ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ

V PRAZE

FAKULTA STROJNÍ

BAKALÁŘSKÁ PRÁCE

2021

MARTIN
HASAL

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hasal** Jméno: **Martin** Osobní číslo: **483974**

Fakulta/ústav: **Fakulta strojní**

Zadávající katedra/ústav: **Ústav přístrojové a řídící techniky**

Studijní program: **Teoretický základ strojního inženýrství**

Studijní obor: **bez oboru**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Detekce volného prostoru pomocí hloubkových map

Název bakalářské práce anglicky:

Free Space Detection Using Depth Maps

Pokyny pro vypracování:

- Proveďte rešerši aktuálních dostupných senzorů pro snímání hloubkových map včetně zhodnocení jejich principů snímání.
- Proveďte rešerši aktuálních metod zpracování hloubkových map vhodných pro využití v detekci volného prostoru před autonomním robotickým podvozkem.
- Navrhněte SW nástroj v jazyce Python implementující vytipované metody, včetně rozhraní pro zvolený senzor.
- Navržený SW nástroj realizujte a otestujte (bez implementace na autonomním robotickém podvozku).

Seznam doporučené literatury:

[1] N. M. DiFilippo and M. K. Jouaneh, “Characterization of Different Microsoft Kinect Sensor Models,” IEEE Sens. J., vol. 15, no. 8, pp. 4554–4564, Aug. 2015, doi: 10.1109/JSEN.2015.2422611.
[2] S. Choi, Q.-Y. Zhou, and V. Koltun, “Robust Reconstruction of Indoor Scenes,” Jun. 2015, doi: 10.1109/CVPR.2015.7299195.
[3] J. Garstka and G. Peters, “Fast and Robust Keypoint Detection in Unstructured 3-D Point Clouds,” Jul. 2015, vol. 2, doi: 10.5220/0005569501310140.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. Cyril Oswald, Ph.D., U12110.3

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **30.04.2021** Termín odevzdání bakalářské práce: **10.06.2021**

Platnost zadání bakalářské práce: _____

Ing. Cyril Oswald, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Michael Valášek, DrSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

. _____ Datum převzetí zadání

_____ Podpis studenta

Prohlášení

Prohlašuji, že jsem tuto práci vypracoval samostatně s použitím literárních pramenů a informací, které cituji a uvádím v seznamu použité literatury a zdrojů informací.

Datum: ……. …….

podpis

Anotace

Jméno autora	Martin Hasal
Název bakalářské práce	Detekce volného prostoru pomocí hloubkových map
Rozsah práce	34 stran
Školní rok vyhotovení	2021
Škola	České vysoké učení technické v Praze
Vedoucí práce	Ing. Cyril Oswald Ph.D.
Využití práce	Detekce volného prostoru před autonomním robotickým podvozkem
Klíčová slova	Detekce, hloubková mapa, point cloud, detekce ploch, Kinect, Python, RANSAC, ICP, RealSense
Anotace	Práce je zaměřena na návrh a realizaci programu, pro snímání prostoru a detekci ploch pomocí hloubkových map. Je provedena rešerše dostupných senzorů pro snímání hloubkových map a byla provedena rešerše algoritmů pro zpracování hloubkových map. Je navržen algoritmus pro snímání hloubkových map ve formě point cloudů a detekci ploch pomocí metody RANSAC. Tento algoritmus je realizován v jazyce Python. Realizovaný algoritmus je vyzkoušen na kamerách Kinect V1 a RealSense D455.

ANNOTATION

Autor	Hasal Martin
Title of graduation work	Free Space Detection Using Depth Maps
Extend	34 pages
Academic year	2021
School	Czech technical university in Prague
Supervisor	Ing. Cyril Oswald Ph.D.
Application	Free space detection in front of an autonomous robotic chassis
Key words	Detection, depth map, point cloud, plane detection, Kinect, Python, RANSAC, ICP, RealSense
Annotation	The work is focused on the design and implementation of a program for space sensing and surface detection using depth maps. A search of available sensors for scanning depth maps is performed and a search of algorithms for processing depth maps was performed. An algorithm for scanning depth maps in the form of point clouds and surface detection using the RANSAC method is proposed. This algorithm is implemented in Python. The implemented algorithm is tested on Kinect V1 and RealSense D455 cameras.

Obsah

1 Úvod

Nově vznikající aplikace vyžadují od robotů nejen zvýšenou výkonost, ale také to, aby robot byl schopen nabytí určité autonomní činnosti, jako určit polohu dílů umístěných náhodně na pohybujících se dopravnících nebo pro pohyb ve volném prostoru, jak po pracovním prostoru, tak i například po hale. Některé systémy strojového vidění, které v minulosti existovaly, se v současnosti montují do robotů, aby pomohly automatizačním systémům takové úkony vykonávat.

Strojové vidění znamená použití správného optického hardwaru a technik zpracování obrazu, k analýze obrazu a nalezení základních informací o objektech a okolním prostředí. Teorie strojového vidění byla vyvinuta od 70. a 80. let, takže strojové vidění je relativně vyspělou oblastí výzkumu. V posledních letech se zvýšila výkonost informačních technologií a zároveň s tím se snížila cena vysoce kvalitních kamer a zařízení pro zpracování obrazu, takže aplikace strojového vidění v reálném čase lze implementovat pomocí standardních výpočetních platforem. Navíc rozmanitost současných technologií senzorů nabízí široké spektrum jak rozlišovacích schopností, tak cenových kategorií.

Tuto technologii je možné rozdělit do dvou hlavních podskupin: na 2D a 3D. 2D strojovým viděním naváděný systém je schopen zpracovávat náhodně rozmístěné díly vzhledem k robotu na rovné ploše. Na druhou stranu u 3D strojového vidění může naváděný systém zpracovávat náhodně rozmístěné díly vzhledem k robotu ve třech rozměrech a též je schopen přesně zjistit orientaci každého dílu. Zároveň je schopen pohybu a detekci překážek, jak z pohledu schopnosti manipulativního, tak z pohledu samotné mobility celého robotu. V praxi se 2D strojní vidění realizuje digitální kamerou a softwarem, který analyzuje digitální obraz 2D umístění tohoto dílu a jeho orientaci pro jeho manipulaci pomocí robotu nebo při jeho zpracování (například při svařování).

Strojové prostorové vidění (3D) je v dnešní době, čím dál tím více využíváno a zasahuje do mnoha odvětví oborů a způsobů využití a tyto oblastní kruhy se stále rozšiřují například do automobilového průmyslu, aktuálně jako bezpečnostní prvek při řízení a pro budoucí autonomní řízení. V ocelárnách jsou úkoly související s měřením rozměrů obvykle pro pracovníka

vyčerpávající a nudné. Proto vzhledem k tomu, že se stroje ani neunaví, ani se nenudí, můžou stroje poskytnout lepší výkon v tomto odvětví. Tuto práci jsem tedy vybral z možného budoucího využití a mého osobního zájmu. [1]

2 Cíle práce

V rámci této bakalářské práce bude navržen způsob zpracování vstupních dat hloubkové mapy, pro vytvoření 3D mapy volného prostoru, který bude rozlišovat, kde je podlaha, po které se může pohybovat například robot a kde jsou pro něj překážky. Následně bude dle návrhu vytvořen realizační program v programovacím jazyce Python. Cílem tedy bude vytvořit program, který zpracuje vstupní data a vytvoří z nich 3D mapu okolního prostředí se zobrazením podlahy jako volného prostoru pro pohyb.

3 Rešerše

Pro svoji práci budu potřebovat nejdříve vědět, co mám použít k dosažení optimálního cíle. Proto se v této části budu zabývat rozborem možných alternativ, ze kterých si následně vyberu. Jako první budu potřebovat senzor, kterým budu snímat okolní prostředí. Z důvodu mnoha využitelných senzorů a dobré cenové dostupnosti jsem vybral produkt Kinect od společnosti Microsoft, který byl vydán ke konzolím Xbox 360 a produkt RealSense D455 od společnosti Intel, mezi kterými se budu rozhodovat. Následně budu muset získaná vstupní data zpracovat. Zpracování bych rozdělil na několik částí. Na předzpracování, kde použiji filtrační metody obrazu a point cloudu pro zpracování, kde vytvořím 3D point cloud (3D mračno bodů), výpočet podobnosti předchozího point cloudu na následné spojení a na závěr detekci ploch pomocí RANSAC metody.

3.1 Prostorové vidění

Počítačové prostorové vidění je způsob, jak může dané zařízení získávat vstupní vizuální a prostorová data z okolí, které následně může analyzovat a vyhodnocovat, co jsou užitečné informace, dle námi určených parametrů. Vstupní vizuální a prostorová data můžeme získávat jedním až několika senzory. Vizuální data můžeme získat ve tvaru RGB obrazu a prostorová ve tvaru hloubkové mapy. Hloubková mapa je forma obrazu, kde každá hodnota pixelu obsahuje

vzdálenost mezi senzorem a bodem, který reprezentuje daný pixel ve snímaném prostředí. Počítačové prostorové vidění se dá využít pro rozpoznávání objektů, tvarů a jejich rozměrů nebo vytváření mapy 3D prostoru. Všechny tyto aspekty se v dnešní době využívají v průmyslu pro automatizaci, vyhodnocování kvality daného produktu nebo pro prostorové vnímání autonomních automobilů a robotů. Máme několik způsobů, jak můžeme získávat vizuální a prostorové data. Některé z nich popíšu a vysvětlím níže.[2], [3]

3.1.1 Stereo kamera

První ze způsobů, je takzvaná stereo kamera, která funguje na stejném principu, jako naše oči. Stereo kamera má dvě RGB kamery, které mají stejnou ohniskovou vzdálenost a jsou od sebe vzdáleny danou vzdáleností, takže každá má jinou perspektivu vidění. Jako vstup máme tedy nejméně dva obrazové snímky, které nejprve propojíme, a následně se na základě triangulace a podle cíle použití vypočítá buď celá hloubková mapa anebo jen vzdálenost jednotlivých bodů.[2], [4]

3.1.2 Strukturované světlo

Dalším ze způsobů, jak získat prostorová data okolí, je pomocí strukturovaného světla. Zde se hlavně zaměříme na infračervené světelné záření neboli IR. Daný senzor se skládá z infračerveného zářiče a jeho přijímače umístěného vedle něho. Infračervený zářič vyzařuje síť bodů, rovnoměrně rozprostřených a snímač tuto síť snímá. Podle vzdálenosti a nerovnosti povrchu, jsou jinak vzdáleny a uspořádány jednotlivé body než body referenční roviny, podle kterých se pomocí korelace určuje jejich vzdálenost od senzoru. Senzor nedokáže detekovat průhledné plochy, jako skleněné materiály a další infračerveným světlem průhledné materiály. Výrobců produktů s těmito senzory máme několik a některé i popíšu v podkapitolách níže. [5]

3.1.2.1 Kinect

Kinect od společnosti Microsoft byl vydán ke konzolím Xbox 360 pro snímání pohybů a je to cenově dostupný produkt obsahující mnoho senzorů, které se dají při experimentech a projektech využít. Skládá se z RGB kamery, infračerveného senzoru pro snímání hloubkové mapy, infračerveného zářiče, mikrofonu, akcelerometru a v podstavci zabudovaného motoru, který umožňuje naklápění Kinectu. Infračervený zářič je vzdálen od infračerveného senzoru 73mm a

vyzařuje infračervené světlo o vlnové délce 830 nm. Kinect dokáže snímat hloubkovou mapu mezi 800mm až 4000 mm a přesné snímání mezi 1200 mm až 3500 mm od senzoru a rozlišení obrazu hloubkové mapy je 640 × 480 pixelů při 30 snímcích za sekundu. Úhel zorného pole je 57° horizontálně a 43° vertikálně. Celkové rozměry Kinectu jsou 279 × 63 × 38 mm o váze 1360 g. Nevýhodou je potřeba externího napájení. [6], [7]

Obrázek 1 – Kinect pro Xbox360

3.1.2.2 Intel RealSense

RealSense od firmy Intel konkrétně model D455 se skládá ze dvou RGB kamer a IR senzoru. Na rozdíl od Kinectu, je RealSense schopen vytvořit hloubkovou mapu, jak ze stereo kamery, tak i z IR senzoru. Infračervený zářič je vzdálen od infračerveného senzoru 95 mm s čímž dosahuje nepřesnosti měření při vzdálenosti 4 metry na méně než 2%. D455 dokáže snímat hloubkovou mapu v minimální vzdálenosti 520 mm a přesné snímání mezi 600 mm až 6000 mm od senzoru a rozlišení obrazu hloubkové mapy je 1280 x 720 pixelů při 90 snímcích za sekundu. Úhel zorného pole je 87° horizontálně a 57° vertikálně. Celkové rozměry D455 jsou 124 x 26 x 29mm o váze 390 g. Výhodou je napájení přímo z USB portu, není tedy potřeba externího napájení. Pro sledování změny pohybu je použit 6-ti osý senzor Bosch BMI055, který využívá akcelerometr s rozsahem ±4 g a gyroskop s rozsahem ±1000°/s.

Obrázek 2 – RealSense D455

Dále by stály za zmínku produkty od jiných firem, ze série Asus Xtion nebo Ensenso.[8], [9]

3.1.3 ToF

Time of Flight, zkráceně ToF je metoda pro měření vzdálenosti mezi senzorem a měřeným objektem, založena na rozdílu časů mezi vyslaným emitovaným signálem a jeho návratem do senzoru po odrazu od objektu. Následně se vypočte vzdálenost jednotlivých pixelů, čímž se získá hloubková mapa. Je několik druhů signálů, které můžou být použity pro metodu ToF a to nejčastěji je využíváno vlnění zvuku a světla. Světlo má výhodu, že je rychlejší než zvuk, dosáhne větších vzdáleností a senzor má nízkou hmotnost. Použitím infračerveného světla se zajistí redukce šumů okolí a je jednodušší ho rozlišit od okolního světelného záření. ToF za použití světelného záření se rozděluje na modulované světlo (modulated light) a pulzní světlo (pulsed light). ToF kamery měří vzdálenost podle časové vzdálenosti vyslaného světelného vlnění, která odpovídá sférickým souřadnicím. Hloubková mapa ale představuje kartézské souřadnice ve směru optické osy, proto je potřeba transformace ze sférické na kartézské souřadnice.[10], [11]

3.1.3.1 Modulované světlo

Modulované světlo neboli světlo s periodickými časovými změnami intensity. Toto světlo je vizuálně detekováno jako blikání. Pokud je modulační frekvence dostatečně vysoká, světlo bude vnímáno jako spojité. Tedy ToF kamera vyzařuje amplitudu modulovaného světla

s frekvencí mezi 10 až 100 MHz. Je potřeba alespoň 3 bodů v modulaci, pro získání rozměrů signálu. Z těchto bodů se pak získá fáze, amplituda a offset. [12], [13]

3.1.3.1.1 SR4000

SR4000 od společnosti SwissRanger je 3D ToF kamera, která vyhodnocuje vzdálenost bodů hloubkové mapy metodou modulovaného infračerveného světla. Kamera má rozměry 65 x 65 x 68 mm a je schopna získat informace o prostoru v rozlišení 176 x 144 bodů za snímek s frekvencí 50 Hz. Úhel zorného pole je 43° horizontálně a 34° vertikálně. [14]

3.1.3.1.2 Lidar

Light Detection and Ranging (LIDAR) v překladu světelná detekce a měření vzdálenosti je ToF kamera využívající metodu modulovaného světla. Tento senzor je velice rychlý, přesný a s velkým dosahem. Z důvodu své vysoké pořizovací ceně je používán převážně do špičkových a komplexních technologií. Lasery jsou rozděleny na amplitudově modulované spojité vlny (AMCW) a na frekvenčně modulované spojité vlny (FMCW). AMCW moduluje intenzitu světla při zachování stejné frekvence a podle požadované přesnosti měření je zapotřebí vysokorychlostní vysokofrekvenční elektroniku, pro modulaci intenzity světla. U přijímače tento požadavek může být snížen pomocí demodulačního přijímače, který dokáže převést vysokofrekvenční signály do základního pásma. FMCW je založen na změně frekvence vyzařovaného světla. Protože je měřen stejný bod několika frekvencemi, je výsledek přesnější. [15], [16]

3.1.3.2 Pulzní světlo

Tato metoda využívá jeden dlouhý pulz a měří se vzdálenost na základě času po odražení signálu od objektu. Tuto metodu využívá například senzor Tiger Eye nebo rotační scanner Velodyne. [13]

3.1.3.2.1 Rotační scanner: Velodyne

Velodyne je rotační ToF scanner se systémem LIDAR, který získává trojrozměrnou informaci o okolním prostoru ve formě bodového pole, který obsahuje data o okolních objektech. Každý bod reprezentuje souřadnice x, y, z v prostoru od senzoru. Pro analýzu a zpracovaní jednotlivých bodů je zapotřebí velké množství času výpočetního systému. Pokud je tato informace vybrána a jsou zanalyzovány pouze ty, které jsou pro nás důležité, je proces mnohem rychlejší. Velodyne

zachytává 3D informaci o okolním prostředí. Model HDL-64E má úhel zorného pole plných 360° horizontálně a 26,8° vertikálně a to s frekvencí až 15 Hz. Tento sken je osazen 64 rotačními lasery, kde každý z laserů tvoří jeden obvod bodů v měřeném prostoru. Celkem pak Velodyne generuje 1,3 mil bodů za sekundu. Továrně nastavená kalibrace průměrné odchylky je přibližně 4 cm. [17], [18]

3.2 Zpracování obrazu

Zpracování obrazu je velice široký pojem, pod který spadá spousty metod a způsobů jak daný obraz zpracovat, analyzovat a vyhodnotit. Hlavním cílem zpracování obrazu je extrakce informací a zvýšení jeho kvality, aby byl lépe vyhodnocen. Příkladem získání obrazu je snímek pořízen digitálním fotoaparátem, senzory na leteckých strojích, zdravotních zařízeních nebo průmyslovými zařízeními na kontrolu kvality.

Různé technologie zpracování obrazu byly vyvinuty, pro získání a extrakci informace ze snímaných dat. Zpracování obrazu má několik částí, které pro tuto získanou informaci jsou potřeba. Před zanalyzováním obrazu je fáze zvaná předzpracování (preprocessing), který se skládá z operací připravujících obraz na následnou analýzu, který se snaží opravit nebo kompenzovat chyby a je nezbytný hlavně při použití různých senzorů a v různých časech se musí geometricky a radiometricky upravit. Mezi časté způsoby předzpracování patří redukce šumu, kalibrace detektoru, geometrická a radiometrická korekce. Následně je obraz upraven, pro usnadnění získání potřebných informací a snížení rozměrů nepotřebných dat. Nakonec je obraz segmentován a vyhodnocován tak, aby vytvořil digitální mapu s potřebnými informacemi.

Filtrace obrazu je část předzpracování a je jedním z důležitých úkolů při zpracovávání obrazu. Mediánový filtr, nelineární střední filtr, vícestupňový střední filtr jsou některé příklady filtrů, které se pokouší odstranit efekt šumu a zároveň zachovat přechodové hrany obrazu.

Mezi další metody zpracování obrazu patří zostřování (sharpening), segmentace obrazu a detekce hran. Zostřování obrazu je metoda, která má zvýraznit hrany a obrysové čáry. Detaily a hrany obrazu mohou být výraznější. Obraz může být vylepšen jak operací diferenciálu, tak operací gradientu. Segmentace a detekce hran je popsána v následujících podkapitolách. [19], [20]

Konvoluce jádrové matice zvaná Kernel, je častá metoda filtrace obrazu, která jde použít pro ostření, rozostřovaní, detekci hran, atd. Obvykle je to 2D matice například 3x3 nebo 5x5. Operace konvoluce se provádí jednotlivě na každém pixelu a tato operace obsahuje 3 kroky. Prvním krokem konvoluce jádrové matice překryje vstupní obraz, tak aby středový pixel jádrové matice byl zarovnán na pixel obrazu, který má být filtrován. Poté se ve druhém kroku každá hodnota pixelů v okolním poli obrazu vynásobí hodnotou překrývající jádrové matice. Ve třetím kroku se sečtou všechny hodnoty získané z druhého kroku a tato suma se stane hodnotou počítaného pixelu obrazu. Pro filtraci celého obrazu se tato operace provede jednotlivě na všechny pixely v obrazu. [21], [22]

3.2.1 Mediánový filtr

Při pořizování snímků obrazu dochází k nechtěnému šumu. Vyhlazování obrazu má několik různých způsobů, jak obraz zpracovat. Mediánový filtr může nejen ochránit přechodové hrany obrazu, ale také současně eliminovat šum. Filtry jsou v zásadě rozděleny na lineární filtry a nelineární filtry. Příkladným problémem lineárních filtrů při redukci šumu je, že dojde k rozmazání hran, čímž se zhorší správnost výstupu. Zatímco nelineární filtr eliminuje šum pixelů, aniž by způsobil rozmazání okrajů. Je mnoho druhů mediánových filtrů jako standartní mediánový filtr, vážený mediánový filtr (weighted median filter), středně vážený mediánový filtr (center weighted median filter), adaptivní mediánový filtr, atd. Některé z nich si blíže popíšeme níže.

Mediánový filtr je příklad nelineárního filtru a je velmi efektivní při zachování vlastností obrazu. Velikost okna filtru ovlivňuje výkonnostní vlastnosti mediánového filtru. Menší okno zachovává vlastnosti, ale vede k omezení potlačení šumu. V případě většího okna je potenciál potlačení šumu vysoké, ale sníží se zachování detailů hran obrazu.

Standartní mediánový filtr je snadno implementovatelný nelineární filtr pro redukci šumu. Cílem je šum nahradit mediánovou hodnotou okolních pixelů. Hodnota mediánu je jednoduše popsána:

$$Median(P) = \frac{Pi}{2} \cdot [k+1] \qquad (1)$$

Kde $P_1, P_2, \ldots P_k$ jsou sousední pixely. Všechny pixely musí být před použitím filtru ve vzestupném nebo sestupném pořadí. Po provedení třídění bude výsledná skupina $P_{i1} \leq P_{i2} \leq \cdots P_{ik}$, kde k je obvykle liché.

Vážený mediánový filtr (weighted median filter) je jedním z rozšíření klasického mediánového filtru, kde jeho operace jsou stejné s rozdílem, že vážený mediánový filtr pro výpočet střední hodnoty počítá s váženou hodnotou každého váženého prvku.

$$S(i,j) = median(k,l) \in w_{m,n}\{D(i+k, j+l)\} \quad (2)$$

Mediánové hodnoty obrazu se počítají pomocí rovnice (2). Filtr váží posuvné okno o velikosti $m \times n$ pixelů středových souřadnic (i,j) tak, že váha klesá, pokud je mimo střed okna filtrování. Tímto způsobem zpracování přikládá větší důležitost středovému pixelu a tím je potlačen šum. V tomto váženém mediánovém filtru se zachová detail obrázku. Zachování detailu obrázku závisí na váhovém koeficientu. V praxi je pro tento filtr obtížné najít váhový koeficient. Vyžaduje vysoký výpočetní čas.

Středně vážený mediánový filtr (center weighted median filter, ve zkratce CWMF) je podtřída váženého mediánového filtru pro zachování detailů. Tento filtr dává větší váhu do středu okna. Je založen na statických a deterministických vlastnostech. Hodnota středu je dána váhou $w\ (0,0)\ =\ (2k\ +\ 1)$ a všechny okolní pixely váhu rovné 1.

$$W = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2k+1 & 1 \\ 1 & 1 & 1 \end{pmatrix} \quad (3)$$

Filtr je specifikován dvěma parametry, a to velikostí okna a váhou středu. Chování filtru lze snadno upravit změnou hodnoty středové váhy. Filtr dokáže dobře zachovat detaily a redukovat šum, zejména velké a rozlehlé množství šumu. Nevýhodou je, že obraz přijde o mnoho dobrých pixelů, které jsou také upraveny k rozmazání.

Rekurzivně vážený mediánový filtr (Recursive weighted median filter, ve zkratce RWMF) je další podtřída mediánového filtru. V RWM filtru je výstup vytvořen z předchozího výstupu a

vstupu. Výpočet váhy se provádí mnoha technikami, jako je technika optimalizace. Pro každé okno vstupu, které je blíže výstupu první filtrace, může mít větší váhu. První iterace výstupu je brána za referenční a další výpočty nových vah jsou s ní porovnávány a z něj vypočten výstup. Tento postup se opakuje, dokud není dosaženo počtu iterací. RWM filtr je méně efektivní a algoritmus má vysokou složitost.

Adaptivní mediánový filtr je rozšíření standartního mediánového filtru a je založen na přizpůsobení velikosti okna. V prvním kroku detekuje, které pixely obsahují šum, a které ne. V druhém kroku se kolem každého pixelu vyhodnoceného jako šum vytvoří 3 × 3 okno. Ve třetím kroku se zkontroluje, jestli je šum vysoký, pokud ano, pak se zvětší okno na 5 × 5. Ve čtvrtém kroku se vypočte medián a uloží se hodnoty daných pixelů.

Algoritmus založený na rozhodování (Decision Based Algorithm, ve zkratce DBA) je algoritmus pro detekování šumu a poškození pixelu založené na kontrole hodnoty daného pixelu a jeho maximální nebo minimální hodnoty okolních pixelů. Minimální hodnota představuje šum zvaný pepř a maximální hodnota šum zvaný sůl. Pokud má pixel hodnotu od 0 do 255, poté je brán jako pixel bez šumu. Pokud ale pixel nemá hodnotu mezi 0 až 255, najde se mediánová hodnota a nahradí hodnotu pixelu hodnotou mediánu. Při vysokých hodnotách šumu je medián roven 0 nebo 255, což opět způsobí šum, proto se v tomto případě nahradí hodnota pixelu libovolným sousedním pixelem. [20], [23] - [25]

3.2.2 3D Point-cloud (3D mračno bodů)

Po získání dat ze senzorů snímajících hloubkovou mapu, je následně potřeba tyto data nějak zpracovat. Point cloud je jednou z aplikací pro zpracování těchto dat. Point cloud zpracuje získané data v body rozprostřené v 3D prostoru. S rostoucí technologií 3D snímání se 3D point cloud stal důležitým a praktickým představitelem 3D objektů a okolních prostředí. Tato technologie má řadu aplikací v oblasti, průmyslového modelování, dálkového průzkumu Země, 3D mapování, generování digitálních modelů nadmořské výšky nebo zemědělství.

Klíčovou výzvou při zpracovávání 3D Point cloudu je zpracování obrovského množství dat, kvůli nejednotným vzorkům a šumu. K tomu je zapotřebí komprese a filtrace dat point cloudu. V této souvislosti jsou měřené chyby způsobené kompresí nedílnou součástí. To znamená, že

musíme vyhodnotit degradaci kvality komprimovaného 3D point cloudu ve srovnání s příchozím 3D point cloudem. K vyřešení tohoto problému byly zváženy vzdálenosti bod na bod. Nejprve je pro každý bod v jednom point cloudu identifikován odpovídající bod z druhého point cloudu. Poté je jako základ pro měření použit průměr nebo maximum euklidovských vzdáleností mezi takovými dvojicemi bodů.

Point cloud získaný z jednoho pohledu nestačí k pokrytí úplné 3D scény v důsledku členitosti scény, stínů objektů a šumu. Proto je zapotřebí spojit několik těchto point cloudů získaných z různých úhlů pohledu do sebe. Pomocí aplikace registrovaných point cloudů, který tyto data spojí do společného souřadného systému pro zobrazení 3D scény. Registrovaný point cloud je klíčovou součástí a tvoří základ pro mnoho následných 3D úkonů, včetně detekce objektů, rozpoznávání objektů a lokalizace. [25] - [26]

3.2.2.1 Matice kamery

Ve 3D prostředí máme světový souřadnicový systém a kamerový souřadnicový systém. Ty jsou spojeny rotací a translací. Těchto šest parametrů (3 rotace a 3 translace) se nazývají vnější parametry kamery.

Pro definování polohy bodů v prostoru potřebujeme počáteční bod (0, 0, 0) a osy X, Y, Z. Tímto způsobem můžeme zjistit souřadnice libovolného bodu měřením jeho vzdálenosti od počátku podél os X, Y a Z. Tento souřadnicový systém se označuje jako světový souřadnicový systém. Bod ve světovém souřadnicovém systému jsou souřadnice dány vztahem (X_w, Y_w, Z_w).

Obraz bude zachycen pomocí kamery, která bude umístěna v prostoru, a proto nás zajímá kamerový souřadnicový systém připojený k této kameře. Kamera je umístěna na libovolném místě (t_x, t_y, t_z) v prostoru vzhledem ke světovým souřadnicím. Rotace ve 3D je zachycena pomocí tří parametrů a tvořena maticí 3 × 3. Světové souřadnice a souřadnice kamery jsou spojeny rotační maticí R a tříprvkovým translačním vektorem $\boldsymbol{t}$. To znamená, že bod, který měl ve světových souřadnicích hodnoty souřadnic (X_w, Y_w, Z_w), bude mít v souřadnicovém systému kamery různé hodnoty souřadnic (X_c, Y_c, Z_c). Translační vektor je tvořen maticí 3 × 1 a je připojen jako sloupec na konci rotační matice 3 × 3. Získaná matice 3 × 4 se nazývá vnější matice.

$$[\boldsymbol{R}|\boldsymbol{t}] = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = [\boldsymbol{R}|\boldsymbol{t}] \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix} \quad (5)$$

Jakmile získáme bod v 3D souřadnicovém systému kamery pomocí rotace a translace převedené na souřadnice světových bodů, jsme v pozici, kdy můžeme promítnout bod na rovinu obrazu, abychom získali umístění bodu v obraze. Kamera má určité vlastnosti, které je zapotřebí zohlednit a to vnitřní maticí $\boldsymbol{K}$.

Obrazová rovina je umístěna ve vzdálenosti f (ohnisková vzdálenost) od optického středu. Pomocí podobnosti trojúhelníků můžeme obraz (x, y) v kamerovém (X_c, Y_c, Z_c) je dán vztahem:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} \quad (6a)$$

Pixely v obrazovém snímači však nemusí být čtvercové, a proto můžeme mít dvě různé ohniskové vzdálenosti f_x a f_y. Optický střed (c_x, c_y) kamery se nemusí shodovat se středem kamerového souřadnicového systému. Kromě toho může být mezi osami x a y senzoru kamery malé zkreslení γ. Při zohlednění všech výše uvedených skutečností lze vnitřní matici kamery formulovat takto:

$$\boldsymbol{K} = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (6b)$$

Výpočet bodu je získán rovnicí:

$$P = \boldsymbol{K} \times [\boldsymbol{R}|\boldsymbol{t}] = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

[27]

3.2.3 Voxelizace

Voxelizace je způsob komprese objemných point cloudů, pro usnadnění následné manipulace a urychlení výpočetního času. Vytvoří síť voxelů o určité velikosti v point cloudu. Tato voxelová reprezentace je spletitá s kulovou konvoluční jádrovou maticí. Koule funguje jako integrální operátor pro všechny voxely obsahující body point cloudu. Konvoluce udává průměr voxelů, které jsou uvnitř point cloudu. Hodnoty průměru se používají k identifikaci podstatných oblastí a z těchto klíčových bodů se extrahují. Všechny části algoritmu jsou vysoce paralelizované a budou proto vypočítané velmi rychle. Dále můžeme snadno simulovat point cloud s nižším rozlišením se zvětšením velikosti voxelů.

Chceme tedy navrhnout mapu z bodů $\boldsymbol{p}_i = (p_x, p_y, p_z)$ na voxely $\boldsymbol{v}_i = (v_x, v_y, v_z)$. Abychom toho dosáhli, musíme nejdříve převést P na počátek V a to nastavením offsetového vektoru q_{min}. Poté zpevníme obal představující vnější povrch objektu aproximací všech $\boldsymbol{p}_i$ do geometrického středu jejich příslušného voxelu $\boldsymbol{v}_i$:

$$\boldsymbol{v}_i = \begin{cases} 1 & pokud\ \boldsymbol{p}_i \in [\boldsymbol{v}_i, \boldsymbol{v}_i + \lambda] \\ 0 & jinak \end{cases} \tag{8}$$

Alternativou, která více přesně zohledňuje těžiště bodů n, pak obsažených $\boldsymbol{v}_i$ v souřadnicích voxelů.

$$\boldsymbol{v}_i = \frac{1}{n} \cdot \sum_{n=1}^{n} \boldsymbol{p}_i \tag{9}$$

Zisk přesnosti však nemá cenu ve srovnání s prodloužením doby zpracování. [28], [29]

3.2.4 Odhad vektoru normály

Odhad normálového vektoru point cloudu používá algoritmus určený ke K (počtu) – nejbližších – sousedních bodů s rovinou, tato rovina měla směr dle pravidla pravé ruky, to je plocha v normálovém vektoru bodů. Nejprve struktura matice K – nejbližších – sousedních bodů, přijímá body $P_1, P_2, \ldots, P_k$, kde k představuje

$$P = \frac{1}{k}\sum_{i=1}^{k} P_i \tag{10}$$

vztahující ke kovarianční matici CV (11).

$$CV = \begin{pmatrix} P_i & - & P \\ \dots & & \dots \\ P_k & - & P \end{pmatrix} \cdot \begin{pmatrix} P_i & - & P \\ \dots & & \dots \\ P_k & - & P \end{pmatrix} \tag{11}$$

Vzhledem k tomu, že kovariance je skutečná symetrická matice, použijeme iterativní řešení matice, podle modelu nejmenší vlastní hodnoty tohoto vektoru jako point cloud povrch namísto vektoru. Vytvoříme si $v_0 = (1, 1, 1)$ pro počáteční iterační vektor,

$$\begin{aligned} u_0 &= \frac{u_0}{\max(v_0)} \\ v_m &= CV^{-1} \cdot u_{m-1} \\ u_m &= \frac{v_m}{\max(v_m)} \end{aligned} \tag{12}$$

u_m je charakteristická hodnota odpovídajícího vektoru funkce. [30]

3.2.5 k-dimenzionální strom (KD-tree)

K-dimenzionální strom (KD-tree), kde K znamená rozměr prostoru. Tréninkovou instanci lze označit uzlem v KD-stromu. Každý uzel KD-stromu lze rozdělit na dva podprostory a speciální člen s názvem diskriminace, který se používá k označení režimu rozdělení v aktuální úrovni stromu. Všechny podprostory jsou rozděleny na dvě části, na levý podprostor a pravý podprostor, nebo na horní podprostor a spodní podprostor. V dotazu na nejbližšího souseda můžeme za účelem zlepšení rychlosti dotazu přidat dalšího člena, který ukazuje na jeho nadprostor v uzlu. Konstrukce KD-stromu na K-dimenzionální datové sadě představuje oddíl K-dimenzionálního prostoru tvořený K-dimenzionální datovou sadou. To znamená, že každý uzel ve stromu odpovídá oblasti K-rozměrného obdélníku.[31], [32]

3.2.6 MLS (Moving Least Squares – metoda klouzavých nejmenších čtverců)

Pro přizpůsobení dat je navržena metoda klouzavých nejmenších čtverců (MLS). Ve srovnání s tradiční metodou nejmenších čtverců má více výhod a široce se používá, například při

generování geometrického modelu. My jej použijeme ve spojení s KD-stromu pro vyhlazení povrchu po kompresi point cloudu. V některých případech, kdy je třeba přizpůsobit data, musí křivka nebo povrch projít některými klíčovými body, musíme vzít v úvahu podmínky interpolace pro metodu nejmenších čtverců nebo pohybujících se nejmenších čtverců. MLS metodu můžeme také zapsat

$$f(x) = \sum_{i=1}^{m} \boldsymbol{p_i}(x)\boldsymbol{a_i}(x) = \boldsymbol{p}^T(x)\boldsymbol{a}(x) \tag{13}$$

kde

$$\begin{aligned} \boldsymbol{a}(x) &= (a_1(x), a_2(x), \dots, a_m(x))^T \\ \boldsymbol{p}(x) &= (p_1(x), p_2(x), \dots, p_m(x))^T \end{aligned} \tag{14}$$

je základní funkcí. Běžně používáme kompletní polynomiální základnu, jako je

pro 2D

Lineární	$\boldsymbol{p}(x) = (1, x, y)^T$	$(m = 3)$
Kvadratická	$\boldsymbol{p}(x) = (1, x, y, x^2, xy, y^2)^T$	$(m = 6)$

(15a)

Pro 3D

Lineární	$\boldsymbol{p}(x) = (1, x, y, z)^T$	$(m = 4)$
Kvadratická	$\boldsymbol{p}(x) = (1, x, y, z, x^2, y^2, z^2, xy^2z, xz)^T$	$(m = 10)$

(15b)

Koeficientový vektor $\boldsymbol{a}(x)$ je určen minimalizací vážené diskretizace L_2 normy definované jako:

$$J = \sum_{i=1}^{n} \omega(x - x_i)[\boldsymbol{p}^T(x_i)\boldsymbol{a}(x) - \boldsymbol{y}_i]^2 = [\boldsymbol{P} \cdot \boldsymbol{a}(x) - y_i]^T \cdot \boldsymbol{W} \cdot [\boldsymbol{P} \cdot \boldsymbol{a}(x) - \boldsymbol{y_i}] \tag{16}$$

kde $\omega_i(x) > 0$ je vážená funkce

$$\boldsymbol{P} = \begin{bmatrix} p_1(x_1) & \cdots & p_m(x_1) \\ \vdots & \ddots & \vdots \\ p_1(x_n) & \cdots & p_m(x_n) \end{bmatrix} \tag{17}$$

$$\boldsymbol{W} = \begin{bmatrix} \omega_1(x) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \omega_n(x) \end{bmatrix} \tag{18}$$

$$\boldsymbol{a}(x) = \boldsymbol{A}^{-1}(x)\boldsymbol{B}(x)\boldsymbol{y_i} \tag{19}$$

$$\boldsymbol{A}(x) = \boldsymbol{P}^T\boldsymbol{W}(x)\boldsymbol{P} \tag{20}$$

$$\boldsymbol{B}(x) = \boldsymbol{P}^T\boldsymbol{W}(x) \tag{21}$$

Teď už víme vše, co potřebujeme a můžeme dosadit do (13). [33], [34].

3.2.7 ICP (Iterativní nejbližší bod)

ICP je registrační algoritmus. Vstupem jsou dva point cloudy a počáteční transformace, která zhruba vyrovná mračno zdrojového bodu s mračnem cílového bodu. Výstupem je transformace, která pevně vyrovná dva point cloudy. Vysvětlím zde dvě varianty ICP, bod-na-bod ICP a bod-na-plochu ICP.

Obecně platí, že algoritmus ICP iteruje ve dvou krocích:

1. Najde korespondující sadu $\boldsymbol{\mathcal{K}} = \{(\mathbf{p}, \mathbf{q})\}$ z cílového point cloudu $\boldsymbol{P}$ a zdrojového point cloudu $\boldsymbol{Q}$ transformovaného aktuální transformační maticí $\boldsymbol{T}$.
2. Aktualizuje se transformace $\boldsymbol{T}$ minimalizací objektivní funkce $E(\boldsymbol{T})$ definované přes sadu korespondence $\boldsymbol{\mathcal{K}}$. Různé varianty ICP používají různé objektivní funkce $E(\boldsymbol{T})$.

Nejprve si ukážeme ICP algoritmus bod-na-bod (point-to-point):

$$E(\boldsymbol{T}) = \sum_{(\boldsymbol{p},\boldsymbol{q}) \in \mathcal{K}} \|\boldsymbol{p} - \boldsymbol{T}\boldsymbol{q}\|^2 \tag{22}$$

Algoritmus ICP typu bod-na-plochu (point-to-plane) používá jinou objektivní funkci:

$$E(\boldsymbol{T}) = \sum_{(\boldsymbol{p},\boldsymbol{q}) \in \mathcal{K}} \Big((\boldsymbol{p} - \boldsymbol{T}\boldsymbol{q}) \cdot \boldsymbol{n_p}\Big)^2 \tag{23}$$

kde $\boldsymbol{n_p}$ je normálou bodu $\boldsymbol{p}$. ICP algoritmus bod-na-plochu má rychlejší konvergenční rychlost než ICP algoritmus bod-na-bod. [38] - [40]

3.2.8 Optimalizace grafu pozice

Optimalizace grafu pozice se používá k získání konečných pozic $\{\boldsymbol{T}_i\}$ v globálním rámci. Graf pozice má dva klíčové prvky: uzly $\boldsymbol{P}_i$ a hrany $\boldsymbol{R}_i$. Uzel je část geometrie $\boldsymbol{P}_i$ spojená s pólovou maticí $\boldsymbol{T}_i$, která transformuje $\boldsymbol{P}_i$ do globálního prostoru. Sada $\boldsymbol{T} = \{\boldsymbol{T}_i\}$ jsou neznámé proměnné, které mají být optimalizovány. Nastavili jsme globální prostor jako prostor $\boldsymbol{P}_0$. $\boldsymbol{T}_0$ je tedy matice identity. Ostatní matice pozic jsou inicializovány akumulací transformace mezi sousedními uzly.

$$E(\boldsymbol{T}) = \sum_i f(\boldsymbol{T}_i, \boldsymbol{T}_{i+1}, \boldsymbol{R}_i) + \sum_{i,j} f(\boldsymbol{T}_i, \boldsymbol{T}_j, \boldsymbol{T}_{i,j}) \tag{24}$$

Sousední uzly mají obvykle velké překrytí a lze je zaregistrovat pomocí metody bod-na-plochu ICP. Hrana grafu pozice spojuje dva uzly (kusy geometrie), které se překrývají. Každá hrana obsahuje transformační matici $\boldsymbol{T}_{i,j}$, která srovnává zdrojovou geometrii $\boldsymbol{P}_i$ s cílovou geometrií $\boldsymbol{P}_j$. Párová registrace je náchylná k chybám. Špatně spárované zarovnání párů může převyšovat počet správně zarovnaných párů. Proto je přidáno do procesu $\boldsymbol{L} = \{l_{i,j}\}$, kde $l_{i,j}$ nabývá hodnot $< 0{,}1 >$ a modeluje platnost příslušného uzavření smyčky.

$$E(\boldsymbol{T}, \boldsymbol{L}) = \sum_i f(\boldsymbol{T}_i, \boldsymbol{T}_{i+1}, \boldsymbol{R}_i) + \sum_{i,j} l_{i,j} f(\boldsymbol{T}_i, \boldsymbol{T}_j, \boldsymbol{T}_{i,j}) + \mu \sum_{i,j} \psi(l_{i,j}) \tag{25}$$

Rozdělují se tedy hrany grafu pozice do dvou tříd. Na hrany trajektorie spojují dočasně blízké sousední uzly. Algoritmus lokální registrace, jako je ICP, je může spolehlivě sladit a na smyčky uzavírající hrany $\psi(l_{i,j})$ spojují všechny nesousedící uzly a $\psi(l_{i,j}) = (\sqrt{l_{i,j}} - 1)^2$. Parametr μ vyvažuje sílu předchozích vztahů. Kromě transformační matice $\boldsymbol{T}_i$ může uživatel nastavit informační matici Λ_i pro každou hranu. Pokud Λ_i je nastavena pomocí funkce get_information_matrix_from_point_clouds z knihovny Open3D, ztráta na této hraně grafu pozice se blíží odpovídající sadě mezi dvěma uzly, s váhou procesem váhy μ. [38], [39]

3.2.9 Pásmový filtr – pásmová propust

Metodou následného zpracování (post-processingu) point cloudu je pásmový filtr, který slouží jako prostředek umožňující odstranění bodů z cloudu, které nejsou ve stanoveném rozsahu. To umožňuje upravit point cloud v libovolném směru souřadnic. Metoda pásmová propust přijímá parametry pro horní a dolní mez a směr podél osy x, y nebo z. Tuto metodu zde uvádím, protože s níž jsem schopen zpřesnit detekci ploch, které zrovna hledám. [40]

3.2.10 Detekce ploch

Aby mobilní zařízení jako robot, manipulátor či automobil byl schopen vizuálního vnímání okolního prostředí a za předpokladu, že se povětšinou bude dané zařízení pohybovat po rovných plochách, je zapotřebí rozeznat podlahu, po které se pohybuje, od okolních překážek. Proto je potřeba detekce ploch. Metod pro jejich určení je hned několik.

Po filtraci získaných dat je na výběr buď pokračovat dále a udělat segmentaci obrazu a detekci hran nebo převést data do 3D point cloudu a následně z tohoto point cloudu detekovat plochy.

3.2.10.1 RANSAC

RANdom SAmple Consensus (RANSAC) je výkonná technika pro odhad parametrů geometrického modelu. S vývojem autonomních systémů a umělé inteligence exponenciálně roste využití senzorů. Skutečné datové sady senzorů však obsahují spoustu šumů v neorganizované nebo nerovnoměrné formě, které mohou způsobit chybný odhad. Náhodně vybrané body z datové sady lze rozdělit do dvou kategorií na základě jejich vztahu k ostatním bodům: inliers a outliers. RANSAC odhaduje podmnožinu dat, která se hodí k modelu, kde iterativně rozděluje outliers (data, která se nehodí k modelu) a inliers (data, která jsou v souladu s modelem). Tyto outliers brání cestě k nalezení přesného řešení vložením chyby do vypočítaného výsledku. Pravděpodobným řešením tohoto problému je tedy odstranit tyto outliers z výpočtu a poté implementovat aproximační potup k nalezení optimálního výsledku.

RANSAC zahrnuje dvoufázový proces, kdy se první krok zabývá optimalizační technikou vzorkování a druhý krok se týká procesu ověřování modelu. Je náhodně vybrána minimální podmnožina vstupních dat a z této podmnožiny jsou odhadnuty parametry vzorku pro ověření

modelu. Tento předpokládaný model se poté vyhodnotí na úplném datovém souboru a jeho konzistence se určí iterativně, dokud porovnání modelu s lepšími inliers nepředběhne aktuální nejlepší model a neklesne pod předdefinovanou prahovou hodnotu (obvykle 1% - 5%.). Výběr této předpokládané minimální podmnožiny dat tvoří klíčový bod vyhodnocení tohoto iteračního algoritmu.

Nalezení dostatečné korespondence může být navíc obtížným úkolem mezi dvěma podobnými obrazy, pokud se podobné objekty objevily na různých obrázcích kvůli nesouladu způsobenému šumy. RANSAC je osvědčená metoda řešení tohoto problému obnovením vícepohledové transformace, protože odstraňuje odlehlé hodnoty a vypočítává výsledek na základě maximálního množství inliers nalezených v datové sadě. Ale jeden problém s touto metodou je a to náhodné vzorkování podmnožiny v iteračním procesu. Někdy se může stát, že náhodně vybrané body datové sady nebudou konvergovat k řešení a proces se stane výpočetně nákladově neefektivním. Metoda sjednocování ploch a detekce ploch RANSAC může způsobit nesprávný výsledek, pokud data obsahují malé kroky (např. Obrubníky, schodiště atd.). Nicméně tato metoda je robustní a nejvíce příznivá, pro mé použití. [41], [42]

3.2.10.2 Segmentace obrazu (image segmentation)

Segmentace obrazu je důležitým krokem při zpracování obrazu i důležitou částí počítačového vidění. Segmentace obrazu provádí rozdělení důležitých informací, které dále může rozdělovat na podkategorie a rozdělení od nepotřebných dat. Segmentace obrazu je proces extrakce smysluplných charakteristik nebo oblastí obrazu. Těmito charakteristikami mohou být původní charakteristiky obrazu, jako je hodnota šedé v pixelu, barva, odrazové prvky a textury atd. Může to být také prostorové spektrum, například prvky histogramu. Účelem segmentace obrazu je rozdělit obraz na několik neprotínajících se oblastí, díky nimž mohou být oblasti konzistentní a vlastnosti sousedních oblastí mají zjevný rozdíl. Segmentace obrazu je jedním z nejdůležitějších problémů výzkumu počítačového vidění. Podle úrovně zapojení uživatele jsou algoritmy segmentace obrazu rozděleny do dvou skupin, kterými jsou automatická segmentace a poloautomatická segmentace a může být realizována čtyřmi různými principy, prahovou metodou, zónovou metodou, hraniční metodou a metodou hran. Metoda segmentace prahů je relativně jednoduchá a její zpracování je velice rychlé. Existuje mnoho způsobů, jak určit

prahovou hodnotu, například průměrnou hodnotou šedi, stavovou metodou, iterační metodou, metodou automatického prahu založeného na maximální odchylce. [20], [43]

3.2.10.3 Detekce hran (Edge detection)

Detekce hran (Edge detection) je jedním z nejklasičtějších výzkumných témat v oblasti počítačového vidění a zpracování obrazu. Hlavním účelem detekce hran je měřit, detekovat a lokalizovat změnu stupně šedé v šedém obrazu. Hrany definují hranice mezi oblastmi v obraze, což pomáhá při segmentaci a rozpoznávání objektů. Práce detektoru hrany nejen odděluje hrany od obrazu, ale také rozlišuje hranu a šum a přesně kalibruje polohu hrany. Podstatou detekce hran je tedy použití některých algoritmů k extrakci hraniční čáry mezi objektem a pozadím v obraze. Hrana je definována jako regionální hranice ostré změny v šedé stupnici v obraze. Bylo zavedeno mnoho operátorů, kteří provádějí detekci hran v různých polích obrazů, přestože má diferenciální vzorec stejný tvar. Ne každý operátor však podá dobrý výkon, záleží na kvalitě obrazu, jako jsou světelné podmínky, přítomnost objektů podobné intenzity, hustota hran ve scéně a šum. Kromě toho je při výběru vhodných technik detekce hran důležité pro lepší efektivitu provést předtím kvalitní předzpracování obrazu.

Metoda detekce hran obrazu založená na gradientu, kde se předpokládá, že na pixelech okrajů je vysoký gradient. Tato metoda je založena na směrové derivaci prvního řádu. Liší se také podle metody nebo vážených koeficientů.

Sobelův operátor se skládá ze dvou 3 × 3 konvolučních jádrových matic. Jedna jádrová matice je určena pro horizontální změny a druhá je otočená o 90 ° pro vertikální změny, a to následovně:

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \quad G_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix} \tag{26}$$

Sobelovy masky jsou navrženy tak, aby maximálně reagovaly na hrany probíhající svisle a vodorovně vzhledem k mřížce obrazových bodů a tyto směrové hrany se nakonec spojí a určí absolutní velikost a směr gradientu. Sobelův operátor používá pouze celočíselné hodnoty pro koeficienty k vážení a určení gradientové aproximace obrázků.

Robertův operátor je operátor, který provádí jednoduchý a rychlý přístup k určení měření prostorového gradientu 2-D obrazu. Operátor se skládá z dvou 2 × 2 konvolučních matic takto:

$$G_x = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \quad G_y = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{27}$$

Robertův operátor je přiřazen k maximální reakci na hrany běžící pod úhlem 45° vzhledem k mřížce pixelů. Absolutní velikost a směr přechodu jsou určeny kombinací těchto matic, podobně jako u Sobelova oprátoru.

Prewittův operátor je operátor, který vytváří obraz, kde vyšší hodnota ve stupních šedi indikuje přítomnost hrany mezi dvěma objekty. Tento operátor je podobný Sobelovu operátoru a používá se k detekci svislých a vodorovných okrajů v obrazech. Operátor se skládá z dvou 3 × 3 konvolučních matic takto:

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad G_y = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} \tag{28}$$

Canny je operátor, který je obecně známý jako optimální pro detekci hran. Má nízkou chybovost a je robustní. Canny operátor převede vstupní obraz na obraz ve stupních šedi a poté provede redukci šumu vyhlazením pomocí Gaussova filtru následujícím způsobem:

$$\ddot{G}(x) = -(\frac{x}{\sigma^2})e^{-\frac{x^2}{2\sigma^2}} \tag{29}$$

Jednoduchý 2-D derivační operátor byl aplikován na vyhlazený obraz se zvýrazněnou oblastí hran. Algoritmus poté sleduje podél horní části těchto hran a nastaví na nulu všechny pixely, které ve skutečnosti nejsou na vrcholu hran, a poskytne tenké čáry na výstupu, co představují hrany obrazu.

$$C_x(x,y) = -(\frac{j}{\sigma^2})e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{30}$$

$$C_y(x,y) = -(\frac{i}{\sigma^2})e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{31}$$

Velikost gradientu a směr gradientu každého pixelu je stanoveno pomocí (30) a (31). Poté se provede proces potlačení k identifikaci pixelů hran. [20], [47] - [46]

3.2.10.3.1 Klastrování (Clustering)

Metoda klastrování je efektivní metoda segmentace obrazu. Mezi technikami analýzy obrazu se klastrování obvykle provádí pro klasifikaci obrazu, pokud nejsou poskytnuty žádné informace o obraze. Klastrování rozděluje obrazovou scénu do skupin podobných objektů. Klasické metody klastrování trpí nedostatečnými výsledky způsobenými mnoha faktory, včetně vysoké dimenzionality a komplexního prostorového vztahu mezi sousedními pixely. Ve srovnání s jinými metodami klastrování, jako jsou k-means, k nearest neighbors nebo hierarchické klastrování, má fuzzy k-means větší flexibilitu při přiřazování příslušnosti každého pixelu ke clusteru během jeho iterací. Ve skutečnosti fuzzy k-means přiřazují každému pixelu k proporce dle relevantnosti a k třídám obrazu. Tato funkce umožňuje přehodnocení správnost pixelu a možného přechodu do jiné třídy obrazu v dalších iteracích algoritmu a představuje tak vhodnější přirozený aspekt nejistoty v datech. [47], [48]

3.2.10.3.2 CNN (konvoluční neuronová síť)

V poslední době konvoluční neuronové sítě (CNN - convolutional neural networks) převládají v řadě výpočetních oblastí. Konvoluční neuronové sítě se skládají z neuronů, které se samy optimalizují učením. Každý z neuronů obdrží vstup a provede operaci. Od vstupních obrazových vektorů po konečný výstup bude celá síť stále vyjadřovat jednu vnímavou funkci. Poslední vrstva bude obsahovat funkce ztráty spojené s třídami. CNN se primárně používají v oblasti rozpoznávání vzorů v obrazech. To umožňuje zakódovat do architektury specifické funkce obrazu a vytvořit tak síť. Jednou z rozšířených konvolučních neuronových sítí, je Regionální-CNN.

Regionální-CNN (R-CNN) přijímá vstupní snímek a pomocí funkcí map vytvořených konvolučními vrstvami vytváří návrhy tam, kde existuje možnost existence objektu. To znamená, že R-CNN spouští klasifikátor na základě každého návrhu a testuje pravděpodobnost přítomnosti objektu a pokud je pravděpodobnost vyšší než prahová hodnota, návrh bude označen, a proto R-CNN považuje síť za několik samostatných částí. Pomocí vytvořených map extrahuje ohraničující boxy a identifikuje třídu objektů s nejvyšší pravděpodobností odpovídající každému

ohraničujícímu rámečku. Dalším modelem založeným na R-CNN je faster R-CNN, který snižuje počet návrhů a přijímá další opatření ke snížení doby detekce na základě potřeb aplikací v reálném čase. Faster R-CNN funguje zhruba 200krát lépe než původní R-CNN. Toto číslo však závisí na použité síti CNN. [49] - [51]

4 Návrh a realizace

4.1 Pseudo algoritmus

- Start
- Načtení RGB a hloubkové mapy
- Prvotní vytvoření point cloudu
- Endless loop:
    - Načtení RGB a hloubkové mapy
    - Vytvoření následujícího point cloudu
    - Spojení point cloudů
    - Filtrace (kd-strom a MLS)
    - Detekce podlahy (RANSAC)
    - Vizualizace

4.2 Seznam použitých knihoven

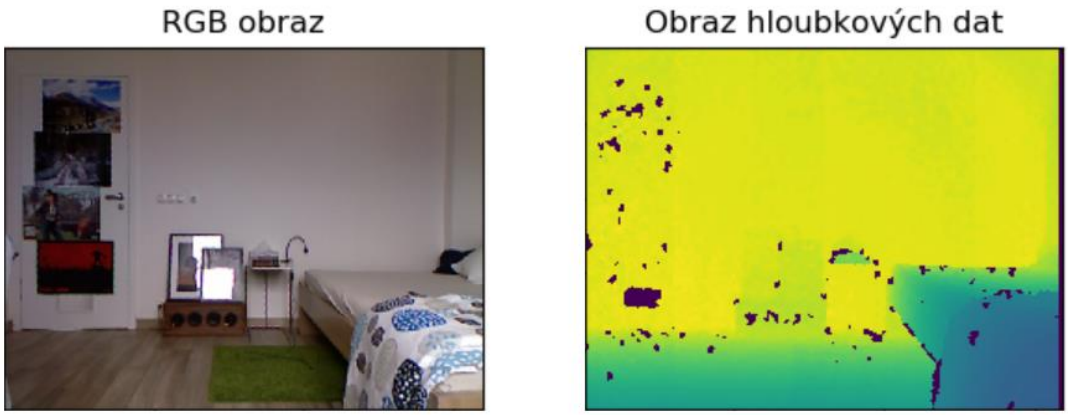
- Numpy
- Pykinect
- Pyusb
- Pyrealsense2
- Open3D
- OpenCV

V této části se budu zabývat popisem navrhovaného programu a řešením problematiky, která v průběhu nastala. Jako první jsem se rozhodl použít pro mou bakalářskou práci Kinect pro Xbox360. Vhledem k jednomu z nejdéle dostupných na trhu, byla očekávána dobrá dostupnost a množství informací, které by mi mohly pomoci k požadovanému cíli. Protože jde o zpracovávání dat, zvolil jsem proto programovací jazyk Python. Ten jako většina programovacích jazyků má možnost implementování knihoven, které slouží k usnadnění například matematickým operacím nebo v tomto případě pro

získávání dat z Kinectu. Na základě vypracovaných rešerší jsem usoudil, že ideální kandidát bude knihovna libfreenect2, která má dobrou dokumentaci a příkladných programů. Po dlouhé době neúspěchu instalace, jsem došel k informaci, že tuto knihovnu lze nainstalovat na operační systém Ubuntu, nikoliv na Windows, který používám. Použil jsem nakonec knihovnu pykinect s méně dostupnými informacemi a podporou, ale schopnou instalace na Windows.

Jako první je zapotřebí zjistit zda je Kinect zapojen a komunikuje. Této otázky dosáhnu pomocí knihovny pyusb, díky němuž se podívám na všechny USB porty a zjistím, zda je k některému z nich připojen Kinect a jestli komunikuje. Pokud ano, nastaví se základní parametry potřebné pro chod programu, v opačném případě se vypíše chybové hlášení. Když už máme vše připravené pro započnutí snímání, první fází bude vytvoření prvotního point cloudu.

Na začátku se spustí funkce, kterou jsem nazval RGB-D, ta načte dvojici vstupních dat RGB obrazu a hloubkovou mapu. Obě se po načtení uloží a na konci funkce se pro bezpečnost zkontroluje, zda máme data opravdu uložené.



Obrázek 3 – Vstupní data z Kinectu, vlevo RGB obraz, vpravo obraz hloubkových dat

Dále se pomocí datasetu SUN RGB-D[52] zkombinuje RGB a hloubková mapa do formátu hloubkových dat RGB-D, která nese informaci o hloubce převzaté ze snímačů. Abychom ji byli schopni vyjádřit ve formátu point cloudu, je zapotřebí vypočítat jednotlivé body. Z hloubkových dat vyjádřených ve 3D poli a parametrů o kalibraci kamery použiji vzorec (7)

pro výpočet. Ten se skládá z vnitřní matice kamery (6b) a vnější matice kamery (4). Po dosazení daných parametrů do vnitřní matice kamery, vypadá matice takto:

$$\boldsymbol{K} = \begin{bmatrix} 522{,}259 & 0 & 330{,}18 \\ 0 & 523{,}419 & 254{,}437 \\ 0 & 0 & 1 \end{bmatrix} \tag{32}$$

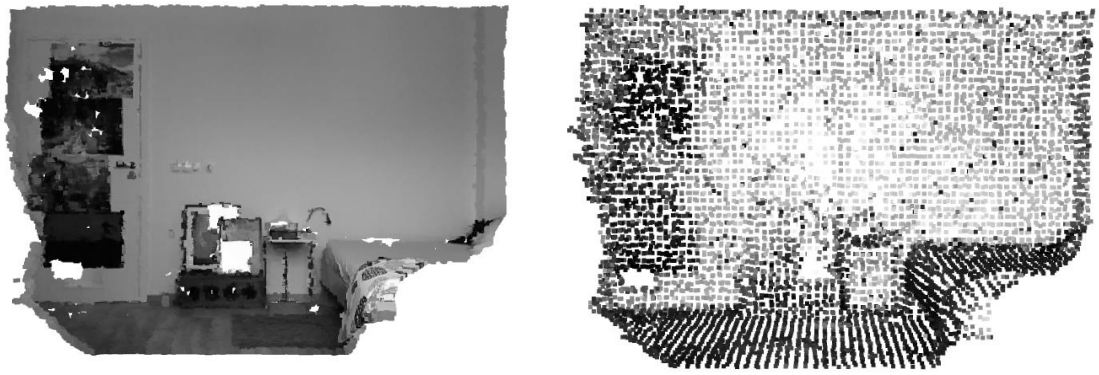
Protože je to prvotní tvorba point cloudu, budu brát, že vnější matice kamery bude ve tvaru jednotkové matice:

$$[\boldsymbol{R}|\boldsymbol{t}] = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{33}$$

Ta se následně bude měnit podle změn polohy a rotace kamery pro následující point cloudy. Vypočtený point cloud převrátíme transformační maticí (30), jinak by byl vzhůru nohama.

$$\text{tranformační matice} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{34}$$

Protože point cloud představuje obsáhlé množství bodů, je zapotřebí pro urychlení výpočetních procesů komprese dat, kterou provedu voxelizačním převzorkováním, který převede body point cloudu na voxely zadané velikosti a všechny body uvnitř voxelu zprůměruje a vytvoří z nich jeden zprůměrovaný bod. Po kompresi dat vypočte program normálové vektory všech bodů kovarianční analýzou, kde zadávám dva argumenty pro výpočet a to rádius, ve kterém bude hledat a maximální počet sousedních bodů, které může použít pro rychlejší výpočetní čas. Point cloud se po této úpravě uloží jako “point-cloud1”.



(a) **(b)**

Obrázek 4 –(a) spojený RGB obraz s hloubkovými daty do hloubkové mapy před kompresí, (b) po kompresi hloubkové mapy

Fáze druhá, po vytvoření prvotního point cloudu přejde program do smyčky, ve které bude cyklit, dokud ho já nebo jeden bezpečnostních prvků neukončí. Na začátku každé smyčky načtou data RGB a hloubkové mapy. Následně se vytvoří point cloud stejným způsobem jako při prvotní tvorbě point cloudu, jen s jediným rozdílem, a to že se uloží s názvem "point-cloud2", který bude po každém cyklu přepsán na nový.

Teď mám vytvořené dva point cloudy, které chci spojit. Pro spojení point cloudů potřebuji znát polohu a úhel natočení snímací kamery, tedy vnější matici kamery (4). Ta byla u prvních dat point cloudu nastavena na střed světového souřadného systému a následující bude posun a změna natočení od tohoto středu. Zde mám více možností a já jsem se konkrétně rozhodoval mezi získávání dat z akcelerometru Kinectu nebo použití vizuální výpočetní metody. První rozhodnutí bylo pro zpracovávání dat z akcelerometru.

Kinect má v sobě zabudovaný akcelerometr Kionix KXSD9. Surové data jsou ve tvaru mg/LSB ($mili - G$ na bit v šestnáctkové soustavě). Musím je tedy upravit na srozumitelnější akcelerační tvar tíhy G. Šestnáctková soustava je rovna rozsahu 65535 bitů a rozsah zrychlení je rovna $\pm 8g$. To znamená:

$$\frac{16 \cdot 10^3}{65535} = 0{,}244 \tag{35}$$

Při vynásobení surových dat touto hodnotou, získáme gravitační zrychlení akcelerometru, tedy translační pohyby os x, y, z. To mi ale nestačí, já potřebuji pro výpočet změny polohy znát celou vnější matici kamery (4). Dále tedy potřebujeme rotační matici (32), kterou zjistím následujícím způsobem.

$$\boldsymbol{R} = \begin{bmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{bmatrix} \tag{36}$$

Data získané z akcelerometru si označím x_s, y_s, z_s, jako surová data a budu je reprezentovat ve tvaru $\boldsymbol{g} = (x_s, y_s, z_s)$. Data jsou kladné, ale já chci, aby tíhové zrychlení bylo reprezentováno pro osu $\boldsymbol{z} = -\boldsymbol{g} = -(x_s, y_s, z_s)$. Následný vzorec bude ve tvaru:

$$\boldsymbol{R_z} = \left(r_{3,1}, r_{3,2}, r_{3,3}\right) = \frac{\boldsymbol{z}}{\|\boldsymbol{z}\|} \tag{37}$$

Osa $\boldsymbol{y}$ bude mít data $\boldsymbol{y} = (0, -z_s, y_s)$ a vzorec bude ve tvaru:

$$\boldsymbol{R_y} = \left(r_{2,1}, r_{2,2}, r_{2,3}\right) = \frac{\boldsymbol{y}}{\|\boldsymbol{y}\|} \tag{38}$$

Osa $\boldsymbol{x}$ bude mít data vektorového součinu $\boldsymbol{R_y}$ a $\boldsymbol{R_z}$:

$$\boldsymbol{R_x} = \left(r_{1,1}, r_{1,2}, r_{1,3}\right) = \boldsymbol{R_y} \times \boldsymbol{R_z} \tag{39}$$

Získal jsem rotační matici (32), kterou dosadím do vnější matice kamery (4) a tím pádem ji máme kompletní.

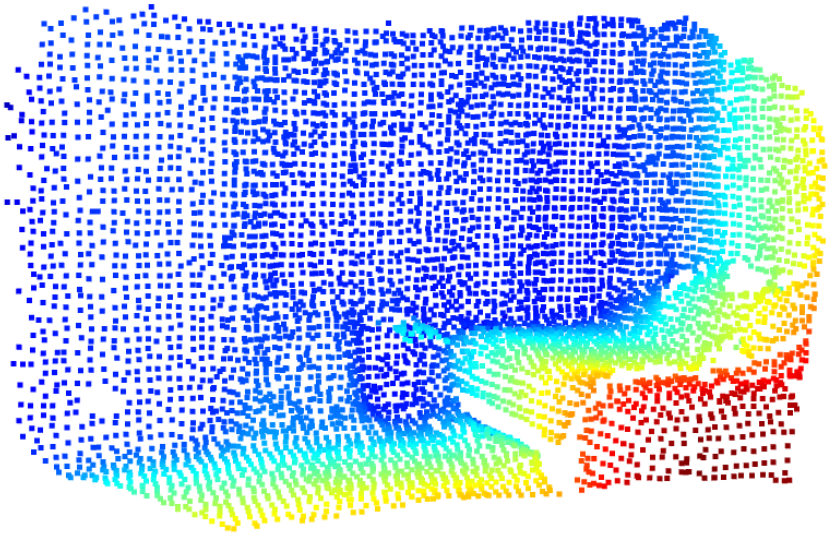
Dále bylo zamýšlena paralelizace snímání akcelerometru a hloubkové mapy, pro co nejpřesnější spojení. Problém nastal hned na začátku paralelizace. Obě knihovny pykinect a knihovna pro získávání dat z akcelerometru pyusb na počátku zapisují konfiguraci USB portu, což vyhodí chybové hlášení, protože může být nastavená pouze jedna konfigurace na jednom USB portu. Pro platformu Ubuntu je možné toto obejít nastavením pravidla konfigurace a čtení z USB portů, bohužel platforma Windows tuto možnost neumožňuje. Tím pádem, jsem byl nucen přejít na druhou možnost získání vnější matice kamery a to

výpočtem z vizuálních změn jednotlivě získaných dat. Ty se mohou například získat ze změn pořízených snímků nebo z vytvořených point cloudů. Já si vybral pro svou práci výpočet ze změn polohy point cloudů.

Nejdříve si načtu nový a předchozí point cloud. Předchozí bude buď brán jako prvotní nebo v dalších cyklech už jako spojené přechozí point cloudy. Protože budeme předpokládat chod programu v reálném čase, s co největším možnou snímkovou frekvencí, nebude mezi novým a předchozím point cloudem velký rozdíl polohy, proto nebude potřebovat velké množství bodů pro výpočet a pomůže nám to k urychlení zpracování. Před začátkem výpočtu tedy použijeme kompresy dat voxelizací. Pro výpočet spojení použiji knihovnu Open3D, která pro spojování používá metodu optimalizace grafu pozice a metodu ICP bod-na-plochu.

Nastavím si hrubou a jemnou maximální korespondenční hodnotu a vytvořím si nulovou matici 4 × 4, do které bude uložena vypočtená vnější matice kamery. Mnou nazvaný “point-cloud2“ bude zdrojový a point-cloud1 bude cílový, do kterého se snažím zdrojový připojit. S těmito známými veličinami nyní vypočtu pomocí ICP bod-na-plochu (23) s hrubou korespondenční hodnotou a výsledek se zapíše do mé nulové matice 4 × 4. Tento výpočet provedu ještě jednou s jemnou korespondenční hodnotou a nově známou transformační matici $\boldsymbol{T}_i$ z předchozího výpočtu. První výpočet tedy provedl hrubý výpočet a druhý na základě prvního zpřesnil transformační matici $\boldsymbol{T}_i$. S tou si nyní ještě vypočtu informační matici Λ_i. Párová registrace je náchylná k chybám. Špatné párové zarovnání může převyšovat počet správně zarovnaných párů. Hrany grafu se rozdělují do dvou tříd. Na hrany odometrie spojují dočasně blízké sousední uzly. Algoritmus lokální registrace, jako je ICP, je může spolehlivě sladit. A na smyčky uzavírající hrany spojují všechny nesousedící uzly. Proto teď pro vypočet konečné pozice $\{\boldsymbol{T}_i\}$ použiji optimalizaci grafu pozice (25). Vytvoří graf pozice se třemi uzly a třemi hranami. Z nichž jsou dvě hrany odometrie a jedna hrana uzavření smyčky. Na zdrojový point cloud použiji transformační matici, která je ve tvaru vnější matice kamery. Nyní mohu pohodlně sloučit dva point cloudy do jednoho. Ještě než tuto fázi ukončím, znovu použiji na nově sloučený jednotný point cloud kompresi dat voxelizací. Toto je podstatná část po sloučení point cloudů, protože to uleví duplikovaným

nebo přehuštěným bodům. Na konec uložím sloučený point cloud jako hlavní s názvem “point-cloud1”.

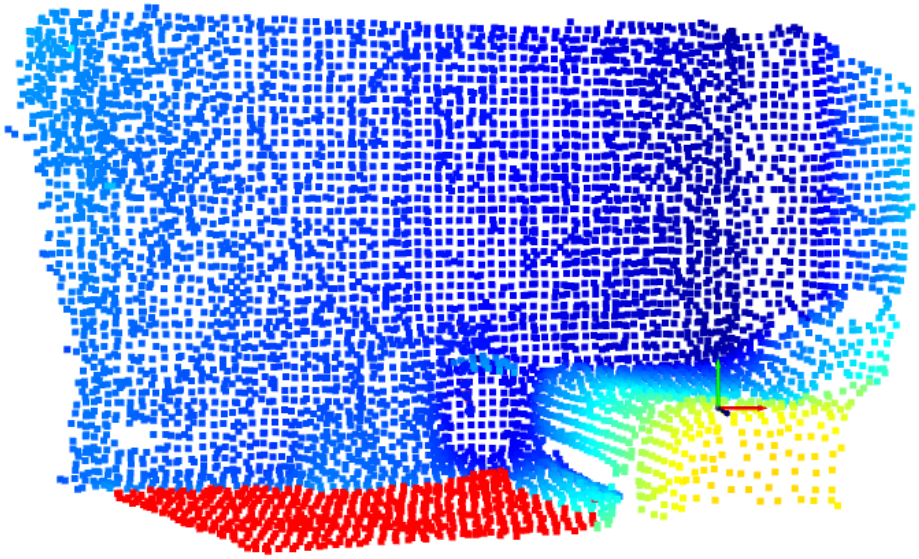


Obrázek 5 – Po spojení více point cloudů

Než přejdu do fáze detekce podlahy je po spojení point cloudů zapotřebí odfiltrovat nepotřebné informace a zpřesnit roviny. Nejdříve si načtu data ze zkombinovaného point cloudu. Vytvořím si KD-strom a v kombinaci s MLS (metodou klouzavých nejmenších čtverců), obou popsaných dříve, se zadaným rádiem zpřesním celý datový set.

V této části, už jsem schopen v rámci možností detekovat plochy, které potřebuji. Protože tato bakalářská práce je se zaměřením na pozemní roboty či jiné mobilní zařízení, které se bude pohybovat po zemi, je pro mě nejpodstatnější detekovat zem a ostatní následně mohu brát jako překážku. Nejdříve vezmu point cloud a použiji na něj pásmový filtr a to pásmovou propust. Je zapotřebí nastavit osu, na kterou se nastaví pásmo filtru a to osu kolmou k zemskému povrchu. Vzhledem k tomu, že senzory nemohou být umístěny níž, než je zem, nastavím pásmo na vzdálenost od senzoru a níže. Tím si zajistím, že point cloud bude obsahovat převážně zem. Teď může předpokládat, že největší plochou bude zem, můžeme tedy použít pro detekci plochy metodu RANSAC, která jak jsem psal dříve, vyhledá největší plochu podle zadaných hodnot, počtu iterací, počtu sousedních bodů a prahové velikosti. Tuto plochu země si uložím jako nový point cloud “ground” a nastavím ji červenou barvu pro následnou vizualizaci.

Vizualizace slouží pouze pro uživatele k názorné ukázce, co se zaznamenává. Před začátkem se z celkového point cloudu odeberou data, kde byla detekována plocha a je nahrazena mojí novou (červenou) plochou země. Otevře se vizualizační okno, pokud už je otevřené, pouze se aktualizují data. Nastaví se geometrie point cloudu, detekované plochy a x, y, z osy reprezentující polohu snímacího zařízení. Po této části se program vrátí na začátek cyklu.



Obrázek 6 – Spojené point cloudy s detekovanou plochou země označenou červenou barvou

Tím bychom měli popsanou strukturu a postup celého programu.

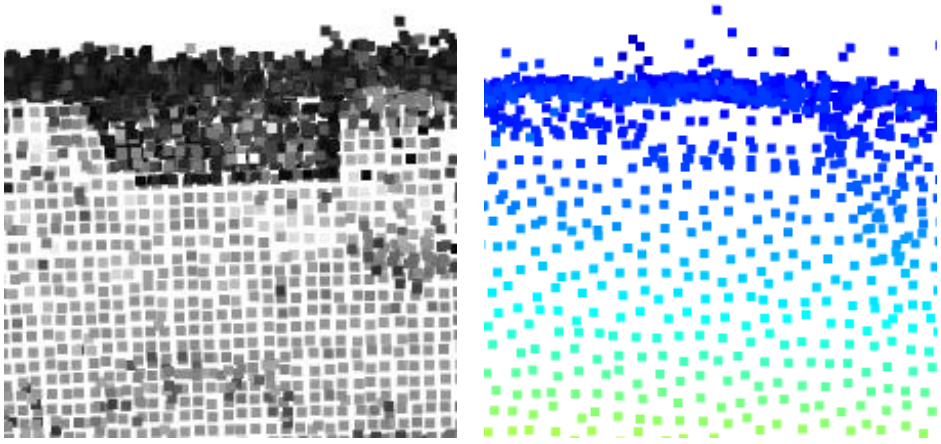
4.3 RealSense D455

V závěru této bakalářské práce se mi naskytla možnost zapůjčit si od školy RealSense D455, který se dá nazvat slovem modernější. Dle Tab. 1, která obsahuje porovnání obou senzorů, je RealSense a jimž obsažené senzory přesnější, rozměry a váha jsou rapidně nižší oproti Kinectu a velikou výhodou je odstranění externího napájení, stačí napájení přímo z USB portu. Rozhodl jsem se otestovat můj program i s touto kamerou. Pro možnost chodu programu, bylo zapotřebí malých změn. Bylo zapotřebí použití náležité knihovny dané výrobcem a k němu přidělené názvy příkazů, pro získání vstupních dat. Tato knihovna je obsáhlejší a umožňuje získávání dat akcelerometru a gyroskopu zároveň při získávání vstupních dat hloubkové mapy.

Tabulka. 1 – Porovnání specifikací Kinectu a RealSensu D455

	Kinect V1	RealSense D455
RGB	640 × 480 pixelů, 30 Hz	1280 × 720 pixelů, 30 Hz
Hloubková mapa	640 × 480 pixelů, 30 Hz	1280 × 720 pixelů, 90 Hz
Zorný úhel senzoru hloubkové mapy	57° horizontálně a 43° vertikálně	87° horizontálně a 57° vertikálně
Ideální vzdálenost	~ 0,5 m až 4 m	~ 0,6 m až 6 m
Přesnost hloubkové mapy	-	<2% na 4 m
Rozměry	279 × 63 × 38mm	124 × 26 × 29mm
Váha	1360 g	390 g

Ještě se vrátím k objasnění, proč jsem nazval RealSense slovem přesnější. Bylo to myšleno ve smyslu preciznosti získání jednotlivých bodů při větších vzdálenostech. Na obr. 7 můžeme názorně vidět porovnání dat získaných z Kinectu obr. 7(a) a z RealSensu obr. 7(b). Obrázky jsou vizualizovány v půdorysu. V horní části je zde vidět stěna reprezentována největší hustotou bodů. Oba senzory byly umístěny 3 metry od stěny. Když tyto dva obrázky porovnáme vedle sebe, jde vidět, že RealSense obr. 7(b) má body reprezentující stěnu blíže u sebe a představuje tak více rovinu. Pro tento důvod a důvody popsané dříve, je RealSense D455 ideální kandidát na možné pokračování této práce, která by pracovala i těmito daty.



(a) **(b)**

Obrázek 7 – Obsahuje point cloudy v půdorysu, kde největší hustota bodů representuje stěnu. Znázorňuji tím přesnost senzorů. Na levém obrázku (a) je použit senzor Kinect, na pravém obrázku (b) je použit senzor RealSense.

5 Závěr

V této bakalářské práci jsem se zabýval návrhem programu, který měl vytvářet 3D mapu okolního prostoru na základě získávaných informací ze senzorů a rozpoznal podlahu, po které by se následně pozemní mobilní zařízení mohlo pohybovat.

Provedl jsem rešerši aktuálně dostupných senzorů pro snímání hloubkových map a jejich principy snímání. Ze všech možností a dostupných senzorů jsem se rozhodoval mezi Kinectem a RealSensem. Byl vybrán senzor Kinect V1, který pro dosažení cíle je zcela dostačující a v době pandemie nejlépe dostupný. Ke konci práce se mi naskytla možnost otestování mého programu i na senzoru RealSense D455, se kterým jsem udělal porovnání vlastností oproti Kinectu v poslední kapitole 4.3.

Provedl jsem rešerši metod zpracování hloubkových map a vhodných metod k detekci volného prostoru před autonomním robotickým podvozkem. Tato rešerše obsahuje možnosti předzpracování, zpracování a následného zpracování. Ze zvažovaných možností jsem vybral metody 3D point cloud, voxelizace, odhad vektoru normály, k – dimenzionální strom, MLS (metoda klouzavých nejmenších čtverců), ICP (Iterativní nejbližší bod), pásmovou propust a na detekci ploch metodu RANSAC.

Navrhnul jsem program v jazyce Python implementující vybrané metody pro zvolený senzor. Navržený program jsem realizoval a úspěšně otestoval v kapitole 4. Všechny metody byly úspěšně implementované a určený cíl byl splněn.

Když budu sebekritický, program funguje správně a je zcela dostačující pro požadovaný cíl. Jsou zde stále slabiny, které by se daly v následné navazující práci zlepšit. Spojování point cloudů probíhá výpočetní metodou. Metoda je výpočetně náročná a brzdí plynulost chodu programu. Kdyby pohybující se rychlost robotického podvozku byla vysoká a způsobila by velký rozdíl poloh jednotlivých point cloudů, výpočetní metoda by mohla nesprávně spojit tyto point cloudy. S tím přichází možné řešení použití metody získání změny polohy pomocí akcelerometru a gyroskopu, která by tuto problematiku mohla vyřešit a umožnila program více se přiblížit k chodu v reálném čase. Pro možné pokračování tedy navrhuji upravení

použitých metod pro spojování point cloudů, následné implementace a otestování senzoru na robotickém podvozku.

Dle mého názoru Kinect i RealSense jsou vhodné senzory pro použití na robotickém podvozku. Senzory mnou použité mají minimální vzdálenost snímání okolo 0,5 m, proto bych také pro bezpečnost přidal i jiné senzory, například ultrazvukové pro získávání informací o okolních překážkách bližších než 0,5 m a tudíž preciznější manipulaci v prostoru.

6 Použitá literatura

[1] N. Mohammadiha, M. Sahraeian, B. V. Vahdat, A. Azizi, and A. S. Ahmadi, "Measuring the Geometrical Parameters of Steel Billets during Molding Process Using Image Processing," in *2006 IEEE International Symposium on Signal Processing and Information Technology*, Aug. 2006, pp. 59–63, doi: 10.1109/ISSPIT.2006.270770.

[2] T. Jiang, *Stereo Vision for Facet Type Cameras*. Logos Verlag Berlin GmbH, 2016.

[3] E. DANDIL and K. K. ÇEVİK, "Computer Vision Based Distance Measurement System using Stereo Camera View," in *2019 3rd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Oct. 2019, pp. 1–4, doi: 10.1109/ISMSIT.2019.8932817.

[4] S. Miriam, *Omnidirectional Stereo Vision for Autonomous Vehicles*. KIT Scientific Publishing, 2015.

[5] C. Shi, J. Feng, S. Tang, and Z. Song, "A Robust Feature Detection Method for an Infrared Single-Shot Structured Light System," in *2018 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, Aug. 2018, pp. 693–697, doi: 10.1109/RCAR.2018.8621672.

[6] N. M. DiFilippo and M. K. Jouaneh, "Characterization of Different Microsoft Kinect Sensor Models," *IEEE Sens. J.*, vol. 15, no. 8, pp. 4554–4564, Aug. 2015, doi: 10.1109/JSEN.2015.2422611.

[7] M. Samir, E. Golkar, and A. A. A. Rahni, "Comparison between the Kinect™ V1 and Kinect™ V2 for respiratory motion tracking," in *2015 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*, Oct. 2015, pp. 150–155, doi: 10.1109/ICSIPA.2015.7412180.

[8] "Intel-RealSense-D400-Series-Datasheet-June-2020.pdf." Accessed: Dec. 03, 2020. [Online]. Available: https://www.intelrealsense.com/wp-content/uploads/2020/06/Intel-RealSense-D400-Series-Datasheet-June-2020.pdf.

[9] "Introducing the Intel® RealSense™ Depth Camera D455," *Intel® RealSense™ Depth and Tracking Cameras*. https://www.intelrealsense.com/depth-camera-d455/ (accessed Apr. 16, 2021).

[10] F. Wermke and B. Meffert, "Interference Model of Two Time-Of-Flight Cameras," in *2019 IEEE SENSORS*, Oct. 2019, pp. 1–4, doi: 10.1109/SENSORS43011.2019.8956892.

[11] M. Kurc, K. Wegner, and M. Domański, "Transformation of depth maps produced by ToF cameras," in *2014 International Conference on Signals and Electronic Systems (ICSES)*, Sep. 2014, pp. 1–4, doi: 10.1109/ICSES.2014.6948713.

[12] Y. S. Huang, Y. P. Huang, M. S. Young, and Ke-Nung Huang, "The Assessment System of Human Visual Spectral Sensitivity Curve by Frequency Modulated Light," in *2005 IEEE Engineering in Medicine and Biology 27th Annual Conference*, Jan. 2005, pp. 263–265, doi: 10.1109/IEMBS.2005.1616394.

[13] Amad-ud-Din, I. A. Halin, and S. B. Shafie, "A review on Solid State time of flight TOF range image sensors," in *2009 IEEE Student Conference on Research and Development (SCOReD)*, Nov. 2009, pp. 246–249, doi: 10.1109/SCORED.2009.5443066.

[14] S. Hong, C. Ye, M. Bruch, and R. Halterman, "Performance evaluation of a Pose Estimation method based on the SwissRanger SR4000," in *2012 IEEE International Conference on Mechatronics and Automation*, Aug. 2012, pp. 499–504, doi: 10.1109/ICMA.2012.6283123.

[15] J. P. Queralta *et al.*, "FPGA-based Architecture for a Low-Cost 3D Lidar Design and Implementation from Multiple Rotating 2D Lidars with ROS," in *2019 IEEE SENSORS*, Oct. 2019, pp. 1–4, doi: 10.1109/SENSORS43011.2019.8956928.

[16] R. Torun, M. M. Bayer, I. U. Zaman, J. E. Velazco, and O. Boyraz, "Realization of Multitone Continuous Wave Lidar," *IEEE Photonics J.*, vol. 11, no. 4, pp. 1–10, Aug. 2019, doi: 10.1109/JPHOT.2019.2922690.

[17] E. D. B. Solis, A. M. Neto, and B. N. Huallpa, "Pearson's Correlation Coefficient for Discarding Redundant Information: Velodyne Lidar Data Analysis," in *2015 12th Latin American Robotics Symposium and 2015 3rd Brazilian Symposium on Robotics (LARS-SBR)*, Oct. 2015, pp. 116–119, doi: 10.1109/LARS-SBR.2015.34.

[18] M. Velas, M. Spanel, M. Hradis, and A. Herout, "CNN for very fast ground segmentation in velodyne LiDAR data," in *2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, Apr. 2018, pp. 97–103, doi: 10.1109/ICARSC.2018.8374167.

[19] Xin Wang, "Moving window-based double haar wavelet transform for image processing," *IEEE Trans. Image Process.*, vol. 15, no. 9, pp. 2771–2779, Sep. 2006, doi: 10.1109/TIP.2006.877316.

[20] L. M. G. Fonseca, L. M. Namikawa, and E. F. Castejon, "Digital Image Processing in Remote Sensing," in *2009 Tutorials of the XXII Brazilian Symposium on Computer Graphics and Image Processing*, Oct. 2009, pp. 59–71, doi: 10.1109/SIBGRAPI-Tutorials.2009.13.

[21] L. Gao, P. Chen, and S. Yu, "Demonstration of Convolution Kernel Operation on Resistive Cross-Point Array," *IEEE Electron Device Lett.*, vol. 37, no. 7, pp. 870–873, Jul. 2016, doi: 10.1109/LED.2016.2573140.

[22] Z. Wang, H. Zhao, H. Xu, W. Liu, K. Liu, and J. Zhao, "Helicopter Target Recognition Based on the Frequency Domain Adaptive Convolution Kernel Filtering," in *2018 China International SAR Symposium (CISS)*, Oct. 2018, pp. 1–6, doi: 10.1109/SARS.2018.8552010.

[23] G. George, R. M. Oommen, S. Shelly, S. S. Philipose, and A. M. Varghese, "A Survey on Various Median Filtering Techniques For Removal of Impulse Noise From Digital Image," in *2018 Conference on Emerging Devices and Smart Systems (ICEDSS)*, Mar. 2018, pp. 235–238, doi: 10.1109/ICEDSS.2018.8544273.

[24] B. Sravani and M. V. N. Rao, "Removing of high density salt and pepper noise using fuzzy median filter," in *2014 International Conference on High Performance Computing and Applications (ICHPCA)*, Dec. 2014, pp. 1–6, doi: 10.1109/ICHPCA.2014.7045370.

[25] D. Tian, H. Ochimizu, C. Feng, R. Cohen, and A. Vetro, "Geometric distortion metrics for point cloud compression," in *2017 IEEE International Conference on Image Processing (ICIP)*, Sep. 2017, pp. 3460–3464, doi: 10.1109/ICIP.2017.8296925.

[26] M. Lu, J. Zhao, Y. Guo, and Y. Ma, "Accelerated Coherent Point Drift for Automatic Three-Dimensional Point Cloud Registration," *IEEE Geosci. Remote Sens. Lett.*, vol. 13, no. 2, pp. 162–166, Feb. 2016, doi: 10.1109/LGRS.2015.2504268.

[27] "Geometry of Image Formation | Learn OpenCV," Feb. 20, 2020. https://learnopencv.com/geometry-of-image-formation/ (accessed Apr. 08, 2021).

[28] J. Garstka and G. Peters, "Fast and Robust Keypoint Detection in Unstructured 3-D Point Clouds," Jul. 2015, vol. 2, doi: 10.5220/0005569501310140.

[29] F. Garcia and B. Ottersten, “CPU-Based Real-Time Surface and Solid Voxelization for Incomplete Point Cloud,” in *2014 22nd International Conference on Pattern Recognition*, Aug. 2014, pp. 2757–2762, doi: 10.1109/ICPR.2014.475.

[30] J. Xu, J. Xu, and X. Yu, “Normal Vector of the 3D Point Cloud Estimates and Close to the Point Normal Vector Adjustment Methods,” in *2012 Second International Conference on Instrumentation, Measurement, Computer, Communication and Control*, Dec. 2012, pp. 610–613, doi: 10.1109/IMCCC.2012.150.

[31] W. Hou, D. Li, C. Xu, H. Zhang, and T. Li, “An Advanced k Nearest Neighbor Classification Algorithm Based on KD-tree,” in *2018 IEEE International Conference of Safety Produce Informatization (IICSPI)*, Dec. 2018, pp. 902–905, doi: 10.1109/IICSPI.2018.8690508.

[32] P. Chen and Y. Wang, “Optimized KD Tree Application in Instance-Based Learning,” in *2008 Fifth International Conference on Fuzzy Systems and Knowledge Discovery*, Oct. 2008, vol. 2, pp. 187–191, doi: 10.1109/FSKD.2008.41.

[33] Y. Fujita, S. Ikuno, T. Itoh, and H. Nakamura, “Modified Improved Interpolating Moving Least Squares Method for Meshless Approaches,” *IEEE Trans. Magn.*, vol. 55, no. 6, pp. 1–4, Jun. 2019, doi: 10.1109/TMAG.2019.2900374.

[34] Hui Ni, Zhong Li, and Hongxing Song, “Moving least square curve and surface fitting with interpolation conditions,” in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, Oct. 2010, vol. 13, pp. V13-300-V13-304, doi: 10.1109/ICCASM.2010.5622770.

[35] J. Park, Q.-Y. Zhou, and V. Koltun, “Colored Point Cloud Registration Revisited,” in *2017 IEEE International Conference on Computer Vision (ICCV)*, Venice, Oct. 2017, pp. 143–152, doi: 10.1109/ICCV.2017.25.

[36] P. Besl and N. McKay, “A Method for Registration of 3-D Shapes,” *IEEE Trans Pattern Anal Mach Intell*, 1992, doi: 10.1109/34.121791.

[37] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” *Image Vis Comput*, 1992, doi: 10.1016/0262-8856(92)90066-C.

[38] “Multiway registration — Open3D 0.12.0 documentation.” http://www.open3d.org/docs/release/tutorial/pipelines/multiway_registration.html (accessed Apr. 17, 2021).

[39] S. Choi, Q.-Y. Zhou, and V. Koltun, “Robust Reconstruction of Indoor Scenes,” Jun. 2015, doi: 10.1109/CVPR.2015.7299195.

[40] M. Miknis, R. Davies, P. Plassmann, and A. Ware, “Near real-time point cloud processing using the PCL,” in *2015 International Conference on Systems, Signals and Image Processing (IWSSIP)*, Sep. 2015, pp. 153–156, doi: 10.1109/IWSSIP.2015.7314200.

[41] M. Rahman, X. Li, and X. Yin, “DL-RANSAC: An Improved RANSAC with Modified Sampling Strategy Based on the Likelihood,” in *2019 IEEE 4th International Conference on Image, Vision and Computing (ICIVC)*, Jul. 2019, pp. 463–468, doi: 10.1109/ICIVC47709.2019.8981025.

[42] P. Trivedi, T. Agarwal, and K. Muthunagai, “MC-RANSAC: A Pre-processing Model for RANSAC using Monte Carlo method implemented on a GPU,” in *2013 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, Aug. 2013, pp. 1380–1383, doi: 10.1109/ICACCI.2013.6637380.

[43] Y. Li, J. Zhang, P. Gao, L. Jiang, and M. Chen, “Grab Cut Image Segmentation Based on Image Region,” in *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, Jun. 2018, pp. 311–315, doi: 10.1109/ICIVC.2018.8492818.

[44] S. Suwanmanee, S. Chatpun, and P. Cabrales, “Comparison of video image edge detection operators on red blood cells in microvasculature,” in *The 6th 2013 Biomedical Engineering International Conference*, Oct. 2013, pp. 1–4, doi: 10.1109/BMEiCon.2013.6687686.

[45] D. Xiaoheng, L. Minghang, M. Jiashu, and W. Zhengyu, “Edge Detection Operator for Underwater Target Image,” in *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, Jun. 2018, pp. 91–95, doi: 10.1109/ICIVC.2018.8492749.

[46] Chen Yuchao and Liu Hui, “Study on the meso-structure image of shale based on the digital image processing technique,” in *2009 International Conference on Image Analysis and Signal Processing*, Apr. 2009, pp. 150–153, doi: 10.1109/IASP.2009.5054656.

[47] M. B. Salem, K. S. Ettabaa, and M. S. Bouhlel, “Hyperspectral image feature selection for the fuzzy c-means spatial and spectral clustering,” in *2016 International Image Processing, Applications and Systems (IPAS)*, Nov. 2016, pp. 1–5, doi: 10.1109/IPAS.2016.7880114.

[48] V. Uslan and İ. Ö. Bucak, “Clustering-based spot segmentation of cDNA microarray images,” in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, Aug. 2010, pp. 1828–1831, doi: 10.1109/IEMBS.2010.5626430.

[49] X. Gao and H. Xiong, “A hybrid wavelet convolution network with sparse-coding for image super-resolution,” in *2016 IEEE International Conference on Image Processing (ICIP)*, Sep. 2016, pp. 1439–1443, doi: 10.1109/ICIP.2016.7532596.

[50] S. Y. Nikouei, Y. Chen, S. Song, R. Xu, B. Choi, and T. R. Faughnan, “Real-Time Human Detection as an Edge Service Enabled by a Lightweight CNN,” in *2018 IEEE International Conference on Edge Computing (EDGE)*, Jul. 2018, pp. 125–129, doi: 10.1109/EDGE.2018.00025.

[51] K. O’Shea and R. Nash, “An Introduction to Convolutional Neural Networks,” *ArXiv151108458 Cs*, Dec. 2015, Accessed: Apr. 21, 2021. [Online]. Available: http://arxiv.org/abs/1511.08458.

[52] “SUN RGB-D: A RGB-D Scene Understanding Benchmark Suite.” http://rgbd.cs.princeton.edu/ (accessed Apr. 08, 2021).