 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 využívaju na nazbieranie RGB obrázkov ktoré sa neskôr pomocov algoritmu strojového videnia ako je AgiSoft Photoscan, PixčD alebo DroneDeploy používaju na vytvorenie RGB point cloudu, kde sa môže požiť vzdialenostná a objemová aproximácia.

3 Neural Network

3.1 History

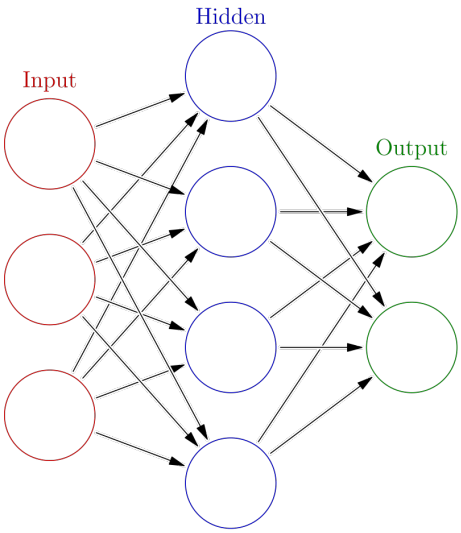
V posledných rokoch, najlepšie systémy s aplikaciou umelej inteligencie - ako sú rozpoznávače reči v mobilných zariadeniach alebo automatické prekladače majú dobré výsledky v technike nazývajúcej sa "deep learning". Deep learning je v podstate iné meno pre aplikaciu umelej inteligencie s názvom Neural 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 mylené na robenie strojového učenia, v ktorom sa počítač učí vykonávať úlohy na zákklade analyzovania trenovácich príkladov, ktore sú zvyčajne ručne oznaČene. Systém na rozpoznávanie objektov by mohol mať prístup k tisíckam obrázkov označených ako auto, dom, guľa a podobne. NN sieť hľada vizuálne paterny v obrázku ktoré koré majú vzajomný vzťah s konkretným 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 dáta nimi prechádzaju iba v jednom smere. Jedna noda môže byť pripijená k viacerím nodam vo vrstvách pod ňou z ktorej príjma data a viacej node vo vrstve nad ňou kde spracované dáta posiela. Ku každemu vstupnému prepoju sa pridelí čislo známe ako "váha". Ak je sieť aktívna, noda príjme rôzne dáta z každého vstupu a vynásoby ich pridelenou váhou, potom všetky výsledky sčita ak je výsledne číslo pod treshold hranicou, noda nepošle žiadne dáta do ďalšej vrstvy. Ak čislo prekročuje treshold hodnotu, tak noda "výstrelí", čo znamená že pošle súčet vstupov na všetky výstupy. Keď sa NN trenuje, všetky váhy a tresholdy sú nastavené na náhoddnu hodnotu. Trénovacie data sú posielané do spodnej vstupnej vrstvy a prechádzaju cez nasledujúce vrstvy, nasobia sa a sčitavajú sa v zložitými cestami, pokiaľ radikálne transformované nedojdu na vrchnú vystupnu vrstvu. Počas tréningu sa váhy a prahove hodnoty neustále upravuju kým treningové údaje s podobnými štítkami neprinášajú podobne výstupy.

3.3 Vývoj

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



Obr. 3.1: Priklad neuronovej siete

4 RANSAC

4.1 Overview

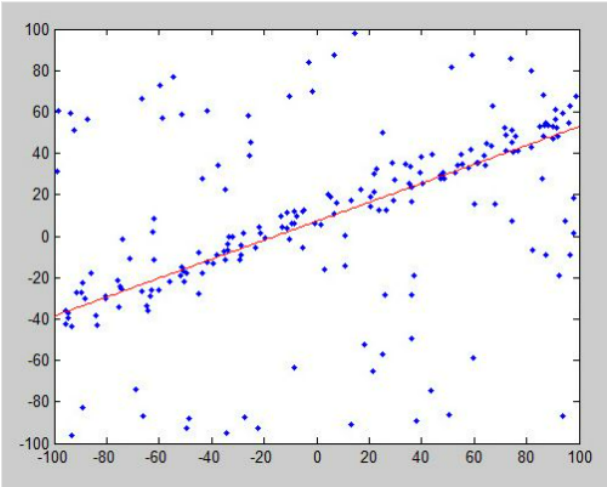
RANSAC je algoritmus vytvorený Fischlerom a Bollesom, určuje všeobecný prístup k odhadu parametrov, s veľkym podielom outliers v stupnom datasete. Na rozdiel od iných výkonných algoritmov odhadu, ako napríklad M-estimators a meródov najmenších štvorcov s prepojením na strojové učenie. RANSAC bol vytváraný komunitov ľudi používajúci strojové učenie. RANSAC je vzorkovacia technika ktorá generuje kandidáta na minimalny pocet pozorovani potrebnych na zistenie odhadu parametrov leziacich pod modelom. Narozdiel od ostatných vzorkovacích algoritmov, ktoré používaju čo najviac bodov ako môžu, RANSAC používa najmenši 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 $\epsilon. Akzlomokbodovležiacichv$

4. Inak, opakuj kroky 1 až 4, s maximálnym N opakovaniami.

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



Obr. 4.1: Priklad algoritmu v 2D rovine

4.3 RANSAC 3D

Podobne ako pri opise vyššie RANSAC 3D funguje s výberom náhodnych troch bodov na ktorých zostaví rovinu a spočita 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 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 dopadnuť 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 učely pretože poskytuje mapovanie priestoru a je lacnejši než profesionálne zariadenia.

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. Hneď ako kinect naskenuje hlbkovú 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 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 požadovane 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 Python script

Z githubu si cloneme freenect2-python, ktorý obsahuje scripty pomocov ktorých vieme získať snímky z Kinectu. Script dump-pcd.py nám vytvorí 2 point cloudy. Po menšej úprave v kóde výstpom je jeden Point cloud a snímka priestoru.

Obr. 5.1: Výstupný Point Cloud z Kinectu

6 Implementavanie RANSAC algoritmu

7 Vystupy

8 Implementovanie RANSAC alfgoritmu s pouzitim NN

9 Kod

```
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.

Prílohy

A Štruktúra elektronického nosiča

B Algoritmus

C Výpis sublime