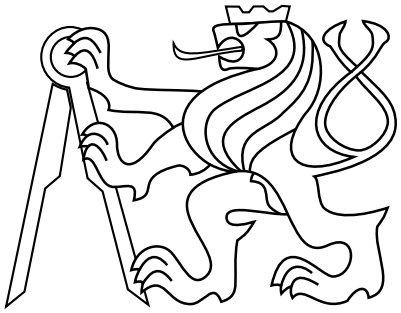
Czech Technical University in Prague

Faculty of Electrical Engineering
Department of Cybernetics
Multi-robot Systems



Self-Localization of an Unmanned Aerial Vehicle in the Transmission Tower Inspection Task

Bachelor's Thesis

Jan Kočí

Prague, May 2022

Study programme: Cybernetics and Robotics

Supervisor: Ing. Matěj Petrlík

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

Prague, 19. May, 2022

Acknowledgments

Firstly, I would like to express my gratitude to my supervisor Ing. Matěj Petrlík, for his help throughout this work. Then I would like to thank the MRS members who helped me with tests on real HW. And finally, my friends and family for their support.

BACHELOR‘S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Kočí Jan** Personal ID number: **492330**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Cybernetics**

Study program: **Cybernetics and Robotics**

II. Bachelor’s thesis details

Bachelor’s thesis title in English:

Self-Localization of an Unmanned Aerial Vehicle in the Transmission Tower Inspection Task

Bachelor’s thesis title in Czech:

Sebelokalizace bezpilotní helikoptéry v úloze inspekce stožárů elektrického vedení

Guidelines:

The aim of this thesis is to develop a method for self-localization of a UAV inspecting the electrical power distribution grid. The GPS and magnetometer measurements are degraded by the strong electromagnetic interference in such tasks. The following tasks will be solved:
• Propose an approach to improve the precision and reliability of pose estimation near transmission towers using onboard sensors. Consider employing a filtering approach [1] similarly as in [2].
• Empirically prove the increased precision over GPS-only localization in the Gazebo simulation environment.
• Integrate the developed method into the Multi-robot Systems Group UAV system [3] so that the estimated pose can be used in the feedback control loop.
• Verify the feasibility of deployment of the developed approach on the real hardware platform.

Bibliography / sources:

[1] Bishop, Gary, and Greg Welch. "An introduction to the kalman filter." Proc of SIGGRAPH, Course 8.27599-23175 (2001): 41.
[2] T Báča, P Štěpán, V Spurný, D Heřt, R Pěnička, M Saska, J Thomas, G Loianno and V Kumar. Autonomous landing on a moving vehicle with an unmanned aerial vehicle. Journal of Field Robotics 36(5):874-891, 2019.
[3] Tomas Baca, Matej Petrlik, Matous Vrba, Vojtech Spurny, Robert Penicka, Daniel Hert and Martin Saska. The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles. Journal of Intelligent & Robotic Systems 102(26):1–28, May 2021.

Name and workplace of bachelor’s thesis supervisor:

Ing. Matěj Petrlík Multi-robot Systems FEE

Name and workplace of second bachelor’s thesis supervisor or consultant:

Date of bachelor’s thesis assignment: **31.01.2022** Deadline for bachelor thesis submission: **20.05.2022**

Assignment valid until: **30.09.2023**

Ing. Matěj Petrlík	prof. Ing. Tomáš Svoboda, Ph.D.	prof. Mgr. Petr Páta, Ph.D.
Supervisor’s signature	Head of department’s signature	Dean’s signature

III. Assignment receipt

The student acknowledges that the bachelor’s thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the bachelor’s thesis, the author must state the names of consultants and include a list of references.

Date of assignment receipt

Student’s signature

Abstract

This thesis presents a localization method based on the fusion of GNSS and LiDAR measurements to improve the precision of the position estimate around high-voltage transmission towers. The study of autonomous unmanned aerial vehicles (UAVs) has become an important subfield of mobile robotics. One of its use cases is inspecting power lines and transmission towers and assisting in their maintenance. UAVs need precise navigation to function properly, especially in this task, as they can easily damage themselves or parts of power lines. Electromagnetic interference in the area further complicates this task by degrading the accuracy of magnetometers and GNSS, which are typically used for self-localization during outdoor missions. To reduce the influence of these interferences, we propose a system based on measurements from 3D LiDAR. This work describes the proposed system, including all the algorithms used, and evaluates the estimation error using a precise ground truth.

Keywords Unmanned Aerial Vehicles, Self-Localization, Kalman Filter, Point Clouds, ICP

Abstrakt

V této práci je představena metoda lokalizace založená na fúzi měření GNSS a LiDAR pro zlepšení přesnosti odhadu polohy v okolí stožárů vysokého napětí. Studium autonomních bezpilotních prostředků (UAV) se stalo významnou podoblastí mobilní robotiky. Jedním z případů jeho využití je kontrola elektrického vedení a pomoc při jeho údržbě. UAV potřebují pro správnou funkci přesnou navigaci, při tomto úkolu obzvlášť, protože mohou snadno poškodit sebe nebo části elektrického vedení. Elektromagnetické rušení v oblasti tento úkol dále komplikuje tím, že zhoršuje přesnost magnetometrů a GPS, které se obvykle používají pro vlastní lokalizaci při venkovních misích. Abychom se těmto rušivým vlivům vyhnuli, navrhujeme systém založený na měření z 3D LiDAR. Tato práce popisuje navrhovaný systém včetně všech použitých algoritmů a vyhodnocuje chybu odhadu oproti skutečné pozici.

Klíčová slova Bezpilotní Prostředky, Sebelokalizace, Kalmanův Filtr, Oblaky Bodů, ICP

Abbreviations

DOF degree-of-freedom

GNSS Global Navigation Satellite System

IMU Inertial Measurement Unit

LKF Linear Kalman Filter

MRS Multi-robot Systems Group

SLAM Simultaneous Localization And Mapping

UAV Unmanned Aerial Vehicle

ICP Iterative Closest Point

FCU Flight Control Unit

GICP Generalized Iterative Closest Point

Contents

1 Introduction **1**
1.1 Related works . . . 1
1.2 Contributions . . . 2
1.3 Thesis outline . . . 2
1.4 Mathematical notation . . . 3

2 Solution overview **4**

3 Point cloud filtration **6**
3.1 Voxel grid downsampling . . . 6
3.2 Extraction of close points . . . 7
3.3 Extraction of ground . . . 7

4 Transmission tower segmentation **9**
4.1 Euclidean Cluster Extraction . . . 9
4.2 Principal Component Analysis (PCA) . . . 10

5 Iterative Closest Points **11**
5.1 Models . . . 11
5.2 Point cloud densification . . . 13
5.3 Constraining degree-of-freedoms (DOFs) . . . 13
5.4 False positives rejection . . . 14
5.5 Estimated position . . . 14

6 Linear Kalman filters (Linear Kalman Filters (LKFs)) **16**
6.1 Horizontal filter . . . 16
6.2 Heading filter . . . 17

7 Experimental results **18**
7.1 Simulation results . . . 18
7.2 Real-world results . . . 19
7.3 Results summary . . . 21

8 Integration into the MRS UAV system **22**

9 Conclusion **24**

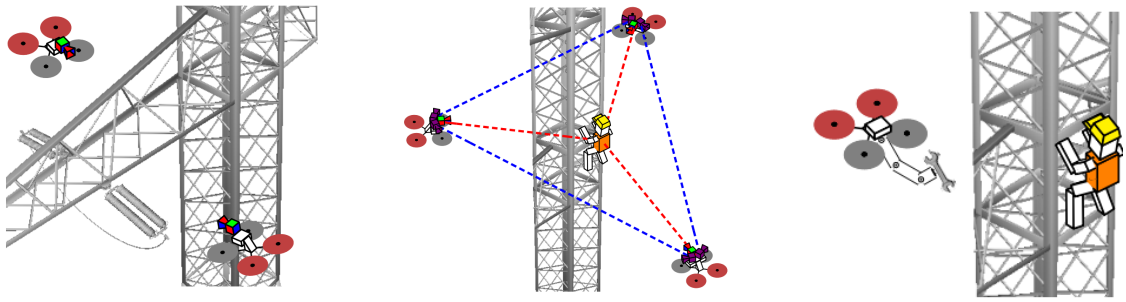
10 References **25**

A Appendix **27**

Chapter 1

Introduction

The inspection of electrical power lines is crucial for the functionality of the current grid. Technicians need to inspect a third of the grid every year. MRS group is researching a way to facilitate this work using Unmanned Aerial Vehicles (UAVs). The idea is that UAVs would inspect the towers or help technicians with repairs as described in [1].



(a) (b) (c)

Figure 1.1: Visualization of possible usage of UAVs in power tower inspection task. Inspection of parts of the power tower Fig. 1.1a. Monitoring the safety of the workers Fig. 1.1b. Delivery of the equipment Fig. 1.1c. Those images were taken from article [1].

Those tasks demand precise navigation to prevent damage to both UAVs and inspected power towers. The subtask of navigation is self-localization, which is the primary goal of this thesis. There is substantial electromagnetic interference in the proximity of the power lines, and it degrades the precision of GPS and magnetometers. We are going to use sensors that are not affected by this electromagnetic field.

Later we are going to describe the used algorithms. Furthermore, their adjustment is needed for use in self-localization tasks. It mainly means finding a compromise between computational speed and precision.

1.1 Related works

Self-localization of the UAVs is a topic studied in several works using different approaches using various sensor modalities. The most commonly used system is based on Global Navigation Satellite System (GNSS), such approach is used in [7], [8], [16] in combination with Inertial Measurement Unit (IMU). In [4] UAV swarms are primarily localized by GNSS, and the drift of individual agents is corrected by mutual distance observations while trying to eliminate information from UAVs that bring false information. When GNSS-based localization

cannot be used, such as in churches or subterranean environments, a solution based on a 2D LiDAR sensor is viable [3]. Employing 2D LiDAR as the main source of localization usually assumes fixed flight altitude or an environment with constant cross-section, such as tunnels in the first round of the DARPA Subterranean challenge [5].

Other authors are using sensors such as RGB-D cameras. They can be used both onboard [10], [17] and offboard [14]. Using offboard cameras can be precise but limits the operational area of the mobile agent. Authors of [11] show the fusion of IMU, camera, and 2D LiDAR compared to 3D LiDAR. In [12], algorithms for navigation in 2D environments based on LiDAR scanning and IMU are described. The navigation system from [20] uses stereo vision cameras, IMU, and altitude sensors. This approach helped the UAV estimate its position without using GPS and artificial landmarks. Authors of [15] propose Simultaneous Localization And Mapping (SLAM) for an indoor environment based on LiDAR and MEMS IMU. Their system can be used even on small UAVs with limited computational power.

We use Iterative Closest Points in our work. This is a popular algorithm for matching point clouds. It is implemented in *Point Cloud Library*[1], many of the algorithms from this library, which we encounter, refer to [22]. There are many versions of this algorithm basic version *point-to-point* is described in [27]. Other version *point-to-plane* is described in [24], and more modern version, which combines these two into one is Generalized Iterative Closest Point (GICP) described in [23]. In [19] there are variants of ICP compared according to their results in mapping tasks in different environments. A review of geometric registration is in [13]. It describes the ICP algorithm and its development in the past.

Later in work, we show the use of the Kalman Filter for fusing data from different sources. This is a commonly used filter for removing Gaussian noise, typically found in measurements. In [18] it is shown in the task of position estimation based on known distance to reference points. In [21] it is used for attitude estimation of fixed-wing UAV. Authors of [25] describe the use of Kalman Filter for 2D and 3D localization. Both of those localizations are useful as we localize UAVs in 3D and simplify our task to 2D because the electromagnetic fields do not distort altitude measurements.

1.2 Contributions

This thesis's primary goal is to propose a new system for self-localization based on onboard sensors. We are using 3D LiDAR as our sensor of choice, and we are going to use it to localize geographically important objects in proximity of the UAV. Knowing our relative position and orientation to an object and the absolute position and orientation of this object in world coordinates, we can estimate our position.

1.3 Thesis outline

In chapter 3 we are introducing methods used for filtering important information from LiDAR scans. Chapter 4 describes methods used for clustering points in a point cloud and for segmenting objects. Later we describe the use of the Iterative Closest Points algorithm in chapter 5 and how we obtain information about the position from it. In the chapter 6, we describe the use of Kalman filter for the fusion of data from different sources. We show

[1] https://pointclouds.org/

an overview of all algorithms used in chapter 2. All used algorithms are there described, and diagrams show their order.

Later in chapter 7, we show our experimental results from both simulation and real-world experiments. Graphs illustrate the quality of both GPS-based and proposed systems.

1.4 Mathematical notation

In this section, we show used mathematical notation.

$\mathbf{x}$, $\boldsymbol{\alpha}$	vector, pseudo-vector, or tuple
$\hat{\mathbf{x}}$, $\hat{\boldsymbol{\omega}}$	unit vector or unit pseudo-vector
$\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \hat{\mathbf{e}}_3$	elements of the *standard basis*
$\mathbf{X}, \boldsymbol{\Omega}$	matrix
$\mathbf{I}$	identity matrix
$x = \mathbf{a}^\intercal \mathbf{b}$	inner product of $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\mathbf{x} = \mathbf{a} \times \mathbf{b}$	cross product of $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\mathbf{x} = \mathbf{a} \circ \mathbf{b}$	element-wise product of $\mathbf{a}$, $\mathbf{b} \in \mathbb{R}^3$
$\mathbf{x}_{(n)} = \mathbf{x}^\intercal \hat{\mathbf{e}}_n$	n^{th} vector element (row), $\mathbf{x}, \mathbf{e} \in \mathbb{R}^3$
$\mathbf{X}_{(a,b)}$	matrix element, (row, column)
x_d	x_d is *desired*, a reference
$\dot{x}, \ddot{x}, \dddot{x}, \ddddot{x}$	1^{st}, 2^{nd}, 3^{rd}, and 4^{th} time derivative of x
$x_{[n]}$	x at the sample n
$\mathbf{A}, \mathbf{B}, \mathbf{x}$	LTI system matrix, input matrix and input vector
SO(3)	3D special orthogonal group of rotations
SE(3)	*SO(3)* $\times \mathbb{R}^3$, special Euclidean group

Table 1.1: Mathematical notation, nomenclature and notable symbols.

Chapter 2

Solution overview

This chapter will summarize the algorithms we have used in this thesis. Fig. 2.1 shows dataflow through nodes described in chapters 3, 4, 5 and 6.

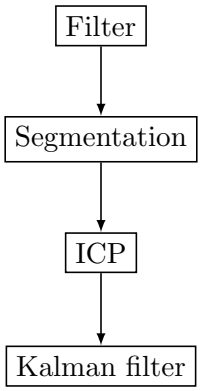


Figure 2.1: Diagram of dataflow through nodes used for this thesis.

The filter node receives raw data from 3D lidar and processes them with downsampling, extraction of close points, and extraction of the ground. The segmentation node uses a processed scan from the filter node and sends out the segment of the point cloud, which is the most similar to the transmission tower. ICP node uses point cloud from clustering node and model of the pillar and its position known in advance and returns estimated position. Kalman filter combines position from ICP node and GPS and returns our final estimation.

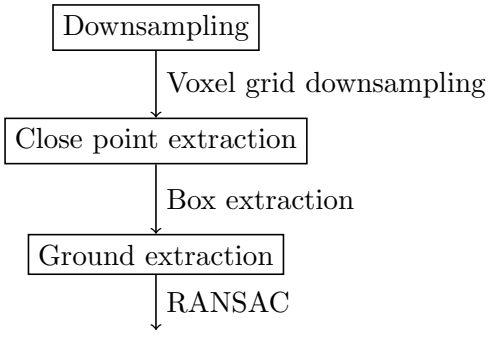


Figure 2.2: Diagram of dataflow in filter node.

In Fig. 2.2 you can see algorithms used in the filter. Algorithms in this work are used in this order, but it is not necessary to strictly follow this order. Order of ground extraction and close point extraction can be interchanged. It is only essential to use downsampling as the first step. Otherwise, the computation would take too long.

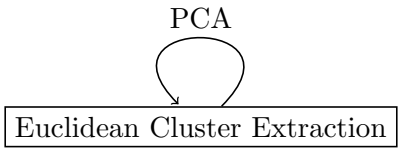


Figure 2.3: Diagram of function of clustering node.

Fig. 2.3 visualize function of clustering node. Euclidean cluster extraction returns several clusters. We then use PCA for each cluster and check if it meets the given conditions. Those conditions are the size of the cluster and a threshold for vectors obtained from the PCA of a given cloud based on the known dimensions of the transmission tower.

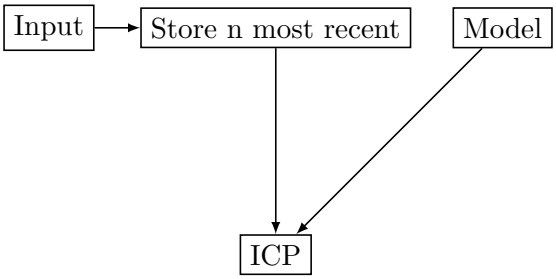


Figure 2.4: Diagram of function of ICP node.

In Fig. 2.4 it is shown, how ICP node works. It takes a cluster from the previous node and stores it in memory. Then it combines n most recent clusters into one. This point cloud and model of the power tower are given to ICP, and it returns the relative position of the power tower. ICP used in this work has constrained DOFs to 3.

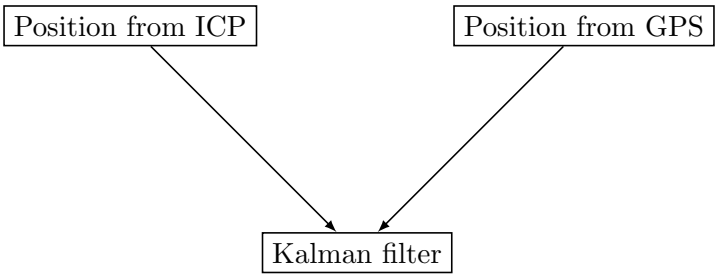


Figure 2.5: Diagram of function of Kalman Filter node.

In Fig. 2.5 it is shown which data Kalman filter uses. It uses data from our system represented in node *Position from ICP* and data from the UAV localization system *Position from GPS*.

Chapter 3

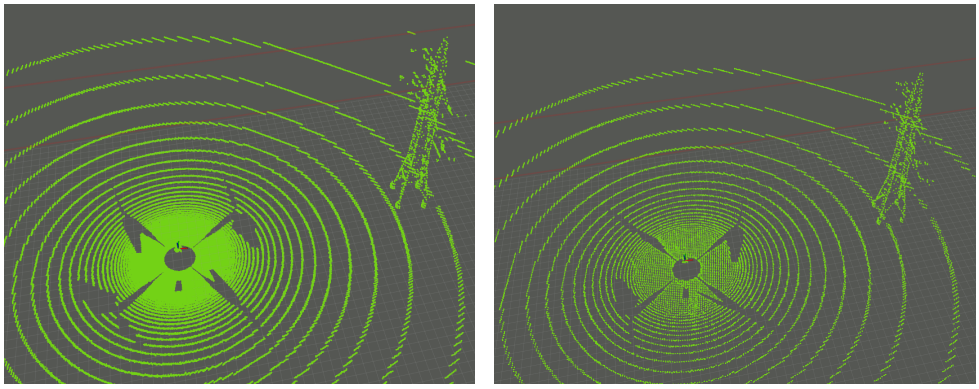
Point cloud filtration

Filtration is a process of separating important and irrelevant data from sensors. The main goals are to remove irrelevant points and reduce the number of points in the LiDAR scan. Point cloud size reduction is essential for the computational speed of the matching process. Removing irrelevant points is helpful for later segmentation.

3.1 Voxel grid downsampling

The method used for downsampling (reducing the number of points) is a voxel filter. 3D voxel grid is created over a point cloud, then all points within a voxel are replaced by its centroid.

In this project, we are using cubical voxels, block voxels could also be used. It is possible to ignore voxels with a less than set amount of points, which might be helpful for removing false positives of a noisy sensor. This was tested but is not further used because it removes points representing thin parts of pillars.



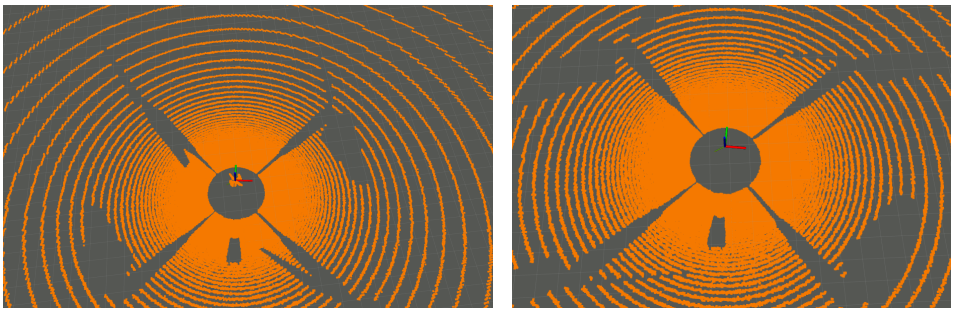
(a) Raw data from the sensor (b) Output of downsampling

Figure 3.1: Comparison of raw Fig. 3.1a and processed Fig. 3.1b data.

In the figure Fig. 3.1 we can see a comparison of point clouds before and after downsampling. The figure Fig. 3.1a has over 262000 points and Fig. 3.1b has only 14000 points, that is 14 times lower this improves speed of computation significantly. However, this example is from a relatively open world with few obstacles. In worlds with more objects in proximity of the sensor, this ratio will not be that significant. Notice that objects in the distance from the UAV are not affected as much as the ground in the area close to the UAV. This means that this procedure does not complicate the recognition of objects in the distance, which would be a typical use case because of safety reasons.

3.2 Extraction of close points

Each laser scan includes some points corresponding to parts of the UAV as rays hit the body of the UAV, the propellers, etc. Those points do not bring any information about the position of a pillar and could further complicate segmentation. Such points should be removed. Knowing the size of the UAV, we can remove all points in a box around the UAV.



(a) Raw data from the sensor (b) Output of downsampling

Figure 3.2: Comparison of raw Fig. 3.1a and processed Fig. 3.1b data.

In testing, we have found that the number of points discarded with this filter is not significant after downsampling. Because there are not enough points that would be recognized as a cluster of points, this filter is only optional.

3.3 Extraction of ground

Similarly, as parts of a UAV, the ground does not bring any information about the position of pillars. Let us assume that the ground around the UAV is flat, then it can be segmented and later removed. To decide which points belong to the ground, we are using RANSAC (Random sample consensus) algorithm.

A model to fit is needed for running RANSAC. Also, the parameters maximal number of iterations m and inlier distance threshold d have to be set to appropriate values for correct operation. In our case the planar model is used, $m = 100$ and $d = 50$ cm.

RANSAC algorithm works in these steps:

1. Randomly picks a subset of points. For fitting the planar model, this subset contains 3 points.
2. Count inliers. Inliers are all points in a distance less than d from the model.

After m iterations, we choose the model with the most inliers as the desired plane.

If this plane corresponds to the ground, then we can proceed to remove it. To check if the plane actually corresponds to the ground, we can look at its geometrical orientation. Used method is returning coefficients a, b, c, d of the plane model:

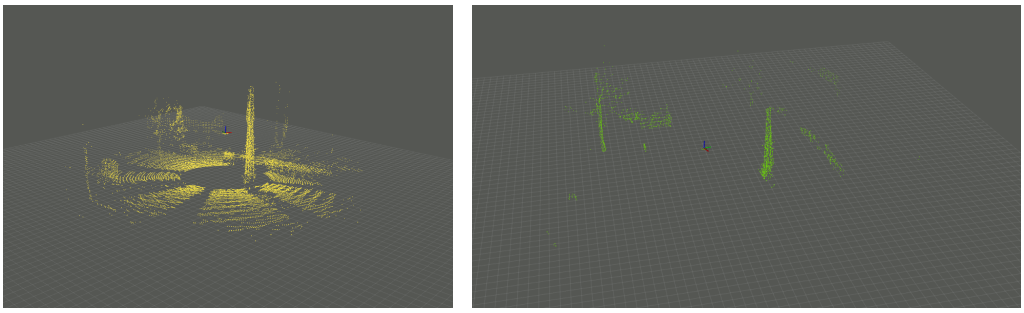
$$ax + by + cz + d = 0. \tag{3.1}$$

Assuming that our ground is perfectly horizontal, i.e., its normal is $\mathbf{n} = [\,0\ 0\ 1\,]^T$, than we can stop the algorithm from removing planes which have the angle θ between their normal $\mathbf{n_p}$ and

$\mathbf{n}$ larger than a predetermined value θ_{max}. This angle is computed simply from

$$\cos\theta = \frac{\mathbf{n}_p^T \mathbf{n}}{|\mathbf{n}_p| \cdot |\mathbf{n}|}. \tag{3.2}$$

Actually, we only need the absolute value of this angle, and in this work, we are using $\theta_{max} = 10°$. Usually, all points of the ground are not removed in one run. To remove the remaining points, we can run ground segmentation again. The problem with this algorithm is that it can always find a planar model when there are enough points. After removing points belonging to the ground, it will start removing parts of the pillars. In the actual use case, we are navigating the UAV around power lines, and there are always some objects that will not be part of the ground, but making sure that this algorithm does not spend too much time removing these points, we stop after there are less than 1% of points remaining (from point cloud after downsampling and extraction of UAVs points, not original). Another time when we stop filtration is after $\theta > \theta_{max}$ repetetively.



(a) Raw data from ouster (b) Output of filtering

Figure 3.3: Comparison of raw Fig. 3.3a and processed Fig. 3.3b data.

In Fig. 3.3 the point clouds before and after filtering are compared. First subfigure Fig. 3.3a shows data directly the from sensor and the second subfigure Fig. 3.3b shows the processed image. Notice missing ground, points of the UAV, and lower density of points. Another noticeable thing is that very few points of the pillar are removed. Only points at the basis of the pillar that is in the distance lower than d from the ground are affected by this filter.

Chapter 4

Transmission tower segmentation

In the context of point clouds, clustering is a process of dividing the entire set of points into smaller compact parts based on some metric. Our goal is to obtain a cluster representing an electrical pillar. In Fig. 4.1 a typical input can be seen. Notice that most ground points are already filtered by techniques from the chapter 3. It consists of points representing pillars, cables, a residue of ground, and other objects, e.g., people.

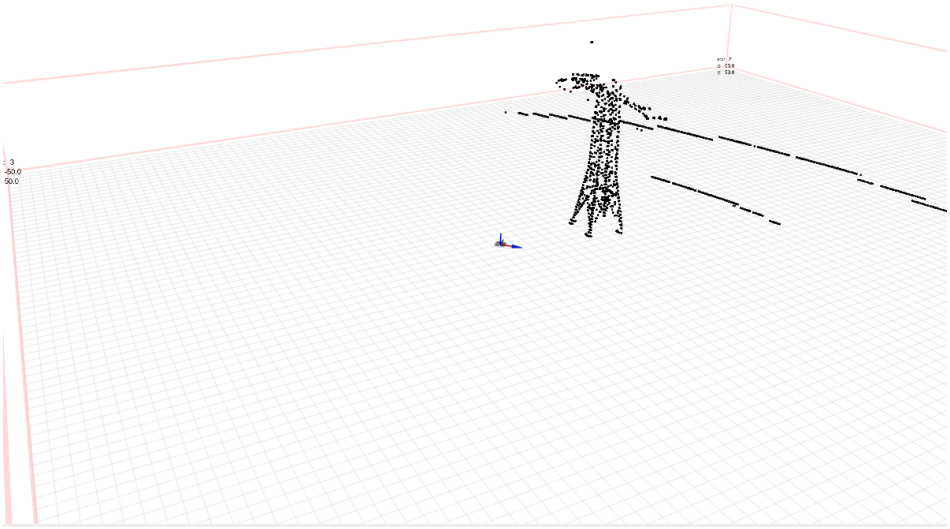


Figure 4.1: Typical input data.

4.1 Euclidean Cluster Extraction

Euclidean clustering extraction is a simple clustering process. Clusters are determined only by the Euclidean distance of points. With this data clustering method, the space is divided into fixed-width boxes. [22]

The algorithm searches for neighboring points and proclaims them as elements of the same cluster. We are using the Kd-tree structure for finding the nearest neighbors to speed up the computation. The algorithm works as follows:

Algorithm 1 Euclidean Cluster Extraction

create a Kd-tree representation for the input point cloud dataset P
initialize empty lists C and Q (clusters, queue of points to check respectively)
for $\mathbf{p}_i \in P$ **do**
  add $\mathbf{p}_i$ to the queue Q
  for $\mathbf{q}_i \in Q$ **do**
    add all unprocessed points within radius r to Q
  end for
  when all point in Q have been processed, add Q to C
  reset Q to an empty list
end for

With the right choice of parameter r, we can successfully segment pillars from up to 35 m. They consist of approximately 400 points from this distance.

4.2 Principal Component Analysis (PCA)

PCA is a method to obtain orthogonal unit vectors corresponding to the directions of the highest point cloud variance. Let $\mathbf{P}$ be the matrix of all points of a cluster, then the eigenvector $\mathbf{v}$ corresponding to the highest eigenvalue of $\mathbf{P}^T\mathbf{P}$ represents the main direction of the cluster. More precisely, it is the direction in which the sum of squared distances of all points is minimal. Assuming the points are in an untilted frame with zero roll and pitch angles, we can ignore all clusters that do not fit in the chosen threshold of the angle between $\mathbf{v}$ and the vertical vector. This should remove objects that are not vertically oriented, such as bushes, cars, fences, etc.

Chapter 5

Iterative Closest Points

Iterative Closest Points (ICPs) is an algorithm to align two sets of points or a set of points and an object in space. One set is used as a reference, and the second tries to fit on it. It is often used to align partially overlapping lidar scans. In our case, we are using it to localize a known object in a scan from the UAV. The output of this algorithm is a transformation. If this transformation is applied to the input set of points, then sets are aligned.

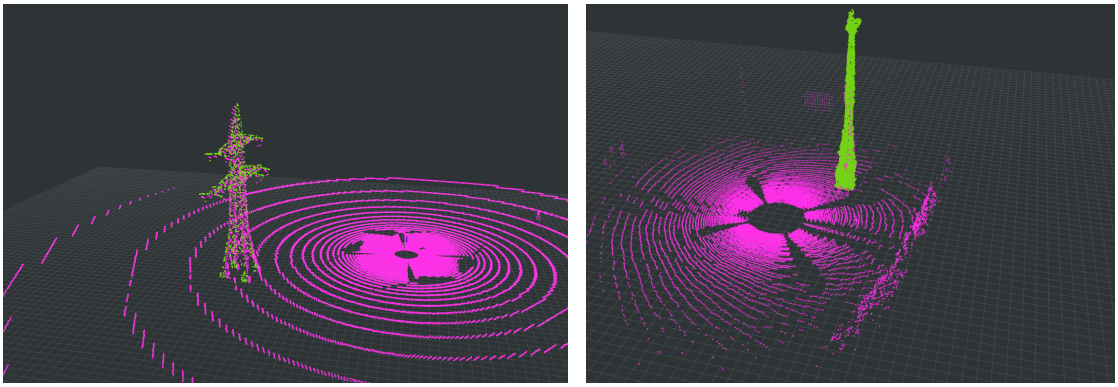
Suppose both sets were the same only with translation and rotation differences. In that case, this algorithm should always converge to the global minimum, and the final value of the optimized metric would be zero. However, this is an ideal scenario only, which can only happen if point clouds are generated from a 3D model of an object, and one is transformed. Therefore, we have to set a threshold on the optimized criterion, and when it is hit algorithm stops [23]. In our case, we use the processed LiDAR scan as the reference and a pillar model as the cloud to be aligned. The simulation model is generated from a model used in Gazebo; in real-world experiments, we are using a scan of a real pillar.

In Fig. 5.1 you can see how the models (green) are aligned to the original point clouds (pink). Notice that in Fig. 5.1b lidar has significantly shorter range then in Fig. 5.1a. To capture a scan of the pillar, the UAV has to be in proximity to the pillar, limiting its ability to see it whole.

5.1 Models

Our models are created using the software Cloud Compere, where we can generate point clouds with an arbitrary number of points on a given mesh object sample. We have settled on 1000 points per model as a compromise between computational time and accuracy through our testing. An example of a model generated by this technique is shown in Fig. 5.2a. Using kd-tree data structure ICP algorithm requires $O(n \log n)$ comparisons [23].

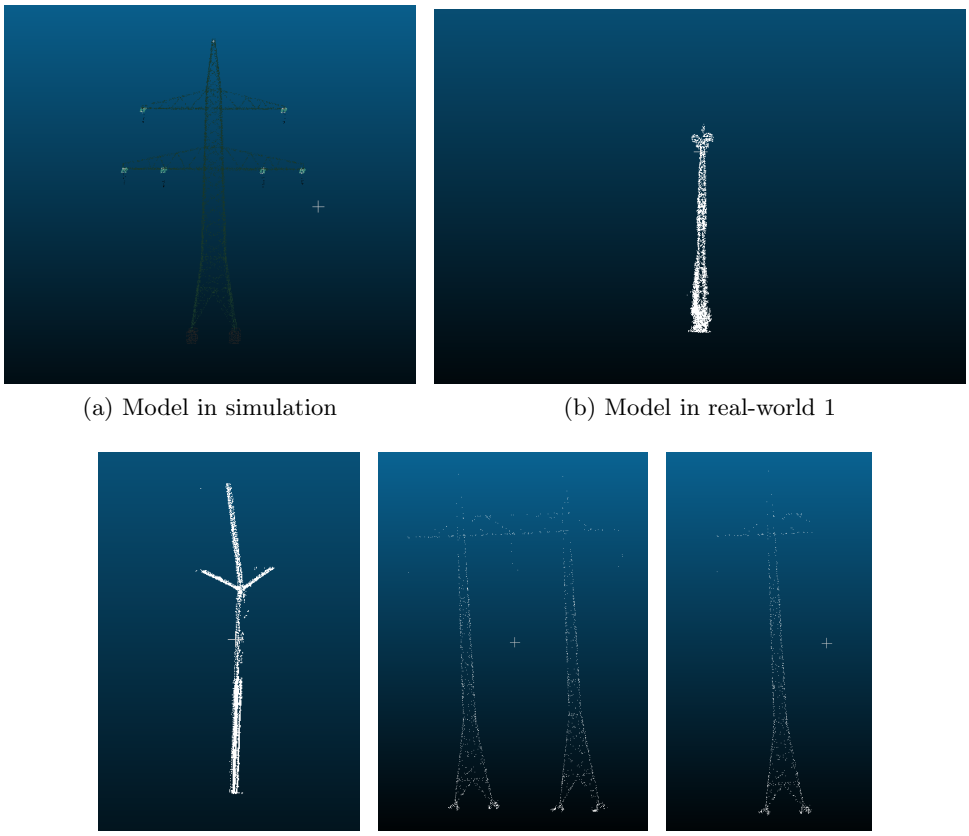
The quality of models is crucial for the correct function of the algorithm. As real-world models are scanned on-site, they can then contain parts of close objects and ground, which have to be removed. Model in Fig. 5.2d and Fig. 5.2e was generated from scan taken on site. However, as it was taken on a hill, it is not vertical, and offset had been used to correct it. Fig. 5.2e is another adjustment of the used model because the used sensor was not able to capture both parts of the pillar from all positions. Furthermore, we wrongly recognize them as two separate objects if it does. Also, it helps if the pillars have significant features like horizontal side-branches.



(a) Alignment of measured lidar scan and model in simulation.

(b) Alignment of measured lidar scan and model in real-world experiment.

Figure 5.1: Comparison of alignments obtaine from ICP in simulation Fig. 5.1a and in real-world experiments Fig. 5.1b.



(a) Model in simulation

(b) Model in real-world 1

(c) Model in real-world 2

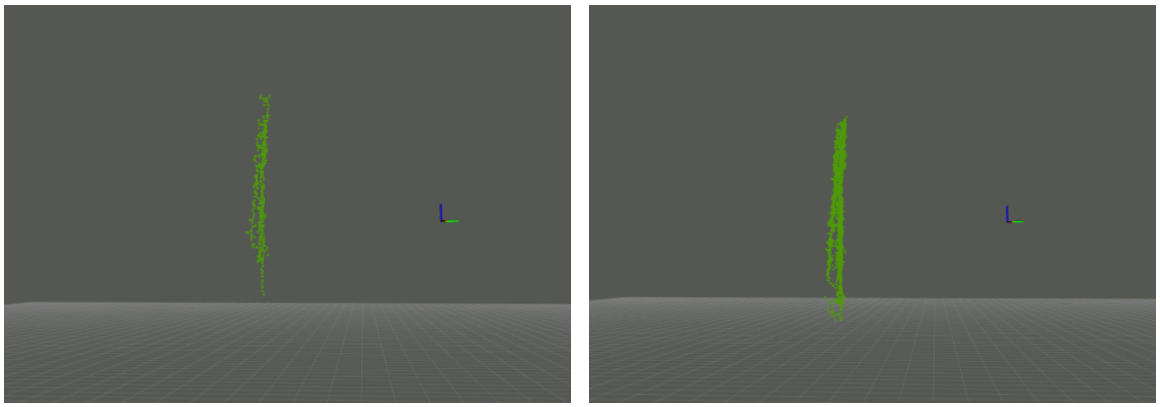
(d) Model in real-world 3

(e) Model in real-world 3 adjusted

Figure 5.2: Models used for ICP in experiments. Fig. 5.2a is model used in simulation, Fig. 5.2b, 5.2c, 5.2d are models used in real-world experiments. Fig. 5.2e is adjusted point cloud of model Fig. 5.2d used in experiments in Libomyšl.

5.2 Point cloud densification

For good convergence, ICP needs a large number of points. Unfortunately, in a single scan, there are usually not enough. We can use more scans and combine them into a single point cloud as a substitution for this. We transform scans into stable frames; those are frames that do not move relative to the world in time and store the most recent ones to achieve this. With a new message, we can transform those scans into the UAV frame and combine them. In our system, we are using five to eight scans. The optimal number depends on the used sensor and the complexity of the used models. With more points in a single cluster, we need fewer of those clusters. Models with significant features need less augmentation. In best-case scenarios, this feature is unnecessary, and the algorithm can converge even from only one most recent scan.



(a) Single scan. (b) Combined scan of five clouds.

Figure 5.3: Input point cloud for ICP Fig. 5.3a shows single cloud, Fig. 5.3b shows combination of 5 most recent scans.

In Fig. 5.3 we can see the results of combining five scans into one point cloud. In Fig. 5.3b there are more points on the distant side of the pillar. This helps ICP with aligning clouds more precisely. With more dynamic flight, there can also be a more significant difference in parts that are seen in every scan, and the combined image would then contain a more significant part of the pillar.

5.3 Constraining DOFs

Traditionally ICP works with 6 DOFs, translations on XYZ axes, and rotations along these axes. In a simulation, this works quite well, and alignments are precise and stable, unlike with real-world data, where the final alignment usually does not even have a vertical orientation. Imperfections probably cause this behavior in the model used in the experiment and in the overall simplicity of the used model. Unlike the simulation model, it lacks significant features. To remove this erratic behavior, we can ease ICP's work by reducing DOFs in which it can transform the model. We only need movement in the horizontal plane for satisfactory results, i.e., x and y axes, and rotation around the z-axis because the processed scan was previously transformed into the untilted frame with zero rotations around the x and y axes.

5.4 False positives rejection

After all limitations from chapter 4, we could have more than one cluster remaining to accurately decide which are true positives (pillars) and false positives (trees, parts of structures, generally tall vertical objects). We use ICP for given clusters to determine which are false positives. It returns a score of fitness. If this score is over a set threshold, we reject this cluster and continue with the next one.

5.5 Estimated position

ICP returns transformation matrix $\mathbf{T_{ICP}}$ from the UAV frame to the pillar, and the position of the pillar is *a priori* known from geo-localization. To get an absolute position of the UAV, we need a relative position and orientation to the pillar. This is obtained from $\mathbf{T_{ICP}}$. The translation represents position, and rotation represents orientation. Unfortunately, the stability of orientation is strongly dependent on the shape of the pillar. The more symmetrical it is, the less precise can the estimation of rotation be. In the case of cylindrical pillars, there is no information about relative orientation.

Within our experiments, we have tried the functionality of different models. In the case of pillars with a shape similar to the model in figure Fig. 5.2a, the branches of this model reduce its symmetry, and ICP is capable of aligning the model to the captured point cloud precisely. However, even then, there are two possible orientations. In other cases, such as the model in Fig. 5.2b with four symmetrical sides and noise, ICP is unable to align the model repetitively, and the whole model is moving and rotating around the vertical axis, returning arbitrary orientations.

However, orientation is vital for the calculation of position, so to obtain some results, we are using orientation from the onboard Flight Control Units (FCUs), which fuses data from accelerometers, gyroscopes, and magnetometers. The absolute position of the UAV $\mathbf{x}_{abs} = (x, y)^T$ is calculated according to the equation

$$\mathbf{x}_{abs} = \mathbf{x}_{pil} - \mathbf{R}(\theta)\mathbf{x}_{icp}, \tag{5.1}$$

where $\mathbf{x}_{pil}$ is absolute position of the pillar, $\mathbf{R}(\theta)$ is the rotation matrix with argument θ, which represents UAVs heading in world coordinates and $\mathbf{x}_{icp}$ is from

$$\mathbf{T_{ICP}} = \left(\begin{array}{ccc|c} & \hat{\mathbf{R}}(\hat{\theta}) & & \begin{matrix}\mathbf{x}_{icp}\\ z\end{matrix} \\ \hline 0 & 0 & 0 & 1 \end{array}\right). \tag{5.2}$$

The matrix $\hat{\mathbf{R}}(\hat{\theta})$ is rotation matrix which can be used for orientation estimation. As we have restricted rotations around x and y, our estimation of heading $\phi \in (-\pi, \pi)$ is simply $\phi = -\hat{\theta} + k\pi$, where $k \in \{-1, 0, 1\}$ for models with two symmetrical sides and $k \in \{\frac{-3}{2}, -1, \frac{-1}{2}, 0, \frac{1}{2}, 1, \frac{3}{2}\}$ for models with four symmetrical sides. The value of k depends on current orientation, so we should choose k that would have the smallest difference between the two measurements.

In testing, we found that ICP has a problem with converging if the model's starting position is far from the final position, limiting localization from a distance. To compensate for this, we can move the model to the centroid of the point cloud we are matching. From there it steadily converges even in less iterations. However that changes equation for position to

$$\mathbf{x}_{abs} = \mathbf{x}_{pil} - \mathbf{R}(\theta)(\mathbf{x}_{icp} + \mathbf{x}_{cen}), \tag{5.3}$$

where $\mathbf{x}_{cen} \in \mathbb{R}^2$ is a vector representing horizontal position of the centroid of the matched point cloud.

However, it cannot be generalized for arbitrary shapes, and if measurements are noisy, it can return any value. For example, if you have model Fig. 5.2a and the scan does not contain side-branches, ICP can converge into a local minimum where the model and the pillar are perpendicular to each other. The heading estimation would then have at least $\frac{\pi}{2}$ error for any k.

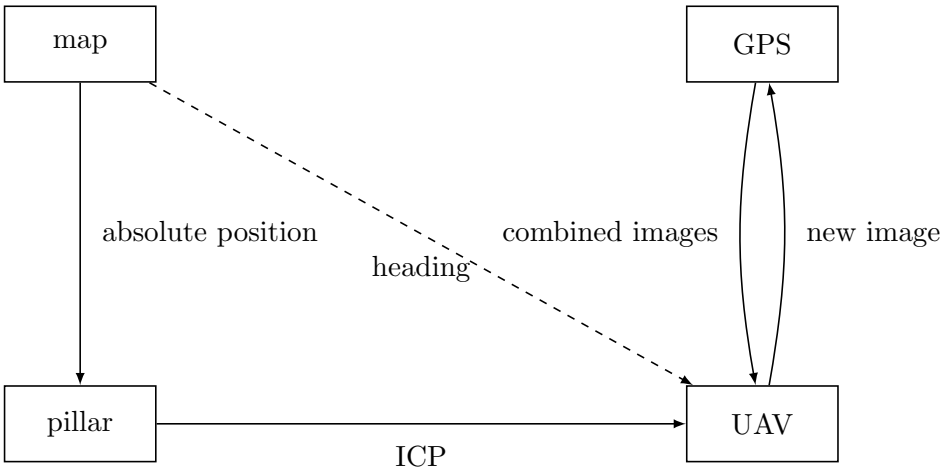


Figure 5.4: Diagram of how the absolute position of the UAV is obtained.

In Fig. 5.4 you can see how estimation of absolute position is obtained. We add the absolute position of the pillar with the relative position of the UAV to the pillar using knowledge about the relative rotation of the UAV. Edges between GPS and UAV nodes show our substitution for lack of points, where we use multiple most recent scans to help the convergence of the ICP.

Chapter 6

Linear Kalman filters (LKFs)

Kalman filtering is an algorithm that improves data corrupted by a zero-mean Gaussian noise. It takes measurements and models of the system and returns a state that is less noisy than a state only from measurement. LKF can also be used to combine (fuse) measurements from multiple sources into a single consistent state, which is employed in this thesis. It works in two stages, prediction and correction, that are repeated periodically. Prediction takes the model and computes expected new states based on past states, and correction takes measurement and corrects estimated state accordingly. The general state $\mathbf{x} \in \mathbb{R}^n$ is described by stochastic difference equation

$$\mathbf{x}_{n+1} = \mathbf{A}_n\mathbf{x}_n + \mathbf{B}\mathbf{u}_n + \mathbf{w}_n, \tag{6.1}$$

and measurement $\mathbf{z}_n \in \mathbb{R}^n$ is

$$\mathbf{z}_n = \mathbf{H}_n\mathbf{x}_n + \mathbf{v}_n. \tag{6.2}$$

The random variables $\mathbf{w}_n$ and $\mathbf{v}_n$ are independent from normal distributions

$$p(\mathbf{w}) \sim N(0, \mathbf{Q}), \tag{6.3}$$
$$p(\mathbf{v}) \sim N(0, \mathbf{R}). \tag{6.4}$$

Matrices $\mathbf{Q}$ and $\mathbf{R}$ refer to the covariance of process and measurement noise, respectively [26].

6.1 Horizontal filter

In our case, we need to align the position obtained from ICP in Chapter 5 and the position estimated by the flight control unit of the UAV based on GPS measurements. We only care about the horizontal position because sensors that determine the vertical position(barometer, laser rangefinder, and IMU) are not affected by disturbances produced by the high-voltage power lines. Thus, our system can be described with similar equations as in [6],

$$\mathbf{x}^e_{n+1} = \mathbf{A}\mathbf{x}^e_n + \mathbf{B}\mathbf{u}_n, \tag{6.5}$$
$$\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \ \mathbf{B} = \begin{bmatrix} \Delta t \\ \Delta t \end{bmatrix}, \tag{6.6}$$

where $\mathbf{x}^e_n \in \mathbb{R}^2$ represents position in iteration n. The upper index means that it is estimation before correction. The input $\mathbf{u}_n \in \mathbb{R}^2$ is

$$\mathbf{u}_n = \frac{(\mathbf{x}^e_n - \mathbf{x}^e_{n-1})}{\Delta t}, \tag{6.7}$$

and correction is described by equation

$$\mathbf{x}_n = \mathbf{x}^e_n + K(\mathbf{z}_n - \mathbf{H}\mathbf{x}^e_n), \tag{6.8}$$

where $\mathbf{z}_n \in \mathbb{R}$ is the measurement, $\mathbf{x}_n \in \mathbb{R}$ is the final estimated state after correction, K is Kalman gain, and $\mathbf{H}$ is a matrix mapping the state to the measurement. Kalman gain influences the effect of the innovation on the final estimate.

6.2 Heading filter

Similarily to the horizontal alignment, we are using LKF to fuse heading measurements from ICP and the internal heading estimate of the flight control unit based on the magnetometers. In this case there is only a single variable, so the equations are simplified to

$$x^e_{n+1} = Ax^e_n + Bu_n \tag{6.9}$$

$$u_n = \frac{(x^e_n - x^e_{n-1})}{\Delta t} \tag{6.10}$$

$$x_n = x^e_n + K(z_n - Hx^e_n) \tag{6.11}$$

$$A = 1, B = \Delta t, \tag{6.12}$$

where $x^e_n, u_n, K, R, x_n, z_n \in \mathbb{R}$ and has similar meaning as in previous section.

Chapter 7

Experimental results

This chapter shows the results of both real-world and simulation experiments. To generate results we are using library *rpg_trajectory_evaluation*[1] which is based on work [9]. We will compare our algorithm to a GPS-only-based system. As a ground truth in real-world experiments, we use RTK, which measures position within centimeter precision. In any of our experiments, we have not used UAVs around active electrical power lines for safety reasons. In Fig. 7.1 you can see photos of UAVs used for real-world experiments.

(a) (b)

Figure 7.1: Photos of used UAV in experiments. Fig. 7.1a shows the UAV during experiments in Libomyšl, Fig. 7.1b shows the UAV during experiments in Temešvár.

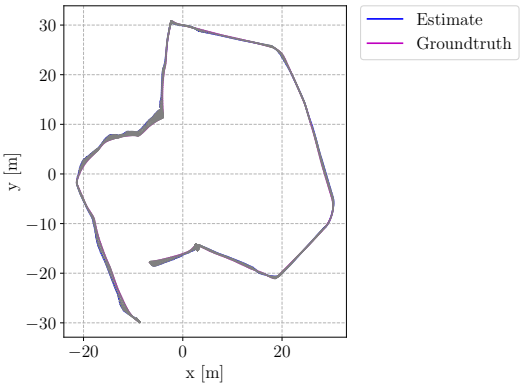
7.1 Simulation results

Fig. 7.2 shows a top view comparison of our systems result and GPS-based system in the simulation environment. The systems are compared on the same trajectory, which is a record of a UAV's flight around the power tower. It was scanned from all sides and at different distances. Those images show alignment in the x and y axes. The z-axis is not implemented in our system. Places with the most significant errors are places with a rapid orientation change, time to compute position is up to 100 ms, and change in position is usually not significant. However, a change in orientation can dramatically alter results.

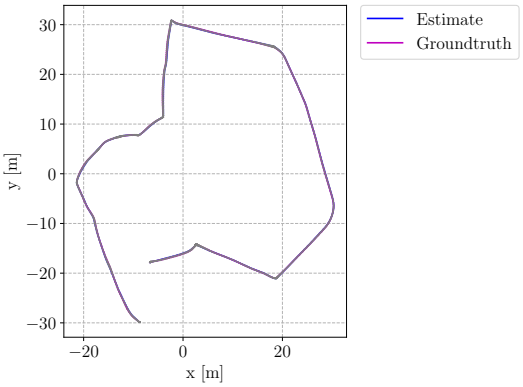
Fig. 7.3 shows relative translation error in particular segments of flight. The median error of our system is higher than the GPS-based system. However, our system should be more robust. Since the electrical towers do not produce any electromagnetic noise in simulation, the GPS is not disturbed more than usual.

A comparison of errors in each axis is shown in Fig. 7.4.

[1] https://github.com/uzh-rpg/rpg_trajectory_evaluation

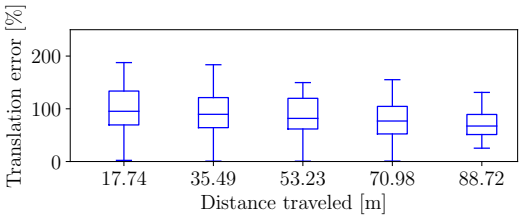


(a) Top view of trajectory alignment of the proposed system and ground truth.

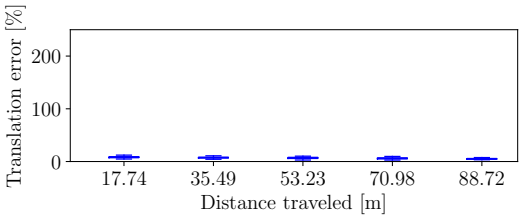


(b) Top view of trajectory alignment of the gps based system and ground truth.

Figure 7.2: Comparison of the alignments. Fig. 7.2a shows our system, Fig. 7.2b shows the GPS-based system.

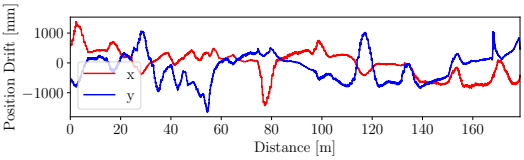


(a) Relative translation error of the proposed system.

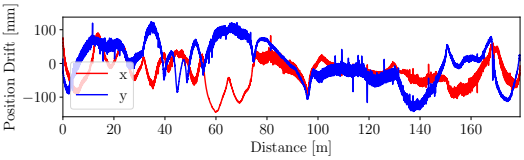


(b) Relative translation error of the gps based system.

Figure 7.3: Comparison of relative translation errors in simulation. Fig. 7.3a shows our system, Fig. 7.3b shows the GPS-based system.



(a) Translation error of the proposed system in specific axes.



(b) Translation error of the gps based system in specific axes.

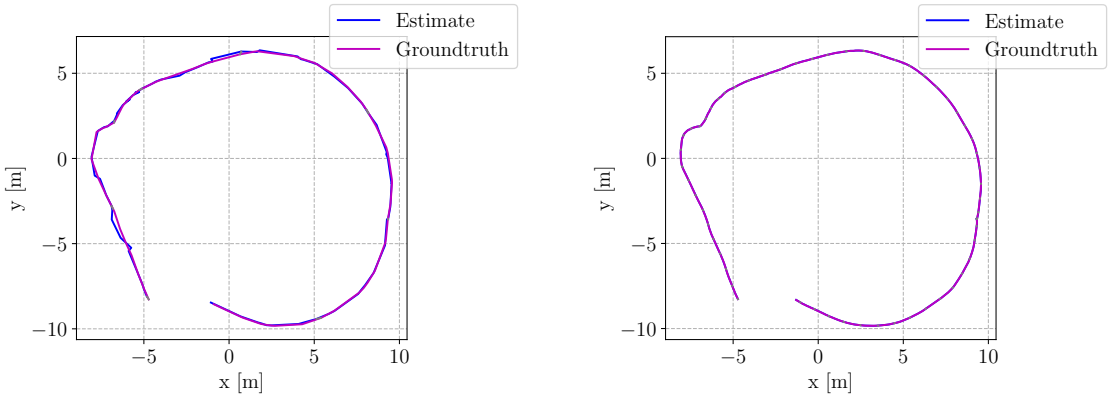
Figure 7.4: Comparison of errors in each axis in simulation. Fig. 7.4a shows our system, Fig. 7.4b shows the gps based system.

7.2 Real-world results

Fig. 7.5 shows a top view comparison of our systems result and current system in a real environment. Notice that our filter during the flight is continuously less disturbed. LKF is setting its weights for used sources during the run.

Here we are using RTK as ground truth and trajectories are evaluated in its coordinate system. Unlike in simulation ground truth trajectory cannot be measured in real-world, how-

ever RTK is several orders more accurate than both GPS and position estimation based on ICP.

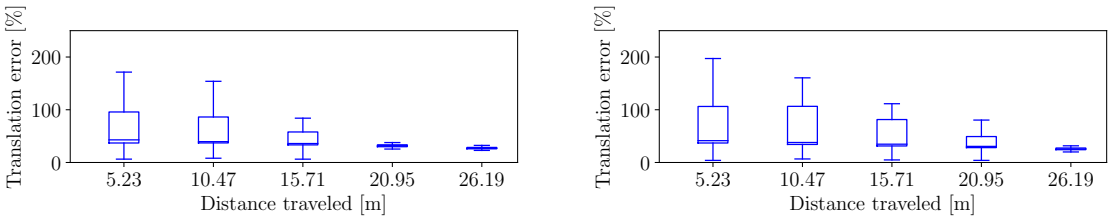


(a) Top view of trajectory alignment of the proposed system and ground truth.

(b) Top view of trajectory alignment of the GPS based system and ground truth.

Figure 7.5: Comparison of the trajectories alignment in real-world. Fig. 7.2a shows our system, Fig. 7.2b shows the gps based system.

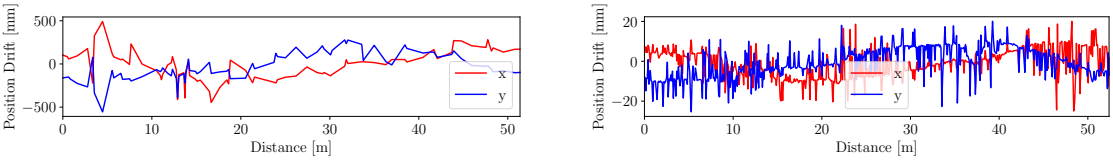
Fig. 7.6 shows relative translation error in particular segments of flight. Notice that disturbance for GPS-based system is more significant than in simulation, but still, there was no external source of electromagnetic noise because the transmission grid at the location of the experiment was turned off for maintenance. In Fig. 7.7 you can see errors in each axis.



(a) Relative translation error of the proposed system.

(b) Relative translation error of the GPS based system.

Figure 7.6: Comparison of relative translation errors in real-world experiment. Fig. 7.6a shows our system, Fig. 7.6b shows the gps based system.



(a) Translation error of our system in specific axes.

(b) Translation error of the gps based system in specific axes.

Figure 7.7: Comparison of errors in each axis in real-world experiment. Fig. 7.7a shows our system, Fig. 7.7b shows gps based system.

7.3 Results summary

Results show that in both simulation and real-world experiments, the GPS-based system is superior in tested conditions. Used LiDAR (ouster OS0-128) has a frequency of 10 Hz, time needed for capturing a scan combined with time needed to compute position from this scan can couse imperfections in position evaluation.

Another problem that our system has is that it is strongly dependent on the accuracy of measurement of the pillars, both orientation, and position. If there is an error, then our system would take it as a systematical error, which means its error would have a non-zero mean value, and that contradicts conditions of correct use of the Kalman filter.

The estimation of the heading is highly inaccurate. Locally, it returns values within 20 degrees of real heading; however, it cannot be used on a global scale. The result of heading estimation differs for every local minimum of the ICP evaluation function. The number of those minima depends on the shape of the pillar and noise. Without noise, we can try to correct those results by adding a constant to results according to their shape, but even this does not generally work for all shapes and positions of the UAV, and it depends on how well we can scan the pillars.

Chapter 8

Integration into the MRS UAV system

In this chapter, we are explaining how our system is supposed to be used in the current system architecture of the MRS UAV system [2]. The whole system uses the ROS framework. Using ROS, we can subscribe to topics used in the system. Essential for us are topics from LiDAR and GPS. From those topics, we obtain messages about the environment and position. Then, we process that information as described in previous chapters. Furthermore, publish our messages containing information about the position and orientation of the UAV to our custom topics. Those topics should be subscribed to and used in the feedback control loop. Architectire of the MRS UAV system can be seen in Fig. 8.1, green node is our system.

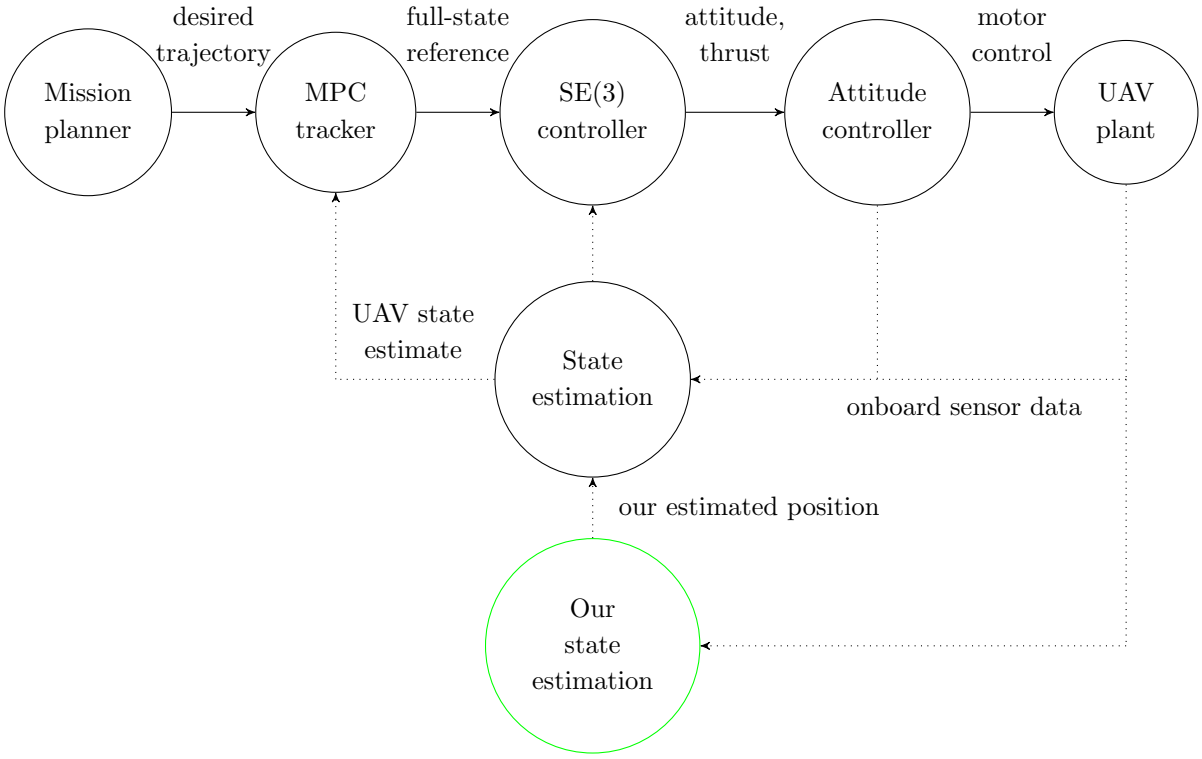


Figure 8.1: Diagram showing integration of our system into current architecture.

Depending on used sensors different state estimators can be combined into final state estimation. State estimation needs information about position $\mathbf{x}_n$ and velocity $\dot{\mathbf{x}}_n$ of the UAV. The position is obtained directly from message and velocity can be obtained from

$$\dot{\mathbf{x}}_n \approx \frac{\mathbf{x}_n - \mathbf{x}_{n-1}}{\Delta t}, \tag{8.1}$$

where we approximate time derivative of position with position difference over time.

Chapter 9

Conclusion

In this thesis, we have described and developed a system for the self-localization of a UAV based on 3D LiDAR data processing. This system should have had increased precision and reliability. We have proved that precision has not increased, but the system should be more robust and work even if GPS is distorted. Empirical tests were run on data both from simulation and real-world obtained from a real UAV. Later in work, we have shown integration of our system into Multi-robot Systems Group (MRS) UAV system, using ROS environment. This covers all tasks of the assignment.

For future work, we could use measurements from other sensors like IMU. The system is currently set to locate a single object and estimate the UAV position according to it. This can be scaled to identify more objects at once. Alternatively, to identify which object from a list of objects is detected.

Chapter 10

References

[1] A. Calvo, G. Silano, and J. Capitán, "Mission planning and execution in heterogeneous teams of aerial robots supporting power line inspection operations," in *International Conference on Unmanned Aircraft Systems*, 2022.

[2] T. Baca, M. Petrlik, M. Vrba, V. Spurny, R. Penicka, D. Hert, and M. Saska, "The MRS UAV System: Pushing the Frontiers of Reproducible Research, Real-world Deployment, and Education with Autonomous Unmanned Aerial Vehicles," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 102, no. 1, 2021. DOI: 10.1007/s10846-021-01383-5. arXiv: 2008.08050.

[3] M. Petrlik, T. Krajnik, and M. Saska, "LIDAR-based Stabilization, Navigation and Localization for UAVs Operating in Dark Indoor Environments," in *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2021, pp. 243–251. DOI: 10.1109/ICUAS51884.2021.9476837.

[4] S. Hacohen, O. Medina, T. Grinshpoun, and N. Shvalb, "Improved gnss localization and byzantine detection in uav swarms," *Sensors (Switzerland)*, vol. 20, no. 24, pp. 1–18, 2020. DOI: 10.3390/s20247239.

[5] M. Petrlík, T. Báča, D. Heřt, M. Vrba, T. Krajník, and M. Saska, "A robust uav system for operations in a constrained environment," *IEEE Robotics and Automation Letters (RAL)*, vol. 5, no. 2, pp. 2169–2176, 2020. DOI: 10.1109/LRA.2020.2970980.

[6] T. Baca, P. Stepan, V. Spurny, D. Hert, R. Penicka, M. Saska, J. Thomas, G. Loianno, and V. Kumar, "Autonomous landing on a moving vehicle with an unmanned aerial vehicle," *Journal of Field Robotics*, vol. 36, no. 5, pp. 874–891, 2019. DOI: 10.1002/rob.21858.

[7] G. Zhang and L. T. Hsu, "A New Path Planning Algorithm Using a GNSS Localization Error Map for UAVs in an Urban Area," *Journal of Intelligent and Robotic Systems: Theory and Applications*, vol. 94, no. 1, pp. 219–235, 2019. DOI: 10.1007/s10846-018-0894-5.

[8] ——, "Intelligent gnss/ins integrated navigation system for a commercial uav flight control system," *Aerospace science and technology*, vol. 80, pp. 368–380, 2018.

[9] Z. Zhang and D. Scaramuzza, "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry," *IEEE International Conference on Intelligent Robots and Systems*, pp. 7244–7251, 2018. DOI: 10.1109/IROS.2018.8593941.

[10] W. G. Aguilar, G. A. Rodríguez, L. Álvarez, S. Sandoval, F. Quisaguano, and A. Limaico, "Visual slam with a rgb-d camera on a quadrotor uav using on-board processing," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10306 LNCS, no. May, pp. 596–606, 2017. DOI: 10.1007/978-3-319-59147-6_51.

[11] H. Deilamsalehy and T. C. Havens, "Sensor fused three-dimensional localization using IMU, camera and LiDAR," *Proceedings of IEEE Sensors*, pp. 7–9, 2017. DOI: 10.1109/ICSENS.2016.7808523.

[12] R. Opromolla, G. Fasano, G. Rufino, M. Grassi, and A. Savvaris, "Lidar-inertial integration for uav localization and mapping in complex environments," in *2016 International Conference on Unmanned Aircraft Systems (ICUAS)*, IEEE, 2016, pp. 649–656.

[13] F. Pomerleau, F. Colas, and R. Siegwart, "A review of point cloud registration algorithms for mobile robotics," *Foundations and Trends in Robotics*, vol. 4, no. 1, pp. 1–104, 2015.

[14] M. C. Santos, L. V. Santana, M. M. Martins, A. S. Brandão, and M. Sarcinelli-Filho, "Estimating and controlling UAV position using RGB-D/IMU data fusion with decentralized information/Kalman filter," *Proceedings of the IEEE International Conference on Industrial Technology*, vol. 2015-June, no. June, pp. 232–239, 2015. DOI: 10.1109/ICIT.2015.7125104.

[15] R. Li, J. Liu, L. Zhang, and Y. Hang, "Lidar/mems imu integrated navigation (slam) method for a small uav in indoor environments," in *2014 DGON Inertial Sensors and Systems (ISS)*, 2014, pp. 1–15. DOI: 10.1109/InertialSensors.2014.7049479.

[16] S. Siebert and J. Teizer, "Mobile 3D mapping for surveying earthwork projects using an Unmanned Aerial Vehicle (UAV) system," *Automation in Construction*, vol. 41, pp. 1–14, 2014. DOI: 10.1016/j.autcon.2014.01.004. [Online]. Available: https://doi.org/10.1016/j.autcon.2014.01.004.

[17] D. Li, Q. Li, N. Cheng, Q. Wu, J. Song, and L. Tang, "Combined RGBD-inertial based state estimation for MAV in GPS-denied indoor environments," *2013 9th Asian Control Conference, ASCC 2013*, no. January, 2013. DOI: 10.1109/ASCC.2013.6606361.

[18] C. Luo, S. I. McClean, G. Parr, L. Teacy, and R. De Nardi, "Uav position estimation and collision avoidance using the extended kalman filter," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 6, pp. 2749–2762, 2013. DOI: 10.1109/TVT.2013.2243480.

[19] F. Pomerleau, F. Colas, R. Siegwart, and S. Magnenat, "Comparing icp variants on real-world data sets," *Autonomous Robots*, vol. 34, no. 3, pp. 133–148, 2013.

[20] L. R. García Carrillo, A. E. Dzul López, R. Lozano, and C. Pégard, "Combining stereo vision and inertial navigation system for a quad-rotor uav," *Journal of intelligent & robotic systems*, vol. 65, no. 1, pp. 373–387, 2012.

[21] H. G. de Marina, F. J. Pereda, J. M. Giron-Sierra, and F. Espinosa, "Uav attitude estimation using unscented kalman filter and triad," *IEEE Transactions on Industrial Electronics*, vol. 59, no. 11, pp. 4465–4474, 2012. DOI: 10.1109/TIE.2011.2163913.

[22] R. B. Rusu, "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments," *KI - Kunstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010. DOI: 10.1007/s13218-010-0059-6.

[23] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp.," in *Robotics: science and systems*, Seattle, WA, vol. 2, 2009, p. 435.

[24] K.-L. Low, "Linear least-squares optimization for point-to-plane icp surface registration," *Chapel Hill, University of North Carolina*, vol. 4, no. 10, pp. 1–3, 2004.

[25] S. Roumeliotis, G. Sukhatme, and G. Bekey, "Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, vol. 2, 1999, 1656–1663 vol.2. DOI: 10.1109/ROBOT.1999.772597.

[26] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter," 1995.

[27] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992. DOI: 10.1109/34.121791.

Chapter A

Appendix

The appendix contains a compressed folder containing the source code for the thesis, a folder with the latex project, and scripts for launching our ROS nodes. Our code uses an older version of API for transformers. It can be used on an older core or changed for a new API. Recorded rosbags and videos are available at http://mrs.felk.cvut.cz/koci2022thesis.