VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV MATEMATIKY
INSTITUTE OF MATHEMATICS

APLIKACE KALMANOVA FILTRU PRO ANALÝZU MRAČEN BODŮ

THE APPLICATION OF KALMAN FILTER IN POINT CLOUD DATA ANALYSIS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **Bc. KAMILA ŽENATÁ**
AUTHOR

VEDOUCÍ PRÁCE **Mgr. JANA PROCHÁZKOVÁ, Ph.D.**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav matematiky
Studentka:	**Bc. Kamila Ženatá**
Studijní program:	Matematické inženýrství
Studijní obor:	bez specializace
Vedoucí práce:	**Mgr. Jana Procházková, Ph.D.**
Akademický rok:	2024/25

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:

Aplikace Kalmanova filtru pro analýzu mračen bodů

Stručná charakteristika problematiky úkolu:

Kalmanův filtr je algoritmus, který umožňuje odhadovat stav dynamických systémů v čase, přičemž tento odhad je založen na neúplných, nepřesných a šumem ovl ivněných měřeních. Používá se často v robotice pro určování polohy vozidla, v teorii řízen í nebo v počítačovém vidění. V diplomové práci se student seznámí s matematickým pozadím Kalmanova fil tru a následně využije filtr pro predikci na datech daných mračny bodů.

Cíle diplomové práce:

1. Nastudování a zpracování teoretického pozadí Kalmanova filtru.
2. Práce s mračny bodů a jejich příprava pro aplikaci Kalmanova filtru (segmentace, identifikace částí).
3. Implementace Kalmanova filtru na reálná data získaná z mračna b odů pro predikci, zhodnocení řešení.

Seznam doporučené literatury:

GREWAL, Mohinder S. a Angus P. ANDREWS. Kalman filtering: theory an d practice using MATLAB. 3rd ed. Hoboken: Wiley, 2008. ISBN 978-0-470-17366-4.

BECKER, Alex. Kalman filter: From the ground up. 2. Kalmanfilter.net, 2023. ISBN 978-965-93120-1-6.

ZHANG, Yihuan, Jun WANG, Xiaonian WANG a John M. DOLAN. Road-Segmentation-Based Curb Detection Method for Self-Driving via a 3D-LiDAR Sensor. IEEE Transactions on Intelligent Transportation Systems [online]. 19(12), 3981-3991, 2018. [cit. 2024-09-10]. ISSN 1524-9050. Dostupné z: doi:10.1109/TITS.2018.2789462.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2024/25

V Brně, dne

L. S.

doc. Mgr. Petr Vašík, Ph.D. ředitel ústavu	doc. Ing. Jiří Hlinka, Ph.D. děkan fakulty

Abstrakt

Diplomová práce se zabývá pojmem Kalmanovy filtrace a jejím využitím v mračnech bodů. Práce poskytuje základní pojmy spojené s mračny bodů, Kalmanovým filtrem a praktické ukázky na konkrétních datech.

Abstract

This master's thesis deals with concept of Kalman filtering and its use in point clouds. The thesis provides basic concepts related to point clouds, Kalman filter and practical examples on specific data.

Klíčová slova

mračna bodů, Kalmanův filtr, RANSAC, clustering

Keywords

point cloud, Kalman filter, RANSAC, clustering

Citace

ŽENATÁ, Kamila. *Aplikace Kalmanova filtru pro analýzu mračen bodů*. Brno, 2025. Diplomová práce. Vysoké učení technické v Brně, Fakulta strojního inženýrství. Vedoucí práce Mgr. Jana Procházková, Ph.D.

Prohlašuji, že jsem tuto diplomovou práci vypracovala samostatně pod vedením Mgr. Jany Procházkové, Ph.D. a všechny využité materiály jsou uvedeny v seznamu literatury.

Kamila Ženatá

Ráda bych zde poděkovala své vedoucí Mgr. Janě Procházkové, Ph.D. za ochotu, trpělivost a cenné rady při vzniku této práce.

Kamila Ženatá

Obsah

Kapitola 1

Úvod

Kalmanův filtr, vyvinutý Rudolfem E. Kalmanem v roce 1960, slouží k odhadu neznámých stavů lineárních dynamických systémů na základě naměřených dat. Tento filtr představuje algoritmus s minimální chybovou odchylkou a poskytuje tak optimální odhad neznámého stavu, zohledňující náhodné chyby v naměřených datech. Kalmanova filtrace je široce využívaný proces v mnoha oblastech, kde je třeba propojit model systému s reálnými daty. Často hraje klíčovou roli v případě, že je potřeba získat odhad budoucí pozice na základě předchozích. Toho se využívá pro navigaci a lokaci například u letadel a dronů, či autonomních vozidel.

V první teoretické části se nejprve zaměříme na mračna bodů. Přiblížíme o co se jedná a pojmy s nimi související. Nejčastěji jsou tato data získávána pomocí LiDARu, jehož princip snímání bude v práci také vysvětlen. Dále uvedeme jaké formáty takto získaná data mohou mít, jaké informace takovéto soubory obsahují a samozřejmě k čemu jsou mračna bodů využívána v mnoha oblastech.

Druhá část je pak zaměřena na teorii spojenou s Kalmanovým filtrem. Uvedeme si model Kalmanova filtru, poté odvodíme jednotlivé rovnice algoritmu a pro snadné pochopení vysvětlíme jeho fungování na jednoduchém příkladě. Na konec této kapitoly uvedeme některé praktické příklady využití Kalmanova filtru.

V poslední praktické části budeme pracovat s několika typy datasetů, ve kterých vždy budeme mít za úkol detekci obrubníků. První krok práce je zpracování vstupních mračen bodů a detekce bodů potřebných pro nasazení Kalmanova filtru. Následně na získaná data vždy použijeme námi implementovaný algoritmus Kalmanova filtru. Pro porovnání bude vždy na stejná data použit i algoritmus implementovaný v Matlabu. V poslední části jsou obě implementace porovnány z hlediska přesnosti.

Kapitola 2

Mračna bodů

Mračno bodů, nebo-li point cloud, je neuspořádaná množina bodů v euklidovském prostoru. Každý bod této množiny je v $\mathbb{R}^3$ určen svými souřadnicemi (x, y, z), ale může obsahovat i další informace jako je např. barva nebo intenzita. Tato mračna vznikají procesem 3D skenování a to zejména laserovým skenováním. V některých aplikacích je mračno bodů vytvářeno z fotografií pomocí fotogrammetrie. Pro strojírenské aplikace se používají často optické skenery.

2.1 Laserové skenování

Laserové skenování, jak již název napovídá, funguje na principu vysílání laserového paprsku, který je vysílán z hlavy skeneru. Po kontaktu paprsku s povrchem skenovaného objektu se paprsek odrazí zpět ke skeneru. Následně přístroj vypočítá vzdálenost a úhel dopadu a z nich zaznamená prostorovou informaci konkrétního daného bodu.

Pro vytvoření podrobného 3D modelu objektu je často pořízeno několik skenů z různých úhlů a pozic pro zachycení všech detailů objektu. Takto pořízená mračna se následně musejí sesadit do jednoho mračna představující povrch skenovaného objektu. Tento výsledný point cloud může být dále použit pro vytvoření sítě povrchu nebo 3D modelu objektu.

Tato metoda skenování je velice rychlá, kdy nejrychlejší modely jsou schopny zaznamenat až 2 000 000 bodů za sekundu[1]. Právě rychlost je jednou z jejích největších výhod spolu s jednoduchostí sběru dat.

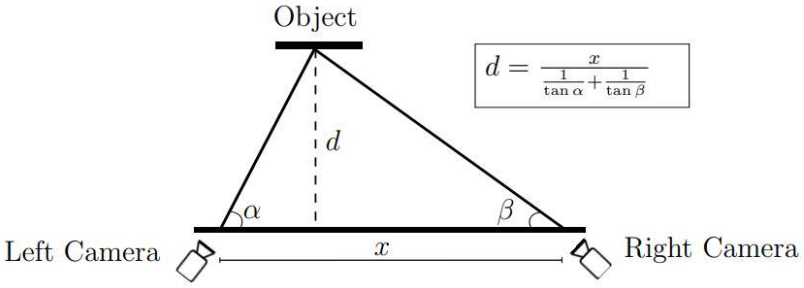
Laserové skenery se dají rozřadit do několika kategorií a to podle způsobu sběru dat na skenování z místa, skenování za chůze nebo jízdy a skenování ze vzduchu.

2.2 Fotogrammetrie

Jedná se pasivní metodu 3D skenování, kdy využíváme pouze obraz pořízený pomocí kamery. Fotogrammetrie pak rekonstruuje původní 3D objekty z daných fotografických snímků. Počet potřebných snímků pro dobrou rekonstrukci se liší v závislosti na tom, jak komplexní je snímaný objekt, kvalitě fotografií a kvalitě fotoaparátu. Zároveň by se fotografie měly postupně překrývat minimálně 60% svého obsahu, jak je uvedeno v [15]. Zde je tento údaj uveden pro vzdušné pořizování snímků, avšak tento údaj platí i obecně. Uveďme si nyní dvě techniky fotogrammetrického zobrazení.

[1] https://www.gp-radar.com/article/how-much-can-be-3d-laser-scanned-in-a-day

První z metod, kterou zde uvedeme, je pasivní triangulace. Pojem pasivní v tomto případě znamená, že zdrojem informací jsou data ze snímku bez pomocného zdroje světla. Základ pro pasivní triangulaci je trojice bodů a to konkrétně dva body reprezentující snímací zařízení a snímaný bod, jak je vidět na obrázku 2.1.



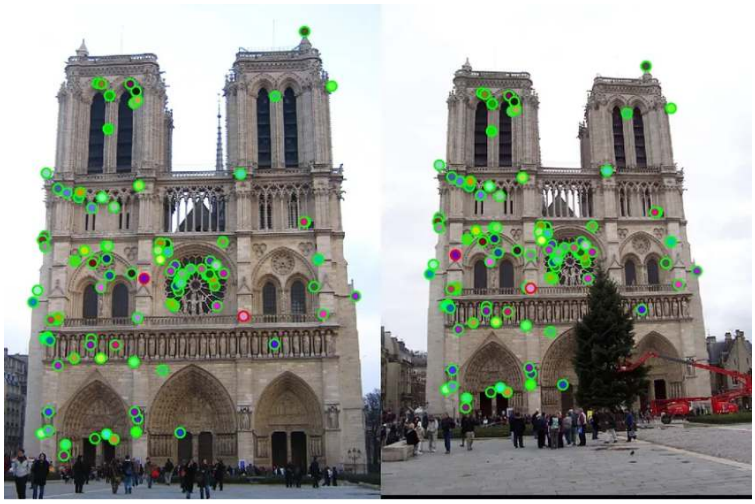
Obrázek 2.1: Princip fungování pasivní triangulace, [6]

Pro zaručení správného měření je samozřejmě třeba, aby byl každý měřený bod zachycen oběma kamerami. Ovšem k určení polohy bodu v prostoru je nutné znát parametry a vzájemné vzdálenosti snímajících kamer, což může být popsáno pomocí projekční matice. Ze znalosti vzájemných poloh snímacích zařízení můžeme určit prostorové souřadnice bodu. Tyto údaje je tedy nutné vždy znát předem nebo zjistit v průběhu procesu kalibrace.

Structure from motion (SfM) [24] je druhá technika fotogrammetrického zobrazení, kterou zde uvedeme. Jedná se o proces přibližné rekonstrukce 3D struktur pomocí sady 2D obrazů. Na rozdíl od pasivní triangulace, structure from motion dovoluje výpočet projekčních matic a 3D bodů zároveň a to pouze za pomoci odpovídajících bodů v každém pohledu. Typické kroky, jak se postupuje u této metody jsou:

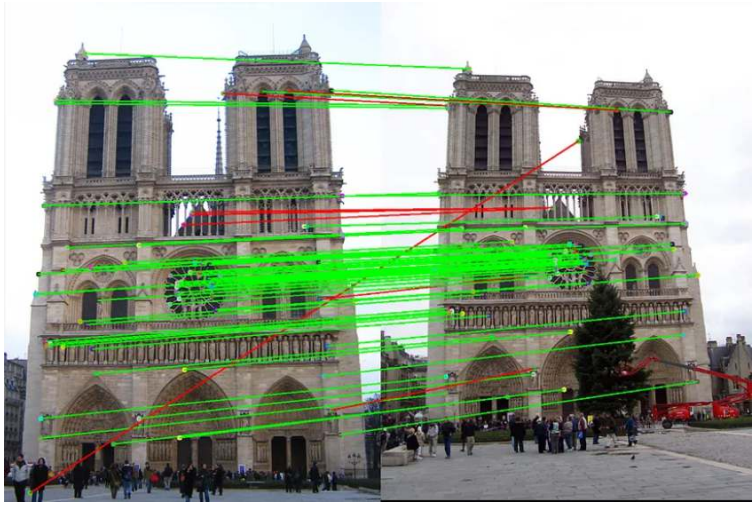
1. feature extraction
2. feature matching
3. 3D reconstruction
4. bundle adjustment

Jako první potřebujeme detekovat klíčové body v každém snímku, což mohou být rohy nebo tzv. SIFT body. SIFT (Scale-Invariant Feature Transform) je algoritmus pro detekci, popis a spojení rysů v obrazech. Minimální počet klíčových bodů, který je nutný, je 8 bodů. Tento proces se nazývá feature extraction a příklad můžeme vidět na obrázku 2.2.



Obrázek 2.2: Feature extraction, [17]

Jako další krok je nutné přiřadit odpovídající si klíčové body v jednotlivých pohledech, což se nazývá feature matching a vidíme jej na obrázku 2.3. Z těchto dvojic je poté rekonstruováno výsledné mračno bodů.



Obrázek 2.3: Feature matching, [17]

2.3 Optické skenování

Tato metoda získávání mračen bodů je velice přesná a je tedy velmi vhodná v oblastech, kde je přesnost kritická, ať ve strojírenství či stavebnictví. Další výhodou může být i bezkontaktní sběr informací. Skener se nijak nedotýká skenovaného objektu, je tedy bezpečné i pro skenování křehkých objektů či nebezpečných a obtížně dostupných míst. Zároveň již při sběru a zpracování dat získáváme komplexní dokumentaci objektu.

2.4 LiDAR (Light Detection And Ranging)

LiDAR, též někdy známý jako LADAR, je technologie používaná pro laserové skenování. Je to metoda měření vzdálenosti na základě výpočtu času odrazu laserového paprsku od snímaného objektu. Tedy laser vydává světelné pulzy a detekuje světlo odražené od objektu. Senzor poté změří čas mezi vyzářením a návratem pulzu a uraženou vzdálenost. Můžeme napsat, že pro uraženou vzdálenost d platí:

$$d = \frac{c \cdot \Delta t}{2},$$

kde c je rychlost světla a t je čas.

LiDAR je možné používat, jak na zemi, tak ve vzduchu. Podle tohoto dělení máme tři typy LiDAR systémů a to pozemní, vzdušný a satelitní. Vzdušný LiDAR je využíván, jak již název napovídá, pro shromažďování dat ze vzduchu prostřednictvím letadel či dronů. Pozemní pak může být instalován například na pohybující se vozidlo nebo na statické stativy. Satelitní LiDAR pak bývá upevněn na satelity obíhající kolem Země. Tyto satelitní systémy pokrývají velké oblasti, avšak s menším množstvím detailů.

Využití LiDARu je rozmanité. Ať se jedná o skenování mostů a budov pro zajištění kvality stavby, či v atmosférických vědách. LiDAR může být také použit pod vodou například pro mapování mořského dna. V tomto případě se samozřejmě podvodní LiDAR liší od pozemního a to konkrétně použitím zelené vlnové délky.

Zajímavým využitím v praxi je situační povědomí v souvislosti s autonomním řízením. V tomto případě je nutné, aby měl systém dobrý přehled o všech stojících i pohybujících se objektech ve svém okolí, a přesně to LiDAR umožňuje. Paprsky jsou vysílány v různých úhlech a je vytvořeno mračno, které reprezentuje okolí. LiDAR velice přesně a také rychle určuje vzdálenost k okolím objektům, což při pohybujících se objektech je klíčové.

2.4.1 Hovermap ST-X

Část testovacích dat použitých v této práci byla pořízena za použití Hovermap ST-X. Tento LiDAR můžeme vidět na obrázcích 2.4. V následující tabulce 2.1 jsou pak vypsány některé jeho mapovací a fyzické specifikace.

Obrázek 2.4: Hovermap ST-X.

Rozsah LiDARu	0,5 až 300 m
Přesnost mapování	±15 mm v běžném prostředí, ±10 mm ve vnitřních a podzemních prostorech ±5 mm schopnost izolované detekce změn
Kanály LiDARu	32
Rychlost pořizování dat	Režim jednoho návratu: až 640 000 bodů za sekundu, Režim více návratů (3 návraty): až 1 920 000 bodů za sekundu
Úložiště	512 gigabajtů - přibližně 4 hodiny dat ze senzorů
Váha	1,57 kg

Tabulka 2.1: Specifikace LiDARu.

2.5 Formáty dat

Výstupem z 3D skenování jsou tedy mračna bodů, která mohou být uložena v různých formátech, jako je .ply, .pcd, .stl, .csv a další. My budeme pracovat pouze s daty ve formátu .ply, zaměřme se proto nyní na něj.

2.5.1 Formát .ply

Formát .ply, známý také jako Polygon File Format, byl navržen především pro ukládání trojrozměrných struktur získaných z 3D skenování. Tento soubor začíná vždy záhlavím, které obsahuje informace o prvcích tohoto souboru. Na dalším řádku je pak specifikováno o který .ply formát se jedná, přičemž by se mělo jednat o jeden z následujících:

- `format ascii 1.0,`
- `format binary_little_endian 1.0,`
- `format binary_big_endian 1.0.`

Poté následuje řádek s klíčovým slovem element, který nám říká, jak přesně jsou data uložena a kolik jich soubor obsahuje. Pro příklad se podívejme na hlavičku jednoho souboru.

```
ply
format ascii 1.0
element vertex 6439024
property double x
property double y
property double z
property ushort red
property ushort green
property ushort blue
property ushort intensity
end_header
```

Tento soubor tedy obsahuje 6 439 024 vrcholů, kde každý je reprezentován souřadnicemi (x, y, z), barvou, pomocí červené, modré a zelené, a intenzitou.

2.6 Využití mračen bodů v praxi

Mračna bodů mají v praxi široké spektrum využití. Umožňují například vizualizaci skenovaných objektů nebo 3D scén, což následně umožňuje jejich detailnější zkoumání a analýzu.

Další možností jejich využití je rekonstrukce a modelování. Mračna bodů často slouží jako vstupní data pro tvorbu různých 3D modelů objektů či terénu. Konkrétně je možné je využít například při rekonstrukci modelů budov pro jejich další zpracování, jak je uvedeno v [3] a [21]. Tyto modely nacházejí uplatnění mimo jiné v architektuře, inženýrství, počítačové grafice a mnoha dalších oblastech. Podobně mohou být využity i ve strojírenství, např. při rekonstrukci CAD modelů z mračen bodů [20], [16].

Mračna bodů rovněž umožňují analýzu povrchu, jako je detekce nerovností, měření vodorovnosti či posouzení rovinnosti. Tato funkce je často využívána pro inspekci kvality povrchu, jak je popsáno v [19].

V neposlední řadě mají mračna bodů uplatnění v geodézii a stavebnictví, kde umožňují přesná měření vzdáleností, objemů, ploch a dalších geometrických parametrů.

Kapitola 3

Kalmanův filtr

Tento filtr nese jméno maďarském matematika Rudolfa Emila Kálmána (19.květen 1930 - 2.červenec 2016) žijícího ve Spojených státech amerických. Jak je uvedeno v [9] v roce 1958 dostal jednoho večera Kalman nápad použít stavové proměnné na problém Wienerovy filtrace. Tato myšlenka se později přeměnila právě v Kalmanův filtr, který Kalman poprvé publikoval v roce 1960 v článku [11].

Kalmanův filtr poskytuje odhady neznámých proměnných na základě předchozích pozorování. Je velice důležité si uvědomit, že algoritmus má v každém kroku k dispozici pouze hodnoty předcházející a z nich predikuje následující bod. Poté co se "dozví" skutečnou novou pozici, svůj odhad opraví a za jeho pomoci odhadne opět další pozici. Algoritmus tedy nemá ihned od počátku znalost všech skutečných pozic, ale získává je postupně, aby vždy opravil svůj aktuální odhad.

Jak již slovo filtr v názvu napovídá, jedná se o výpočty, které berou v úvahu i různé odchylky a možné chyby ať už měření, či např. vlivy prostředí. Pokud bychom při nějaké predikci dat nepřipouštěli možné chyby a pouze bychom brali výpočty na základě fyzikálních rovnic, mohlo by se stát, že predikce a realita by byly velice odlišné. Použitím Kalmanova filtru zohledňujeme v predikcích různé chyby měření či odchylky od výsledků fyzikálních rovnic vlivem vnějších jevů. Hlavními zdroji pro tuto část jsou [2], [9], [22], [4], [13].

3.1 Základní pojmy z pravděpodobnosti a statistiky

Připomeňme si na začátek některé pojmy ze statistiky, které se budou dále vyskytovat. Pro podrobnější zavedení viz např. [1].

Náhodná veličina X je funkce, která zobrazuje elementární jevy $\omega \in \Omega$ na reálná čísla. Náhodná veličina může být dvojího typu, a to spojitá, nebo diskrétní. V této práci bude dále použita pouze diskrétní náhodná veličina a pojmy s ní související, omezíme se proto v tomto odstavci pouze na ni. Diskrétní náhodná veličina je taková, která může nabývat jen konečně nebo spočetně mnoha různých hodnot. Dále zavádíme pojem **distribuční funkce** $F(X)$, kterou můžeme definovat jako pravděpodobnost, že náhodná proměnná X nabude hodnoty menší než určitá hodnota x, zapsané předpisem $F(X) = P(X < x)$.

Střední hodnota náhodné proměnné X je reálné číslo $E(X)$, kterou pro diskrétní náhodnou proměnnou vypočítáme

$$E(X) = \sum_{x \in Z} x \cdot p(x),$$

kde $p(x)$ je pravděpodobnostní funkce diskrétní náhodné veličiny X. Pomocí středních hodnot můžeme dále definovat pojem **rozptyl** jako reálné číslo $D(X) = E(X-E(X))^2$. Důležité dva pojmy, které budeme dále potřebovat jsou **korelace** a **kovariance**. Kovariance dvou náhodných proměnných je definována jako

$$cov(X, Y) = E[(X - E[X])(Y - E[Y])].$$

Korelace vyjadřuje vzájemnou závislost dvou náhodných veličin

$$R(X, Y) = \frac{E[(X - E[X])(Y - E[Y])]}{\sqrt{D(X)}\sqrt{D(Y)}}.$$

3.2 Model procesu

Model procesu pro časově-diskrétní Kalmanův filtr definuje změny ze stavu v čase $k-1$ do stavu v čase k:

$$x_k = F_{k-1}x_{k-1} + B_{k-1}u_{k-1} + w_{k-1}, \tag{3.1}$$

kde F_k je matice přechodu mezi stavy, B_k je řídící vstupní matice aplikovaná na řídící vektor u_k a w_k je procesní šum. O vektoru procesního šumu navíc předpokládáme, že je Gaussův s nulovou střední hodnotou a kovariancí Q tedy $w_k \sim N(0, Q_k)$. Velice často se však stává, že řídicí vektor nemáme. Rovnice se nám tedy zjednoduší na tvar:

$$x_k = F_{k-1}x_{k-1} + w_{k-1}, \tag{3.2}$$

kde všechny koeficienty mají stejný význam, jak je uvedeno výše.

Dále zavádíme model měření, který popisuje vztah mezi stavem a měřením v čase k:

$$z_k = H_k x_k + v_k, \tag{3.3}$$

kde z_k je vektor měření, H_k je matice měření a v_k je vektor šumu měření, o kterém předpokládáme, že je Gaussův s nulovou střední hodnotou a kovariancí R_k tedy $v_k \sim N(0, R_k)$.

3.3 Střední kvadratická chyba

Naším cílem při filtraci je odhad stavu systému v čase k. Rozdíl mezi odhadem $\hat{x}_k$ a skutečnou hodnotou x_k pak nazýváme chybou

$$f(e_k) = f(x_k - \hat{x}_k).$$

Tvar funkce $f(e_k)$ závisí na aplikaci, avšak měla by být kladná a monotónně rostoucí. Funkce splňující tyto vlastnosti je funkce kvadratické chyby

$$f(e_k) = (x_k - \hat{x}_k)^2.$$

Vzhledem k tomu, že je třeba vzít v úvahu schopnost filtru předpovídat hodně údajů za určité období, je vhodnější uvažovat očekávanou hodnotu chybové funkce

$$loss\_function = E(f(e_k)).$$

Toto vše nakonec vede ke střední kvadratické chybě (mean squarred error, MSE)

$$\epsilon(t) = E(e_k^2). \tag{3.4}$$

3.4 Odvození algoritmu

Mějme proces zadán modelem, který je uveden výše (rovnice (3.2), (3.3)), tedy lineární dynamický model diskrétní v čase

$$x_k = F_{k-1}x_{k-1} + w_{k-1},$$
$$z_k = H_k x_k + v_k,$$

kde w_{k-1} je procesní šum a v_k je vektor šumu měření. Předpokládáme, že kovariance obou šumů v našem modelu jsou stacionární v čase a dány vztahy

$$Q = E[w_k w_k^T],$$
$$R = E[v_k v_k^T].$$

Algoritmus Kalmanova filtru má dvě fáze a to jsou predikce a aktualizace. Naším cílem je najít odhad $\hat{x}_k$ skutečné hodnoty x_k tak, aby funkce střední kvadratické chyby byla minimální. Tento algoritmus je rekurzivní, což znamená, že využívá aposteriorní odhad $\hat{x}_{k-1(+)}$ v čase t_k k získání apriorního odhadu $\hat{x}_{k+1(-)}$ v čase t_{k+1}

$$\hat{x}_{k(-)} = F_{k-1}\hat{x}_{k-1(+)}. \tag{3.5}$$

Pro aposteriorní odhad požadujeme, aby byl optimální, tedy minimalizoval střední kvadratickou chybu. Budeme předpokládat, že x a z mají sdružené normální rozdělení a je tedy možné $\hat{x}_{k(+)}$ vyjádřit jako lineární funkci předchozího odhadu a měření z

$$\hat{x}_{k(+)} = K_k^1 \hat{x}_{k(-)} + K_k z_k, \tag{3.6}$$

kde $\hat{x}_{k(-)}$ je předchozí odhad x_k a $\hat{x}_{k(+)}$ aposteriorní odhad. Matice K_k^1, K_k zatím neznáme, ale chceme takové hodnoty těchto matic, aby $\hat{x}_{k(+)}$ splňovalo princip ortogonality

$$E[(x_k - \hat{x}_{k(+)})z_i^T] = 0,\ i = 1, 2, ..., k-1, \tag{3.7}$$
$$E[(x_k - \hat{x}_{k(+)})z_k^T] = 0. \tag{3.8}$$

Princip ortogonality je nutnou a postačující podmínkou optimality bayesovského odhadu. Tento princip nám říká, že chybový vektor optimálního odhadu, ve smyslu minimalizace střední kvadratické chyby, je ortogonální vzhledem k libovolnému možnému odhadu. Podrobněji viz např. [18].

Z rovnic (3.2), (3.3) můžeme vidět, že data $z_1, ..., z_k$ neobsahují šum w_k. Tedy jelikož náhodné vektory **w** a **v** jsou nekorelované, vyplývá, že $Ew_k z_i^T = 0$ pro $1 \leq i \leq k$. Pokud nyní dosadíme za x_k z rovnice (3.2) a za $\hat{x}_{k(+)}$ z rovnice (3.6) do rovnice (3.7) dostaneme následující

$$E[(F_{k-1}x_{k-1} + w_{k-1} - K_k^1 \hat{x}_{k(-)} - K_k z_k)z_i^T] = 0. \tag{3.9}$$

Využijeme-li výše uvedeného, že $Ew_k z_i^T = 0$ rovnice (3.9) se upraví na tvar

$$E[(F_{k-1}x_{k-1} - K_k^1 \hat{x}_{k(-)} - K_k z_k)z_i^T] = 0. \tag{3.10}$$

Podobně bude platit, že $Ev_k z_i^T = 0$ pro $i = 1, 2, ..., k-1$ a rovnice (3.10) se tak postupně upraví

$$F_{k-1}E[x_{k-1}z_i^T] - K_k^1 E[\hat{x}_{k(-)}z_i^T] - K_k H_k E[x_k z_i^T] - K_k E[v_k z_i^T] = 0,$$
$$F_{k-1}E[x_{k-1}z_i^T] - K_k^1 E[\hat{x}_{k(-)}z_i^T] - K_k H_k F_{k-1} E[x_{k-1}z_i^T] = 0, \tag{3.11}$$

kde jsme za x_k dosadili $F_{k-1}x_{k-1}$ a využili $Ev_k z_i^T = 0$. Princip ortogonality (rovnice (3.7), (3.8)) platí i pro apriorní odhad $\hat{x}_{k(-)}$, tedy

$$E[(x_k - \hat{x}_{k(-)})z_i^T] = 0,\ i = 1, 2, ..., k-1.$$

S využitím tohoto můžeme dále postupně rovnici (3.11) upravit

$$\begin{aligned} F_{k-1}Ex_{k-1}z_i^T - K_k^1 F_{k-1}Ex_{k-1}z_i^T - K_k H_k F_{k-1}Ex_{k-1}z_i^T = 0, \\ (I - K_k^1 - K_k H_k)Ex_{k-1}z_i^T = 0. \end{aligned} \tag{3.12}$$

Pro libovolné x_k bude rovnice (3.12) splněna, pokud

$$K_k^1 = I - K_k H_k. \tag{3.13}$$

Nyní je třeba nalézt matici K_k tak, aby byla splněna druhá podmínka principu ortogonality (3.8).

Zaveďme následující značení chyb

$$\tilde{x}_{k(+)} = \hat{x}_{k(+)} - x_k, \tag{3.14}$$

$$\tilde{x}_{k(-)} = \hat{x}_{k(-)} - x_k, \tag{3.15}$$

$$\tilde{z}_k = \hat{z}_{k(-)} - z_k = H_k \hat{x}_{k(-)} - z_k. \tag{3.16}$$

Vektory $\tilde{x}_{k(+)}, \tilde{x}_{k(-)}$ jsou chyby odhadu před a po korekci. Rovnici (3.8) můžeme napsat také jako

$$E[(x_k - \hat{x}_{k(+)})z_{k(-)}^T] = 0. \tag{3.17}$$

Odečtením rovnice (3.8) od rovnice (3.17) získáme

$$E[(x_k - \hat{x}_{k(+)})\tilde{z}_k^T] = 0. \tag{3.18}$$

Dosadíme-li postupně za $x_k, \hat{x}_{k(+)}$ a $\tilde{z}_k$ z rovnic (3.2), (3.6) a (3.16), pak získáme

$$E[(F_{k-1}x_{k-1} + w_{k-1} - K_{k(-)}^1 - K_k z_k)(H_k \hat{x}_{k(-)} - z_k)] = 0.$$

Zároveň však platí

$$E[w_k z_k^T] = E[w_k \hat{x}_{k(+)}] = 0,$$

a tedy

$$E[F_{k-1}x_{k-1} - K_k^1 \hat{x}_{k(-)} - K_k z_k][H_k \hat{x}_{k(-)} - z_k] = 0. \tag{3.19}$$

Pokud nyní do rovnice (3.19) dosadíme z (3.13) za K_k^1, z rovnice (3.3) za z_k a ze vztahu (3.15) za $\tilde{x}_{k(-)}$ postupnými úpravami získáme

$$\begin{aligned} E[(F_{k-1}x_{k-1} - \hat{x}_{k(-)} + K_k H_k \hat{x}_{k(-)} - K_k H_k x_k - K_k v_k)(H_k \hat{x}_{k(-)} - H_k x_k - v_k)^T] = 0 \\ E[((x_k - \hat{x}_{k(-)}) + K_k H_k (\hat{x}_{k(-)} - x_k) - K_k v_k)(H_k \tilde{x}_{k(-)} - v_k)^T] = 0 \\ E[(-\tilde{x}_{k(-)} + K_k H_k \tilde{x}_{k(-)} - K_k v_k)(H_k \tilde{x}_{k(-)} - v_k)^T] = 0. \end{aligned} \tag{3.20}$$

Z definice apriorní chybová kovarianční matice

$$P_{k(-)} = E[\tilde{x}_{k(-)} \tilde{x}_{k(-)}^T].$$

Dosazením $P_{k(-)}$ a postupnými úpravami vztahu (3.20) dostaneme

$$(I - K_k H_k) P_{k(-)} H_k^T - K_k R_k = 0.$$

Tudíž matici K_k, která se nazývá Kalmanův zisk (Kalman gain) můžeme vyjádřit

$$K_k = P_{k(-)} H_k^T (H_k P_{k(-)} H_k^T + R_k)^{-1}. \tag{3.21}$$

Nyní budeme chtít odvodit vztah pro aposteriorní odhad kovarianční matice $P_{k(+)}$ definovanou podobně jako apriorní vztahem

$$P_{k(+)} = E[\tilde{x}_{k(+)} \tilde{x}_{k(+)}^T]. \tag{3.22}$$

Dosazením rovnice (3.13) do (3.6) získáme rovnici pro aposteriorní odhad $\hat{x}_{k(+)}$

$$\begin{aligned} \hat{x}_{k(+)} &= (I - K_k H_k) \hat{x}_{k(-)} + K_k z_k, \\ \hat{x}_{k(+)} &= \hat{x}_{k(-)} + K_k (z_k - H_k \hat{x}_{k(-)}) \end{aligned} \tag{3.23}$$

Odečtením x_k od obou stran rovnice (3.23) dostaneme

$$\begin{aligned} \hat{x}_{k(+)} - x_k &= \hat{x}_{k(-)} + K_k H_k x_k + K_k v_k - K_k H_k \hat{x}_{k(-)} - x_k, \\ \tilde{x}_{k(+)} &= \tilde{x}_{k(-)} - K_k H_k \tilde{x}_{k(-)} + K_k v_k, \\ \tilde{x}_{k(+)} &= (I - K_k H_k) \tilde{x}_{k(-)} + K_k v_k. \end{aligned} \tag{3.24}$$

Pokud využijeme toho, že $E[\tilde{x}_{k(-)} v_k^T] = 0$ a dosadíme (3.24) do (3.22) získáme

$$\begin{aligned} P_{k(+)} &= E[(I - K_k H_k) \tilde{x}_{k(-)} \tilde{x}_{k(-)}^T (I - K_k H_k)^T + K_k v_k v_k^T K_k^T] \\ P_{k(+)} &= (I - K_k H_k) P_{k(-)} (I - K_k H_k)^T + K_k R_k K_k^T \end{aligned} \tag{3.25}$$

Tato poslední rovnice je tzv. Josephův tvar aposteriorní kovarianční chybové matice. Pokud dosadíme za K_k z (3.21), rovnice se nám upraví na tvar

$$\begin{aligned} P_{k(+)} &= P_{k(-)} - K_k H_k P_{k(-)} - P_{k(-)} H_k^T K_k^T + K_k H_k P_{k(-)} H_k^T K_k^T + K_k R_k K_l^T, \\ P_{k(+)} &= (I - K_k H_k) P_{k(-)} - P_{k(-)} H_k^T K_k^T + K_k (H_k P_{k(-)} H_k^T + R_k) K_k^T, \\ P_{k(+)} &= (I - K_k H_k) P_{k(-)}. \end{aligned} \tag{3.26}$$

Poslední krok je určit vztah pro apriorní chybovou kovarianční matici. Z definice opět víme, že

$$P_{k(-)} = E[\tilde{x}_{k(-)} \tilde{x}_{k(-)}^T]. \tag{3.27}$$

Odečteme nyní x_k od obou stran rovnice (3.5)

$$\begin{aligned} \hat{x}_{k(-)} - x_k &= F_{k-1} \hat{x}_{k-1(+)} - x_k \\ \tilde{x}_{k(-)} &= F_{k-1} (\hat{x}_{k-1(+)} - x_{k-1}) - w_{k-1}, \\ \tilde{x}_{k(-)} &= F_{k-1} \tilde{x}_{k-1(+)} - w_{k-1}. \end{aligned} \tag{3.28}$$

Pokud nyní do (3.27) dosadíme (3.28) a využijeme $E[\tilde{x}_{k-1(+)} w_{k-1}^T] = 0$ získáme

$$\begin{aligned} P_{k(-)} &= F_{k-1} E[\tilde{x}_{k-1(+)} \tilde{x}_{k-1(+)}^T] F_{k-1}^T + E[w_{k-1} w_{k-1}^T], \\ P_{k(-)} &= F_{k-1} P_{k-1(+)} F_{k-1}^T + Q_{k-1}. \end{aligned} \tag{3.29}$$

3.5 Shrnutí algoritmu časově-diskrétního Kalmanova filtru

Rovnice, které jsme odvodili v předchozí sekci jsou shrnuty v této části.
Dynamický model systému:

$$x_k = F_{k-1}x_{k-1} + w_{k-1}$$
$$w_k \sim N(0, Q_k)$$

Model měření:

$$z_k = H_k x_k + v_k$$
$$v_k \sim N(0, R_k)$$

Apriorní odhad stavu:

$$\hat{x}_{k(-)} = F_{k-1}\hat{x}_{k-1(+)} \tag{3.30}$$

Apriorní odhad kovarianční chybové matice:

$$P_{k(-)} = F_{k-1}P_{k-1(+)}F_{k-1}^T + Q_{k-1} \tag{3.31}$$

Výpočet Kalmanova zisku:

$$K_k = P_{k(-)}H_k^T(H_k P_{k(-)}H_k^T + R_k)^{-1} \tag{3.32}$$

Aposteriorní odhad stavu:

$$\hat{x}_{k(+)} = \hat{x}_{k(-)} + K_k(z_k - H_k\hat{x}_{k(-)}) \tag{3.33}$$

Aposteriorní odhad chybové kovarianční matice:

$$P_{k(+)} = (I - K_k H_k)P_{k(-)} \tag{3.34}$$

Výpočty v (3.30) a (3.31) tvoří predikční část algoritmu a vztahy (3.33), (3.32) a (3.34) jsou pak korekční částí algoritmu.

3.6 Příklad v 1D

Naším úkolem je určit teplotu tekutiny v nádrži. Předpokládáme stálý stav a konstantní teplotu, avšak může zde docházet k mírným fluktuacím skutečné teploty tekutiny. Tento systém lze popsat rovnicí

$$x_k = T + w_k,$$

kde T je teplota tekutiny a w_k je procesní šum s variancí q.

Předpokládejme, že opravdová teplota tekutiny je $50°C$ a že model odpovídá skutečnosti, tedy rozptyl procesního šumu nastavíme $Q = 0.0001$. Šum měření je $0.1°C$ a měření probíhá každých 5 sekund. Skutečné a naměřené hodnoty v jednotlivých časech můžeme vidět v následující tabulce.

Číslo měření	Skutečná teplota	Naměřená teplota
1	$50,005°C$	$49,986°C$
2	$49,994°C$	$49,963°C$
3	$49,993°C$	$50,090°C$
4	$50,001°C$	$50,001°C$
5	$50,006°C$	$50,018°C$
6	$49,998°C$	$50,050°C$
7	$50,021°C$	$49,938°C$
8	$50,005°C$	$49,858°C$
9	$50,000°C$	$49,965°C$
10	$49,997°C$	$50,114°C$

Nultá iterace

Před provedením první iterace je nutné inicializovat Kalmanův filtr a predikovat následující stav, tedy první iteraci. Skutečnou teplotu tekutiny neznáme, náš odhad bude tedy:

$$\hat{x}_{0(+)} = 60°C.$$

Náš prvotní odhad bude nepřesný, nastavíme proto chybu inicializačního odhadu (σ) na 100. Odhad rozptylu inicializace je rozptyl chyby (σ^2), tedy:

$$P_{0(+)} = 10000.$$

Vzhledem k tomu, že dynamika našeho modelu je konstantní, tedy teplota je konstantní, predikovaný odhad je roven aktuálnímu odhadu:

$$\hat{x}_{1(-)} = 60°C.$$

Rozptyl extrapolovaného odhadu je potom:

$$P_{1(-)} = P_{0(+)} + Q = 10000 + 0,0001 = 10000,0001.$$

První iterace

Z tabulky můžeme vidět, že první naměřená hodnota je:

$$z_1 = 49,986°C.$$

Jelikož chyba měření je 0,1, rozptyl je 0,01, tedy rozptyl měření je potom:

$$R_1 = 0,01.$$

Kalman gain vypočítáme následujícím způsobem:

$$K_1 = \frac{P_{1(-)}}{P_{1(-)} + R_1} = \frac{10000,0001}{10000,0001 + 0,01} = 0,999999.$$

Odhad aktuálního stavu:

$$\hat{x}_{1(+)} = \hat{x}_{1(-)} + K_1(z_1 - \hat{x}_{1(-)}) = 60 + 0.999999(49,986 - 60) = 49.986°C.$$

Aktualizace aktuálního odhadu rozptylu:

$$P_{1(+)} = (1 - K_1)P_{1(-)} = (1 - 0,999999)10000,0001 = 0,01.$$

Opět jelikož předpokládáme, že teplota se nemění tak získáváme:

$$\hat{x}_{2(-)} = \hat{x}_{1(+)} = 49,986°C.$$

Rozptyl extrapolovaného odhadu je potom:

$$P_{2(-)} = P_{1(+)} + Q = 0,01 + 0,0001 = 0,0101.$$

Druhá iterace

Druhá naměřená hodnota je:

$$z_2 = 49,963°C.$$

Chyba měření je 0,1 a rozptyl měření je tedy opět:

$$R_2 = 0,01.$$

Výpočet Kalman gain:

$$K_2 = \frac{P_{2(-)}}{P_{2(-)} + R_2} = \frac{0,0101}{0,0101 + 0,01} = 0,5.$$

Odhad aktuálního stavu:

$$\hat{x}_{2(+)} = \hat{x}_{2(-)} + K_2(z_2 - \hat{x}_{2(-)}) = 49,986 + 0,5(49,963 - 49,986) = 49,974°C.$$

Aktualizace aktuálního odhadu rozptylu:

$$P_{2(+)} = (1 - K_2)P_{2(-)} = (1 - 0,5)0,0101 = 0,005.$$

Teplota se v čase nemění, takže získáváme:

$$\hat{x}_{3(-)} = \hat{x}_{2(+)} = 49,974°C.$$

Rozptyl extrapolovaného odhadu je potom:

$$P_{3(-)} = P_{2(+)} + Q = 0,005 + 0,0001 = 0,0051.$$

Třetí iterace

$$\begin{aligned}
z_3 &= 50,090°C \\
R_3 &= 0,01 \\
K_3 &= \frac{P_{3(-)}}{P_{3(-)} + R_3} = \frac{0,0051}{0,0051 + 0,01} = 0,3388 \\
\hat{x}_{3(+)} &= \hat{x}_{3(-)} + K_3(z_3 - \hat{x}_{3(-)}) = 49,974 + 0,3388(50,090 - 49,974) = 50,016°C \\
P_{3(+)} &= (1 - K_3)P_{3(-)} = (1 - 0,3388)0,0051 = 0,0034 \\
\hat{x}_{4(-)} &= \hat{x}_{3(+)} = 50,016°C \\
P_{4(-)} &= P_{3(+)} + Q = 0,0034 + 0,0001 = 0,0035
\end{aligned}$$

Čtvrtá iterace

$$
\begin{aligned}
z_4 &= 50,001°C \\
R_4 &= 0,01 \\
K_4 &= \frac{P_{4(-)}}{P_{4(-)} + R_4} = \frac{0,0035}{0,0035 + 0,01} = 0,2586 \\
\hat{x}_{4(+)} &= \hat{x}_{4(-)} + K_4(z_4 - \hat{x}_{4(-)}) = 50,016 + 0,2586(50,001 - 50,016) = 50,012°C \\
P_{4(+)} &= (1 - K_4)P_{4(-)} = (1 - 0,2586)0,0035 = 0,0026 \\
\hat{x}_{5(-)} &= \hat{x}_{4(+)} = 50,012°C \\
P_{5(-)} &= P_{4(+)} + Q = 0,0026 + 0,0001 = 0,0027
\end{aligned}
$$

Pátá iterace

$$
\begin{aligned}
z_5 &= 50,018°C \\
R_5 &= 0,01 \\
K_5 &= \frac{P_{5(-)}}{P_{5(-)} + R_5} = \frac{0,0027}{0,0027 + 0,01} = 0,2117 \\
\hat{x}_{5(+)} &= \hat{x}_{5(-)} + K_5(z_5 - \hat{x}_{5(-)}) = 50,012 + 0,2117(50,018 - 50,012) = 50,013°C \\
P_{5(+)} &= (1 - K_5)P_{5(-)} = (1 - 0,2117)0,0027 = 0,0021 \\
\hat{x}_{6(-)} &= \hat{x}_{5(+)} = 50,013°C \\
P_{6(-)} &= P_{5(+)} + Q = 0,0021 + 0,0001 = 0,0022
\end{aligned}
$$

Šestá iterace

$$
\begin{aligned}
z_6 &= 50,050°C \\
R_6 &= 0,01 \\
K_6 &= \frac{P_{6(-)}}{P_{6(-)} + R_6} = \frac{0,0022}{0,0022 + 0,01} = 0,1815 \\
\hat{x}_{6(+)} &= \hat{x}_{6(-)} + K_6(z_6 - \hat{x}_{6(-)}) = 50,013 + 0,1815(50,050 - 50,013) = 50,020°C \\
P_{6(+)} &= (1 - K_6)P_{6(-)} = (1 - 0,1815)0,0022 = 0,0018 \\
\hat{x}_{7(-)} &= \hat{x}_{6(+)} = 50,020°C \\
P_{7(-)} &= P_{6(+)} + Q = 0,0018 + 0,0001 = 0,0019
\end{aligned}
$$

Sedmá iterace

$$
\begin{aligned}
z_7 &= 49,938°C \\
R_7 &= 0,01 \\
K_7 &= \frac{P_{7(-)}}{P_{7(-)} + R_7} = \frac{0,0019}{0,0019 + 0,01} = 0,1607 \\
\hat{x}_{7(+)} &= \hat{x}_{7(-)} + K_7(z_7 - \hat{x}_{7(-)}) = 50,020 + 0,1607(49,938 - 50,020) = 50,007°C \\
P_{7(+)} &= (1 - K_7)P_{7(-)} = (1 - 0,1607)0,0019 = 0,0016 \\
\hat{x}_{8(-)} &= \hat{x}_{7(+)} = 50,007°C \\
P_{8(-)} &= P_{7(+)} + Q = 0,0016 + 0,0001 = 0,0017
\end{aligned}
$$

Osmá iterace

$$
\begin{aligned}
z_8 &= 49,858°C \\
R_8 &= 0,01 \\
K_8 &= \frac{P_{8(-)}}{P_{8(-)} + R_8} = \frac{0,0017}{0,0017 + 0,01} = 0,1458 \\
\hat{x}_{8(+)} &= \hat{x}_{8(-)} + K_8(z_8 - \hat{x}_{8(-)}) = 50,007 + 0,1458(49,858 - 50,007) = 49,985°C \\
P_{8(+)} &= (1 - K_8)P_{8(-)} = (1 - 0,1458)0,0017 = 0,0015 \\
\hat{x}_{9(-)} &= \hat{x}_{8(+)} = 49,985°C \\
P_{9(-)} &= P_{8(+)} + Q = 0,0015 + 0,0001 = 0,0016
\end{aligned}
$$

Devátá iterace

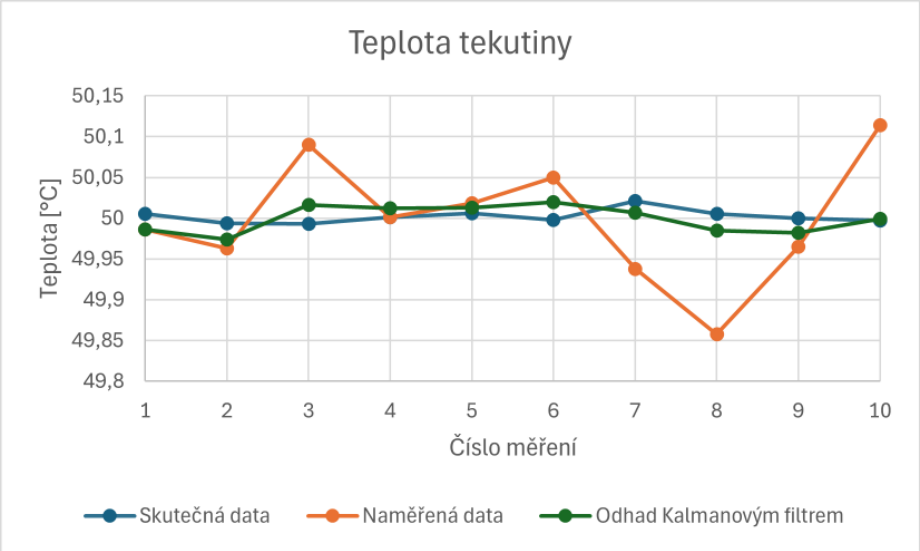
$$
\begin{aligned}
z_9 &= 49,965°C \\
R_9 &= 0,01 \\
K_9 &= \frac{P_{9(-)}}{P_{9(-)} + R_9} = \frac{0,0016}{0,0016 + 0,01} = 0,1348 \\
\hat{x}_{9(+)} &= \hat{x}_{9(-)} + K_9(z_9 - \hat{x}_{(-)}) = 49,985 + 0,1348(49,965 - 49,985) = 49,982°C \\
P_{9(+)} &= (1 - K_9)P_{9(-)} = (1 - 0,1348)0,0016 = 0,0014 \\
\hat{x}_{10(-)} &= \hat{x}_{9(+)} = 49,982°C \\
P_{10(-)} &= P_{9(+)} + Q = 0,0014 + 0,0001 = 0,0015
\end{aligned}
$$

Desátá iterace

$$z_10 = 50,114°C$$
$$R_10 = 0,01$$
$$K_10 = \frac{P_{10(-)}}{P_{10(-)} + R_10} = \frac{0,0015}{0,0015 + 0,01} = 0,1265$$
$$\hat{x}_{10(+)} = \hat{x}_{10(-)} + K_10(z_10 - \hat{x}_{10(-)}) = 49,982 + 0,1265(50,114 - 49,982) = 49,999°C$$
$$P_{10(+)} = (1 - K_10)P_{10(-)} = (1 - 0,1265)0,0015 = 0,0013$$
$$\hat{x}_{11(-)} = \hat{x}_{10(+)} = 49,999°C$$
$$P_{11(-)} = P_{10(+)} + Q = 0,0013 + 0,0001 = 0,0014$$

Shrnutí výsledků

Podívejme se nyní na výsledky tohoto příkladu. V následujícím grafu můžeme vidět porovnány skutečné hodnoty, naměřené hodnoty a odhady získané pomocí Kalmanova filtru.



Obrázek 3.1: Teploty tekutiny.

Z grafu je možné vidět, že odhady Kalmanova filtru postupně konvergují ke skutečným hodnotám. Můžeme tedy říct, že i při přidání náhodného šumu poskytuje Kalmanův filtr dobrý odhad hodnot.

3.7 Využití Kalmanova filtru

Kalmanův filtr je široce využívaný nástroj pro odhadování stavů dynamických systémů z neúplných a šumových měření. Je to klíčový nástroj v případech, kde je nutné sloučení dat z různých senzorů, což umožňuje velice přesné sledování polohy a rychlosti. To je zásadní pro aplikace v letectví, autonomních vozidlech a mobilních zařízeních. Konkrétně můžeme

mluvit například o SLAM (Simultaneous Localization and Mapping). Tato metoda umožňuje vozidlu lokalizovat se v neznámém prostředí a současně toto prostředí zmapovat, což může být výhodné např. není-li k dispozici mapa [5]. Metoda využívá algoritmy jako právě Kalmanův filtr nebo rozšířený Kalmanův filtr.

Dalším možným využitím Kalmanova filtru je v oblasti energetiky pro odhady procesních poruch v budovách. Procesní poruchou je myšlen souhrn tepelných zisků a ztrát způsobených vnitřními zdroji tepla, např. lidmi, světly, zařízením, a prouděním vzduchu. Tyto zdroje tepla je obtížné měřit a proto je pro odhad použit Kalmanův filtr [12].

Kalmanův filtr nachází své místo i v ekonomice a zpracování časových řad [10], nebo také ve financích. Tam se využívá pro odhad parametrů modelu, ať se jedná o modely časové struktury úrokových sazeb či modely časové struktury cen komodit. Zároveň může být také vhodnou metodou v případě velkého množství informací, vzhledem k jeho rychlosti [14].

Kapitola 4

Praktická část

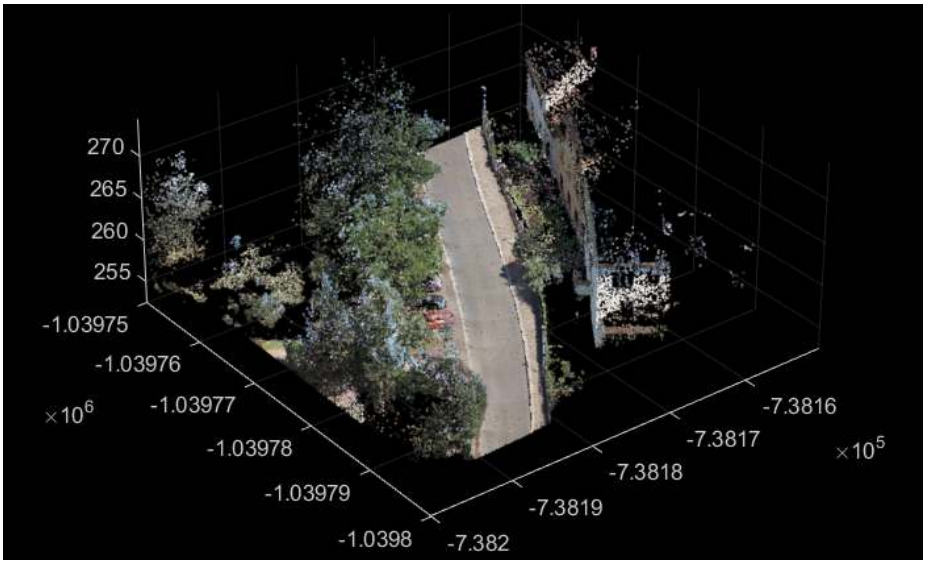
Hlavním tématem práce bylo nasazení Kalmanova filtru na reálná mračna bodů a detekce obrubníků. Tento problém je v praxi součást algoritmů pro kontrolu polohy vozidla. Obrubníky jsou totiž přítomné ve většině městské zástavby, mají obvykle standardní výšky a rozměry, jsou tedy dobře rozpoznatelné a pomáhají s lokalizací vozidla i s detekcí možného sjetí z trati.

Pracovat budeme s reálnými mračny bodů naskenovanými pomocí LiDARu. Úkolem této části je v prvé řadě nalezení pozemních bodů (v anglicky psané literatuře nazývané ground points). Poté v těchto detekovaných bodech určit, kde se nacházejí obrubníky pro následné použití Kalmanova filtru na tato data. Dále budeme mít také uměle vytvořená data skupiny domů, na které bude nejdříve použit clustering, abychom od sebe oddělili jednotlivé domy a následně bude opět použit Kalmanův filtr pro detekci hrany jednoho z domů. Hrany domů jsou v datech statické a výrazné, proto jsou často používány při algoritmech sledování polohy vozidla [8]. Následně to stejné provedeme i na reálných datech. Pro každou situaci bude vždy použit náš naprogramovaný algoritmus Kalmanova filtru a funkce Kalmanova filtru implementovaná v Matlabu a výsledky budou porovnány.

4.1 Detekce obrubníků v reálných datech

První data, která budeme na aplikace Kalmanova filtru využívat, jsou zobrazena na obrázku 4.1. Tato data byla získána od společnosti Cyclomedia [1] a jedná se o úsek ulice skenovaný v Praze.

[1] https://www.cyclomedia.com/cs



Obrázek 4.1: Obarvený point cloud (zdroj cyclomedia.com).

Pro další práci s mračnem potřebujeme z modelu získat pozemní body a odstranit body pro nás nepotřebné, jako jsou okolní domy, stromy a podobně. Pro odstranění těchto bodů a nalezení bodů pozemních využijeme algoritmus RANSAC (Random Sample Consensus), který je popsán v následující části.

4.1.1 RANSAC

Jedná se o algoritmus pro odhad parametrů modelu pomocí náhodného výběru pozorovaných dat. Tento algoritmus poprvé publikovali Martin A. Fischler a Robert C. Bolles v roce 1981 [7]. RANSAC je velice vhodný zvláště v případech, kdy víme, že zpracovávaná data obsahují outliery. V těchto případech je většina jiných metod téměř nepoužitelná, jelikož jsou ovlivněny i již zmíněnými outliery.

Jako vstupní údaje pro tento algoritmus potřebujeme určit nejmenší počet parametrů modelu. V případě mračna bodů se jedná o počet bodů n. Toto číslo závisí na tom, co chceme modelovat. Budeme-li požadovat přímku, pak $n = 2$, v případě roviny $n = 3$, pro afinní transformaci je potřeba $n = 6$ atd. Dále je třeba určit maximální počet iterací a tzv. treshold, což je hodnota udávající, zda je daný bod správný. Často se jedná o euklidovskou vzdálenost, ale může se jednat i o jiné metriky. Jako poslední vstupní hodnota je minimální počet bodů, který má model obsahovat. Postup samotného algoritmu je poté následující.

1. Vybereme náhodně n bodů a vytvoříme z nich model.
2. Vypočteme vzdálenosti pro všechny body mračna od modelu.
3. Bude-li v předem dané vzdálenosti od modelu (treshold) námi určený počet bodů, znamená to, že algoritmus našel požadované body a končí.
4. V opačném případě vybere náhodně jiných n bodů a postup se opakuje, dokud podmínka v bodě 3. není splněna, nebo není překročen maximální počet iterací.

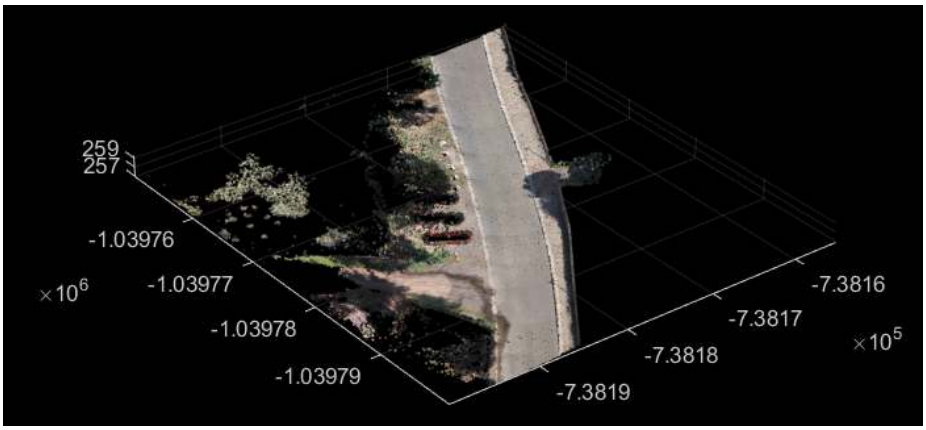
Nastavení parametrů maximálního počtu iterací a treshold může být obtížné a je třeba hodnoty vyladit. Například pokud bychom treshold zvolili moc malý, pak algoritmus nemusí

vůbec nalézt model. Naopak v případě příliš velkého tresholdu může být špatný model určen jako správný. Pro použití tohoto algoritmu je také třeba mít nějaké znalosti o datech, se kterými pracujeme, což může být nevýhoda v případě nám neznámých dat. Naopak jeho výhodou je velice snadná implementace a srozumitelnost. Schematicky pak tento algoritmus bude vypadat následujícím způsobem.

```
while pocet iteraci < maximalni pocet iteraci
    vytvor model z n nahodne vybranych bodu
    for ostatni body
        spocitej vzdalenost d bodu od modelu
        if d < thr
            pridej bod do mnoziny spravnych bodu
        end
    end
    if velikost mnoziny spravnych bodu > T
        urceny model je spravny
    end
end
```

Předpokládáme, že pozemní body tvoří rovinu. Výsledek aplikace RANSAC algoritmu je na obr. 4.2.

V tomto případě, jelikož se jedná o rovinu, $n = 3$. Treshold byl nastaven na $thr = 0,3$, minimální počet bodů, který má rovina obsahovat, T byla třetina s celkového počtu bodů point cloudu a algoritmus potřeboval 27 iterací.



Obrázek 4.2: Použití RANSAC algoritmu.

Po proběhnutí algoritmu RANSAC získáváme mračno bodů představující pozemní body. Na tomto mračnu chceme najít obrubníky, tedy určit kde končí vozovka a začíná chodník. Toho dosáhneme využitím několika kroků. První z nich je určení normál v mračnu bodů. Obrubníky jsou totiž místa, kde dochází k velké změně normál. Na základě hustoty okolí možných bodů jsou poté některé odstraněny a poslední část je doplnění chybějících částí.

4.1.2 Určení normál k bodům mračna

V euklidovském prostoru je normála přímka kolmá k tečné rovině plochy v daném bodě. Zde máme však pouze mračna bodů. Řešení je proložení okolích bodů rovinou a

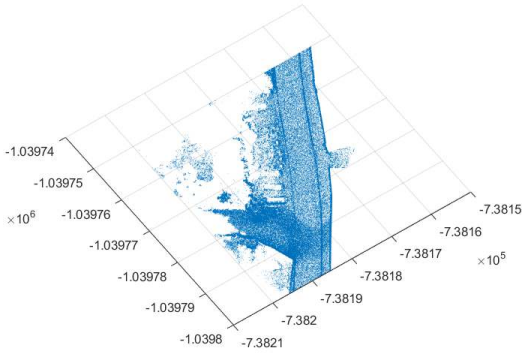
určení normály k rovině procházející daným bodem. V našem případě byla použita Matlab funkce `pcnormals`. Tato funkce použije šest bodů sousedících s bodem, ke kterému hledáme normálu, a z nich lokálně aproximuje rovinu. Normála k této rovině pak bude normálou daného bodu point cloudu. Normály ke všem bodům mračna můžeme vidět na obrázku 4.3. Jedná se o pohled shora, proto světlejší místa značí, kde jsou normály kolmé na povrch vozovky. Naopak tmavá místa říkají, že normály nejsou k vozovce kolmé, ale nějak sklopené a v obrázku se tak nejeví pouze jako body, ale jako malé úsečky a jsou tedy výraznější.

Dále vybereme libovolný bod (na obrázku 4.4 vpravo zvýrazněn jako tmavě modrý bod), který se nachází na vozovce, a jeho normálu si zvolíme jako referenční. Nyní pro každý bod vypočítáme hodnotu úhlu, který svírá normála daného bodu a referenční normála, přesněji odchylku jejich normálových vektorů.

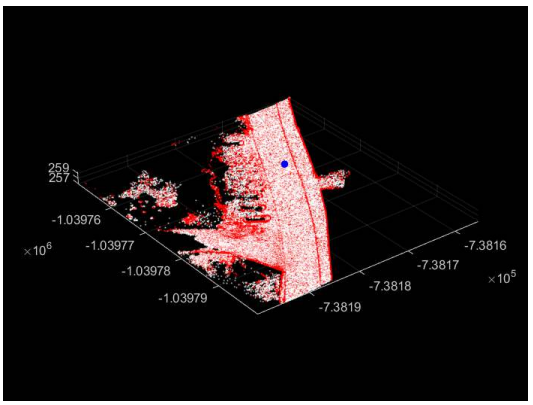
K výpočtu odchylky dvou nenulových vektorů $\mathbf{u}$, $\mathbf{v}$ v euklidovské rovině, či prostoru můžeme použít známý vzorec:

$$\varphi = \cos^{-1} \frac{\mathbf{uv}}{|\mathbf{u}| \cdot |\mathbf{v}|}. \tag{4.1}$$

Pro body na vozovce bude platit, že úhly mezi jejich a referenční normálou budou téměř nulové s možnými malými odchylkami. Naopak body na obrubníku budou s referenční normálou svírat úhly podstatně větší. Ponecháme tedy pouze body, jejichž úhel je větší než předem daná velikost úhlu, kterou si určíme. V případě testovaných dat byla tato hodnota nastavena na 20 stupňů.



Obrázek 4.3: Vstupní mračno bodů s vypočítanými normálami.



Obrázek 4.4: Červeně zobrazené body s normálami splňujícími podmínku $\varphi < \text{thr}$.

Červené body na obrázku 4.4 značí body, které jsme si ponechali a dále s nimi budeme pracovat, naopak bílé jsou pak body k odstranění. Tímto postupem odstraníme velkou část bodů, které pro nás nejsou důležité, avšak na vozovce se mohou vyskytovat různé nerovnosti a je proto třeba využít další algoritmy, jak výsledky zlepšit.

4.1.3 Výběr obrubníků

Další algoritmus, který využijeme pro zlepšení dosažených výsledků, posuzuje počet bodů v okolí daného bodu. Abychom si mohli zavést pojem okolí bodu musíme si uvědomit, že euklidovský prostor je metrický prostor. Metriku a metrický prostor definujeme následovně.

Definice 4.1. Nechť M je neprázdná množina. Tzv. metrika na M se definuje jako zobrazení $\rho : M \times M \to [0, \infty)$, které splňuje pro každé $x, y, z \in M$ následující tři axiomy:

1. $\rho(x, y) = 0$, právě tehdy když $x = y$;
2. $\rho(x, y) = \rho(y, x)$;
3. $\rho(x, y) + \rho(y, z) \geq \rho(x, z)$.

Dvojici (M, ρ) nazýváme metrickým prostorem. Číslo $\rho(x, y)$ se někdy nazývá vzdálenost bodů x, y.

Pro dané $n \in \mathbb{N}$ nechť $M := \mathbb{R}^n$ a nechť $p \in [1, \infty)$ je pevné reálné číslo. Funkce

$$\rho_p(x, y) := \left(\sum_{k=1}^{n} |x_k - y_k|^p \right)^{\frac{1}{p}}, x := (x_1, ..., x_n), y := (y_1, ..., y_n) \in \mathbb{R}^n, \tag{4.2}$$

je potom metrikou na množině M a dvojice (M, ρ_p) je metrický prostor. V případě $p = 2$ dostáváme tzv. euklidovskou metriku ρ_2 , tj.

$$\rho_2(x, y) := \sqrt{\sum_{k=1}^{n} |x_k - y_k|^2}, x := (x_1, ..., x_n), y := (y_1, ..., y_n) \in \mathbb{R}^n, \tag{4.3}$$

metrický prostor (M, ρ_2) nazýváme euklidovský prostor a označujeme jej $\mathbb{E}^n$.

Potom pojmem okolí bodu rozumíme v metrickém prostoru otevřenou kouli se středem v daném bodě.

Definice 4.2. Nechť $(M,\ \rho)$ je metrický prostor a nechť $x_0 \in M$ a $r \in \mathbb{R}^+$ jsou dané. Otevřenou koulí se středem v bodě x_0 a poloměrem r rozumíme množinu

$$B(x_0, r) := \{x \in M, \rho(x, x_0) < r\}. \tag{4.4}$$

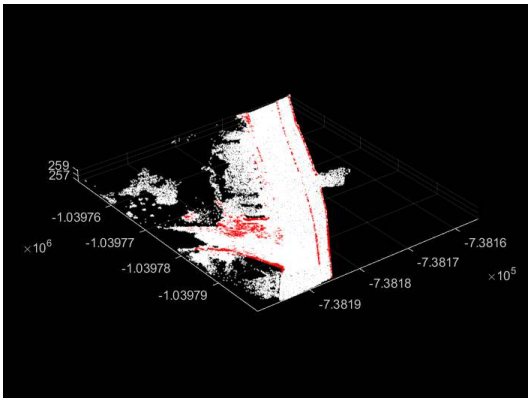
V našem případě M představuje $\mathbb{R}^3$ a ρ je euklidovská metrika

$$\rho_e(x, y) = \sqrt{\sum_{i=1}^{3} (x_i - y_i)^2}.$$

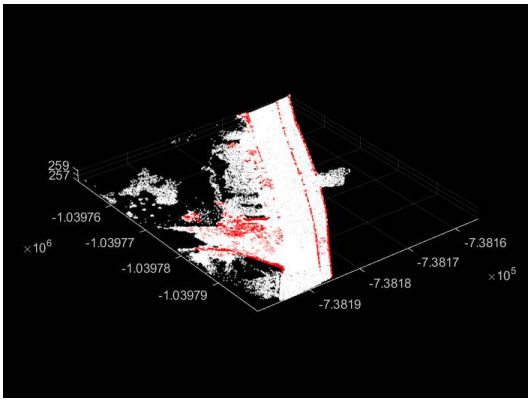
Projdeme všechny body, které jsme předchozími kroky určili jako správné a zjistíme kolik bodů se nachází v jejich blízkém okolí. K výpočtu vzdálenosti používáme euklidovskou metriku. Velikost okolí určujeme my a stejně tak i kolik bodů požadujeme, aby toto okolí obsahovalo. V místech, kde se skutečně jedná o obrubník, se bude vyskytovat velké množství bodů. Naopak na vozovce, kde jsou pouze různé nerovnosti, bude bodů podstatně méně. Body s malým počtem okolních bodů jsou odstraněny. Hodnoty byly nastaveny empiricky, pro zde uvedený příklad to bylo velikost okolí 0,2 a minimální počet bodů v okolí 13.

4.1.4 Doplnění bodů

Předchozí algoritmus je nastaven tak, abychom odstranili co největší počet chybně označených bodů. Tím ovšem ztratíme i některé body, které bychom chtěli ponechat, avšak z nějakých důvodů kolem sebe neměly dostatečný počet jiných bodů. Proto se nyní pokusíme některé získat zpět. Pro každý bod, který předchozí algoritmy označily jako bod obrubníku, najdeme nejbližší bod z tohoto mračna. Nazpět potom doplníme všechny body z původního mračna, které mají od daného bodu menší vzdálenost, než je od nejbližšího nalezeného bodu.



Obrázek 4.5: Výběr obrubníků.



Obrázek 4.6: Doplnění.

4.1.5 Separace obrubníku

Abychom mohli použít Kalmanův filtr na určitý obrubník, je třeba z dat získat pouze body odpovídající tomuto obrubníku. V datech na obrázku 4.1 je na levé straně vozovky obrubník částečně sklopený kvůli parkovišti, budeme se proto nyní na začátek zabývat pravým obrubníkem a levý bude poté zpracován v příloze D.

V tomto případě není těžké pravý obrubník oddělit, jelikož se jedná téměř o přímku. Využijeme parametrického vyjádření přímky v prostoru. V mračnu obrubníku zjistíme souřadnice dvou bodů $P1$ a $P2$, o kterých jistě víme, že na obrubníku skutečně leží. Z těchto bodů vypočítáme směrový vektor s a vytvoříme parametrické vyjádření přímky procházející těmito body

$$\begin{aligned} x &= P1_1 + t \cdot s_1, \\ y &= P1_2 + t \cdot s_2, \\ z &= P1_3 + t \cdot s_3, \end{aligned} \tag{4.5}$$

kde t je parametr. Nyní abychom zjistili vzdálenost bodu BOD od této přímky musíme prvně najít bod $X[x, y, z]$, který bude ležet na přímce a platí pro něj $(BOD - X)s = 0$. Tuto podmínky rozepíšeme pomocí jednotlivých souřadnic a vyjádříme parametr t.

$$\begin{aligned} (BOD_1 - (P1_1 + ts_1))s_1 + (BOD_2 - (P1_2 + ts_2))s_2 + (BOD_3 - (P1_3 + ts_3))s_3 &= 0 \\ (BOD_1 - P1_1)s_1 - s_1^2 t + (BOD_2 - P1_2)s_2 - s_2^2 t + (BOD_3 - P1_3)s_3 - s_3^2 t &= 0 \\ (BOD_1 - P1_1)s_1 + (BOD_2 - P1_2)s_2 + (BOD_3 - P1_3)s_3 - (s_1^2 + s_2^2 + s_3^2)t &= 0 \\ \frac{(BOD_1 - P1_1)s_1 + (BOD_2 - P1_2)s_2 + (BOD_3 - P1_3)s_3}{(s_1^2 + s_2^2 + s_3^2)} &= t \end{aligned}$$

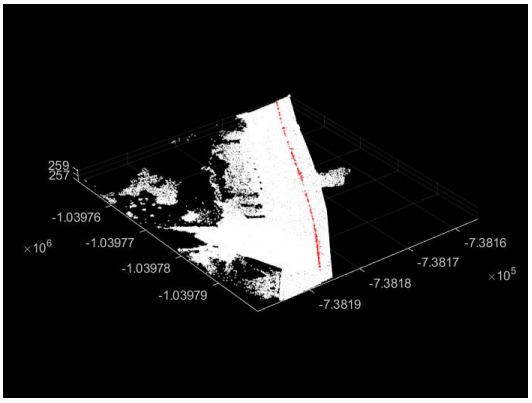
Za parametr t dosadíme do soustavy přímky (4.5) a zjistíme souřadnice bodu X. Nakonec vypočítáme vzdálenost bodů X a BOD jako klasickou euklidovskou vzdálenost dvou bodů

$$\begin{aligned} XBOD &= BOD - X \\ d &= \sqrt{XBOD(1)^2 + XBOD(2)^2 + XBOD(3)^2}. \end{aligned}$$

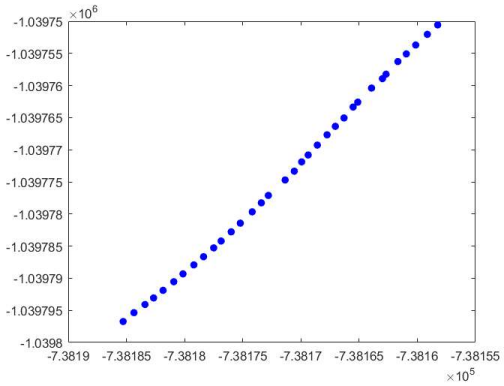
Pokud bude vzdálenost d menší než námi daný treshold, je bod označen jako bod obrubníku. V případě, že je vzdálenost větší než d, pak se nejedná o bod obrubníku a dále s ním již nepracujeme. Následující kód ukazuje implementaci tohoto výpočtu v Matlabu.

```
for i = 1:ptCloud.Count
    BOD = ptCloud.Location(i,:);
    t = (s(1)*(BOD(1)-P1(1))+s(2)*(BOD(2)-P1(2))+s(3)*(BOD(3)-P1(3)))
        /(s(1)^2+s(2)^2+s(3)^2);
    X = [P1(1) + s(1)*t,P1(2) + s(2)*t,P1(3) + s(3)*t];
    XBOD = BOD - X;
    XB = sqrt(XBOD(1)^2 + XBOD(2)^2 + XBOD(3)^2);
    if XB < thr
        o = [o,i];
    end
end
```

Bodů, které představují obrubník, nyní máme hodně a abychom na ně mohli použít Kalmanův filtr, je potřeba data navzorkovat. Pro toto navzorkování použijeme ekvidistantní délku kroku. Nalezneme první bod obrubníku a vezmeme všechny body v určitém poloměru od něj a vypočítáme jejich těžiště. To bude nyní představovat první bod v navzorkování. Tento postup zopakujeme postupně na všechny zbylé body. Na takto připravená data již můžeme použít Kalmanův filtr.



Obrázek 4.7: Detekovaný obrubník v mračnu bodů.



Obrázek 4.8: Navzorkovaný obrubník.

4.1.6 Použití Kalmanova filtru

V implementaci algoritmu, na rozdíl od obecného odvození v části 3.4, předpokládáme, že matice F, H, R a Q jsou konstantní. Upravený algoritmus a naše implementace pak má následující podobu.

```
% inicializace F, H, R, Q, P, M, N
% a prvni predikce pozice Xn
Xn = F*X; % nulta iterace
N = [N;Xn'];
Pn = F*P*F' + Q; % predikce
for k = 1:length(teziste)
    % korekce
    K = Pn*H'*inv(H*Pn*H' + R);
    Xpos = (teziste(k,:))';
    X = Xn + K*(Xpos - H*Xn);
    P = (eye(4,4) - K*H)*Pn;
    M = [M;X'];
    % predikce
    Xn = F*X;
    N = [N;Xn'];
    Pn = F*P*F' + Q;
end
```

Apriorní odhad stavu:

$\hat{x}_{k(-)} = F\hat{x}_{k-1(+)}$

Apriorní odhad chybové kovarianční matice:

$P_{k(-)} = FP_{k-1(+)}F^T + Q$

Výpočet Kalmanova zisku:

$K_k = P_{k(-)}H^T(HP_{k(-)}H^T + R)^{-1}$

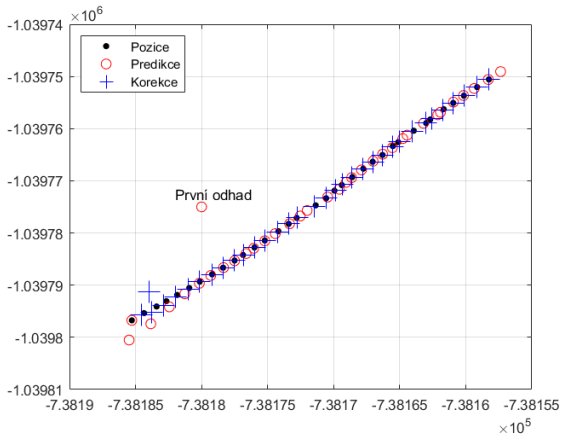
Aposteriorní odhad stavu:

$\hat{x}_{k(+)} = \hat{x}_{k(-)} + K_k(z_k - H\hat{x}_{k(-)})$

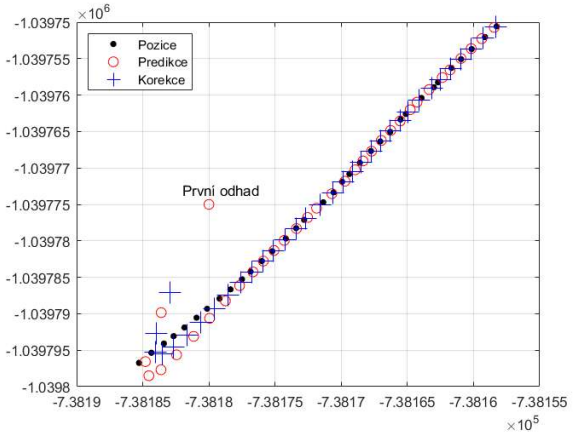
Aposteriorní odhad chybové kovarianční matice:

$P_{k(+)} = (I - K_kH)P_{k(-)}$

Jako první použijeme na připravená data námi navržený algoritmus a vzápětí pro porovnání použijeme také Kalmanův filtr implementovaný v Matlabu.

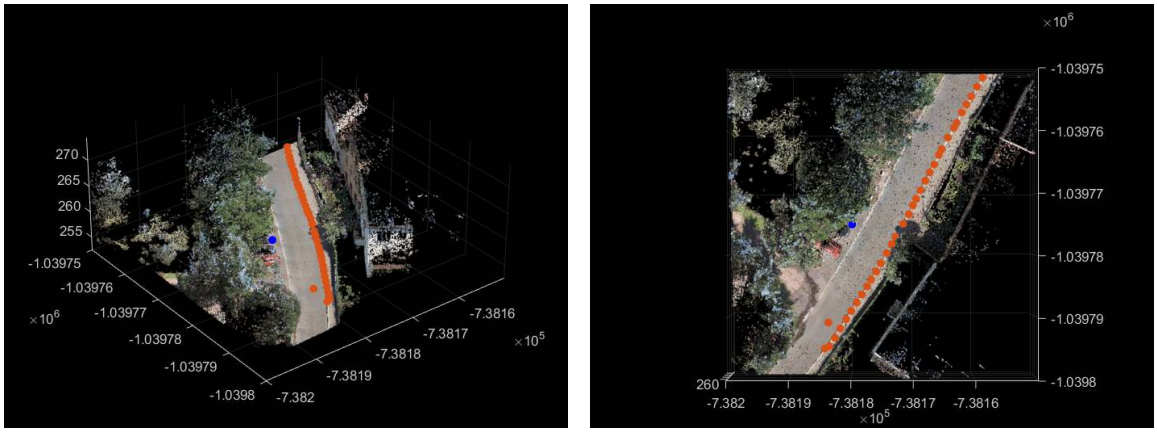


Obrázek 4.9: Navržený Kalmanův filtr.



Obrázek 4.10: Kalmanův filtr implementovaný v Matlabu.

Údaje, získané pomocí námi navrženého algoritmu nyní "vrátíme"do původního mračna bodů, abychom lépe viděli, zda algoritmus skutečně konverguje k pravému obrubníku. Výsledek můžeme vidět na obrázku 4.11, kde modrý bod představuje první odhad pozice.



Obrázek 4.11: Zasazení do původního mračna.

4.1.7 Porovnání výsledků

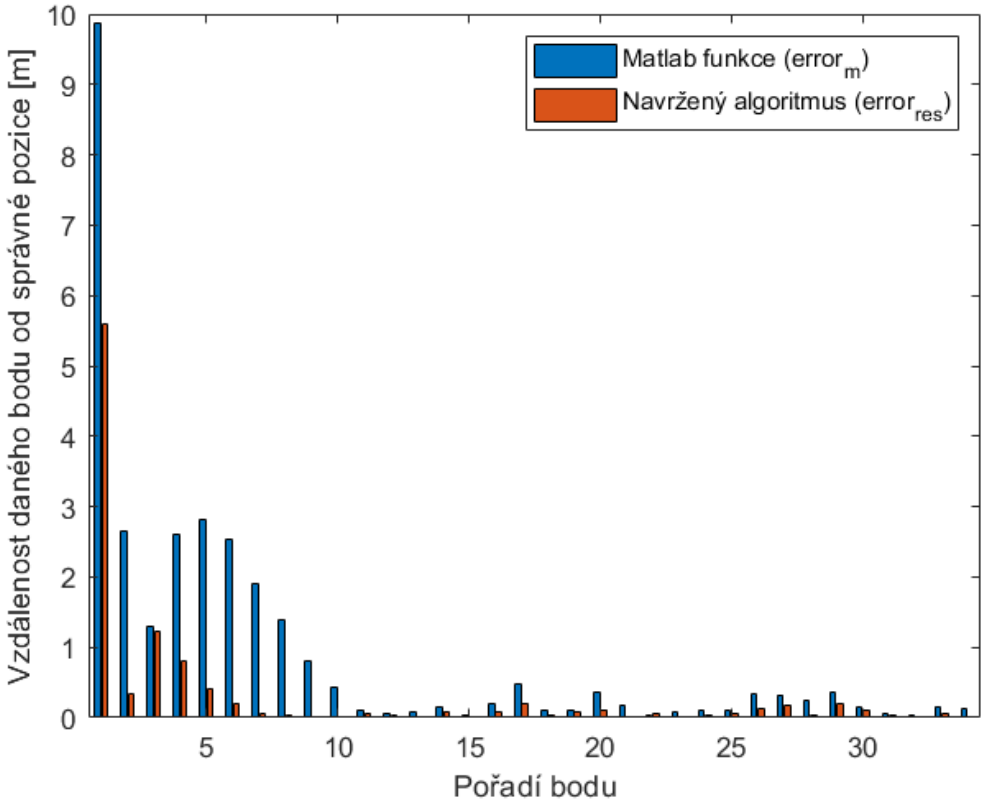
Podívejme se nyní na hodnoty, které nám daly oba filtry a porovnejme je. Pro každý bod vezměme skutečné souřadnice bodu (x_{real}, y_{real}), souřadnice dané Matlab funkcí (x_m, y_m) pro daný bod a souřadnice dané naším navrženým Kalmanovým filtrem (x_{res}, y_{res}) pro daný bod. Nyní pomocí těchto údajů vypočteme jednotlivé vzdálenosti

$$error_m = \sqrt{(x_m - x_{real})^2 + (y_m - y_{real})^2}$$
$$error_{res} = \sqrt{(x_{res} - x_{real})^2 + (y_{res} - y_{real})^2}.$$

Tyto hodnoty vypočítáme pro každý bod a následně je zaneseme do grafu, jak je možné vidět na obrázku 4.12. Samotné vzdálenosti od správných pozic jednotlivých bodů pro oba algoritmy pak můžeme vidět v tabulce 4.1.

Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$	Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$
1	987,494	559,323	18	11,325	4,604
2	265,105	33,278	19	11,568	7,926
3	129,377	121,791	20	35,118	11,119
4	260,019	79,989	21	16,713	2,054
5	282,153	41,938	22	2,408	5,012
6	254,201	20,679	23	7,395	2,160
7	191,151	5,579	24	10,165	2,708
8	138,173	4,283	25	9,651	5,419
9	80,496	0,720	26	32,940	13,618
10	43,794	0,981	27	32,581	17,788
11	9,495	6,315	28	23,505	3,724
12	5,329	2,928	29	35,830	18,884
13	8,411	1,454	30	14,654	11,541
14	14,583	8,914	31	4,907	3,041
15	2,898	1,353	32	4,037	0,938
16	20,242	7,971	33	16,314	6,199
17	47,148	21,021	34	11,685	0,947

Tabulka 4.1: Vzdálenosti od správné pozice jednotlivých bodů.



Obrázek 4.12: Graf chyb algoritmů.

Z grafu je dobře vidět, že oba algoritmy postupně konvergují ke správnému řešení, jak by měly. Chyba se tedy v obou případech postupně zmenšuje, avšak z tabulky i z grafu je patrné, že chyba, které se Matlab funkce dopouští, je po první iteraci téměř dvojnásobná, než námi navržený algoritmus. Náš algoritmus se na malé odchylky od správné pozice dostává při sedmé iteraci, kdežto Matlab funkce zhruba při jedenácté iteraci. Od této pozice jsou pak oba algoritmy, co se chyby týče velice podobně.

4.2 Nasazení Kalmanova filtru pro clusterovaná data

V této části aplikujeme Kalmanův filtr na jiný druh dat. V praxi při použití komplexních algoritmů lokalizace je nutné zachytávat body na výrazných objektech v okolí. K tomu se využívají také domy a jejich hrany. Proto jsme vytvořili scénu několika domů a pomocí clusteringu a detekčních algoritmů našli hrany. Na obrázku 4.13 můžeme vidět, že se jedná o point cloud několika domů a povrchu země vytvořené za pomoci programu Rhino.

V prvé řadě je třeba odstranit pozemní body, což provedeme pomocí algoritmu RANSAC, který byl popsán v sekci 4.1.1. Tentokrát však, na rozdíl do předchozího případu, všechny takto nalezené pozemní body odstraníme. Jelikož se jedná o uměle vygenerovaná data, výsledky po použití tohoto algoritmu jsou téměř dokonalé, i přesto, že byl do point cloudu uměle přidán šum.

4.2.1 Clustering

Nyní potřebujeme ve zbylém mračnu určit tzv. clustery neboli shluky bodů, představující jednotlivé domy. Pro identifikaci těchto clusterů je možné využít hned několik metod. V našem případě byl použit algoritmus k-means.

K-means

Cílem algoritmu je rozdělit point cloud na předem definovaný počet clusterů (shluků) k. Každý cluster je reprezentován svým středem, ke kterému jsou přiřazeny nejbližší body podle zvolené metriky, obvykle euklidovské vzdálenosti. Uveďme nyní popis fungování tohoto algoritmu.

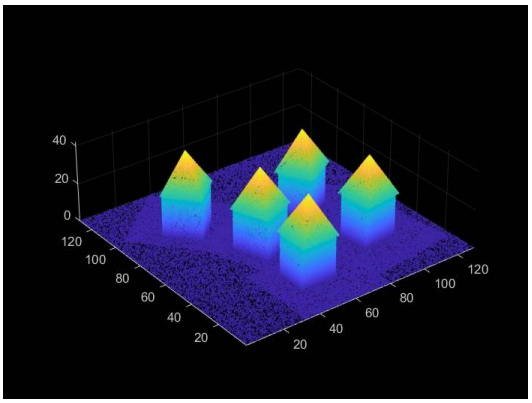
1. Vybereme k bodů, které budou představovat počáteční středy clusterů.
2. Pro všechny body v mračnu vypočteme vzdálenosti od počátečních středů clusterů.
3. Každý bod přiřadíme svému středu na základě toho, ke kterému z k bodů má nejkratší vzdálenost. Tímto získáme k clusterů.
4. Vypočítáme nové středy clusterů jako jejich těžiště.
5. Pro všechny body mračna opět vypočítáme vzdálenosti od těchto nových středů a znovu každý bod přiřadíme nejbližšímu středu.
6. Tento postup výpočtu nových středů a clusterů opakujeme dokud nově vypočtené středy nebudou měnit svoji polohu, případně velice nepatrně, nebo nepřekročíme maximální počet iterací.

Velice důležité pro tento algoritmus je první výběr středů. Na základě prvotního náhodného určení startovacích bodů se výsledky, ke kterým algoritmus dojde, mohou významně

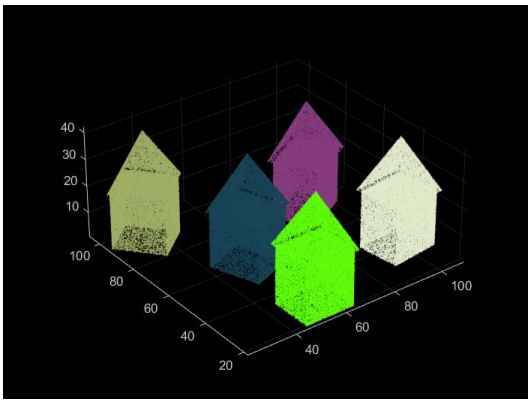
lišit. Pro zlepšení vybírání těchto prvních bodů byl použit algoritmus k-means++. Tento algoritmus náhodně vybere jeden bod z celého mračna a pro všechny ostatní body se opět vypočítá vzdálenost k tomuto bodu. Jako druhý střed se poté vybere ten, který je od prvního nejvzdálenější. Jako třetí střed pak hledáme bod, jehož součet vzdáleností od obou středů je ze všech největší. Tímto způsobem analogicky najdeme všech K potřebných počátečních středů.

Tento vylepšený algoritmus ovšem opět nezaručí vždy stejné a správné nalezení clusterů, které bychom chtěli. Přidáme proto ještě podmínku na velikost získaných clusterů, jelikož máme předem znalost, jak vypadá naše mračno bodů a tedy víme, že všechny clustery by měly mít přibližně stejnou velikost. Pokud tato podmínka tedy nebude splněna, proběhne algoritmus znovu.

Poukažme opět na fakt, že se v tomto případě jedná o uměle vytvořená data, proto i clustering, stejně jako v případě RANSACu, bude fungovat bezchybně. Domy jsou navíc umístěny dostatečně od sebe, aby se nespojovalo více domů v jeden cluster. Získáme tak velmi pěkné výsledky, které můžeme vidět na obrázku 4.14.



Obrázek 4.13: Vygenerované mračno bodů objektů se šumem.



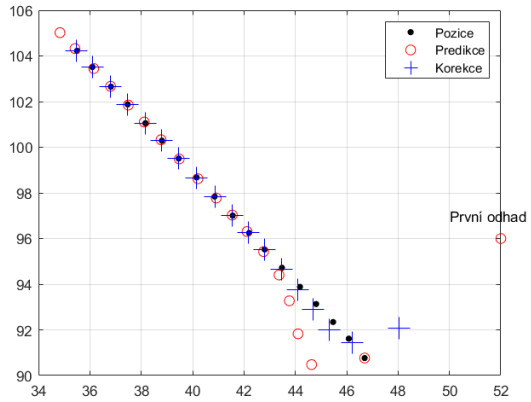
Obrázek 4.14: Clustering.

4.2.2 Detekce hrany objektu

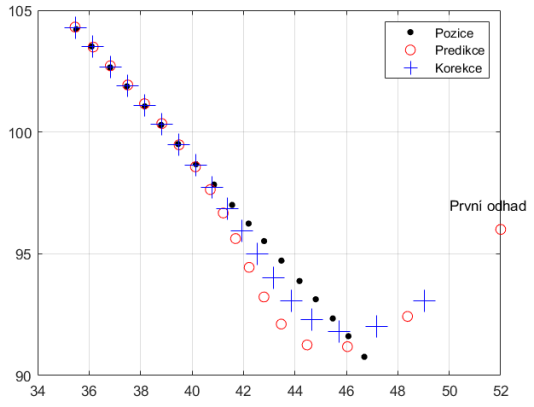
Po nalezení jednotlivých objektů nyní, podobně jako v případě hledání obrubníků, oddělíme část, na kterou chceme použít Kalmanův filtr. V tomto případě se bude jednat o zeď jednoho z domů. Oddělení bude velice jednoduché, jelikož zeď domu bude tvořit úsečku. Proto postup bude stejný jako případě dat z předchozí části. Tu následně promítneme do 2D a navzorkujeme opět stejným způsobem jako v části 4.1.5.

4.2.3 Použití Kalmanova filtru

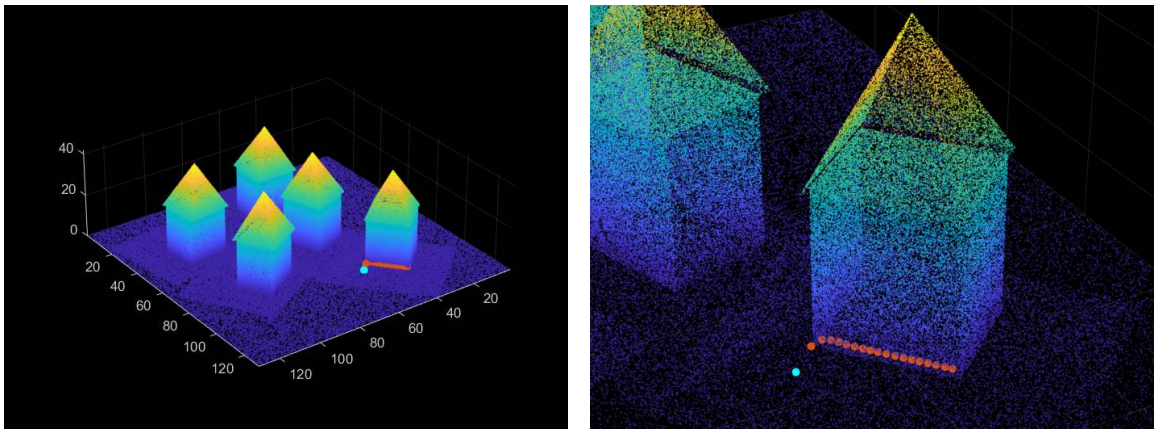
Na tato připravená data nyní použijeme Kalmanův filtr a to stejně jako v případě předcházející části prvně námi navržený algoritmus a následně Matlab funkci pro Kalmanův filtr. Nakonec údaje získané pomocí našeho algoritmu opět vrátíme do původního mračna. Jako světle tyrkysový bod je na obrázcích 4.17 opět vyznačen první odhad pozice.



Obrázek 4.15: Navržený Kalmanův filtr.



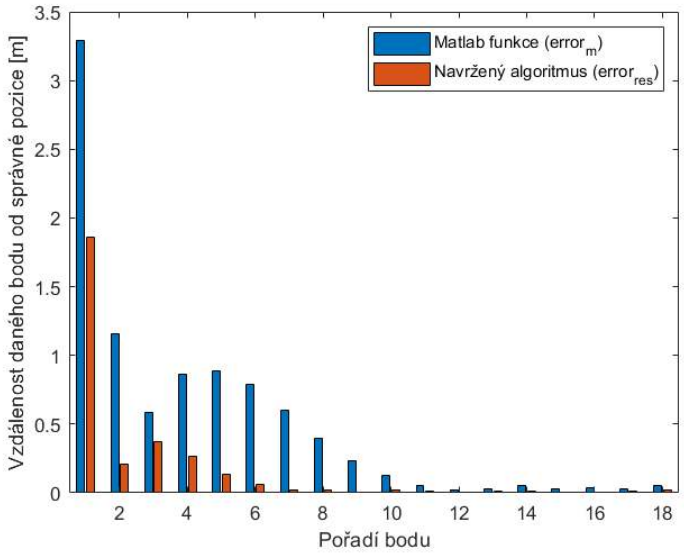
Obrázek 4.16: Kalmanův filtr implementovaný v Matlabu.



Obrázek 4.17: Zasazení do původního mračna.

4.2.4 Porovnání výsledků

Chyby obou algoritmů vypočítáme stejně jako v sekci 4.1.7 a opět zaneseme do grafů. Konkrétní hodnoty jsou poté uvedeny v tabulce 4.2.



Obrázek 4.18: Graf chyb algoritmů.

Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$
1	328,849	186,262
2	115,996	20,953
3	58,623	36,856
4	86,187	26,373
5	88,737	13,116
6	78,583	5,984
7	59,841	1,861
8	39,307	1,905
9	23,641	0,489
10	12,422	1,639
11	5,405	1,346
12	2,345	0,488
13	3,042	0,898
14	5,544	1,251
15	3,196	0,679
16	3,297	0,377
17	3,232	1,510
18	4,962	2,289

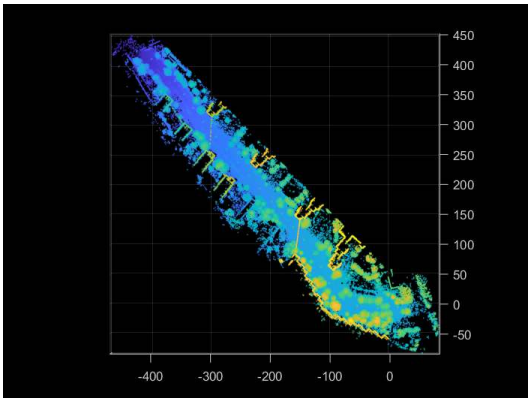
Tabulka 4.2: Vzdálenosti od správné pozice jednotlivých bodů.

Stejně jako v předchozím případě oba algoritmy docházejí ke správnému výsledku. V tomto případě není rozdíl v první iteraci tak znatelný, avšak stále je poměrně podstatný. Náš algoritmus se dostává na malé chyby při šesté iteraci, kdežto Matlab až při jedenácté iteraci. Od dvanácté iterace jsou chyby obou algoritmů velice podobné.

4.3 Nalezení hrany domu na reálných datech

Nyní se budeme věnovat stejnému problému jako v předchozí kapitole, ovšem tentokrát budeme pracovat s reálnými daty ulice Kosmonautů v Brně. Původní data, se kterými budeme pracovat můžeme vidět na obrázku 4.19, avšak pro další práci s nimi si vyřízneme pouze určitý úsek, abychom nepracovali s příliš velkými daty. Jedná se o data pořízená za pomoci ručního LiDARu z z Czechglobe, Ústavu výzkumu globální změny AV ČR.[2]

[2]https://www.czechglobe.cz/cs/kontakty/1133/



Obrázek 4.19: Mračno bodů - reálná data, ulice Kosmonautů, Brno.

Obrázek 4.20: Data zobrazená do ortho-mapy.

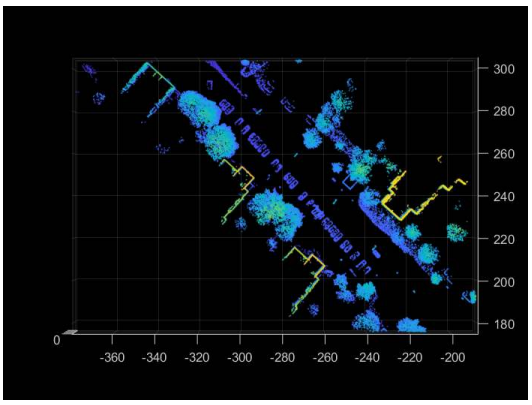
4.3.1 Data a jejich zpracování

Jak již bylo uvedeno výše, data nejdříve ořízneme a následně použijeme algoritmus RANSAC, abychom se opět jako v předchozí sekci zbavili většiny pozemních bodů.

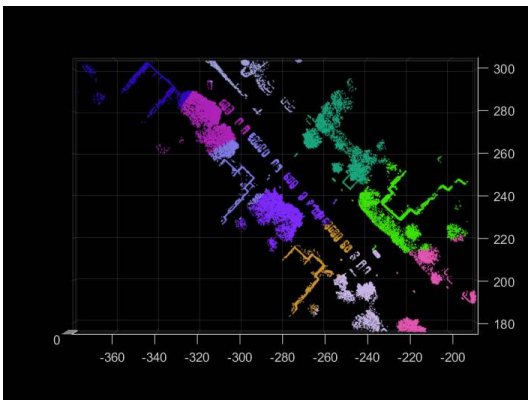
Následně odstraníme šum, kterého je v reálných datech, na rozdíl od předcházejících umělých, podstatně větší množství. Pro tento krok budeme porovnávat počet sousedních bodů v určitém okolí bodu, podobně jako jsme to dělali v případě obrubníku v části 4.1.3. V případě, že bod bude mít ve svém okolí pouze menší počet bodů, budeme ho považovat za šum a dále ho nebudeme používat.

4.3.2 Clustering

Nyní se pustíme do clusteringu stejným způsobem jako je uvedeno v sekci 4.2.1 pouze s malými rozdíly. Jako první změna je, že na rozdíl od předchozího použití nebudeme používat k-means++. Tato změna je z důvodu fungování k-means++, jelikož z jeho principu jsou jako první středy určeny body na okrajích mračna a představují zbytky šumu, kterých se nám nepodařilo zbavit v předchozím kroku. V tomto případě budeme tedy první body volit zcela náhodně.



Obrázek 4.21: RANSAC a odstranění šumu.



Obrázek 4.22: Clustering.

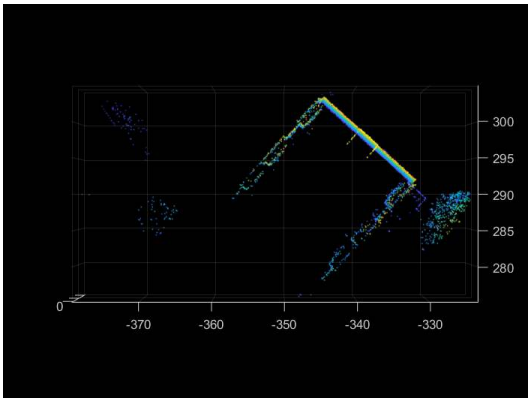
Na oříznutých datech vidíme čtyři domy, avšak hledaný počet clusterů musíme nastavit na větší. Důvodem jsou hlavně stromy, které se nám zde vyskytují. Pokud bychom počet

hledaných clusterů nastavili přesně na čtyři, žádný z nich by neobsahoval pouze dům. V tomto případě byl počet clusterů nastaven na deset. Na obrázku 4.22 můžeme vidět, že tmavě modrý a oranžový byly poměrně dobře odděleny od ostatních bodů, avšak clustery druhých dvou domů obsahují i části stromů, jelikož byly velice blízko těmto domům.

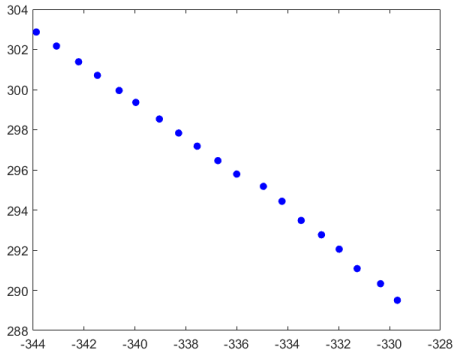
4.3.3 Detekce hrany domu

Pro další práci si vybereme cluster domu, který je na obrázku 4.22 zobrazen jako modrý dům vlevo nahoře.

V tomto malém point cloudu domu nyní sestavíme přímku z bodů domu těsně nad zemí představující stěnu domu, kterou chceme trackovat pomocí Kalmanova filtru. Tyto body nyní zobrazíme do 2D a navzorkujeme jako v předchozích případech.



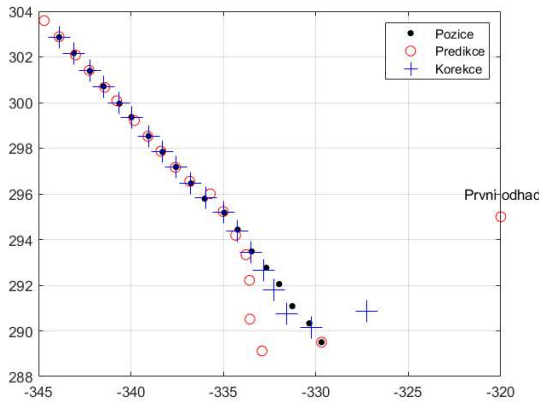
Obrázek 4.23: Vybraná část mračna bodů odpovídající jednomu domu.



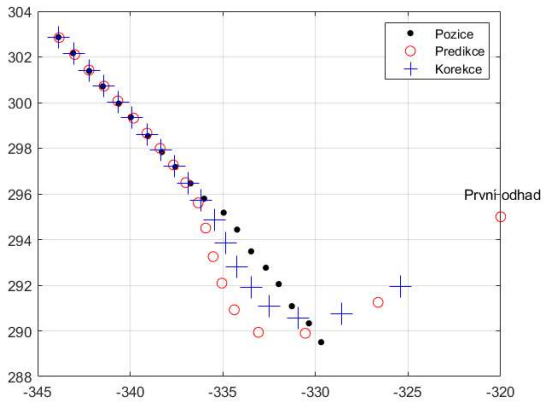
Obrázek 4.24: Navzorkování hrany domu.

4.3.4 Použití Kalmanova filtru a zobrazení výsledků

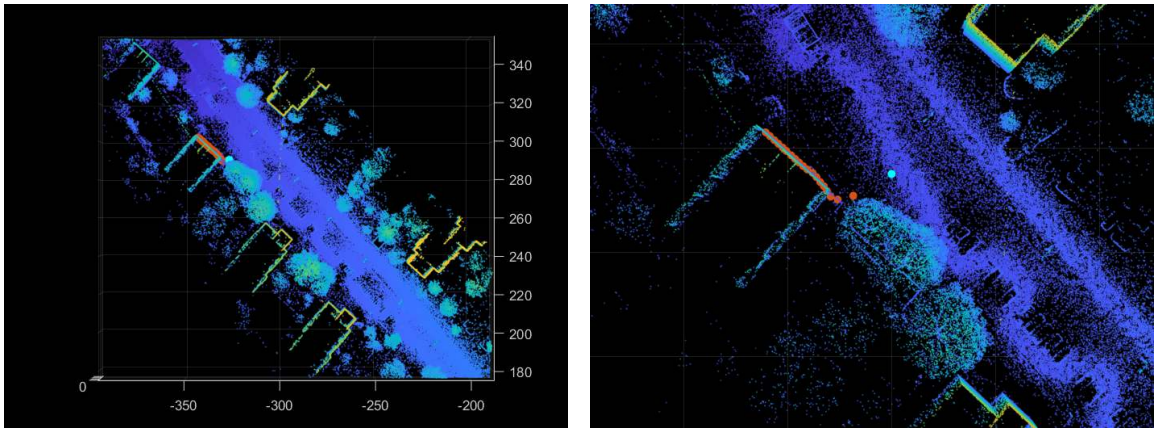
Data máme připravená, jak je zobrazeno na obrázku 4.24, můžeme tedy přejít k použití Kalmanova filtru. Opět budeme používat jak Matlab algoritmus, tak námi navržený a výsledky si zobrazíme v původním point cloudu.



Obrázek 4.25: Navržený Kalmanův filtr.



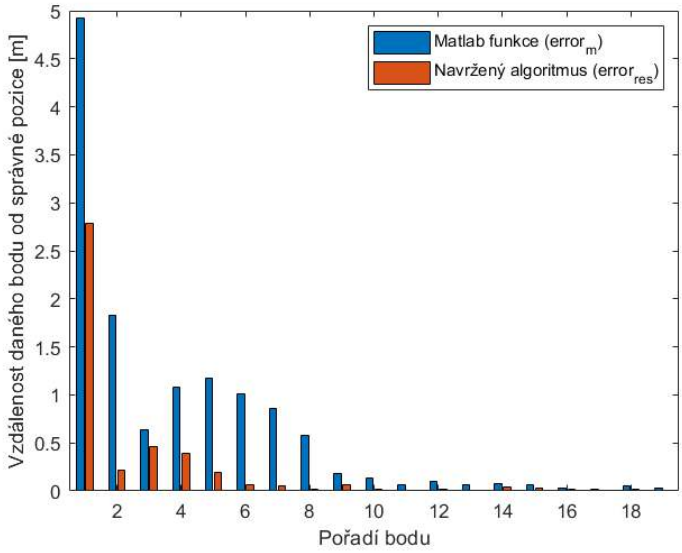
Obrázek 4.26: Kalmanův filtr implementovaný v Matlabu.



Obrázek 4.27: Zasazení do původního mračna.

4.3.5 Porovnání výsledků

Nakonec stejně jako v obou předchozích případech vypočítáme chyby obou algoritmů stejným způsobem jako v sekci 4.1.7 a zobrazíme je v grafu 4.28 a tabulce 4.3.



Obrázek 4.28: Graf chyb algoritmů.

Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$
1	491,8308	278,5761
2	183,2380	21,1309
3	63,2371	46,0596
4	107,8009	39,1977
5	116,8528	18,9768
6	100,7072	6,0562
7	85,5509	4,8354
8	58,1491	1,2875
9	18,2461	6,5333
10	12,9631	2,1778
11	6,7531	0,7094
12	9,3645	1,8653
13	6,8077	0,7612
14	7,9237	3,8423
15	6,1371	3,4334
16	3,1901	1,2855
17	2,0465	0,9349
18	5,4209	2,1565
19	2,4267	0,3435

Tabulka 4.3: Vzdálenosti od správné pozice jednotlivých bodů.

V tomto případě vidíme, že podobně jako v prvním případě rozdíl chyb při první iteraci je téměř dvojnásobný. Zároveň náš algoritmus se k malým chybám dostává již při šesté iteraci na rozdíl od Matlab funkce, která se na podobné hodnoty dostává až v jedenácté

iteraci. Je tedy vidět, že v tomto případě se náš algoritmus dostal k relativně malým chybám skoro dvakrát rychleji než Matlab funkce, i když samozřejmě ke správným hodnotám konvergují oba.

4.4 Seznam použitých mračen bodů a celkové vyhodnocení

Navržený Kalmanův filtr fungoval dobře na všech použitých datech a to i v případě, že se nejednalo pouze o přímku, jak je možné vidět v příloze A. Zde se jedná o cestu do zatáčky, což znamená, že obrubník zde bude do oblouku. I v tomto případě algoritmus dosáhl skvělých výsledků. Je třeba říci, že náš příklad na detekci hrany domu, ať vygenerované či reálné, byl poněkud uměle vytvořen s úmyslem využití clusteringu a v praxi se příliš často nepoužívá.

V tabulce 4.4 pak vidíme seznam všech mračen bodů použitých v práci a ve které části se vyskytují. Většina mračen byla pro další práci příliš veliká a bylo tedy třeba je zmenšit. V případě mračna cloudPraha.ply a cloudHouse.ply bylo použito downsamplování a pro mračna cloudHouseBrno.ply a cloudToronto.ply byly použity výřezy pro rychlejší výpočet. Pouze v případě cloudBrno.ply nebyl proveden žádný downsampling.

Pokud se rozhodneme, že se algoritmus dostatečně přiblížil skutečnosti, v případě, že chyba je pět a méně centimetrů od skutečnosti, pak můžeme v tabulce pro každá data vidět, při které iteraci z kolika byly tyto hodnoty dosaženy. Vidíme, že ve všech případech se algoritmus dostatečně přiblížil při sedmé nebo osmé iteraci, což bylo vždy podstatně rychleji než v případě Matlab funkce.

V příloze A a B je jako vstupní data použito mračno zobrazující část ulice v Torontu. V příloze C se jedná o výřez z větší verze mračna cloudHouseBrno.ply a v poslední příloze D se jedná o stejné mračno jako v sekci 4.1.

Souhrnně můžeme říci, že navržený algoritmus pracuje velice přesně a rychle na všech uvedených datech v této práci.

Název mračna	Výskyt v práci	Počet bodů mračna	Reálný počet bodů při zpracování	Výpočetní čas [s]	Chyba ≤ 5 cm
cloudPraha.ply	4.1	6 439 024	321 951	0,196	8/34
cloudHouse.ply	4.2	414 100	262 050	0,259	7/18
cloudHouseBrno.ply	4.3	1 186 864	278 284	0,047	7/19
cloudToronto.ply	A	10 283 800	504 998	0,083	7/28
cloudToronto.ply	B	10 283 800	326 277	0,025	8/27
cloudBrno.ply	C	1 881 899	1 881 899	0,082	8/32
cloudPraha.ply	D	6 439 024	321 951	0,094	8/27

Tabulka 4.4: Seznam použitých mračen bodů.

Kapitola 5

Závěr

Kalmanův filtr je nepostradatelný nástroj v mnoha různých oblastech. Hlavním úkolem této práce bylo přiblížit pojem Kalmanovy filtrace a fungování tohoto algoritmu i s jeho následnou implementací a využití na konkrétních datech mračen bodů.

V první teoretické části byl nejdříve přiblížen pojem mračen bodů a pojmů s nimi souvisejících. Dále byl pak představen samotný Kalmanův filtr. V této práci byl přiblížen pouze nejjednodušší typ Kalmanova filtru, jistě by se dalo zabývat dalšími typy Kalmanových filtrů.

V praktické části je pak představeno několik různých datasetů, které byly vždy potřebným způsobem předzpracovány pro následné použití Kalmanova filtru. Námi navržený algoritmus byl implementován v Matlabu a výsledky byly porovnány s funkcí integrovanou v Matlabu.

Obecně můžeme říci, že námi navržený algoritmus podával lepší výsledky než funkce pro Kalmanův filtr implementovaná v Matlabu. Jeho odchylka od skutečných hodnot byla od první iterace menší a klesala v každé iteraci rychleji, tedy se i dříve dostal ke skutečným hodnotám.

V praxi je možné Kalmanův filtr využít podobným způsobem, jako je tomu zde, a to například ve SLAM pro lokalizaci pozice vozidla. V této práci byl použit velice základní a jednoduchý Kalmanův filtr a v reálném využití bude komplexnější a rozšířen o další popis jako rychlost nebo zrychlení.

Literatura

[1] Anděl, J. *Základy matematické statistiky*. 3. vyd. Praha: Matfyzpres, 2011. ISBN 978-80-7378-162-0.

[2] Becker, A. *Kalman Filter from the Ground Up*. 1. vyd. Alex Becker, 2023. ISBN 9789655984392.

[3] Chen, Z., Ledoux, H., Khademi, S. a Nan, L. Reconstructing compact building models from point clouds using deep implicit fields. *ISPRS Journal of Photogrammetry and Remote Sensing*. 2022, sv. 194, s. 58–73. DOI: https://doi.org/10.1016/j.isprsjprs.2022.09.017. ISSN 0924-2716. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0924271622002611.

[4] Chui, C. a Chen, G. Kalman Filtering, with Real-Time Applications. *Kalman Filtering: With Real-Time Applications*. Leden 2009. DOI: 10.1007/978-3-540-87849-0.

[5] Dissanayake, M., Newman, P., Clark, S., Durrant Whyte, H. a Csorba, M. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Transactions on Robotics and Automation*. 2001, sv. 17, č. 3, s. 229–241.

[6] El Henawi, H. *Indirect Time-of-Flight 3D Imaging Using Fast Segmented Large-Area Electroabsorption Modulators*. Disertační práce.

[7] Fischler, M. A. a Bolles, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*. New York, NY, USA: Association for Computing Machinery. červen 1981, sv. 24, č. 6, s. 381–395. DOI: 10.1145/358669.358692. ISSN 0001-0782. Dostupné z: https://doi.org/10.1145/358669.358692.

[8] Gaia, J., Orosco, E., Rossomando, F. a Soria, C. Mapping the Landscape of SLAM Research: A Review. *IEEE Latin America Transactions*. 2023, sv. 21, č. 12, s. 1313–1336. DOI: 10.1109/TLA.2023.10305240.

[9] Grewal, M. a Andrews, A. Kalman filtering: theory and practice using MATLAB. *New York: John Wiley and Sons*. Leden 2001, sv. 14. DOI: 10.1002/9780470377819.

[10] Harvey, A. C. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, 1990.

[11] Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*. Březen 1960, sv. 82, č. 1, s. 35–45. DOI: 10.1115/1.3662552. ISSN 0021-9223. Dostupné z: https://doi.org/10.1115/1.3662552.

[12] Kim, D.-W. a Park, C.-S. Application of Kalman Filter for Estimating a Process Disturbance in a Building Space. *Sustainability*. 2017, sv. 9, č. 10. DOI: 10.3390/su9101868. ISSN 2071-1050. Dostupné z: https://www.mdpi.com/2071-1050/9/10/1868.

[13] Lacey, T. Tutorial: the Kalman Filter 11.1 Introduction 11.2 Mean Squared Error. In:. 1998. Dostupné z: https://api.semanticscholar.org/CorpusID:48574040.

[14] Lautier, D. THE KALMAN FILTER IN FINANCE: AN APPLICATION TO TERM STRUCTURE MODELS OF COMMODITY PRICES AND A COMPARISON BETWEEN THE SIMPLE AND THE EXTENDED FILTERS. Leden 2002.

[15] Lind, A. *Determining Overlap Percentage in Drone/UAV Collected Imagery* [online]. Blue Marble Geographics, 20. prosince 2022 [cit. 2024-11-26]. Dostupné z: https://www.bluemarblegeo.com/blog/determining-overlap-percentage-in-drone-collected-imagery/.

[16] Liu, Y., Obukhov, A., Wegner, J. D. a Schindler, K. *Point2CAD: Reverse Engineering CAD Models from 3D Point Clouds*. 2023. Dostupné z: https://arxiv.org/abs/2312.04962.

[17] Lobo, T. *Understanding Structure From Motion Algorithms* [online]. FIT VUT v Brně, 18. prosince 2023. Dostupné z: https://medium.com/@loboateresa/understanding-structure-from-motion-algorithms-fc034875fd0c.

[18] Mihoc, I. a Fătu, C. The orthogonality principle in the theory of the statistical estimation. Listopad 2020.

[19] Ntoulmperis, M., Catti, P., Discepolo, S., Kamp, W. van de, Castellini, P. et al. 3D point cloud analysis for surface quality inspection: A steel parts use case. *Procedia CIRP*. 2024, sv. 122, s. 509–514. DOI: https://doi.org/10.1016/j.procir.2024.01.074. ISSN 2212-8271. Dostupné z: https://www.sciencedirect.com/science/article/pii/S221282712400101X.

[20] Rukhovich, D., Dupont, E., Mallis, D., Cherenkova, K., Kacem, A. et al. *CAD-Recode: Reverse Engineering CAD Code from Point Clouds*. 2025. Dostupné z: https://arxiv.org/abs/2412.14042.

[21] Shinohara, T., YongHe, L., Sakamoto, M. a Satoh, T. Building CAD Model Reconstruction from Point Clouds via Instance Segmentation, Signed Distance Function, and Graph Cut. *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*. 2023, s. 1727–1736. DOI: 10.1109/ICCVW60793.2023.00189. ISSN 2473-9944.

[22] Simon, D. The discrete-time Kalman filter. In: *Optimal State Estimation*. John Wiley & Sons, Ltd, 2006, kap. 5, s. 121–148. DOI: https://doi.org/10.1002/0470045345.ch5. ISBN 9780470045343. Dostupné z: https://onlinelibrary.wiley.com/doi/abs/10.1002/0470045345.ch5.

[23] Tan, W., Qin, N., Ma, L., Li, Y., Du, J. et al. Toronto-3D: A large-scale mobile lidar dataset for semantic segmentation of urban roadways. In: *Proceedings of the*

IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 2020, s. 202–203.

[24] Westoby, M., Brasington, J., Glasser, N., Hambrey, M. a Reynolds, J. Structure-from-Motion photogrammetry: a novel, low-cost tool for geomorphological applications. Duben 2012, s. 936–.

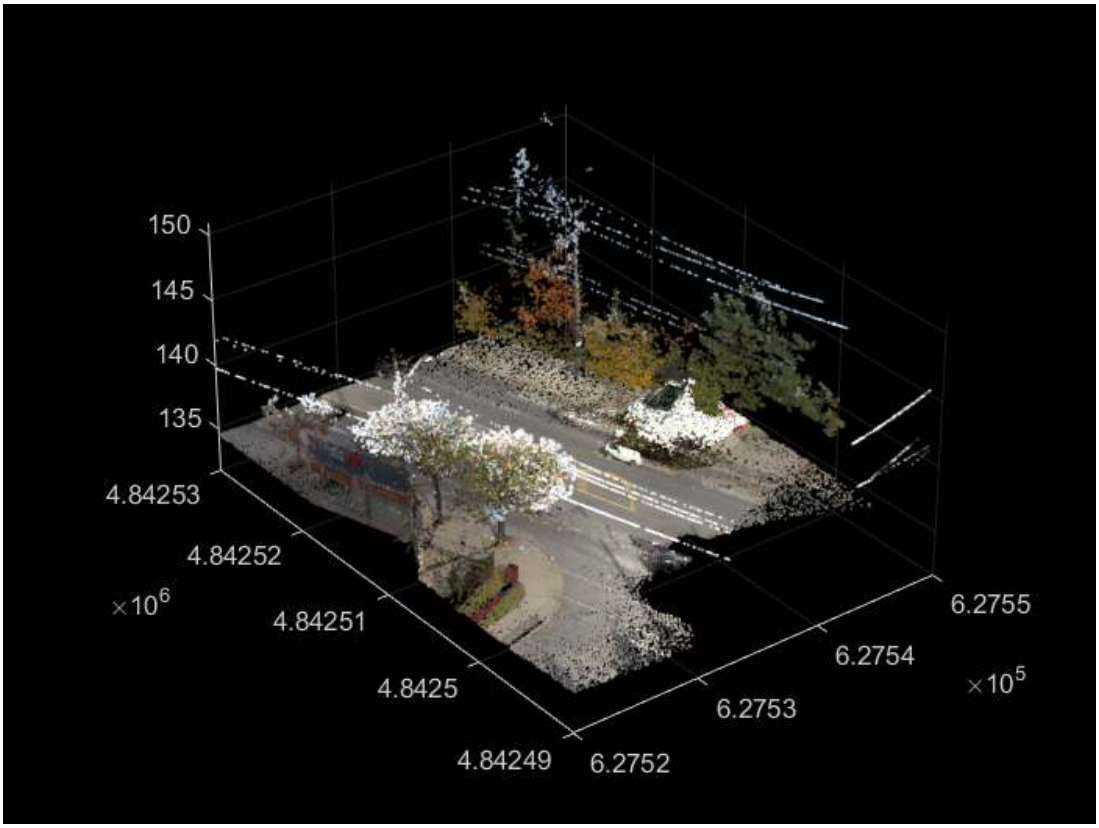
Přílohy

Seznam příloh

Příloha A

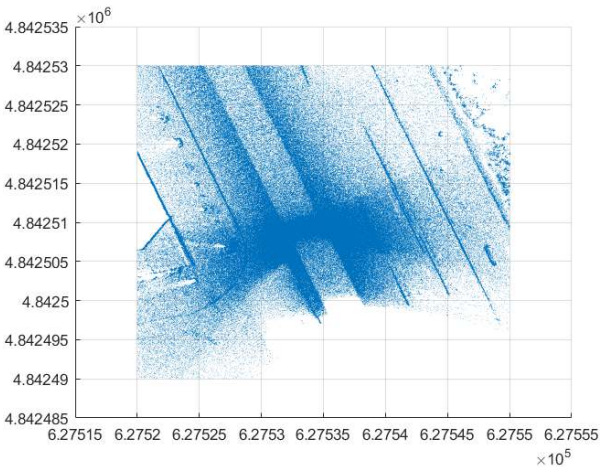
Point Cloud 1

První data, která si v příloze přestavíme je možné vidět na obrázku A.1. Jedná se o data z [23]. Hned na obrázku A.1 nebo A.2 můžeme vidět jisté problémy v těchto datech, jako například šum vznikající od pohybujících se aut. Pro naše potřeby jsou však plně postačující, musíme si pouze poradit s větším množstvím šumu.

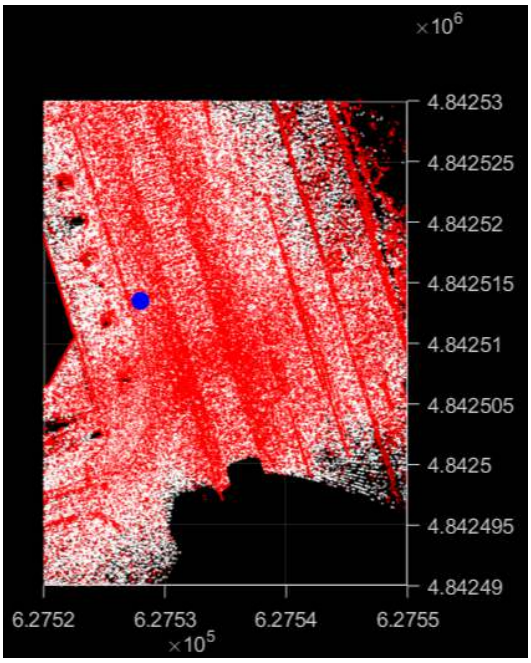


Obrázek A.1: Vstupní mračno bodů, ulice Toronto, zdroj [23], počet bodů (10 283 800).

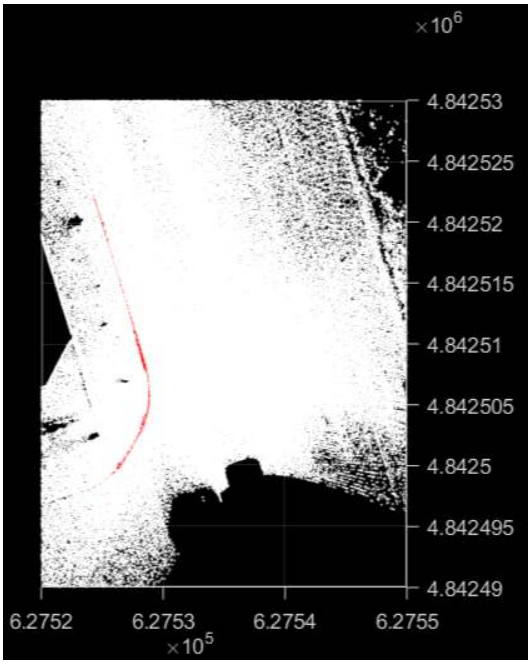
Obrázek A.2: Pozemní body.



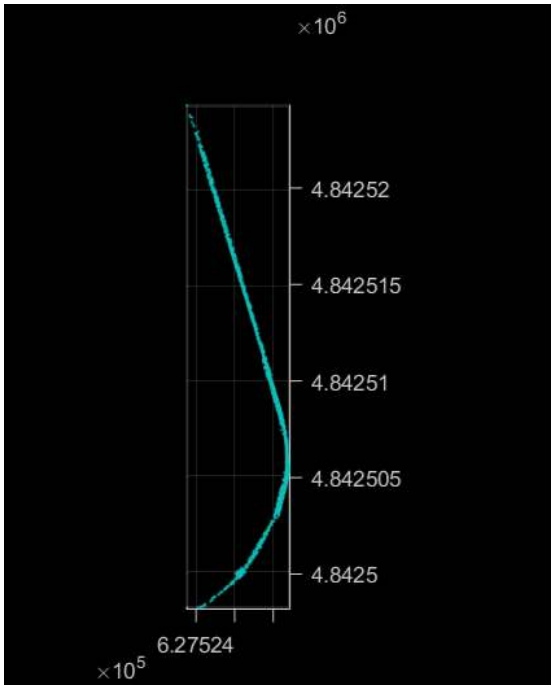
Obrázek A.3: Vstupní mračno bodů s vypočítanými normálami.



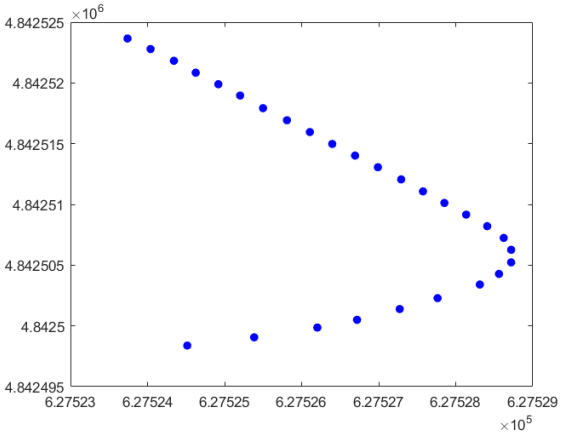
Obrázek A.4: Červeně zobrazené body s normálami splňujícími podmínku $\varphi < \text{thr}$.



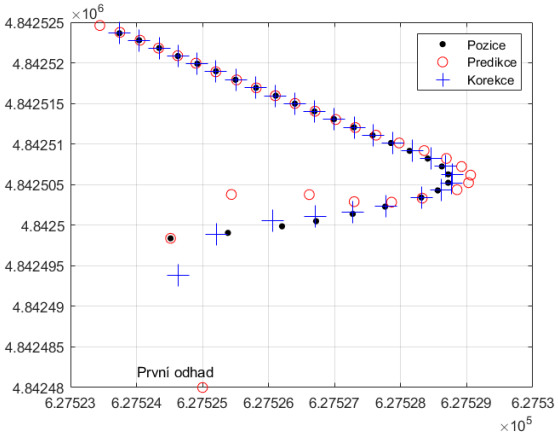
Obrázek A.5: Oddělený obrubník (červeně označené body).



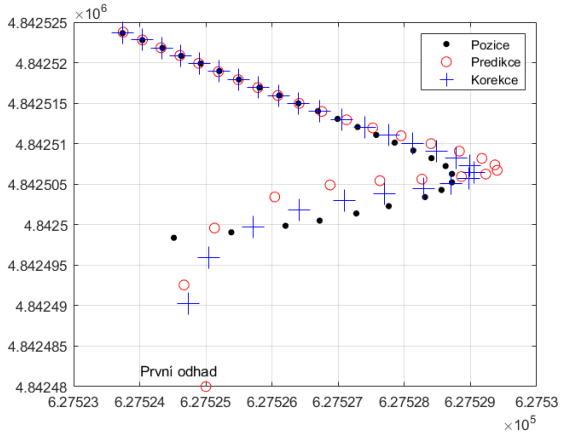
Obrázek A.6: Promítnutí do 2D.



Obrázek A.7: Navzorkovaný obrubník.



Obrázek A.8: Navržený Kalmanův filtr.

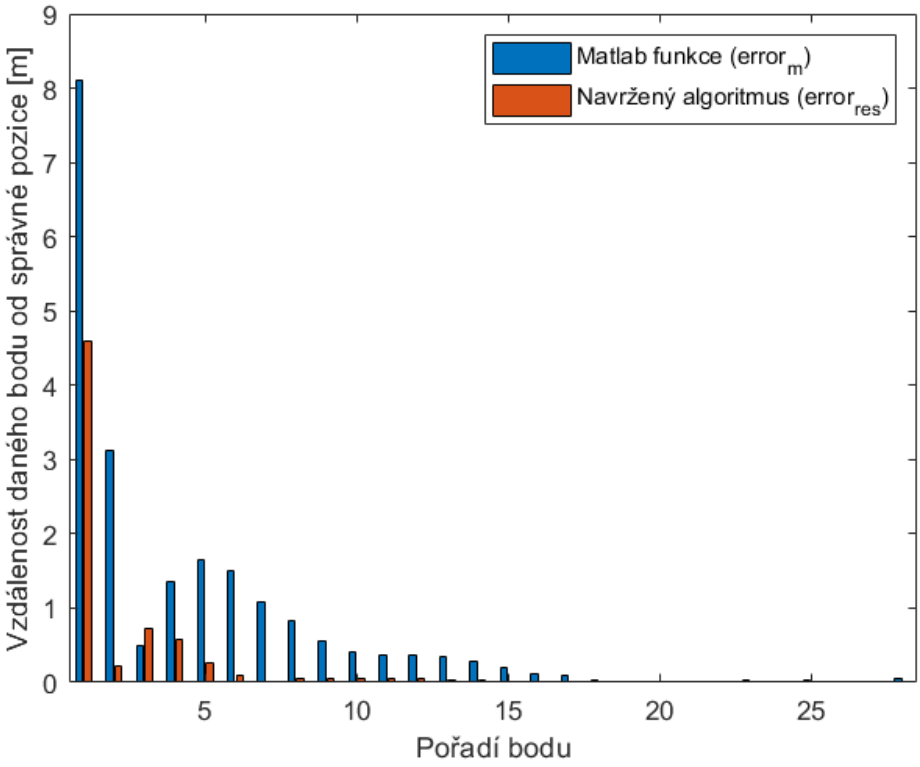


Obrázek A.9: Kalmanův filtr implementovaný v Matlabu.

Obrázek A.10: Zasazení do původního mračna.

Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$	Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$
1	811,095	459,409	15	17,197	0,919
2	312,050	22,208	16	10,353	0,276
3	50,409	72,022	17	8,325	1,322
4	135,706	58,322	18	3,570	0,291
5	164,525	26,973	19	1,821	0,251
6	150,622	9,955	20	1,088	0,300
7	108,403	1,377	21	0,988	0,127
8	83,042	5,462	22	2,090	0,838
9	55,848	5,396	23	3,422	1,008
10	40,286	6,298	24	1,767	1,357
11	37,827	5,431	25	3,003	0,838
12	39,347	5,561	26	4,164	0,818
13	36,428	4,169	27	7,308	3,609
14	26,484	1,333	28	8,445	4,140

Tabulka A.1: Vzdálenosti od správné pozice jednotlivých bodů.

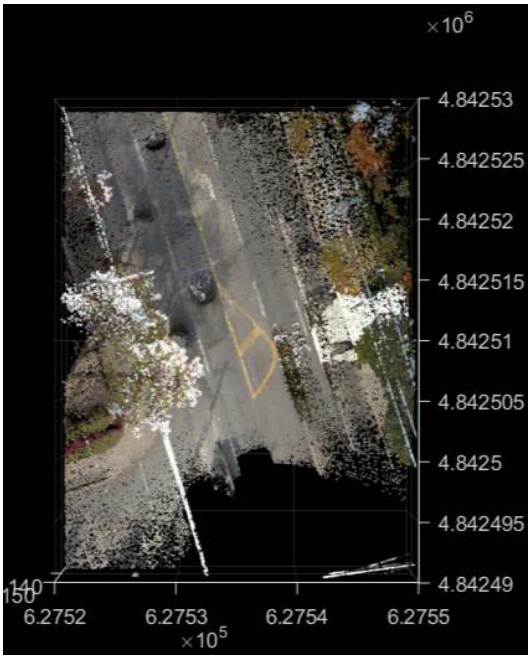


Obrázek A.11: Graf chyb algoritmů.

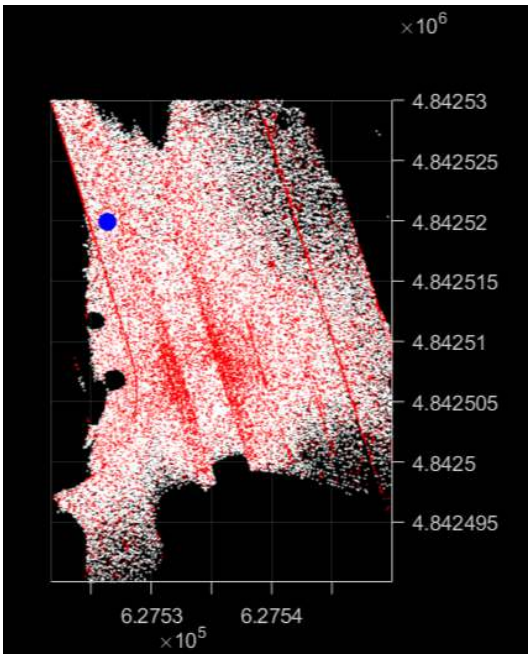
Příloha B

Point Cloud 2

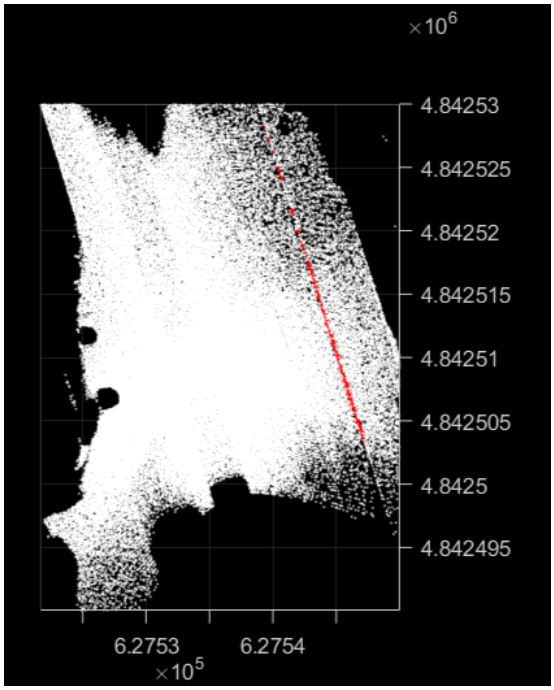
Druhý point cloud je stejný jako v předchozí části, rozdíl bude pouze ten, že tentokrát budeme trackovat druhý obrubník.



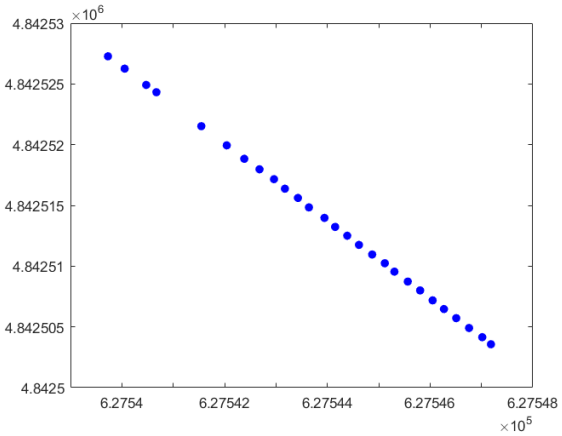
Obrázek B.1: Vstupní mračno bodů, ulice Toronto, zdroj [23], počet bodů (10 283 800).



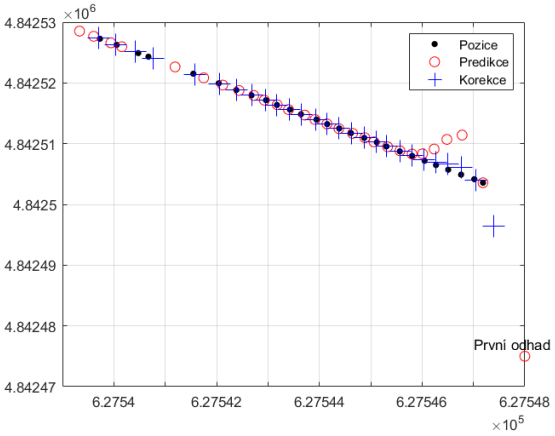
Obrázek B.2: Červeně zobrazené body s normálami splňujícími podmínku $\varphi <$ thr.



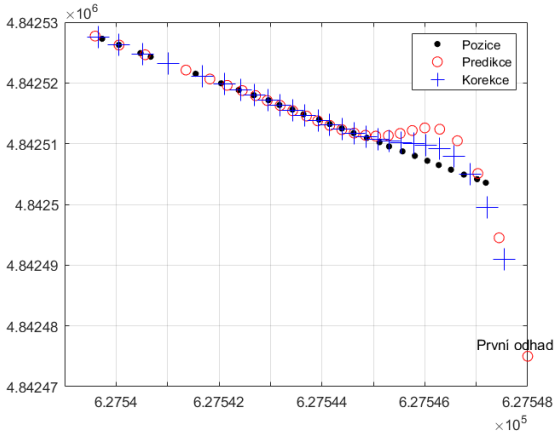
Obrázek B.3: Oddělený obrubník (červeně označené body).



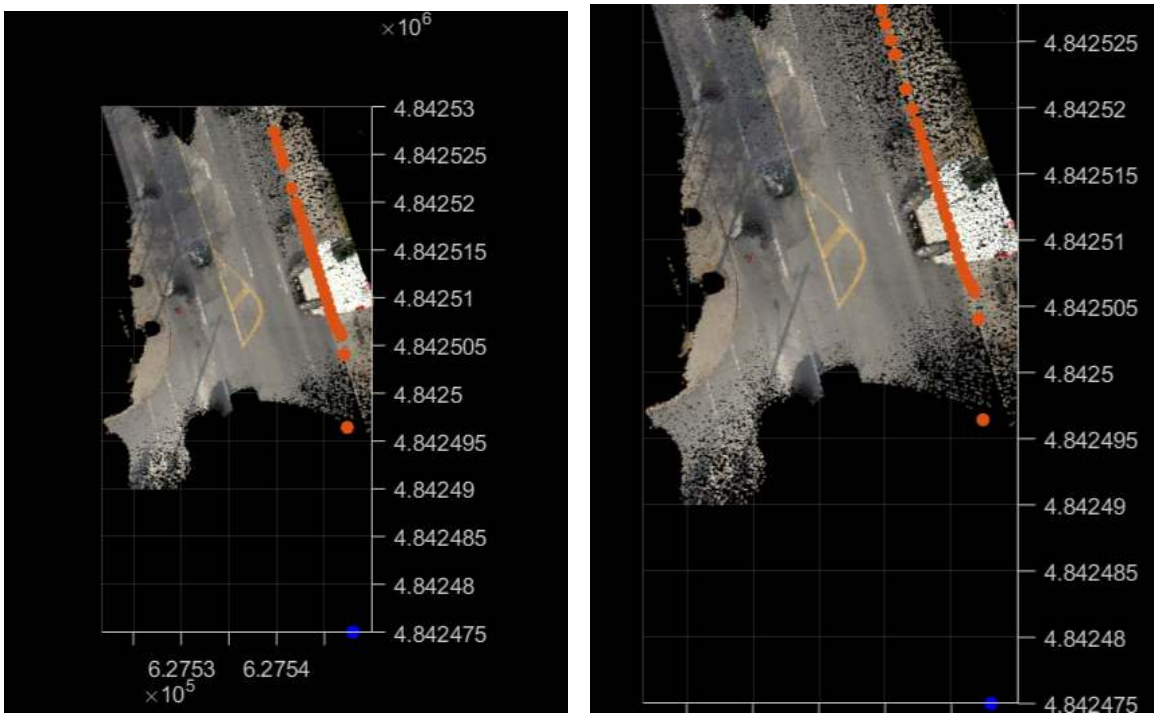
Obrázek B.4: Navzorkovaný obrubník.



Obrázek B.5: Navržený Kalmanův filtr.



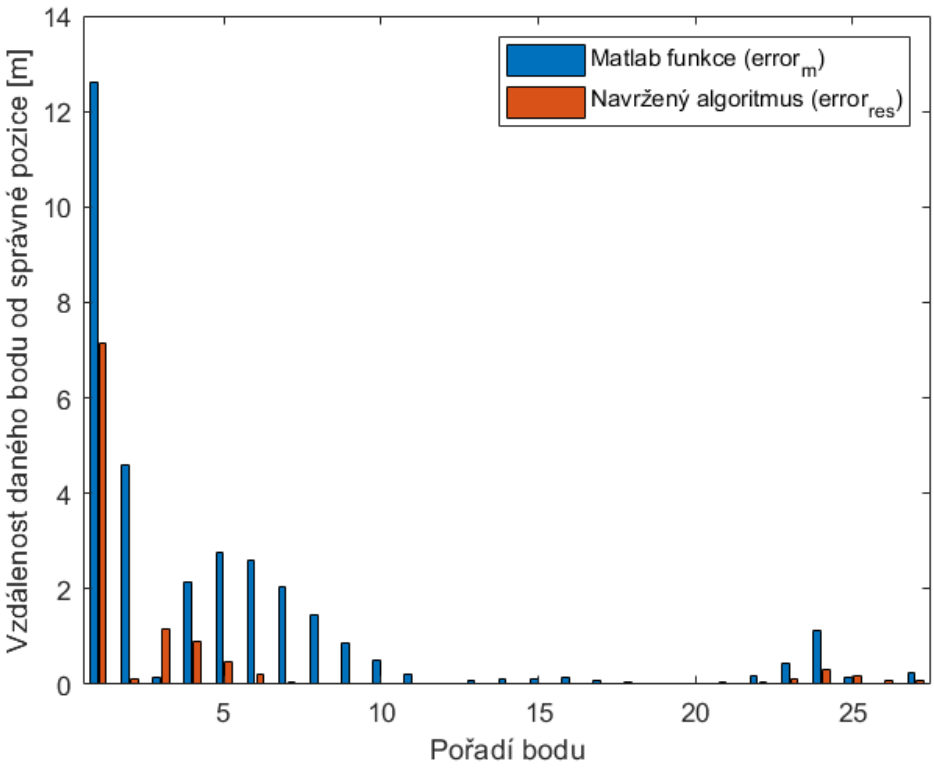
Obrázek B.6: Kalmanův filtr implementovaný v Matlabu.



Obrázek B.7: Zasazení do původního mračna.

Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$	Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$
1	1261,515	714,530	15	10,227	0,525
2	459,863	12,174	16	14,598	2,546
3	15,048	117,257	17	9,457	0,502
4	215,812	88,613	18	4,490	0,662
5	274,875	47,131	19	2,780	0,554
6	258,945	20,557	20	2,880	0,965
7	203,343	5,334	21	5,769	1,713
8	145,667	1,851	22	17,678	5,357
9	87,702	1,170	23	45,230	12,628
10	50,314	1,553	24	114,076	31,331
11	21,106	0,501	25	14,492	19,455
12	0,627	1,183	26	0,855	6,546
13	9,259	0,319	27	22,919	7,552
14	10,503	0,527			

Tabulka B.1: Vzdálenosti od správné pozice jednotlivých bodů.

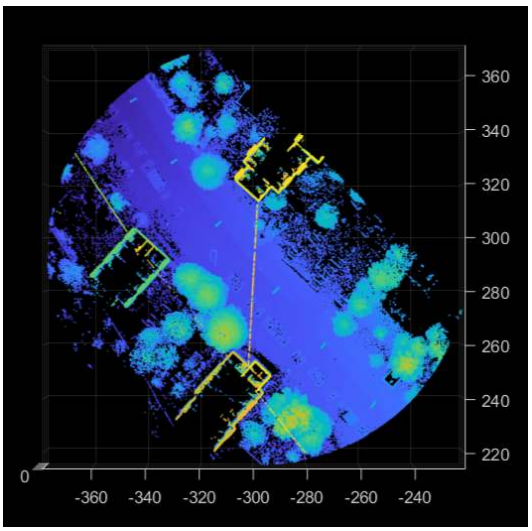


Obrázek B.8: Graf chyb algoritmů.

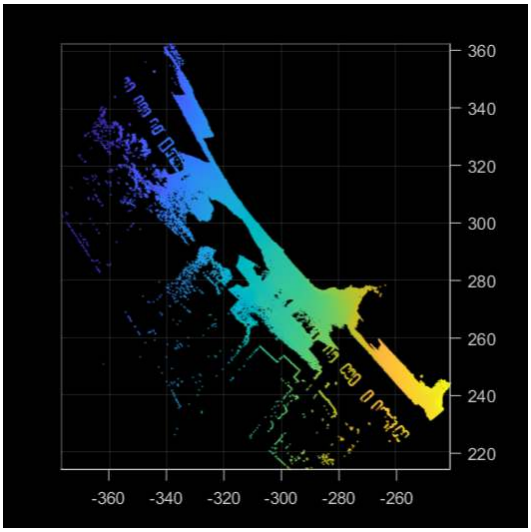
Příloha C

Point Cloud 3

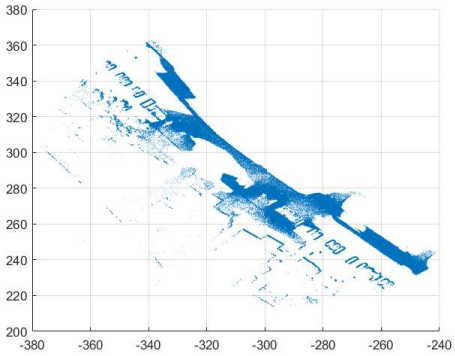
V tomto případě se jedná o stejná data jako byla představena v části 4.3, s jediným rozdílem, že nyní budeme pracovat s výřezem vyobrazeným na obrázku C.1.



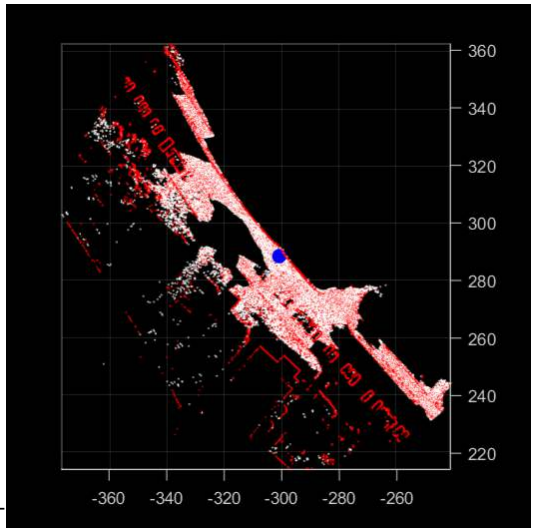
Obrázek C.1: Vstupní mračno bodů, ulice Kosmonautů Brno, počet bodů (1 881 899).



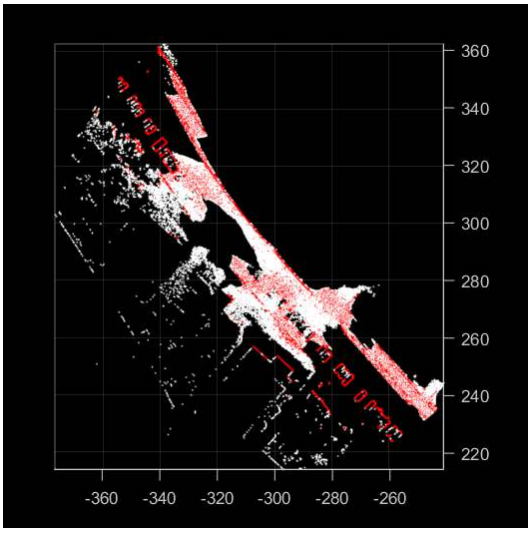
Obrázek C.2: Pozemní body.



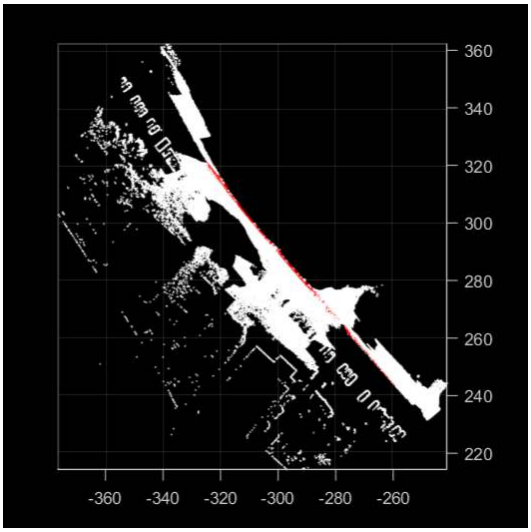
Obrázek C.3: Vstupní mračno bodů s vypočítanými normálami.



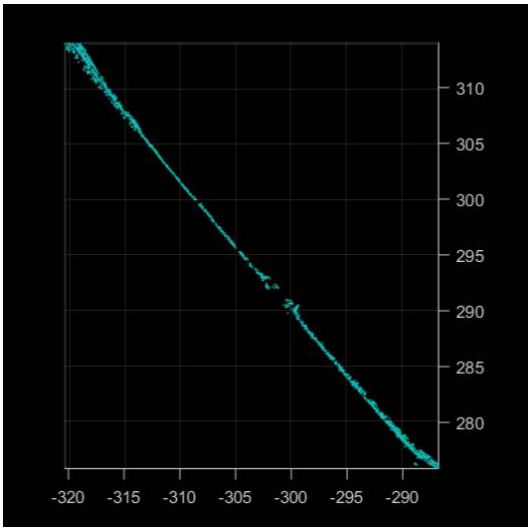
Obrázek C.4: Červeně zobrazené body s normálami splňujícími podmínku $\varphi < \text{thr}$.



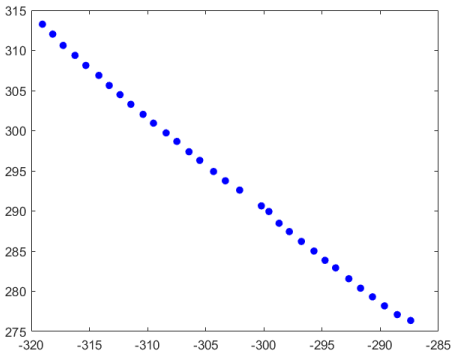
Obrázek C.5: Výběr obrubníku.



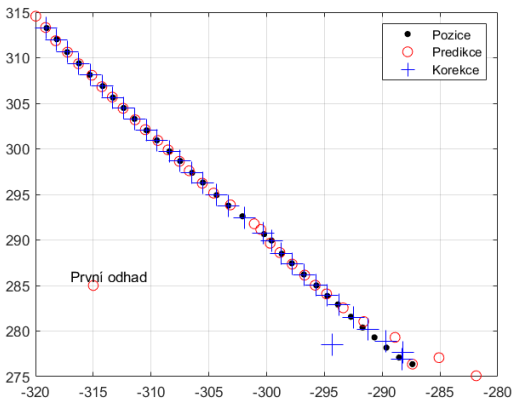
Obrázek C.6: Oddělený obrubník (červeně označené body).



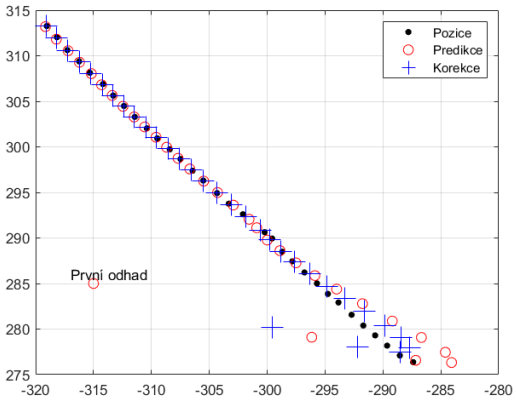
Obrázek C.7: Promítnutí do 2D.



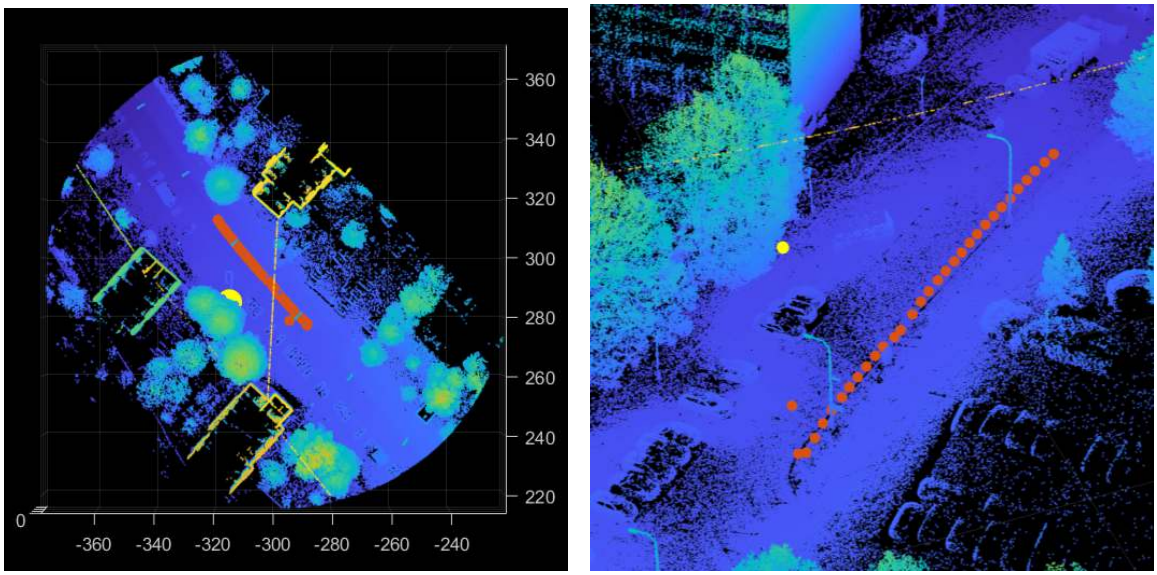
Obrázek C.8: Navzorkovaná obrubník.



Obrázek C.9: Navržený Kalmanův filtr.



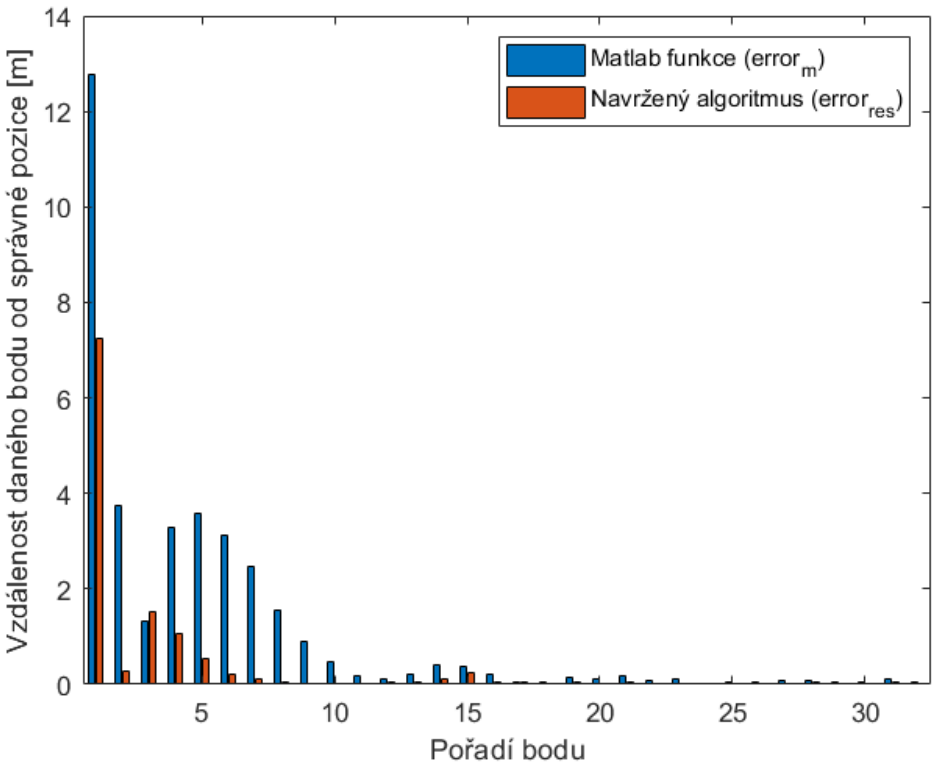
Obrázek C.10: Kalmanův filtr implementovaný v Matlabu.



Obrázek C.11: Zasazení do původního mračna.

Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$	Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$
1	1276,866	723,225	17	3,841	5,976
2	374,503	27,370	18	3,512	1,674
3	134,074	150,708	19	15,291	5,971
4	327,757	106,951	20	11,164	1,348
5	356,971	53,588	21	17,814	3,523
6	312,354	21,928	22	7,569	1,506
7	247,446	10,569	23	12,087	2,336
8	156,279	4,125	24	2,264	2,069
9	89,666	1,766	25	4,556	1,175
10	47,794	1,694	26	4,336	1,212
11	17,616	1,627	27	6,768	1,724
12	11,713	3,617	28	7,480	3,363
13	22,341	5,832	29	5,107	0,921
14	41,119	10,427	30	4,688	0,501
15	37,695	23,354	31	10,829	3,442
16	21,031	3,589	32	6,384	1,827

Tabulka C.1: Vzdálenosti od správné pozice jednotlivých bodů.

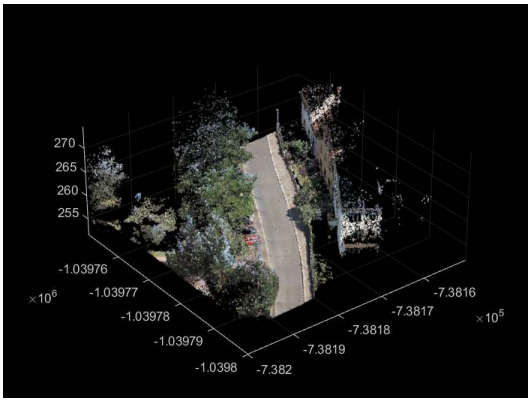


Obrázek C.12: Graf chyb algoritmů.

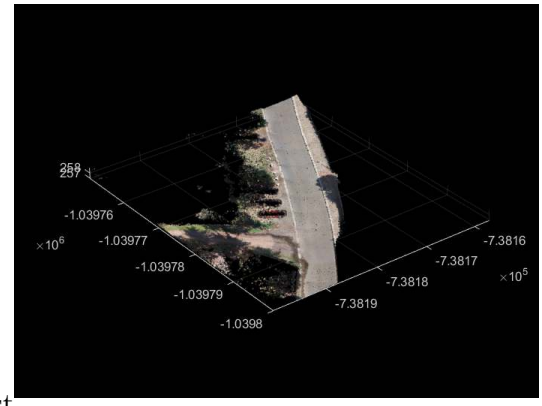
Příloha D

Point Cloud 4

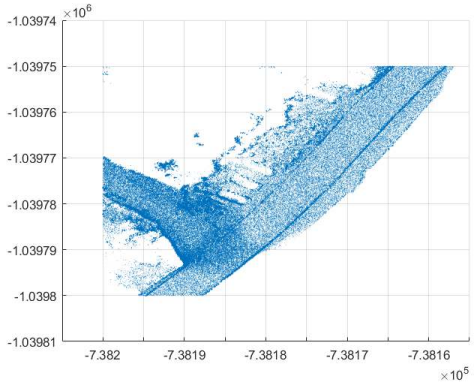
V této části budeme pracovat s mračnem, které bylo představeno v části 4.1. Tentokrát se však budeme zabývat druhým obrubníkem.



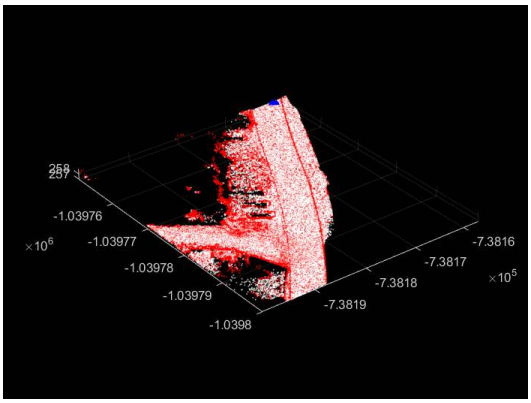
Obrázek D.1: Vstupní mračno bodů, oblast zástavba Praha, zdroj Cyclomedia, počet bodů (6 439 024).



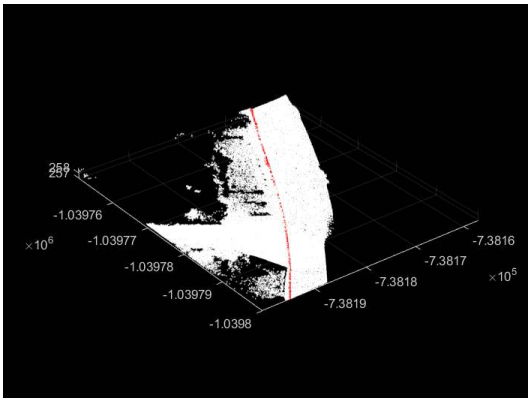
Obrázek D.2: Pozemní body.



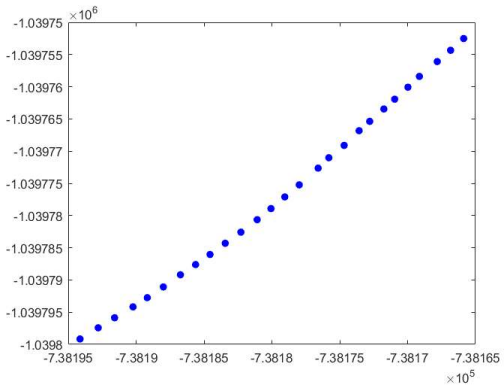
Obrázek D.3: Vstupní mračno bodů s vypočítanými normálami.



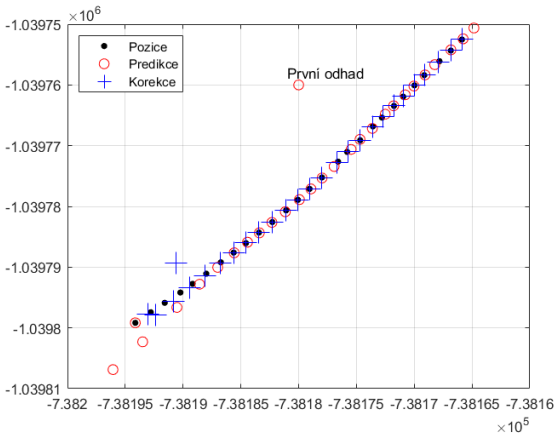
Obrázek D.4: Červeně zobrazené body s normálami splňujícími podmínku $\varphi < \text{thr}$.



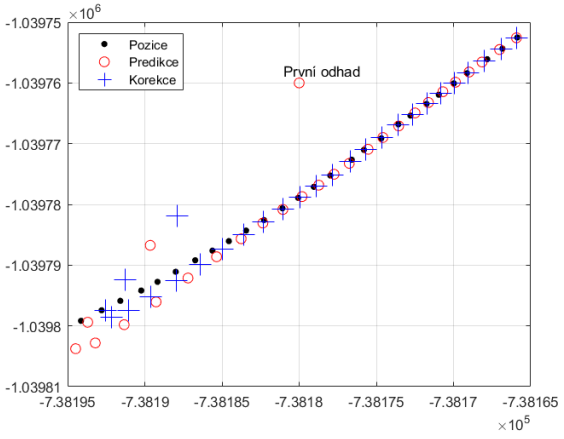
Obrázek D.5: Oddělený obrubník (červeně označené body).



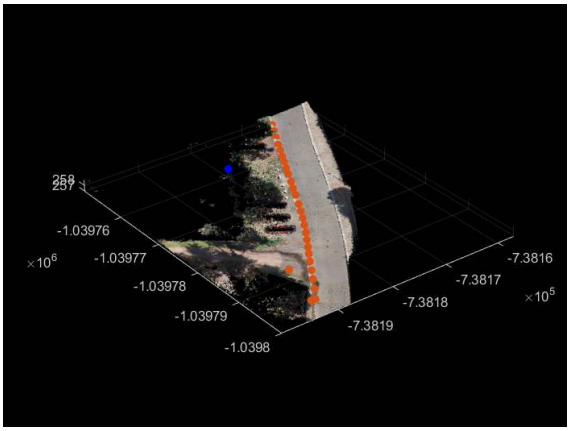
Obrázek D.6: Navzorkovaný obrubník.

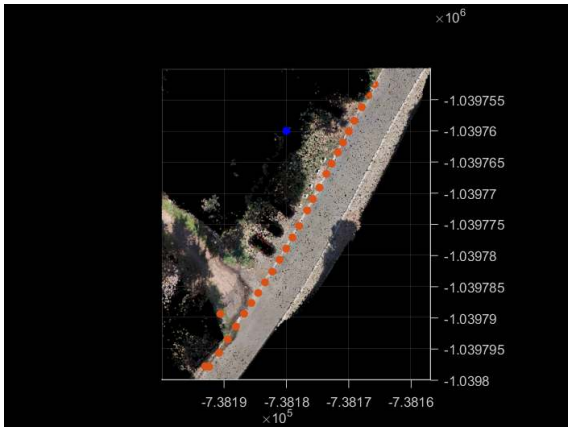


Obrázek D.7: Navržený Kalmanův filtr.



Obrázek D.8: Kalmanův filtr implementovaný v Matlabu.

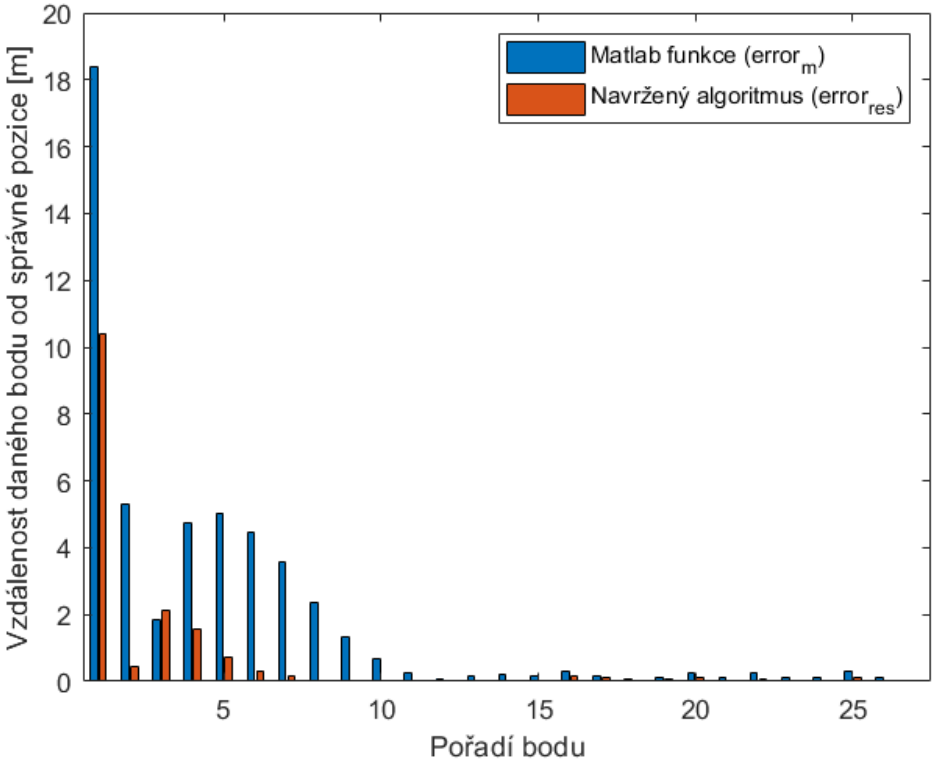




Obrázek D.9: Zasazení do původního mračna.

Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$	Pořadí bodu	$error_m$ $[cm]$	$error_{res}$ $[cm]$
1	1837,719	1040,895	15	15,734	1,389
2	530,474	43,842	16	30,682	15,248
3	186,812	213,931	17	14,209	9,506
4	474,157	155,588	18	7,820	2,642
5	503,877	73,107	19	12,769	6,011
6	446,878	32,437	20	26,094	11,152
7	356,344	15,613	21	13,098	0,666
8	235,412	0,879	22	27,377	6,212
9	133,114	3,601	23	10,771	2,085
10	68,488	1,139	24	11,563	1,385
11	24,794	1,228	25	28,293	12,438
12	9,283	4,347	26	11,880	2,031
13	15,920	2,415	27	3,794	2,613
14	20,865	2,068			

Tabulka D.1: Vzdálenosti od správné pozice jednotlivých bodů.



Obrázek D.10: Graf chyb algoritmů.