F3

Fakulta elektrotechnická
Katedra kybernetiky

Bakalářská práce

Geometrická reprezentace sloupů vysokého napětí

Petr Hrnčíř

Květen 2024

Vedoucí práce: RNDr. Petr Štěpán, Ph.D.

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Hrnčíř** Jméno: **Petr** Osobní číslo: **507326**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra kybernetiky**

Studijní program: **Otevřená informatika**

Specializace: **Základy umělé inteligence a počítačových věd**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Geometrická reprezentace sloupů vysokého napětí

Název bakalářské práce anglicky:

Geometric Representation of High Voltage Pylons

Pokyny pro vypracování:

1) Seznamte se s principem činnosti Lidarů a algoritmy pro zpracování dat z Lidarů.
2) Navrhněte algoritmus, který bude v prostředí v blízkosti stožáru vysokého napětí vytvářet geometrický model stožáru a současně lokalizovat dron v 3D světě za pomoci výškových senzorů dronu.
3) Vyhodnoťte spolehlivost detekce geometrických primitiv v jednotlivých datech od lidaru v závislosti na vzdálenosti od stožáru. Vyhodnoťte přesnost vytvořeného modelu pro dodané typy stožárů.

Seznam doporučené literatury:

[1] J.Mandík:“Lokalizace v prostředí stožáru vysokého napětí“, bakalářská práce, katedra kybernetiky, FEL, ČVU v Praze, 2022
[2] Kusenbach, M., Himmelsbach, M., & Wuensche, H. J. (2016, June). A new geometric 3D LiDAR feature for model creation and classification of moving objects. In 2016 IEEE Intelligent Vehicles Symposium (IV) (pp. 272-278). IEEE.
[3] Im, B., Baek, N., & Lee, J. (2016, September). An Efficient Implenentation of LiDAR Data and its Geometric Representation Extraction. In 2016 6th International Conference on IT Convergence and Security (ICITCS) (pp. 1-4). IEEE.
[4] Zhao, Y., Bai, L., Zhang, Z., & Huang, X. (2021). A surface geometry model for lidar depth completion. IEEE Robotics and Automation Letters, 6(3), 4457-4464.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

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

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

Datum zadání bakalářské práce: **31.01.2024** Termín odevzdání bakalářské práce: **24.05.2024**

Platnost zadání bakalářské práce: **21.09.2025**

RNDr. Petr Štěpán, Ph.D.	prof. Dr. Ing. Jan Kybic	prof. Mgr. Petr Páta, Ph.D.
podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

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

. Datum převzetí zadání

Podpis studenta

Poděkování / Prohlášení

Chtěl bych poděkovat své rodině za podporu nejen finanční v průběhu celého studia a za to, že se mohu věnovat tomu, co mě baví.

Také bych chtěl poděkovat RNDr. Petru Štěpánovi, Ph.D. za vedení a cenné rady.

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

V Pelhřimově dne 24. 5. 2024

.....

Abstrakt / Abstract

Bakalářská práce se zabývá geometrickou reprezentací sloupů vysokého napětí. Cílem bylo vytvořit algoritmus detekce stožáru v mračnu bodů (pointcloud) získaného z LiDARu od společnosti Ouster, který jeho konstrukční části extrahuje v podobě úseček. Detekce se provádí pomocí algoritmu RANSAC. Ze získaných úseček se vytváří geometrický model stožáru a současně se lokalizuje dron ve 3D světě za pomoci výškových senzorů dronu. Vytváření modelu a lokalizace se provádí algoritmem SLAM ve spojení s algoritmem ICL nebo ICP pro registraci úseček ze dvou snímků, tj. nalezení optimální rotace a translace mezi snímky. Algoritmus je implementován v rámci systému ROS. Otestování algoritmu bylo provedeno v Gazebo simulátoru a na reálných datech.

Klíčová slova: geometrická reprezentace sloupu, stožáru vysokého napětí; Ouster, LiDAR, mračno bodů; RANSAC, ICL, ICP, SLAM, lokalizace, dron, výškový senzor; ROS, Gazebo

The bachelor thesis deals with the geometric representation of high voltage pylons. The aim was to develop an algorithm for detecting a pylon in a point cloud obtained from LiDAR by Ouster, which extracts its structural parts in the form of line segments. The detection is performed using the RANSAC algorithm. The extracted segments are used to create a geometric model of the tower and simultaneously locate the drone in the 3D world using the drone's height sensors. The model creation and localization is done by SLAM algorithm in conjunction with ICL or ICP algorithm to register the segments from two frames, i.e., finding the optimal rotation and translation between frames. The algorithm is implemented within the ROS system. Testing of the algorithm was performed in the Gazebo simulator and on real data.

Keywords: geometric representation of high voltage pylons; Ouster, LiDAR, pointcloud; RANSAC, ICL, ICP, SLAM, localization, UAV, rangefinder; ROS, Gazebo

Title translation: Geometric Representation of High Voltage Pylons

/ Obsah

Tabulky / Obrázky

Kapitola 1
Úvod

V oblasti autonomní robotiky a dronů neboli UAVs (unmanned aerial vehicles) je základním problémem lokalizace v neznámém prostředí za pomoci senzorů robota/dronu. Problém je klíčový pro plánování a vyhnutí se překážkám v prostoru. Jedním z běžně používaných senzorů je LiDAR. Pomocí dvou mračen bodů z něho získaných se určí poloha dronu. Pro nejpřesnější lokalizaci je zapotřebí, aby se v prostředí vyskytovali statické dostatečně výrazné prvky. Dalším často používaným senzorem pro lokalizaci je GPS (Global Positioning System), jehož signál se dá ale snadno zarušit obzvlášť v okolí elektrického vedení, od elektromagnetického pole vyskytující se v okolí vodičů. Proto jsou zapotřebí i algoritmy řešící problém lokalizace za pomoci i jiných senzorů.

V dnešní době se drony používají především k inspekci stožárů vysokého napětí, ke kontrole citlivých míst, zda není součást stožáru příliš opotřebena nebo poškozena. Snahou je tento proces dále zefektivnit a zautomatizovat. Cílem je více využívat drony při údržbě, kolaboraci s dělníky a zajišťování jejich bezpečnosti. Dalším cílem je snižování počtu potřebných proškolených operátorů a tím zlevnění celého procesu, až po autonomní údržbu sloupů vysokého napětí.

V této práci se zabýváme geometrickou reprezentací sloupů vysokého napětí, využitou při lokalizaci dronu. Cílem bylo navrhnout algoritmus, který z LiDARových dat dokáže detekovat konstrukční části stožáru a extrahovat je v podobě úseček. Dále za pomoci dalších snímků z LiDARu stožár rekonstruovat a s tím paralelně řešit problém lokalizace dronu ve 3D světě pomocí výškových senzorů dronu, konkrétně senzor *rangefinder*, tj. laserový paprsek namířený směrem dolů k určení vzdálenosti dronu od země.

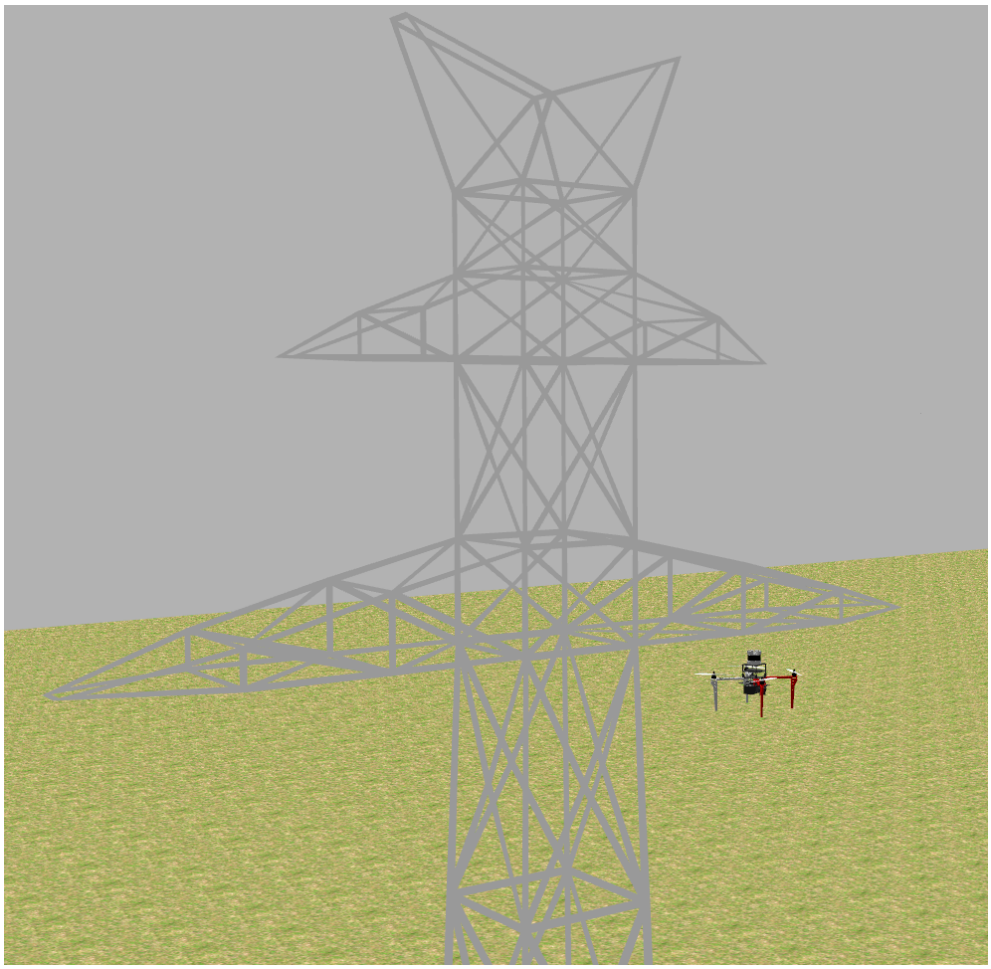
Nejdříve představíme zařízení LiDAR. Popíšeme základní princip jeho fungování, modely od společnosti Ouster, jejich vlastnosti a parametry modelu použitého v našem případě. Dále vysvětlíme výstup tohoto senzoru, se kterým se dále pracuje.

V následující kapitole uvedeme algoritmus pro detekci stožáru vysokého napětí. Předtím ale vysvětlíme fungování algoritmu RANSAC, který se v algoritmu používá. Poté popíšeme a okomentujeme jednotlivé části algoritmu pro detekci stožáru a zhodnotíme funkčnost algoritmu.

Když získáváme úsečky z dat LiDARu, můžeme je využít k řešení problému lokalizace a s tím i postupně vytvářet geometrický model stožáru. K tomu si nejdříve představíme dva potřebné algoritmy, ICL (Iterative Closest Line) a ICP (Iterative Closest Point), které jsme zvolili pro řešení tohoto problému. Algoritmus ICL jsme navrhli tak, aby mohl přímo počítat s úsečkami. Zatímco u ICP algoritmu jsme museli upravit úsečky do podoby mračen bodů, aby s nimi mohl ICP algoritmus počítat. Jeden z těchto dvou přístupů se následně uplatní v SLAM (Simultaneous Localization And Mapping) algoritmu, který jsme použili pro lokalizaci a rekonstrukci stožáru. Konkrétně jeho variantu s ICL nebo ICP algoritmem. ICL/ICP SLAM algoritmus řeší problém lokalizace a vytváření/aktualizování mapy robota/dronu v neznámém prostředí, kde mapa je v našem případě geometrický model stožáru vysokého napětí.

Dále se zaměříme na popis programu, ve kterém byl algoritmus implementován. Vysvětlíme základní koncepty systému ROS (Robot Operating System) a předávání dat mezi programem a LiDARem. Také ukážeme dvě struktury, které reprezentují geometrický model stožáru.

A konečně otestujeme algoritmus v Gazebo simulátoru na třech různých typech stožárů vysokého napětí v závislosti na vzdálenosti od stožáru a na reálných datech.



Obrázek 1.1. UAV se stožárem vysokého napětí v Gazebo simulátoru

Kapitola 2
LiDAR

Light Detection And Ranging (zkráceně LiDAR) je metoda měření vzdálenosti mezi senzorem a povrchem objektu za pomocí laseru. Metoda funguje na principu změření doby šíření laserového pulsu odraženého od povrchu objektu zpět k senzoru a následného výpočtu vzdálenosti pomocí vzorce

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

kde d je vzdálenost senzoru od objektu, t je doba šíření laserového pulsu a c je rychlost světla.

Zařízení nemusí měřit vzdálenost pouze v jednom směru, ale pomocí otáčejícího a naklánějícího zrcadla může měřit vzdálenost v různých směrech. Výstupem je potom tzv. *mračno bodů*, více v sekci 2.2.

Vlnová délka laseru se nejčastěji pohybuje okolo 600–1000 nm. Maximální výkon je limitován z důvodu nebezpečí poškození lidského oka. [1]

2.1 Ouster LiDAR

Data využívaná v projektu byla nasnímána pomocí LiDARu od společnosti `Ouster`[1]. Konkrétně byl použit model **OS0** s vertikálním rozlišením 128 paprsků, horizontálním rozlišením 1024 paprsků a frekvencí otáčení 10 Hz.

2.1.1 Parametry LiDARů

- **Vertikální rozlišení** (počet řádků) je počet paprsků vysílaných ve svislém směru.
- **Horizontální rozlišení** (počet sloupců) je počet paprsků vysílaných ve vodorovném směru.
- **Vertikální FOV** (Field of View) udává úhel v jehož rozsahu jsou paprsky vysílány ve svislém směru.
- **Horizontální FOV** (Field of View) udává úhel v jehož rozsahu jsou paprsky vysílány ve vodorovném směru.

2.1.2 Modely

Společnost Ouster vyrábí 4 hlavní modely:

model	dosah [m]	min. dosah [m]	vertikální FOV [°]	příkon [W]
OSDome	20–45	0.5	180	14–20
OS0	35–75	0.5	90	14–20
OS1	90–170	0.5	45	14–20
OS2	200–350	0.8	22.5	18–24

Tabulka 2.1. Vlastnosti LiDARů od Ouster Inc. [2]

[1] `https://ouster.com`

Zároveň všechny modely mají společné tyto vlastnosti:

- vlnovou délku laseru 865 nm
- vertikální rozlišení 32, 64 nebo 128 paprsků
- horizontální rozlišení 512, 1024 nebo 2048 paprsků
- horizontální FOV 360°
- frekvenci otáčení 10 nebo 20 Hz

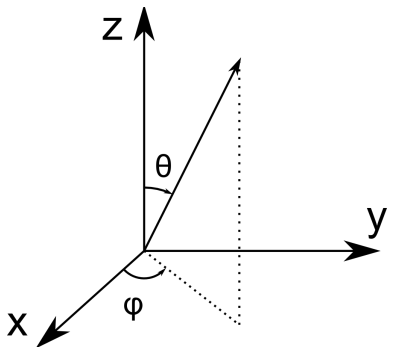
Obrázek 2.1. 4 hlavní modely Ouster Inc., OSDome je ten nejvíce vlevo a následují vzestupně seřazeny ostatní modely. [3]

2.2 Výstup LiDARu

Ke každému vyslanému paprsku známe vzdálenost od objektu d a jeho směr určený úhly φ a θ podle obrázku 2.2, které reprezentují natočení a náklon zrcadla. Abychom mohli s daty lépe pracovat je potřeba tyto hodnoty přepočítat z sférických souřadnic do kartézských souřadnic pomocí následujících vzorců.

$$(d, \varphi, \theta) \mapsto (x, y, z)$$

$$\begin{aligned} x &= d \sin\theta \cos\varphi \\ y &= d \sin\theta \sin\varphi \\ z &= d \cos\theta \end{aligned}$$



Obrázek 2.2. Sférické souřadnice [4]

Společnost Ouster dává k dispozici open-source software, který zajišťuje komunikaci se senzorem a výše zmíněný přepočet. Výstupem LiDARu jsou jednotlivé body v souřadném systému senzoru, tzv. *mračno bodů (point cloud)*, se kterým uživatel dále pracuje dle svých potřeb. Pro paprsky, jejichž odraz senzor nezachytí, jsou příslušné body nastaveny na $(0, 0, 0)$.

Obrázek 2.3. Ukázka mračna bodů

Při našem výše zmíněném nastavení LiDARu (vertikální rozlišení 128 paprsků, horizontální rozlišení 1024 paprsků, frekvence otáčení 10 Hz) je v jednom snímku $128 \cdot 1024 = 131\ 072$ bodů, tj. 1 310 720 bodů za sekundu. Jeden snímek s ohledem na frekvenci otáčení by bylo tedy nejlepší zpracovat do 100 ms.

LiDAR OS0 s 128 řádky může dosáhnout největšího počtu bodů při 2048 paprscích v horizontálním rozlišení a frekvenci otáčení 20 Hz, kdy vychází pro jeden snímek $128 \cdot 2048 = 262\ 144$ bodů, tj. 5 242 880 bodů za sekundu.

2.3 Současný stav problematiky

Po zavedení LiDARů do robotiky se začal řešit problém tvorby modelu prostředí. Protože první LiDARy pracovali pouze v jedné rovině, první modely prostředí byly pouze dvourozměrné. Jedním z nejjednodušších modelů prostředí je mřížka obsazenosti (z angl. *occupancy grid*), která rozděluje prostor na pravidelné části a u každé si drží informaci, zda je prostor volný, nebo obsazen objektem prostředí. [5] Již v této době začínají vznikat i dvourozměrné mapy, které se snaží aproximovat tvar prostředí geometrickými primitivy. [6–7]

S nástupem LiDARů pracujícími s rotujícími laserovými paprsky ve více rovinách dochází i k vytváření algoritmů pro 3D mapy. Začínají se objevovat 3D voxelové mapy, obdoba 2D mřížek [8], stejně tak se začínají objevovat přístupy, snažící se z LiDARových dat získat geometrické příznaky [9] nebo i geometrické modely [10].

V této práci se budeme zabývat problémem detekce stožáru vysokého napětí, což je objekt, který se skládá pouze z ocelových nosníků o šířce průměrně 10 cm. Tyto nosníky jsou pro LiDAR špatně detekovatelné, protože jsou velmi úzké. Například při použití rozlišení 2048 bodů Ouster LiDAR tyto body většinou maže, protože se z jeho pohledu jedná o šum v měření. Proto jsme pro naše snímání využívaly rozlišení 1024 bodů, kdy již senzor funguje spolehlivěji. Detekce a vytváření modelů stožárů vysokého napětí z LiDARových dat se nám nepovedlo najít v žádné dostupné literatuře. V této práci navážeme na současný stav problematiky a pokusíme se ho rozšířit o modelování úzkých konstrukcí špatně detekovatelných pro LiDARy.

Kapitola 3
Detekce stožáru vysokého napětí

V této kapitole se zaměříme na popis algoritmu pro detekci stožáru vysokého napětí. Nejprve uvedeme metodu RANSAC, která byla použita v několika částech algoritmu. Poté představíme samotný algoritmus detekce stožáru a popíšeme jeho části. Na závěr této kapitoly zhodnotíme silné a slabé stránky tohoto algoritmu.

3.1 RANSAC

RANdom SAmple Consensus (zkráceně RANSAC) [11] je obecný algoritmus pro odhad parametrů modelu, který uvedli Martin A. Fischler a Robert C. Bolles v roce 1981. Body patřící do modelu se nazývají *inliers*[1]. Algoritmus byl navržen tak, aby se vypořádal s velkým množstvím bodů nepatřících do modelu, tzv. *outliers*[2]. Proto, na rozdíl od ostatních přístupů té doby, algoritmus používá nejmenší možný počet bodů pro počáteční odhad parametrů modelu. A tímto způsobem se snaží nalézt nejlepší odhad parametrů.

Algoritmus RANSAC:

Vstup:
$M = \{x_i\}_{i=1}^{L}$
ϵ.....tolerance
τ.....práh (threshold) ukončovací podmínky
N.....maximální počet iterací

Výstup:
Parametry modelu s největším počtem *inliers*

1. Vyber náhodně nejmenší počet bodů pro výpočet parametrů modelu z množiny M.
2. Vypočti parametry modelu.
3. Spočti počet bodů sedících do modelu s danou tolerancí ϵ, tzv. *inliers*.
4. Pokud $\frac{\#inliers}{L} > \tau$, pak přepočítej parametry modelu pomocí všech *inliers* a skonči.
5. Jinak, opakuj krok 1–4 maximálně N-krát.

Tolerance ϵ určuje, jak daleko může být bod od modelu, aby byl ještě uznán jako *inlier*.

Ukončovací práh τ zvolíme podle odhadu počtu zastoupení *inliers* v množině M. Pokud algoritmus nalezne model s již s dostatečným počtem *inliers*, skončí. Tento vstupní parametr je volitelný a v případě nezvolení algoritmus ukončovací podmínku přeskakuje.

[1] ležící uvnitř
[2] ležící vně

Maximální počet iterací N je potřeba zvolit tak, aby pravděpodobnost p, že v alespoň jedné množině náhodně vybíraných bodů nebude žádný *outlier*, byla co největší.

Pro pevně dané p (např. $p = 0.99$), můžeme N zvolit následovně. Nechť i je pravděpodobnost, že náhodně vybraný bod je *inlier*. Potom $o = 1 - i$ je pravděpodobnost, že náhodně zvolený bod je *outlier*. A m je počet náhodně vybíraných bodů. Pak platí následující rovnost.

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

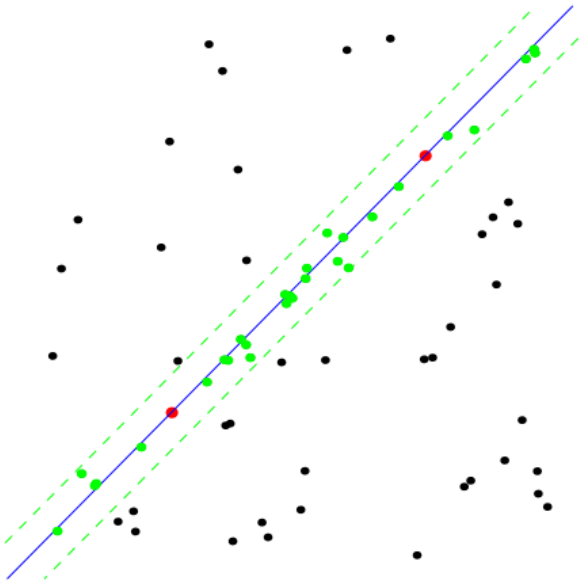
Kde $1 - p$ je pravděpodobnost, že všechny množiny obsahují alespoň jeden *outlier*. A $1-i^m$ je pravděpodobnost, že množina obsahuje alespoň jeden *outlier*. Z toho úpravou rovnice dostaneme N.

$$\log(1 - p) = N \cdot \log(1 - i^m)$$

$$N = \frac{\log(1 - p)}{\log(1 - i^m)}$$

Pro dobrý odhad N bychom potřebovali znát počet zastoupení *inliers* a *outliers* v našich datech. Tudíž pokud tuto informaci nemáme a nejsme schopni udělat ani dobrý odhad. Nezbývá nám než N získat empiricky postupným zvyšováním dokud nedojdeme k uspokojivým výsledkům. Protože jak je vidět z rovnice (1), zvyšování N nám zvětšuje pravděpodobnost p. [12]

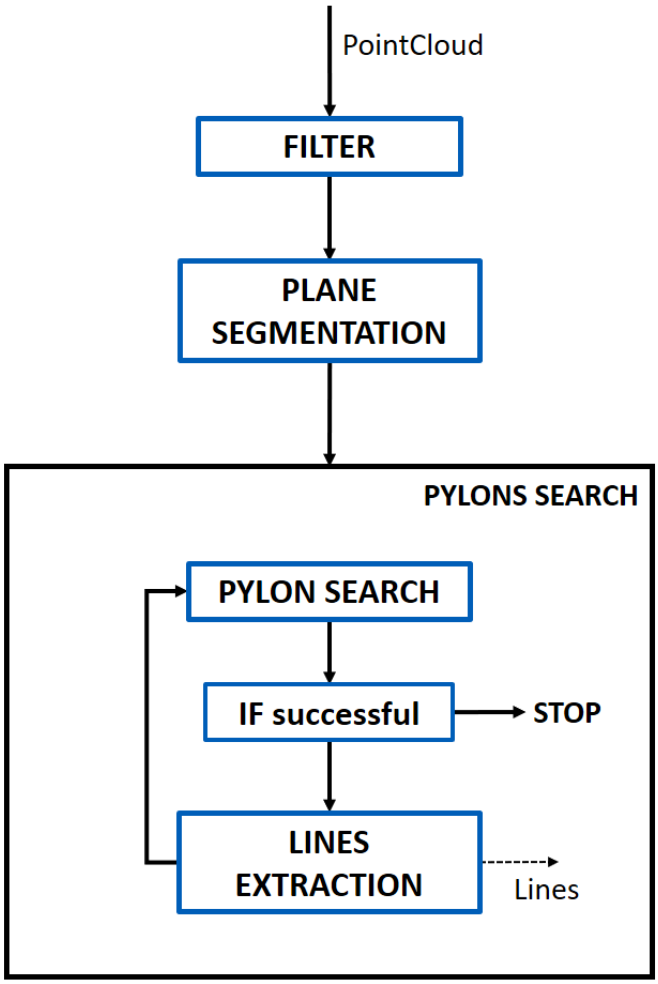
Na obrázku 3.1 můžeme vidět ukázku detekce přímky pomocí algoritmu RANSAC. Červené jsou dva body, kterými byl spočítán aktuální model přímky. Zelené přerušované přímky znázorňují toleranci ϵ. *Inliers* jsou zelené body. Černé body jsou *outliers*.



Obrázek 3.1. Ukázka detekce přímky pomocí algoritmu RANSAC [13]

3.2 Algoritmus detekce stožáru

Algoritmus dostává na vstupu snímek LiDARu neboli **mračno bodů (PointCloud)**, které postupně zpracovává. Výstup tvoří detekovaný stožár(y) ve formě množiny úseček. Tyto úsečky reprezentují konstrukci stožáru/ů. Průběh algoritmu je znázorněn na obrázku 3.2. Nyní popíšeme a okomentujeme jeho jednotlivé části.



Obrázek 3.2. Průběh algoritmu detekce stožáru vysokého napětí

3.2.1 Filtrace

Filter provede redukci počtu bodů. Dojde k odstranění bodů s hodnotami $(0, 0, 0)$, které přestavují paprsky, u nichž nebyl detekován odraz a pravděpodobně se v tomto směru nenachází žádný blízký objekt. Dále odstraní body jejíž vzdálenost od počátku je větší jak 30 m, z důvodu odstranění vzdálenějších objektů. Práh 30 metrů byl určen na základě předpokladu, že dron nebude létat ve větší vzdálenosti od stožáru a navíc stožár ve vzdálenosti 30 metrů již není laserovým paprskem detekovatelný. Také odstraníme body jejichž vzdálenost je menší jak 0.5 metrů, z důvodu odstranění detekovaného rámu dronu.

V dalších částech algoritmu je potřeba předpoklad, aby dron byl ve vodorovné poloze. Pro případ, že by dron nebyl ve vodorovné poloze, pomocí dat z jednotky IMU (Inertial Measurement Unit), která díky gyroskopům zná náklon dronu, natočíme mračno bodů do potřebné polohy. Po skončení detekce stožáru natočíme získané úsečky zpátky do původní polohy, aby změna natočení nevadila navazující části, kvůli nepřesnostem senzorů jednotky IMU. LiDAR i řídící jednotka dronu disponují jednotkou IMU.

3.2.2 Detekce země

Plane segmentation (detekce plochy) slouží k odstranění bodů představující zemi. K tomu se využívá algoritmus RANSAC pro detekci modelu roviny, která je kolmá k *ose z* (tj. vektor $(0, 0, 1)$) s maximální odchylkou 0.1 rad (5.73°). Kolmost zde uvažujeme, protože v našich datech se stožár vyskytoval na rovině. Také musí platit, aby dron byl ve vodorovné poloze. Detekovanou plochu odebíráme, pokud počet *inliers* je větší jak 500 bodů, aby nedocházelo k odstranění bodů stožáru v situaci, kdy dron je vysoko a země není zastoupena v mračnu bodů.

Hodnoty parametrů RANSAC jsou následující:

- $N = 100\ 000$
- $\epsilon = 0.5$ m

Parametry byli voleny empiricky tak, aby došlo k odstranění největšího počtu bodů země a zároveň nedocházelo k přílišnému uříznutí spodní části stožáru. Protože nejsme schopni odhadnout zastoupení počtu *inliers* v datech, byl maximální počet iterací zvolen na hodnotu, která byla dostatečná. A hodnota τ zvolena nebyla.

Předchozí dvě procedury jsou důležité kvůli redukci bodů, které by následující části algoritmu překáželi. Ať už z hlediska detekce stožáru nebo zvýšení časové náročnosti.

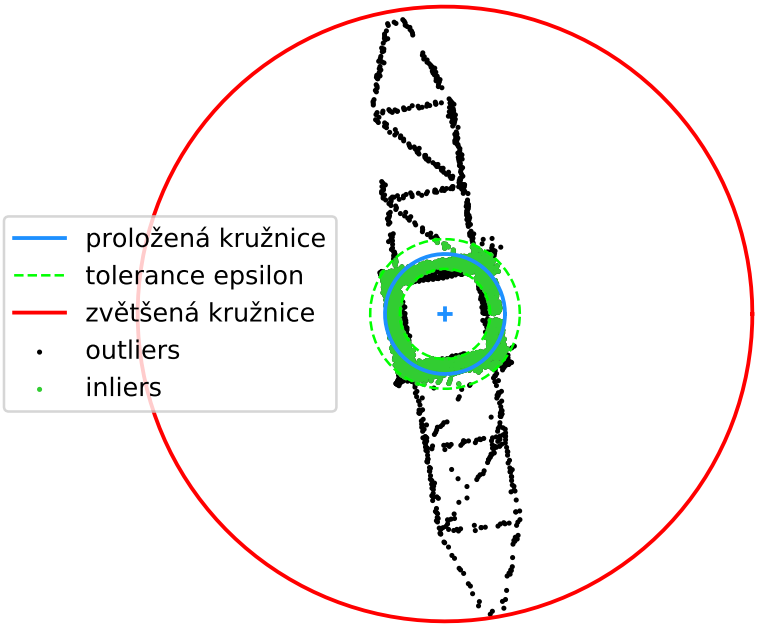
Po redukci počtu bodů přichází část pojmenovaná **Pylons search**, kde dochází k samotné detekci stožáru/ů a extrakci do úseček. Tato část zahrnuje dvě procedury.

3.2.3 Detekce stožáru

Pylon search (detekce stožáru) provádí najití části v datech, kde se potenciálně vyskytuje stožár. K tomu je použit algoritmus RANSAC pro detekci 2D kružnice. Uvažování ve 2D způsobí zanedbání třetí souřadnice. To si můžeme představit jako promítnutí všech bodů do roviny *xy*. Tím se nám konstrukce promítne do čtverce/obdélníku s mnoha body. A protože okolních bodů není mnoho, dochází k proložení kružnice. Ukázku detekce můžeme vidět na obrázku 3.3.

RANSAC parametry:

- $N = 100\ 000$
- $\epsilon = 0.3$ m
- poloměr kružnice musí být mezi 0.5 a 10 metry



Obrázek 3.3. Ukázka detekce kružnice algoritmem RANSAC

Hodnoty byli zvoleny empiricky. Dolní limit poloměru zajišťuje filtraci nesmyslně malých kružnic, horní naopak příliš velkých. Maximální velikost poloměru byla odhadnuta na 10 metrů.

Pokud nedojde k detekování žádné kružnice algoritmus **končí**.

Po nalezení kružnice s poloměrem r dojde k oříznutí okolí zvětšené kružnice s poloměrem $R = r + 5$ [m]. A poté tyto body pokračují do procedury pojmenované **Lines extraction**. V původním mračnu bodů se odebere vnitřek zvětšené kružnice s poloměrem R. Zvětšování kružnice se provádí z důvodu zahrnutí vybočujících částí stožáru.

Díky této proceduře dojde k snížení počtu bodů a tím usnadní následující části detekci přímek.

3.2.4 Extrakce úseček

Lines extraction (extrakce úseček) detekuje přímky a převádí je na úsečky, které reprezentují konstrukci stožáru.

Procedura postupuje následovně:

```
1  úsečky ← {}
2  while |úsečky| < MAX_LINES:
3      přímka ← RANSAC detekce přímky
4      if |inliers| < 10:
5          break
6      odebrání inliers z mračna bodů
7      úsečka ← převod na úsečku ← přímka
8      if délka úsečky < 0.5 [m]:
9          úsečka nepřijata
10         continue
11     úsečky ← úsečky ∪ úsečka
12
13 if |úsečky| < 2:
14     oblast není uznána jako stožár a úsečky nejsou předány na výstup
15 else
16     úsečky se předají na výstup
```

Poté se pokračuje částí **Pylon search**.

Parametry algoritmu RANSAC s modelem přímky:

- $N = 100$
- $\epsilon = 0.2$ m

Tolerance ϵ zvolená na 20 cm odpovídá odhadu šířky konstrukčních částí stožáru. Dáváme podmínku na délku úsečky 0.5 metru k nepřijímání úseček, které by nemohli tvořit stožár. Dále bylo zvoleno `MAX_LINES = 20` tak, aby detekce úseček netrvala příliš dlouho a přesto byl získán dostatečný počet úseček pro rekonstrukci stožáru.

Během postupu procedury dochází k převodu z přímky na úsečku. Přímka je reprezentovaná jejími *inliers* a vzorcem

$$p + t \cdot d$$

kde $p \in \mathbb{R}^3$ je bod ležící na přímce, $d \in \mathbb{R}^3$ je směrový vektor a $t \in \mathbb{R}$ je parametr. Pro převod na úsečku je potřeba nalézt krajní body z *inliers* kolmě promítnutých na přímku. Nalezení krajních bodů úsečky se provede následujícím způsobem.

$$T = \{t \mid t = \frac{\langle d \mid x - p \rangle}{\langle d \mid d \rangle}, x \in inliers\} \tag{2}$$

$$start = p + \min T \cdot d$$
$$end = p + \max T \cdot d$$

Kde T je množina parametrů přímky bodů z *inliers* kolmě promítnutých na přímku, *start* a *end* jsou krajní body úsečky.

Vzorec z (2) se získá takto. Označme x_p jako kolmý průmět bodu x na přímku. Potom platí

$$\begin{aligned} p + t \cdot d &= x_p \\ t \cdot d &= x_p - p \\ ||t \cdot d|| &= ||x_p - p|| \\ |t| \cdot ||d|| &= ||x_p - p|| \\ |t| &= \frac{||x_p - p||}{||d||} \end{aligned}$$

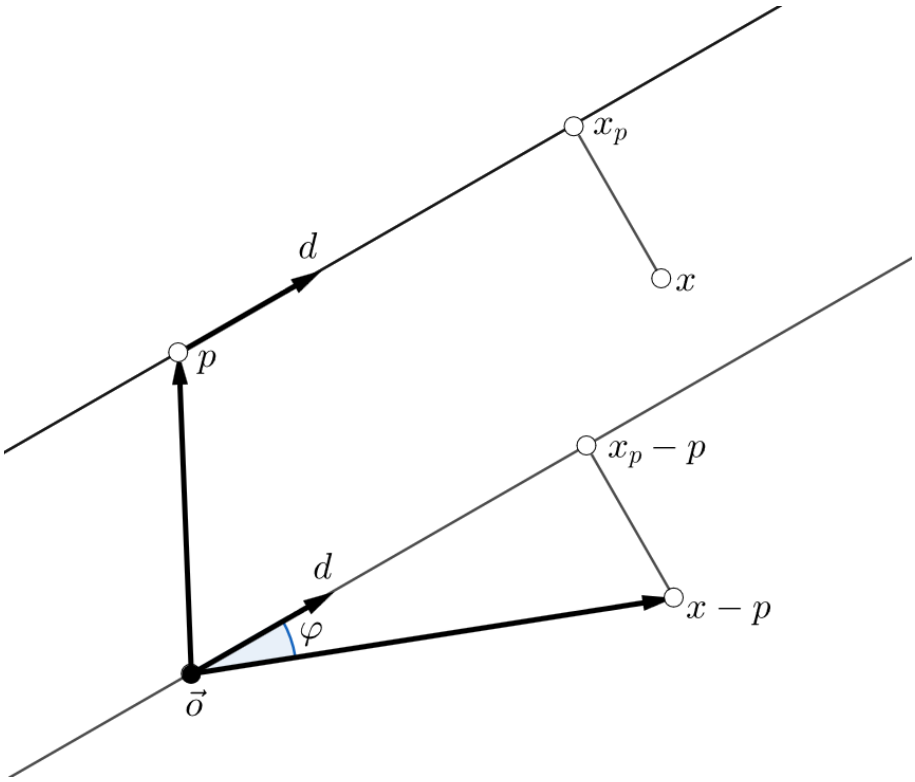
Dále platí

$$\begin{aligned} \langle d \mid x - p \rangle &= ||d|| \cdot ||x - p|| \cdot \cos\varphi \\ |\langle d \mid x - p \rangle| &= ||d|| \cdot ||x - p|| \cdot \frac{||x_p - p||}{||x - p||} \\ ||x_p - p|| &= \frac{|\langle d \mid x - p \rangle|}{||d||} \end{aligned}$$

Po dosazení dostáváme

$$t = \frac{\frac{\langle d|x-p\rangle}{||d||}}{||d||} = \frac{\langle d \mid x - p \rangle}{\langle d \mid d \rangle}$$

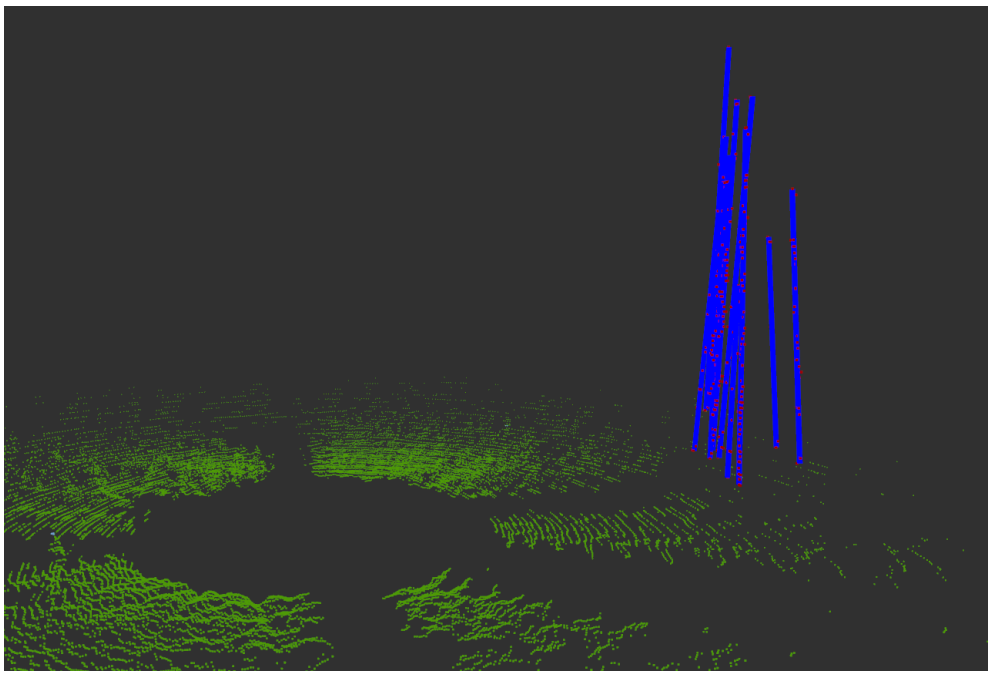
■



Obrázek 3.4. Doplňující obrázek k odvození vzorce z (2)

3.3 Zhodnocení

Na obrázku 3.5 můžeme vidět úspěšně detekovaný stožár vysokého napětí. Zelené body představují detekovanou zemi. Modré úsečky reprezentují konstrukci stožáru a červené body jsou *inliers* jednotlivých úseček. Můžeme také vidět, že LiDAR zachytil především přední část stožáru a zadní část je z dat hůře detekovatelná.

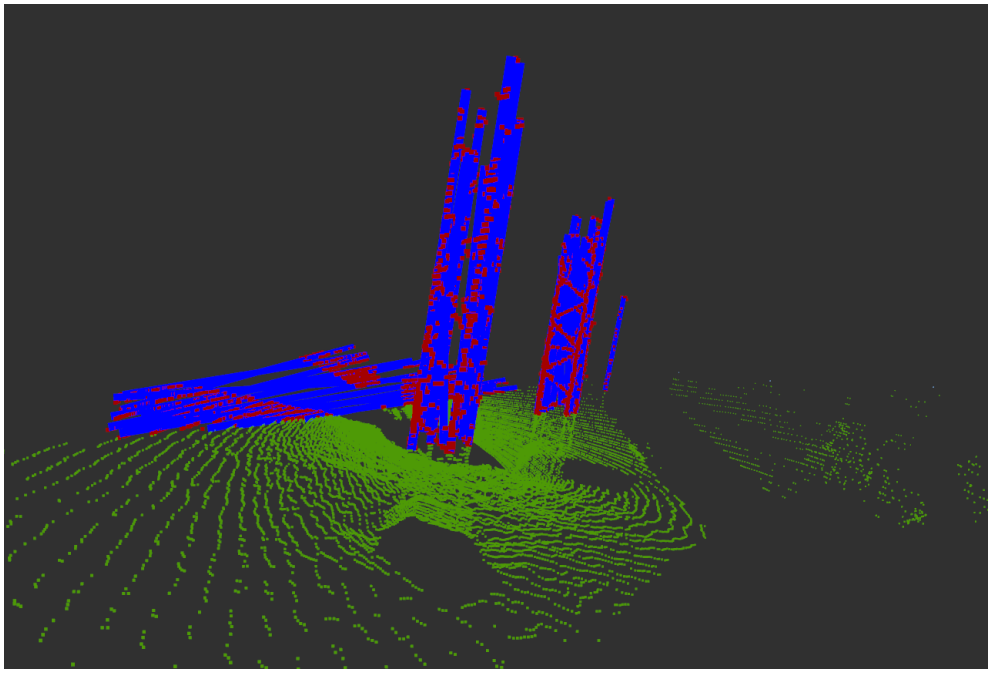


Obrázek 3.5. Ukázka detekce stožáru vysokého napětí

Na ukázce z jiných dat, obrázek 3.6, vidíme správně detekované oba sloupy stožáru a také chybně detekované objekty.

První z nich je pouliční lampa, která je vidět napravo od sloupů. Detekce lampy nám vadit nemusí, protože je to statický objekt a při využití výstupu algoritmu pro lokalizaci dronu nám tato detekce nijak lokalizaci nenarušuje.

Druhý špatně detekovaný objekt, který je vidět nalevo od stožárů, je země. Chyba vznikla nakloněním dronu a tím nesplnění podmínky vodorovné polohy dronu. To způsobí špatné proložení plochy pro odstranění země. Také je vidět, že dojde k odstranění části stožáru. Náklon dronu může způsobit i chybnou detekci úseček stožáru, protože po náklonu není stožár rovnoběžný s *osou z*. Proto před těmito částmi algoritmu s využitím dat z jednotky IMU natáčíme mračno bodů do potřebné polohy. Dále se stožár musí nacházet na rovině, jinak by jsme museli odstranit podmínku kolmosti plochy k *ose z*.



Obrázek 3.6. Ukázka detekce stožáru vysokého napětí s chybně nalezenými úsečkami

Algoritmus zvládá obstojně detekovat konstrukční části stožáru. Problémem jsou okolní objekty tvořené velkým množstvím bodů, kterými jsou např. keře.

Možná úprava algoritmu by mohla být změna podmínek pro detekované přímky. Další možná úprava algoritmu by mohlo být použití jedné z vylepšených variant algoritmu RANSAC, např. MLESAC [14] nebo PROSAC [15]. Pro detekci úseček by se případně dala použít *Houghova transformace* [16], která má ale vyšší výpočetní nároky.

Kapitola 4
Lokalizace dronu

Po extrakci úseček ze snímků LiDARu můžeme řešit problém lokalizace dronu, tj. určení jeho polohy ve světě. Nejprve si uvedeme a popíšeme dva algoritmy pro nalezení rotace a translace mezi snímky. Odvodíme řešení příslušné optimalizační úlohy. Poté vysvětlíme jejich použití pro lokalizaci dronu a tvorbu mapy za pomoci výškového senzoru dronu, neboli rekonstrukci stožáru vysokého napětí.

4.1 Algoritmus Iterative Closest Line

Algoritmus *Iterative Closest Line* (zkráceně ICL) je upravený algoritmus *Iterative Closest Point* (zkráceně ICP) [17], který uvedli Paul J. Besl a Neil D. McKay v roce 1992. Úpravu pro úsečky a i pro trojúhelníky provedli Q. Li a J.G. Griffiths v článku z roku 2000 [18]. Ale protože jejich kriteriální funkce počítá s registrací stejně dlouhých úseček, není tento způsob pro nás vhodný a použili jsme jinou úpravu algoritmu.

Tento algoritmus řeší iterativně problém nalezení optimální rotace a translace mezi dvěma snímky tak, aby vzdálenost mezi spárovanými úsečkami byla minimální.

Postup algoritmu je následující:

```
ICL(target_lines, source_lines) {
    tmp_lines = source_lines
    R = Matrix3x3::Identity
    t = Vector3::Zero
    error = INFINITY
    iter  = 0
    while (iter < MAX_ITERATIONS) {
        // sekce 4.1.1
        pairs = make_pairs(target_lines, tmp_lines)
        // sekce 4.1.2
        R,t = rotation_translation(target_lines, source_lines, pairs)
        // sekce 4.1.3
        tmp_lines = transform_lines(source_lines, R, t)
        // sekce 4.1.4
        new_error = pairs_error(target_lines, tmp_lines, pairs)
        if (|new_error - error| < EPSILON) {
            error = new_error
            break
        }
        error = new_error
        iter += 1
    }
    return R,t,error
}
```

Ve funkci se používají dvě konstanty, které je potřeba nastavit.

- maximální počet iterací `MAX_ITERATION = 100`
- práh ukončovací podmínky `EPSILON = 1e-8`

Nyní popíšeme jednotlivé části algoritmu.

4.1.1 Párování

V prvním kroku iterace je potřeba provést párování mezi úsečky. V ICP algoritmu se body párují tak, že se ke každému zdrojovému bodu najde nejbližší cílový bod. V případě úseček aplikujeme stejný postup.

Vzdálenost mezi úsečky definujeme takto

$$\frac{||\hat{q}_i - p_i|| + ||\hat{q}'_i - p'_i||}{2}$$

kde p_i, resp. p'_i je počáteční, resp. koncový bod zdrojové úsečky, $\hat{q}_i = q_i + \frac{\langle d_i | p_i - q_i \rangle}{||d_i||^2} d_i$, $\hat{q}'_i = q_i + \frac{\langle d_i | p'_i - q_i \rangle}{||d_i||^2} d_i$ kde d_i je směrový vektor cílové úsečky, q_i, resp. q'_i je počáteční, resp. koncový bod cílové úsečky. Ke zdrojové úsečce vybereme cílovou úsečku, ke které má nejmenší vzdálenost.

V článku [18] definují vzdálenost mezi úsečky jako velikost plochy čtyřúhelníku vzniklého spojením krajních bodů úseček. Tato definice není pro nás vhodná, protože počítá se stejně dlouhými úsečkami. Zatímco naše definování pomocí kolmých projekcí krajních bodů úsečky na přímku danou druhou úsečkou jsme zvolili proto, že se lépe vypořádá se situací, kdy jsou úsečky různě dlouhé. Tento způsob je výhodný i v další části při hledání rotace a translace.

Získané páry pokračují do dalších částí algoritmu.

4.1.2 Najití rotace a translace

Rotace a translace se nalezne pomocí následující optimalizační úlohy:

$$\mathbf{R}^*, t^* = \arg\min_{\mathbf{R},t} \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} (||\hat{q}_i - \mathbf{R}p_i - t||^2 + ||\hat{q}'_i - \mathbf{R}p'_i - t||^2) \tag{1}$$

kde

- N je počet párů
- p_i, resp. p'_i je počáteční resp. koncový bod zdrojové úsečky
- $\hat{q}_i = q_i + \frac{\langle d_i | p_i - q_i \rangle}{||d_i||^2} d_i$, $\hat{q}'_i = q_i + \frac{\langle d_i | p'_i - q_i \rangle}{||d_i||^2} d_i$ kde d_i je směrový vektor cílové úsečky, q_i, resp. q'_i je počáteční, resp. koncový bod cílové úsečky
- w_i je součet počtu *inliers* zdrojové a cílové úsečky, $w = \sum_{i=1}^{N} w_i$

Poznámka 4.1. Body $\hat{q}_i$ a $\hat{q}'_i$ jsou kolmé průměty bodů p_i a p'_i na přímku danou cílovou úsečkou. Oproti ICL v [18] a ICP [17] zavádíme v kriteriální funkci (1) váhy, aby páry s velkým počtem *inliers* měli velkou váhu oproti těm s malým počtem *inliers* z důvodu např. chybně detekovaných úseček. Dále $\frac{1}{2}$ je v kriteriální funkci jenom pro lepší interpretaci výsledné chyby podobně jako v 4.1.1.

Nejprve vypočteme optimální translaci. Nechť

$$E(\mathbf{R}, t) = \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} (||\hat{q}_i - \mathbf{R}p_i - t||^2 + ||\hat{q}'_i - \mathbf{R}p'_i - t||^2)$$

pak

$$\begin{aligned}
\frac{\partial E}{\partial t} &= \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} [-2(\hat{q}_i - \mathbf{R}p_i - t) - 2(\hat{q}'_i - \mathbf{R}p'_i - t)] = 0 \\
\frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} [-2(\hat{q}_i - \mathbf{R}p_i) - 2(\hat{q}'_i - \mathbf{R}p'_i) + 4t] &= 0 \\
\frac{1}{2} \left\{ \sum_{i=1}^{N} \frac{w_i}{w} [-2(\hat{q}_i - \mathbf{R}p_i) - 2(\hat{q}'_i - \mathbf{R}p'_i)] + \sum_{i=1}^{N} 4 \frac{w_i}{w} t \right\} &= 0 \\
\frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} [-2(\hat{q}_i - \mathbf{R}p_i) - 2(\hat{q}'_i - \mathbf{R}p'_i)] + \frac{4t}{2} &= 0 \\
\frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} [2(\hat{q}_i - \mathbf{R}p_i) + 2(\hat{q}'_i - \mathbf{R}p'_i)] &= 2t \\
\sum_{i=1}^{N} \frac{w_i}{w} [\hat{q}_i - \mathbf{R}p_i + \hat{q}'_i - \mathbf{R}p'_i] &= 2t \\
\frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} [\hat{q}_i + \hat{q}'_i - \mathbf{R}(p_i + p'_i)] &= t \\
\sum_{i=1}^{N} \frac{w_i}{w} \frac{\hat{q}_i + \hat{q}'_i}{2} - \mathbf{R} \sum_{i=1}^{N} \frac{w_i}{w} \frac{p_i + p'_i}{2} &= t
\end{aligned}$$

tedy

$$\begin{aligned}
\bar{q} &= \sum_{i=1}^{N} \frac{w_i}{w} \frac{\hat{q}_i + \hat{q}'_i}{2} \\
\bar{p} &= \sum_{i=1}^{N} \frac{w_i}{w} \frac{p_i + p'_i}{2} \\
t &= \bar{q} - \mathbf{R}\bar{p}
\end{aligned}$$

Nyní takto vypočtenou translaci dosadíme zpátky do kriteriální funkce. Předtím ale zavedeme dva pojmy, které budeme později potřebovat.

Definice 4.2. Nechť $\mathbf{A} = (a_{ij}), \mathbf{B} = (b_{ij}) \in \mathbb{R}^{n\times m}$. **Standardní skalární součin matic** je

$$\langle \mathbf{A}|\mathbf{B}\rangle = \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}b_{ij}$$

Tvrzení 4.3. Nechť $q = (q_i) \in \mathbb{R}^n$, $p = (p_i) \in \mathbb{R}^m$ a $\mathbf{A} \in \mathbb{R}^{n\times m}$. Pak platí

$$q^\top \mathbf{A} p = \langle \mathbf{A}|qp^\top\rangle$$

DŮKAZ:

$$\begin{aligned} q^\top \mathbf{A} p &= \sum_{i=1}^{n} q_i \sum_{j=1}^{m} a_{ij}p_j \\ &= \sum_{i=1}^{n}\sum_{j=1}^{m} a_{ij}(q_i p_j) \\ &= \langle \mathbf{A}|qp^\top\rangle \end{aligned}$$

■

Definice 4.4. Nechť $\mathbf{A} = (a_{ij}) \in \mathbb{R}^{n\times n}$. **Stopa** čtvercové matice je

$$\mathrm{tr}(\mathbf{A}) = a_{11} + a_{22} + \ldots + a_{nn}$$

$$\langle \mathbf{A}|\mathbf{B}\rangle = \mathrm{tr}(\mathbf{A}^\top\mathbf{B})$$

Tvrzení 4.5. Nechť $\mathbf{A}, \mathbf{B}, \mathbf{C}$ jsou matice příslušných rozměrů. Pak platí

$$\langle \mathbf{A}|\mathbf{BC}\rangle = \langle \mathbf{B}^\top\mathbf{A}|\mathbf{C}\rangle = \langle \mathbf{AC}^\top|\mathbf{B}\rangle$$

DŮKAZ:

$$\begin{aligned} \langle \mathbf{A}|\mathbf{BC}\rangle &= \mathrm{tr}(\mathbf{A}^\top\mathbf{BC}) = \mathrm{tr}((\mathbf{B}^\top\mathbf{A})^\top\mathbf{C}) = \langle \mathbf{B}^\top\mathbf{A}|\mathbf{C}\rangle \\ \langle \mathbf{A}|\mathbf{BC}\rangle &= \mathrm{tr}(\mathbf{A}^\top\mathbf{BC}) = \mathrm{tr}(\mathbf{CA}^\top\mathbf{B}) = \mathrm{tr}((\mathbf{AC}^\top)\mathbf{B}) = \langle \mathbf{AC}^\top|\mathbf{B}\rangle \end{aligned}$$

■

Tvrzení 4.6. Nechť $\mathbf{A} = (a_1\ a_2\ \cdots\ a_m) \in \mathbb{R}^{n\times m}$. Jestliže $\mathbf{A}^\top\mathbf{A} = \mathbf{I}$, kde $\mathbf{I}$ je jednotková matice, pak $|a_{ij}| \le 1\ \forall i, j$.
DŮKAZ:
Z předpokladu $a_j^\top a_j = 1\ \forall j = 1, \ldots, m$

$$a^\top a = \underbrace{a_1^2}_{\ge 0} + \ldots + \underbrace{a_n^2}_{\ge 0} = 1$$

proto platí $|a_{ij}| \le 1\ \forall i, j$

■

Dosadíme $t = \bar{q} - \mathbf{R}\bar{p}$ do (1)

$$\min_{\mathbf{R}} \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} (||\hat{q}_i - \mathbf{R}p_i - (\bar{q} - \mathbf{R}\bar{p})||^2 + ||\hat{q}'_i - \mathbf{R}p'_i - (\bar{q} - \mathbf{R}\bar{p})||^2)$$

$$\min_{\mathbf{R}} \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} (||(\hat{q}_i - \bar{q}) - \mathbf{R}(p_i - \bar{p})||^2 + ||(\hat{q}'_i - \bar{q}) - \mathbf{R}(p'_i - \bar{p})||^2)$$

Označme

$$\begin{aligned} \bar{q}_i &= \hat{q}_i - \bar{q} \\ \bar{q}'_i &= \hat{q}'_i - \bar{q} \\ \bar{p}_i &= p_i - \bar{p} \\ \bar{p}'_i &= p'_i - \bar{p} \end{aligned}$$

potom

$$\min_{\mathbf{R}} \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} (||\bar{q}_i - \mathbf{R}\bar{p}_i||^2 + ||\bar{q}'_i - \mathbf{R}\bar{p}'_i||^2)$$

$$\min_{\mathbf{R}} \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} [(\bar{q}_i - \mathbf{R}\bar{p}_i)^\top (\bar{q}_i - \mathbf{R}\bar{p}_i) + (\bar{q}'_i - \mathbf{R}\bar{p}'_i)^\top (\bar{q}'_i - \mathbf{R}\bar{p}'_i)]$$

$$\min_{\mathbf{R}} \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} [(\bar{q}_i^\top \bar{q}_i - \bar{q}_i^\top \mathbf{R}\bar{p}_i - \bar{p}_i^\top \mathbf{R}^\top \bar{q}_i + \bar{p}_i^\top \mathbf{R}^\top \mathbf{R}\bar{p}_i) + (\bar{q}_i'^\top \bar{q}'_i - \bar{q}_i'^\top \mathbf{R}\bar{p}'_i - \bar{p}_i'^\top \mathbf{R}^\top \bar{q}'_i + \bar{p}_i'^\top \mathbf{R}^\top \mathbf{R}\bar{p}'_i)]$$

Matice $\mathbf{R}$ je rotace, tedy ortogonální matice, proto $\mathbf{R}^\top \mathbf{R} = \mathbf{I}$, kde $\mathbf{I}$ je jednotková matice. Dále díky tvrzení 4.3 platí $\bar{q}_i^\top \mathbf{R}\bar{p}_i = \langle \mathbf{R} | \bar{q}_i \bar{p}_i^\top \rangle$.

Pak

$$\min_{\mathbf{R}} \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} [(\bar{q}_i^\top \bar{q}_i - 2\langle \mathbf{R} | \bar{q}_i \bar{p}_i^\top \rangle + \bar{p}_i^\top \bar{p}_i) + (\bar{q}_i'^\top \bar{q}'_i - 2\langle \mathbf{R} | \bar{q}'_i \bar{p}_i'^\top \rangle + \bar{p}_i'^\top \bar{p}'_i)]$$

$$\min_{\mathbf{R}} \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} (\bar{q}_i^\top \bar{q}_i + \bar{q}_i'^\top \bar{q}'_i + \bar{p}_i^\top \bar{p}_i + \bar{p}_i'^\top \bar{p}'_i) - \frac{2}{2} \sum_{i=1}^{N} \frac{w_i}{w} (\langle \mathbf{R} | \bar{q}_i \bar{p}_i^\top \rangle + \langle \mathbf{R} | \bar{q}'_i \bar{p}_i'^\top \rangle)$$

$$\min_{\mathbf{R}} \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} (\bar{q}_i^\top \bar{q}_i + \bar{q}_i'^\top \bar{q}'_i + \bar{p}_i^\top \bar{p}_i + \bar{p}_i'^\top \bar{p}'_i) - \left\langle \mathbf{R} \,\middle|\, \sum_{i=1}^{N} \frac{w_i}{w} (\bar{q}_i \bar{p}_i^\top + \bar{q}'_i \bar{p}_i'^\top) \right\rangle$$

Označíme

$$\mathbf{A} = \sum_{i=1}^{N} \frac{w_i}{w} (\bar{q}_i \bar{p}_i^\top + \bar{q}'_i \bar{p}_i'^\top)$$

Potom

$$\begin{aligned} \mathbf{R}^* &= \arg\min_{\mathbf{R}} \frac{1}{2} \sum_{i=1}^{N} \frac{w_i}{w} (\bar{q}_i^\top \bar{q}_i + \bar{q}_i'^\top \bar{q}'_i + \bar{p}_i^\top \bar{p}_i + \bar{p}_i'^\top \bar{p}'_i) - \langle \mathbf{R} | \mathbf{A} \rangle \\ &\Updownarrow \\ \mathbf{R}^* &= \arg\max_{\mathbf{R}} \langle \mathbf{R} | \mathbf{A} \rangle \end{aligned} \tag{2}$$

Provedeme SVD: $\mathbf{A} = \mathbf{USV}^\top$, pak díky tvrzení 4.5 platí

$$\langle \mathbf{R}|\mathbf{USV}^\top\rangle \overset{\text{T4.5}}{=} \langle \mathbf{U}^\top\mathbf{RV}|\mathbf{S}\rangle = \langle \mathbf{X}|\mathbf{S}\rangle = x_{11}s_1 + x_{22}s_2 + x_{33}s_3 \text{ kde } s_1 \geq s_2 \geq s_3 \geq 0$$

Dále platí

$$\mathbf{X}^\top\mathbf{X} = (\mathbf{U}^\top\mathbf{RV})^\top\mathbf{U}^\top\mathbf{RV} = \mathbf{V}^\top\mathbf{R}^\top\mathbf{U}\mathbf{U}^\top\mathbf{RV} = \mathbf{I}$$

Proto z tvrzení 4.6 platí $|x_{ij}| \leq 1\ \forall i,j$, tedy aby byla (2) maximimální, $x_{ii} = 1\ \forall i$, tj. $\mathbf{X} = \mathbf{I}$

$$\begin{aligned}\mathbf{X} &= \mathbf{I}\\ \mathbf{U}^\top\mathbf{RV} &= \mathbf{I}\\ \mathbf{R} &= \mathbf{UV}^\top\end{aligned}$$

V případě, že $\det(\mathbf{UV}^\top) = -1$, pak $\mathbf{UV}^\top$ je zrcadlení a rotaci $\mathbf{R}$ vypočteme takto

$$\mathbf{R} = \mathbf{UDV}^\top$$

kde $\mathbf{D} = \text{diag}(1, 1, -1)$. Pokud $s_3 = 0$, pak $\mathbf{D}$ neovlivní výslednou hodnotu, protože

$$\langle \mathbf{R}|\mathbf{A}\rangle = \langle \mathbf{UDV}^\top|\mathbf{USV}^\top\rangle = \langle \mathbf{U}^\top\mathbf{UDV}^\top\mathbf{V}|\mathbf{S}\rangle = \langle \mathbf{D}|\mathbf{S}\rangle = d_1s_1 + d_2s_2 + d_3s_3$$

Výsledné $\mathbf{R}^*$ a t^* vypadá tedy následovně

$$\begin{aligned}\mathbf{R}^* &= \mathbf{UDV}^\top \text{ kde } \mathbf{D} = \text{diag}(1, 1, \det(\mathbf{UV}^\top))\\ t^* &= \bar{q} - \mathbf{R}^*\bar{p}\end{aligned}$$

■

4.1.3 Transformace úseček

Po získání rotace a translace se provede transformace úseček tak, že se transformují krajní body úseček pomocí

$$\begin{bmatrix}\mathbf{R}^* & t^*\\ \vec{o}^\top & 1\end{bmatrix}\begin{bmatrix}p\\ 1\end{bmatrix}$$

kde p je krajní bod úsečky.

Existují dvě možnosti:

1) V každé iteraci se úsečky transformují a postupují do další iterace. Výsledná rotace a translace se akumuluje.

$$\begin{aligned}\mathbf{R}_{k+1} &= \mathbf{R}^*\mathbf{R}_k & \mathbf{R}_0 &= \mathbf{I}\\ t_{k+1} &= \mathbf{R}^*t_k + t^* & t_0 &= \vec{o}\end{aligned}$$

2) Transformují se původní úsečky. Chyba a nové párování se provádí z transformovaných úseček, ale nalezení rotace a translace se provádí s původními úsečkami. Tím se nemusí výsledná rotace a translace akumulovat.

V naší verzi používáme druhou možnost.

4.1.4 Výpočet chyby

Chyba se vypočte z kriteriální funkce (1) v sekci 4.1.2 mezi cílovými úsečkami a transformovanými úsečkami s párováním získaným na začátku iterace. Pokud se tato chyba dostatečně nezmění od předchozí, tedy $|e_k - e_{k-1}| < \epsilon$, pak algoritmus končí, jinak pokračuje další iterací.

4.2 Algoritmus Iterative Closest Point

Algoritmus ICP má stejný postup jako ICL algoritmus popsaný v sekci 4.1, jen se liší v některých částech algoritmu, protože místo s úsečkami pracuje s mračny bodů.

Při párovaní se ke každému zdrojovému bodu najde nejbližší cílový bod a pro najití optimální rotace a translace se řeší následující optimalizační úloha:

$$\mathbf{R}^*, t^* = \arg\min_{\mathbf{R},t} \frac{1}{N} \sum_{i=1}^{N} ||q_i - \mathbf{R}p_i - t||^2 \tag{3}$$

kde

- N je počet párů
- p_i jsou zdrojové body
- q_i jsou cílové body

Řešení se nalezne stejným postupem jako u ICL v sekci 4.1.2. Kriteriální funkce (3) se používá také pro výpočet chyby.

Abychom mohli ICP algoritmus použít na model s úsečkami, zvolili jsme přístup pro vytvoření mračen bodů pomocí vzorkování úseček s danou konstantou skoku `GAP = 0.1` [m]. Oproti kdybychom brali přímo *inliers*, získáváme přesnější aproximaci úseček. Nad takto získanými mračny bodů spouštíme algoritmus ICP.

Výhodou tohoto přístupu proti ICL může být přesnější reprezentace pomocí mračen bodů a tím někdy přesnější výpočet příslušných transformací.

Nevýhodou tohoto přístupu proti ICL je výpočetní náročnost, protože ICP musí počítat s mnohem větším množstvím bodů než ICL. Množství bodů lze snížit pomocí konstanty `GAP`, ale tím se sníží i přesnost reprezentace úsečky. Další nevýhodou může být situace, kdy se dva body z téže úsečky napárují na body ze dvou různých úseček.

4.3 ICL/ICP SLAM

Simultaneous Localization And Mapping (zkráceně SLAM) [19] je problém lokalizace a vytváření/aktualizování mapy robota/dronu v neznámém prostředí. Jedna z variant je za použití ICL/ICP algoritmu.

Mapa je pro nás model stožáru vysokého napětí, který je reprezentován množinou úseček. Na tuto množinu úseček budeme registrovat úsečky z nového snímku pomocí algoritmu ICL/ICP a následně přidávat tyto úsečky do modelu. Tomuto přístupu se říká *frame to map*. Naopak přístup *frame to frame* pracuje pouze s novým a posledním snímkem.

Kromě senzoru LiDARu využíváme při lokalizaci pro určení výšky i výškový senzor dronu, konkrétně *rangefinder*, tj. jeden laserový paprsek namířený dolů pro získání vzdálenosti od země. Pro správné určení výšky je tedy potřeba předpoklad relativně rovného povrchu kolem stožáru. Také by se měla získaná vzdálenost přepočítávat podle náklonu dronu, ale protože za letu se dron snaží držet ve vodorovné poloze, tyto drobné odchylky zanedbáváme.

Algoritmus ICL/ICP odhaduje polohu dronu v rovině xy a s pomocí výškoměru se dron lokalizuje v prostoru.

Postup vypadá následovně:

```
geometric_model {
    Set<Line> lines = {}
}
rotation = Matrix3x3::Identity
position = Vector3::Zero

SLAM(new_lines, height) {
    if (geometric_model.lines.size() == 0) {
        // inicializace
        position  = (0, 0, height)
        new_lines = transform_lines(new_lines, rotation, position)
        geometric_model.lines = new_lines
    } else {
        position.z = height
        new_lines  = transform_lines(new_lines, rotation, position)

        // sekce 4.3.1
        source_lines = select_new_lines(new_lines)
        target_lines = select_model_lines(geometric_model.lines)

        R,t,error  = ICL/ICP(target_lines, source_lines)
        if (error < TAU) {
            new_lines = transform_lines(new_lines, R, t)

            // sekce 4.3.2
            merging_lines(new_lines, geometric_model.lines)

            // sekce 4.3.3
            filter_lines(geometric_model.lines)

            rotation = R*rotation
            new_pos  = R*position + t
            position = (new_pos.x, new_pos.y, height)
        }
    }
}
```

V algoritmu se musí nastavit konstanta `TAU`, kdy je chyba registrace dostatečně dobrá pro pokračování v algoritmu. V našem případě `TAU = 1.0`.

V momentě, kdy dron poprvé vidí stožár inicializuje se SLAM, tudíž první pozice dronu je v počátku roviny xy a v aktuální výšce podle výškového senzoru. Počátek roviny xy je tedy určen prvním snímkem a rovina xy s $z = 0$ reprezentuje zemi.

Před spuštěním algoritmu ICL/ICP je potřeba nové úsečky transformovat za pomoci posledního odhadu rotace a pozice s aktuální výškou, aby byli správně umístěny v souřadnicích mapy.

4.3.1 Výběr úseček

Do algoritmu ICL/ICP se předává redukovaný počet úseček, z kterých se vypočítává rotace a translace. Díku tomu se nepočítá např. s špatně detekovanými úsečkami a navíc rychlost algoritmu nezávisí na počtu nových a modelových úseček, protože vždy počítá s daným maximálním počtem.

V případě nových úseček se vybere **K** s největším počtem *inliers*.

V případě modelových úseček se vybere **L** s největším počtem `num_merge` > 0 ze sekce 4.3.3, zbytek do **L** s největším počtem *inliers*.

V našem případě jsme zvolili **K** $=$ **L** $= 20$ s ohledem na kapitolu 3.

4.3.2 Přidání úseček do modelu

Každá nová úsečka se sloučí s nejbližší modelovou, pokud je blíž než **0.5 metru**, jinak se přidává do množiny úseček modelu, 0.5 metru bylo zvoleno s ohledem na velikost konstrukčních částí stožáru. Vzdálenost mezi úsečkami počítáme stejně jako v 4.1.1.

Proces sloučení probíhá takto:

1) Spočítáme dva body

$$p_s = \frac{\#inliers_1 \cdot p_{s1} + \#inliers_2 \cdot p_{s2}}{\#inliers_1 + \#inliers_2}$$
$$p_e = \frac{\#inliers_1 \cdot p_{e1} + \#inliers_2 \cdot p_{e2}}{\#inliers_1 + \#inliers_2}$$

   kde p_{s1}, p_{s2} jsou počáteční body 1. a 2. úsečky, p_{e1}, p_{e2} jsou koncové body 1. a 2. úsečky a $\#inliers_1$, $\#inliers_2$ jsou počty *inliers* jednotlivých úseček.
2) Body p_s a p_e nám určují přímku, na kterou kolmě promítneme krajní body úseček, z nich vybereme dva nejkrajnější body, které určují finální úsečku.
3) $inliers = inliers_1 \cup inliers_2$

4.3.3 Filtrování úseček

Z důvodu, že se do modelu mohou přidávat úsečky, které nereprezentují stožár, chceme tyto úsečky odebírat. Proto dochází k filtrování úseček.

Každá úsečka má svůj tzv. `line_time`, ten se zvýší o jedna vždy když dochází k přidávání úseček do modelu, a počet sloučení, kdy se úsečka sloučila s jinou úsečkou, tzv. `num_merge`.

```
if (line_time % PERIOD == 0 && num_merge < line_time/2) {
    // odeber úsečku z modelu
}
```

Konstanta `PERIOD` určuje, jak často se úsečka kontroluje. Při kontrole pokud úsečka nebyla dostatečně slučována, tj. minimálně polovina jednotek času doby od přidání do modelu, je odebrána. Nastavení `PERIOD` na 16 se během testování ukázalo jako rozumná volba.

Kapitola 5
Popis programu

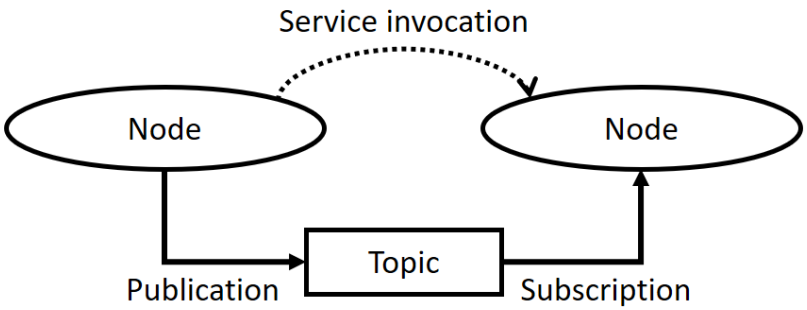
V této kapitole se zaměříme na popis programu. Nejdříve uvedeme systém ROS, v rámci kterého je program implementován. Dále popíšeme, jak program komunikuje s LiDARem a jakým způsobem se využívá knihovna PCL (PointCloud Library). Uvedeme struktury důležité pro výstup. A na závěr vysvětlíme kroky pro spuštění programu.

5.1 ROS

Robot Operating System (zkráceně ROS) [20] je open-source sada frameworků pro vývoj robotického softwaru. Poskytuje nástroje pro distribuovaně běžící procesy. Zejména posílání zpráv mezi procesy, dále hardwarovou abstrakci, ovladače pro zařízení a implementaci běžně používaných funkcionalit. Název se může zdát klamavý, protože se nejedná přímo o operační systém, ale vyplývá z poskytování zmíněných služeb, které běžný operační systém poskytuje.

Jednotlivé části ROSu [21]:

- **Nodes** (uzly) jsou v ROSu jednotlivé procesy/programy, které mezi sebou komunikují pomocí zpráv.
- **Master** je hlavní uzel, pomocí kterého probíhá komunikace mezi uzly. Slouží jako tabulka s registrací jmen a uzly se ho ptají, kam se má zpráva poslat. Bez tohoto uzlu by si nemohli uzly posílat zprávy nebo vyvolávat služby.
- **Messages** (zprávy) jsou definované struktury typů dat, které se posílají mezi uzly.
- **Topics** jsou takové komunikační kanály, přes které se posílají konkrétní typy zpráv. Uzly se na *topics* registrují jako tzv. **publishers**, ty co posílají, a **subscribers**, ty co poslouchají. Tímto konceptem vzniká anonymita komunikace mezi uzly. *Topics* představují asynchronní typ komunikace.
- **Services** (služby) představují synchronní typ komunikace ve stylu *request and reply*. Služba je definovaná dvojicí typů zpráv, pro *request* a pro *reply*. Na uzel, který službu provozuje, klient posílá *request* zprávy a čeká na *reply* zprávy.
- **Bags** (záznamy) slouží pro ukládání a opětovné přehrávaní záznamu zpráv.



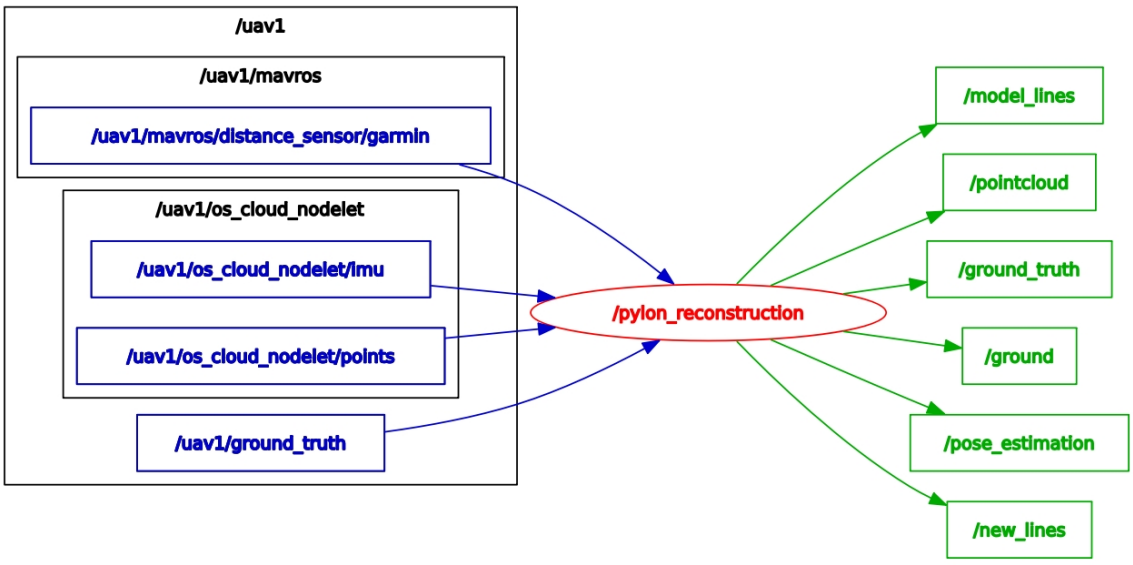
Obrázek 5.1. Ukázka komunikace mezi uzly [21]

5.2 Graf komunikace

Graf 5.2 znázorňuje komunikaci mezi uzly. Uzel `/pylon_reconstruction` je program, kde je implementován algoritmus pro rekonstrukci stožáru. Komunikaci s LiDARem zajišťuje uzel `/uav1/os_cloud_nodelet`, který provádí přepočet zmíněný v sekci 2.2 a následně tyto data předává do *topicu* `/uav31/os_cloud_nodelet/points`. Tento uzel také posílá data z IMU LiDARu do *topicu* `/uav31/os_cloud_nodelet/imu`.

Seznam *topiců*, ze kterých uzel `/pylon_reconstruction` přijímá zprávy:

- `/uav1/os_cloud_nodelet/points` – mračno bodů z LiDARu [`sensor_msgs/PointCloud2`]
- `/uav1/os_cloud_nodelet/imu` – data z IMU LiDARu [`sensor_msgs::Imu`]
- `/uav1/mavros/distance_sensor/garmin` – data z výškového senzoru dronu [`sensor_msgs::Range`]
- `/uav1/ground_truth` – *ground truth* z Gazebo simulátoru (pouze pro vyhodnocení lokalizace) [`nav_msgs::Odometry`]



Obrázek 5.2. Graf komunikace mezi uzly

Program používá funkce z knihovny `PCL`[1] (PointCloud Library). Po přijetí zprávy dojde ke konverzi na standardní typ knihovny PCL `pcl::PointCloud<pcl::PointXYZ>`, se kterým program dále pracuje podle algoritmu pro detekci stožáru popisovaném v sekci 3.2 a poté nad získanými úsečkami provádí algoritmus ICL/ICP SLAM popisovaný v sekci 4.3. Po dokončení výpočtu posílá uzel jednotlivé výsledky do následujících *topiců*:

- `/pointcloud` – *mračno bodů* po filtraci [`sensor_msgs/PointCloud2`]
- `/ground` – detekovaná a odstraněná země [`sensor_msgs/PointCloud2`]

[1] `https://pointclouds.org`

- `/model_lines` – úsečky tvořící rekonstruovaný stožár
  [`visualization_msgs/Marker`]
- `/new_lines` – úsečky z nového snímku získané algoritmem pro detekci stožáru
  [`visualization_msgs/Marker`]
- `/pose_estimation` – odhadovaná trajektorie letu získaná ICL/ICP SLAM algoritmem
  [`visualization_msgs/Marker`]
- `/ground_truth` – *ground truth* trajektorie letu transformovaná do souřadnic mapy/modelu (použitá pro vyhodnocení lokalizace)
  [`visualization_msgs/Marker`]

5.3 Výstup programu

Hlavním výstupem programu je geometrická reprezentace stožáru vysokého napětí, která je tvořena množinou úseček představující konstrukci stožáru. Úsečka je popsána následující třídou s jejími hlavními atributy.

```
class Line {
    public:
        PointCloudPtr point_cloud; // inliers
        Eigen::Vector3f start;
        Eigen::Vector3f end;
        Eigen::Vector3f direction; // unit vector
};
```

Tyto úsečky jsou seskupovány ve struktuře stožáru, která představuje geometrický model stožáru vysokého napětí.

```
struct Pylon {
    std::vector<Line> lines;
};
```

5.4 Spuštění programu

Výsledkem bakalářské práce je ROS balíček (package) s názvem `proj`, v rámci kterého je implementován program `pylon_reconstruction`.

Nejprve musíme zdrojové kódy zkompilovat pomocí příkazu:

```
catkin_make
```

Buď můžeme spustit Gazebo simulátor s nachystaným modelem stožáru vysokého napětí pomocí skriptu `start.sh`

```
gazebo_simulation/start.sh
```

Nebo pro použití *bagu* před spuštěním programu potřebujeme nejdříve spustit `roscore` (Master) a uzel `ouster_ros` pomocí příkazu:

```
roslaunch ouster_ros uav.launch replay:=True metadata:=$LIDAR_METADATA
```

Soubor `uav.launch` v balíčku `ouster_ros` je připraven v rámci `MRS UAV System`[2] pro účely práce s LiDARem od společnosti Ouster.

Pro správné spuštění je potřeba mít nastavené v proměnném prostředí `UAV_NAME`, `OUSTER_IP` a `OUSTER_UDP_DEST_IP`. V případě přehrávání záznamů zpráv od LiDARu v podobě *bags* se musí přidat argumenty `replay:=True` a `metadata` s cestou k *json* souboru s vlastnostmi LiDARu. Potom hodnoty v `OUSTER_IP` a `OUSTER_UDP_DEST_IP` nejsou podstatné. Proměnná `UAV_NAME` by měla být nastavena tak, aby korespondovala s názvem dronu v *bagu*.

Poté můžeme spustit uzel `pylon_reconstruction` pomocí:

```
rosrun proj pylon_reconstruction [uav_name]
```

Můžeme uvést argument `uav_name` pro specifikování tématu `/uav_name/os_cloud_nodelet/points`, z kterého bude uzel dostávat *mračna bodů*, a pro další témata ostatních potřebných senzorů. Výchozí název je `uav1`.

Pro použití *bagu* jen stačí spustit odpovídající *bag*:

```
rosbag play *.bag
```

Pro vizualizaci lze využít program `rviz` s před připraveným konfiguračním souborem `pylons.rviz`.

```
rosrun rviz rviz -d pylons.rviz
```

Pokud nechceme spouštět všechny potřebné ROS uzly. Můžeme použít program `pcd_pylons_search`, který načte *mračno bodů* z nějakého souboru `*.pcd`. A na něm následně pustí algoritmus detekce stožáru. Potřebujeme tedy spustit pouze `roscore` a případně `rviz` pro vizualizaci.

```
rosrun proj pcd_pylons_search *.pcd
```

[2] `https://github.com/ctu-mrs/mrs_uav_system`

Kapitola 6
Otestování algoritmu

Testování proběhlo v simulačním prostředí `Gazebo`[1] ve spojení s `MRS UAV System`[2] a na reálných datech. V simulaci jsou dostupné všechny potřebné senzory a tzv. *ground truth*, tj. přesná poloha dronu v prostředí, od které jsme měřili chybu lokalizace. Také jsme otestovali algoritmus i na různé typy stožárů vysokého napětí.

Testovali jsme oba přístupy algoritmu, ICL i ICP, na trajektorii, kterou byla spirála kolem středu stožáru s poloměrem 10, 15 a 20 metrů. Větší vzdálenosti jsme netestovali z důvodu špatné viditelnosti stožáru. V simulaci jsme neuvažovali dráty elektrického vedení za účelem zjednodušeného testovaní a demonstrace algoritmu. Testování bylo provedeno na počítači s procesorem AMD Ryzen 7 4800H.

Za letu jsme měřili následující parametry:

- **Chyba lokalizace** – vzdálenost mezi odhadovanou polohou a *ground truth* polohou transformovanou do souřadnic mapy/modelu
- **Počet snímků**
- **Počet bodů po filtraci**
- **Doba trvání**
  - *Pylons search* algoritmu
  - ICL/ICP SLAM algoritmu
  - celkem
- **Finální počet úseček tvořící model stožáru**

6.1 Výsledky simulace

Konkrétně jsme rekonstruovali algoritmem tři typy stožárů vysokého napětí prezentovaných v [22]. První typ jsme testovali i v závislosti na vzdálenosti od stožáru, ostatní typy jsme pouze zkoušeli na trajektorii ve vzdálenosti 10 metrů od středu stožáru.

6.1.1 Model stožáru typu 1

Na obrázku 6.1 můžeme vidět model stožáru vysokého napětí typu 1. V tabulce 6.1 nalezneme průměrný počet zpracovávaných bodů a k nim rychlosti jednotlivých částí algoritmu. Všimněme si, že ICL algoritmus je několikrát rychlejší proti ICP algoritmu z důvodu počítání s menším množstvím bodů. Dále vidíme narůst počtu bodů ve vzdálenosti 15 metrů od stožáru proti 10 metrům, přestože by toto číslo mělo klesnou. To je způsobeno zvolenou trajektorií, která končí v menší výšce, jak vidíme na obrázcích 6.2.

V tabulce 6.2 je vidět, že ICL algoritmus má nepatrně menší průměrnou chybu lokalizace. Počet slučovaných snímků je několikrát vyšší než u ICP algoritmu, kvůli rozdílným rychlostem obou algoritmů. Dále finální počty úseček jsou dle očekávání srovnatelné. Na grafech 6.3 vidíme průběhy lokalizačních odchylek. Můžeme si všimnout periodicity způsobené kruhovou dráhou letu kolem stožáru.

[1] `https://gazebosim.org`
[2] `https://ctu-mrs.github.io/docs/simulation/gazebo/gazebo.html`

Jak je vidět rostoucí vzdálenost od stožáru způsobuje horší schopnosti detekce a lokalizace, kvůli klesajícímu počtu bodů. Ve vzdálenosti 20 metrů od středu stožáru ICP SLAM nedoletěl danou trajektorii, kvůli chybě, která po příliš dlouhou dobu nebyla dostatečně dobrá pro určení polohy a algoritmus nebyl schopen dále pokračovat.

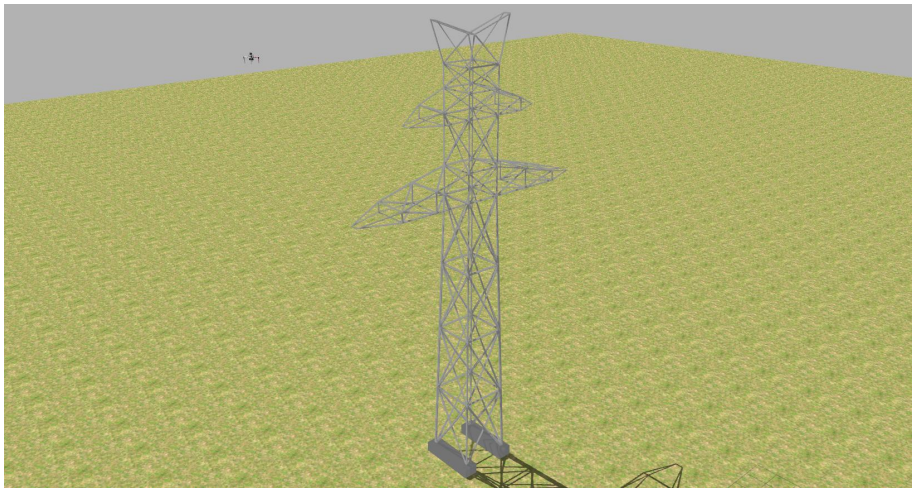
Celkově obě metody dávají model, který uspokojivě aproximuje stožár vysokého napětí a mají přijatelnou průměrnou chybu lokalizace.

			průměrná doba trvání [ms]		
		průměrný počet bodů	*Pylons search*	SLAM	celkem
10m	ICL	30 855	81.82	**217.64**	301.18
	ICP	30 404	79.34	936.02	1 011.26
15m	ICL	43 688	62.52	**209.51**	273.59
	ICP	38 126	60.46	1 038.86	1 095.60
20m	ICL	20 456	50.96	**290.70**	343.52
	ICP	19 431	41.19	874.93	914.73

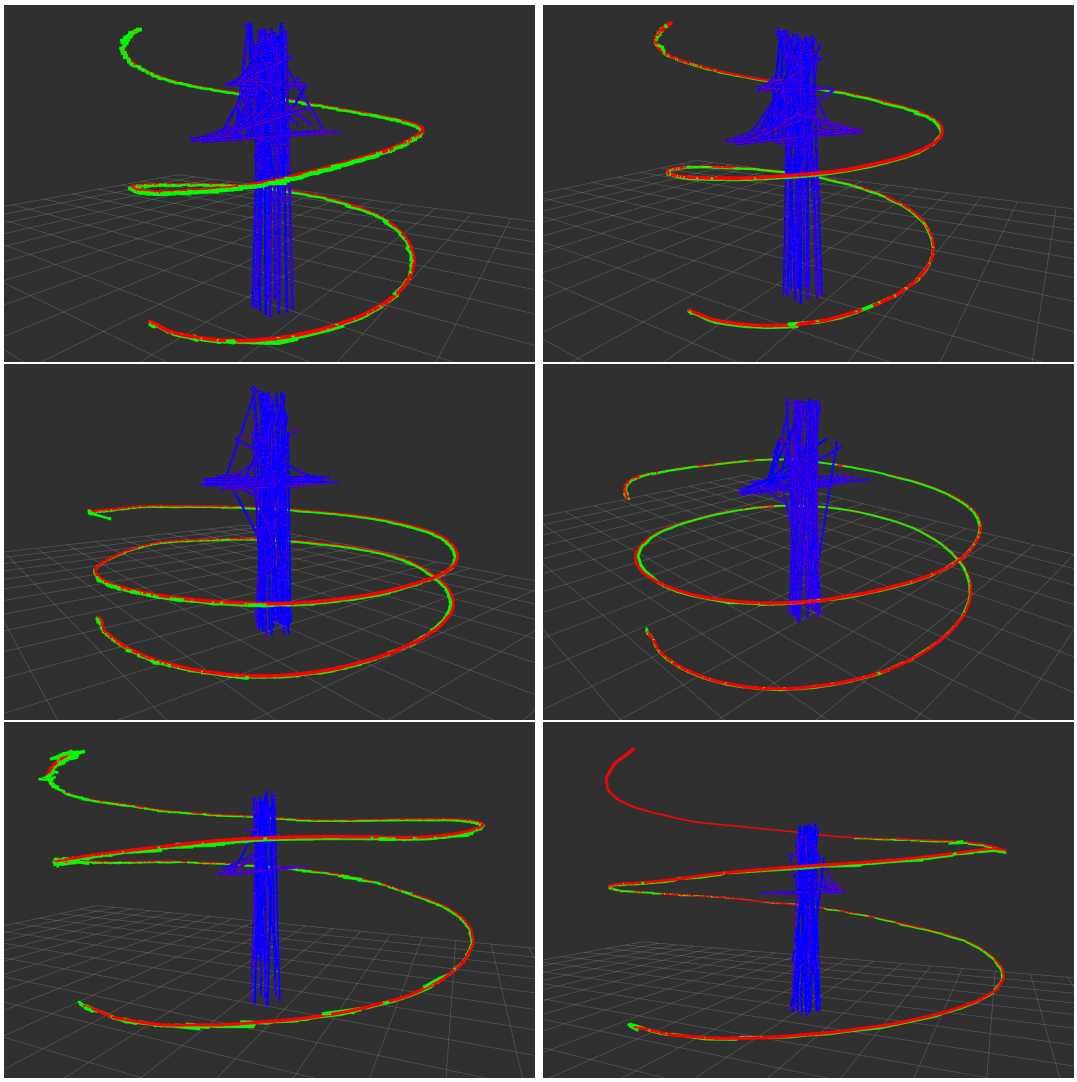
Tabulka 6.1. Průměrné doby trvání algoritmů, stožár typu 1

		průměrná chyba lokalizace [m]	počet snímků	počet úseček
10m	ICL	**0.3000**	486	57
	ICP	0.3182	146	70
15m	ICL	**0.3398**	779	61
	ICP	0.4306	194	64
20m	ICL	**0.5679**	798	32
	ICP	0.6084	211	46

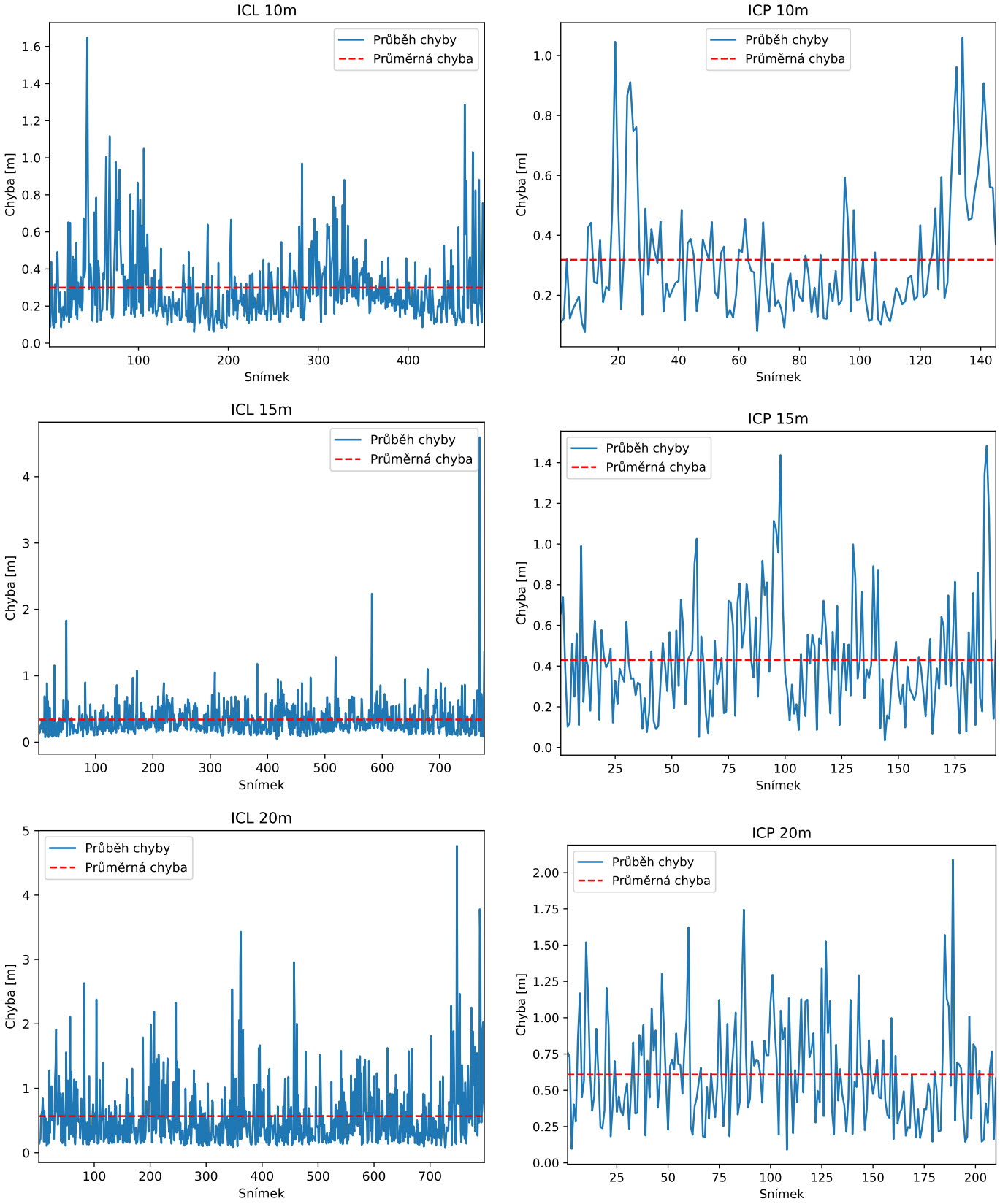
Tabulka 6.2. Výsledky lokalizace, stožár typu 1



Obrázek 6.1. Model stožáru typu 1 v Gazebo simulátoru



Obrázek 6.2. Výsledky simulace ICL/ICP SLAM stožáru typu 1 (levá/pravá strana) ve vzdálenosti od středu stožáru 10, 15 a 20 metrů (seshora dolů). Zelená ukazuje odhadovanou trajektorii letu, červená představuje *ground truth* trajektorii letu.



Obrázek 6.3. Chyby lokalizace ICL/ICP SLAM, stožár typu 1

6.1.2 Model stožáru typu 2 a 3

Modely stožárů typu 2 a 3 vidíme na obrázcích 6.4 a 6.7. Naměřené hodnoty nalezneme v tabulkách 6.3 a 6.4. Vytvořené modely stožárů, odhadovaná poloha a průběh chyby lokalizace jsou vidět na obrázcích 6.5, 6.6, 6.8 a 6.9.

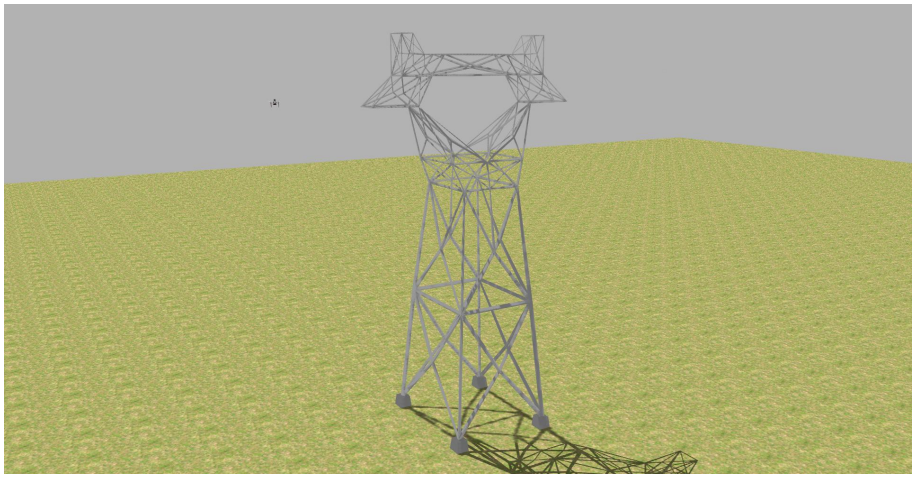
ICL je dle očekávání opět rychlejší než ICP, ale průměrná chyba vyšla pro tyto 2 typy stožárů menší ve prospěch ICP algoritmu. Znovu si můžeme všimnout periodicity v průběhu chyby lokalizace. U stožáru typu 3 došlo u ICL algoritmu na krátkou dobu k většímu vychýlení. Výsledné modely stožárů uspokojivě aproximují původní model.

			průměrná doba trvání [ms]		
		průměrný počet bodů	*Pylons search*	SLAM	celkem
typ 2	ICL	37 747	112.84	**150.46**	262.66
	ICP	40 716	107.81	623.86	697.78
typ 3	ICL	39 151	86.27	**285.48**	304.43
	ICP	39 677	89.10	931.83	833.87

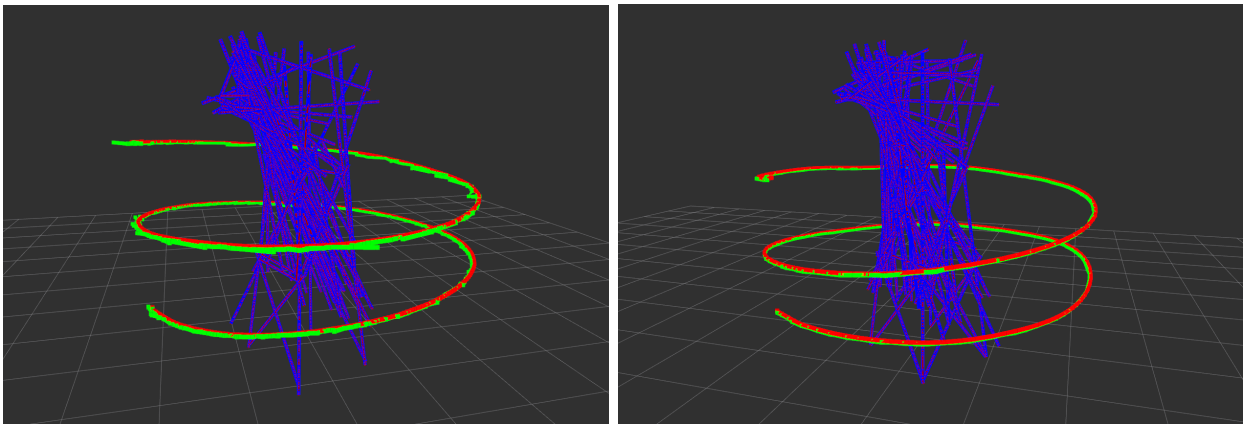
Tabulka 6.3. Průměrné doby trvání algoritmů, stožár typu 2 a 3

		průměrná chyba lokalizace [m]	počet snímků	počet úseček
typ 2	ICL	0.3462	535	71
	ICP	**0.1930**	200	95
typ 3	ICL	0.3717	354	47
	ICP	**0.2482**	143	50

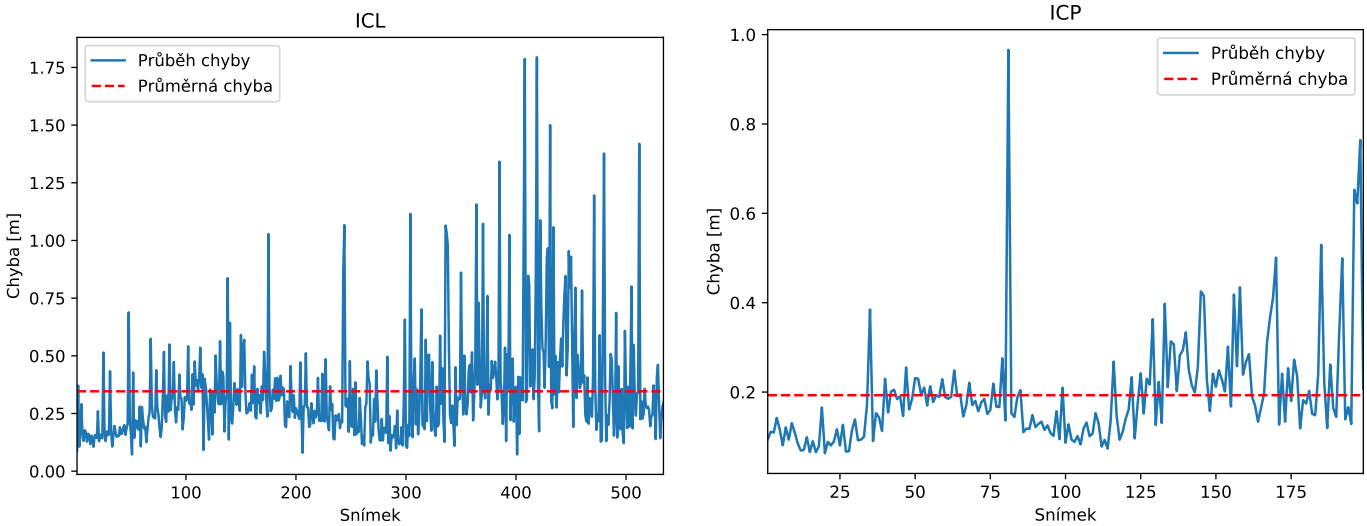
Tabulka 6.4. Výsledky lokalizace, stožár typu 2 a 3



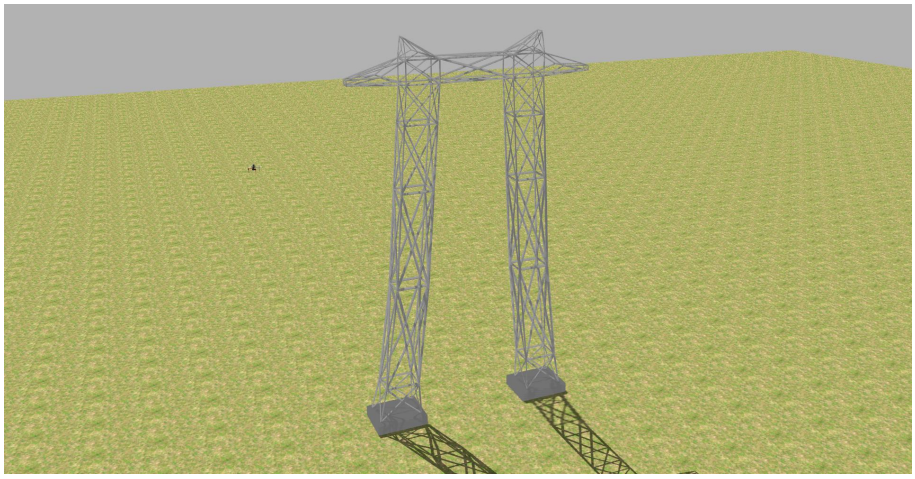
Obrázek 6.4. Model stožáru typu 2 v Gazebo simulátoru



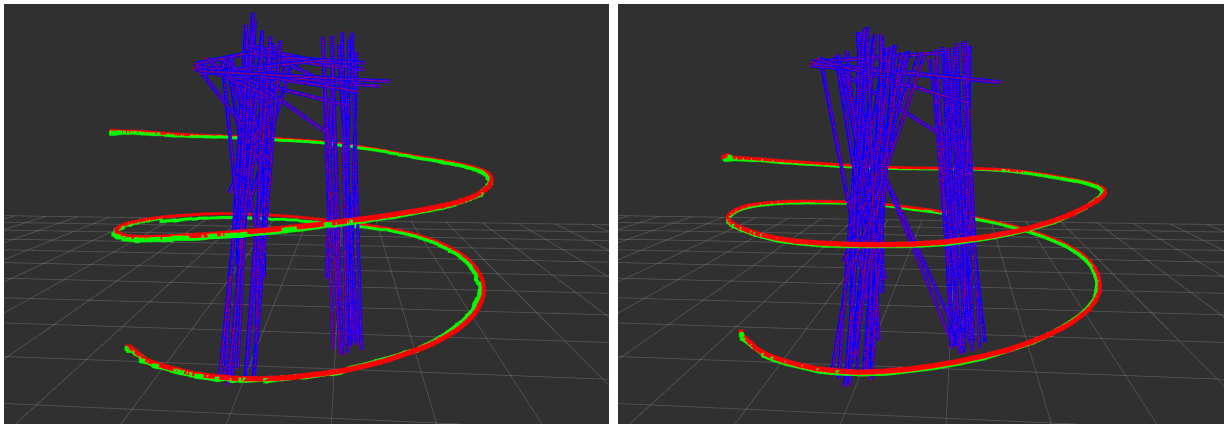
Obrázek 6.5. Výsledky simulace ICL/ICP SLAM stožáru typu 2 (levá/pravá strana) ve vzdálenosti od středu stožáru 10 metrů. Zelená ukazuje odhadovanou trajektorii letu, červená představuje *ground truth* trajektorii letu.



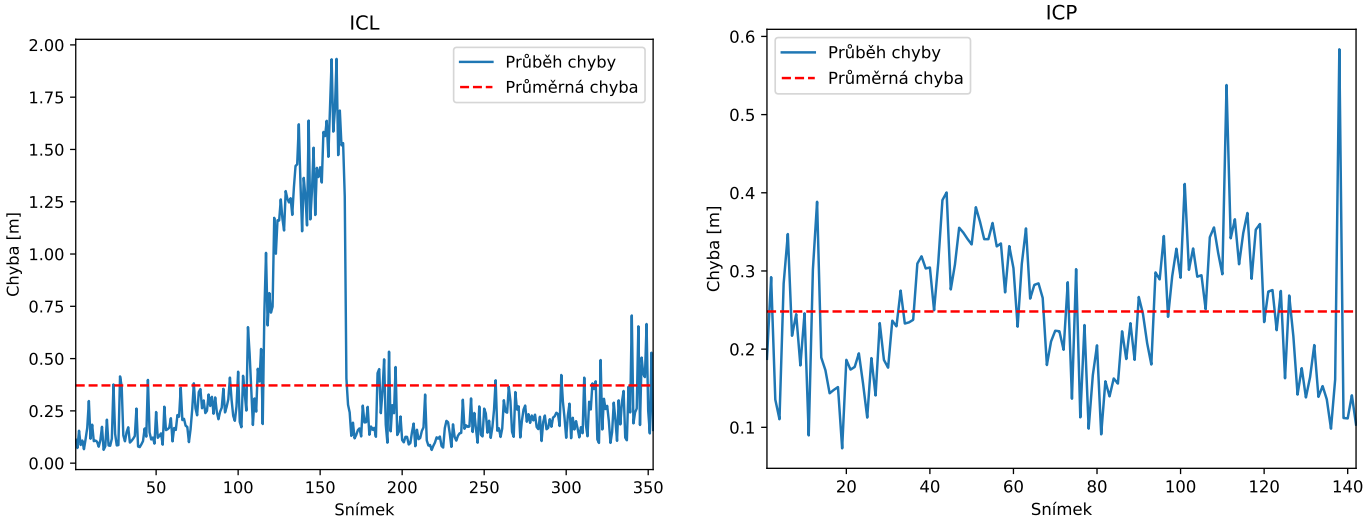
Obrázek 6.6. Chyby lokalizace ICL/ICP SLAM, stožár typu 2, vzdálenost 10 metrů od stožáru



Obrázek 6.7. Model stožáru typu 3 v Gazebo simulátoru



Obrázek 6.8. Výsledky simulace ICL/ICP SLAM stožáru typu 3 (levá/pravá strana) ve vzdálenosti od středu stožáru 10 metrů. Zelená ukazuje odhadovanou trajektorii letu, červená představuje *ground truth* trajektorii letu.

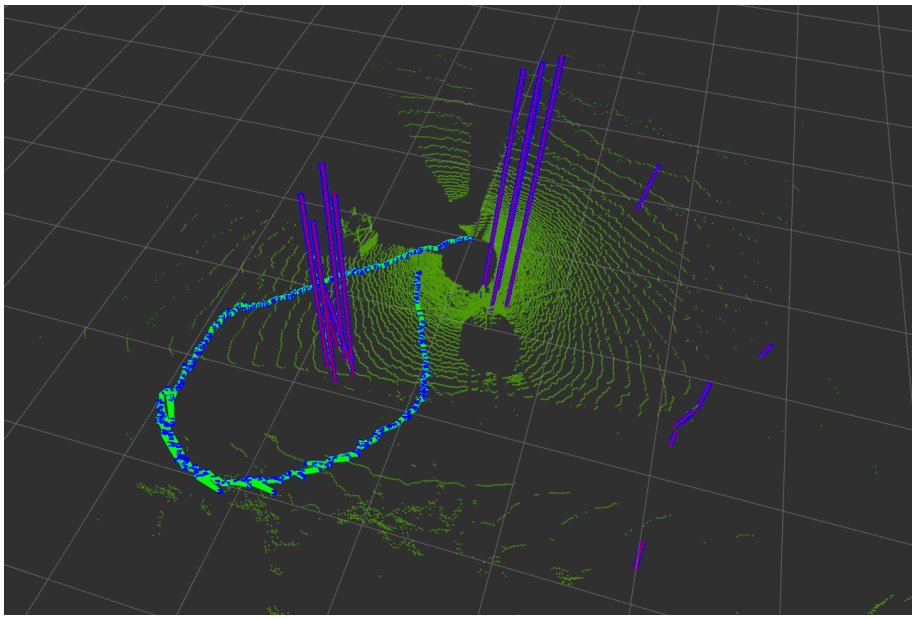


Obrázek 6.9. Chyby lokalizace ICL/ICP SLAM, stožár typu 3, vzdálenost 10 metrů od stožáru

6.2 Výsledky na reálných datech

Na reálných datech není možné změřit chybu lokalizace, protože nemáme skutečnou polohu dronu, proto jsme zobrazili pouze odhadovanou trajektorii získanou z algoritmu a výsledný model stožáru. Také jsme neměli k dispozici výškový senzor dronu, lokalizace se tedy prováděla pouze ve 2D. Uvádíme zde pouze testování varianty s ICL algoritmem, protože ICP algoritmus na těchto datech nebyl úspěšný a nepodařilo se mu dosáhnout dostatečně malé chyby.

Na obrázku 6.10 vidíme, že stožár byl tvořen dvěma sloupy. Vizuálně můžeme posoudit průběh lokalizace, který vypadá plynule až na část vlevo, kde zadní sloup je zastíněn předním a je tedy špatně detekovatelný. V tabulce 6.5 můžeme vidět výsledky měření, které jsou srovnatelné s výsledky simulace.



Obrázek 6.10. Výsledek algoritmu na reálných datech. Zeleno-modrá ukazuje odhadovanou trajektorii letu. Zelené body jsou odebraná země s posledního snímku.

			průměrná doba trvání [ms]		
		průměrný počet bodů	*Pylons search*	SLAM	celkem
reálná data	ICL	42 986	71.22	347.79	418.61

Tabulka 6.5. Průměrné doby trvání algoritmu, reálná data

Kapitola 7
Závěr

V rámci bakalářské práce byl vytvořen algoritmus pro vytváření geometrického modelu stožáru vysokého napětí. Byla zvolena vhodná geometrická reprezentace v podobě úseček představující konstrukční části stožáru. Nejprve byl navrhnut algoritmus pro detekci stožáru z LiDARových dat pomocí algoritmu RANSAC a extrakci úseček. Dále bylo potřeba vyřešit problém nalezení optimální rotace a translace mezi dvěma snímky LiDARu. Představili jsme dva způsoby řešení, algoritmus ICL (Iterative Closest Line) a ICP (Iterative Closest Point), které pracují se získanými úsečkami. Jeden z těchto algoritmů se dále použil k rekonstrukci stožáru za pomoci algoritmu SLAM (Simultaneous Localization And Mapping), který se používá pro tvorbu/aktualizování mapy, v našem případě mapa je geometrický model stožáru. Algoritmus byl implementován v rámci systému ROS (Robot Operating System). Na závěr jsme funkčnost algoritmu otestovali v Gazebo simulátoru a na reálných datech.

Testování ukázalo, že algoritmus podává uspokojivé výsledky srovnatelné v obou verzích algoritmu až na rychlost, kde ICL algoritmus je několikrát rychlejší než ICP. Výsledkem je geometrický model stožáru aproximující původní model.

Na práci by se dalo navázat spojením více senzorů dohromady, tzv. fúzování, pro získání přesnější lokalizace. Dále by se geometrický model stožáru mohl použít při plánování trajektorie letu dronu v okolí sloupu vysokého napětí v průběhu jakéhokoli prováděného úkonu.

Závěrem můžeme říci, že cíle bakalářské práce byli splněny.

Literatura

[1] Angus Pacala. *How Multi-Beam Flash Lidar Works*. online. 8. 11. 2018. [citováno 4. 1. 2024].
https://ouster.com/insights/blog/how-multi-beam-flash-lidar-works.

[2] Ouster Inc. *Společnost Ouster*. online. 2024. [citováno 4.1.2024].
https://ouster.com.

[3] Ouster Inc. *Ouster Product Family*. online. 12. 12. 2022. [citováno 4. 1. 2024].
https://www.robotics247.com/article/ouster_to_reveal_new_lidar_product_designed_for_smart_infrastructure_apps_at_ces_2023.

[4] Arttu Miettinen. *Sferical coordinates*. online. 2011. [citováno 10. 1. 2024].
https://pi2-docs.readthedocs.io/en/stable/spherical_coordinates.html.

[5] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*. 1989, 22 (6), 46–57.

[6] Ingemar J. Cox. Blanche-an experiment in guidance and navigation of an autonomous robot vehicle. *IEEE Transactions on robotics and automation*. 1991, 7 (2), 193–204.

[7] Oscar Martınez Mozos, Axel Rottmann, Rudolph Triebel, Patric Jensfelt, Wolfram Burgard a others. Semantic labeling of places using information extracted from laser and vision sensor data. *From Sensors to Human Spatial Concepts*. 2006, 33.

[8] You Li a Javier Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*. 2020, 37 (4), 50–61.

[9] Shea Hagstrom, David Messinger a Scott Brown. *Feature extraction using voxel aggregation of focused discrete lidar data*. In: *Laser Radar Technology and Applications XV*. 2010. 300–310.

[10] Shaobo Xia, Dong Chen, Ruisheng Wang, Jonathan Li a Xinchang Zhang. Geometric primitives in LiDAR point clouds: A review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*. 2020, 13 685–707.

[11] Martin A. Fischler a Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. 1981, 24 (6).

[12] Konstantinos G. Derpanis. *Overview of the RANSAC Algorithm*. PDF. 13. 5. 2010. [citováno 10. 1. 2024].
http://www.cse.yorku.ca/~kosta/CompVis_Notes/ransac.pdf.

[13] Václav Hlaváč. CTU in Prague. *Line detection by RANSAC*. PDF. 2020. [citováno 12. 1. 2024].
http://people.ciirc.cvut.cz/~hlavac/TeachPresEn/15ImageAnalysis/44RANSAC.pdf.

[14] Philip Torr a A. Zisserman. MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*. 2000, 78 138-156. DOI 10.1006/cviu.1999.0832.

[15] O. Chum a J. Matas. *Matching with PROSAC - progressive sample consensus*. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005. 220-226 vol. 1.

[16] Christoph Dalitz, Tilman Schramke a Manuel Jeltsch. Iterative Hough Transform for Line Detection in 3D Point Clouds. *Image Processing On Line*. 2017, 7 184–196. `https://doi.org/10.5201/ipol.2017.208`.

[17] P.J. Besl a Neil D. McKay. A method for registration of 3-D shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1992, 14 (2), 239-256. DOI 10.1109/34.121791.

[18] Qingde Li a J.G. Griffiths. Iterative closest geometric objects registration. *Computers & Mathematics with Applications*. 2000, 40 (10), 1171-1188. DOI https://doi.org/10.1016/S0898-1221(00)00230-3.

[19] Hao Bai. ICP Algorithm: Theory, Practice and Its SLAM-oriented Taxonomy. *Applied and Computational Engineering*. 2023, 2 (1), 10–21. DOI 10.54254/2755--2721/2/20220512.

[20] *ROS Introduction*. online. 8. 8. 2018. [citováno 12. 1. 2024]. `http://wiki.ros.org/ROS/Introduction`.

[21] *ROS Concepts*. online. 20. 9. 2022. [citováno 12. 1. 2024]. `http://wiki.ros.org/ROS/Concepts`.

[22] iPoly3D. *Lowpoly Electrical Towers Free low-poly 3D model*. online. 19. 2. 2021. [citováno 12. 5. 2024]. `https://www.cgtrader.com/free-3d-models/exterior/industrial-exterior/lowpoly-electrical-towers`.

Příloha A
Přiložené soubory

K práci je přiložen soubor `priloha.zip` obsahující tři složky popsanými níže.

`proj/` ROS balíček s hlavním programem `pylon_reconstruction`, ve kterém je implementován algoritmus pro rekonstrukci stožáru vysokého napětí. Obsahuje všechny potřebné zdrojové soubory `*.cpp` ve složce `src/` a hlavičkové soubory `*.h` ve složce `include/`. Nachází se zde také konfigurační soubor `pylons.rviz` pro nastavení vizualizace v programu `rviz`.

`gazebo_simulation/` Složka obsahující soubory pro simulaci v Gazebo simulátoru, pro měření chyby lokalizace, obrázky a grafy s výsledky.

`others/` Složka se zbytkem souborů. Obsahuje `*.json` soubory s vlastnostmi LiDARu. Soubory `*.pcd` s uloženými mračny bodů a obrázky s grafem komunikace programu `pylon_reconstruction`.