SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

UKÁŽKOVÝ LaTeX DOKUMENT S DLHÝM NÁZVOM

SEMINÁRNA PRÁCA

2022 Ján Srnka

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

UKÁŽKOVÝ LaTeX DOKUMENT S DLHÝM NÁZVOM

SEMINÁRNA PRÁCA

Študijný program:	Aplikovaná informatika
Predmet:	I–ASOS – Architektúra softvérových systémov
Prednášajúci:	RNDr. John Doe, CSc.
Cvičiaci:	Ing. Peter Párker

Bratislava 2022 **Ján Srnka**

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. Random sample consensus (RANSAC) je dlhá skratka naopak NN je skratka v krátkej forme.PCL je skratka v krátkej forme.

1.1 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> .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.

1.2 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$_k nižnice_1.h$ > .

1.3 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.

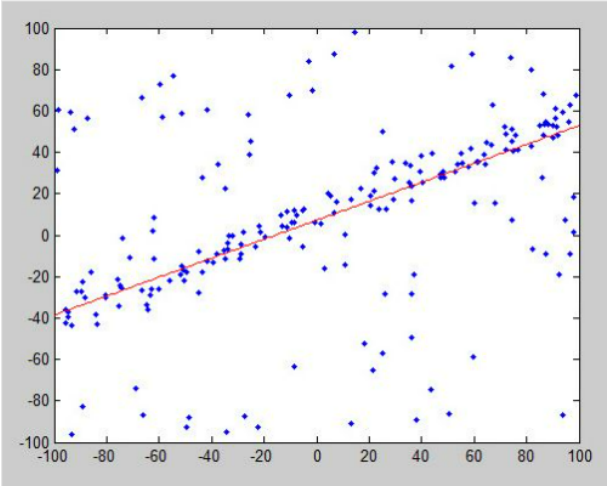
3 RANSAC

3.1 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.

3.2 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 .
4. Ak zlomok bodov ležiacich v preddefinovanej , presahuje preddefinovaný prah , prehodnotí parametre modelu použitím všetkých identifikovaných inliers a ukonči.
5. Inak, opakuj kroky 1 až 4, s maximálnym N opakovaniami.



Obr. 3.1: Priklad 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.

3.3 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.

3.4 Neural guided RANSAC

4 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.

4.1 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.

4.2 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.

4.2.1 List potrebných Knižnic pre Ubuntu 22.04 LTS

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

- OpenNI2

4.3 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. 4.1: Výstupný Point Cloud z Kinectu

5 Implementavanie RANSAC algoritmu

6 Vystupy

7 Implementovanie RANSAC alfgoritmu s pouzitim NN

8 Zaver

```
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

/CHANGELOG.md

· file describing changes made to FEIstyle

/example.tex

· main example *.tex* file for diploma thesis

/example_paper.tex

· example *.tex* file for seminar paper

/Makefile

· simply Makefile – build system

/fei.sublime-project

· is project file with build in Build System for Sublime Text 3

/img

· folder with images

/includes

· files with content

/bibliography.bib

· bibliography file

/attachmentA.tex

· this very file

B Algoritmus

Algoritmus B.1 Vypočítaj $y = x^n$

Require: $n \geq 0 \vee x \neq 0$

Ensure: $y = x^n$

```
y ⇐ 1
if n < 0 then
    X ⇐ 1/x
    N ⇐ −n
else
    X ⇐ x
    N ⇐ n
end if
while N ≠ 0 do
    if N is even then
        X ⇐ X × X
        N ⇐ N/2
    else {N is odd}
        y ⇐ y × X
        N ⇐ N − 1
    end if
end while
```

C Výpis sublime

../../ fei .sublime−project

Výpis C.1: Ukážka sublime-project