VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ

ÚSTAV MECHANIKY TĚLES, MECHATRONIKY A BIOMECHANIKY

FACULTY OF MECHANICAL ENGINEERING

INSTITUTE OF SOLID MECHANICS, MECHATRONICS AND BIOMECHANICS

3D MAPOVÁNÍ VNITŘNÍHO PROSTŘEDÍ SENZOREM MICROSOFT KINECT

3D INDOOR MAPPING USING MICROSOFT KINECT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. PETR PILCH

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. JIŘÍ KREJSA, Ph.D.

SUPERVISOR

BRNO 2013

Vysoké učení technické v Brně, Fakulta strojního inženýrství

Ústav mechaniky těles, mechatroniky a biomechaniky
Akademický rok: 2012/2013

ZADÁNÍ DIPLOMOVÉ PRÁCE

student(ka): Bc. Petr Pilch

který/která studuje v **magisterském navazujícím studijním programu**

obor: **Mechatronika (3906T001)**

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

3D mapování vnitřního prostředí senzorem Microsoft Kinect

v anglickém jazyce:

3D indoor mapping using Microsoft Kinect

Stručná charakteristika problematiky úkolu:

Senzor Microsoft Kinect je schopný dodat hloubkovou mapu prostředí. Cílem práce je využít série takto získaných hloubkových map k vytvoření celkového 3D modelu prostředí. Při sesazování dílčích hloubkových map se předpokládá neznalost relativního pohybu senzoru. Uvažujte pouze vnitřní prostředí.

Cíle diplomové práce:

1. Rešerše způsobů sesazování dílčích hloubkových map
2. Výběr vhodného algoritmu
3. Implementace (s možným využitím OpenSource knihoven)
4. Ověření na různých typech vnitřního prostředí

Seznam odborné literatury:

R.A.Newcombe et.al.: KinectFusion: Real-Time Dense Surface Mapping and Tracking , ISMAR 2011

Vedoucí diplomové práce: doc. Ing. Jiří Krejsa, Ph.D.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2012/2013.

V Brně, dne 20.11.2012

L.S.

_____	_____
prof. Ing. Jindřich Petruška, CSc.	prof. RNDr. Miroslav Doupovec, CSc., dr. h. c.
Ředitel ústavu	Děkan fakulty

Abstrakt

Tato práce se zabývá vytvářením 3D map vnitřního prostředí pomocí senzoru Microsoft Kinect. V první části je uveden popis senzoru Microsoft Kinect, metody získávání a zpracování hloubkových dat a jejich následné skládání pomocí různých algoritmů. V druhé části práce je ukázáno použití algoritmu pro skládání prostorových map a výsledné 3D mapy vnitřního prostředí.

Summary

This work is focused on creating 3D maps of indoor enviroment using Microsoft Kinect sensor. The first part shows the description of Microsoft Kinect sensor, the methods for acquisition and processing of depth data and their registration using different algorithms. The second part shows application of algorithms for map registration and final 3D maps of indoor enviroment

Klíčová slova

Kinect, 3D mapování, skládání obrazů, Iterative closest point

Keywords

Kinect, 3D mapping, image registration, Iterative closest point

PILCH, P.*3D mapování vnitřního prostředí senzorem Microsoft Kinect*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, 2013. 52 s. Vedoucí doc. Ing. Jiří Krejsa, Ph.D.

Já, Petr Pilch, prohlašuji, že jsem diplomovou práci vypracoval samostatně a že jsem uvedl všechny použité prameny a literaturu.

Bc. Petr Pilch

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce doc. Ing. Jiřímu Krejsovi, Ph.D., za rady a připomínky, které mi udělil při zpracovávání této diplomové práce. Dále bych chtěl poděkovat rodičům a přítelkyni za podporu.

Bc. Petr Pilch

Obsah

1. Úvod

Počítačové vidění je relativně nový a rychle se rozvíjející obor. Obsahuje snímání, zpracování, analýzu a následné vyhodnocení obrazu pomocí počítače. Jeho uplatnění je široké: od dopravy, kde může například pomáhat s řízením auta, přes lékařství, průmysl, armádu až po kosmonautiku, kde se využívá u vesmírných vozítek k mapování okolí a lokalizaci. V nynější době je počítačové vidění čím dál tím více rozšířené. Je to proto, že roste výkon počítačů a výpočetní techniky vůbec, takže je výpočet rychlejší, ale také proto, že klesá cena jednotlivých senzorů, které umožňuji kvalitní snímání okolí. Jedním z těchto senzorů je Microsoft Kinect, který dokáže snímat okolí v dostatečné kvalitě, aby bylo možné provést následné zpracování.

Tato práce je zaměřena na využití senzoru Microsoft Kinect pro vytváření 3D mapy v indoor prostředí. Důvodem pro volbu tohoto senzoru je zejména jeho snadná dostupnost. V poslední době dochází k velkému rozšíření tohoto senzoru, díky čemuž se vytvářejí různé knihovny, které umožňují jednodušší vývoj a používání tohoto senzoru. Důvodem pro měření v indoor prostředí je jeho jednodušší zpracování než ve venkovních otevřených prostorech, a poté zejména omezení hloubkového senzoru u Kinectu, který má dosah okolo šesti metrů, což není dostatečný dosah pro měření velkých venkovních prostorů. Takto vytvořená prostorová mapa může lidem sloužit například k lepší představě o vzhledu daného prostředí, nebo může být použita k lepšímu řízení robota v tomto prostředí.

Cílem této práce je tedy prozkoumat možnosti vytváření 3D map, zjistit, jaké existují algoritmy pro skládání jednotlivých snímků do jednoho, a objasnit problematiku vytváření takovéto mapy. Dalším cílem je výběr vhodného algoritmu, který provede toto skládání, jeho implementace a následné ověření funkčnosti na různých typech vnitřního prostředí.

Část I

Vytváření prostorové mapy

2. Získávání dat

K vytvoření prostorové mapy je potřeba nejprve naměřit data pomocí senzoru, který může zjistit vzdálenost mezi objektem a senzorem. K získávání takovýchto dat existuje několik typů senzorů. Některé z těchto senzorů jsou založeny na principu, že je vyzářen signál, nejčastěji světelný paprsek a čidlo měří odražený signál. Tento způsob měření je ale velmi závislý na povrchu objektu, protože u některých materiálů se vyslaný signál nemusí odrazit zpět. V následujících kapitolách budou představeny některé z nich.

2.1. Time of Flight

Prvním typem senzoru je Time of Flight. Tento typ senzoru využívá známé rychlosti vysílaného signálu. Pro změření vzdálenosti se vyšle signál o známé rychlosti, kterou se signál šíří okolím, a změří se čas, za který se tento odražený signál vrátí. Pro výpočet vzdálenosti se použije rovnice

$$D = \frac{v \cdot t}{2} \tag{2.1}$$

kde D je měřená vzdálenost, v je známá rychlost signálu a t je změřený čas, jak dlouho se signál šířil. Toto je celé poděleno dvěma, protože signál musí urazit vzdálenost tam i zpět. Výhody této metody tedy jsou:

- velká přesnost,
- hustotu naměřených dat lze ovlivnit počtem měření.

Mezi nevýhody této metody patří

- vysoká cena,
- musí být známa rychlost šíření signálu v daném prostředí,
- je závislý na materiálu měřeného objektu,
- může docházet k několikanásobnému odrazu,
- je nutné použít přesné časování.

Tato metoda se nejčastěji používá u laserových snímačů, nebo sonarů. Laserové snímače bývají nejčastěji 2D a třetí dimenze se přidává pomocí mechanické části, která otáčí snímačem.

2.2. Phase shift

Druhý princip měření vzdálenosti se nazývá Phase shift. Tento princip je založený na měření fázového posunu a umožňuje, kromě měření vzdálenosti, i měření rychlosti, kterou se objekt pohybuje vzhledem k senzoru. Celý senzor se skládá z vysílače, který vysílá buď modulované laserové světlo, rádiové vlny, nebo zvuk. Tento signál se po dopadu na objekt odrazí zpět do snímače, který porovnává relativní fázový posun získaného signálu s vysílaným. U senzorů, jež využívají laserový signál se ještě používá heterodyn, který upravuje frekvenci přijímaného signálu, aby snímač mohl pracovat lépe na nižší frekvenci. Pro výpočet vzdálenosti senzoru od objektu se použije rovnice

$$d = \frac{\phi\lambda}{4\pi} = \frac{\phi c}{4\pi f}[5, str.170] \tag{2.2}$$

kde ϕ je změřený fázový posun, λ je vlnová délka signálu, c je rychlost světla a f je modulovaná frekvence signálu.

Výhody této metody jsou:

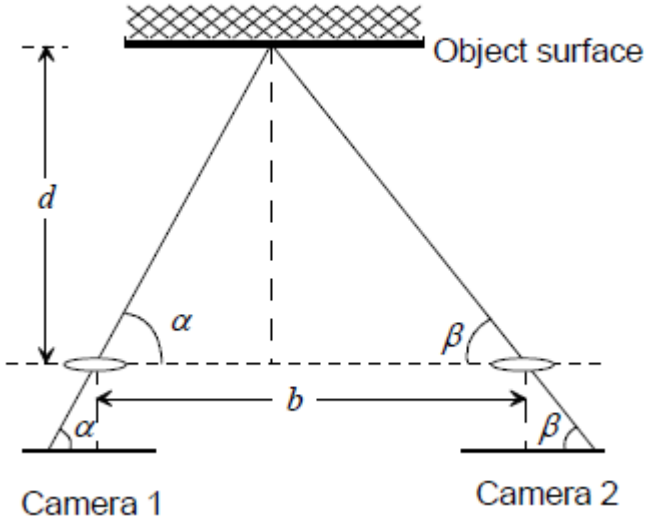
- velká přesnost,
- hustotu dat lze také ovlivnit počtem měření.

Mezi nevýhody této metody patří:

- musí být známa rychlost šíření signálu v daném prostředí,
- je také závislá na materiálu měřeného objektu,
- je nutné použít přesné časování,
- hodí se spíše na měření menších vzdáleností.

2.3. Pasivní triangulace

Třetí metodou, jak zjistit vzdálenost objektu, je pasivní triangulace. Tato metoda využívá dvou kamer, takže je v podstatě stejná, jakou používá člověk a také většina živočichů k odhadu vzdálenosti. Její princip je ukázán na obr. 2.1



Obrázek 2.1: Princip pasivní triangulace[6, str. 5]

Nejprve se musí určit nějaký významný bod, na který se dá zaměřit. To se dělá tak, že se hledá bod, kde je velký gradient intenzity, takže je možné ho lehce lokalizovat i na druhé kameře. K výpočtu vzdálenosti bodu od kamer se použije následující rovnice

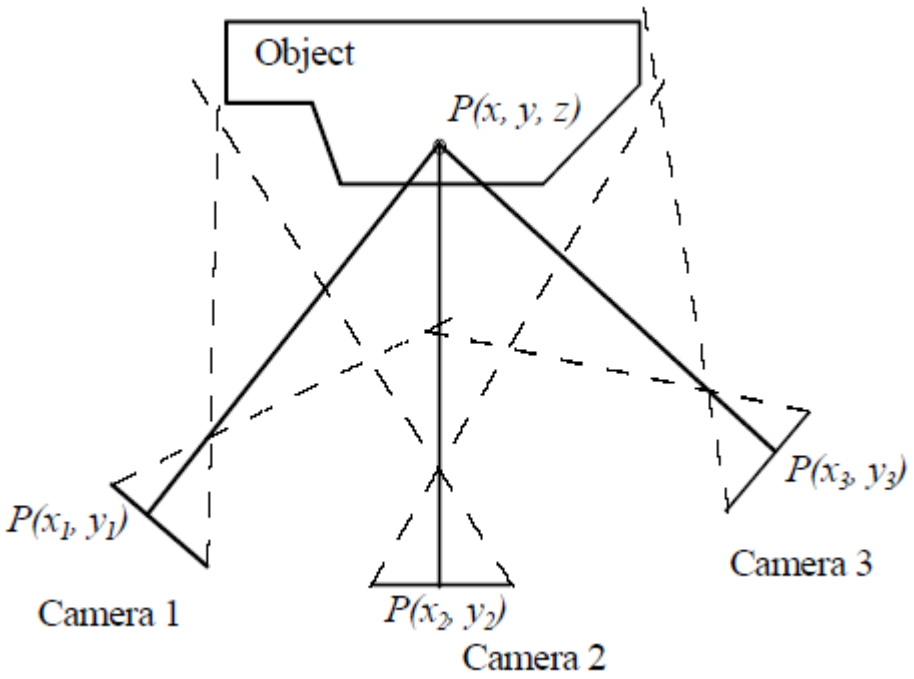
$$d = \frac{b}{\frac{1}{\tan\alpha} + \frac{1}{\tan\beta}}[6, str.5] \tag{2.3}$$

kde d je hledaná vzdálenost, b je vzdálenost mezi oběma kamerami a α a β jsou úhly pod kterým jednotlivé kamery vidí daný bod. Takto je ale nemožné spočítat vzdálenost všech pixelů, protože ne všechny se dají použít jako významný bod. Pro eliminaci tohoto omezení je možno použít například laser, který může vytvářet jeden bod, nebo i celou mřížku. Tento laser pomůže kamerám se přesně zaměřit, díky čemu poté mohou změřit vzdálenost jinak souvislé plochy.

Dalším problémem této metody je, že s rostoucí vzdáleností klesá její přesnost. Toto lze eliminovat tím, že se umístí kamery dále od sebe, takže vzroste přesnost na velké vzdálenosti, ale zvýší se minimální vzdálenost, na kterou můžou měřit. Pro udržení stále stejné přesnosti při měření různě vzdálených bodů není možné mít stále stejnou vzdálenost

mezi jednotlivými kamerami. Takže se nemůžou použít dvě kamery, které mají konstantní vzdálenost od sebe, pro měření různě vzdálených objektů se stejnou přesností.

Pro další zvýšení přesnosti se může použít soustava dvou a více kamerových systémů, jak je ukázáno na obr. 2.2



Obrázek 2.2: Měření pomocí většího počtu kamer [6, str. 5]

Hlavním požadavkem pro měření většího počtu kamerových systémů je znalost přesné polohy jednotlivých systémů vůči ostatním.

Třetím a asi největším nedostatkem této metody je velká závislost na osvětlení. Senzory používané v těchto systémech jsou vetšinou obyčejné kamery a proto vyžadují dostatek světla a při nižší intenzitě osvětlení může být problém určit významné body.

Výhody této metody tedy jsou:

- možnost zjistit vzdálenost u několika bodů zároveň,
- nízká cena.

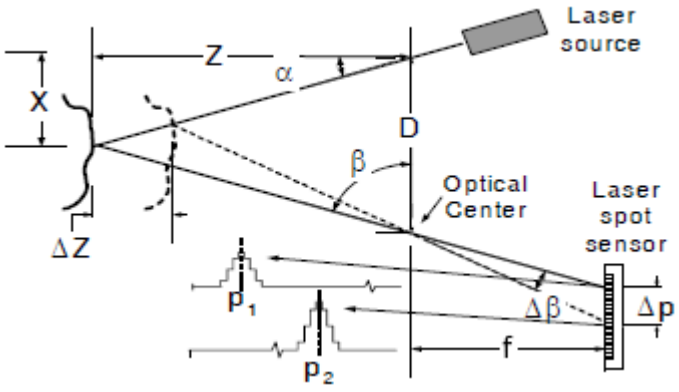
Mezi nevýhody této metody patří:

- velká závislost na intenzitě osvětlení,
- u velmi členitých, nebo souvislých ploch může být obtížné najít významné body,

- nižší přesnost při měření vzdálených objektů.

2.4. Aktivní triangulace

Poslední ze zde představených metod pro získávání informace o vzdálenosti objektu je Aktivní triangulace. Tento princip snímání používá jeden aktivní vysílač a jeden senzor, který snímá odražený signál viz obr. 2.3



Obrázek 2.3: Princip aktivní triangulace[1, str. 1]

Tento vysílač může vysílat signál o určité vlnové délce, viditelné nebo infračervené světlo, nebo také může vysílat světlo o různých vlnových délkách a umožnit tak i zjištění barvy snímaného objektu. Pro výpočet vzdálenosti od objektu měřené pomocí aktivní triangulace se použije rovnice

$$Z = \frac{Df}{p + f\tan\alpha}[1, str.2] \tag{2.4}$$

kde Z je požadovaná vzdálenost, D je známá vzdálenost mezi vysílačem a čočkami snímače, f je vzdálenost mezi senzorem a čočkami, p je pozice na senzoru a α je úhel vysílaného paprsku.

Jelikož tento systém obsahuje vysílač, který osvětluje objekt, netrpí metoda aktivní triangulace problémem nedostatku osvětlení jako pasivní triangulace. Proto se může tato metoda použít i v místech, kde je nedostatek světla. Další výhodou této metody je, že je možno eliminovat další nedostatek předchozí metody, kterým je problém při zaměřování významných bodů. U aktivní triangulace se vytváří mřížka vyslaným, například pomocí infračerveného záření nebo laseru, která se odráží k snímači. Tato mřížka ulehčuje zaměřování a umožňuje mít konstantní vzdálenost mezi jednotlivými měřenými body.

Výhody této metody tedy jsou:

- snadné měření vzdáleností souvislých ploch,
- není závislá na okolním osvětlení,
- umožňuje zjistit vzdálenost více bodů najednou.

Mezi nevýhody této metody patří:

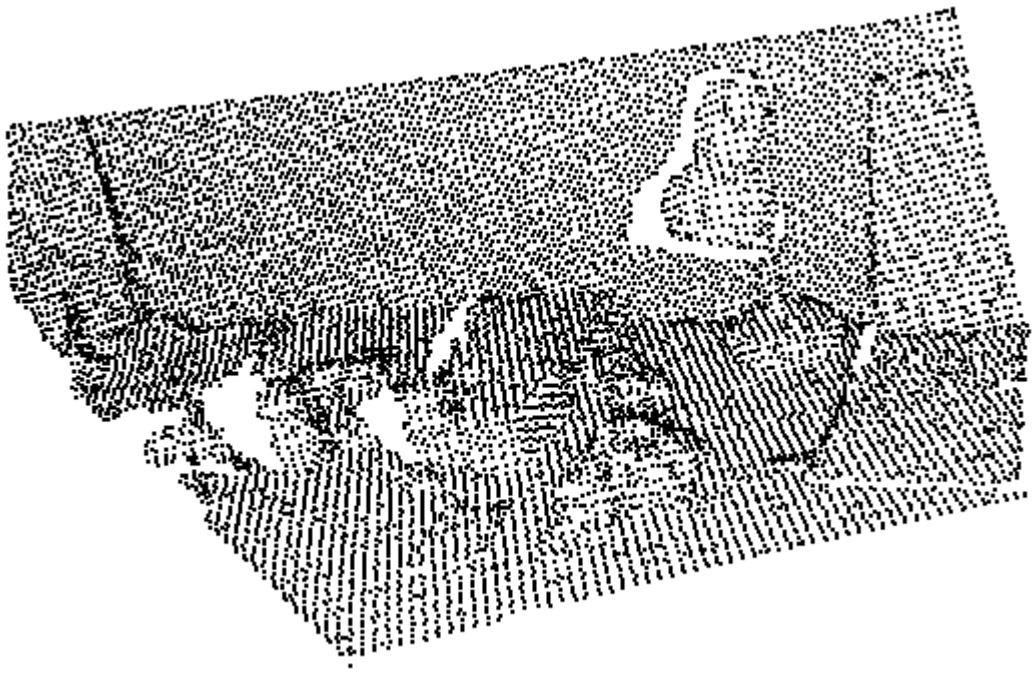
- menší přesnost než u Time of fligh metody,
- velká závislost na povrch snímaných objektů.

3. Zpracovávání naměřených dat

Dalším krokem k vytvoření prostorové mapy je následná úprava naměřených dat. V následující kapitole je uvedeno jaký formát nejčastěji mají naměřená data a jaké jsou možnosti jejich úpravy, aby se z nich dala lehce sestavit výsledná mapa.

3.1. Point cloud

Point cloud, neboli mračno bodů, je množina bodů v trojrozměrném souřadném systému. Každý z těchto bodů je definován v globálním souřadnicovém systému pomocí souřadnic X, Y, Z, což umožňuje zjistit polohu každého z bodů v prostoru a jejich vzájemné vzdálenosti. Mračno bodů je základní reprezentace dat, která se již může použít k sestavení prostorové mapy, nebo může sloužit k dalšímu zpracování. Mračno bodů může vypadat jako na obr. 3.1.



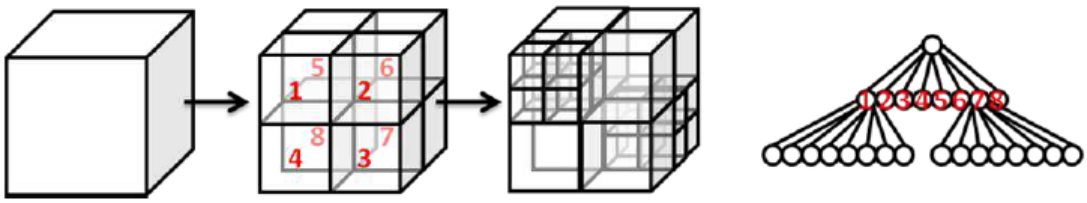
Obrázek 3.1: Mračno bodů před zpracováním

Kvalita mračna bodů závisí také na množství bodů. Počet bodů se dá měnit počtem měření v případě laserového měřiče, nebo zvolením snímače s větším rozlišení v případě

aktivní triangulace. Mezi dalšími možnostmi zpracovávání mračna bodů je oktalový strom, K-D strom, nebo segmentace objektů, které jsou uvedeny v následujících kapitolách.

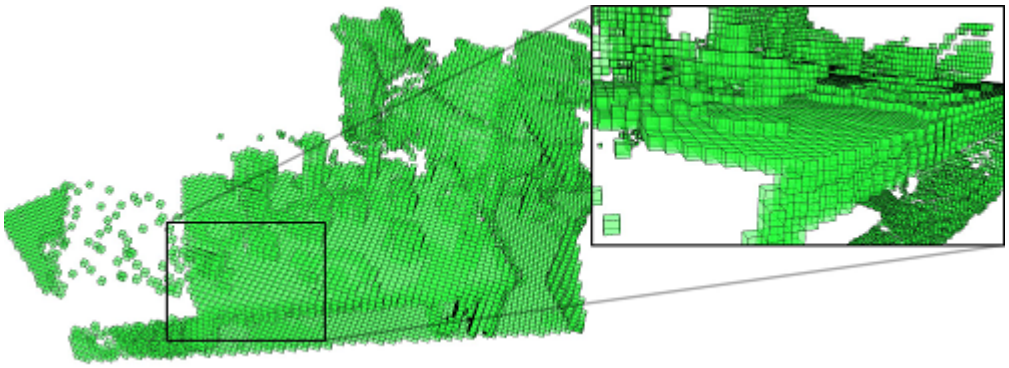
3.2. Oktalový strom

Oktalový strom je datová struktura, pomocí které je možné provést zpracování získaných dat během měření. Oktalový strom je možno si představit tak, že se celý prostor vloží do krychle, která se nazývá uzel, nebo buňka. Dalším krokem je rozdělení tohoto uzlu na menší části. Toto dělení se provádí pomocí tří na sebe kolmých rovin, které prochází středem strany krychle. Tímto se vytvoří osm menších buněk. Poté se zjistí, které buňky obsahují body. Ty buňky které neobsahují žádný bod se označí za listové a již se dále nedělí a u těch buněk, u kterých se zjistí přítomnost bodů se opět provede dělení pomocí tří řezných rovin, viz obr. 3.2.



Obrázek 3.2: Princip tvorby oktalového stromu[10, str. 130]

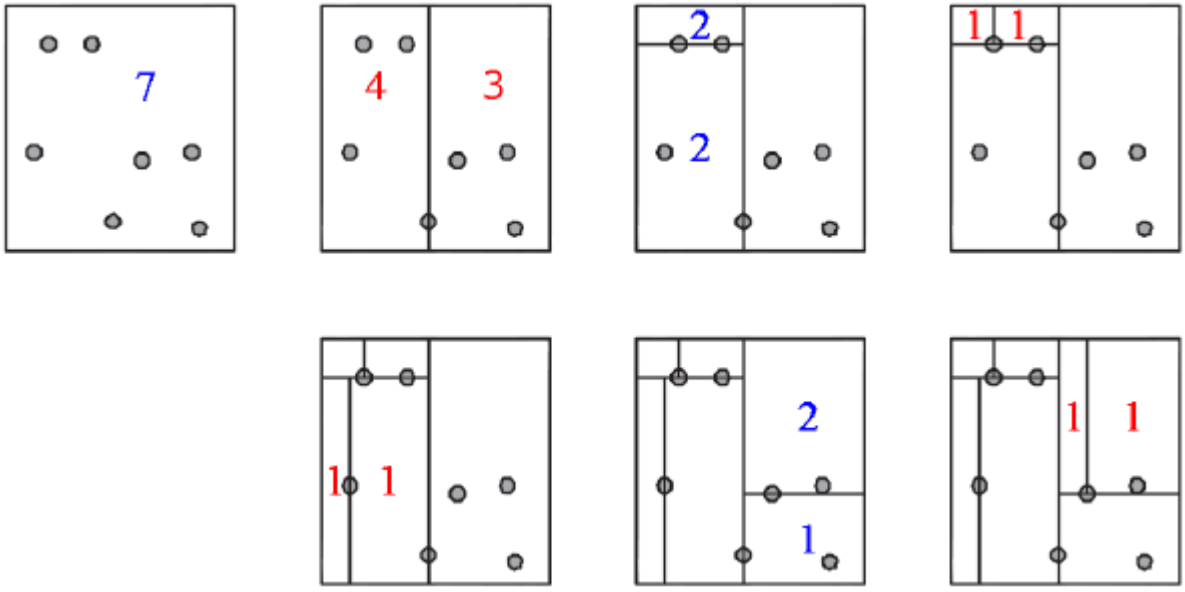
Toto dělení probíhá dokud nedosáhneme předem stanovené hloubky dělení. Pokud by se nestanovila konečná hloubka dělení, tak by toto dělení probíhalo do té doby, dokud by v každé buňce nebyl pouze jeden bod, takže by se nedosáhlo žádného zjednodušení. Výsledek zpracování naměřených dat pomocí oktalového stromu může vypadat jako na obr. 3.3.



Obrázek 3.3: Výsledek po zpracování dat pomocí oktalového stromu[9, str. 25]

3.3. K-D strom

Druhou metodou jak zpracovat naměřené body je K-D strom. Je to datová struktura podobná předchozímu oktalovému stromu. Pro vytvoření K-D stromu se opět celý prostor vloží do krychle, nebo čtverce, podle toho kolik máme dimenzí. Pro vysvětlení principu je zde ukázáno tvoření K-D stromu ve 2D viz obr. 3.4. Po vytvoření prvního čtverce provádíme postupně řezy rovinou tak, abychom postupně dostali v každém podprostoru právě jeden bod. Největším problémem u vytváření K-D stromu je zjistit, kterým bodem je nejvhodnější právě vést řez. Jedním ze způsobů jak toho dosáhnout je medián daného prostoru. Dalším způsobem je prováděním řezu uprostřed daného prostoru.



Obrázek 3.4: Princip tvorby KD stromu[10, str. 129]

3.4. Segmentace objektů

Poslední zde představenou metodou zpracování naměřených dat je segmentace objektů. Takto vytvořená mapa se skládá ze základních geometrických tvarů.

Principem vytváření objektů z množiny bodů je nahrazení několika bodů pomocí úsečky, kružnice, nebo elipsy. Poté se tyto základní tvary dále skládají dohromady a vytvoří se z nich složitější geometrické tvary, např. krychle, válec, koule a podobně.

Mezi výhody takto vytvořené mapy patří lepší práce s dynamicky se měnícím prostředím a lepší rozpoznání jednotlivých objektů pro člověka, než třeba u oktalového stromu. Nevýhodou mapy vytvořené segmentací objektů je náročnější vytváření složitých tvarů.

4. Metody skládání prostorových map

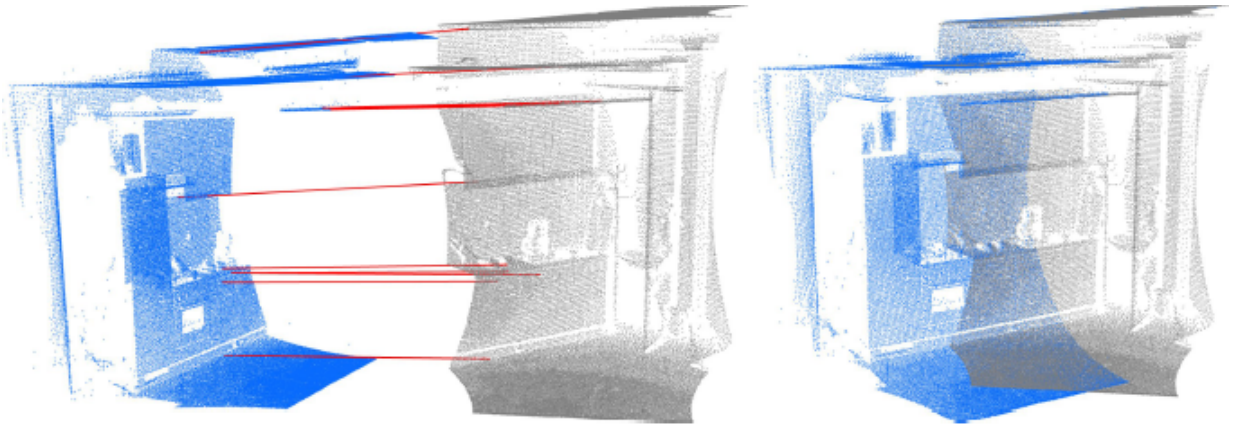
Po získání dat a následném zpracování je třeba použít získaná data a postupně je složit do výsledné mapy. K tomu se dá využít mnoho metod, kdy každá z nich má své klady i zápory. V následující kapitole je uvedeno několik z nich.

4.1. Iterative Closest point

Iterative closest point (ICP) [2] je jedna z nejpoužívanějších metod, která slouží k sestavování jednotlivých snímků do jednoho. Princip ICP je:

1. Najít na každém snímku body, jejichž vzdálenost budeme měřit a které budou sloužit jako referenční body,
2. zjistit vzdálenost mezi referenčními body,
3. provést translaci a rotaci jednoho ze snímků,
4. znovu zkontrolovat vzdálenost mezi referenčními body,
5. opakovat translaci a rotaci snímku a kontrolovat vzdálenost jednotlivých bodů dokud se nedosáhneme požadované vzdálenosti, nebo se nedosáhneme maximálního počtu iterací.

Výsledkem této iterace je transformační matice, pomocí které můžeme přetransformovat dva snímky do jednoho. Jak taková transformace dopadne, lze vidět na obr.4.1



Obrázek 4.1: Skládání dvou snímků do jednoho a výsledek transformace[9, str. 72]

Pokud se skládají snímky, které obsahují velké množství bodů, je hledání odpovídajících bodů na jiném obrazu a následné iterace výpočetně velmi náročné, proto se vytvořily různé modifikace, které tento proces urychlují. Jednou z možností je nepoužívat každý bod, ale vybrat pouze každý n-tý bod z obrazu, nebo tyto body vybírat náhodně. Dalším způsobem je vybírat dobře rozeznatelné body, např. hrany nebo rohy. Problém může nastat, jestliže se skládají dvě rovné plochy. V tomto případě se může stát, že k sobě budou přiřazeny nesprávné body a algoritmus nebude schopen najít výslednou transformační matici, nebo nalezne takovou, která sestaví dané obrazy špatně.

Další věc, kterou je třeba brát v úvahu při skládání dvou snímků, jsou krajní body. Pokud se ponechají, může se stát, že algoritmus přiřadí krajní body, které se na druhém snímku nenacházejí. Proto je při hledání korespondujících bodů bezpečnější tyto body vynechat.

Výsledný pseudokód algoritmu ICP může vypadat takto

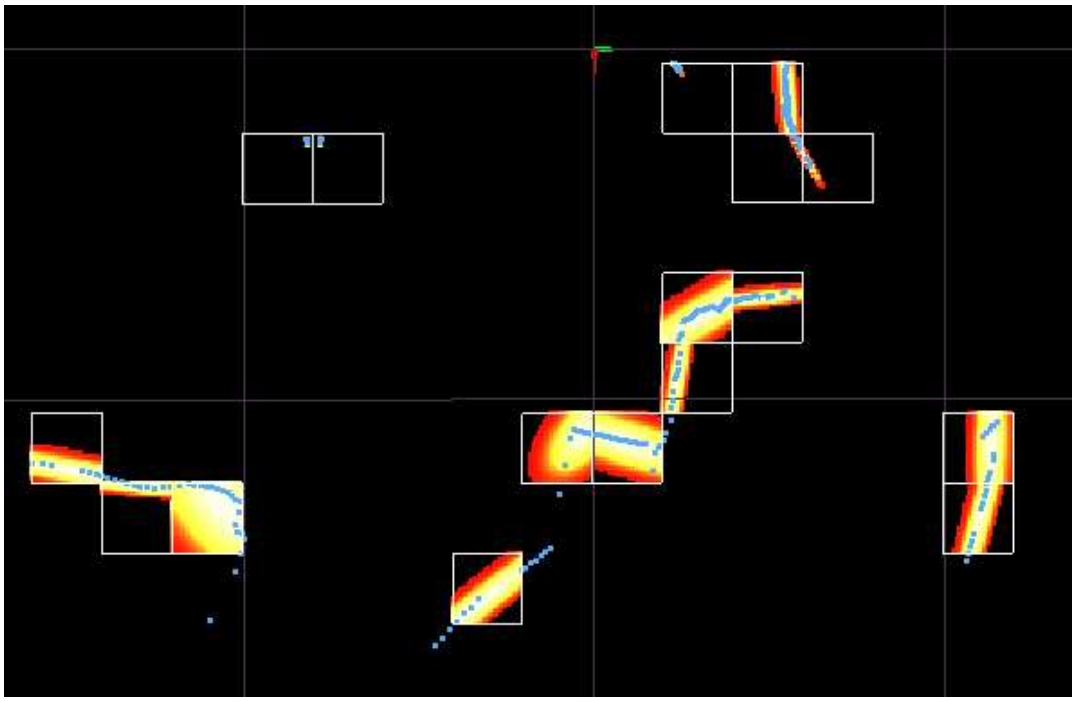
```
Iterative_closest_point
begin
        podminka1 = maximalni pocet iteraci
        podminka2 = odchylka
        while  podminka1 nebo podminka2
        {
                najit vvyznamne body
                provest priblizeni
                spocitat vzajemne odchylky vsech bodu
                iterace + 1
        }
end
```

Algorithm 4.1: Pseudokód algoritmu ICP

4.2. Normal distribution transfer

Narozdíl od ICP, který využívá k reprezentaci dat jednotlivé body z mračna bodů, Normal distribution transfer (NDT)[3] používá k reprezentaci jednotlivých bodů normální rozdělení. Toto pravděpodobnostní rozdělení umožňuje snadnější vytvoření povrchu objektu.

Pro použití NDT se musí daný prostor v 2D případě rozdělit na čtverce, nebo krychle pokud jsou data ve 3D. Jestliže je v dané buňce určitý počet bodů tak se podle hustoty bodů spočítá pravděpodobností rozložení. V místě, kde je tato pravděpodobnost největší, tam bude povrch daného objektu. To, jak může vypadat 2D mapa pomocí NDT, je ukázano na obr. 4.2.



Obrázek 4.2: Mapa vytvořená pomocí pravděpodobnostního rozdělení[7, str. 56]

Při skládání dvou snímků do jednoho se hledá vzájemná poloha snímků, která maximalizuje pravděpodobnost že snímky leží jeden na druhém viz rovnice 4.1

$$\Psi = \prod_{k=1}^{n} p(T(\vec{p}, \vec{x}_k))[7, str.58] \tag{4.1}$$

Výsledkem je opět transformační matice.

4.3. Probabilistic scan registration

Dalším algoritmem vhodným pro vytváření prostorových map je Probabilistic scan registration. Je to pravděpodobnostní metoda použitelná pro Time of flight snímače, která kombinuje gaussovo a rovnoměrné rozdělení. Pro skládání snímků se používá rovnice

$$p(S|S', l) = \prod_{s \in S} p(s|d_e(s, S', l))[4] \tag{4.2}$$

kde $d_e(s, S', l)$ je předpokládaná vzdálenost k objektu, s je směr, S' je předchozí snímek a l je poloha senzoru. Pro určení posunu snímku S se opakovaně mění poloha senzoru l tak dlouho, dokud roste pravděpodobnost $p(S|S', l)$.

Podle [4] má tato metoda tři výhody oproti ICP. První je, že zvládá sledování paprsku během pohybu, takže nepotřebuje žádné další výpočty pro vypořádaní se se skluzem. Druhou výhodou je lepší zpracovávání na velké vzdálenosti a třetí je jednoduché nastavování parametrů pro měření, podle typu použitého senzoru.

5. Microsoft Kinect

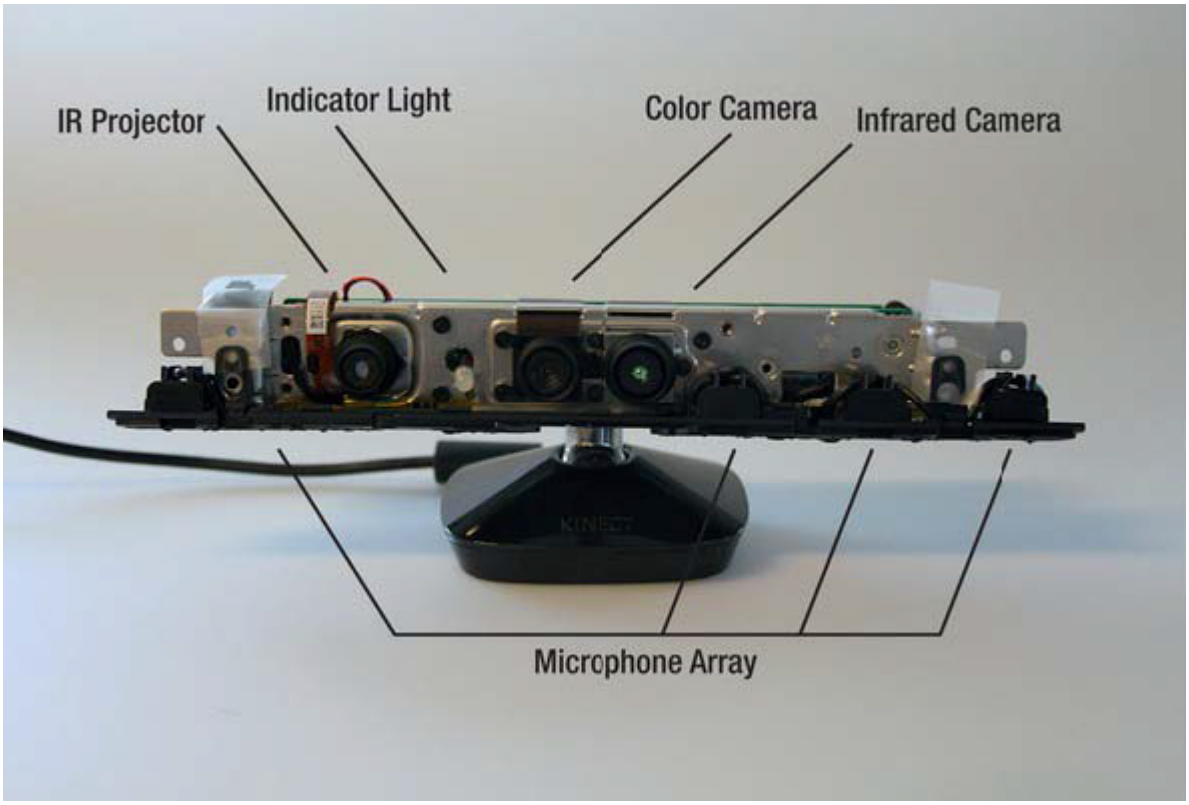
5.1. Historie

Microsoft Kinect(viz obr.5.1) je zařízení vyrobené v roce 2010 firmou Microsoft pro herní konzoli Xbox 360. Účelem Microsoftu bylo vytvořit herní zařízení, které umožňuje ovládat hru jen pomocí gest a hlasu. Po nějaké době se začaly objevovat neoficiální ovladače, které umožňovaly připojit a provozovat MS Kinect na PC. V roce 2011 byly také vydány oficiální ovladače, které umožňují propojení PC a Kinectu a poté v roce 2012 Microsoft vydal upravený Kinect tak, aby byl lépe optimalizován pro práci s počítačem. Toto zařízení se jmenuje Kinect for Windows.

Obrázek 5.1: Microsoft Kinect [11, str. 9]

5.2. Hardware

MS Kinect obsahuje RGB kameru, infračervený vysílač a snímač a následně několik mikrofonů. Rozmístění jednotlivých komponentů, po sejmutí krytu, lze vidět na obr.5.2.



Obrázek 5.2: Microsoft Kinect po sejmutí krytu [11, str. 10]

5.2.1. Hloubkový senzor

Hloubkový senzor se skládá z infračerveného zářiče a snímače. Vysílač vysílá infračervené světlo do okolí a to se od objektů odráží zpět do senzoru, který podle úhlu dopadu spočítá vzdálenost objektu od senzoru. MS Kinect využívá principu aktivní triangulace, který je popsán v kapitole 2.4.

5.2.2. RGB kamera

Barevná kamera slouží k natáčení barevného videa. Může natáčet v rozlišení 640x480 při 30 FPS nebo 1280x960 kdy dosahuje 12 FPS.

5.2.3. Mikrofony

MS Kinect obsahuje 4 mikrofony, které jsou rozmístěny vedle sebe, kde jeden je na jedné straně zařízení a zbylé tři na druhé. Toto rozmístění umožňuje zaznamenávat okolní zvuky, zjistit směr ze kterého přicházejí a omezit šum a ozvěnu. Tyto mikrofony lze využít k případnému ovládání pomocí hlasu.

5.3. Software

Pro používání senzoru Microsoft Kinect je možno využít několik softwarových nástrojů. V následujících kapitolách uvedu několik volně šiřitelných, ale existuje i několik placených programů, které jsou úzce zaměřené a uživatel nepotřebuje hlubší znalosti k jejich použití.

5.3.1. Kinect for Windows

Prvním z těchto volně šiřitelných nástrojů je Kinect for Windows. Jedná se sice o oficiální nástroje vydané firmou Microsoft pro Kinect for Windows, ale s malým omezením lze pomocí nich provozovat i Kinect, který byl původně určen pouze pro XBox 360. Jak už název napovídá, tyto nástroje jsou určeny pouze pro Windows 7 a novější.

Součástí Kinect for Windows, kromě ovladačů pro samotný Kinect, je i Software developement kit, který umožňuje využívat předem naprogramované funkce pro vlastní programy. Následně je doporučeno si nainstalovat i ToolKit, který obsahuje dokumentaci a ukázkové aplikace se zdrojovými kódy pro lepší pochopení práce s těmito nástroji.

5.3.2. OpenNI

Další z možností jak připojit Microsoft Kinect k počítači je OpenNI. Tyto nástroje slouží zejména k propojení Kinectu s počítačem a získání dat ze samotného senzoru.

5.3.3. Point Cloud Library

Posledním ze zde uvedených nástrojů je Point Cloud Library(PCL)[12]. Jsou to open-source nástroje, představené v roce 2011, jejichž původními autory jsou Radu Bogdan Rusu a Steve Cousins. Tyto nástroje umožňují pomocí výše zmíněných ovladačů OpenNI naměřit data a především umožňují velké množství následujících úprav. Jelikož jsou tyto nástroje k dispozici už dva roky a vývoj stále pokračuje, je možno využít kromě základních filtrů i mnoho dalších pokročilých algoritmů pro úpravu obrazu a naměřených dat. Tyto nástroje lze využít jak na operačním systému Microsoft Windows tak i v operačních systémech Mac OS a Linux.

Část II

Praktická část

6. Měřicí přístroje

6.1. Hardware

Měření bylo prováděno pomocí senzoru Microsoft Kinect pro XBox 360. Kinect byl pomocí USB kabelu propojen s počítačem. Konfigurace počítače který byl použit pro nahrání a následné zpracování dat je:

- procesor Intel Core 2 Duo T6600 2,2 GHz,
- 4 GB RAM,
- grafická karta Nvidia GeForce GT 220M.

6.2. Software

Celé měření a zpracovávání bylo prováděno v operačním systému Ubuntu 12.10. Pro naměření hloubkových dat a jejich následné zpracování byla zvolena open-source knihovna Point Cloud Library ve verzi 1.7. Důvody pro tuto volbu jsou:

- snadné nahrávání a opětovné přehrávání naměřených dat,
- velké množství způsobů úpravy naměřených dat,
- kvalitní doprovodná dokumentace.

Tato verze PCL knihovny sice není oficiálně vydaná jako stabilní, ale nachází se v ní úpravy dat, které v poslední stabilní verzi nejsou, a pomocí nich je zpracování jednodušší.

7. Popis programu

Celý program byl rozdělen do tří částí. První z nich je nahrávání dat pomocí senzoru Microsoft Kinect. Druhá část programu obsahuje následné zpracování naměřených dat a jejich složení do jednoho obrazu. Třetí část obsahuje následné zpracování a a úpravu povrchu. Toto rozdělení celého procesu bylo zvoleno z důvodu velké výpočetní náročnosti a nedostatečně výkonnému hardwaru, který byl použit během měření.

Z důvodů možnosti porovnat jednotlivé variace algoritmu Iterative Closest point (viz kapitola 4.1), byly vytvořeny dvě verze programu. Vybraným algoritmem pro skládání prostorové mapy je iterative closest point s vyhledáváním klíčových bodů pro urychlení výpočtů.

7.1. Recorder

Nahrávání dat pomocí senzoru Microsoft Kinect byl vytvořen program který využívá třídu OpenNIGrabber. Tato třída umožňuje snímání okolí pomocí senzorů, které jsou kompatibilní s ovladači OpenNI. Snímaná data jsou dále ukládány do .pcd (point cloud data) souboru. Tento formát umožňuje ukládat buď pouze polohu jednotlivých bodů, nebo současně i jejich barvu. Tato data je možno ukládat v ASCII formátu, který je pro člověka čitelnější, nebo je možno je také ukládat v binární komprimované podobě.

Pro lepší práci s naměřenými daty jsou jednotlivé snímky očíslovány podle pořadí v jakém byly nahrány.

7.2. Skládání mapy

Poté co jsou data indoor prostředí nahraná, může se přistoupit k jejich zpracování. Pro zpracování naměřených dat byl, za využití open-source knihovny Point Cloud Library, vytvořen program, který nejprve provede zpracování a filtrování, poté složí jednotlivé snímky do jednoho. V následujících podkapitolách jsou popsány následující metody, které byly vytvořeny pro potřeby diplomové práce.

7.2.1. Voxel grid filtr

První použitou úpravou na naměřená data je voxel grid filtr. Tento filtr slouží k redukci počtu bodů v mračnu. Filtrování probíhá tak, že je vybrán kvádr, jehož všechny body se nahradí jedním bodem, který se nachází jeho geometrickém středu.

```
PointCloud<PointT>::Ptr
Voxel_Grid_Filter(PointCloud<PointT>::Ptr &cloud,
                                          float leaf_Size)
{
        PointCloud<PointT>::Ptr points_Vox (new PointCloud<PointT>);
        VoxelGrid<PointT> vox;
        vox.setLeafSize(leaf_Size, leaf_Size, leaf_Size);
        vox.setInputCloud(cloud);
        vox.filter(*points_Vox);
        return points_Vox;
}
```

Algorithm 7.1: Voxel grid filtr

Vstupními hodnotami pro tuto metodu je velikost strany krychle v metrech a mračno bodů určené pro vyfiltrování. Výstupem je vyfiltrované mračno bodů, které se bude dále upravovat.

7.2.2. Radius outlier removal filtr

Druhou úpravou dat, která je použita je radius outlier removal filtr. Tento filtr slouží k odstranění samostatných bodů v mračnu, proto je vhodný pro odfiltrování šumu. Jeho funkce je založena na postupném zkoumání okolí všech bodů a pokud se v okolí nenachází dostatečné množství bodů, tak je odstraněn. Hlavními parametry tohoto filtru jsou vzdálenost, v které probíhá prohledávání, a minimální počet okolních bodů v dané vzdálenosti od bodu.

```
PointCloud<PointT>::Ptr
Radius_Outlier_Removal_Filter(PointCloud<PointT>::Ptr &cloud,
                                                    double radius_Search,
                                                    int min_Neighbors)
{
        PointCloud<PointT>::Ptr points_Ror (new PointCloud<PointT>);
```

```
        RadiusOutlierRemoval<PointT> ror;
        ror.setRadiusSearch(radius_Search);
        ror.setMinNeighborsInRadius(min_Neighbors);
        ror.setInputCloud(cloud);
        ror.filter(*points_Ror);
        return points_Ror;
}
```

Algorithm 7.2: Radius outlier removal filtr

Vstupem pro tento filtr je tedy mračno bodů určené pro vyfiltrování, vzdálenost v metrech ve které probíhá hledání a minimální počet okolních bodů v dané vzdálenosti. Výstupem je opět vyfiltrované mračno bodů.

7.2.3. Vyhledávání významných bodů

Toto vyhledávání využívá algoritmus SIFT. Používá gaussovo rozostření obrazu k nalezení hran a jiných významných bodů. Vstupem této metody je mračno bodů a parametry, které ovlivňují počet a kvalitu nalezených bodů. Výstupem je mračno nalezených klíčových bodů.

```
PointCloud<PointT>::Ptr
Detect_Keypoints(PointCloud<PointT>::Ptr &points,
                 float min_Scale,
                 int nr_Octaves,
                 int nr_Scales_Per_Octave,
                 float min_Contrast)
{
        SIFTKeypoint<PointT, PointWithScale> sift_Keypoints;
        search::KdTree<PointT>::Ptr tree (new search::KdTree<PointT>);
        PointCloud<PointWithScale> keypoints_Temp;
        PointCloud<PointT>::Ptr keypoints (new PointCloud<PointT>);
        sift_Keypoints.setSearchMethod (tree);
        sift_Keypoints.setScales (min_Scale, nr_Octaves, nr_Scales_Per_Octave);
        sift_Keypoints.setMinimumContrast (min_Contrast);
        sift_Keypoints.setInputCloud (points);
        sift_Keypoints.setSearchSurface(points);
        sift_Keypoints.compute (keypoints_Temp);
        copyPointCloud (keypoints_Temp, *keypoints);
```

```
        return (keypoints);
}
```

Algorithm 7.3: Vyhledávání klíčových bodů

7.2.4. Iterative closest point

Třetím krokem je složení vyfiltrovaných dat do jednoho mračna bodů. Vstupem této metody, která provádí spojování, jsou 2 mračna bodů. První mračno(source) se bude transformovat do druhého(target). Výstupem je složené mračno bodů.

```
PointCloud<PointT>::Ptr
Iter_Closest_Point(PointCloud<PointT>::Ptr cloud_Source_Key,
                   PointCloud<PointT>::Ptr cloud_Target_Key,
                   PointCloud<PointT>::Ptr cloud_Source,
                   PointCloud<PointT>::Ptr cloud_Target)
{
        PointCloud<PointT>::Ptr result(new PointCloud<PointT>);
        PointCloud<PointT>::Ptr output(new PointCloud<PointT>);
        Eigen::Matrix4f tr = Eigen::Matrix4f::Identity();
        IterativeClosestPoint<PointT, PointT> iter;
        iter.setInputCloud(cloud_Source_Key);
        iter.setInputTarget(cloud_Target_Key);
        iter.setMaxCorrespondenceDistance(1);
        iter.setMaximumIterations(50);
        iter.setTransformationEpsilon(1e-8);
        iter.setEuclideanFitnessEpsilon(0.0001);
        iter.align(*result);
        tr = iter.getFinalTransformation();
        transformPointCloud(*cloud_Source, *output, tr);
        *output += *cloud_Target;
        return output;
}
```

Algorithm 7.4: Skládání dvou snímků pomocí ICP

Důležité metody, které ovlivňují výsledek jsou:

- setMaximumIterations() - maximální počet iterací,

- setTransformationEpsilon() - pokud je rozdíl mezi předchozí transformací a následující menší než zadaná hodnota, tak se ukončí výpočet,
- setEuclideanFitnessEpsilon() - pokud je součet kvadratických chyb vzdáleností všech bodů, které se k sobě přiřazují menší než zadaná hodnota, tak se ukončí výpočet,
- setMaxCorrespondenceDistance() - maximální vzdálenost mezi dvěma body, které se mají přiřadit,
- align() - výpočet transformační matice, je možné vložit počáteční odhad pro zpřesnění výsledků,
- getFinalTransformation() - výsledná transformační matice.

První tři parametry jsou podmínkami, přičemž splění jedné znich ukončuje výpočet.

7.2.5. Match

Poslední metodou je Match. Tato metoda zajišťuje celý běh programu. Obsahuje deklarace a definice použitých parametrů, volá jednotlivé metody a obsahuje cyklus, který řídí načítání dat a jejich následné zpracování ve správném pořadí.

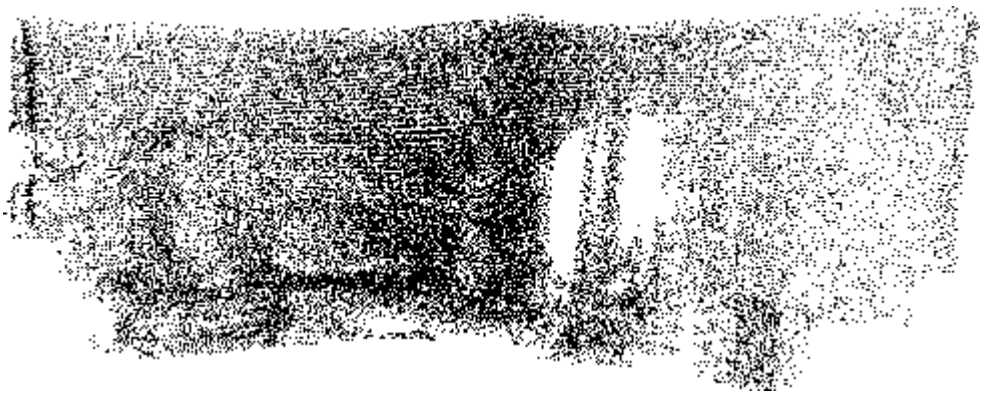
```
output = Iter_Closest_Point(cloud_Key_Source,
             cloud_Key_Target, cloud_Source, cloud_Target, transform);
if (counter > 0)
{
        global = transform * global;
}
switch(counter)
{
        case 20: case 40: case 60:
        cloud = Load(sf.str());
        transformPointCloud(*output, *result, global);
        *result += *cloud;
        result = Voxel_Grid_Filter(result, leaf_Size);
        Save(result, sf.str());
        break;
}
```

Algorithm 7.5: Řízení skládání snímků

V této ukázce kódu lze vidět metodu pro skládání zpracovaných dat z které se získává také výsledná transformační matice. Ta se s každým krokem kumuluje v proměnné "global". Ukládání transformovaných snímků neprobíhá v každém kroku, ale pouze každý dvacátý snímek je nejprve transformován pomocí transformační matice "global"a až poté je uložen. Důvodem pro tuto volbu je snížení počtu bodů ve výsledné mapě, zvýšení její přehlednosti a méně šumu.

7.3. Vytváření povrchu

Vytvořená prostorová mapa je po složení někdy nepřehledná a je poměrně těžké rozpoznat tvary objektů nebo prostoru, viz obr.7.1. Je proto nutné provést další úpravy, aby bylo snadnější se v ní orientovat.



Obrázek 7.1: Výsledek skládání před úpravami

7.3.1. Vyhlazování povrchu

Po vytvoření povrchu je nutné provést několik úprav, aby byl povrch co nejvíce hladký. Nejprve se provede snížení počtů bodů pomocí voxel grid filtru, odstraní se samostatné body pomocí radius outlier filtru. Třetí úpravou je vyhlazení pomocí algoritmu nejmenších čtverců a zároveň je provedeno zvýšení počtu bodů pro vytvoření jemnějšího povrchu. K tomu byla vytvořena metoda 7.6.

```
PointCloud<PointT>::Ptr MLS(PointCloud<PointT>::Ptr &cloud)
    {
        pcl::MovingLeastSquares<PointT, PointT> mls;
        mls.setInputCloud (cloud);
```

```
        mls.setSearchRadius (0.2);
        mls.setUpsamplingMethod (pcl::MovingLeastSquares<PointT, PointT>::SAMPLE_L
        mls.setUpsamplingRadius (0.005);
        mls.setUpsamplingStepSize (0.003);
        PointCloud<PointT>::Ptr cloud_smoothed (new PointCloud<PointT>);
        mls.process (*cloud_smoothed);
        return cloud_smoothed;
    }
```

Algorithm 7.6: Vyhlazování

Vstupem je mračno bodů které chceme upravit. Důležité parametry jsou:

- setSearchRadius() - vzdálenost od bodu, podle které se provádí vyhlazení,
- setUpsamplingRad() - vzdálenost od bodu, kam až se přidávají další body,
- setUpsamplingStepSize() - velikost kroku po kterém se přidávají body udávané v metrech, stejně jako předchozí parametry.

7.3.2. Vytváření trojúhelníkové sítě

Posledním krokem k vytvoření povrchu je vytvoření trojúhelníkové sítě. K tomuto účelu byla vytvořena metoda Meshig, která využívá algoritmu greedy triangulation. Vstupem pro tuto metodu je mračno bodů, výstupem je soubor uložený ve formátu .vtk.

```
PointCloud<Normal>::Ptr MLS(PointCloud<PointNormal>::Ptr &cloud)
{
        GreedyProjectionTriangulation<PointNormal> mesh;
        PolygonMesh triangles;
        mesh.setSearchRadius (0.01);
        mesh.setMu (2);
        mesh.setMaximumNearestNeighbors (200);
        mesh.setMaximumSurfaceAngle(M_PI/4);
        mesh.setMinimumAngle(M_PI/18);
        mesh.setMaximumAngle(2*M_PI/3);
        mesh.setInputCloud (cloud_with_normals);
        mesh.setSearchMethod (tree2);
        mesh.reconstruct (triangles);
```

```
}
```

Algorithm 7.7: Vytváření sítě

Důležité parametry jsou:

- setSearchRadius() - určuje vzdálenost, ve které se hledají nejbližší body,
- setMu() - násobek vzdálenosti k nejbližšímu bodu, tato vzdálenost je poté použita ke spojování,
- setMaximumNearestNeighbors() - maximální počet sousedních bodů,
- setAngle() - nastavuje minimální nebo maximální úhly mezi jednotlivými stranami trojúhelníku a maximální úhel mezi dvěma trojúhelníky.

8. Průběh měření

Měření probíhalo tak, že se spustil program pro nahrávání a poté byl pomocí pomalých pohybů senzorem nasnímán daný prostor. K tomu bylo vybráno několik rozličných prostorů.

První prostor je pokoj 1. Tento pokoj je menší a to byl také důvod pro výběr této místnosti. Dalšími důvody jsou zkosená střecha a velké množství nerovných ploch.

Další prostor, ve kterém bylo provedeno nahrávání, je chodba. Je to o něco větší prostor než předchozí pokoj. Obsahuje pouze dveře a schody. Důvodem pro volbu tohoto prostoru bylo zjistit chování algoritmu u holých stěn a schodiště.

Jako třetí místnost byla vybrána kuchyň. Důvody pro výběr jsou menší rozměry místnosti a výskyt mnoha menších předmětů, které mohou usnadnit skládání snímků.

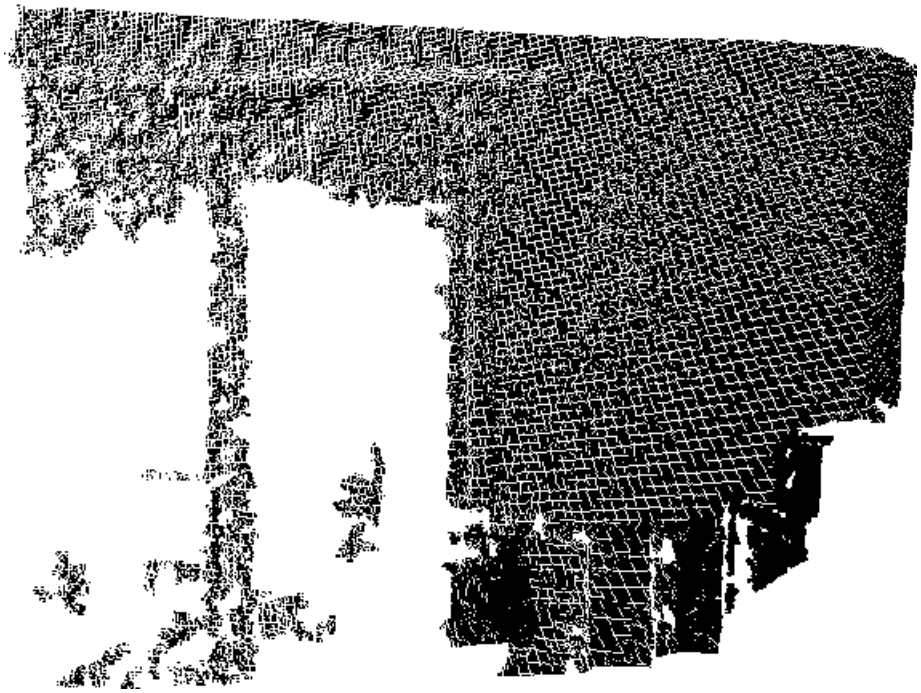
Posledním prostorem je pokoj 2. Je to větší místnost s mnoha okny u kterých byly z důvodu lepšího skládání odstraněny záclony. Důvodem pro výběr tohoto pokoje byl zejména jeho větší rozměr.

Bylo také provedeno pokusné měření v outdoor prostředí, ale senzor nebyl schopný sejmout jakákoli data. Důvodem byl zřejmě příliš drsný povrch venkovní stěny a velmi nepravidelný povrch travnatých ploch.

9. Výsledky měření

9.1. Pokoj 1

První měření probíhalo v pokoji 1. Na obr. 9.1 lze vidět první nezpracovaný snímek. Je na něm možné pozorovat pouze stěnu a bílá místa, která představují okna. Takový nezpracovaný snímek obsahuje 307200 bodů, proto je nutné provést filtraci.

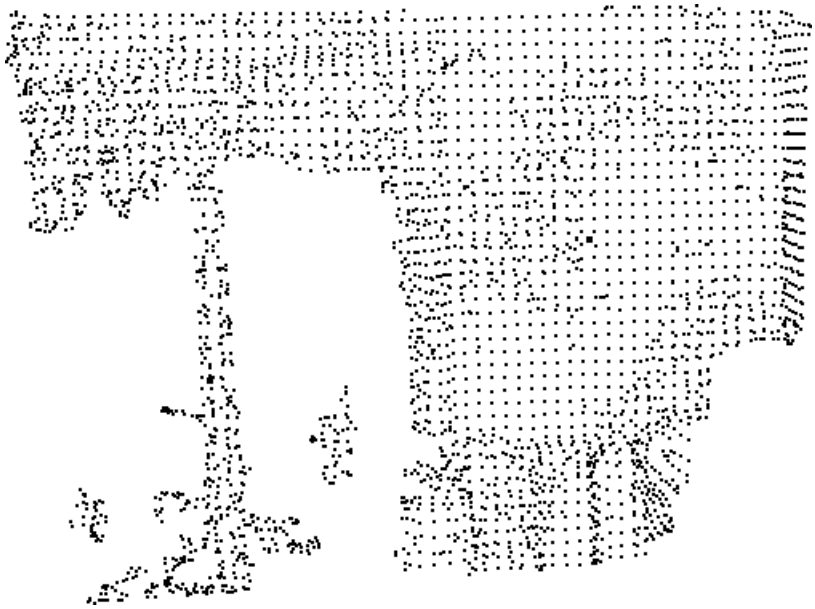


Obrázek 9.1: Snímek z pokoje 1 bez úprav

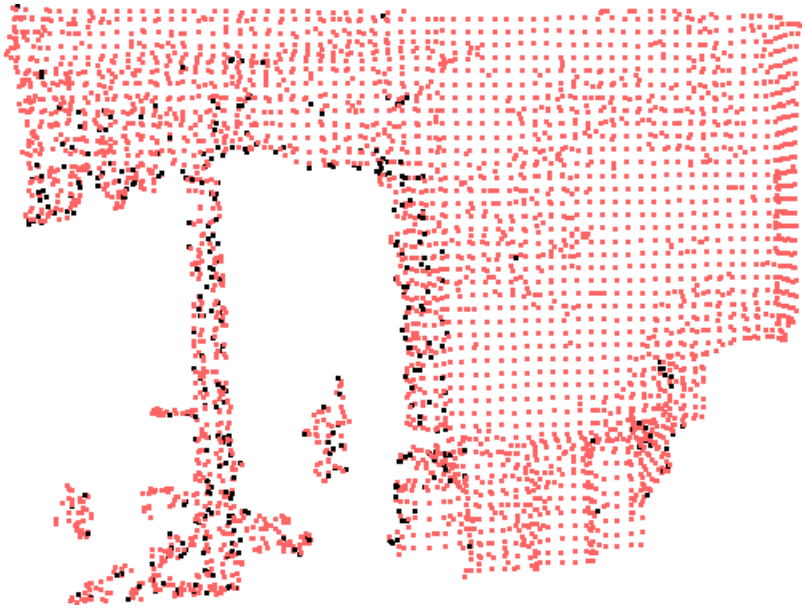
Po aplikování filtrů se počet bodů může, v závislosti na nastavení filtru, velmi snížit. Na obr. 9.2 lze vidět stejnou scénu jako v přechozím případě. Počet bodů se snížil na 3690, což velmi urychlí následné skládání snímků.

Dalším krokem při zpracování bylo vyhledávání klíčových významných bodů. Výsledné klíčové body mohou vypadat jako na obr. 9.3. Pro lepší představu byly klíčové body zvýrazněny černě a původní data mají červenou barvu. Nalezených klíčových bodů je v tomto případě 388.

Dalším krokem je samotné skládání snímků do jednoho. I přes všechny úpravy byla tato operace zdlouhavá a složení šedesáti snímků do jednoho trvalo přibližně půl hodiny. Výřez z výsledného mračna je na obr. 9.4. Lze vidět, že výsledných bodů je poměrně dost, takže nelze dobře rozeznat jednotlivé objekty. Proto je nutné provést další úpravy.



Obrázek 9.2: Snímek z pokoje 1 po aplikování filtrů

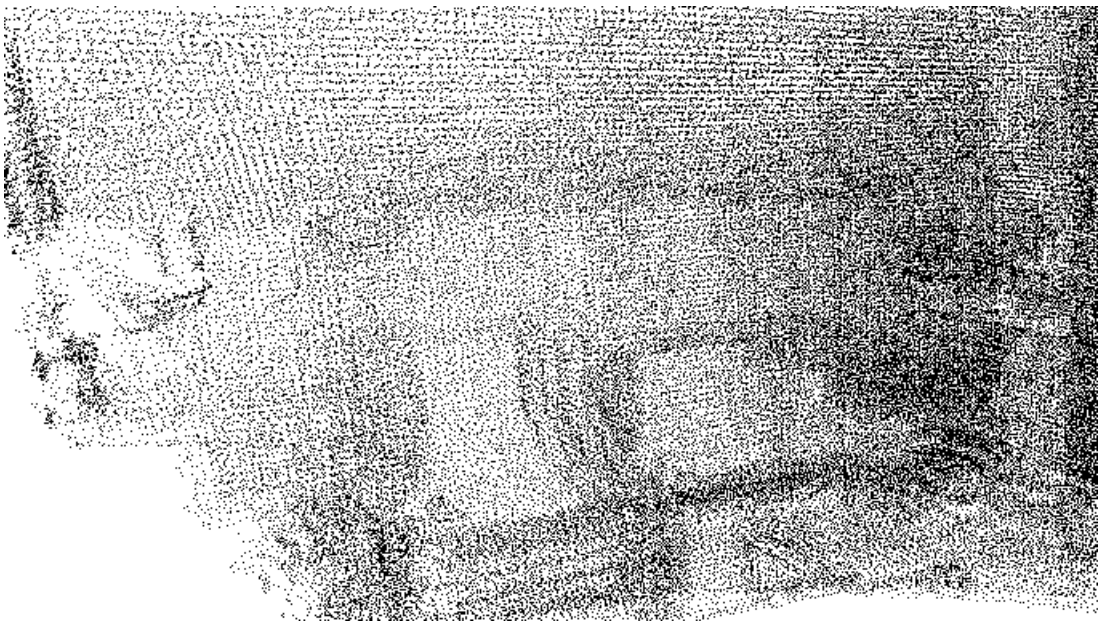


Obrázek 9.3: Nalezené klíčové body

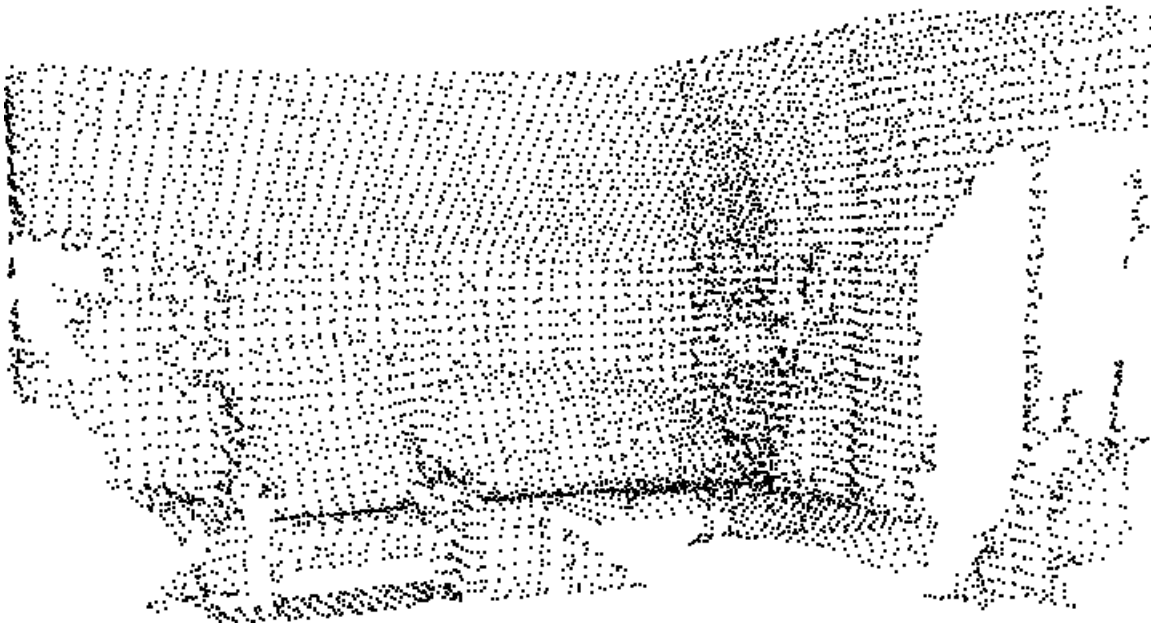
Výsledek dalšího snížení počtu bodů a jejich vyhlazení lze vidět na obr. 9.5. Na tomto snímku lze již vidět, že se začínají rýsovat obrysy některých objektů.

Posledním krokem je vytváření trojúhelníkové sítě. Výsledek takto vytvořeného povrchu lze vidět na obr. 9.6

Z předchozích snímků je patrné, že vytvoření prostorové mapy se v tomto případě povedlo. Na obrázcích není těžké najít jednotlivé objekty a i stěny místnosti nevykazují větší odchylku.

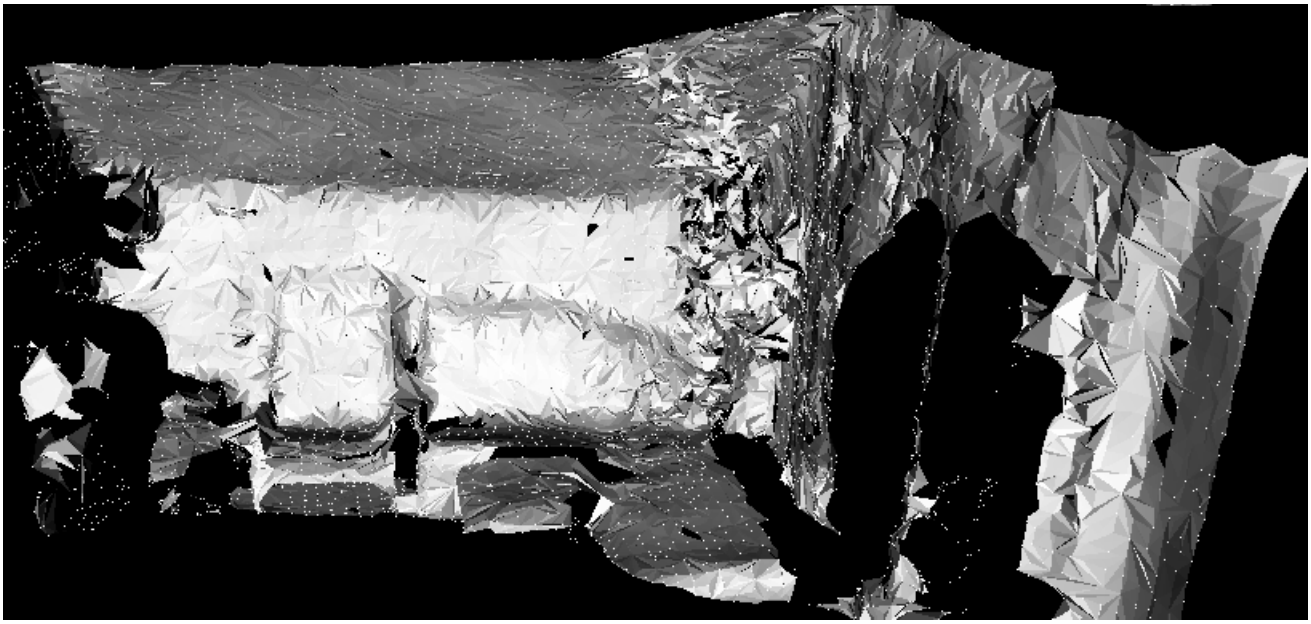


Obrázek 9.4: Výsledek algoritmu ICP



Obrázek 9.5: Prostorová mapa po vyhlazení

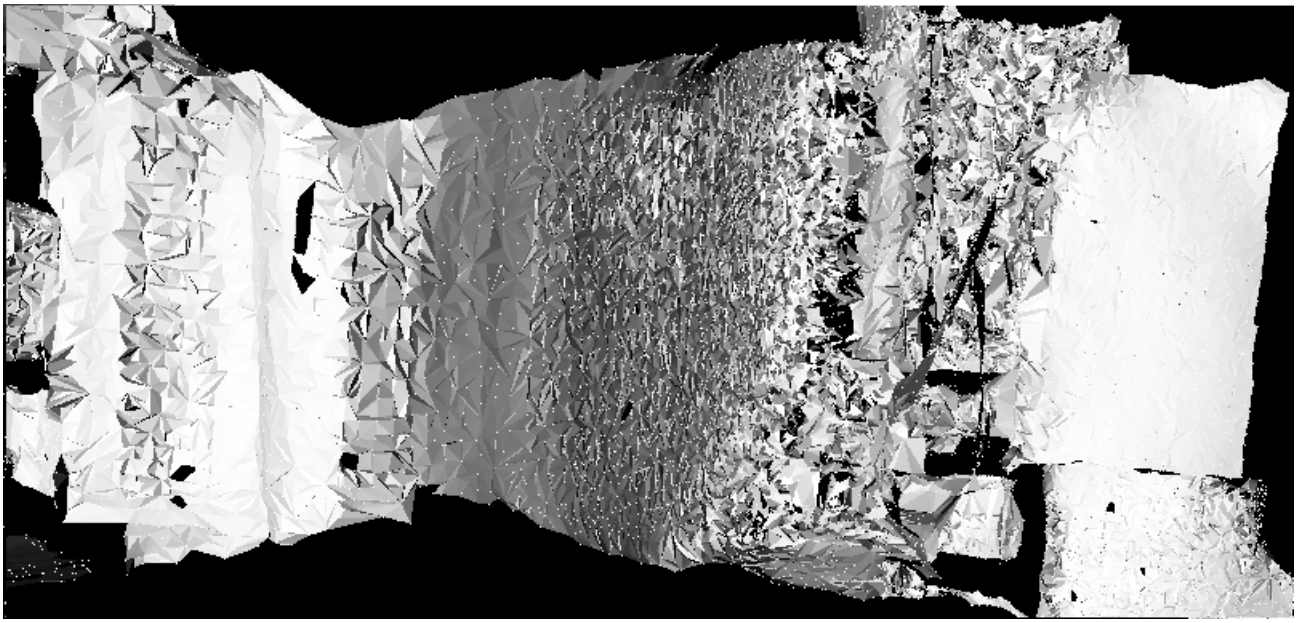
U dalších měření je ukázána pouze výsledná prostorová mapa.



Obrázek 9.6: Výsledná prostorová mapa pokoje 1

9.2. Chodba

Během výpočtů se ukázalo, že holé zdi a schody dělaly programu největší problém. Pravděpodobně to bylo z důvodů nepřesně vybraných klíčových bodů a proto jsou zdi vůči sobě nepřirozeně posunuté. Výsledná prostorová mapa tohoto prostředí je na obr. 9.7

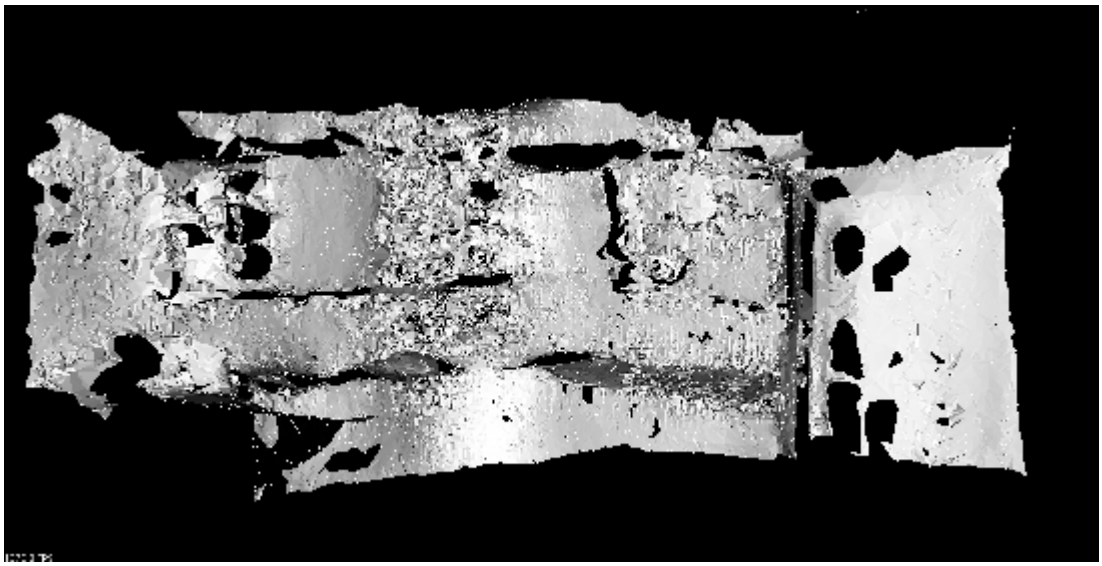


Obrázek 9.7: Výsledná prostorová mapa chodby

Z výsledné prostorové mapy je možno usuzovat, že v tomto případě je nutné provést počáteční odhad transformace pro lepší složení snímků.

9.3. Kuchyň

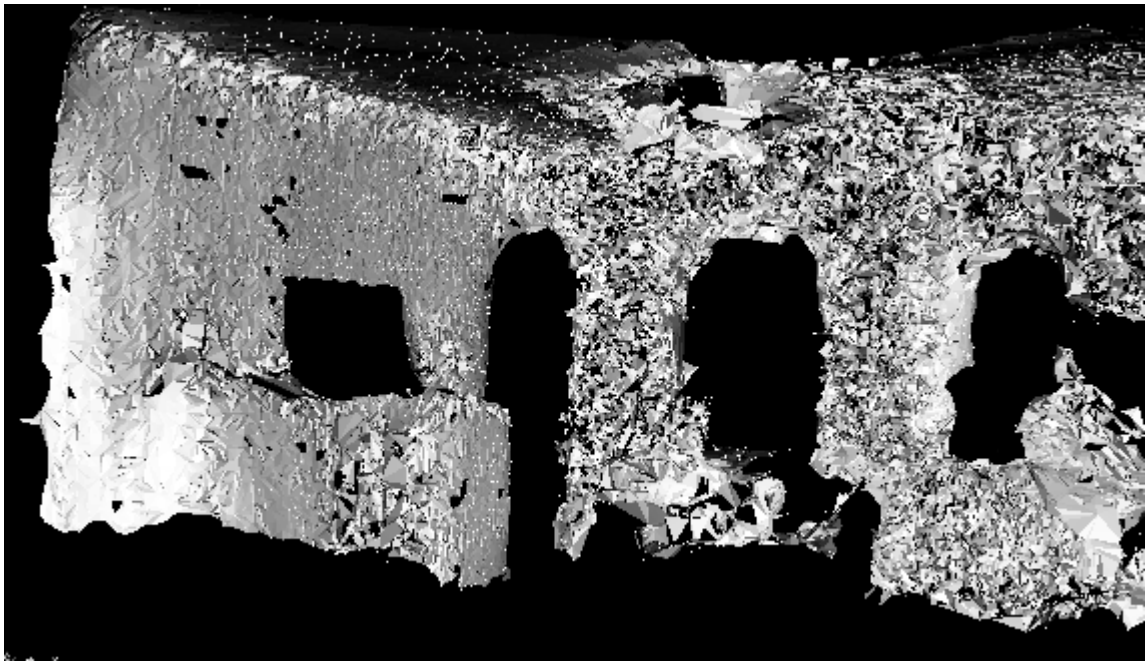
Třetím prostorem je kuchyň. Výsledná prostorová mapa, viz obr. 9.8, se povedla celkem dobře sestavit. Problém nastal při vytváření povrchu, kdy v důsledku nízké hustoty bodů vznikly v některých místech ostré hrany. Úspěšnější sestavování mapy bylo pravděpodobně způsobeno výskytem většího množství různorodých předmětů a ostré hrany kuchyňského koutu.



Obrázek 9.8: Výsledná prostorová mapa kuchyně

9.4. Pokoj 2

Poslední typově odlišnou místností je pokoj 2. Výsledná prostorová mapa, viz obr. 9.9, se nesestavila tak dobře jako v předchozím případě. Důvody pro chyby v sestavování byly zřejmě větší rozměry této místnosti než v předchozích případech a umístění větších rostlin v protějších rozích, které vnášely chyby do měření.



Obrázek 9.9: Výsledná prostorová mapa pokoje 2

10. Závěr

V teoretické části jsem představil některé typy senzorů, které se dají využít k získávání hloubkových dat. Ukázal jsem možné reprezentace dat, jak mohou být naměřená data zobrazena. Popsal jsem základní princip některých metod sloužících ke skládání prostorových map a představil jsem samotný senzor Microsoft Kinect a software, který umožňuje zpracování naměřených dat.

V praktické části jsem popsal měřicí aparát, ukázal důležité části zdrojového kódu programu, představil jednotlivé prostory a objasnil, v čem se liší. Odůvodnil jsem volby těchto prostor a představil výsledky skládání prostorových map a zhodnocení výsledků.

Pro vytváření prostorových map jsem provedl měření ve čtyřech různých prostředích. Vybral jsem dva různé pokoje. Jejich rozdíl je zejména ve velikosti, ale také v doplňcích, které se v nich vyskytují. Dalším prostorem byla chodba. Ta se liší od ostatních prostor zejména holými stěnami, ale také i prázdnými schody. Poslední ze zde představených prostorů je kuchyň, ve které se na rozdíl od ostatních prostor vyskytuje větší množství menších předmětů. Provedl jsem také pokus o naměření dat v outdoor prostředí, ale toto měření bylo z důvodu velké členitosti povrchu neúspěšné.

Po naměření dat přišlo na řadu zpracování a následné vytváření prostorové mapy. Abych mohl mapu zpracovat, musel jsem použít několik filtrů a pro následné vytvoření mapy jsem použil algoritmus pro hledání klíčových bodů, které jsem následně využil k skládání pomocí algoritmu Iterative closest point.

Z výsledků je patrné, že použitý algoritmus měl s některými prostory problém při sestavování což mělo za důsledek nepřesně sestavené mapy. Důvodů je několik, ale nejčastěji to jsou rušivé elementy jako rostliny, nebo v jednom případě schody. Ale při celkovém pohledu na výsledky je možno konstatovat, že mnou vytvořený program je schopen vytvářet prostorové mapy indoor prostředí.

Celý proces zpracovávání výsledků, z důvodů nízkého výkonu počítače na kterém probíhal výpočet, se uskutečňoval offline, což je škoda, protože se objevují algoritmy, které provádějí měření, skládání i rekonstrukci povrchu v reálném čase. Výstupem takového algoritmu je mnohem podrobnější a přesnější prostorová mapa, zejména díky možnosti okamžité kontroly naměřených a složených dat a kontroly polohy senzoru vůči danému

objektu. Dalším stupněm je simultánní lokalizace a mapování, což má využití zejména u robotů, které se pohybují v neznámém prostředí.

Literatura

[1] BERALDIN, J.-A., et. al.: Optimized Position Sensors for Flying-Spot Active Triangulation Systems. In *Proceedings of the Fourth International Conference on 3-D Digital Imaging and Modeling (3DIM).* Banff, Alberta, Canada. October 6-10, 2003. National Research Council of Canada, 2003. s. 334-341.

[2] BESL, P.J. and MCKAY, N.D. A method for registration of 3-d shapes. In *IEEE Transactions on Pattern Analysis and Machine Intelligence,* Feb 1992, vol. 14, no. 2, s. 239 - 256.

[3] BIBER, P. and STRASSER, W. The normal distributions transform: a new approach to laser scan matching. In *Proceedings of the IEEE International Conference on Intelligent Robots and Systems(IROS),* 2003, vol. 3, s. 2743 - 2748.

[4] BURGARD, W. and HÄHNEL, D. Probabilistic matching for 3d scan registration. In *Proc. of the VDI-Conference Robotik,* 2002.

[5] EVERETT, H. *Sensors for mobile Robots.* Canada, 1995, 528 s. ISBN 1-56881-048-2.

[6] LUAN, X. *Experimental Investigation of Photonic Mixer Device and Development of TOF 3D Ranging Systems Based on PMD Technology.* University of Siegen. DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE, 2001. 136 s., Vedoucí disertační práce Prof. Rudolf Schwarte

[7] MAGNUSSON, M. *The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection.* Örebro University, Örebro Studies in Technology, 2009. 201 s., Vedoucí disertační práce Achim Lilienthal.

[8] NEWCOMBE, R.A. et.al. *KinectFusion: Real-Time Dense Surface Mapping and Tracking*, ISMAR 2011

[9] RUSU, R.B. *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments.* München: Technische Universität. Institut für Informatik, 2009. 284 s., Vedoucí disertační práce Univ.-Prof. Nassir Navab, Ph.D.

[10] TŘÍSKA, V. Komprese shlukových dat s využitím geometrického dělení prostoru. In *Počítačové architektury & diagnostika 2012*. Praha: ČVUT, 2012. s. 127-132, ISBN 978-80-01-05106-1

[11] WEBB, J. and ASHLEY, J. *Beginning Kinect programming with the Microsoft Kinect SDK*

[12] *Point Cloud Library.* URL: http://www.pointclouds.org/.

Seznam obrázků

Seznam kódů

Seznam příloh

tocchapterSeznam příloh CD

na CD jsou adresáře

/text - diplomová práce ve formátu pdf

/zdrojovy kod - zdrojový kód programu pro zpracování dat

/pokoj1 - naměřená data a výsledek ve formátu vtk a pcl

/pokoj2 - naměřená data a výsledek ve formátu vtk a pcl

/kuchyn - naměřená data a výsledek ve formátu vtk a pcl

/chodba - naměřená data a výsledek ve formátu vtk a pcl