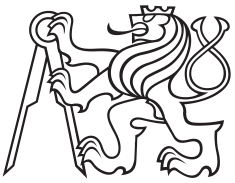
Bakalářská práce



České
vysoké
učení technické
v Praze

F3

Fakulta elektrotechnická
Katedra kybernetiky

Detekce objektů z hloubkové kamery

Lukáš Kunt

Vedoucí: RNDr. Petr Štěpán, Ph.D.
Studijní program: Kybernetika a robotika
Květen 2020

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Kunt** Jméno: **Lukáš** Osobní číslo: **478073**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Kybernetika a robotika**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Detekce objektů z hloubkové kamery

Název bakalářské práce anglicky:

Object Detection from Depth Camera

Pokyny pro vypracování:

1) Seznamte se s metodami zpracování dat z hloubkových kamer.
2) Analyzujte data z kamery Realsense D435 a navrhněte algoritmus pro detekci zdi sestavené z kvádrů předem daných velikostí. Výstupem algoritmu by měl být seznam relativních 3D poloh kvádrů.
3) Otestujte algoritmus na datech z dronu i na datech pořízených z kamery nesené člověkem (bez informace o poloze kamery ze senzorů IMU).

Seznam doporučené literatury:

[1] Holz D., Holzer S., Rusu R.B., Behnke S. (2012) Real-Time Plane Segmentation Using RGB-D Cameras. In: Röfer T., Mayer N.M., Savage J., Saranlı U. (eds) RoboCup 2011: Robot Soccer World Cup XV. RoboCup 2011. Lecture Notes in Computer Science, vol 7416. Springer, Berlin, Heidelberg.
[2] Dou M., Guan L., Frahm JM., Fuchs H. (2013) Exploring High-Level Plane Primitives for Indoor 3D Reconstruction with a Hand-held RGB-D Camera. In: Park JI., Kim J. (eds) Computer Vision - ACCV 2012 Workshops. ACCV 2012. Lecture Notes in Computer Science, vol 7729. Springer, Berlin, Heidelberg.
[3] Ma, L., Kerl, C., Stückler, J., & Cremers, D. (2016, May). CPA-SLAM: Consistent plane-model alignment for direct RGB-D SLAM. In 2016 IEEE International Conference on Robotics and Automation (ICRA) (pp. 1285-1291). IEEE.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

RNDr. Petr Štěpán, Ph.D., Multirobotické systémy FEL

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

Datum zadání bakalářské práce: **10.01.2020** Termín odevzdání bakalářské práce: _____

Platnost zadání bakalářské práce: **30.09.2021**

RNDr. Petr Štěpán, Ph.D. podpis vedoucí(ho) práce	doc. Ing. Tomáš Svoboda, Ph.D. podpis vedoucí(ho) ústavu/katedry	prof. Mgr. Petr Páta, Ph.D. 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

Poděkování

Tímto bych chtěl poděkovat panu RNDr. Petru Štěpánovi, Ph.D. za vedení práce. Zároveň bych chtěl poděkovat své rodině a přítelkyni za podporu při studiu a tvorbě práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.
V Praze dne 22. května 2020

.....
podpis autora práce

Abstrakt

Tato práce se zabývá detekcí objektů z hloubkové složky RGB-D dat, jejichž zdrojem je stereokamera Intel® Realsense^{TM}. Nejprve je provedena rešerše dostupných metod pro hledání normálového vektoru plochy v mračnech bodů, segmentaci obrazu a detekci objektů. Posléze je těchto znalostí využito k navržení vlastních algoritmů na detekci normálového vektoru plochy a následnému přesnému určení polohy hledaných objektů. Tyto programy vycházejí zejména z RANSAC algoritmu a PCA, dále jsou kombinovány s naši apriorní znalostí tvaru objektů. Navržené programy jsou následně vyhodnoceny na manuálně označených datech, čímž je ověřeno, že jsou schopny přesné detekce v reálném čase.

Klíčová slova: detekce objektů, stereokamera, konvexní obal, mračno bodů, RGB-D, RANSAC, PCA

Vedoucí: RNDr. Petr Štěpán, Ph.D.
KN:E-116
Karlovo náměstí 13
12135 Praha 2

Abstract

This work deals with object detection from the depth part of RGB-D data, which are produced by stereo camera Intel® Realsense^{TM}. Firstly, research of available methods for plane normal estimation from point clouds, picture segmentation and object detection are conducted. Based on this knowledge, a program for detection of the surface normal and then the precise location of on surface laying objects is proposed. Our approach is mainly based on RANSAC algorithm and PCA, which are combined with our a prior knowledge of the object's shapes. Proposed programs are finally evaluated on a manually labelled dataset. The evaluation proves that presented algorithms are capable of high accuracy object detection in real-time.

Keywords: object detection, stereo camera, convex hull, point cloud, RGB-D, RANSAC, PCA

Title translation: Object Detection from Depth camera

Obsah

Obrázky

Tabulky

Seznam použitých zkratek

PCA	Principal Component Analysis
IMU	Intertial measurement unit
RGB-D	Red, green, blue - depth
RANSAC	Random sample consensus
ICP	Iterative Closest Point
CCL	Connected-component labeling
DBSCAN	Density-based spatial clustering of applications with noise
MBZIRC	Mohamed Bin Zayed International Robotics Challenge
LIDAR	Light Detection and Ranging
HOG	Histogram of Oriented Gradients
CAD	Computer Aided Design
SURF	Speeded up robust features
SIFT	Scale-invariant feature transform
BRIEF	Binary Robust Independent Elementary Features
PCL	Point Cloud Library
ORB	Oriented FAST and rotated BRIEF
SVM	Support vector machines

ctuthesis t1606152353

Kapitola 1

Úvod

V poslední době došlo k prudkému nárůstu počtu bezpilotních letounů. Nejpoužívanějšími z nich jsou kvadrokoptéry, které se často označují jako drony.[1] Drony se často využívají na pořizování videí a fotografií, za účelem strategického průzkumu, mapování, trasnportu a podobně. S postupným rozšiřováním bezpilotních letounů přibývá i množství těch autonomních. Ty jsou schopny vnímat okolní prostředí pomocí různých druhů senzorů a na základě podnětů upravovat chování za účelem splnění předem zadaného úkolu.

Základními zdroji informací pro řízení dronu bývají různé verze Inerciální měřící jednotky (IMU), což je typ zařízení, který měří úhlovou rychlost, zrychlení a někdy i polohu bezpilotního letounu. K těmto úkonům využívají IMU senzory, jako je například akcelerometr, gyroskop a magnetometr. Dalším senzorem, který poskytuje informaci o okolí dronu, je kamera. Její použití je však vzhledem k absenci informace o vzdálenosti jednotlivých bodů omezené a musí být často kombinována s dalšími senzory jako je například Light Detection And Ranging (LIDAR). To jest zařízení, které laserovým paprskem skenuje daný prostor a měří vzdálenost od objektů nacházejících se v daném prostoru.

Uceleným řešením pro získání informace o okolním prostředí je stereokamera, která poskytuje jak obraz okolí, tak i informaci o vzdálenosti jednotlivých bodů obrazu. Vzhledem k postupnému zvyšování přesnosti těchto kamer a nízké ceně oproti LIDARu se jich začíná stále více využívat pro získávání informací u samořiditelných strojů.

Motivací této práce byla soutěž Mohamed Bin Zayed International Robotics Challenge (MBZIRC), kde skupina dronů a autonomních pozemních vozidel spolupracuje na připravených výzvách. Jednou z těchto výzev byl sběr na zemi ležících cihel s předem známými rozměry a následné skládání těchto cihel do zdi. Cílem této práce je vytvoření programu, který bude schopen v reálném čase přesně detekovat pozice jednotlivých cihel a to pouze z hloubkových dat poskytovaných stereokamerou Intel® RealSense^{TM}. Program by měl být schopen detekce bez použití dalších senzorů, jako je například IMU, kterou je dron vybaven. Informace o poloze cihel je dále bezpilotními letouny zpracována a využita k uchopení jednotlivých cihel a jejich následnému umístění do zdi.

Tato práce je rozdělena na pět částí. V první z nich jsou shrnuty hlavní prvky matematického aparátu, které byly použity při zhotovení praktické

části práce a dále je popsán princip použitých algoritmů. Ve druhé části je představen základní model kamery. Tento model je později využit k popsání funkce stereokamery. Následně se zabýváme principem hledání stereo páru. Za tímto účelem jsou popsány základy epipolární geometrie a jsou představeny základní algoritmy, které se využívají pro hledání stereo páru. Na konci druhé části je představena stereokamera Intel® RealSense^{TM} D435, kterou jsme v naši práci využili. Ve třetí části práce je čtenář seznámen s používanými metodami detekce z hloubkových dat. Tato kapitola je členěna na metody pro detekci plochy, segmentaci obrazu a nalezení jednotlivých objektů z již segmentovaných dat. Následující část představuje naše postupy pro detekci cihel ze stereokamery. Jedná se především o kombinaci jednotlivých postupů představených ve čtvrté kapitole a jejich úpravu pro zlepšní výsledků našeho programu. Výsledky jsou následně prezentovány v páté části práce, kde je zároveň čtenář seznámen s použtou metodou vyhodnocování výsledků.

Kapitola 2
Matematický aparát

2.1 Homogenní souřadnice

Během celé práce budeme pracovat jak s kartézskými, tak i s homogenními souřadnicemi. Ty jsou v Euklidovském prostoru dimenze n reprezentovány pomocí vektoru, který má $n+1$ prvků. Je-li $\boldsymbol{r}$ bod v trojdimenzionálním Euklidovském prostoru $\mathbb{E}^3$ reprezentován parametry x, y, z, pak pro převod mezi kartézskými a homogenními souřadnicemi platí následující vztahy.

$$\mathbf{r} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \rightarrow \mathbf{r}_{hom} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.1}$$

$$\mathbf{r}_{hom} = \begin{bmatrix} x \\ y \\ z \\ w \end{bmatrix} \rightarrow \mathbf{r} = \begin{bmatrix} x/w \\ y/w \\ z/w \end{bmatrix} \tag{2.2}$$

Pomocích homogenních souřadnic můžeme sestavit transformační matici $\mathbf{A}$. Ta se skládá z matice rotace $\mathbf{R}$ a vektoru translace $\mathbf{t}$.

$$\mathbf{A} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.3}$$

2.2 Rotace a rotační matice

Natočení souřadného systému, popřípadě objektu v tomto souřadném systému, vzhledem k jinému souřadnému systému v prostoru dimenze n se nejjednodušeji popíše pomocí rotační matice $\mathbf{R} \in \mathbb{R}^n$. Mezi nejběžnější rotační matice patří rotace o úhel θ kolem os kartézského souřadného systému v $\mathbb{E}^3$. Tyto

matice vypadají následovně.

$$\mathbf{R_x} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \tag{2.4}$$

$$\mathbf{R_y} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix} \tag{2.5}$$

$$\mathbf{R_z} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.6}$$

Obecné natočení pak můžeme dostat složením třech a více těchto rotací[1]. Pro vytvoření matice rotace $\mathbf{R}$ v prostoru $\mathbb{R}^3$ kolem obecné osy dané normalizovaným vektorem $\mathbf{u}$ o úhel θ slouží Rodriguesův vzorec [2].

$$\mathbf{R} = \mathbf{I} + \tilde{\mathbf{u}}\sin\theta + \tilde{\mathbf{u}}^2(1-\cos\theta); \quad \tilde{\mathbf{u}} = \begin{bmatrix} 0 & -u_z & u_y \\ u_z & 0 & -u_x \\ -u_y & u_x & 0 \end{bmatrix} \tag{2.7}$$

Chceme-li natočit vektor $\mathbf{a}$ tak, aby byl rovnoběžný s vektorem $\mathbf{b}$, použijeme vzorec 2.7. Úhel rotace θ a vektor $\tilde{\mathbf{u}}$, kolem kterého rotace proběhne, se určí následovně.

$$\tilde{\mathbf{u}} = \frac{\mathbf{a}\times\mathbf{b}}{\|\mathbf{a}\times\mathbf{b}\|} \tag{2.8}$$

$$\theta = \arccos\left(\frac{\mathbf{a}\cdot\mathbf{b}}{\|\mathbf{a}\|\,\|\mathbf{b}\|}\right) \tag{2.9}$$

Podle Walkera [3] je tato metoda nevhodná pro výpočet na počítači. Zejména pak v případech, kdy bychom měli počícat arccos hodnot blízko ± 1. Toto je způsobeno omezenou přesností reprezentace desetinných čísel v počítači (takzvaná floating point aritmetika). Přesnější je tedy použít vzorec

$$\theta = \operatorname{atan2}\left(\|\mathbf{a}\times\mathbf{b}\|, \mathbf{a}\cdot\mathbf{b}\right). \tag{2.10}$$

Výhoda funkce atan2 oproti arcus tangens je ošetřené dělení nulou, ke kterému může dojít při zaokrouhlování desetinných čísel. Navíc výstupem funkce je úhel v rozmezí $[0, 2\pi)$.

2.3 Sobelův operátor pro detekci hran

Hrana v obrazu může být pozorována jako prudká změna intenzity obrazu v daném místě. Rychlé změny se dají detekovat pomocí gradientu, který je pro spojitou funcki $f(x,y)$ v bodě (x,y) vyjádřen následovně.

$$\nabla f(x,y) = \begin{bmatrix} \mathbf{G}_x & \mathbf{G}_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix} \tag{2.11}$$

[1] Na pořadí zde záleží. Není možné dostat ‚libovolnou rotační matici, pokud budou dvě po sobě doucí rotace probíhat kolem stejné osy.

Velikost a směr gradientu se určí pomocí následujících vztahů.

$$\text{mag}(\nabla f) = ||\nabla f||_2 = \sqrt[2]{\mathbf{G}_x^2 + \mathbf{G}_y^2} \tag{2.12}$$

$$\boldsymbol{\theta} = \arctan(\frac{\mathbf{G}_y}{\mathbf{G}_x}) \tag{2.13}$$

Parciální derivace musí být spočtené pro každý pixel (x, y) obrazu $\mathbf{A}$. K tomuto se využívá Sobelových kernelů $\mathbf{K}_x$, $\mathbf{K}_y$. Ty se konvolvují s obrazem a v každém bodě aproximují hodnotu obou parciálních derivací $g_x(x, y), g_y(x, y)$, ze kterých se pak dle vztahu 2.11 určí gradient.

$$\mathbf{K}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & -1 \end{bmatrix} \quad \mathbf{K}_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.14}$$

$$g_x(x, y) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} \mathbf{K}_x(k, l)\mathbf{A}(x + k, y + l) \tag{2.15}$$

$$g_y(x, y) = \sum_{k=-1}^{1} \sum_{l=-1}^{1} \mathbf{K}_y(k, l)\mathbf{A}(x + k, y + l) \tag{2.16}$$

$$\tag{2.17}$$

Určení gradientu intenzity tímto způsobem je relativně nenáročné. Pro zrychlení výpočtu se používá aproximace magnitudy gradientu místo exaktního výpočtu 2.12. Nejjednodušší aproximací je

$$\text{mag}(\nabla f) \approx ||\nabla f||_1 = |\mathbf{G}_x| + |\mathbf{G}_y|. \tag{2.18}$$

Ta dává sice velice nepřesný výsledek, ale pro aplikace, kde stačí hrubý odhad velikosti gradientu, může značně snížit výpočetní dobu.[4, 5]

2.4 Hledání konvexního obalu množiny bodů

Je-li dána množina S obsahující n bodů v euklidovském prostoru $\mathbb{E}^3$, popřípadě euklidovské ploše $\mathbb{E}^2$, pak konvexní obal H množiny S je nejmenší konvexní možina obsahující množinu S.[6]

Mezi nejpoužívanější algoritmy pro hledání konvexního obalu bodů v euklidovské ploše $\mathbb{E}^2$ patří následující.

2.4.1 Grahamův algoritmus

Je nalezen bod $\mathbf{p}_0$, jehož y-ová souřadnice nabývá nejmenší hodnoty ze všech bodů množiny S. Všechny ostatní body jsou seřazeny podle úhlu, který svírají s osou x. K tomuto se využívá obecného třídícího algoritmu, jako je například heapsort, a tento krok má tedy časovou náročnost $\mathcal{O}(n \log n)$. Seřazené body jsou postupně procházeny a na základě následujícího kritéria

zařazeny[2], popřípadě vyloučeny, z konvexního obalu H

$$O(\mathbf{p},\mathbf{r},\mathbf{q}) = \det\begin{pmatrix} p_x & r_x & q_x \\ p_y & r_y & q_y \\ 1 & 1 & 1 \end{pmatrix}, \qquad \begin{Bmatrix} \mathbf{r} \in H, \mathbf{p} = \mathbf{r}, \mathbf{r} = \mathbf{q}, \mathbf{q} = \mathbf{p_i} \text{ iff } O > 0 \\ \mathbf{r} \notin H, \mathbf{p} = \mathbf{p}, \mathbf{r} = \mathbf{q}, \mathbf{q} = \mathbf{p_i} \text{ iff } O \leq 0 \end{Bmatrix}, \tag{2.19}$$

kde $\mathbf{p}_i$ je další bod v pořadí seřazených bodů z S. Výše popsané ověřování bodů je provedeno v čase $\mathcal{O}(n)$.[7]

Na podobném principu funguje i Andrewův algoritmus. Ten využívá lexikografického uspořádání bodů podle x-ové souřadnice, čímž se vyhne výpočtům s destinnou čárkou, které mohou být zdrojem nepřesností.[8]

2.4.2 Jarvisův pochod (Jarvis march)

Je zvolen bod $\mathbf{p}_0$ náležící konvexnímu obalu H, což je například první bod v lexikografickém uspořádání bodů S. Ten je možné najit v čase $\mathcal{O}(n)$. Dále je spočítán relativní úhel mezi $\mathbf{p}_0$ a ostatními body. Z těch je pak vybrán bod $\mathbf{p}_n$, jehož úhel nabývá nejmenší hodnoty. Bod $\mathbf{p}_n$ je přidán do obalu H a postup se opakuje z bodu $\mathbf{p}_n$. Algoritmus končí ve chvíli, kdy platí $\mathbf{p}_n = \mathbf{p}_0$. Tento postup odpovídá postupnému procházení všech vrcholů polygonu, který reprezentuje konvexní obal, a jeho časová náročnost je $\mathcal{O}(hn)$, kde h je počet vrcholů. Může tedy být pro aplikace s předem známým počtem vrcholů rychlejší než Grahamův algoritmus.

2.4.3 Chanův algoritmus

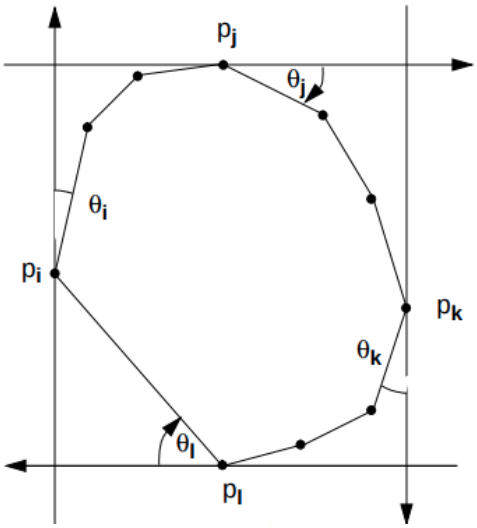
Jedná se o ideální na výstup citlivý algoritmus určený k výpočtu konvexního obalu množiny. Výpočet dosahuje náročnosti $\mathcal{O}(n \log h)$

Množina bodů S je rozdělena do K množin Q_i. V každé množině se nachází maximálně m bodů, kde m je předem určená hodnota, pro kterou musí platit $m \geq h$. Pokud tato rovnost není splněna, musí být hodnota m zvýšena a algoritmus se opakuje. Následně je proveden Grahamův algoritmus na každou množinu Q_i s časovou náročností $\mathcal{O}(n \log m)$. Poté je zvolen bod, který náleží H a je užit Jarvisův pochod. Ten musí najít nejmenší prvek mezi maximálně m seřazenými vrcholy v K množinách, což je realizovatelné v čase $\mathcal{O}(K \log m)$. Pro h vrcholů je tedy výsledná časová náročnost $\mathcal{O}(n \log h)$ za předpokladu, že hodnota m je blízká hodnotě h.[6]

2.5 Nejmenší ohraničující obdélník

Nejmenší ohraničující obdélník je možné určit z konvexního obalu H množiny S. Ten může být nalezen v čase $\mathcal{O}(h)$ například pomocí star algoritmu [9] nebo pomocí algoritmu otáčejících se třmenů (rotating calipers).

[2] Bod zařezený do konvexního obalu může být v dalším kroku vyloučen.



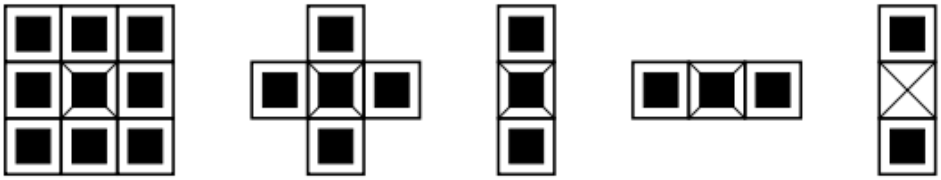
Obrázek 2.1: Otáčející se třmeny [9]

Kolem množiny jsou umístěny dva páry na sebe ortogonální přímek, takzvaných třmenů (viz. obrázek 2.1), které se konvexního obalu dotýkají v některém z vrcholů $\mathbf{p}_i$, nebo mají společnou přímku spojující vrcholy $\mathbf{p}_{i-1}$ a $\mathbf{p}_i$. V každém kroku je spočítán úhel θ_i mezi třmenem, který se dotýká vrcholu $\mathbf{p}_i$, a vrcholem $\mathbf{p}_{i+1}$. Tento výpočet se opakuje pro všechny 4 třmeny, viz. obrázek 2.1. Následně je výbrán úhel $\theta = \min\{\theta_i, \theta_j, \theta_k, \theta_l\}$, o který je provedena rotace všech třmenů. Třmeny jsou pak ve směru svého normálového vektoru posunty tak, aby platila výše uvedená podmínka dotyku třmenu s konvexní množinou v jednom, resp. dvou vrcholech. Dále je vypočten obsah obdélníku, jehož strany tvoří výše zmíněné třmeny. Celý postup se poté opakuje, dokud není vyzkoušeno všech h obdélníků. Následně je vybrán ten s nejmenší plochou.[9]

2.6 Matematická morfologie ve zpracování obrazu

Matematická morfologie zpracovává obraz v závislosti na jeho tvaru. Jejím účelem je zjednodušit obraz při zachování základního tvaru a potlačení nepodstatných informací obsažených v obrazu [10]. K filtraci obrazu se využívá strukturní element H. Tento element slouží jako lokální sonda, citlivá na geometrickou informaci. Příklad strukturních elementů je na obrázku 2.2, kde symbolem $\times$ je znázorněn lokální počátek, ke kterému je element H vztažen. Základními morfologickými operacemi jsou: dilatace, eroze, otevření a uzavření. Tyto operace jsou reprezentovány symboly $\oplus, \ominus, \circ, \bullet$.

Morfologické operace mohou být definovány pro binární, černobílý i barevný obraz. Nechť $X \subseteq \mathbb{Z}^2$ je binární obraz, pak základní morfologické operace



Obrázek 2.2: Běžné sondy používané při v matematické morfologii

jsou definovány následovně.[11]

$$X \oplus H = \{(x, y) : H_{(x,y)} \cap X \neq \emptyset\} \tag{2.20}$$

$$X \ominus H = \{(x, y) : H_{(x,y)} \cap X \subseteq X\} \tag{2.21}$$

$$X \circ H = (X \ominus H) \oplus H \tag{2.22}$$

$$X \bullet H = (X \oplus H) \ominus H \tag{2.23}$$

Kde jako $H_{(x,y)}$ je označena sonda H, jejíž počátek byl posunut do bodu (x, y).

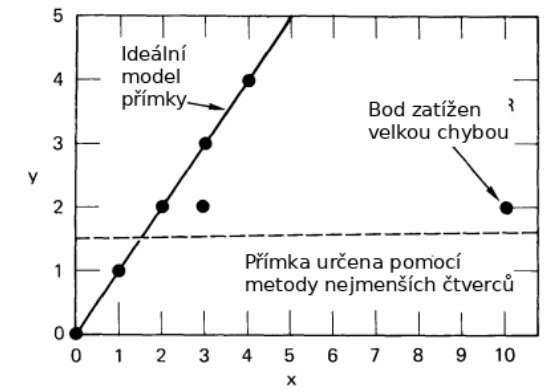
2.7 Random sample consensus

Při určování modelu podle statistických metod, jako je prokládaní pomocí nejmenších čtverců, může jediný bod zatížený velkou chybou znehodnodit aproximaci matematickým modelem. Příklad takové situace je vidět na obrázku 2.3. Iterativní algoritmus Random sample consensus (RANSAC) je vhodný v situacích, kde jsou data zatížena velkým množstvým chybových bodů, popřípadě body s velkou magnitudou chyby. Princip algoritmu je následující: [12]

Algorithm 1 RANSAC

1: Je náhodně vybrán minimální počet bodů potřebných k určení parametrů modelu.
2: Jsou nalezeny parametry modelu.
3: Je určeno kolik bodů z množiny S je od modelu vzdáleno maximálně ϵ.
4: Pokud poměr bodů určených v kroku 3 ku celkovému počtu je větší než předem určený limit τ, pak algoritmus končí a model je určen z bodů určených v bodě 3.
5: Jinak jsou kroky 1 až 4 jsou opakovány až n krát.
6: Pokud během n běhů nebyl algoritmus ukončen, je vybrán model, pro který byl počet bodů získených v kroku 3 největší.

Algoritmus tedy není deterministický a je citlivý na správné nastavení jednotlivých parametrů. Zejména pak na počet opakování n. Tento se parametr se volí experimentálně. Pokud je známa provděpodobnost výskytu chybového bodu, pak můžeme parametr n určit na základě požadované pravděpodobnosti p, že alespoň jedna sada náhodných vzorků neobsahuje bod zatížený velkou



Obrázek 2.3: Vliv bodu zatíženého chybou na prokládání přímky pomocí nejmenších čtverců [13]

chybou (tzv. outlier). Pokud jako v označíme pravděpodobnost bodu s velkou chybou a u bod jehož chyba není markantní, pak platí následující vztahy.[13]

$$1 - p = (1 - u^m)^N \tag{2.24}$$

$$n = \frac{\log(1 - p)}{\log(1 - (1 - v)^m)} \tag{2.25}$$

Písmenem m je označen počet bodů potřebný k určení modelu.

Kapitola 3

Kamera

3.1 Dírkový model kamery

Kamera zobrazuje body euklidovského prostoru $\mathbb{E}^3$, které jsou popsány pomocí světových souřadnic, na body v euklidovské ploše $\mathbb{E}^2$, která se nazývá obraz. Body obrazu se označují jako pixely.

Nejjednodušším modelem kamery je model dírkový, který má optické centrum, neboli centrum projekce, v konkrétním bodě. Centrum projekce umístíme do počátku kartézského souřadného systému a budeme uvažovat plochu kolmou na osu Z našeho souřadného systému, kterou nazveme obrazovou rovinou. Ta je popsána jediným parametrem a to ohniskovou vzdáleností f. V dírkovém modelu kamery se bod $\mathbf{p} = (x, y, z)^T$ v prostoru promítne na bod obrazu pomocí zobrazení

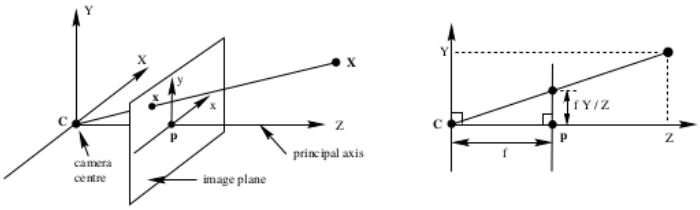
$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \mapsto \begin{bmatrix} fx/z \\ fx/z \end{bmatrix}, \tag{3.1}$$

což odpovídá bodu, kde přímka spojující centrum projekce a bod v prostoru protne obrazovou rovinu (viz obrázek 3.1). Ve vztahu 3.1 jsme předpokládali, že počátek souřadného systému obrazu je v optickém středu. Konvencí však je za počátek souřadnic zvolit levý horní roh obrazu. K zápisu využijeme homogenních souřadnic a dostaneme následující rovnici

$$\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \mapsto \begin{bmatrix} fx + zc_x \\ fy + zc_y \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & c_x & 0 \\ 0 & f & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{3.2}$$

kde jsme jako c_x a c_y označili souřadnice optického středu. Označíme-li obecný bod v prostoru, který je popsán v souřadném systému kamery, $\mathbf{x}_p$ a odpovídající bod v rovině obrazu $\mathbf{x}$, pak můžeme rovnici 3.2 přepsat jako

$$\mathbf{x} = \mathbf{K}[\mathbf{I} \mid 0]\mathbf{x}_p \quad \mathbf{K} = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.3}$$



Obrázek 3.1: Model dírkové kamery

Matice $\mathbf{K}$ se nazýva matice vnitřních parametrů kamery.

Obecně je bod $\mathbf{x}$ v euklidovském prostoru popsán v jiném souřadném systému, než je ten se kterým je svázána kamera. Tento souřadný systém se nejčastěji nazývá světový. Přechod do souřadného systému kamery ze světového je možno popsat následovně

$$\mathbf{x}_p = \begin{bmatrix} \mathbf{R} & -\mathbf{R}\tilde{\mathbf{C}} \\ 0 & 1 \end{bmatrix} \mathbf{x}. \tag{3.4}$$

Jako $\tilde{\mathbf{C}}$ jsme označili počátek souřadného systému kamery vyjádřený v světových souřadnicích a $\mathbf{R}$ rotaci světového souřadného systému vzhledem ke kameře. Dosadíme-li nyní rovnici 3.3 do 3.4 dostaneme vztah popisující transformaci bodu ze světových souřadnic do souřadnic obrazu

$$\mathbf{x} = \mathbf{K}\mathbf{R}[\mathbf{I} \mid -\tilde{\mathbf{C}}]\mathbf{x}. \tag{3.5}$$

Parametry $\tilde{\mathbf{C}}$ a $\mathbf{R}$ se souhrně nazývají vnější parametry kamery.

3.2 Stereo kamera

3.2.1 Výpočet vzdálenosti

Stereo kamera je tvořena dvěmi nezávislými čočkami. Zpracováním obrazů z obou čoček jsme schopni získat informace o hloubce každého pixelu[1]. Budeme uvažovat model stereokamery, kde jsou osy obou čoček rovnoběžné (viz obrázek 3.2). Osy jsou od sebe ve vzdálenosti b. Tomuto parametru se říka báze (baseline). Z podobnosti trojúhelníků plyne

$$\frac{z}{f} = \frac{x}{xl} \tag{3.6}$$

$$\frac{z}{f} = \frac{x-b}{xr}. \tag{3.7}$$

Pokud z rovnice 3.6 vyjádříme x a dosadíme do rovnice 3.7, pak po úpravě dostaneme výsledný vztah pro vzdálenost bodu od kamery

$$z = \frac{fb}{xl - xr} = \frac{fb}{d}. \tag{3.8}$$

[1]Hloubkou bodu $\mathbf{p}$ je myšlena vzdálenost ortogonální projekce $\mathbf{p}$ na optickou osu kamery od počátku souřadného systému, který je svázan s kamerou.

Hodnota d se nazývá rozdíl (disparity) a xl, xr jsou souřadnice bodu $\mathbf{p}$ v levém, resp. pravém obrazu stereokamery. Pro každý bod v levém obrazu je tedy potřeba najít odpovídající bod v obrazu pravém (tzv. stereo pár). Bez jakékoliv apriorní znalosti by to znamenalo pro každý pixel prohledat celý druhý obraz, tedy náročnost hledání $\mathcal{O}(n^2)$, kde n odpovídá velikosti strany obrazu v pixelech. Pomocí epipolární geometrie můžeme hledání zúžit na přímku a snížit tak náročnost hledání stereo páru na $\mathcal{O}(n)$.[14]

3.2.2 Chyba výpočtu vzdálenosti

Chyba výpočtu vzdálenosti se určí derivací rovnice 3.8 podle d [15]

$$\frac{\partial z}{\partial d} = \frac{z}{fb} \tag{3.9}$$

$$|\epsilon_z| = \frac{z^2}{fb}|\epsilon_d|, \tag{3.10}$$

kde ϵ_z je chyba určení vzdálenosti bodu a ϵ_d je chyba rozdílu. Tato chyba je vlastností kamery, popřípadě algoritmu použitého ke hledání odpovídajících párů, a nabývá téměř konstantních hodnot [15]. Z rovnice 3.10 tedy plyne, že chyba je úměrná kvadrátu vzdálenosti daného bodu a přesnost stereokamer se vzdáleností rychle klesá.

3.2.3 Epipolární geometrie

Epipolární geometrie se zabývá průnikem obrazových rovin se svazkem ploch, jejichž společná osa je báze. Uvažujme dvě dírkové kamery v obecném natočení, tedy jejich obrazové roviny nemusí být rovnoběžné, viz 3.3. Z dírkového modelu kamery plyne, že bod $\mathbf{X}$, který se nachází v euklidovském prostoru, a optické středy kamer C, C' jsou koplanární a tvoří rovinu p. V místě, kde tato rovina protíná roviny obrazu, se nachází epipolární přímky $\mathbf{l}$ a $\mathbf{l}'$[2]. Hledáme-li stereo pár, pak pro známý bod $\mathbf{x}$ (tj. projekce bodu $\mathbf{X}$ do optické roviny levé kamery) musíme najít odpovídající bod $\mathbf{x}'$. Ten může ležet pouze na epipolární přímce $\mathbf{l}'$, která je jednoznačně určena epipólem $\mathbf{e}'$ a bodem $\mathbf{x}'$. Epipól se nachází v místě, kde přímka spojující optická centra kamer, protíná rovinu pravého obrazu. Pro bod $\mathbf{x}'$ platí

$$\mathbf{x}' = \mathbf{H}\mathbf{x}, \tag{3.11}$$

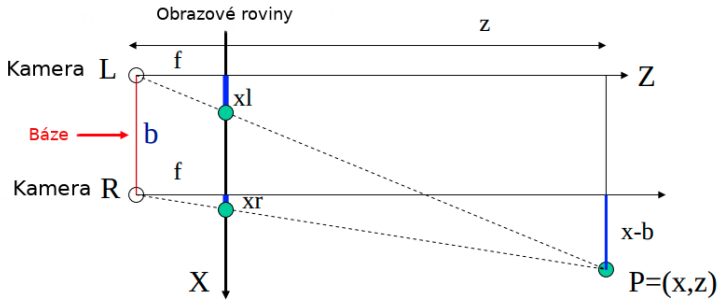
kde $\mathbf{H}$ je homografie mezi rovinami obrazu. Toto zobrazení je ovlivněno pouze vzájemnou polohou kamer a jejich vnitřními parametry. Nezávisí tedy na scéně. Z bodu $\mathbf{x}'$ lze určit epipolární přímku

$$\mathbf{l}' = \mathbf{e}' \times \mathbf{x}' \tag{3.12}$$

$$\mathbf{l}' = \mathbf{e}' \times \mathbf{H}\mathbf{x} \tag{3.13}$$

$$\mathbf{l}' = \mathbf{F}\mathbf{x}. \tag{3.14}$$

[2] Přímky tll a $\mathbf{l}'$ jsou popsány pomocí homogenních souřadnic.



Obrázek 3.2: Určení vzdálenosti bodu ze stereokamery, převzato z [17]

Matice $\mathbf{F}$ se nazývá fundamentální a představuje zobrazení $f : \mathbb{R}^3 \mapsto \mathbb{R}^3$. Pokud jsou obě optické osy stereokamery rovnoběžné, pak se výpočet epipolární přímky velice zjednoduší, jelikož homografie mezi rovinami obrazu je pouze posunem ve směru báze.[16]

3.3 Hledání stereo páru

Metody užívané pro hledání stereo páru se dělí na metody plošné (area based) a metody založené na porovnávání rysů (features). Při hledání stereo páru se často využívá následujících omezení, která tento proces zjednoduší. Tato omezení však v extrémních případech, jako je například okluze, nemusí platit.

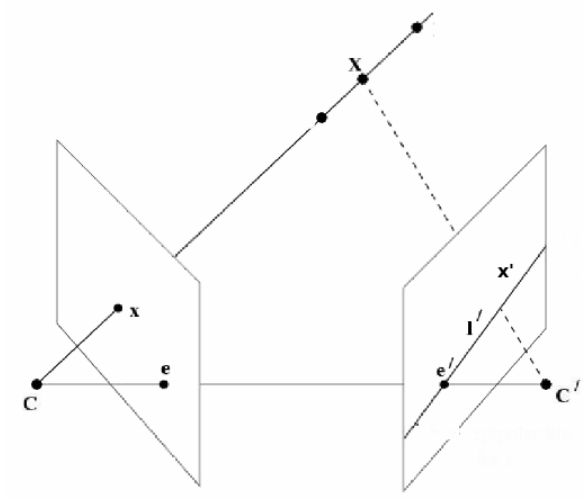
- Epipolární omezení: Pixel v druhém obrazu se nachází (pokud existuje) na epipolární přímce, viz. sekce 3.2.3.
- Spojitost: Rozdílová, popř. hloubková, mapa by měla být po částech spojitá.
- Jedinečnost: Každý pixel v levém obrazu má k sobě právě jeden odpovídající v obrazu pravém.
- Pořadí: Pořadí pixelů v levém obrazu odpovídá pořadí v pravém.

3.3.1 Plošné metody

Plošné metody využívají informaci z okolí pixelu. Tyto informace jsou porovnány se všemi přípustnými pozicemi v druhém obrazu a bod, jehož hodnota je nejblíže vzoru, tvoří stereo pár. Mezi nejpoužívanější variace plošných metod patří následující.

Součet absolutních rozdílů

Je definována konstatní velikost okolí pixelu $\mathbf{x}(u, v)$. Následně jsou hodnoty všech pixelů v okně sečteny a výsledná hodnota je provnána s body v pravém



Obrázek 3.3: Projekce bodu na obrazové roviny, převzato z [18]

obrazu, kde se může korespondující pixel nacházet, tj. body které splnují námi používaná omezení. Pro odpovídající bod $\mathbf{x}'$ tedy platí

$$\mathbf{x}' = \text{argmin} \left(\sum_i \sum_j |\mathbf{I}_l(u+i, v+j) - \mathbf{I}_r(u+i, v+j+d)| \right), \qquad (3.15)$$

kde $\mathbf{I}(u, v)$ je intenzita příslušného pixelu, d je posun okna ve druhém obrazu a součet probíha přes celé okno. Vztah byl zjednodušen uvažováním horizontálních epipolárních přímek. Toto odpovídá konfiguraci kamery s rovnoběžnými optickými osami.

Existuje mnoho podobných algoritmů, které se liší pouze metrickou funkcí. Místo absolutní hodnoty rozdílu je užit například kvadrát rozdílu nebo jeho normalizovaný kvadrát. Jedná se o nejrychlejší algoritmy pro hledání stereo páru, přesto však dosahují vysokých přesností.[19, 16]

Cenzus

Jedná se o algoritmus patřící do rodiny plošných metod, které před porovnáním hodnot pixelů provedou určitou transformaci obou obrazů. Patří sem naříklad i transformace podle hodnoty (rank transform). Cenzus opět pracuje s definovanou maskou kolem pixelu, pro který hledá stero pár. Pixely, které se nachází v šabloně, se transformují na vektor jedniček a nul podle toho, zda je hodnota intenzity pixelů masky větší, popřípadě menší, než intenzita centrálního pixelu. Stejná transformace se provede i v pravém obrazu pro každý pixel, který může doplnit stereo pár. Vektory jsou následně porovnány a je vybrán ten, jehož vektor má nejmenší Hammingovu vzdálenost [19, 14].

3.3.2 Porovnávání rysů

Pro každý obraz jsou vygenerovány důležité body. To mohou být například hrany či rohy v obrazu. Tyto rysy jsou porovnány s analogicky vygenerová-

Obrázek 3.4: Fotografie kamery Intel® Realsense™ D435 [20]

nými rysy v druhém obrazu. Tento postup se v některých aplikacích opakuje, přičemž v každém dalším kroku je každý rys rozložen na několik menších, o kterých již máme apriorní znalost jejich přibližné polohy. Aby bylo možné sestavit obraz pouze z několika rysů, je nutné provést časově náročné předzpracování obrazu za účelem najití vhodných rysů a po nalezení korespondujících rysů provést zpětnou rekonstrukci obrazu.[14]

Pro porovnání rysů mezi levým a pravým obrazem se využívá algoritmů jako je například SIFT , SURF nebo BRIEF.

3.4 Intel Realsense D435

Intel® Realsense™ D435 je širokoúhlá stereokamera, která se skládá z RGB kamery, infračerveného projektoru a dvou infračervených kamer, jejichž data jsou zpracovává přímo v čipu kamery dedikovaným procesorem. Výstupem z kamery je tedy barevný obraz a vzdálenost jednotlivých pixelů, neboli hloubková mapa.

Dvě infračervené kamery jsou v konfiguraci s rovnoběžnými optickými osami ve vzdálenosti 50 mm. Infračervený projektor promítá na scénu statický obrazec, který se nachází mimo viditelné spektrum a není tedy zachycen RGB kamerou. Tento obrazec slouží k nalezení stereo párů, zejména u objektů s řídkou texturou.

Kamera je vybavena specializovaným procsorem Vision Processor D4 pro výpočet hloubkové mapy. Ten je schopen při rozlišení hloubkové kamery 848x480 zpracovávat až 90 snímků za sekundu. Při této rychlosti snímání je přesnost kamery výrazně snížena [15]. Pro maximální přesnost je vhodné nastavit frekvenci na 30 snímků za sekundu.

Procesor má stejně jako celá kamera velice nízký odběr elektrické energie. Celá kamera včetně IR projektoru má odběr menší než jeden watt a je tedy vhodná pro mobilní aplikace, kde je velikost baterie a spotřeba energie omezujícím faktorem [21].

Pro hledání stereo párů je využit algoritmus Cenzu popsaný v 3.3.1 s maskou o velikosti 7×7. Výsledky jsou následně ověřovány sadou filtrů, které měří důvěryhodnost shody. Podle nastaveného limitu důvěryhodnosti pak pro tento pixel buď vygenerují záznam v hloubkové mapě, nebo pixel zůstane nevyplněn. Výsledky kamery pro různé hodnoty limitu jsou vidět v tabulce 3.1.

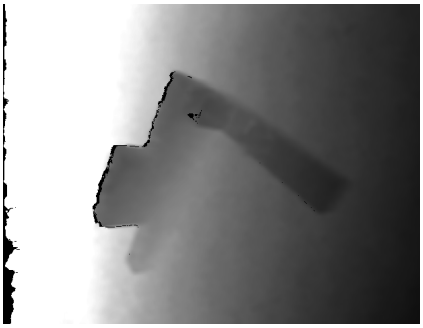
Důvěryhodnost	FPR	$r_p = 95\%$	$\max(\sigma_x, \sigma_y) > \sigma_z$
Vypnuta	91,3%	7.0 m	6.1 m
Nízká	19,8%	6,9 m	6.6 m
Střední	5,8%	5,8 m	6.7 m
Vysoká	0,5%	4,2 m	6.8 m

Tabulka 3.1: Výstup kamery v závislosti na nastavené důvěryhodnosti. Postupně zleva: Hodnota limitu, počet falešných shod pokud je scéna blíže než je minimální vzdálenost detekce kamerou. Vzdálenost, kdy vyplněnost hloubkové mapy klesne pod 95%, a vzdálenost při které je směrodatná odchylka ve směru osy z menší než ve směru x a y. Převzato z [15].

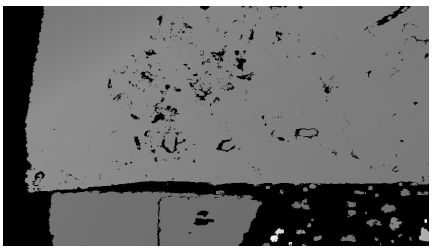
Resolution	D400/D410/D415	D420/D430
	Min-Z (mm)	Min-Z (mm)
1280x720	450	280
848X480	310	195
640x480	310	175
640x360	240	150
480x270	180	120
424x240	160	105

Tabulka 3.2: Tabulka minimální detekovatelné hloubky [21]

Procesor kamery není schopen určit hloubku bodů, pokud se objekt nachází příliš blízko. Konkrétní hodnoty je možno vidět v tabulce 3.2. Maximální měřitelná vzdálenost je za ideálních podmínek až 40 metrů [15]. S rostoucí vzdáleností se kamera chová podle rovnice 3.10 a přesnost tedy rychle klesá. Toto chování neodpovídá rovnici 3.10, pokud kamera operuje s objekty, které se nachází na obou hranicích měřitelné vzdálenosti. Ilustrace výstupu kamery, pokud se objekt nachází ve vzdálenosti meší než specifikuje tabulka 3.2, je vidět na obrázku 3.5b.



(a) : Snímek bez výrazných výpadků dat hloubkové mapy



(b) : Snímek s chybějícími hloubkovými daty

Obrázek 3.5: Výstupní data kamery RealSense, hloubková data byla převedena do odstínů šedé

Kapitola 4

Metody v detekci objektů z hloubkových dat

Existuje mnoho různých metod pro detekci objektů z hloubkových dat. Tyto přístupy se liší podle různorodosti detekovaných objektů, požadavků na čas výpočtu, možnosti získání trénovacích dat nebo formátem vstupních dat. Obvyklým formátem je tak zvané mračno bodů (anglicky point cloud), což je nestrukturovaná množina bodů v trojrozměrném prostoru. Zejména v posledních letech s nástupem hloubkových kamer jako je například Microsoft Kinect, Asus Xtion nebo námi používaný Intel RealSense, se čím dál více používájí data ve formátu RGB-D obrazu. Ten se dá pomocí vztahu 3.2 převést na mračno bodů. Výstupní data z hloubkové kamery bývají většinou oproti LIDARu méně přesná. Výhodou naopak je vysoká vzorkovací frekvence, velké množství výstupních bodů a nízká cena zařízení.

4.1 Určení normálového vektoru plochy z nestrukturovaných dat

Téměř všehny metody detekce objektů z mračna bodů se skládájí z více kroků a jedním ze základních kroků většinou bývá určení normálového vektoru snímané plochy. Obvykle se jedná o plochu reprezentující zem, popřípadě jinou podložku, nebo plochy reprezentující stěny místnosti. Přístupy k hledání ploch v obrazu se výrazně liší, v této práci uvedeme pouze příklady některých z nich.

4.1.1 Použití hrubé síly

Postupně jsou vyzkoušeny všechny možnosti, popřípadě část z nich, a vybere se nejlépe vyhovující. Využívá se zde zejména algoritmů jako je RANSAC, případně jeho modifikace. Tato metoda je vhodná zejména pokud jsou data zatížena silným šumem, jelikož při správném počtu iterací téměř vždy vrátí správný výsledek. Toto je však vykoupeno vysokou časovou náročností [22, 23], která se dá snížit vhodným předzpracováním dat, například segmentací na menší celky a určením plochy pouze pro tyto segmenty. [24]

4.1.2 Výpočet normálového vektoru v každém bodě

Normálový vektor je kolmý na plochu reprezentující okolí bodu. Toto okolí je tvořeno body nacházejícími se v okruhu o poloměru r [25] ,popřípadě k nejbližšími body [26, 27]. Poté je provedena segmentace pomocí metody rostoucích oblastí viz. sekce 4.2. Výstupem jsou tedy skupiny bodů, které představují jednotlivé plochy. V některých případech je navíc použit algoritmus RANSAC na již nalezené segmenty za účelem vyloučení bodů, které leží mimo plochu představovanou daným segmentem, přestože jejich normálový vektor dané ploše odpovídá [28].

4.1.3 Principal Component Analysis

Jsou nalezeny body, které patří do dané roviny, a těmi je poté pomocí Principal Component Analysis (PCA) [29] proložena rovina. Tato metoda vyžaduje přesné určení bodů, o kterých se předpokládá, že tvoří danou rovinu. Jediný bod ležící ve velké vzdálenosti od roviny (outlier) může způsobit velké nepřesnosti, jelikož rovina je proložena body metodou nejměnších čtverců, viz obrázek 2.3.[30]

4.2 Segmentace obrazu

Důležitým krokem pro porozumění obrazu a detekci objektů je segmentace obrazu, tj. rozdělení vstupních dat na skupiny. Rozdělujeme dva typy segmentace. Sémantickou, která sjednocuje body reprezentující danou skupinu objektů (např. všechny bloky ležící na zemi), a segmentaci jednotlivých instancí, kdy je skupina objektů rozdělena na jednotlivé zástupce dané skupiny.

Přístup k segmentaci se líší podle dané aplikce. Nejpoužívanějšími metodami jsou tyto.

4.2.1 Metoda prahování

Body $\mathbf{p}(x, y)$ jsou rozděleny do m skupin podle toho, zda jejich vlastnost V, což je například vzdálenost bodu od kamery nebo od plochy, dosahuje stanovené hodnoty T_m. Tato hodnota může být pevně určená, nebo se může dynamicky měnit, a to v závislosti na okolních bodech, nebo v závislosti na pozici v obrazu, tj. $T(x, y)$. Pro prahovací metodu tedy platí následující vztah [31]

$$\left\{ \begin{array}{l} \mathbf{p} \in m \text{ iff } V(\mathbf{p}) > T_m \\ \mathbf{p} \in n \text{ iff } V(\mathbf{p}) > T_n \\ \mathbf{p} \in \emptyset \text{ iff } V(\mathbf{p}) \leq T_1 \end{array} \right\}. \tag{4.1}$$

4.2.2 Metoda detekce hran

V obrazu jsou detekovány prudké změny pozorované vlastnosti V v bodě $\mathbf{p}$ za pomoci první derivace vlastnosti V v bodě $\mathbf{p}$. Pokud hodnota derivace

dosáhne daného prahu je **p** registrován jako hrana. Po výpočtu bývá provedena úprava těchto hran, kterou je například filtrování, popřípadě spojení některých sousedních bodů. Nakonec jsou v obrazu identifikovány segmenty bodů, které jsou od zbytku odděleny hranou. K určení hran se využívá například Sobelova operátoru, který byl popsán v sekci 2.3, nebo Cannyho operátoru.[23]

4.2.3 Metoda rostoucí oblasti

Metoda rostoucí oblasti se dělí na dva typy. S počátečním bodem (seed) a bez počátečního bodu.

Je zvolen jeden, popřípadě několik počátečního bodů. K vybraným bodům se postupně přidávají sousední body, pokud splňují předem definované podmínky. Těmi jsou například maximální odchylka normálových vektorů bodů od normálového vektoru počátečního bodu, nebo rozdíl velikosti některé souřadnice. Metoda bez počátečního bodu pak rozdělí všechny body do skupin, jejichž body spolu sousedí a zároveň mají podobné vlastnoti. Jedná se o algoritmus, který pracuje s konstatní časovou náročností odpovídající $\mathcal{O}(2n)$, kde n je počet pixelů obrazu. [27, 26]

4.2.4 Metoda rozdělovaní a slučování

Obraz je postupně rozdělen na části, které mají podobnou charakteristiku. Následně jsou sousední regiony, které jsou si podobné, sloučeny do jedoho. Pro reprezentaci regionů se obvykle využívá kvadrantový strom (quadtree).

Nechť o je obraz a V daná vlastnost jednotlivého bodu **p**. Skupina bodů má vlastnost V, pokud tuto vlastnost má každý bod skupiny. Metodu rozdělování a slučování lze za těchto podmínek popsat následovně.[31]

Algorithm 2 Rozdělování a slučování

1: Region R_1 je roven o.
2: Pokud platí $V(R_i) = False$, pak je region rozdělen na několik menších.
3: Pokud platí $V(R_i) = True$, pak je R_i sloučen se všemi sousedními regiony R_j, přičemž musí platit $V(R_i \cup R_j)$.
4: Krok 3 se opakuje, dokud je možné některý region sloučit.

4.2.5 Metoda shlukování (clustering)

Obraz je rozdělen do shluků, ve kterém má každý bod podobné vlastnosti, resp. jejich vlastnosti jsou si v rámci obrazu nejbližší. Vlastnost v tomto případě bývá nejčastěji euklidovská vzdálenost. Mezi nejpopulárnější algoritmy patří[1]

- K-means algoritmus. V tomto algoritmu je předem nutné znát počet shluků m, do kterých budou body rozděleny. Následně se náhodně vybere m bodů, které představují střed shluku. Poté je pro každý bod spočítána

[1] Princip algoritmů bude vysvětlen při shlukování podle euklidovské vzdálenosti.

vzdálenost od jednotlivých středů. Bod je přiřazen do daného shluku, jehož vzdálenost středu je nejmenší. Následuje přepočítaní středů a postup se opakuje. Ve chvíli kdy již nedochází ke změnám mezi jednotlivými shluky je zaznamenán součet rozptylů bodů v rámci jednotlivých shluků a celý proces počínaje náhodnou volbou středů se n-krát opakuje. Nakonec je vybrán výsledek, který má nejmenší rozptyl.[32]

- Hladový (greedy) algoritmus. Je opět definován počet shluků m. Následně je vybrán náhodně bod $\mathbf{p}$. Tento bod je přidán do množiny středů shluků M. Poté je najit bod, jehož vzdálenost od nejbližšího bodu množiny M je největší. Tento bod je přidán do množiny M a postup se opakuje, dokud množina M neobsahuje m bodů. Tento algoritmus dosahuje menší přesnosti, než K-means. Dokáže však najít shluky v čase $\mathcal{O}(nm)$, kde n je počet všech bodů. [27, 24]

- Mean-shift algoritmus. Je předem definován poloměr kruhového okna r. Následně je náhodně určeno (popřípadě podle předem definované masky rozmístěno) n bodů $\mathbf{p}$. V každé iteraci se spočítá střed bodů $\mathbf{t}$, pro které platí $||\mathbf{t} - \mathbf{p}|| < r$, ty nahradí bod $\mathbf{p}$. Tento postup se opakuje, dokud body konvergují. Výsledná poloha bodů $\mathbf{p}_i$ pak představuje sřed jednotlivých shluků. [33]

- DBSCAN algoritmus. Všechny body obrazu jsou označeny jako nenavštívené. Náhodně je vybrán nenavštívený bod $\mathbf{a}$ a pokud se v jeho okolí o velikosti ϵ nachází minimální předem zvolený počet bodů, je celé okolí bodu $\mathbf{p}$ přidáno do shluku a postup se opakuje pro přidané body. Pokud již nelze žadný další bod přidat, zvolí se další nenavštívený bod a proces se opakuje pro další shluk. Toto probíhá dokud existují nenavštívené body. [34]

4.3 Detekce objektů

Ve velké části aplikací je postačujícím výstupem segmentace obrazu, která sama o sobě poskytuje informace potřebné k porozumění daného obrazu. Některé aplikce, jako je třeba manipulování s objekty, potřebují přesně identifikovat polohu daného objektu.

4.3.1 Detekce ze segmentovaných dat

Nejjednodušším případem je detekce objektu z již segmentovaných dat. Máme tedy množinů bodů, které představují daný objekt. Obvykle se v takovémto případně používá nalezení nejmenšího kvádru, který tyto body ohraničuje. Hledání ohraničující kvádru v euklidovském prostoru $\mathbb{E}^3$ je realizováno pomocí O'Rourkeho algoritmu, který má časovou náročnost $\mathcal{O}(n^3)$[35]. Pro většinu aplikací je tedy exaktní výpočet pomalý a jsou použity heuristické metody. Mezi tyto metody patří například aproximace pomocí PCA s časovou náročností $\mathcal{O}(n)$. Přesnějších výsledků dosahují modifikace PCA, jako je min-PCA

a max-PCA. Tyto algoritmy najdou jednu z os ohraničujícího kvádru a zjednoduší tím problém na hledání minimální ohraničujícího obdélníku, což je $\mathcal{O}(n)$ problém, viz sekce 2.5. Další používanou heuristikou je proložení segmentovaných bodů dvěmi kolmými rovinami pomocí RANSAC. Provedením projekce bodů na tyto roviny a najitím nejmenšího ohraničujícího obdélníku dostaneme dostatek informací pro zkonstruování ohraničující kvádru [36]. [35]

Reprezentace objektu kvádrem není pro některé tvary a aplikace vhodná. Můžeme tedy objekt rozdělit na více částí a tyto popsat ohraničujícím kvádrem, jako například v [37]. Pokud máme apriorní znalost o tvaru objektů nacházejících se na scéně, může využít RANSAC algoritmu pro hledání primitivních tvarů, jako jsou například válce a kužely [38]. Pokud máme CAD modely hledaných objektů je možné pomocí ICP (Iterative Closest Points) porovnat CAD model s mračnem bodů [39].

4.3.2 Detekce z nestrukturovaných dat

Nejprve je potřeba určit oblast, kde se daný objekt může nacházet. K tomuto se využívá například algoritmus posouvacího okénka. Tento postup se často opakuje pro různé velikosti okénka, jelikož velikost objektu se v závislosti na vzdálenosti mění. Následně je okénko porovnáno se všemi typy objektů, které se mohou ve scéně vyskytovat. Toto se obvykle provádí vygenerovaním příznaků, ty jsou pak porovnány pomocí algoritmů jako je SURF, či ORB. Pokud je k dispozici trénovací množina je možné transformovat obrazu pomocí HOG a následně ke klasifikaci využít SVM [40, 41].

Kapitola 5

Navržené řešení

Navržené řešení se skládá ze 3 částí a to v souladu se sekcí 4, tedy nalezení normálového vektoru podložky, segmentace obrazu na jednotlivé cihly a následné nalezení jejich ohraničujících kvádrů. Pro všechny části řešení bylo navrhnuto více postupů a jejich výsledky jsou porovnány v sekci 7. Ve všech částech vychazíme ze zadání, které specifikuje, že na zemi nemůže být kromě cihel žadný jiný objekt podobných rozměrů. Velikost cihel je předem známa mimo jejich délky, která může nabývat různých rozměrů. Cihly na sebe mohou být naskládany v libovolném počtu vrstev, mohou se objevit v libovolném natočení a navzájem se dotýkat. Dále platí, že pokud se některá cihla nachází patře n, pak se ve všech patrech $0, ..., n-1$ pod touto cihlou také nachází cihly. To znamená, že pokud jsou cihly postaveny do zdi, pak tato zeď nemá díry.

V některých částech byl zaveden předpodklad doteku dvou cihel pouze po celé délce odpovídajících stěn za účelem rychlejší alternativy detekce.

5.1 Detekce normálového vektoru a sémantická segmentace obrazu

Začneme určením normálového vektoru roviny reprezentující zem. Z jeho znalosti můžeme odečtením plochy představující zem najít body ležící nad zemí, tj. body představující cihly. Dostaneme tedy sémanticky segmentovaný obraz.

5.1.1 Přístup založený na výšce

Vytvoříme mřížku bodů, které se nachází v místech, kde předpokládáme přesný výstup kamery. U těchto bodů ověříme, zda pro ně byla kamerou vygenerována hloubka a zda bod neleží ve větší vzdálenosti než jsou 4 m [1]. Množinu bodů $M = \mathbf{a}_1 \dots \mathbf{a}_m$, které splňují výše uvedená kritéria, proložíme rovinou pomocí PCA algoritmu a dostaneme normálový vektor $\mathbf{n}$ plochy p minimalizující kvadrát vzdáleností od bodů. Tato plocha je posunuta mimo

[1] Vzdálenost 4m byla určena podle [15], kdy chyba stereokamery RealSense D435 nad tuto vdálenost začíná být výrazná a přesné určení normálového vektoru již není možné.

počátek o d.

$$\bar{\mathbf{a}} = \frac{1}{m}(\mathbf{a}_1 \dots \mathbf{a}_m) \tag{5.1}$$

$$d = \mathbf{n} \cdot \bar{\mathbf{a}} = \mathbf{n}^T \bar{\mathbf{a}} \tag{5.2}$$

$$p : n_1 x + n_2 y + n_3 z - d = 0 \tag{5.3}$$

Každý bod $\mathbf{a}$ je buď součástí země nebo součastí cihly. Je-li v množině M k bodů, které jsou součastí země G, pak zbylých $m - k$ bodů musí být součastí mračna bodů reprezentující cihlu C. Tyto body se vždy nachází ve výšce z[2]nad zemí a tedy pro každý bod $\mathbf{a}$ platí

$$n_1 a_1 + n_2 a_2 + n_3 a_3 + \epsilon > d \;\rightarrow\; \mathbf{a} \in C, \tag{5.4}$$

$$n_1 a_1 + n_2 a_2 + n_3 a_3 + \epsilon \leq d \;\rightarrow\; \mathbf{a} \in G, \tag{5.5}$$

kde ϵ je experimentálně určená hodnota ošetřující případy při kterých platí $k = m$, tj. všechny body reprezentují zem. V tomto případě bude část bodů ležet nad plochou p. Tato skutečnost je dána šumem dat a nepřesností kamery. Přidáním parametru ϵ je zajištěno, že každý bod reprezentující zem, bude správně klasifikován i při datech zatížených šumem kamery.

Po výběru bodů z mřížky dle výše uvedených kriterií dostaneme k bodů, které reprezentují zem a nacházejí se v místech, kde je přesnost kamery v rámci jejich možností nejvyšší. Tyto body je možné znovu proložit plochou, čímž získáme lepší aproximaci natočení země pomocí normálového vektoru $\mathbf{n}$. Vzhledem k důležitosti přesnosti určení $\mathbf{n}$ získáme další body reprezentující zem pomocí algoritmu 3, čímž zvýšíme počet bodů, o kterých víme, že se nachází na zemi, a snížíme tak vliv šumu kamery v jednotlivých bodech. Získané body opět pomocí PCA proložíme rovinou a dostaneme přesnější aproximaci normálového vektoru země, viz. sekce 7. Na obrázku 5.1 lze pozorovat výsledek algoritmu 3.

Nyní vypočteme výšku každého bodu nad rovinou a prahováním této vzdálenosti přidělíme bodu hodnotu 0 až k, kde číslo reprezentuje očekávaný počet na sobě naskládaných cihel a k maximální počet cihel, jenž může být na sebe naskládán. Vizualizace tohoto výsledku lze pozorvat na obrázku 5.2

5.1.2 Detekce normálového vektoru pomocí RANSAC

Body jsou postupně prokládány rovinou pomocí RANSAC algoritmu, viz. 2.7. Zde je potřeba správně určit kritérium, při jehož splnění budeme mít jistotu, že detekovaná plocha představuje zem. Pokud by algoritmus skončil po detekci plochy, v jejimž okolí leží nejvíce bodů, mohli bychom při velkém zastoupení kostek v obrazu dostat rovinu popisující horní stěnu těchto kostek. Jako terminační kritérium je tedy zvolena detekce dvou rovin a jejich následným porovnáním vybereme tu, která popisuje zemi.

[2]Osa z směřuje směrem z kamery a body nacházející se nad zemí mají tedy menší hodnotu z.

Algorithm 3 Expandování bodů

```
Stack ← ∅
G ← ∅
for a in M do
  Stack.push((a))
  z ← height of a
  while not Stack.empt() do
    t ← Stack.pop()
    G ← G ∪ t
    V ← V ∪ p
    for n in neighbours of t do
      zn ← height of n
      if zn ∈ [z − δ, z + δ] and not n ∈ G then
        stack.push(n)
      end if
    end for
  end while
end for
```

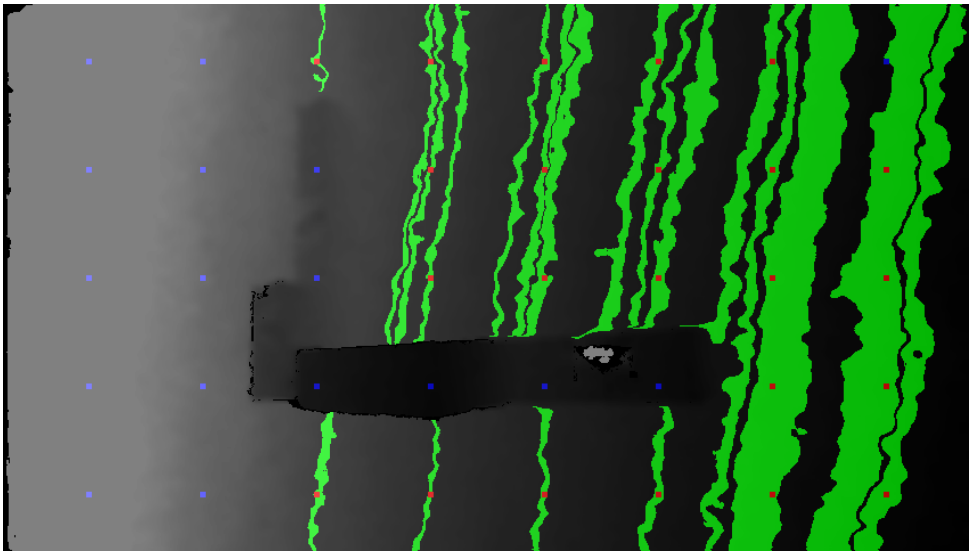
5.1.3 Přístup založený na změně výšky

Velice populární postup při segmentaci obrazu je založen na porovnávání normálového vektoru v bodech obrazu, viz sekce 4.1. Normálový vektor je vypočten pro každý bod obrazu a následně jsou body, jejichž normálový vektor má podobný směr, popřípadě i velikost, sloučeny do větších celků. V našem případě je vrchní stěna cihly paralelní se zemí a má tedy stejný normálový vektor. Nachází se však v jiné výšce. Pomocí výpočtu gradientu, který budeme realizovat užitím Sobelova operátoru aplikovaného na výšku bodů, spočítáme derivaci výšky. Ta by měla být nejvyšší v oblasti přechodu země - cihla. Následně pak sloučíme body s hodnotou gradientu výšky, která se liší v rámci skupiny maximálně o δ. K tomu bylo využito upraveného Connected-component labeling (CCL) algoritmu.

Tato metoda při správném odladění parametru δ pro danou scénu dosahuje velmi dobrých výsledků. Problémem je však robustnost. Hodnoty parametru δ fungující na cihly v blízkosti kamery nefungují na cihly ve vzdálenosti řádově $2\,m$ a více a naopak. Toto je způsobeno jednak šumem kamery, který podle rovnice 3.10 roste kvadraticky, a zejména pak zkreslením vzdálenější hrany cihly kamerou, viz. obrázek 5.3. Kamera zde špatně nachází stereo páry a generuje mračna bodů, která ve skutečnosti neexistují. Ty zmírňují přechod cihla-zem a snižují tak hodnotu gradientu.

5.2 Detekce objektů ze segmentovaného obrazu

Stejně jako při segmentaci obrazu, tak i zde bylo vyzkoušeno několik algoritmů. Některé z nich byly nastaveny na detekci za zjednodušujících podmínek a



Obrázek 5.1: Expandování bodů odpovídajících zemi. Červenou barvou jsou znázorněny body, ze kterých probíhal algoritmus 3. Modré body byly podle výše uvedených kritérií vyřazeny a zeleně jsou znázorněny výsledné body reprezentující zem.

nabízí tak nižší výpočetní náročnost při stejné přesnosti detekce. Vstupem všech prezentovaných algoritmů bude segmentovaný obraz X, jako je například obrázek 5.2. Tedy pro každý bod obrazu $\mathbf{p}$ známe jeho atribut $L(\mathbf{p})$, který reprezentuje počet na sobě naskládaných cihel v bodě $\mathbf{p}$.

5.2.1 Detekce pomocí nejmenšího ohraničujícího obdélníku

V tomto algoritmu předpokládáme, že cihly se navzájem nedotýkají, popřípadě se dotýkají pouze po celé délce navzájem si odpovídajících stěn.

Principem tohoto algoritmu je transformace jednotlivých vrstev tak, aby jejich půdorys nebyl zkreslený prespektivní transformací. Následně je z půdorysu vrstvy určena poloha kostky v dané vrstvě.

Chceme-li určit půdorys n-té vrstvy cihel, pak je k tomu potřeba znát půdorysy vrstev $n, ..., m$, kde m je nejvyšší vrstva cihel v obrazu. Obraz X je nejprve prahován podle L, viz sekce 4.2, kde spodním limitem prahování je n. Na prahovaný obraz je použita morfologická operace otevření, viz sekce 2.6. Pomocí této operace je obraz vyfiltrován a jsou odstraněny body představující šum, viz obrázek 5.4. Následně jsou nalezeny vnější obrysy shluků cihel[3] pomocí Suzukiho algoritmu [42], kde jsme využili již implementovaného programu v knihovně OpenCV [43].

Pomocí průměrového filtru jsou vyhlazena data reprezentujících obrys shluku cihel, resp. jejich výška. Poté je obrys aproximován konvexním obalem, čímž se zmenší počet bodů, kterými je daný shluk popisován.

Obraz horní stěny kvádru nepodléhá prespektivnímu zkreslení, je-li horní

[3] Shlukem cihel se myslí skupina cihel, které jsou spojeny dotekem.

Obrázek 5.2: Vzdálenost bodů od roviny zobrazena jako počet na sobě lžících cihel

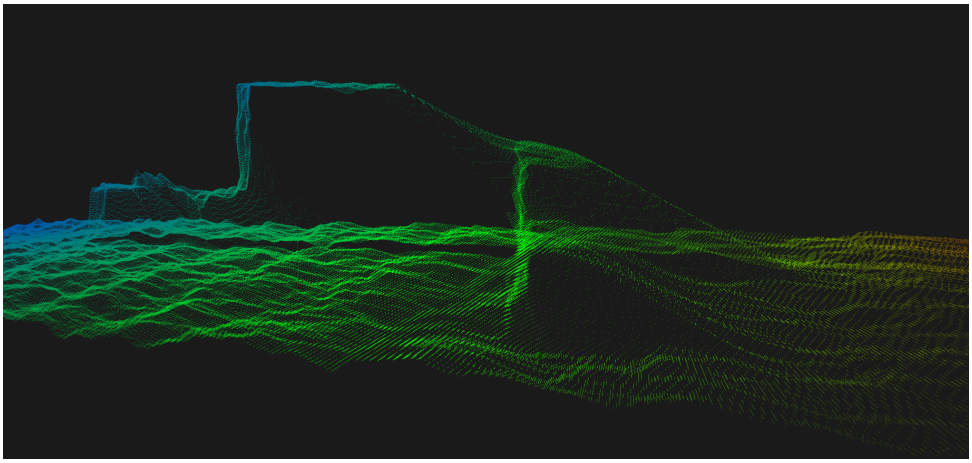
stěna kolmá na optickou osu kamery. Tedy normálový vektor reprezentující horní stěnu kvádru musí být rovnoběžný s optickou osou kamery. Normálový vektor horní stěny kvádru ležícího na zemi je stejný s normálovým vektorem $\mathbf{n}$ země, který byl určen v sekci 5.1. Natočení země vůči kameře je vnější parametr kamery. Body obrazu můžeme tedy transformovat do jejich projekce na rovinu země pomocí upraveného vzorce 3.5 jako

$$\mathbf{x} = \begin{bmatrix} \mathbf{I} \\ 0 \end{bmatrix} \mathbf{K}^{-1}\mathbf{R}^{-1}\mathbf{x}_p, \tag{5.6}$$

kde $\mathbf{R}$ je matice určená pomocí Rodriguezova vzorce 2.7. Úhel θ mezi optickou osou a normálovým vektorem byl určen pomocí vztahu 2.9 a vektor $\tilde{\mathbf{u}}$, kolem které rotace probíhá, pomocí vztahu 2.8. Po provedení této transformace dostaneme body, které představují konvexní obal, zbavené prespektivního zkreslení.

Následně je nalezen nejmenší obdélník, který opisuje tyto body, viz sekce 2.5. Tímto dostaneme řešení ve světovém souřadném systému. Obdélník je plně popsán svými vrcholy. Na tyto vrcholy aplikujeme transformaci 3.5, kde matice $\mathbf{R}$ je transpnovaná matice $\mathbf{R}$ určena výše. Dostaneme body v obraze, které odpovídají ohraničujícímu obdélníku po aplikování prespektivního zkreslení. Jelikož výška h každé cihly je stejná a ze zadání známá, je možné nalezené vrcholy posunout o h a získat tak zbylé vrcholy cihly. Tímto způsobem bude ohraničující kvádr obsahovat i prostorovou informaci.

Výše uvedený postup je proveden pro každý obrys odpovídající skupině kostek a následně pro každé patro. Výsledek algoritmu je vidět na obrázku 5.5.



Obrázek 5.3: Výstup kamery - mračno bodů zobrazující skupinu cihel. V pravé části je vidět zkreslení hrany způsobující problém při segmentaci

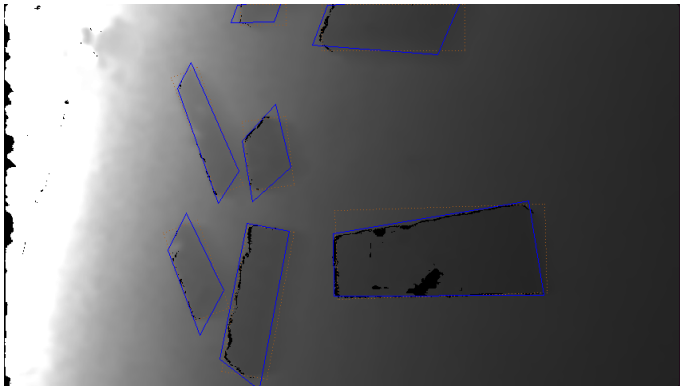
(a) : Obraz před filtrací **(b) :** Obraz po filtraci

Obrázek 5.4: Filtrování obrazu pomocí morgologické oprace otevření

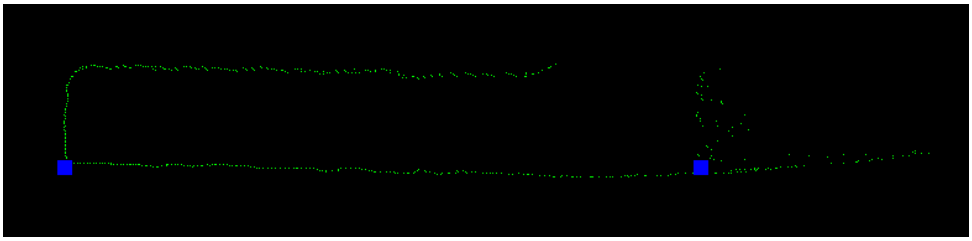
5.2.2 Detekce pomocí RANSAC

Postup popsaný v sekci 5.2.1 funguje pouze za zjednodušujících předpokladů a navíc jeho přesnost značně klesá při nesprávném generování stereo párů, jak je vidět na obrázku 5.3. Tato metoda funguje pro všechna natočení cihel a je i velice robustní.

Začneme obdobně jako v sekci 5.2.1 transformací bodů představující obrys pomocí vztahu 5.6. Mezi body představující prespektivně nezkreslený obrys najdeme pomocí RANSAC přímku p_1. Ta odpovídá jedné z delších stěn cihly, viz. obrázek 5.7. Následně je opět použit algoritmus RANSAC a mezi zbylými body je nalezena přímka p_2 rovnoběžná s p_1. Tímto dostaneme obě protější stěny cihly. Hledání zbylých dvou kratších stěn je pomocí RANSAC algoritmu nerealizovatelné, jelikož tyto stěny jsou tvořeny malým počtem bodů. Pokud je obraz zašuměn, pak není možné najít přímky odpovídající těmto krátkým stěnám.



Obrázek 5.5: Příklad detekce cihel z hloubkových dat. Modře ohraničující obdélník po odstranění prespektivního zkreslení, hnědě ohraničující obdélník u kterého není bráno v potaz prespektivní zkreslení



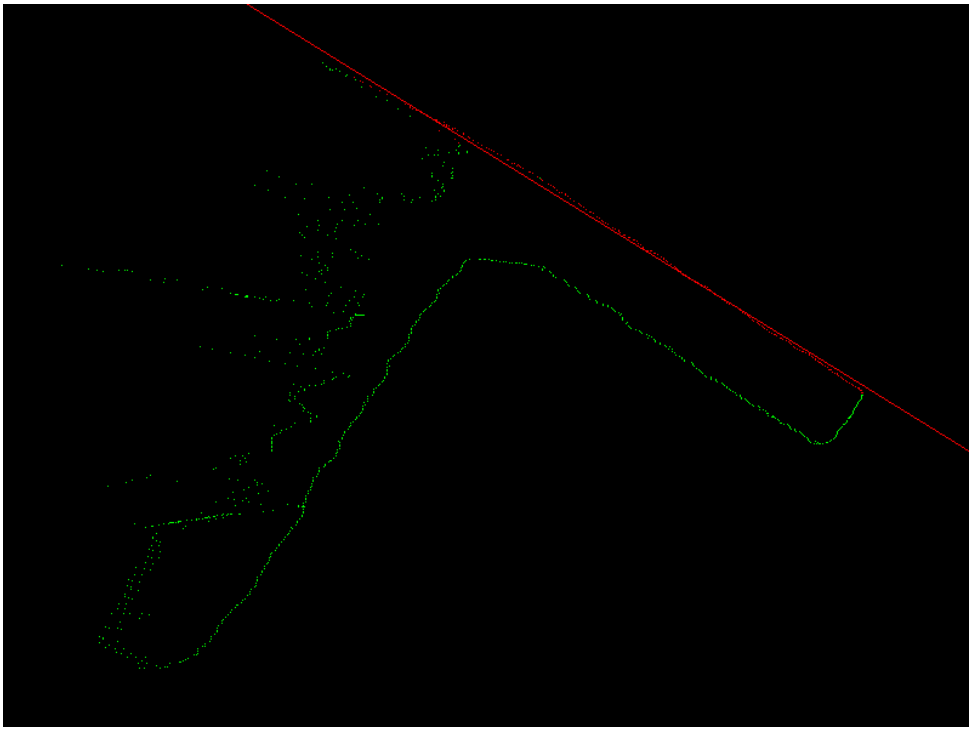
Obrázek 5.6: Detekce kratších hran cihly.

Kratší stěny cihly tedy určíme následovně. Vybereme množinu bodů, které leží mezi přímkami p_1 a p_2. Poté tyto body promítneme na přímku p_1. Výpočet numericky zjednodušíme otočením bodů a přímek o θ, což je úhel sevřený přímkou p_1 a osou x. Projekce na přímku p_1 se tedy stane vyčtením x-ové souřadnice každého bodu.

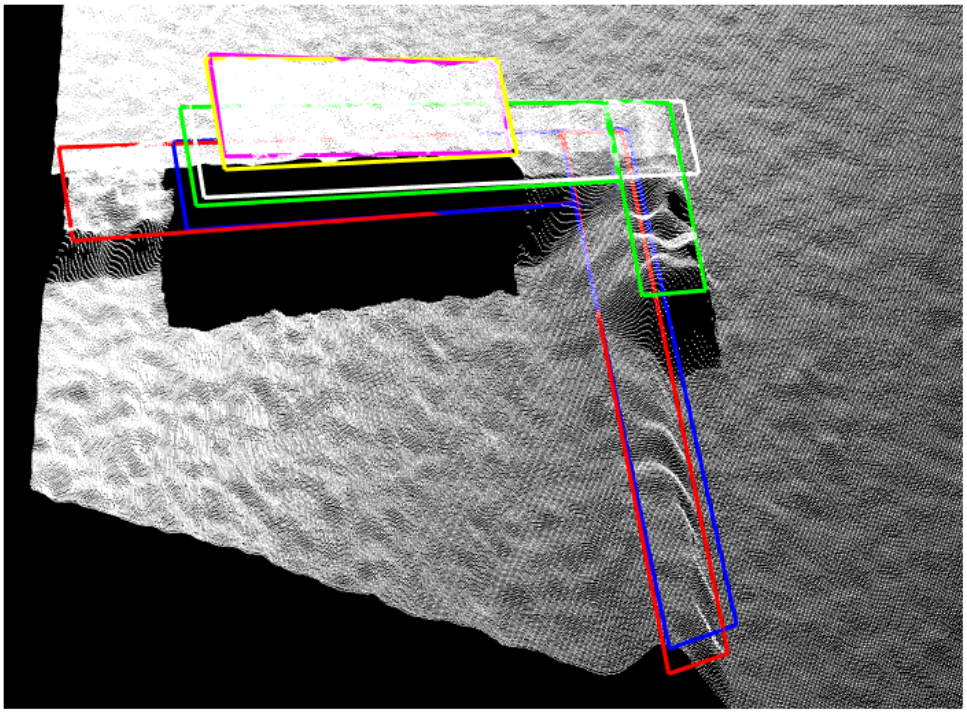
Následně je pro každý bod $\mathbf{x} \in p_1$ určen počet bodů promítnutých do okolí o velikosti ϵ a jsou vybrány dva body s nejvyšší hodnotou. Tyto body jsou ilustrovány na obrázku 5.6 modrou barvou. Těmito body prochází přímky p_3 a p_4, které jsou kolmé na p_1, popřípadě p_2, a představují zbylé dvě stěny hledaného obdélníku. Vrcholy obdélníku r pak určíme jako průsečíky přímek p.

Nyní se z bodů představující obrys shluku cihel odečtou ty, které popisují obdélník r. Pokud počet zbývajících bodů obrysu je větší než předem určený limit, pak se proces počínaje hledáním přímky p_1 opakuje.

Ve chvíli kdy jsou nalezeny všechny obdélníky a nezbývají již žádné volné body, jsou tyto obdélníky, stejně jako v sekci 5.2.1, pomocí svých vrcholů transformovány zpět do obrazu a celý postup se opakuje pro další shluky a vrstvy cihel.



Obrázek 5.7: Detekce delší hrany cihly pomocí RANSAC algoritmu



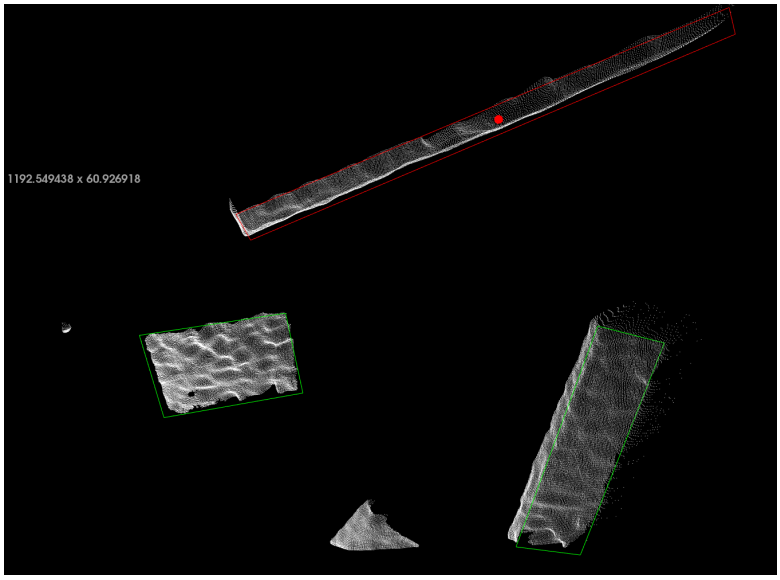
Obrázek 5.8: Výsledek detekce cihel zanesený v mračně bodů. Červenou, bílou a růžovou barvou jsou znázorněny manuálně zanesené polohy cihel.

Kapitola 6

Výsledky

6.1 Označení polohy cihel v mračnech bodů

Za účelem vyhodnocení přesnosti algoritmů prezentovaných v sekci 5, jsme ručně zanesli polohu jednotlivých cihel do mračen bodů. Vzhledem k nedostupnosti nekomerčních softwarů umožňujících manuální označování v mračnech bodů jsme vytvořili vlastní aplikaci v prostředí Point Cloud Library (PCL) [44]. Ta pro každé mračno bodů kombinací RANSAC a PCA algoritmu přesně určí rovinu reprezentující zem. Následně jsou odstraněny body reprezentující zem a do zbylých bodů jsme ručně zanesli polohu první vrstvy cihel. Tento postup se opakuje pro každé patro zdi, jak je možné vidět na obrázku 6.1. Tímto způsobem jsme označili 20 náhodně vybraných obrázků obsahujících nedotýkající se cihly a 15 obrázků, na kterých cihly tvoří složitější struktury. Poloha všech cihel je nakonec uložena do textového souboru, kde je každá cihla reprezentována čtyřmi body v euklidovském prostoru a číslem označujícím vrstvu, ve které se nachází.



Obrázek 6.1: Prostředí aplikace pro označení pozice cihel

6.2 Vyhodnocovací kritéria

Úspěšnost detekce je hodnocena podle časové náročnosti daného programu a přesnosti detekce jednotlivých cihel. Při vyhodnocení přesnosti detekce hodnotíme pouze přesnost umístění horní stěny cihly, jelikož ostatní parametry je možné dopočítat ze znalosti normálového vektoru země.

Nechť C je mračno bodů, pro které bylo manuálně zaznamenáno m poloh obdélníků L_i, které reprezuntují polohu cihel. Pokud náš program na detekci cihel vrátí množinu obdélníků $S_1 ... S_n$, pak se přesnost $p(C)$ určí následovně.

$$A(L) = \sum_{i=1}^{i=m} |\square L_i| \quad A(S) = \sum_{i=1}^{i=n} |\square S_i|, \quad I = \sum_{i=1}^{i=n} \sum_{j=1}^{j=m} |\square S_i \cup \square L_j| \tag{6.1}$$

$$\mathrm{TP} = I, \quad \mathrm{FP} = A(S) - I, \quad \mathrm{FN} = A(L) - I \tag{6.2}$$

$$p = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN} + \mathrm{FP}} \tag{6.3}$$

V rovnici 6.1 jsme symbolem $|\square L_i|$ označili plochu určenou obdélníkem L_i. Časová náročnost programu je určena měřením času běhu programu na jednom jádře procesoru Intel® Core™ i5-7200U, který pracuje s frekvencí 2,5 GHz v normálním módu a s frekvencí 3,1 GHz v turbo módu.

6.3 Určení normálového vektoru

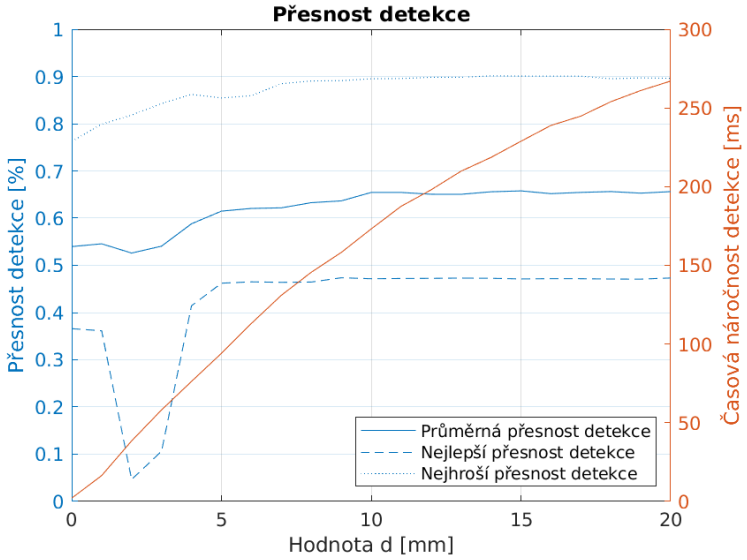
Přesnost určení normálového vektoru jsme vyhodnotili na základě přesnosti detekce objektů, jelikož tato detekce silně závisí na přesnosti určení normálového vektoru. Jedná se navíc o jediné relevantní kritérium, podle kterého můžeme přesnost určit.

Výsledky algoritmu prezentovaného v sekci 5.1.3 nebudeme vyhodnocovat, jelikož nebylo možné nastavit parametry algoritmu tak, aby detekoval normálový vektor země při různé vzdálenosti kamery od obrazu.

Přesnost a doba průběhu programu, který byl popsán v sekci 5.1.1, záleží zejména na velikosti parametru δ, který určuje množství bodů, kterými bude v druhém kroku programu proložena rovina algoritmem PCA. Závislost přesnosti na časové náročnosti programu můžeme pozorovat na grafu 6.2, kde detekce byla provedena algoritmem popsaným v sekci 5.2.1. RANSAC algoritmus 5.2.2 vykazoval podobnou tendenci přesnosti detekce v závislosti na parametru δ a není tedy uváděn.

Chování algoritmu 5.1.2 závisí především na dvou parametrech a to maximálním počtu iterací n a vzdálenosti ϵ od modelu, ve které se musí nacházet bod, aby byl uvažován součástí modelu. Závislost přesnosti a časové náročnosti na ϵ je vidět na grafu 6.3, kde byl algoritmus testován při neomezeném počtu iterací. Můžeme pozorovat, že algoritmus dosahuje nejlepších výsledků při $\epsilon = 60\,mm - 100\,mm$.

Pro hodnotu parametru $\epsilon = 80\,mm$ jsme provedli test závislosti přesnosti na maximálním počtu iterací. Výsledky na obrázku 6.6 jsou průměrem pěti



Obrázek 6.2: Přesnost detekce v závislosti na parametru δ

průběhů testu, jelikož RANSAC je nedeterministický algoritmus a jeho výsledky se v každém běhu mohou výrazně lišit. Z grafu je vidět, že maximální přesnost je vždy dosažena nejpozději po 12-ti iteracích. Časová náročnost stále stoupá, jelikož jsme využili již implementovaného RANSAC algoritmu v knihovně PCL [44], který byl lépe optimalizovan než naše implementace. Nenabízí však možnost ukončení algoritmu, pokud pro daný počet bodů $\mathbf{p}_i$ platí $|\mathbf{p}_i - \mathbf{m}| < \epsilon$, kde $\mathbf{m}$ je bod modelu, který je nejblíže bodu $\mathbf{p}_i$.

Po detekci normálového vektoru následuje sémantická segmentace obrazu. Tato segmentace probíhá pro všechny metody detekce prahováním vzdálenosti jednotlivých bodů od nelezené plochy reprezuntující zem. Jedná se o deterministický algoritmus, který proběhne v čase 62 ms.

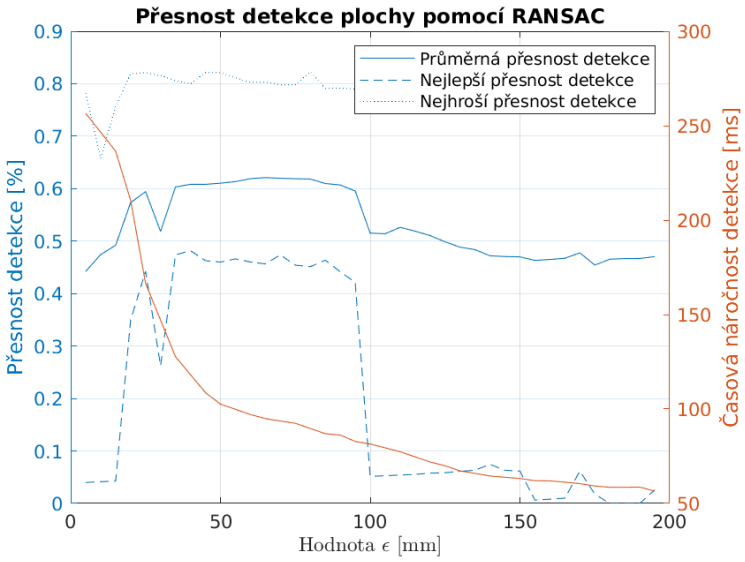
6.4 Detekce objektů

6.4.1 Nedotýkající se cihly

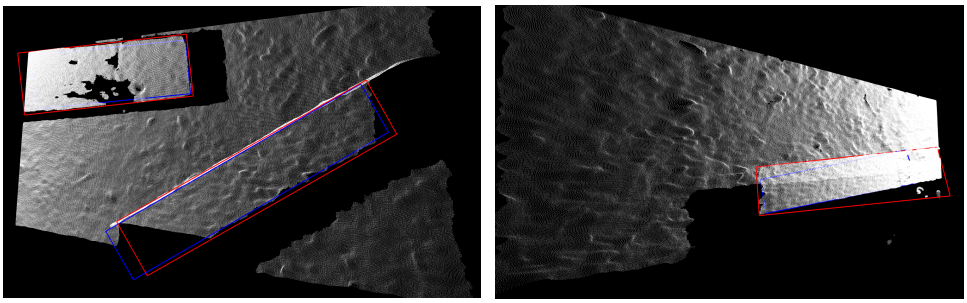
V této sekci budou všechny programy testovány na datasetu, kde se jednotlivé cihly vzájemně nedotýkají.

Přesnost algoritmu, který byl popsáného v sekci 5.2.1 je vidět na grafech 6.2, 6.3, 6.6. Doba průběhu algoritmu závisí na počtu cihel, které se nachází ve scéně. Průměrně je detekce provedena v čase 872 ns a nejpomalejší detekce v testovaném datasetu byla provedena za čas 978 ns.

Algoritmus pro detekci založený na bázi RANSAC algoritmu popsaný v sekci 5.2.2 si dynamicky určuje maximální počet iterací v závislosti na počtu bodů, které tvoří obrys skupiny kostek. Chování tohoto algoritmu je tedy určeno parametrem ϵ, jehož význam je stejný jako v sekci 6.3. Přesnost detekce a časová náročnost v závislosti na ϵ je vidět na obrázku 6.5. Normálový vektor



Obrázek 6.3: Přesnost RANSAC algoritmu pro detekci plochy v závislosti na ϵ



(a) : Nejlepší detekce v datasetu **(b) :** Nejhroší detekce v datasetu

Obrázek 6.4: Ilustrace výsledků detekce cihel

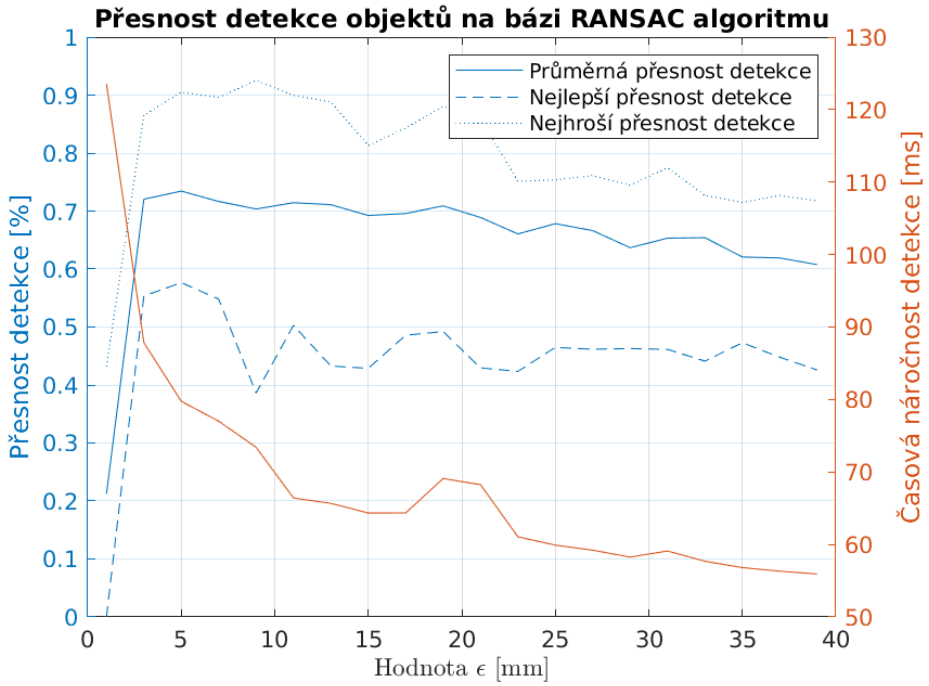
země, který je pro detekci potřeba, byl určen pomocí modifikovaného PCA algoritmu, který byl popsan v sekci 5.1.1.

Výsledky detekce můžeme pozorovat na obrázku 6.4, kde modrou barvou je značen výsledek programu a červeně manuálně zanesená poloha cihel. Z obrázku 6.4a je vidět, že detekce probíhá velice přesně a chyba algoritmu může být způsobena nepřesností při označování skutečných poloh cihel. Dokonce i nejhorší detekce v celém datasetu, která je na obrázku 6.4b, správně nalezne nacházející se cihlu, pouze špatně odhadne její natočení a velikost.

6.4.2 Dotýkající se cihly

Jediný algoritmus schopný detekce dotýkajících se cihel je popsán v sekci 5.2.2. Jeho výsledky na datasetu s nedotýkajícími se cihlami lze pozorovat na obrázku 6.5. Pro parametr $\epsilon = 12\,mm$ dostaneme na datasetu, který obsahuje pouze složitější struktury cihel, následující výsledky.

- Průměrná úspěšnost detekce $68,24\,\%$



Obrázek 6.5:

- Nejlépe klasifikované mračno bodů $84,3\,\%$
- Nejhůře klasifikované mračno bodů $39,7\,\%$
- Průměrná časová náročnost $78,8\,ms$

Algoritmus tedy dosahuje podobných výsledků, jako při detekci nedotýkajících se cihel, pouze v horším časem. Příklad úspěšné detekce je vidět na obrázku 5.8.

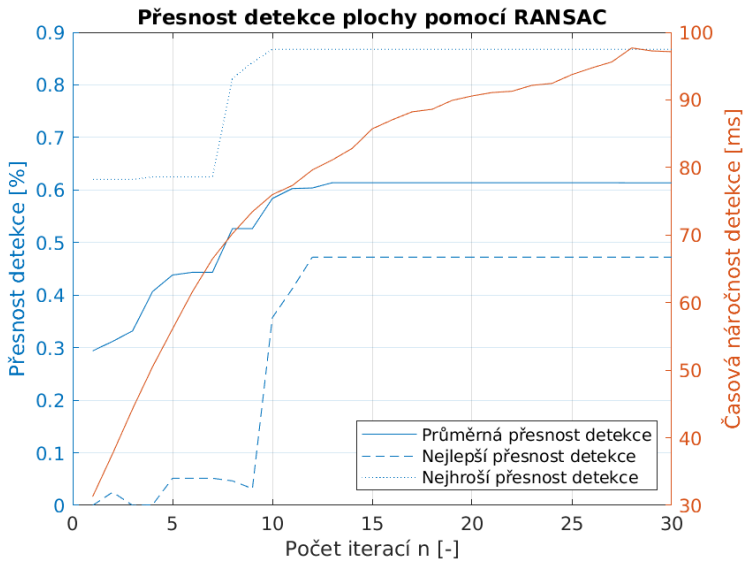
6.5 Shrnutí výsledků

V této sekci srovnáme výsledky kombinací jednotlivých algoritmů. Součástí každého z nich je prahování bodů podle vzdálenosti od roviny, které trvá $62\,ms$.

V tabulce 6.1 jsou použity následující zkratky RANSAC - detekce plochy pomocí ransac algoritmu, kde *li* a *hi* značí nízký a vysoký počet maximálních iterací. Jako PCA je označen algoritmus pro detekci plochy pospaný v sekci 5.1.1, kde m a v značí nízkou a vysokou hodnoty parametru d. Označením RANSAC-det je myšlen algoritmus popsaný v sekci 5.2.2.

algoritmus	přesnost [%]	čas [ms]
RANSAC-*li* a Nejmenší ohraničující odbélník	43,2	108
RANSAC-*hi* a Nejmenší ohraničující obdélník	61,3	143
PCA-*n* a Nejmenší ohraničující obdélníku	57,5	148
PCA-*v* a Nejmenší ohraničující obdélník	65,2	243
RANSAC-*li* a RANSAC-det	58,4	182
RANSAC-*hi* a RANSAC-det	68,7	217
PCA-*n* a RANSAC-det	63,1	223
PCA-*v* a RANSAC-det	70,8	328
PCA-*v* a RANSAC-det - složité objekty	68,24	341

Tabulka 6.1: Porovnání výsledků jednotlivých algoritmů



Obrázek 6.6: Přesnost RANSAC algoritmu pro detekci plochy v závislosti na n

Kapitola 7

Závěr

V této práci jsme se seznámili s principem funkce stereokamery a metodami detekce z hloubkových dat. Těchto znalostí jsme následně využili k navržení vlastních postupů pro detekci na zemi ležících cihel. Vstupními informacemi našeho programu byla data ze stereokamery Intel® RealSense^{TM}. Následně byly tyto postupy implementovány v jazyce C++ s využitím knihoven OpenCV [43] a PCL [44].

Nejpřesnější z námi prezentovaných postupů je schopn detekovat objekty s úspěšností 70,8 %. Je však omezen na bloky cihel, mezi kterými nedochází k vzájemnému doteku, popřípadě se bloky mohou dotýkat po celé délce odpovídající stěny.

Tento program byl dále rozšířen, aby byl schopen detekce cihel nacházejících se ve složitějších strukturách, jako je třeba zeď tvořící roh, viz obrázek 5.8. Tyto ztěžující podmínky mají pouze minimální vliv na výsledky programu. Detekce je provedena se ztrátou přesnosti 2,56 % a pouze o 13 ms pomaleji.

Prezentované výsledky byly získány na datasetu, který byl pořízen člověkem držícím Intel® RealSense^{TM} kameru. Toto zapříčinilo menší úhel mezi rovinou země a optickou osou kamery. Kvůli tomu je na všech datech pozorovatelné velké prespektivní zkreslení. Předpokládáme, že při použití RealSense na dronu bude sníženo prespektivní zkreslení obrazu a přesnost detekce se tak výrazně zvýší.

Detekcí objektů podobných tvarů z RGB-D dat se zabýval Jia at al. 2013 [36]. Jejich program dosahoval v závislosti na použitém datasetu přesnosti 61,7%-70%. Tento program však nepracuje v reálném čase a může tedy využít časově náročné iterační metody pro segmentaci. Oproti námi navrženému programu však detekované objekty nemají předem známé tvary.

Omezujícím faktorem našeho programu je rychlost, kdy i při snížené přesnosti detekce na 68,7 % jsme schopni zpracovat pouze 5 snímků za sekundu při použitém rozlišení 848×480 pixelů. Oproti tomu používaná kamera Intel® Realsense^{TM} pracuje při tomto rozlišení s rychlostí 30 snímků za sekundu. Tento problém by mohl být vyřešen použitím výkonějšího procesoru, popřípadě paralelizací programu na více jader procesoru. Další možností zrychlení programu je použití IMU senzorů, čímž bychom získali informaci o natočení roviny, na které se nachází cihly.

Obdobnou časovou náročnost má i přístup prezentováný v Holz et al. 2011

[26], kde jejich program dosahuje až 30 snímků za sekundu. Tato rychlost je podmíněna snížením rozlišení obrazu na úroveň 160×120 pixelů. Při použití rozlišení 640×480 pixelů rychlost prgramu klesne na 7 snímků za sekundu. K testování časové náročnost použil Holz et al. 2011 [26], stejně jako my, jedno jádro procesoru Intel®. Na rozdíl od našeho programu není však výstupem v Holz et al. 2011 [26] přesný seznam poloh, ale obraz segmentovaný na jednotlivé instance.

Zvýšené přesnosti detekce a nižší časové náročnosti námi prezentovaného programu by bylo možné dosáhnout, pokud by byla použita stereokamera, která dosahuje přesnějšího snímání hloubkových dat než námi použitý Intel® Realsense^{TM} D435. Výstup z kamery Intel bývá často nepřesný a to zejména při prudké změně hloubky bodů (viz obrázek 5.3), což značně zvyšuje náročnost přesné detekce. RealSense navíc generuje velké množství bodů a tím zvyšuje časovou náročnost zpracování těchto dat. Pouhý výpočet lineární rovnice pro každý bod a následné prahování trvalo na námi použitém procesoru 62 ms. Ideální by bylo použití LIDARu, který většinou generuje menší počet bodů. Tyto body však bývají daleko přesnější.

Příloha A

Literatura

[1] Drone market outlook: industry growth trends, market stats and forecast. `https://www.businessinsider.com/drone-industry-analysis-market-trends-growth-forecasts`. [cit. 2020-05-19].

[2] Serge Weisstein Eric W., Belonie. Rodrigues' rotation formula. `https://mathworld.wolfram.com/RodriguesRotationFormula.html`. [cit. 2020-04-08].

[3] Walker James W. Angle Between Vectors. `https://www.jwwalker.com/pages/angle-between-vectors.html`, nov 2014. [cit. 2020-04-09].

[4] Wenshuo Gao, Xiaoguang Zhang, Lei Yang, and Huizhong Liu. An improved sobel edge detection. In *2010 3rd international conference on computer science and information technology*, volume 5, pages 67–71. IEEE, 2010.

[5] Zhang Jin-Yu, Chen Yan, and Huang Xian-Xiang. Edge detection of images based on improved sobel operator and genetic algorithms. In *2009 International Conference on Image Analysis and Signal Processing*, pages 31–35. IEEE, 2009.

[6] Timothy M Chan. Optimal output-sensitive convex hull algorithms in two and three dimensions. *Discrete & Computational Geometry*, 16(4):361–368, 1996.

[7] David Pichardie and Yves Bertot. Formalizing convex hull algorithms. In *International Conference on Theorem Proving in Higher Order Logics*, pages 346–361. Springer, 2001.

[8] R. V. Chadnov and A. V. Skvortsov. Convex hull algorithms review. In *Proceedings. The 8th Russian-Korean International Symposium on Science and Technology, 2004. KORUS 2004.*, volume 2, pages 112–115 vol. 2, 2004.

[9] Godfried T Toussaint. Complexity, convexity, and unimodality. *International journal of computer & information sciences*, 13(3):197–217, 1984.

[10] Robert M Haralick, Stanley R Sternberg, and Xinhua Zhuang. Image analysis using mathematical morphology. *IEEE transactions on pattern analysis and machine intelligence*, (4):532–550, 1987.

[11] Mary L Comer and Edward J Delp. Morphological operations for color image processing. *J. Electronic Imaging*, 8(3):279–289, 1999.

[12] Konstantinos G Derpanis. Overview of the ransac algorithm. *Image Rochester NY*, 4(1):2–3, 2010.

[13] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[14] Myron Z Brown, Darius Burschka, and Gregory D Hager. Advances in computational stereo. *IEEE transactions on pattern analysis and machine intelligence*, 25(8):993–1008, 2003.

[15] Leonid Keselman, John Iselin Woodfill, Anders Grunnet-Jepsen, and Achintya Bhowmik. Intel realsense stereoscopic depth cameras. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2017.

[16] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004.

[17] Stereo and 3d vision. Dostupné z:`https://courses.cs.washington.edu/courses/cse455/09wi/Lects/lect16.pdf`. [cit. 2020-04-09].

[18] Epipolar geometry. `http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/EPSRC_SSAZ/node18.html`. [cit. 2020-05-19].

[19] Annika Kuhl. Comparison of stereo matching algorithms for mobile robots. *The University of Western Australia Faculty of Engineering, Computing and Mathematics*, 2005.

[20] ©Intel. Depth camera d435. `https://www.intelrealsense.com/depth-camera-d435/`. [cit. 2020-04-08].

[21] Intel® realsense™ d400 series product family. `https://www.intel.com/content/dam/support/us/en/documents/emerging-technologies/intel-realsense-technology/Intel-RealSense-D400-Series-Datasheet.pdf`, 2019. [cit. 2020-04-08].

[22] Orazio Gallo, Roberto Manduchi, and Abbas Rafii. Cc-ransac: Fitting planes in the presence of multiple surfaces in range data. *Pattern Recognition Letters*, 32(3):403–410, 2011.

[23] Panagiotis-Alexandros Bokaris, Damien Muselet, and Alain Trémeau. 3D reconstruction of indoor scenes using a single RGB-D image. In *12th*

International Conference on Computer Vision Theory and Applications (VISAPP 2017), Porto, Portugal, February 2017.

[24] Radu Bogdan Rusu, Nico Blodow, Zoltan Csaba Marton, and Michael Beetz. Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1–6. IEEE, 2009.

[25] Jiefei Wang, Matthew Garratt, and Sreenatha Anavatti. Dominant plane detection using a rgb-d camera for autonomous navigation. In *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, pages 456–460. IEEE, 2015.

[26] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using rgb-d cameras. In *Robot Soccer World Cup*, pages 306–317. Springer, 2011.

[27] Alexander JB Trevor, Suat Gedikli, Radu B Rusu, and Henrik I Christensen. Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)*, 2013.

[28] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. A large-scale hierarchical multi-view rgb-d object dataset. In *2011 IEEE international conference on robotics and automation*, pages 1817–1824. IEEE, 2011.

[29] Tomáš Werner. Optimalizace - elektronickcká skripta předmětu b0b33opt. Dostupné z:`https://cw.fel.cvut.cz/b191/_media/courses/b0b33opt/opt.pdf`, 2020. [cit. 2020-04-09].

[30] Lizhi Zhang, Diansheng Chen, and Weihui Liu. Fast plane segmentation with line primitives for rgb-d sensor. *International Journal of Advanced Robotic Systems*, 13(6):1729881416665846, 2016.

[31] Dilpreet Kaur and Yadwinder Kaur. Various image segmentation techniques: a review. *International Journal of Computer Science and Mobile Computing*, 3(5):809–814, 2014.

[32] Nesrine Chehata, Nicolas David, and Frédéric Bretar. Lidar data classification using hierarchical k-means clustering. In *ISPRS Congress Beijing 2008*, volume 37, pages 325–330. Citeseer, 2008.

[33] Konstantinos G Derpanis. Mean shift clustering. *Lecture Notes*, page 32, 2005.

[34] Marj Tonini and Antonio Abellan. Rockfall detection from terrestrial lidar point clouds: A clustering approach using r. *Journal of Spatial Information Science*, 2014(8):95–110, 2014.

[35] Chia-Tche Chang, Bastien Gorissen, and Samuel Melchior. Fast oriented bounding box optimization on the rotation group so (3, r). *ACM Transactions on Graphics (TOG)*, 30(5):1–16, 2011.

[36] Zhaoyin Jia, Andrew Gallagher, Ashutosh Saxena, and Tsuhan Chen. 3d-based reasoning with blocks, support, and stability. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2013.

[37] Kai Huebner, Steffen Ruthotto, and Danica Kragic. Minimum volume bounding box decomposition for shape approximation in robot grasping. In *2008 IEEE International Conference on Robotics and Automation*, pages 1628–1633. IEEE, 2008.

[38] Sergio Garcia. Fitting primitive shapes to point clouds for robotic grasping. *Master of Science Thesis. School of Computer Science and Communication, Royal Institute of Technology, Stockholm, Sweden*, 2009.

[39] Jun-Sik Kim. Object detection using rgbd data for interactive robotic manipulation. In *2014 11th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 339–343. IEEE, 2014.

[40] Kevin Lai, Liefeng Bo, Xiaofeng Ren, and Dieter Fox. Detection-based object labeling in 3d scenes. In *2012 IEEE International Conference on Robotics and Automation*, pages 1330–1337. IEEE, 2012.

[41] Isaac Ronald Ward, Hamid Laga, and Mohammed Bennamoun. Rgb-d image-based object detection: From traditional methods to deep learning techniques. In *RGB-D Image Analysis and Processing*, pages 169–201. Springer, 2019.

[42] Satoshi Suzuki et al. Topological structural analysis of digitized binary images by border following. *Computer vision, graphics, and image processing*, 30(1):32–46, 1985.

[43] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[44] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

Příloha B

Obsah přiloženého CD