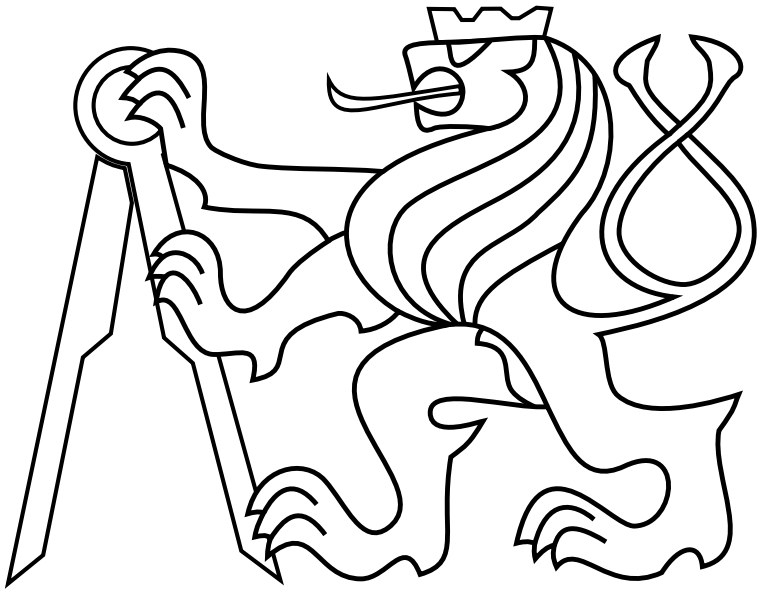
CZECH TECHNICAL UNIVERSITY IN PRAGUE

Faculty of Electrical Engineering

MASTER'S THESIS



Matěj Petrlík

Onboard Localization of an Unmanned Aerial Vehicle in an Unknown Environment

Department of Control Engineering

Thesis supervisor: **Dr. Martin Saska**

May 2018

Declaration of authorship

I hereby declare that I wrote the presented thesis on my own and that I cited all the used information sources in compliance with the Methodical instructions about the ethical principles for writing an academic thesis.

In Prague on ....

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Petrlík Matěj** Personal ID number: **406423**

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

Department / Institute: **Department of Control Engineering**

Study program: **Cybernetics and Robotics**

Branch of study: **Cybernetics and Robotics**

II. Master's thesis details

Master's thesis title in English:

Onboard Localization of an Unmanned Aerial Vehicle in an Unknown Environment

Master's thesis title in Czech:

Lokalizace bezpilotní helikoptéry v neznámém prostředí pomocí palubních senzorů

Guidelines:

The focus of this thesis is to design and implement a self-localizing system for unmanned aerial vehicles (UAVs) in an unknown environment without access to an external localization service (GPS). The developed method should process data from onboard sensors including a 2D laser scanner and an inertial measurement unit (IMU).
The following main tasks will be solved:
1. Implement a scan matching algorithm that finds the transformation between successive 2D laser scans.
2. Fuse the obtained transformation with data from the IMU and other sensors to provide an estimate of UAV local position.
3. Integrate the self-localizing technique into the existing UAV system used in the MRS group.
4. Conduct experiments for establishing the performance and limitations of the designed localization technique in the realistic Gazebo simulator.
5. Prepare the system for integration into the position feedback control loop of the UAV and test it in the Gazebo simulator (an online experiment with real system with the onboard position feedback is not planned, since it requires significant modification of the platforms, which goes beyond the content of this thesis).
6. Test the usability of the complete system on datasets recorded during real-world flights with a precise GPS ground truth (offline post-processing).

Bibliography / sources:

[1] Feng Lu and E. E. Milios, "Robot pose estimation in unknown environments by matching 2D range scans," 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Seattle, WA, 1994, pp. 935-938.
[2] M. Kam, Xiaoxun Zhu and P. Kalata, "Sensor fusion for mobile robot navigation," in Proceedings of the IEEE, vol. 85, no. 1, pp. 108-119, Jan. 1997.
[3] G. Loianno, V. Spurny, J. Thomas, T. Baca, D. Thakur, D. Hert, R. Penicka, T. Krajnik, A. Zhou, A. Cho, M. Saska, and V. Kumar, "Localization, Grasping, and Transportation of Magnetic Objects by a team of MAVs in Challenging Desert like Environments," Accepted in IEEE ICRA and RAL, 2018.

Name and workplace of master's thesis supervisor:

Ing. Martin Saska, Dr. rer. nat., Multi-robot Systems, FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **31.01.2018** Deadline for master's thesis submission: **25.05.2018**

Assignment valid until: **30.09.2019**

Ing. Martin Saska, Dr. rer. nat.	prof. Ing. Michael Šebek, DrSc.	prof. Ing. Pavel Ripka, CSc.
Supervisor's signature	Head of department's signature	Dean's signature

Acknowledgements

I would like to thank my family for their encouraging support throughout my studies. Further, I thank my supervisor Martin Saska for his guidance, and Vojtěch Spurný for his advice and valuable insights. My thanks also go to all Multi-robot Systems group members, who helped me immensely with this thesis.

Abstract

This thesis is concerned with onboard localization of an unmanned aerial vehicle without the access to global navigation satellite system services. The central focus of this work lies in design and implementation of simultaneous localization and mapping method that uses laser scans from a rotating laser rangefinder to estimate the position of the vehicle. A scan matching technique was implemented to estimate the displacement and rotation between two laser scans by aligning corresponding measurements from the two scans. The proposed solution involves fusion of position estimate from the inertial measurement unit, the relative displacement obtained by aligning successive laser scans, and the absolute position obtained by aligning laser scans into the gradually built map. The fused position estimate closes the outer feedback loop of the model predictive control. The developed system is first evaluated in simulations, and then its capabilities are demonstrated on a set of hardware experiments with a real drone.

Keywords: unmanned aerial vehicle, simultaneous localization and mapping, inertial measurement unit, LIDAR, laser scan, scan matching, iterative closes point, point cloud, estimation, Kalman filter.

Abstrakt

Tato práce se zabývá onboard lokalizací bezpilotního letounu bez přístupu k službám globálních navigačních systémů. Hlavní cíl této práce spočívá v návrhu a implementaci metody pro simultánní lokalizaci a mapování, která využívá laserové skeny z rotačního laserového dálkoměru k odhadování pozice letounu. Byla implementována technika pro odhadování posunutí a rotace mezi dvěma laserovými skeny pomocí zarovnání korespondujících měření ze zmíněných laserových skenů. Navržené řešení zahrnuje fúzi odhadu pozice z inercialní měřicí jednotky, relativní posunutí získané ze zarovnání po sobě jdoucích skenů a absolutní pozice získané ze zarovnání laserových skenů do postupně stavěné mapy. Fúzovaný odhad pozice uzavírá vnější zpětnovazební smyčku prediktivního řízení. Vyvinutý systém je nejprve posouzen v simulaci a poté jsou jeho schopnosti předvedeny na sadě hardwarových experimentů s reálným dronem.

Klíčová slova bezpilotní letoun, simultánní lokalizace a mapování, inerciální měřicí jednotka, LIDAR, laserový sken, zarovnání laserových skenů, iterativní nejbližší body, mrak bodů, odhadování, Kalmanův filtr.

Contents

1 Introduction **1**

1.1 UAV autonomy . . . 1

1.2 Preliminaries . . . 4

1.2.1 Mathematical notation . . . 4

1.2.2 Coordinate frames . . . 4

1.2.3 UAV pose . . . 6

1.3 Problem statement . . . 6

1.4 Related work . . . 8

1.4.1 Simultaneous localization and mapping . . . 8

1.4.2 Scan matching methods . . . 9

1.5 Contribution . . . 10

1.6 Outline . . . 11

2 UAV Platform **13**

2.1 Hardware . . . 13

2.2 Software . . . 14

2.3 Control pipeline . . . 15

2.4 GNSS localization . . . 16

3 System description **19**

3.1 Scan processing . . . 19

3.1.1 Close points removal . . . 19

3.1.2 Conversion from polar coordinates . . . 22

3.1.3 Scan attitude . . . 22

3.1.4 Ground removal . . . 23

3.1.5 Noise filtering . . . 23

3.1.6 Points outside flight area removal . . . 24

3.2 Sequential matching . . . 25

3.3 Global matching . . . 28

3.4 Mapping . . . 28

4 Scan Matching **31**
4.1 Establishing correspondences . . . 31
4.1.1 ICP, IMRP and IDC . . . 32
4.1.2 Outlier rejection . . . 33
4.2 Computing transformation . . . 34
4.3 Evaluation and iteration . . . 36

5 State Estimation **37**
5.1 Linear Kalman filter . . . 37
5.2 Altitude estimation . . . 38
5.3 Lateral estimation . . . 38

6 Validation in simulation **41**
6.1 Simulation setup . . . 41
6.2 Trajectory generation . . . 42
6.3 Goal of validation . . . 44
6.4 Simulation results . . . 44
6.5 Summary . . . 46

7 Experimental verification **49**
7.1 Mapping module analysis . . . 52
7.2 Localization verification . . . 53
7.3 SLAM experiment . . . 59

8 Conclusion **61**
8.1 Future work . . . 62

Appendix A CD Content **v**

Appendix B List of abbreviations **vi**

List of Figures

1 Examples of different UAV types . . . 2
2 Reference frames . . . 6
3 Hardware componentes . . . 14
4 Multi-layer control pipeline diagram . . . 16
5 Localization pipeline diagram . . . 20
6 Effect of UAV tilt on laser scans . . . 20
7 Scan processing pipeline diagram . . . 21
8 Optimal mounting position of the Sweep LIDAR . . . 21
9 Projection from polar to Cartesian coordinates . . . 22
10 Ground removal bounds . . . 24
11 Scan matching with ICP correspondences . . . 26
12 Scan matching with IMRP correspondences . . . 27
13 Global map downsampling . . . 29
14 Scan matching pipeline diagram . . . 32
15 Correspondence search at line segments . . . 33
16 ICP and IMRP correspondences . . . 34
17 Exponential convergence of scan matching . . . 36
18 Altitude fusion . . . 39
19 Gazebo simulator environment and UAV . . . 42
20 Trajectory planning interface . . . 43
21 Planned trajectory in time . . . 43
22 Planned and target velocity . . . 44
23 Position estimates . . . 45
24 Deviation of estimates from ground truth . . . 45
25 Aerial view of flight area in simulator . . . 46
26 Velocity estimates . . . 47
27 U-shaped room model . . . 50
28 Models of trees . . . 50
29 Top view of the experiment setup . . . 51
30 Number of points during scan processing . . . 53
31 Global map point cloud . . . 54
32 Dense point cloud from Stará Voda church . . . 55
33 Position estimates from localization experiment . . . 56

34 Detailed view of position estimates 57
35 Velocity estimates from the localization experiment 58
36 Detailed view of velocity estimates 58
37 Stitched image of UAV flight 59

1 Introduction

There has been a significant expansion of Unmanned Aerial Vehicles (UAV) in the past years. Although the term UAV incorporates different kinds of flying vehicles such as airplanes, balloons and helicopters, all of them can be operated without a pilot on board, most often we are talking about multi-rotors (quad-rotors, hexa-rotors). The reasons behind the rising popularity of this specific category of UAVs are many.

First, it is an amazingly flexible platform in terms of scalability – the smallest vehicles (often referred to as Micro Aerial Vehicles or MAVs) are not larger than a few centimeters whose typical purpose is to provide fun to the pilot, while the largest measuring several meters can carry hundreds of kilograms with the aim of parcel delivery, military use or passenger transportation.

Second, relatively simple construction with cheap and available parts make it an attractive option for RC enthusiasts and DIY community, who can build their flying device with commonly available tools and equipment.

Third, the versatility of use cases is immense – considering that UAVs are capable of hovering on one position, vertical takeoff and landing (VTOL) in constrained space and complex maneuvers in 6 Degrees of Freedom (DoF), the exact purpose of the system is defined mainly by the size, payload, and imagination. Examples of different UAV types are shown in Figure 1.

1.1 UAV autonomy

Traditionally, the UAVs are controlled by a human operator through remote radio control. The UAV is a highly unstable system, so the simplest MAVs require the operator to constantly provide action inputs, even when his goal is only to maintain a constant position of the MAV. Most of the commercially available systems offer various levels of autonomy to facilitate the task of the pilot. By equipping the UAV with a barometer and a downward facing laser rangefinder, the pilot can fly the vehicle at a constant altitude without the need of manual adjustment of thrust. This mode of operation is often referred to as the altitude mode. Adding a Global Position System (GPS) receiver to the platform provides the ability to hover at one position without any control input from the user as well as more advanced commands like flying a specified distance in a direction, remembering takeoff position and autonomous return. Operating a GPS-equipped UAV (flying in GPS mode) is relatively effortless and straightforward which allows amateur users to shoot photographs and videos with an onboard high-resolution camera without the need to spend many hours practicing flying the UAV. Most of the commercially available drones like the DJI Phantom, Mavic or Spark already offer these features for a few years.

Despite bringing many advantages, the GPS system is not perfect. The accuracy of position is around 2 meters in open spaces with good satellite visibility. While this accuracy is usually acceptable for typical commercial UAVs for photography, it is insufficient for precise maneuvers as flying through tight spaces, avoiding collisions with objects at known locations,

Figure 1: Examples of different UAV types. The photo at the top left corner shows a custom built UAV on the DJI F450 frame. The commercially available DJI Phantom 3 Professional is in the top right corner. The bottom left corner shows the DJI F550 frame equipped with hardware for RFID signal detection. In the bottom right corner is a photo of A250 racing drone frame equipped with WhyCon localization marker.

flying along a wall, manipulation with the environment or landing on a precise position. The accuracy can be improved by using Differential GPS (DGPS) which reaches up to 10 centimeters accuracy by including a reference base station which broadcasts the difference between GPS position and the known position of the base station. The difference is expected to be the same for the UAV flying in the relatively close vicinity of the base station due to the visibility of the same satellites which have the same clock errors and same ionospheric delay. Another possibility with an even higher accuracy of up to 1 centimeter is to use Real Time Kinematic (RTK) which obtains the position by measuring the phase of the carrier wave and calculating the number of whole carrier cycles with the help of the base station. Even though both GPS improvements provide a satisfactory accuracy, the price of these solutions is too high and the operation unreliable, which with the need of a fixed base station prevents wider adoption of this technology.

The most significant disadvantage of the GPS localization is that it works only in unobstructed outdoor areas. This disadvantage disqualifies GPS-localized UAVs from all indoor applications together with deployments for missions in obstructed environments, e.g., finding a missing person in a forest or analyzing the state of collapsed buildings in disaster areas. For the situations where GPS cannot provide sufficient accuracy or is impossible to use, the

UAV can be localized using Vicon motion capture system that uses infrared light source and multiple cameras capturing reflections of markers mounted on the UAV. Vicon provides high accuracy in the range of tens of millimeters [1], however, it is quite expensive, and the movement of the UAV is constrained to the space defined by the camera placement. Vicon is better suited to provide a ground-truth position in experiments that test other algorithms. A cheaper alternative called WhyCon [2, 3, 4, 5] exists. Instead of capturing reflections of infrared light, the localized object is equipped with a black and white circular pattern which is detected in a video stream from a regular camera. WhyCon can be used in both directions, i.e., the cameras are fixed in the environment, and the distance to the marker-equipped UAV is computed from the size and deformation of the marker, or the camera is mounted on the UAV which is flying through an environment containing the markers necessary for the UAV to localize itself. Aside from being a cheap localization solution requiring only a camera and a printed marker, it can also be used for relative localization of a UAV swarm [6]. In [7] the relative localization is solved by mounting UV LEDs on the arms of the UAV frame. The second drone is equipped with a camera with a UV-pass-through filter. Since the only sources of UV light in the camera images are likely to be the LEDs of the first UAV and the sun, detecting the UAV in the image is straightforward. An algorithm estimates the relative pose of the UAV based on the distances between individual LEDs in the image.

All of the mentioned localization systems need an external device, which is not very convenient or even possible in many cases. The GPS based methods rely on the signal transmitted from satellites. Additionally, DGPS and RTK require a base station. Vicon and WhyCon are both systems requiring an external camera and markers. If the goal is to deploy the UAV to an unknown area with no GPS coverage, then none of these localization methods can be used, and as the UAV has to rely exclusively on its onboard sensors. The most popular sensor choices for localization can be divided into two principal categories: passive cameras and active LIDARs. The camera-based solutions, collectively called visual odometry [8], estimate the displacement of the vehicle by examining the motion between consecutive images. LIDAR, which stands for Light Detection and Ranging, is a method for measuring the distance from the sensor to an object by emitting laser pulses and measuring the time until the pulse is reflected back to the sensor. The method along with the principles of time-of-flight range measurements, analysis of resolution and precision are presented in [9]. The devices, which inherited the name from the method, use near-infrared light typically and often have moving parts or more emitters, to generate scans consisting of multiple range measurements of the surroundings. Methods which are using LIDARs, match measurements from successive scans to find the transformation between them. These methods are thus labeled as scan matching methods [10].

The term visual odometry contains localization methods which have in common the processing of an image stream from an onboard camera to estimate the motion between individual images. One of the typical approaches to visual odometry, which can be seen even in commercially available hobby UAVs (DJI Phantom 3), uses a downward facing camera with high frame rate to provide a stream of images. The gradient of the brightness is used to construct the optical flow field [11], from which the camera motion is estimated. The gradient-based methods for estimating the optical flow are thoroughly described in the tutorial [12].

The camera is often combined with a sonar or laser range-finder to provide altitude

measurements since the optical flow estimates movement only in the plane perpendicular to the direction the camera is facing. Another commonly used approach is to mount a stereo camera (two cameras on a fixed baseline) in the direction of the flight. It is possible to calculate the depth information [13] from the shift of corresponding features between the left and right images taken by the stereo camera at the same time. The coordinates of each pixel in the depth image can be transformed to the metric position relative to the sensor, which is then typically stored in a three-dimensional point cloud data structure. The motion can be estimated either from the point cloud directly, or after further processing and feature extraction.

Vision-based methods have two considerable disadvantages: First, the requirements on the computational power are quite demanding, primarily when working with stereo cameras and point clouds. Because the onboard CPU has to perform several tasks in real time at once, e.g., control of the UAV, trajectory planning, obstacle detection, collision avoidance, it may not be suitable for some platforms with lower computational power. Second, the performance relies heavily on lighting conditions and properties of the camera sensor and lens. An algorithm may work flawlessly under overcast conditions, however, when noon sunlight is introduced to the scene, creating hard shadows and reflections, the dynamic range of the sensor cannot capture the high contrast of the scene. Consequently, the texture in highlights and shadows is lost, making it impossible to detect any features. On the other hand, LIDARs can work in all illumination conditions, indoor, outdoor, or night. LIDARs operating in the infra-red spectrum can have a slightly lower range in sunlight, nevertheless, it does not degrade the motion estimation. The scans are also easier to process without much strain on the CPU. Before matching of successive scans to estimate the movement, a preprocessing phase takes place, during which noise and points which are too close or too far to be part of the scan are removed. After that, correspondences between the points of two scans are found, so that a distance measure is minimized for each pair of points. With established correspondences, a transformation minimizing the sum of distances between each pair is computed. The found transformation is the estimate of the motion resulting in a change of the scans. The simplest of the scan matching algorithms is Iterative Closest Points (ICP) which uses Euclidean distance metric with closed-form least squares solution [14].

1.2 Preliminaries

Before we delve deeper into more technical sections, it is fitting to introduce conventions and mathematical notation used throughout the thesis.

1.2.1 Mathematical notation

The Table 1 outlines the basic mathematical notation used throughout this thesis.

1.2.2 Coordinate frames

Each sensor has its coordinate frame (frame) in which it provides measurements. The transformation between these frames is defined by a 4×4 affine transformation matrix which

Symbol	Description
lower of uppercase letter, e.g., n, N	a scalar
bold lowercase letter, e.g., $\mathbf{x}$	a column vector
bold uppercase letter, e.g., $\mathbf{Z}$	a matrix or set
$\mathbf{x}^T$, $\mathbf{Z}^T$	vector and matrix transpose
x_t, $\mathbf{x}_t$	x, $\mathbf{x}$ in time t
$\mathcal{E}\{\mathbf{x}\}$	mean of $\mathbf{x}$
$\mathbf{I}$, $\mathbf{0}$	identity matrix and zero matrix
$\mathbb{R}$	set of real numbers
$\mathcal{N}$	normal distribution

Table 1: Overview of the mathematical notation.

transforms a point $\mathbf{p} = (x, y, z)^T$ from one frame to another. The process of transformation is realized by multiplying the point in the homogeneous coordinates $\mathbf{p}^{(h)} = (x, y, z, 1)^T$ by the transformation matrix

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{1}$$

where a_{mn} are the affine transformation coefficients. An affine transformation is a combination of translation, rotation, scale and shear with the property of preserving parallelism of lines.

As mentioned before, the multiplication by transformation matrix transforms the points from one reference frame to another. Let us label all frames of reference relevant for this thesis and visualize them with corresponding basis vectors in Figure 2. When a sensor generates a measurement, it is attached to the frame of the sensor. In the case of the Sweep LIDAR, the frame in which the laser scans are generated is called *sweep_frame*. The common point to which all sensor frames should have a transformation is the Pixhawk autopilot (the flight control unit). This frame is located in the center of the UAV and is called *fcu_uav*. The UAV moves freely in the *local_origin* frame which has the origin fixed point in the area where the UAV is going to fly. The origin depends on the specific application. For instance, when the mission involves flying outdoor in an environment with GNSS reception, the origin is located at GPS coordinates which are averaged from multiple measurements before the flight. Without GPS reception, the absolute position of the frame origin cannot be established. Therefore it is usually located at the position of UAV takeoff. Please note that the transformation from *fcu_uav* to *local_origin* corresponds to the current position and orientation of the UAV in the *local_origin* frame. Consequently, the transformation from *fcu_uav* to *local_origin* changes in time, while the transformations from sensor frames to *fcu_uav* are constant in time as the sensor is rigidly attached to the UAV body and its position relative to the FCU does not change.

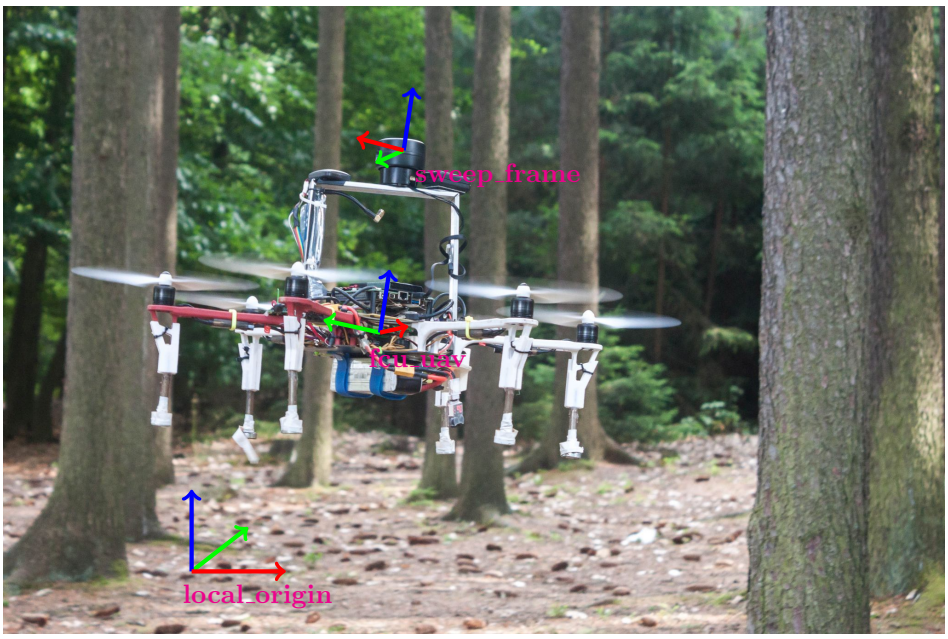


Figure 2: Photo of the UAV equipped with the Sweep LIDAR. The overlay shows the basis vectors of individual reference frames. The basis vectors are color-coded with red, green and blue for $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ respectively.

1.2.3 UAV pose

A flying UAV is moving in a 3-dimensional space and has 6 Degrees of Freedom (DoF). Its pose can be fully described by $\mathbf{x}_{uav} = (x, y, z, \psi, \theta, \phi)$, where x, y, z are the coordinates in the *local_origin* frame and ψ, θ, ϕ are the roll, pitch, yaw orientation angles, respectively.

1.3 Problem statement

The goal of this thesis is to design, implement and verify a self-localizing system for a UAV deployed in an unknown environment, using onboard sensors only. The proposed solution must be usable for both indoor and outdoor missions. Expected environments for indoor usage are old architectonic monuments such as churches, cathedrals, castles or ruins of buildings destroyed by a catastrophe. The area of interest for outdoor applications are autonomous flights in forests. Large open environments without any physical objects are not considered. On board the UAV, the following sensors are available: downward facing laser range-finder, rotating LIDAR, cameras with various mounting possibilities and IMU.

The crucial sensor in this work is the rotating LIDAR, Sweep v1 [15] which is an affordable ($250) and lightweight sensor (120 g) released in May 2016 by the Scanse company. The low weight and low price predetermine the sensor for integration to small UAV platforms with limited payload capacity. It is a time-of-flight near-infrared (905 nm wavelength) laser rangefinder (Garmin LIDAR-Lite v3) mounted on a rotating base with rotation frequency 2-10 Hz. Thanks to the rotating base it provides range measurements of 1000 points per second

over the full 360 deg horizontal field of view. The declared range is 40 m, however, in reality, it is rarely more than 10 meters, depending on the sunshine intensity and the reflectivity of the scanned surface. After every rotation, a two-dimensional scan with each point represented by an angle and distance from the sensor is transmitted over the USB to the onboard computer. It is possible to control the performance of the sensor by modifying the rotation speed. Slower speed means more samples per scan, while faster speed results in smaller displacement of the UAV between consecutive laser scans. The number of samples per second can be tuned as well – setting lower samples per second produces a more accurate reading of point distances.

The localization is defined as determining the global pose $\mathbf{x}_{uav}$ of the UAV. No single onboard sensor can accurately and reliably measure all the variables of the UAV pose, so it is necessary to fuse the measurements of multiple different sensors. The orientation is measured by the gyroscopes and magnetometers in the IMU, the position in z axis is determined by fusing measurements of a barometer and a downward-facing laser rangefinder. The position in x and y axes can be obtained after double integration of accelerometer measurements after subtracting the gravity. This approach is very imprecise due to the accumulation of an error s_t which grows quadratically with time when integrating the acceleration measurements suffering from a constant bias ϵ

$$s_t = \epsilon \frac{t^2}{2}. \tag{2}$$

so another approach to determining the x and y position must be found. Assuming the altitude z and orientation of the UAV is known, we can simplify the 3-dimensional localization problem to the well-known 2-dimensional localization problem.

Based on the research done in [16], the scan matching techniques were chosen for localization. According to the authors, scan matching in combination with Kalman filtering is more accurate than Markov localization when sufficient information is available from the sensor. The difference in accuracy was found to be up to an order of magnitude. Let us define the position of a robot moving on a horizontal plane as $\mathbf{x}_t = (x, y, \phi)$, where x and y are the Cartesian coordinates in the global coordinate frame, and ϕ is the heading of the robot. A laser scan is taken from a known position $\mathbf{x}_{t-1}$. The scan consists of multiple range measurements $\mathbf{Z}_t = [\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_n]$. Another scan is taken from the position $\mathbf{x}_t$ in the same way. The scan matching problem is defined as estimation of the rigid transform $\mathbf{T}$ between the known position $\mathbf{x}_{t-1}$ and the unknown position $\mathbf{x}_t$ based on the alignment of scans $\mathbf{Z}_{t-1}$ and $\mathbf{z}_t$. A 2-dimensional rigid transform consists of translation in x and y axes and rotation θ:

$$\mathbf{T} = \begin{bmatrix} \cos\theta & -\sin\theta & t_x \\ \sin\theta & \cos\theta & t_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3}$$

Generally, the scan matching algorithms try to find a transform $\mathbf{T}$ which projects the scan $\mathbf{Z}_t$ to the reference frame of scan $\mathbf{Z}_{t-1}$ so that the difference between corresponding measurements is minimized. The survey of widely used scan matching techniques is presented in Subsection 1.4.

The proposed system must be compatible with the UAV platform developed in the Multi-robot Systems group. The UAV was explicitly built for the Mohamed Bin Zayed International Robotic Challenge (MBZIRC) held in Abu Dhabi in March 2017. All the challenges

of the competition took place outdoors in an obstacle-free area with reliable GPS reception. A trajectory tracking solution based on model predictive control (MPC) [17] was implemented, thoroughly tested and experimentally verified in various environments and use cases. The method fuses IMU and GPS measurements to estimate the UAV position and then predict the control inputs using the MPC. This thesis aims to allow using the same approach for indoor and GPS-denied outdoor areas by replacing the GPS measurements with the position estimate obtained from matching consecutive laser scans.

1.4 Related work

1.4.1 Simultaneous localization and mapping

In [18] the authors present autonomous localization and mapping in GPS denied environments for UAVs. They use IMU and LIDAR measurements to estimate the position by matching consecutive scans while simultaneously building a map. The current laser scan is matched to the previous one by a customized ICP algorithm to obtain the current position of the UAV. The ICP features an autonomous failure detection based on thresholding of the ICP error metric. When a failure is detected, the IMU accelerations are integrated to update the current position instead. For building the map, Principal Component Analysis (PCA) was used. The method analyzes the points in laser scan to fit line segments to them. First, clusters are found in the data, and covariance matrices of points inside each cluster are established. Second, the eigenvectors and eigenvalues are found by spectral decomposition of the covariance matrix. Then the ratio between maximum and minimum eigenvalue is compared to a threshold, and when the ratio is larger than the threshold, the corresponding cluster is considered to be a line feature with the direction of the principal eigenvector. Although the proposed method has a similar motivation as the system proposed in this thesis, it has a few shortcomings compared to our approach. The gradually built map is not used for localization - the mapping and localization subsystems run separately. As a consequence, the localization is sequential and thus prone to drift due to small errors in each ICP iteration integrating into a significant position error over time. The absolute localization in global map proposed in our method corrects this drift to provide a reliable, robust and globally consistent position estimate. Further, the mapping module based on PCA as stated in the manuscript is intended to be used only in environments with line features. The system in this thesis, on the other hand, is expected to work in general environments without any assumptions about the geometry of the environment. Finally, the experimental verification of the algorithm is done by carrying the devices through a short maze in hand after which the localization and mapping is done offline which does not guarantee the suitability for use with a real UAV platform. In contrast to [18], we present a clear demonstration of applicability to our platform in Section 7. Our solution in addition to the tasks solved in this paper contains the fusion of position obtained from localization in a global map, velocity from the displacement of sequential scans and attitude measurements from UAV gyroscopes. The fusion realized by the Linear Kalman Filter with identified model provides estimates of states which are appropriate for closing the feedback loop of MPC tracker.

A comprehensive control, localization, navigation, and mapping solution for indoor UAV

is introduced in [19]. The controller accepts state estimates from a linear Kalman filter which is fusing IMU accelerations with velocities obtained by computing the optical flow of a downward facing camera. The position estimate can, however, drift over time, since no measurement of absolute position is fused in the Kalman filter. The authors elegantly suppress this issue by implementing reactive path planning which takes a laser scan from LIDAR, creates a potential field with repulsive forces originating in all of the scanned points and an attractive force 2 m ahead of the UAV. Thus, even when the absolute position estimate drifts from the real position, the UAV flies in the correct direction which is relative to the UAV position in the current laser scan. Despite no need for a global map in their reactive solution, the authors present a FastSLAM algorithm in the second part of the work. The FastSLAM algorithm, which is introduced in [20], is based on particle filter, where each particle contains a covariance matrix of map features and UAV pose. It is faster than the tradition EKF SLAM thanks to less computationally demanding matrix inversion of many small covariance matrices instead of one large EKF matrix. Still, the algorithm in [19] was run offline on a desktop computer with the dataset obtained during the flight. We propose to connect localization, navigation, control, and mapping into one pipeline running online onboard of the UAV. Our approach has the benefit of serving as a low-level no-maintenance layer for higher level algorithms for which it supplies the globally consistent map and the absolute position.

In the manuscript [21] a SLAM solution aimed at autonomous navigation in forests is presented. The authors use an approach similar to our system. The laser scans from LIDAR are processed, clustering is applied, and clusters that are validated by a series of geometric descriptors are classified as features corresponding to trees. These features are matched between successive scans to obtain a translation measurement, which is then fused with IMU estimate in a Kalman filter. The fused position estimate tends to drift so a back-end GraphSLAM [22] algorithm is implemented to detect loop closures with sufficient overlap between the current feature set and the previously visited feature sets. The proposed method is backed by convincing experiments with both simulated dataset, and a real UAV platform. Nevertheless, the geometric-based feature detector restricts the use for environments with tree-like features only, hence the technique cannot be used for a UAV deployed in an unknown environment.

1.4.2 Scan matching methods

A thorough research of scan matching methods was done to find a suitable solution for estimating displacement between two laser scans. The final system uses a custom combined technique that is inspired by ideas in multiple manuscripts, which are described in Section 4.

In [23] the authors present a probabilistically-motivated scan-matching algorithm that produces higher quality and more robust results at the cost of additional computational time. However, they also present a multi-resolution implementation for conventional CPUs which achieves real-time performance as well as a parallelized approach for GPUs. Their goal is to find the posterior distribution over the robot's position, $p(x_i|x_{i-1}, u, m, z)$ and the central contribution is a method for efficient computing of the distribution $p(z|x_i, m)$ (the observation model) which is then used to compute the posterior. Although the early implementations of our solution used this method, it was later found to be too slow for real-time localization even with the multi-resolution implementation. Moreover, the precision of

the algorithm is dependent on the resolution of the grid, so the localization result would always be a compromise between precision and run-time.

The authors of [24] propose to represent range scans by the Normal Distribution Transform. The 2D plane is subdivided into a grid, where a normal distribution is assigned to each of the cells based on the points contained within the cell. The discrete set of points is thus transformed into a differentiable probability density function. Matching a scan to the grid is then defined as maximizing the sum that the scan points score on the density function:

$$\text{score}(\mathbf{p}) = \sum_i \exp -\frac{(\mathbf{x}-\mathbf{q})^T \mathbf{\Sigma}^{-1} (\mathbf{x}-\mathbf{q})}{2}, \tag{4}$$

where $\mathbf{p}$ is the vector of parameters to estimate, $\mathbf{x}$ is the 2D point, $\mathbf{q}$ is the mean, and Σ is the covariance matrix. The solution to the optimization problem is found by the Newton's algorithm. Examples of relative position tracking and SLAM applications in indoor environment are demonstrated in the manuscript. The algorithm was not tested in outdoor environment, in which the obstacles are usually of different type than indoor.

1.5 Contribution

In this thesis, the whole localization and mapping pipeline was developed. The pipeline starts with a laser scan, generated by a LIDAR, which is preprocessed to get rid of unwanted measurements, noise, and outliers. A complex scan matching algorithm that combines multiple ideas from state-of-the-art methods estimates the velocity of the UAV from the displacement of subsequent scans. Furthermore, the method builds a global map in which the scans are matched to estimate the absolute position of the UAV. Both estimates are then fused in the Kalman filter to provide a robust and reliable pose estimate which closes the outer feedback loop.

Two primary contributions should be noted. First, in contrast to the other state-of-the-art methods, the developed system runs online onboard and provides real-time estimates which are fed back to the controller to stabilize the UAV. From the research of related work, it was found that using a LIDAR for localization of UAVs is quite common. However, the solutions are never complete. Either the UAV is localized by matching subsequent scans which naturally leads to a position drift over time since the localization is relative to the previous pose. When the LIDAR is used for mapping, the localization is done by another method such as optical flow or GNSS. Another problem is the lacking generalization of the techniques, as geometric assumptions about the environment are often made which prevents deployment to truly unknown environments. Our solution, on the other hand, is designed to work in an arbitrary environment with obstacles and provide a black-box no-maintenance solution allowing other systems to be built on top. The LIDAR-localized UAV thus provides the same interface as the GNSS-localized one, accepts the same commands and behaves in the same way.

Second, the system is deployed to a real UAV and experimentally verified. It is common that when a new algorithm is introduced or a state-of-the-art algorithm is modified, the verification is done only in simulation or by a hand-carried or cable-suspended platform.

Similar practices are insufficient to prove the suitability to a real UAV platform. Although the SITL simulations are realistic, not all phenomena can be modeled (unexpected events and faults), some effects are neglected (wind, uneven terrain, etc.), therefore without the experimental evaluation it is not clear whether the algorithms are ready to be used on a real UAV. This is not the case of the work presented in this thesis. The development was followed by the practical deployment on an existing UAV platform, and after the system was finished, it was thoroughly tested in an artificially built environment and proven to be reliable even during not ideal conditions.

1.6 Outline

The rest of the thesis is outlined as follows. First, the UAV platform, for which the localization and mapping system is developed, is introduced in Section 2. Second, the individual components forming the localization pipeline are described in Section 3. A detailed explanation of the scan matching technique resides in Section 4 followed by Section 5 that describes the fusion of position estimates. Then the validation analysis results of the developed system from the realistic Gazebo simulator are presented in Section 6. After simulation, the whole pipeline was deployed to the real UAV and tested in an artificially built environment. The real world experiment is evaluated in Section 7. The thesis is concluded by a discussion of the results and achieved goals in Section 8.

2 UAV Platform

In this section, we describe the UAV platform for which the onboard localization system was developed. Both hardware and software aspects of the platform are presented, followed a by a brief overview of the control pipeline and GNSS localization system.

The UAV platform development started in 2016 when the MRS group registered to compete in the MBZIRC 2017 competition. It was built to match the specific requirements of the competition, i.e., the maximum size of the UAV, the capability to grasp and carry a payload, ability to land on a moving vehicle. However, the future of the platform was also considered so that it could be reused for other research projects of the group outside the scope of the MBZIRC competition.

2.1 Hardware

Most of the components are commercially available, and those which are not (mounts, holders, legs) can be printed on a 3D printer. Each UAV is built on a DJI hexacopter F550 frame equipped with E310 DJI motors [25] which provides enough thrust for carrying sensors, landing gear, gripper and additional application-specific payload. The low-level control is provided by a PixHawk Flight Control Unit (FCU) [26] which has a built-in IMU consisting of accelerometers, gyroscopes, magnetometers that are fused in an estimate of the UAV state for attitude stabilization. Additionally, a GPS module connected to the FCU facilitates manual control of the UAV outdoors by enabling flights in GPS mode. The altitude is estimated by fusing sensor data from an accelerometer, barometer and TeraRanger One Time-of-Flight IR LED rangefinder. Recently the TeraRanger One was substituted by a Garmin LIDAR-Lite v3 which provides measurements up to 40 m improving the 14 m range of TeraRanger One by using IR laser instead of IR LED technology. All onboard computations are performed by a powerful Intel NUC computer featuring an Intel i7 CPU, 8 GB of RAM, SSD, a WiFi module and USB 3.0 ports, all tightly packed in a compact PC. These are the components that are vital to the basic autonomous functionality of the drones and as such are present on the UAV at all times.

Every application and experiment requires additional equipment which is not necessary for the basic functionality but is important for the particular use case. Depending on the usage frequency, cost, size, and weight of the gear, it can be present on most of the flights (cameras) or removed after the specific experiment (magnetic gripper). The UAV can be quickly modified to a specific need thanks to the modularity of the platform. During every outdoor flight, there is an RTK GPS receiver (PRECIS-BX305 GNSS RTK BOARD [27]) mounted on an aluminum bridge on top of the UAV. The RTK uses phase measurements to provide 10 mm position accuracy which is a significant improvement from a few meters accuracy of regular GPS. So far most use cases are outdoors, so the receiver is present in almost all flights. However, the mounting location on top of the UAV is convenient for sensors requiring unobscured view such as a rotating LIDAR that replaces the receiver for flights in GPS-denied environments. For vision-based applications, two types of monocular cameras with changeable lenses are used. The first one is a fast global shutter Matrix-vision mvBlueFOX-MLC200w [28] camera with

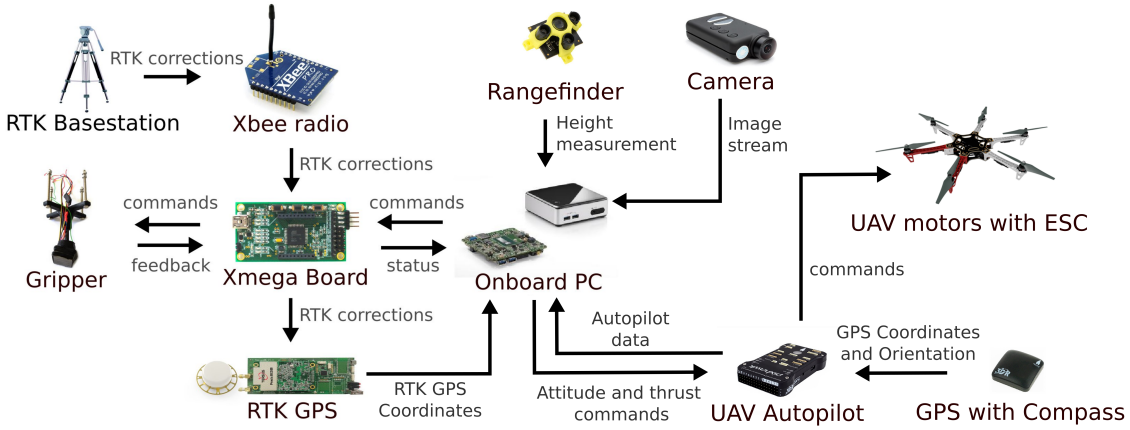


Figure 3: The hardware components and their connections which were used in the MBZIRC 2017 competition.

SuperFisheye lens used for agile vision-based maneuvers such as landing on a fast-moving vehicle [29]. The second one is a high-resolution rolling shutter Mobius ActionCam camera which was used for color object detection in [30], for relative localization using black and white circular patterns presented in [31] and for documentation of large historic buildings in [32]. In addition to these two monocular cameras, Intel Realsense R200 IR stereo camera [33], ZED and Tara stereo cameras were used to obtain datasets for depth perception. Safe landing on the ground is assured by a set of 3D-printed legs assembled with metal sticks into a landing gear with modifiable height. When landing on a moving vehicle, where a fall of the UAV due to slippage is possible, strong magnets are added to ensure reliable hold. Grasping and carrying of metal objects is possible after mounting a custom-built magnetic gripper [34] based on OpenGrab EPM v3 electro-permanent magnet [35]. The hardware configuration for the task of autonomous collective object grasping in MBZIRC 2017 is shown in Figure 3. The reader can find additional information about the platform in [36].

2.2 Software

On the software side of the platform, the Ubuntu 16.04 operating system is running together with the Robotic Operating System (ROS) middle-ware [37] which provides a framework for developing complex systems by dividing them in individual modules (ROS nodes) with asynchronous communication between them using the publish-subscribe messaging pattern. ROS supports nodes written in C++ and Python programming languages and thanks to its extensive community, many useful tools for visualizations, logging and debugging already exist as well as various libraries for working with point clouds, transformations or maps. Moreover when a new sensor is introduced to the market, either its developers directly provide a ROS interface, or it is written by the community shortly after the release which allows painless integration of new sensors to the platform. Every new node is thoroughly tested in a realistic Simulation In The Loop (SITL) Gazebo simulator before it is deployed to the real system. This prevents unnecessary crashes of the expensive UAV due to untested programs

and bugs. Furthermore, it accelerates the development of new subsystems, whose functionality can be evaluated quickly and without any risks. The subsystem proved in the simulation can be safely deployed to the real UAV for a hardware experiment which does not have to be repeated many times, since a lot of potential flaws were already discovered in the simulation.

2.3 Control pipeline

The control system of the UAV follows a multi-layer structure depicted in Figure 4. The control pipeline is thoroughly described in [38] and [39]. This thesis does not involve modifying the control more than changing the gains and other parameters, so only a brief description follows.

On the lowest level is the control of the speeds of individual motors realized by the Electronic Speed Controllers (ESC) of the DJI E310 motors. The motor speed reference is set by the attitude controller. The attitude controller is part of the PX4 stack [40] running on the PixHawk Flight Controller. It is responsible for maintaining the desired attitude $\mathbf{R}(\phi, \theta, \psi) \in SO(3)$, where ϕ, θ, ψ are the Euler angles corresponding to yaw, pitch and roll motions of the UAV. The attitude controller outputs motor speeds required to maintain the desired orientation $\mathbf{R}_D$.

The orientation commands are governed by the non-linear SO(3) state feedback controller based on the work in [41] and [42]. The controller uses the model

$$\begin{aligned} \dot{\mathbf{r}} &= \mathbf{v} \\ m\dot{\mathbf{v}} &= f\mathbf{R}\mathbf{e}_z + mg\mathbf{e}_z \\ \dot{\mathbf{R}} &= \mathbf{R}\hat{\mathbf{\Omega}} \\ \mathbf{M} &= \mathbf{J}\dot{\mathbf{\Omega}} + \mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega}, \end{aligned} \tag{5}$$

where $\mathbf{r} = [x, y, z]^T$ is the position and $\mathbf{R}(\phi, \theta, \psi)$ the orientation of the UAV in the world coordinate frame. The gravitational force g is acting on the vehicle which exerts total thrust force f. The mass of the UAV is m, the angular velocity in the body frame $\mathbf{\Omega}$ and the inertia matrix $\mathbf{J}$. The operator *hat map* $\hat{\cdot} : \mathbb{R}^3 \to SO(3)$ is defined by the condition $\hat{x}y = x \times y$ for all $x, y \in \mathbb{R}^3$. The total moment exerted by the propellers onto the UAV is $\mathbf{M} = [M_1, M_2, M_3]$. The control inputs $f \in \mathbb{R}$ and $\mathbf{M} \in \mathbb{R}^3$ are chosen as

$$\mathbf{M} = -k_R\mathbf{e}_R - k_\Omega\mathbf{e}_\Omega + \mathbf{\Omega} \times \mathbf{J}\mathbf{\Omega} - \mathbf{J}\left(\hat{\mathbf{\Omega}}\mathbf{R}^T\mathbf{R}_c\mathbf{\Omega}_c - \mathbf{R}^T\mathbf{R}_c\dot{\mathbf{\Omega}}_c\right) \tag{6}$$

$$f = -(-k_x\mathbf{e}_r - k_{ib}\mathbf{R}\int_0^t R(\tau)^T\mathbf{e}_r d\tau - k_{iw}\int_0^t \mathbf{e}_r d\tau - k_v\mathbf{e}_v - mg\mathbf{e}_3 + m\ddot{\mathbf{x}}_d) \cdot \mathbf{R}\mathbf{e}_3 \tag{7}$$

with $\ddot{\mathbf{x}}_d$ the desired acceleration. The control errors in rotation, position and velocity are denoted by $\mathbf{e}_R$, $\mathbf{e}_r$ and $\mathbf{e}_v$, respectively. The resulting control action consists of f and $\mathbf{R}_C$, which is the orientation command.

The highest level control block is realized by the MPC tracker. It utilizes a linear MPC

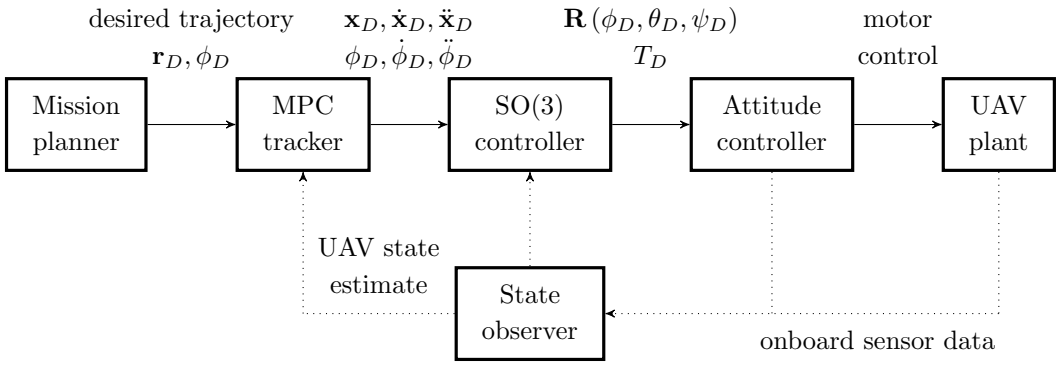


Figure 4: Schematic of the multi-layer control pipeline. The reference trajectory $\mathbf{r}_D, \phi_D$ serves as a setpoint for the MPC in the *MPC tracker*, which outputs a real time command $\mathbf{x}_D, \dot{\mathbf{x}}_D, \ddot{\mathbf{x}}_D$, $\phi_D, \dot{\phi}_D, \ddot{\phi}_D$ for the non-linear $SO(3)$ controller. The non-linear controller produces orientation and thrust reference for the embedded attitude controller.

with LTI model with n states and k inputs defined as

$$\begin{aligned} \mathbf{x}_{t+1} &= \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t \\ \mathbf{y}_t &= \mathbf{C}\mathbf{x}_t + \mathbf{D}\mathbf{u}_t, \end{aligned} \tag{8}$$

where $\mathbf{x} \in \mathbb{R}^n$ denotes the state vector and $\mathbf{u} \in \mathbb{R}^k$ the input vector. Matrices $\mathbf{A} \in \mathbb{R}^{n \times n}$ and $\mathbf{B} \in \mathbb{R}^{n \times k}$ are the main system matrix and the input matrix, respectively. In contrast to the traditional full description of the LTI system, we assume $\mathbf{C} = \mathbf{I}$, and $\mathbf{D} = \mathbf{0}$. Therefore $\mathbf{y}_t = \mathbf{x}_t$ holds.

Finding the control signals by an MPC requires to repeatedly solve the optimization problem

$$\min_{\mathbf{u}_t, \mathbf{x}_t} \mathrm{V}(\mathbf{x}, \mathbf{u}) = \frac{1}{2} \sum_{i=1}^{m-1} \left(\mathbf{e}_i^T \mathbf{Q} \mathbf{e}_i + \mathbf{u}_i^T \mathbf{P} \mathbf{u}_i \right) \tag{9}$$

$$\begin{aligned} \text{s.t. } \mathbf{x}_{t+1} &= \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t, && \forall t \in \{0, \dots, m-1\} \\ \mathbf{x}_t &\leq \mathbf{x\_max}_t, && \forall t \in \{1, \dots, m\} \\ \mathbf{x}_t &\geq \mathbf{x\_min}_t, && \forall t \in \{1, \dots, m\}, \end{aligned} \tag{10}$$

where the quadratic cost function penalizes the control error and the input action over a horizon of length $m \in \mathbb{R}$. Penalization matrices $\mathbf{Q}$ and $\mathbf{P}$ are positive semi-definite. The constraints force the states to follow the model and bound the states to a box to limit the maximum acceleration and velocity.

2.4 GNSS localization

The knowledge of precise UAV position is necessary for all tasks including following trajectories, avoiding obstacles, flying in close formations, interactions with the environment.

Although the position obtained from the PixHawk FCU, which uses an Extended Kalman Filter to fuse measurements from the IMU and GPS, is sufficient to allow autonomous flight, it is not accurate enough to enable precise maneuvers and interactions.

A localization system based on RTK was developed for high-precision outdoor experiments. The system consists of two RTK receivers. One is mounted on the UAV and the second one is mounted on a fixed-location tripod which serves as the base station. The base station broadcasts its position together with the carrier measurements for all visible satellites. With this information, the UAV can resolve the carrier phase ambiguity (the unknown number of whole carrier wavelength cycles) and determine its location relative to the base station with high precision. If the carrier phase ambiguity solution is exact (the result is an integer number of wavelength cycles), the highest precision of 1 cm is achieved, and the system is said to be working in RTK FIX mode. Even when obtaining the exact solution is not possible, it is still possible to obtain the number of cycles which is not whole (it has digits after decimal point) leading to a worse precision that is still more precise than regular GNSS. This lower precision mode of operation is called RTK FLOAT.

3 System description

This section describes the structure and functionality of the proposed solution. The whole system is divided into smaller modules, for higher modularity and easier maintenance, forming a pipeline illustrated in Figure 5. The individual modules include scan processing, sequential and global matching, and mapping.

An accurate and precise position estimate is needed to navigate the UAV in a tight environment containing obstacles, otherwise, the UAV can easily collide with them. In this thesis, the scan matching algorithms have been used to provide such estimate by using the laser scans obtained from the Sweep LIDAR. However, the raw data from the Sweep sensor cannot be used for scan matching directly, because it contains noise, false detections, laser beams hitting the frame of the drone, etc. Moreover, the Sweep LIDAR is a 2D sensor that is in our case mounted on a robot with 6 DoF moving in a 3D environment. As a result, the raw 2D laser scan without compensating the UAV tilt often does not represent the surroundings of the UAV accurately when placed in the 3D space. The deformation of distances between the UAV and the scanned obstacles when the UAV is tilted as shown in Figure 6. Thus additional steps are needed to process each scan before scan matching so that it faithfully represents the surroundings.

The position cannot be reliably estimated directly from relative displacement of the drone between two successive scans, due to a drift accumulated from small errors in each scan matching. Filtering and fusion with the position in the global map and other measurements are needed to produce a smooth position estimate without drift.

3.1 Scan processing

The scan data has to be preprocessed to obtain a more accurate representation of the environment without noise and false detections before it can be used to estimate relative displacement or global position. The sequence of operations performed on the point cloud is shown in Figure 7. First, points that are too close to the sensor are removed, to get rid of points which correspond to the frame of the drone. Second, the scan composed of pairs of angles and ranges is converted into 2D Cartesian coordinates, followed by projection into the 3D space using the orientation and altitude of the UAV. After that, the points which are projected below a predefined altitude are considered as detections of the ground which have to be removed. Finally, the noise and the points which are outside of a predefined area of operation are filtered out. They do not provide any useful information and lead to incorrect alignment of the scans in the matching modules. Therefore these points are removed.

3.1.1 Close points removal

Depending on the mounting position of the LIDAR sensor on the drone, some laser beams might hit the frame of the drone, the propellers, sensors, cables or any other part of the drone. The optimal mounting position is in our case on top of the UAV which minimizes

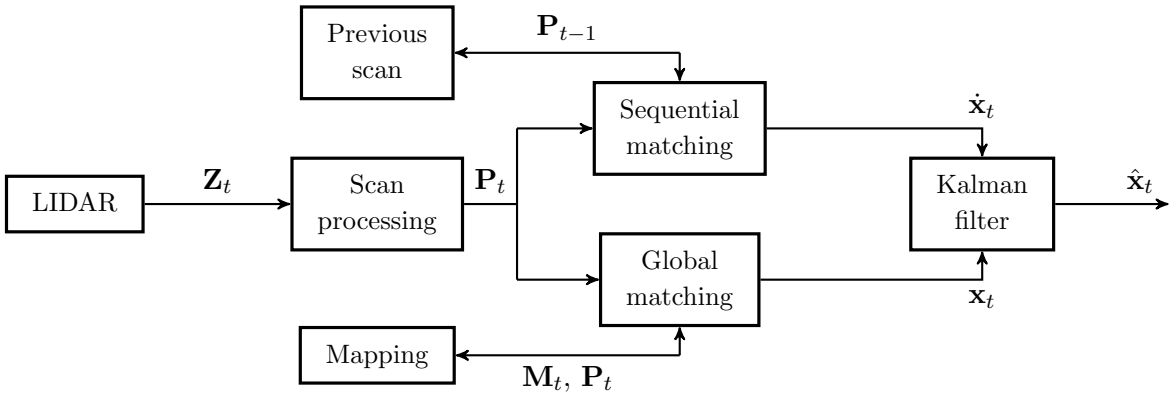


Figure 5: Diagram showing the localization pipeline. The process begins when a laser scan $\mathbf{Z}_t$ is produced by the LIDAR. The scan is processed and converted into a point cloud $\mathbf{P}_t$ in the Scan processing block. Then the processed point cloud $\mathbf{P}_t$ is aligned to the previous point cloud $\mathbf{P}_{t-1}$ in the Sequential matching subsystem to obtain the velocity estimate $\dot{\mathbf{x}}_t$. Simultaneously, $\mathbf{P}_t$ is also aligned by the Global matching to the global map, which is gradually build in the Mapping module, to obtain the global position estimate $\mathbf{x}_t$. Both estimates $\mathbf{x}_t$ and $\dot{\mathbf{x}}_t$ are fused in the Kalman filter into the final estimate $\hat{\mathbf{x}}_t$.

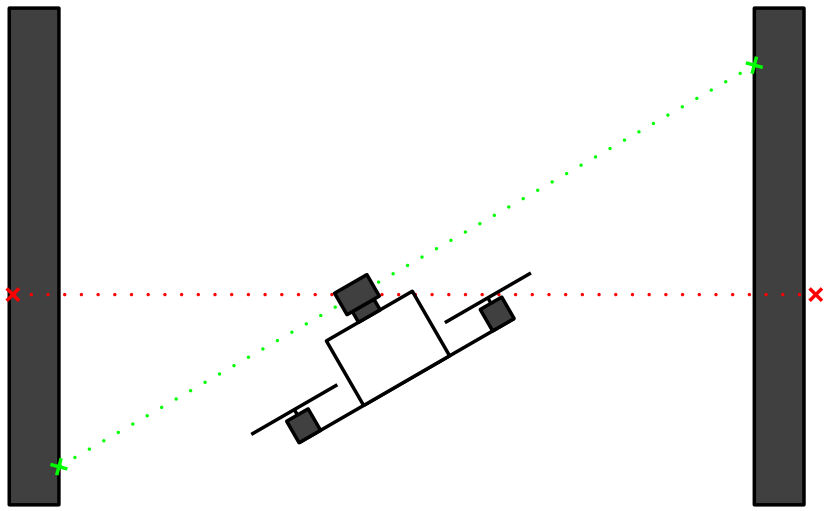


Figure 6: An UAV equipped with Sweep LIDAR flying between two walls. When the UAV is tilted, the laser beams, represented by green dotted lines, hit the walls and produce 2D measurements (green crosses) lying on the plane perpendicular to the axis of rotation of the LIDAR. Not considering the UAV tilt when placing the 2D scan into 3D space results in the incorrect position of the points (red crosses).

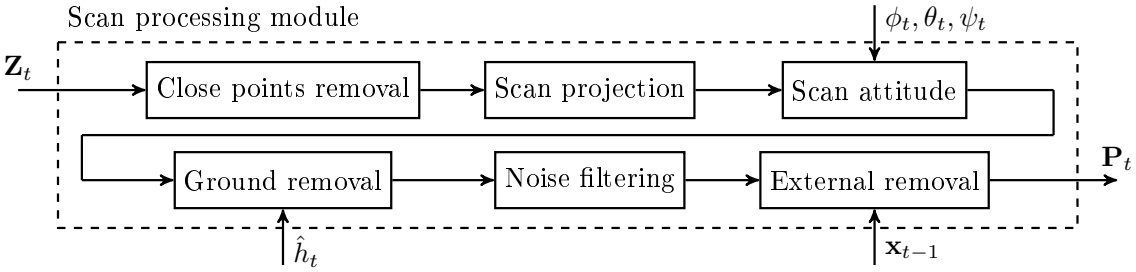


Figure 7: The individual submodules of the scan processing pipeline, with the raw laser scan $\mathbf{Z}_t$ on the input and processed point cloud $\mathbf{P}_t$ on the output. The scan attitude module had to be supplied by UAV roll, pitch, roll angles (ϕ, θ, ψ), the ground removal module needs the altitude estimate $\hat{h}_t$, and the external removal module requires the last known position $\mathbf{x}_{t-1}$.

the number of such false measurements that might worsen the displacement estimated by scan alignment (see Figure 8). However, it is not always possible to mount the sensor at the optimal place, due to other sensors already occupying the spot (RTK antenna), so a simple filter is applied to remove the measurements which are potentially part of the drone. Moreover, the Sweep LIDAR reports a measurement of range 0, when the laser beam at the specific angle did not hit any obstacle. Those measurements are removed too.

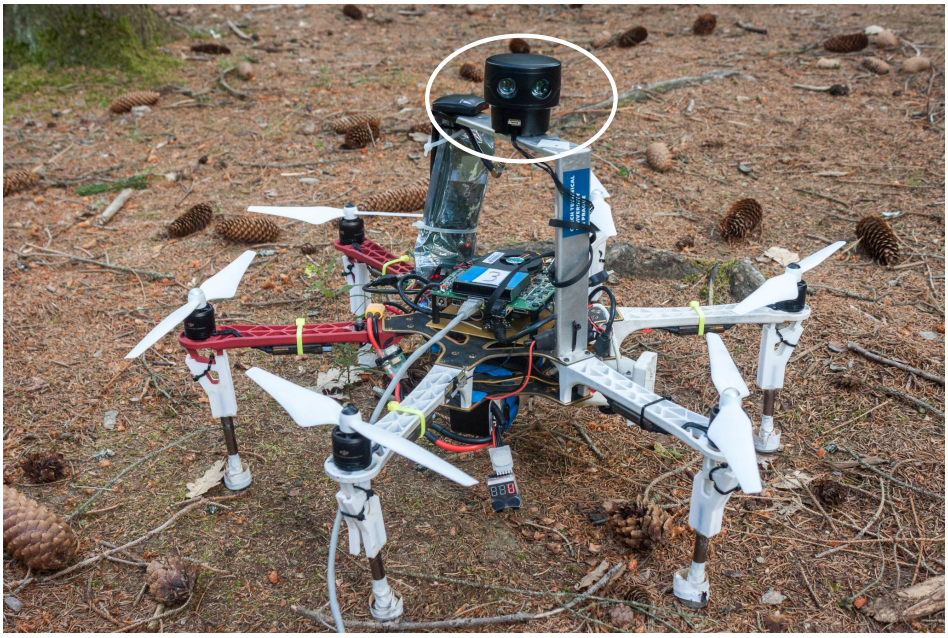


Figure 8: The Sweep LIDAR sensor mounted at the optimal position on top of the UAV.

The filter is applied in the following way. The range r of each measurement $\mathbf{z}_i$ is compared to the radius of the UAV (R_{uav}), and measurements that do not satisfy the inequality $r > R_{uav}$ are discarded. In the case of the DJI F550 frame $R_{uav} = 0.395$ m since the total diameter of the body frame is 0.55 m plus the radius of two propellers, which is 0.12 m for each. Thus only measurements that come from the environment and not from the UAV frame

continue further down the pipeline.

3.1.2 Conversion from polar coordinates

Each laser measurement $\mathbf{z}_i = (r_i, \varphi_i)$ from laser scan $\mathbf{Z}_t$ is projected into a point $\mathbf{p}_i$ in Cartesian coordinates lying on the sensor plane. All projected measurements from $\mathbf{Z}_t$ then form a point cloud $\mathbf{P}_t$. The position of a point $\mathbf{p}_i = (p_x, p_y, p_z)^T$ on the sensor plane is obtained as

$$\begin{aligned} p_x &= r_i \cos \varphi_i, \\ p_y &= r_i \sin \varphi_i, \\ p_z &= 0, \end{aligned} \tag{11}$$

where the origin of the Cartesian coordinate system $\mathcal{O}(0, 0)$ is located in the center of the sensor and the x-axis corresponds to angle $\varphi = 0$. The projection is shown in Figure 9.

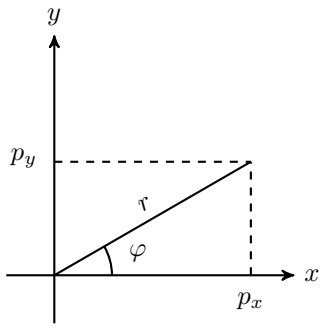


Figure 9: The projection of a point from polar to Cartesian coordinates in two dimensions.

3.1.3 Scan attitude

The 2D laser measurements have to be placed into the 3D space so that scans that were taken with different attitudes and altitudes can be matched correctly. The UAV attitude is represented by a unit quaternion $\mathbf{q}_{uav}$ that is obtained from the gyroscopes and compass of the IMU. Every point $\mathbf{p}_i \in \mathbf{P}_t$ is rotated by the UAV attitude. A translation in the z-axis of *local_origin* follows by adding the altitude of the UAV $\hat{h}_{uav}$. The UAV is equipped with 2 sensors measuring the altitude: the barometer and the laser rangefinder. An altitude measurement from the barometer often has an offset from changing air pressure due to the weather while the rangefinder is quite noisy and is sensitive to the ground relief below it. Therefore, the measurements of either sensor alone cannot be used. Fusing barometer and laser rangefinder in a Linear Kalman filter (LKF) as described in Subsection 5.2 provides an estimate $\hat{h}_{uav}$ which suppresses both issues.

3.1.4 Ground removal

When the UAV is tilted, the LIDAR can capture the ground plane which appears as a curved line in the laser scan. This happens especially during takeoffs or during flying in lower altitudes as even a small tilt results in projection of the beam on the ground. Since the number of points lying on the line representing the ground is often higher than the number of points representing the obstacles in the environment, the scan matching process prefers the ground points, and the useful obstacle points are discarded as outliers leading to unsuccessful matching.

The solution is to identify ground points and remove them from the scan. One option is to use a line detection algorithm to detect the ground line and remove it. Flying indoor, many walls also form lines which would be discarded by the algorithm, so another method based on the known altitude of each point and the altitude of UAV is used. In the previous step of the processing pipeline the 2D scan was placed into the 3D space by rotating it by the UAV attitude and shifting it in the z-axis by the UAV altitude. Therefore the approximate altitude of each point is known.

A filter which keeps only points that fall into a specified region of altitudes (Figure 10) is constructed. We define the lower bound as:

$$B_l = \max(h_{min},\, h_{uav} - z_{mar}), \tag{12}$$

where h_{min} is the minimal absolute altitude for points to be considered for matching and its value is the altitude of the FCU when the UAV is stationary on the ground, h_{uav} is the current altitude of the UAV, and z_{mar} is the maximum relative altitude difference between the UAV and the scan point. The value of h_{min} was set experimentally to 0.2 m and z_{mar} to 1 m during both simulation and real world flights. The max function is used to saturate the lower bound in a way that when flying in lower altitudes (mostly during takeoff) the value h_{min} is used as the threshold, and in higher altitudes, $h_{uav} - z_{mar}$ is used instead. Similarly, we define the upper bound as:

$$B_u = \min(h_{max},\, h_{uav} + z_{mar}), \tag{13}$$

with h_{max} being the maximal expected altitude of obstacles. Once the bounds are defined, all points $\mathbf{p}_i(p_x,\, p_y,\, p_z)^T$ must pass through the filter which discards those not satisfying the inequality

$$B_l < p_z < B_h. \tag{14}$$

By applying this filter the points representing ground (or ceiling) are removed, and the points that are left continue to the next block of the processing pipeline.

3.1.5 Noise filtering

The output of every sensor is noisy, and LIDARs are no exception. There are two types of noise in LIDAR scans: First, the range measurements are not exact. A 10 m range measurement from the Sweep LIDAR has a ±5 cm accuracy according to the datasheet [15].

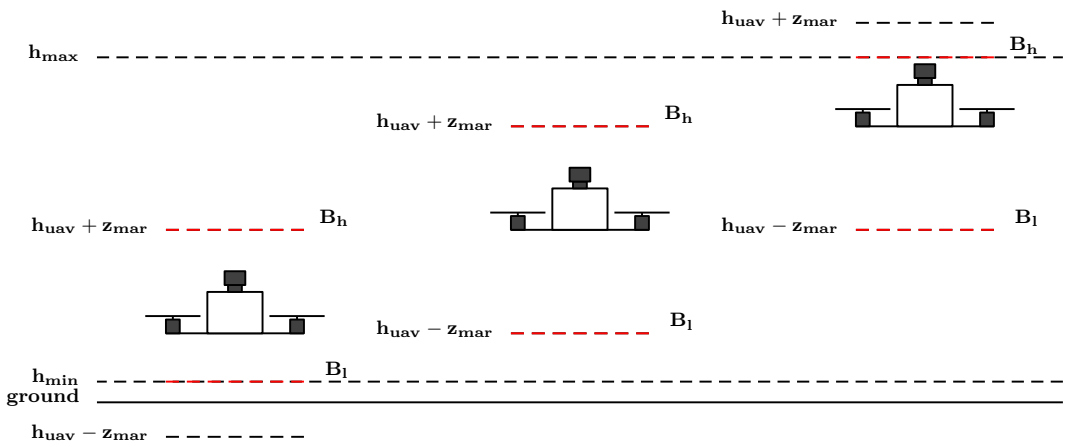


Figure 10: UAVs flying in different altitudes. The first UAV is flying close to the ground, the second is in the desired flight altitude, the third one is close to the maximal allowed altitude. Each case has the ground removal bounds depicted by red dashed lines. All points outside of the bounds are removed.

Second, the sensor produces false detections. A few points in each scan appear randomly in empty spaces of the environment. While the first type of noise does not present a significant problem since the accuracy is sufficient, the second noise type, on the other hand, presents a serious issue.

A noise removing filter was designed to find the outliers in each scan. Each point has its neighborhood analyzed in a predefined radius. If a point has no neighbor in the radius, it is likely to be a false detection or a small object. Since we know, what environment the UAV is flying in, and how large obstacles it can encounter, it is possible to tune the filter accordingly to remove only the outliers without any false positives correctly. Reasonable values which worked well in experiments presented later in the Section 7 were at least two neighbors in 1-meter radius for the point to be accepted as inlier. Of course for other LIDAR sensor or environment with smaller obstacles, the values would have to be changed.

3.1.6 Points outside flight area removal

Sometimes it is necessary to limit the flight area to prevent the UAV flying into a group of people, fragile objects, other aerial vehicles or out of sight of the safety pilot. When flying in a constrained area, the laser scan measurements that fall outside of the area should generally not be considered for adding to the map, as they come from objects that are not relevant to the flight objective. Furthermore, some of such points can be highly dynamic (a crowd of people) and degrade the position estimate and built map. All points from the scan are transformed to the *local_origin* frame using the last available position estimate. After that, it is trivial to check every point whether it falls into the area specified by ($\mathbf{x}_{min}$, $\mathbf{x}_{max}$, $\mathbf{y}_{min}$, $\mathbf{y}_{max}$) or not, and keep only the points that do. This simple check potentially prevents unwanted objects in the map and improves the robustness of the system.

3.2 Sequential matching

The next step after scan processing is sequential matching. The purpose of this module is to take two successive scans $\mathbf{P}_{t-1}$ and $\mathbf{P}_t$ which are first preprocessed in the processing module and align them in a way, which minimizes the cost function that will be introduced later in Section 4. The transformation which has to be applied to acquire such alignment is the transformation between the poses $\mathbf{x}_{t-1}$ and $\mathbf{x}_t$ from which those two scans were obtained. How the two scans are aligned is in detail described in Section 4. Fundamentally, tentative correspondences with minimal Euclidean distance are found between the two scans, and outlier correspondences are removed by finding a subset which minimizes the Fractional Root Mean Squared Distance (FRMSD) defined in [43]. The optimal transformation between the correspondences minimizing the sum of the squares of the residuals is obtained by a closed form solution of the least squares problem. The obtained transformation is applied to $\mathbf{P}_t$, and the process is iterated until a stopping condition is reached. The stopping condition is one of the following: absolute FRMSD is less than a threshold, no change in FRMSD since the last iteration, and time constraint reached. The alignment of two consecutive point clouds acquired from the Stará Voda church dataset before matching and after 5, 10, and 15 iterations is shown in Figure 11 in Cartesian coordinates with ICP correspondences and in Figure 12 in polar coordinates with IMRP correspondences. As the scans arrive at 5 Hz and the UAV is constrained to a low velocity, the difference between two successive scans is small, and the alignment is easy. Therefore the matching is most often stopped by the first or second condition within the time limit. The output of the matching method is a transformation in 2D which is a combination of rotation of angle φ and translation $[t_x, t_y]$. The 2D transformation $\mathbf{T}$ is represented by a 3x3 matrix in homogeneous coordinates:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\varphi & -\sin\varphi & t_x \\ \sin\varphi & \cos\varphi & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{15}$$

After the scans are aligned, and the final transformation between them is known, the current velocity of the UAV is obtained in the following way:

$$\dot{\mathbf{x}}_t = \begin{bmatrix} t_x \\ t_y \end{bmatrix} / \Delta t, \tag{16}$$

where Δt is the time difference between scans $\mathbf{P}_t$ and $\mathbf{P}_{t-1}$ obtained from the timestamps which are filled every time a new scan is acquired from the sensor. A sanity check of the estimated velocity follows. Assuming the velocity of the MPC tracker is constrained to 0.2 m/s, the actual velocity of the UAV is below 1 m/s most of the time based on preliminary test flights. If the estimated velocity is higher than 2 m/s, the scan matching is regarded as failed and the estimated velocity changed to zero, which is more robust than saturation, because a failure in matching does not mean, that the velocity estimate is not accurate enough, it means that the transformation is completely wrong and propagating it further would only introduce unpredictable error. Finally, the resulting velocity estimate is ready for fusion in the lateral LKF, and the module waits for the next scan.

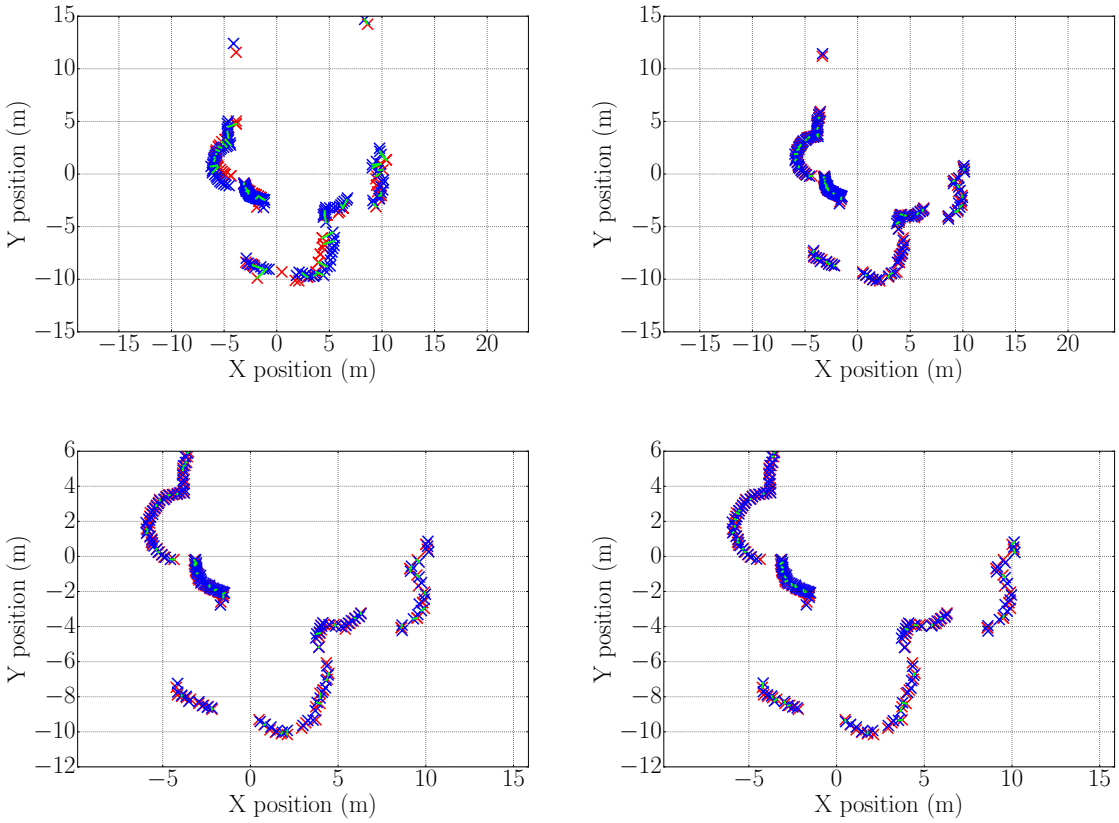


Figure 11: The sequence of scan alignment in Cartesian coordinates before matching (top left), after 5 iterations (top right), after 10 iterations (bottom left), and 15 iterations (bottom right). The source point cloud is shown in red markers, while blue points represent the target points. Green lines show the ICP correspondences between the two point clouds.

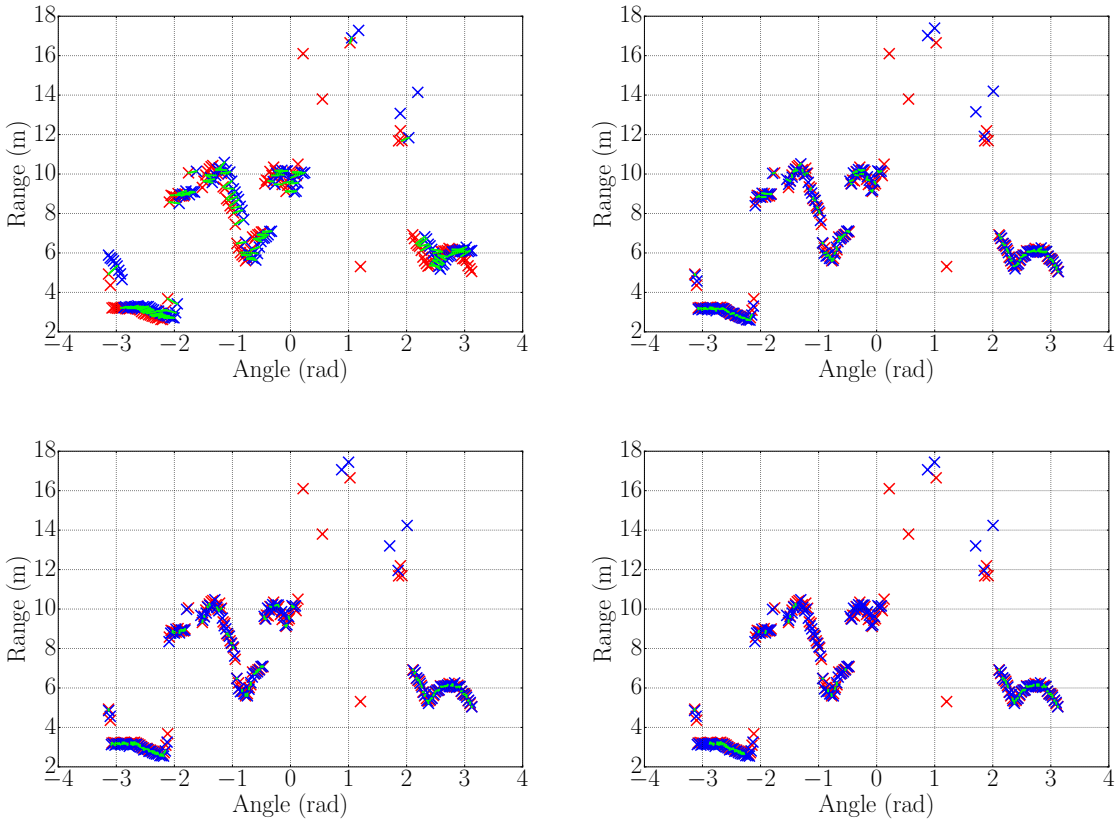


Figure 12: The sequence of scan alignment in polar coordinates before matching (top left), after 5 iterations (top right), after 10 iterations (bottom left), and 15 iterations (bottom right). The source point cloud is shown in red markers, while blue points represent the target points. Green lines show the IMRP correspondences between the two point clouds.

3.3 Global matching

This module estimates the absolute position of the UAV in the area of flight. To achieve this goal, it needs the current preprocessed scan $\mathbf{P}_t$ and the last available map state $\mathbf{M}_{t-1}$ to localize the UAV. It is not important for the module whether the map is prepared in advance manually, comes from a previous flight, or is built simultaneously during the flight, thanks to the modularity of the pipeline. The only requirement imposed on the map is that it has to arrive in the form of a point cloud. Every time a new scan arrives, a current map query is sent to the mapping module, which has to provide the current map no matter whether the map changes over time or not. After receiving both $\mathbf{P}_t$ and $\mathbf{M}_{t-1}$, the global localization can begin.

First, $\mathbf{M}_{t-1}$ is transformed into the *fcu_uav* frame using the inverse of $\mathbf{T}_{t-1}$ to have it in the same frame as $\mathbf{P}_t$. Second, $\mathbf{M}_{t-1}$ is cropped to the same size as $\mathbf{P}_t$ which depends on the sensor range. The cropping reduces the complexity of correspondence search and also limits the search to a small neighborhood which has the advantage that when there are two places in the map in which the crops are identical (typically corners formed by two walls), there is no ambiguity and the wrong match is not considered. On the other hand, long loop closures might not be possible if the total accumulated position error between returning to a previous position is more than the sensor range.

The actual scan alignment process is the same combination of state-of-the-art methods as in the sequential matching. A detailed explanation resides in Section 4. When scan matching of $\mathbf{P}_t$ to $\mathbf{M}_{t-1}$ is finished, the module outputs the transformation matrix $\Delta\mathbf{T}$. To obtain the final transformation from *fcu_uav* to *local_origin* the update equation

$$\mathbf{T}_t = \mathbf{T}_{t-1} \cdot \Delta\mathbf{T} \tag{17}$$

is used. The scan $\mathbf{P}_t$ is then transformed into *local_origin* by the transform $\mathbf{T}_t$ and current absolute position estimate is extracted from $\mathbf{T}_t$. If the UAV traveled more then 0.5 m since last map update, $\mathbf{P}_t$ is passed to the mapping module to get integrated into the global map in the case the map is built gradually. The resulting estimated absolute position is passed to the lateral LKF to be fused with other measurements. The global matching is run at 1 Hz, which is less than the frequency of sequential matching. The reason is that the global matching estimate should correct the drift of the more precise sequential matching. The drift after 1 s is negligible, therefore, running the global matching more frequently does not bring any advantage.

3.4 Mapping

The mapping module is in charge of managing the global map. It has two main responsibilities: One, when it receives a scan from the global matching module, it has to integrate the scan into the map in a robust way, ensuring consistency of the map. Two, when queried for the current map state, it has to supply a point cloud representation of the map, ideally in real time, to minimize the time delay, which is already high enough due to the low sampling frequency of the LIDAR. The module allows two modes of operation: localizing in an existing map and gradually building the map. It is possible to switch the modes as needed. The

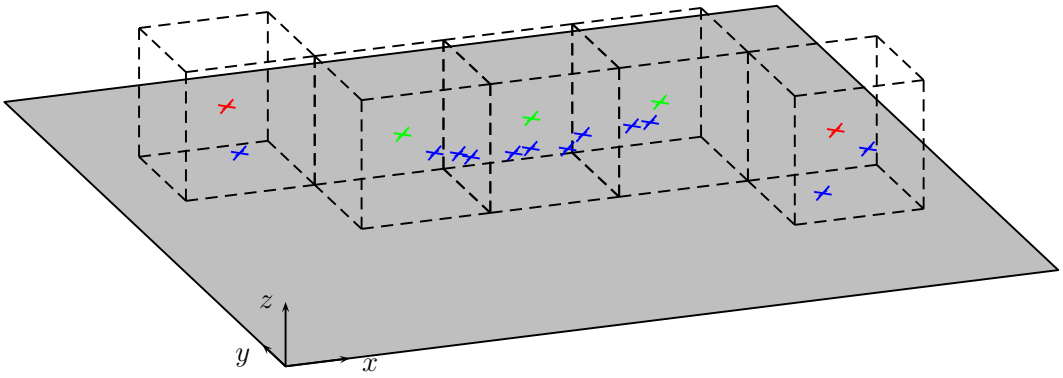


Figure 13: The octree-based global map is updated with a new incoming point cloud. The green centers of octree cells containing blue points of incoming point cloud are added to the map. The red points are centers of cells which are declined as outliers in the processing module, as the points inside them do not contain enough neighbors in their neighborhood.

chosen data structure used for storing the map is octree, thanks to its efficiency in memory and access times.

An octree is an efficient recursive data structure, in which the root node of the tree contains the whole considered space. Each node, including the root node, subdivides the space it contains into 8 identical child nodes. The nodes are recursively divided until the desired resolution, which in our case is 0.2 m, is reached. The 3D occupancy grid mapping implemented on top of the octree data structure is described in [44]. When we refer to an octree cell, we mean the lowest-level node which cannot be further subdivided.

When a point cloud arrives from the Global matching module, and the current mode of operation involves adding new points to the map, the point cloud has to be integrated into the global map. First, the new points are checked whether they are far enough from the existing octree cells. If they are further than the resolution of the octree, they are added. This step is important as it prevents map deformation in the case of suboptimal scan alignment. Second, a new point cloud is formed from the centers of occupied cells. This point cloud is relatively sparse with all points separated by at least the distance of the octree resolution, which is convenient as a map can get relatively large which would slow down the correspondence search. The process of point cloud integration into the global map is shown in Figure 13. The mapping module also keeps a dense point cloud that is not downsampled. Its purpose is to provide a detailed offline visualization of the environment for interested historians and restorers, and also as a dataset for further point cloud processing, 3D reconstruction, etc.

Additionally, a few services were implemented to control the state of the map. Load map service takes a .pcd file which contains a map. PCD is the file format from the Point Cloud Library (PCL) [45] package used for storing point clouds. The point cloud is read from the file and converted to the octree cells which are then ready to use for localization or adding additional points. When a map is loaded it may or may not be useful to enable adding new points to it. If the map is detailed enough, and not too old, the adding of new points should be disabled to prevent degrading the map by adding incorrectly aligned scans or dynamic obstacles. However, if the environment changed significantly since the map was

acquired or there are missing parts in the map, new points should be added to fill the missing information. For this purpose, the toggle mapping service was implemented. By calling the service, the adding of points from new scans can be turned on and off operatively. The last service that was implemented can reset the map. After calling the service, all points in the map are removed, and the octree structure is cleared. The use case for this service is, for example, the situation when the UAV enters a new environment without the intention of returning to the previous one. Resetting the map, in this case, can facilitate the correspondence search and decrease matching errors due to ambiguities in the environment. Another situation which may need to use the reset service is when the map becomes too inconsistent to provide a reliable position estimate. Resetting a map in these circumstances can help the system to recover and continue in normal operation.

4 Scan Matching

The core of the localization system developed in this thesis is the method that estimates the change between two consecutive UAV positions by matching scans. This section presents the details of the implemented scan matching method. Let us call $\mathbf{Z}_t$ the source scan and $\mathbf{Z}_{t-1}$ the reference scan. The task is to find the transformation from the source to the reference. The group of algorithms solving this task are collectively called scan matching or scan aligning methods.

Despite the developed alignment technique being quite complex due to the robustness requirement imposed by the need of localizing in a multitude of different environments with various scan densities and noise. The algorithm follows the same structure as the basic ICP proposed by [14]:

1. Establish correspondences between the source and reference scans that minimize a cost function.
2. Compute the transformation which transforms the source scan into the reference scan.
3. Apply the obtained transformation to the source scan.
4. Evaluate the quality of matching and iterate all steps until a stopping condition is met.

The cycle which is iterated until convergence is illustrated in Figure 14. Each step will be analyzed in detail in the following subsections.

4.1 Establishing correspondences

Finding the appropriate correspondences is key to correct scan alignment. The rate of convergence (number of iterations until reaching local minimum), robustness against noise and outliers, and the time complexity all depend on the choice of correspondences. There exist two principal types of correspondences, both of them suitable for different algorithms.

The first type is features which are extracted from the data by a feature detector algorithm. Those are mainly used when working with images, for example, panorama stitching [46], tracking of moving objects [47] or assessment of photos quality [48]. Most often features as sharp changes in brightness or color are used since they appear in distinctive areas of the image as corners and edges. For laser scans, the features are extracted based on the geometric properties of the scan (lines, corners, arcs or polygons).

In the second type of correspondences, no features are detected and the search is performed in the raw data. For images, it is highly impractical to use raw data, since every picture consists of millions of pixels so that correspondence search would be too time demanding. This is also true for laser scans from sensors with the high sampling rate, which output millions of points per scan. Such scans can be downsampled prior to the correspondence search, or distinctive areas can be localized and used as features.

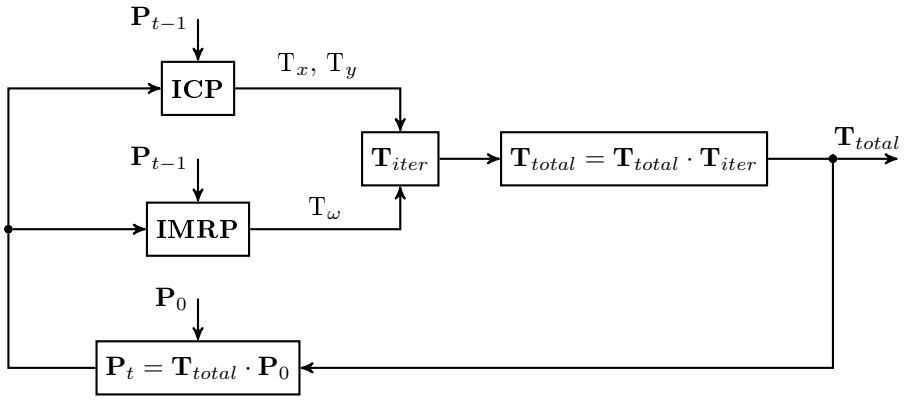


Figure 14: The diagram of the scan matching process. A transformation $\mathbf{T}_{iter}$ is found by combining translation T_x, T_y, from the ICP algorithm, with the rotation ω obtained from IMRP method. Then the total transformation is updated and applied to the source point cloud which is used to estimate $\mathbf{T}_{iter}$ in the next iteration.

On the other hand, for a sensor with lower sampling rates using point-to-point correspondences is more appropriate since the measured point cloud is rather sparse and obtaining unambiguous geometric features is not possible due to low detail of scanned objects. While feature extraction can provide a unique representation of the environment, with low density scans it cannot find sufficient amount of features to estimate the transformation between scans reliably. Since the point-to-point correspondence search does not discard any points, no spatial information is lost in the process, and the transformation can still be estimated even in situations where feature-based methods fail.

4.1.1 ICP, IMRP and IDC

In the ICP algorithm [14] the closest point correspondences are used. The correspondence of a point $\mathbf{p}_i \in \mathbf{P}_t$ is found as the closest point $\mathbf{p}'_i$ from $\mathbf{P}_{t-1}$ in terms of Euclidean distance. To improve the accuracy of matching, especially with sparse point clouds, the linear interpolation of $\mathbf{P}_{t-1}$ is used. The neighbor point of $\mathbf{p}'_i$ that is closer to $\mathbf{p}_i$ is selected to be the adjacent point $\mathbf{p}'_a$. A virtual correspondence $\mathbf{p}'_v$ is formed on the closest point of the line segment $|\mathbf{p}'_i\mathbf{p}'_a|$ as shown in Figure 15.

As mentioned in [49], the Euclidean distance metric leads to a fast convergence to a local minimum of the translation between two scans. However, the convergence of the rotation θ is much slower, which is to be expected, as the Euclidean distance metric of two points has no rotational component. As a result, the algorithm tries to decrease the rotational error by translation rather than rotation, thus causing an unnecessary increase in the rate of convergence of the whole algorithm, which is a problem when the matching is constrained by the real-time requirement. We use the Iterative Matching Range Point (IMRP) to improve the performance of rotation estimation. The structure of the algorithm is the same as of ICP, and the only difference is the correspondence search that is using a rotation-based

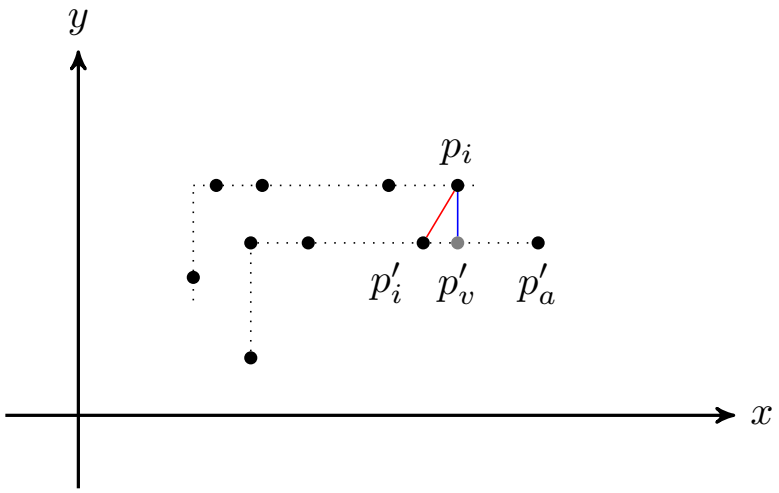


Figure 15: The correspondence search at line segments. Red line represents the correspondence found by regular ICP, i.e., the closest point. Searching correspondence at line segment produces a virtual gray point $\mathbf{p}'_v$ lying between the closest point $\mathbf{p}'_i$ and the neighbor $\mathbf{p}'_a$. The point, that is connected to $\mathbf{p}_i$ of source point cloud by a blue line, is then used as correspondence for transformation computation.

metric. The correspondences are searched in the polar coordinates in which the laser scans are natively represented. For every measurement $\mathbf{z}_i = (r_i, \varphi_i) \in \mathbf{Z}_t$, we find the corresponding measurement $\mathbf{z}'_i = (r'_i, \varphi'_i) \in \mathbf{Z}_{t-1}$ that satisfies $\varphi'_i \in [\varphi_i - B_\omega, \varphi_i + B_\omega]$ minimizing the metric $\|r_i - r'_i\|^2$, where B_ω is the maximal expected rotation error. The expected maximal rotation error is B_ω is subject to change in every iteration as the scan alignment converges to a local minimum. Since the rotational residual decreases exponentially with the number of iterations, we set $B_\omega(t) = B_\omega(0) \exp^{-\alpha t}$ to prevent incorrect correspondences in later iterations.

Scan matching based on IMRP converges faster in the rotational component, on the other hand, the convergence of translation is slower than in ICP. The obvious solution is to combine both algorithms in a way, that ICP correspondences are used for estimating the translation, while rotation is estimated from IMRP correspondences, which in literature is referred to as the Iterative Dual Correspondence (IDC). Both types of correspondences are visualized in Figure 16.

4.1.2 Outlier rejection

Another problem of the basic ICP algorithm is that it is not robust to outliers. The points measured from position $\mathbf{x}_{t-1}$ which cannot be seen from position $\mathbf{x}_t$ due to moving out of range of the sensor or occlusion do not have a correspondence pair and have to be rejected to prevent convergence impairment or even incorrect alignment. This issue is solved by incorporating the fraction of inliers into the distance function [43] thus forming a new

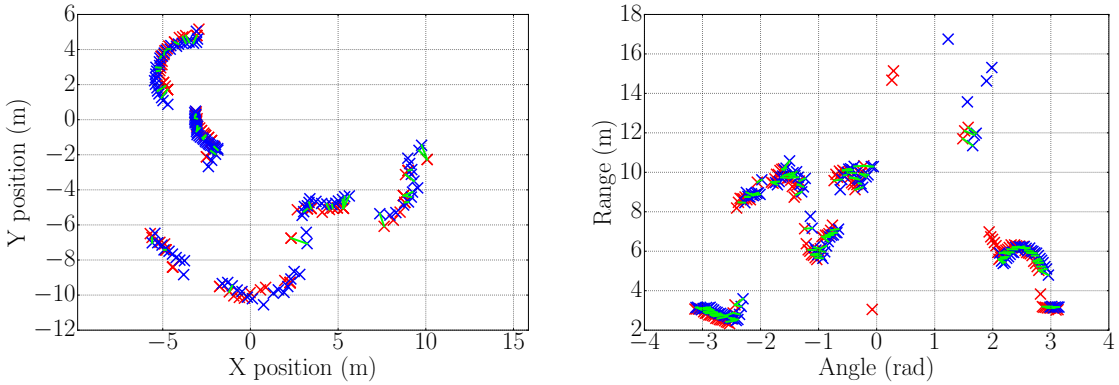


Figure 16: There are two types of correspondences. In the left plot, Cartesian correspondences which use the Euclidean distance measure are shown, while the polar correspondences are on the right. Red markers represent the source point cloud, and blue is used for target points. The corresponding points between the two point clouds are connected by a green line. The points come from a dataset acquired during a manual flight in a church.

distance measure Fractional Root Mean Squared Distance (FRMSD) which is defined as

$$\text{FRMSD}\left(\mathbf{P}_t,\, \mathbf{P}_{t-1},\, f, \mu\right) = \frac{1}{f^{\lambda}} \sqrt{\frac{1}{f \cdot n} \sum_{\mathbf{p} \in \mathbf{P}_t} \|\mathbf{p} - \mu(\mathbf{p})\|^2}, \tag{18}$$

where $f \in [0,\, 1]$ is the fraction of n points in $\mathbf{P}_t$ sorted by $\|\mathbf{p} - \mu(\mathbf{p})\|^2$ that will be used for alignment, $\mu(\mathbf{p})$ is the function which finds the closest point of $\mathbf{p}$ in $\mathbf{P}_{t-1}$ and λ is a parameter which depends on the expected outlier density and according to [43] should be set to $\lambda = 1.2$ for general-purpose scan matching in two dimensions. By fine-tuning λ, one can adjust the ratio of false positives to false negatives. A smaller value of λ can reject correct correspondences as outliers while on the other hand, higher values result in more outliers classified as inliers.

Even with the outliers filtered out, correspondences with greater distance still contribute to the estimated transform more than correspondences separated by a larger gap. The performance of the matching is improved by assigning a weight to each correspondence pair as:

$$w(d_i) = 1 - \frac{d_i}{\max(d_{1,\,\ldots,\,n})}, \tag{19}$$

where $w()$ is the weighting function and d_i the distance of i-th correspondence pair.

4.2 Computing transformation

Now that the correspondences have been established we can compute the registration, i.e., the transformation which minimizes the cost function define as the sum of squared resid-

uals

$$E(\mathbf{T}) = \sum_{i=1}^{n} w \|\mathbf{T}\mathbf{p}_i - \mathbf{p}'_i\|^2, \tag{20}$$

where $\mathbf{p}_i$ is the i-th point from the source point cloud, $\mathbf{p}'_i$ the i-th point from the reference point cloud and the 2D transformation matrix $\mathbf{T}$ with translation components T_x, T_y and rotation component ω. A closed form solution of the least squares problem above can be derived as follows.

$$E(\mathbf{T}) = \sum_{i=1}^{n} \left(\left(x_i \cos\omega - y_i \sin\omega + T_x - x'_i \right)^2 + \left(x_i \sin\omega - y_i \cos\omega + T_y - y'_i \right)^2 \right) \tag{21}$$

Minimizing $E(\mathbf{T})$ we get

$$\begin{aligned} \omega &= \arctan \frac{S_{xy'} - S_{yx'}}{S_{xx'} + S_{yy'}} \\ T_x &= \mu'_x - (\mu_x \cos\omega - \mu_y \sin\omega) \\ T_y &= \mu'_y - (\mu_y \sin\omega + \mu_y \cos\omega) \end{aligned} \tag{22}$$

with the mean values defined as

$$\begin{aligned} \mu_x &= \frac{1}{n} \sum_{i=1}^{n} x_i & \mu_y &= \frac{1}{n} \sum_{i=1}^{n} y_i \\ \mu'_x &= \frac{1}{n} \sum_{i=1}^{n} x'_i & \mu'_y &= \frac{1}{n} \sum_{i=1}^{n} y'_i \\ S_{xx'} &= \sum_{i=1}^{n} (x_i - \mu_x)(x'_i - \mu'_x) & S_{yy'} &= \sum_{i=1}^{n} (y_i - \mu_y)(y'_i - \mu'_y) \\ S_{xy'} &= \sum_{i=1}^{n} (x_i - \mu_x)(y'_i - \mu'_y) & S_{yx'} &= \sum_{i=1}^{n} (y_i - \mu_y)(x'_i - \mu'_x) \end{aligned} \tag{23}$$

or

$$\begin{aligned} \mu_x &= \frac{1}{n} \sum_{i=1}^{n} w_i x_i & \mu_y &= \frac{1}{n} \sum_{i=1}^{n} w_i y_i \\ \mu'_x &= \frac{1}{n} \sum_{i=1}^{n} w_i x'_i & \mu'_y &= \frac{1}{n} \sum_{i=1}^{n} w_i y'_i \\ S_{xx'} &= \sum_{i=1}^{n} w_i (x_i - \mu_x)(x'_i - \mu'_x) & S_{yy'} &= \sum_{i=1}^{n} w_i (y_i - \mu_y)(y'_i - \mu'_y) \\ S_{xy'} &= \sum_{i=1}^{n} w_i (x_i - \mu_x)(y'_i - \mu'_y) & S_{yx'} &= \sum_{i=1}^{n} w_i (y_i - \mu_y)(x'_i - \mu'_x) \end{aligned} \tag{24}$$

for the weighted variant of the algorithm.

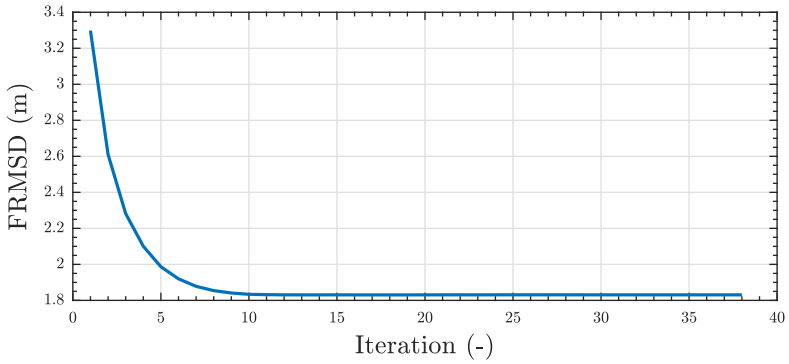


Figure 17: The exponential convergence of the developed scan matching module. The plotted line is the mean taken from 142 runs of the global scan matching during one flight in simulation.

4.3 Evaluation and iteration

Since the algorithm cannot estimate the transformation in one step, the total transformation $\mathbf{T}_{total}$ is updated after every iteration by multiplying it by $\mathbf{T}_{iter}$. After the update, $\mathbf{T}_{total}$ is used to transform the source point cloud $\mathbf{P}$. This is the end of one iteration of the algorithm, and it is time to analyze whether the found solution converged or additional iterations are needed.

The convergence is checked by the difference of FRMSD between the current and last iteration. Since the error typically decreases exponentially, which is shown in Figure 17, the difference between the consecutive errors is compared against a close-to-zero constant and if the difference is smaller, the algorithm has converged and further iterations are not necessary. Similarly, when the FRMSD is less than 1 cm the matching does not continue. Although the error might not be entirely converged and could still improve, it does not add any useful information as the error is below the noise level of the sensor. Additionally, the matching will be stopped, regardless of the current error, if a time limit is exceeded. The time available for matching is set to 50 ms to satisfy the real-time requirement of localization with the Sweep LIDAR which supplies laser scans at 5 Hz, thus introducing a considerable delay into the system, which is further increased by the scan processing and matching algorithms.

5 State Estimation

In robotic applications which involve the simultaneous operation of several sensors with different sensing principles, precisions and sample rates, the question arises: How to translate different sensory inputs into reliable estimates that can be used by other subsystems? The answer is to use one of sensor fusion methods. Sensor fusion is the process of integrating data from different sensors to estimate the state of the robot. According to a survey of sensor fusion techniques in [50], the Linear Kalman Filter and the Extended Kalman Filter (for models with non-linear dynamics) are the most popular tools proposed in the literature for sensor fusion in mobile robot navigation.

5.1 Linear Kalman filter

Linear Kalman filter is essentially an optimal state observer for stochastic systems with a linear model. The estimation task can be formulated as finding the Linear Mean Square (LMS) state estimate that minimizes the criterion

$$J_{LMS} = \mathcal{E}\left\{(\mathbf{x} - \hat{\mathbf{x}})^T (\mathbf{x} - \hat{\mathbf{x}})\right\}. \tag{25}$$

Consider a state-space description of a stochastic LTI system:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{u}_t + \mathbf{v}_t \tag{26}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{D}\mathbf{u}_t + \mathbf{e}_t, \tag{27}$$

where $\mathbf{A}$ is the system matrix, $\mathbf{B}$ the inputs matrix, $\mathbf{C}$ the output matrix, and $\mathbf{D}$ the feedforward matrix. The vectors $\mathbf{x}_t$, $\mathbf{u}_t$, $\mathbf{y}_t$ represent the state, input, and output vectors in time t respectively. The stochastic part is described by the process noise $\mathbf{v}_t$ drawn from the normal distribution $\mathcal{N}(\mathbf{0}, \mathbf{Q})$, and the measurement noise $\mathbf{e}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$.

To use the Linear Kalman filter, the system must be observable, must have linear dynamics[1], the system model must be known, and the process and measurement noises must be uncorrelated. When these conditions are satisfied, the Kalman filter provides an optimal state estimate that minimizes the mean square error. The algorithm consists of two steps: First, the data step updates the state estimate $\hat{\mathbf{x}}_t$ and its covariance matrix $\mathbf{\Sigma}_t$ based on new measurements

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t + \mathbf{K}_t \epsilon_t \tag{28}$$

$$\mathbf{\Sigma}_t = \mathbf{\Sigma}_t - \mathbf{K}_t \mathbf{P} \mathbf{\Sigma}_t, \tag{29}$$

where $\mathbf{z}_t$ is the measurement, $\mathbf{P}$ is the measurement mapping matrix, ϵ_t is the error of output estimate (innovation)

$$\epsilon_t = \mathbf{z}_t - \mathbf{P}\hat{\mathbf{x}}_t - \mathbf{D}\mathbf{u}_t, \tag{30}$$

and $\mathbf{K}_t$ is the Kalman gain

$$\mathbf{K}_t = \mathbf{\Sigma}_t \mathbf{P}^T \left[\mathbf{P}\mathbf{\Sigma}_t \mathbf{P}^T + \mathbf{R}\right]^{-1}. \tag{31}$$

[1] For nonlinear system models the Extended Kalman Filter (EKF) or Unscented Kalman Filter (UKF) [51] can be used instead

Second, the time step updates the state estimate based on the model

$$\hat{\mathbf{x}}_{t+1} = \mathbf{A}\hat{\mathbf{x}}_t + \mathbf{B}\mathbf{u}_t \tag{32}$$

$$\mathbf{\Sigma}_{t+1} = \mathbf{A}\mathbf{\Sigma}_t\mathbf{A}^T + \mathbf{Q}. \tag{33}$$

In this work, we implemented Kalman filter for estimating the altitude of the UAV. Second and third Kalman filters were implemented to estimate the position in the x-axis and y-axis. The two lateral Kalman filters are equivalent and could have been implemented as one. However, we decided to decouple the two axes for faster calculation of matrix inverse.

5.2 Altitude estimation

The scan-matching-based localization method itself does not involve estimation of altitude of the UAV. However, the altitude estimate is still required to place the 2D laser scan into the 3D global map and ground removal during scan processing. The UAV is equipped with 2 sensors measuring the altitude: the downward facing laser rangefinder, and the IMU that fuses accelerometer and barometer measurements. The laser rangefinder measurements are noisy and sensitive to uneven terrain, and the IMU altitude estimate has an offset that depends on the atmospheric pressure. We can fuse the measurements from both sensors in the LKF to eliminate their flaws and obtain a smooth altitude estimate without offset.

The altitude estimate $\hat{h}_t$ from the IMU enters the Kalman filter as the input $\mathbf{u}_t \in \mathbb{R}^{1\times1}$ through the input matrix $\mathbf{B} \in \mathbb{R}^{1\times1}$ with the process covariance matrix $\mathbf{Q} \in \mathbb{R}^{1\times1}$. The altitude measured by the laser rangefinder is integrated as a measurement $\mathbf{z}_t$ through the measurement mapping matrix $\mathbf{P} \in \mathbb{R}^{1\times1}$ with the measurement covariance $\mathbf{R} \in \mathbb{R}^{1\times1}$. The system matrix $\mathbf{A} \in \mathbb{R}^{1\times1}$ governs the dynamics of the model. The only state of the model is the altitude. The summary of values that were used for the model and covariances follows:

$$\mathbf{A} = \begin{bmatrix}1\end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix}0.01\end{bmatrix}, \quad \mathbf{P} = \begin{bmatrix}1\end{bmatrix}, \tag{34}$$

$$\mathbf{Q} = \begin{bmatrix}0.001\end{bmatrix}, \quad \mathbf{R} = \begin{bmatrix}1000\end{bmatrix}, \tag{35}$$

$$\mathbf{u}_t = (\hat{h}_t - \hat{h}_{t-1})/\Delta t. \tag{36}$$

The IMU generates the estimates on 100 Hz rate, thus $\Delta t = 0.01$. The PixHawk altitude estimate, Garmin measurements, and fused altitude estimate from a real-world experiment can be seen in Figure 18.

5.3 Lateral estimation

The position in global map coming from the global matching module cannot be used by the controller directly due to the low frequency (1 Hz) of the position estimates, and abrupt changes in the estimate that might destabilize the UAV. The sequential estimates are coming at a faster rate (10 Hz), however, the estimates are not absolute and tend to drift. The lateral Kalman filter fuses these two measurements with measurements of the attitude of the UAV and the attitude command from the controller.

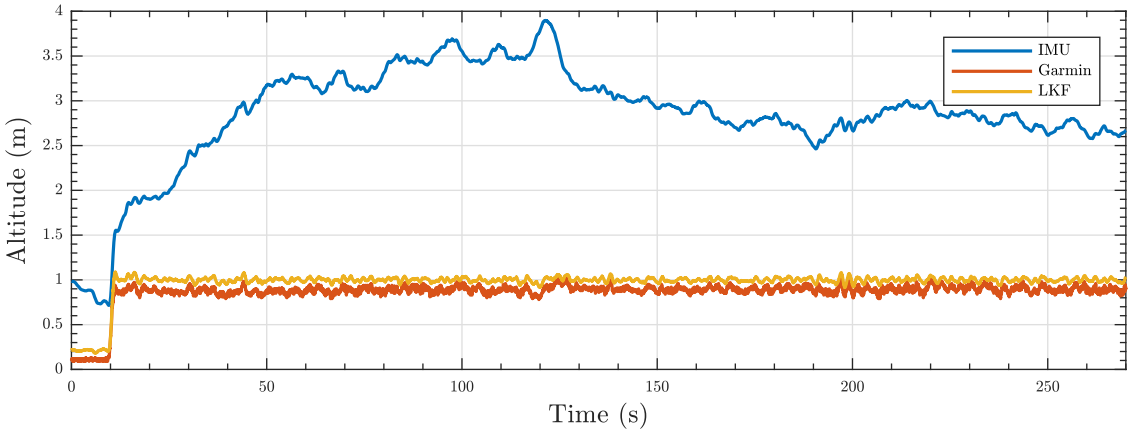


Figure 18: The fusion of altitude from IMU and Garmin laser measurements. The data comes from the hardware SLAM experiment.

The model that was used consists of 6 states $\mathbf{x} = \left(x, \dot{x}, \ddot{x}, \ddot{x}^{(u)}, \ddot{x}^{(d)}, \theta\right)$ – position, velocity, acceleration, acceleration from control input, disturbance acceleration, and rotation around the y-axis in *local_origin* frame. We do not know the orientation of the UAV in the *local_origin* frame explicitly. Nevertheless, we can obtain it from the orientation in the *fcu_uav* frame easily by rotating the roll and pitch angles by the yaw angle. The acceleration state $\ddot{x}$ is a sum of two components: the acceleration from the control input $\ddot{x}^{(u)}$ and the acceleration from external disturbances $\ddot{x}^{(d)}$. Thanks to the disturbance accelerations state, the external disturbances do not appear in the estimate of position.

The parameters of the transfer function from the attitude command to the actual attitude were identified previously based on data from a test flight. The identification flight involved slow oscillations in x and y axes, during which the dynamics is near linear. An overdetermined set of linear equations was constructed from the flight data, and the parameters were estimated by finding a solution of the overdetermined system by minimizing the least squares residuals. The resulting transfer function is

$$\theta_t = 0.903 \cdot \theta_{t-1} + 0.097 \cdot \theta_t^{(u)}. \tag{37}$$

Similarly, the transfer function from the attitude to acceleration had to be identified. The relation between attitude and acceleration is defined by

$$\ddot{x} = \frac{f}{m} \sin\theta, \tag{38}$$

where f is the total thrust force exerted by the propellers in the z-axis of the *fcu_uav*, and m the mass of the UAV. Since the flight velocity was low during the identification experiment, and the tilt of the drone small, we can say that

$$\sin\theta \simeq \theta. \tag{39}$$

Now we have a linear equation with single parameter

$$\ddot{x} \simeq \frac{f}{m}\theta = p\theta. \tag{40}$$

From the identification flight we found the value of the parameter minimizing the least square residuals to be

$$p = 6.3512. \tag{41}$$

Having estimated the parameters of the model, we set the system matrix $\mathbf{A} \in \mathbb{R}^{6\times6}$ and input matrix $\mathbf{B} \in \mathbb{R}^{6\times1}$ to

$$\mathbf{A} = \begin{bmatrix} 1 & 0.01 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 6.3512 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.903 \end{bmatrix}, \quad \mathbf{B} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0.097 \end{bmatrix}. \tag{42}$$

The process covariance was set to $\mathbf{Q} = \mathbf{I} \in \mathbb{R}^{6\times6}$. We fuse the position x and velocity $\dot{x}$ estimates from the scan matching techniques as measurements of the states. Since the scan matching estimates are delayed, the attitude measurement θ was added to decrease the delay in the final fused estimate. The list of measurement mapping matrices $\mathbf{P}$ and their covariances $\mathbf{R}$ follows:

$$\begin{aligned} x: &\quad \mathbf{P}_{pos} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{R}_{pos} = \begin{bmatrix} 100 \end{bmatrix}, \\ \dot{x}: &\quad \mathbf{P}_{vel} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{R}_{vel} = \begin{bmatrix} 0.1 \end{bmatrix}, \\ \theta: &\quad \mathbf{P}_{ang} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{R}_{ang} = \begin{bmatrix} 100 \end{bmatrix}. \end{aligned} \tag{43}$$

Note that the described lateral Kalman filter is only for the x-axis. The Kalman filter used for y-axis is identical with the same model and covariances. The estimates from the lateral Kalman filter can be seen in the experiments done in both simulation in Section 6 and experiments in Section 7.

6 Validation in simulation

This section presents a validation of the developed method in the simulation. The functionality of the localization system had to be analyzed in a realistic simulator to decide whether it performs reasonably well before it could be deployed to the real UAV. For this purposes, the Gazebo simulator was chosen thanks to its large community and close integration with ROS. The advantage of using a simulator is easy access to ground truth data which allows precise evaluation of the localization output by comparing it to the ground truth values. This is impossible in real-world experiments without a precise position estimate which could be used as ground truth.

At the beginning of this section, the simulation setup is described, the parameter values which were used in respective modules of the localization system are introduced, the trajectory generation is presented, and the goal of the validation stated. An analysis of the data acquired during the simulated flight follows and the section ends with a discussion of the results.

6.1 Simulation setup

The primary purpose of the simulation is to evaluate the suitability for deployment to the real UAV, and so the simulated environment, conditions, and parameters had to reflect the reality as close as possible. In Figure 19, a screenshot of the simulation environment from the Gazebo simulator is shown. Two types of obstacles are spread out on a featureless grass plane. Each type substitutes the target environment for on-board-localized UAV deployment. The first type is a thin 2 meters wide and high segment representing a model of a wall which is connected with other segments to form longer walls and corners typically encountered in indoor environments such as churches. The second type represents the trunk of a tree with the diameter of 0.5 m which in larger quantities can serve as a simplified forest environment. The same types of obstacles are also used in the section Section 7, where the localization system capabilities are demonstrated in the real environment.

The simulated drone (Figure 19) is an authentic model of the real one, which implies identical flight dynamics. Due to the low rate of laser scans which arrive at 5 Hz (the rotation frequency of the Sweep LIDAR), the localization system is not suitable for aggressive flights. Flights in a church should be slow and steady so that the camera can use longer exposure times to take sharp photos even in dark areas. Moreover, fast flying could present a risk of damaging the church interiors as there would not be enough time for failsafe mechanisms to trigger or operator to take control of the UAV during a failure. Considering the above, the maximal flight velocity was set to 1 m/s. A UAV flying at this velocity traverses 0.2 m between scans which is close enough for the scan matching algorithm to converge into the global minimum. The flight altitude was set to 1 m taking into account the 2 m high obstacles. Flying at this height provides the maximal maneuverability in roll and pitch angles, as it allows the UAV to tilt more before the LIDAR captures the ground or misses the obstacle. The covariances

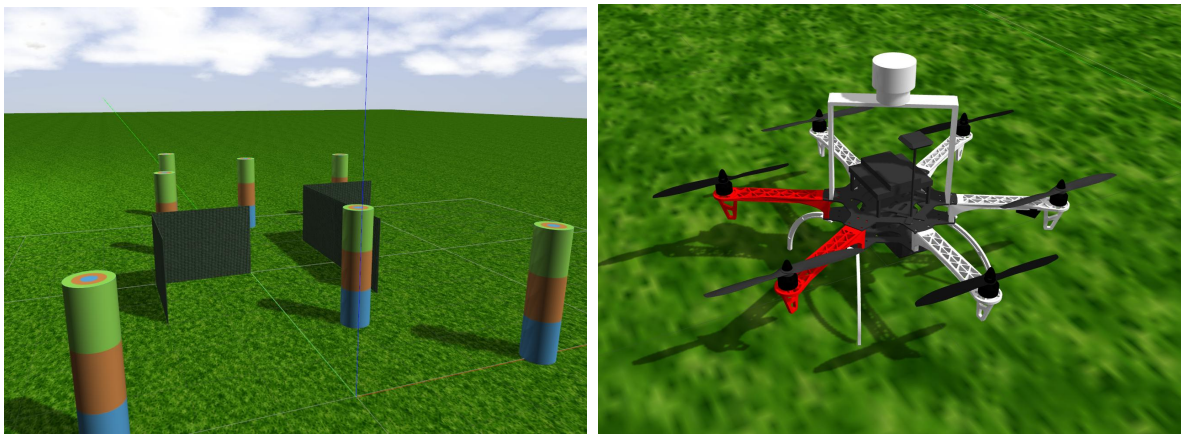


Figure 19: The environment prepared for validation of the developed method in the Gazebo simulator is on the left image. On the right picture, there is the model of UAV used in the simulation.

of lateral linear Kalman filter were set as follows:

$$\mathbf{R}_{pos} = \begin{bmatrix}100\end{bmatrix}, \quad \mathbf{R}_{vel} = \begin{bmatrix}0.1\end{bmatrix}, \quad \mathbf{R}_{ang} = \begin{bmatrix}100\end{bmatrix},$$
$$\mathbf{Q} = \begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\\ 0 & 1 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & 0 & 0 & 0\\ 0 & 0 & 0 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}. \tag{44}$$

Regarding the gains of the SO(3) controller from Equation (6), they were set to be softer ($k_x = 3$, $k_v = 2$) than in usual GPS localized flights ($k_x = 7$, $k_v = 2.4$). The reason for that, is to limit aggressive maneuvers which involve high tilt angles and also to avoid destabilizing the control system by delayed position estimate in the feedback loop.

6.2 Trajectory generation

The MPC tracker supports trajectory following through position setpoints specified in an csv file. However, it does not guarantee to follow the trajectory accurately when it is not feasible due to the current constraints of the MPC. A simple Matlab script was written to generate smooth and feasible trajectories by interpolating manually positioned setpoints by splines. First, the XML file describing the simulation world is read and the positions of obstacles are parsed and visualized in a map shown in Figure 20. The user can then add points with an arbitrary gap into the map. When all points are added, the algorithm fits splines defined by 3rd order polynomials onto the points to produce a smooth trajectory. The spline fitting is implemented by the internal Matlab function `interp1`. Each axis is interpolated independently. To keep a constant velocity, the time where the UAV will be in every input point has to be known. If the UAV moves linearly with constant velocity between

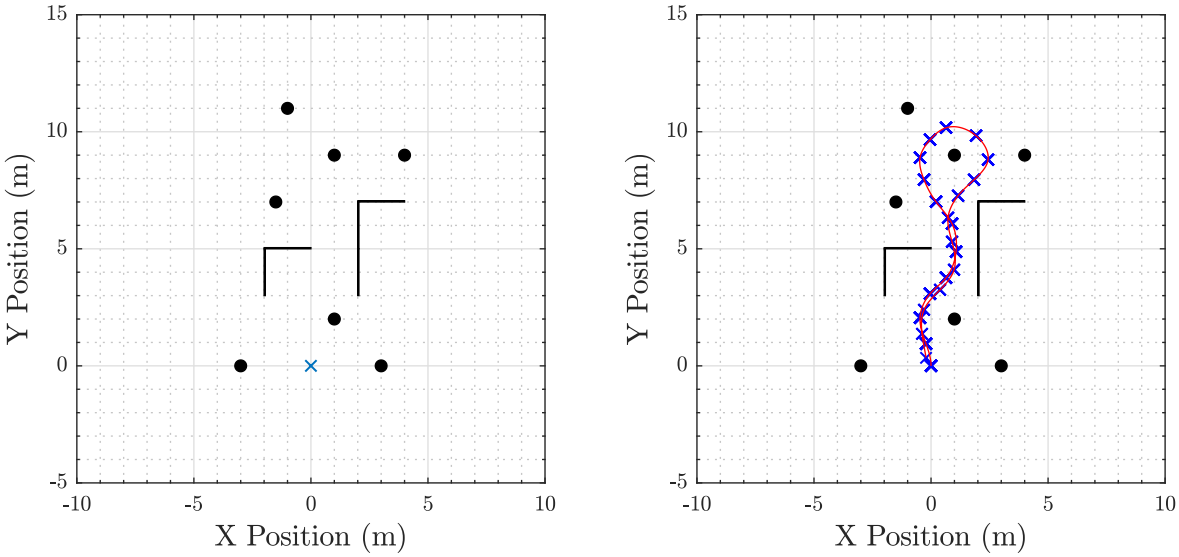


Figure 20: The interface for the planning of trajectories. The map in the left visualize a top-down view of the simulation world as loaded from the XML file. The obstacles are shown in black. Red splines are fitted onto the blue input points in the right map.

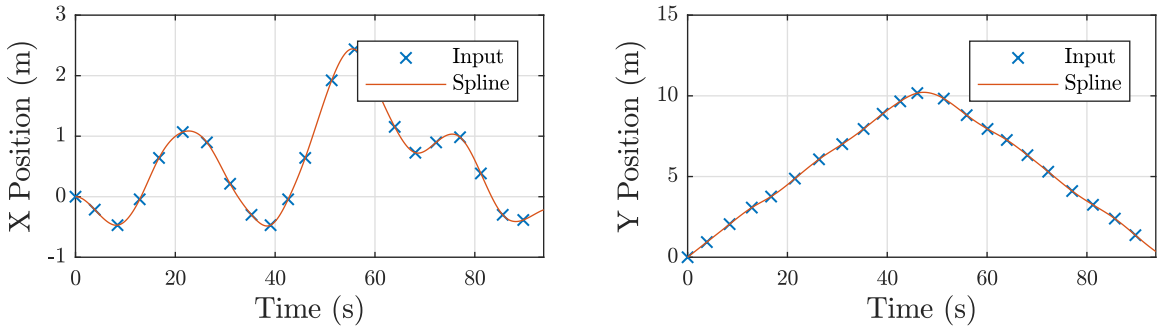


Figure 21: The development of trajectory position in time for both axes.

the points, the time t_i in point $\mathbf{x}_i = (x, y)^T$ can be calculated as

$$t_i = \sum_{j=1}^{i-1} t_j + \frac{\|\mathbf{x}_i - \mathbf{x}_{i-1}\|^2}{v}, \tag{45}$$

where v is the target velocity. Since the interpolation method in not linear, a small error in velocity is introduced this way (see Figure 22), which is not an issue since the MPC velocity limit (1 m/s) is not exceeded. For a more complex motion planning with an exact velocity profile, the algorithm based on Hermite splines in [52] can be used. The Root Mean Squared Error (RMSE) of the velocity in the case of this trajectory is 0.0194 m. The resulting interpolated trajectory for both axes can be seen in Figure 21. It is feasible, smooth and within MPC velocity limit.

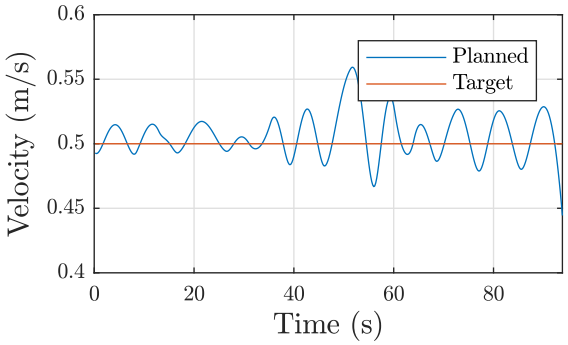


Figure 22: The planned velocity compared to the target velocity.

6.3 Goal of validation

There are two major objectives in the simulation experiments. First, to qualitatively analyze the position estimates produced by the system. Second, to safely test the whole pipeline whether it is robust and reliable enough to be deployed on real hardware.

Thanks to the ground truth from the simulator we can assess the exact position error in each moment of flight which is generally not possible with real UAV, as the even the precise RTK GPS has an error of few centimeters. Moreover, the RTK is often not available, due to communication errors or bad GNSS reception. The quality estimate in simulation will be measured by the root-mean-square error (RMSE):

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (\hat{x}_i - x_i)^2}{n}} \tag{46}$$

6.4 Simulation results

During the simulation flight, the UAV took off from the position $\mathbf{x}_{home} = (-1, 1)$ and shortly after takeoff ($t \approx 10\,s$) received command sending it to the trajectory start. After stabilizing at the initial position $\mathbf{x}_0 = (0, 0)$ of the trajectory ($t \approx 15\,s$) the trajectory following started. The flown trajectory was the one prepared by the trajectory planner in Subsection 6.2. In $t \approx 125\,s$ the trajectory ends with the UAV back at $\mathbf{x}_0$ followed by hovering at the position for the rest of the flight. The mapping was allowed for the whole flight and no points were in the map prior to the experiment start.

The position estimates compared to the ground truth can be seen in Figure 23. At $t = 10$ s, the error was the largest due to the flight to trajectory start, which was faster than the planned trajectory. There was a moment around $t \approx 40\,s$ where the method had problems with matching the scan into the map. In the plot of X position, both estimates deviated from the ground truth by up to 0.6 m in the middle of the flight where the peak deviation was reached. The RMSE in the Y position (0.2319 m) was half of the X position RMSE (0.1164 m) indicating that the X direction was poorly defined by the obstacles. The values of RMSE from the whole flight are reported in the Table 2. The deviation from the

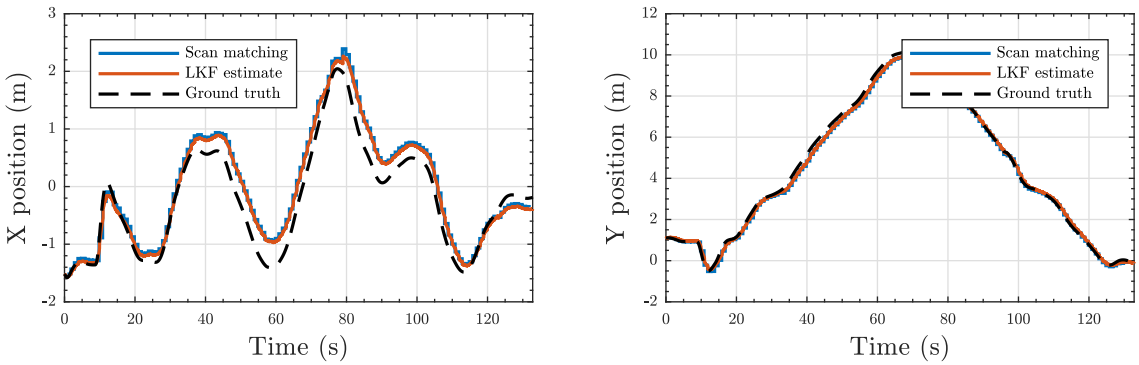


Figure 23: Position estimates from the system validation simulation run. Scan matching is the estimate from matching scans to global map which after fusion of additional measurements (velocity and tilt) forms the LKF estimate. The ground truth is the exact position coming straight from the Gazebo simulator.

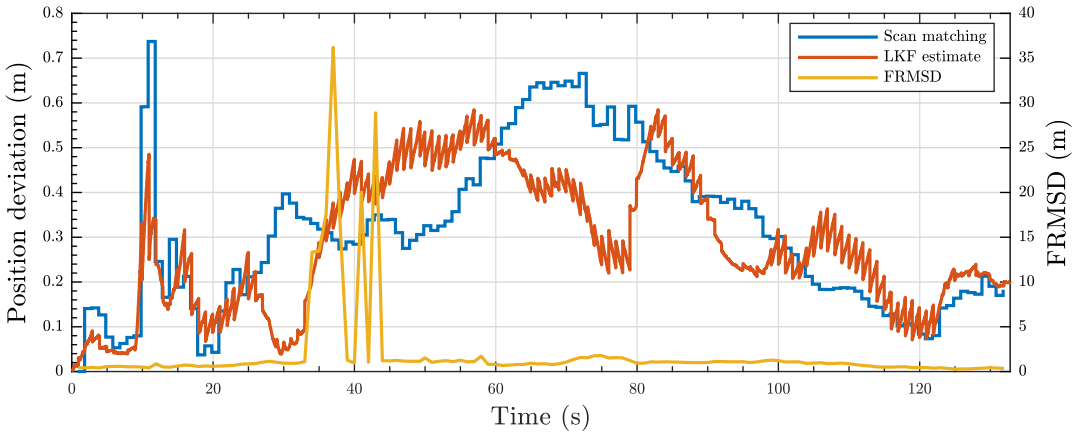


Figure 24: The deviation of the scan matching and LKF estimate from the ground truth. The Fractional Root Mean Square Distance of correspondences from the global map matching is plotted into the same graph to illustrate where the algorithm had problems aligning the scans.

ground truth position in time is shown in Figure 24 along with the FRMSD. The FRMSD at $t \approx 40$ s is more than 10 times higher than the average FRMSD through the whole flight and also the position deviation in the x-axis starts to increase. In Figure 25 the position estimates with the ground truth position are plotted into the simulation environment. The largest error occurred in the top part of the map, due to a lower number of points in the laser scans. From $t \approx 90$ s, the deviation starts to decrease thanks to UAV returning to a previously mapped part of the surroundings.

To calculate the velocity, the displacement between successive scans obtained from the sequential matcher module is divided by the time between the scans, which depends on the rotational frequency of the LIDAR. In the simulation, as well as in the case of real Sweep sensor, the rotational frequency was set to 5 Hz, corresponding to 200 ms between scans. The Figure 26 shows the velocity estimated by the sequential matcher module by alignment of successive scans. The scan matching estimate is highly accurate with only 0.0006 m RMSE

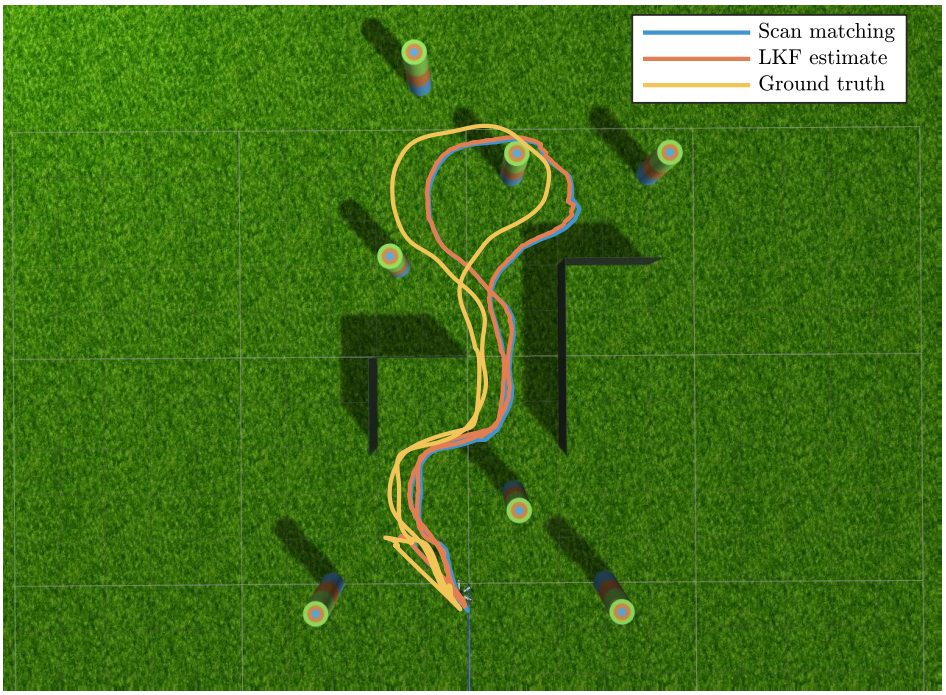


Figure 25: An aerial view of the flight area in the simulator. The UAV is hovering at the initial position of the trajectory $\mathbf{x}_0$. The largest error around 0.5 m occurred in the top of the map due to low number of points in laser scans.

	Position RMSE (m)		Velocity RMSE (m)	
	x	**y**	**x**	**y**
Scan matching	0.2319	0.1164	0.0006	0.0007
Kalman filtered	0.1747	0.0847	0.0038	0.0030

Table 2: The Root Mean Squared Error of position and velocity in both axes calculated from the whole simulation flight.

deviation from the ground truth in the x-axis and 0.0007 m RMSE in the y-axis. On the other hand, the estimated velocity state of the LKF filter obtained by fusion of position, velocity and attitude measurements has a larger deviation of 0.0038 m RMSE in the x-axis and 0.0030 m RMSE in the y-axis. The estimate does not reach the peak velocity values, even though the scan matching output does, which implies, that the covariance of velocity measurements in $\mathbf{R}_{vel}$ has to be decreased.

6.5 Summary

Based on the presented results from simulation flights we conclude, that the experiments have successfully proven the applicability of the localization and mapping system. The system was found to be reliable, accurate and robust to failed scan alignments, from which it always managed to recover when the UAV returned to a known part of the map. The position Kalman

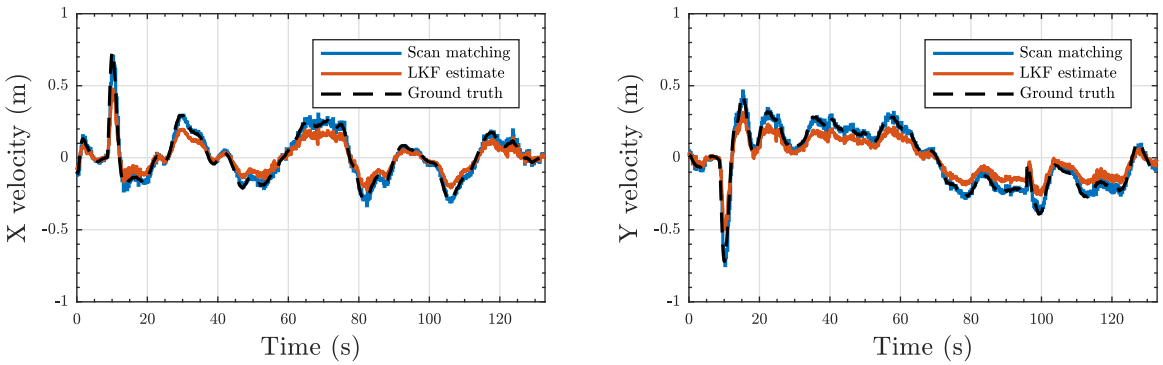


Figure 26: Velocity estimates from the system validation simulation run. Scan matching estimate is obtained from the sequential matcher module, which aligns subsequent scans to estimate relative displacement between them, from which the velocity is calculated. The LKF estimate is the velocity state obtained from the Linear Kalman Filter.

filtered position estimate is smooth, without harsh changes. It was confirmed, that the delay of the estimate does not destabilize the controller when the gains are soft, and the flying speeds are around 0.5 m/s. Hence, the pipeline is ready to be deployed on a real UAV.

Regarding the numerical evaluation, the positional RMSE from the ground truth was twice higher in one axis than in the other which points to the fact, that the direction of the higher error was worse defined by the obstacles than the one with the smaller error. The second thing which was found by the analysis is that the velocity measurement covariance matrix needs to be further tuned, due to the RMSE of the filtered estimate being worse than the RMSE of the scan matching output.

7 Experimental verification

Several experiments were conducted to analyze the functionality of the developed localization system. As the obtained position estimate is wired in a feedback loop to the SO(3) controller, it would be dangerous to test the system directly. Malfunctions of the localization system could result in damage to the objects in the environment, a crash of the UAV, or even injury to the people conducting the experiments. Of course, such risks had to be minimized. The testing procedure has started by conducting relatively simple experiments. Before moving to a next more demanding experiment, the functionality of each subsystem had to be demonstrated by passing all the previous tests successfully. To further reduce the chance of an accident, the testing was first realized in the Gazebo simulator. The results from simulation are presented in Section 6. Even though the conditions in the simulator are close to ideal (no wind, straight planar terrain with uniform height, ground truth position for comparison available), it provides a valuable estimate of the system reliability as well as bootstrap parameter values for the real UAV (gains, covariances, weights).

Despite one of the planned use cases being flying in historic buildings such as churches or castles, the majority of experiments were carried out outdoor, where a GPS position is available for comparison with the developed method. Also testing the system for the first time, the indoor environment would present a severe risk of damaging the ancient murals and statues. The experiments took place on a grass plane in the countryside in the South Bohemia region of the Czech Republic during a whole week-long camp in April 2018. Since the grass plane had no natural vertical features such as trees or buildings, an artificial environment had to be built. The first type of obstacles, simulating walls of an indoor environment, was realized by garden sails tightened between metal poles, which were hammered into the ground. Each wall segment (sail), measuring 2 meters in width and height, formed a part of a wall in a U-shaped room (Figure 27). The second obstacle type was a playground crawling tunnel mounted vertically on a metal pole to serve as a model of a tree. A side view of multiple tree models can be seen in Figure 28. The diameter of the tree model is 0.5 meters with the same height as the sails (2 meters). In Figure 29 the whole environment with all obstacles is captured from the aerial perspective, with GPS positions of individual obstacles denoted.

The first experiment was designed to evaluate the capabilities and limitations of the mapping module. The goal was to prove that the produced global map is an accurate representation of the environment with little noise and outliers. The second experiment was conducted to analyze the properties of position estimate from the localization subsystem. The last and most demanding experiment combines the localization and mapping into a complete system, which creates a global map while simultaneously providing a position estimate for the MPC tracker, without any prior knowledge about the environment. The parameters of used algorithms were not changed between the experiments. Unfortunately, the precise RTK GPS position which was meant to be used as ground truth for analyzing the accuracy of the localization system was not available due to communication errors and the GNSS receiver being partially occluded by the Sweep LIDAR. Nevertheless, the accuracy was already analyzed in simulation in Section 6 while the hardware experiments prove the usability of the developed system on a real platform.

Figure 27: A model of a U-shaped room built from garden sails. The purpose of the room is to simulate localization in a typical indoor environment.

Figure 28: A simulated forest used for the experiments. The models of the trees are formed by a playground crawling tunnel secured on a metal pole.

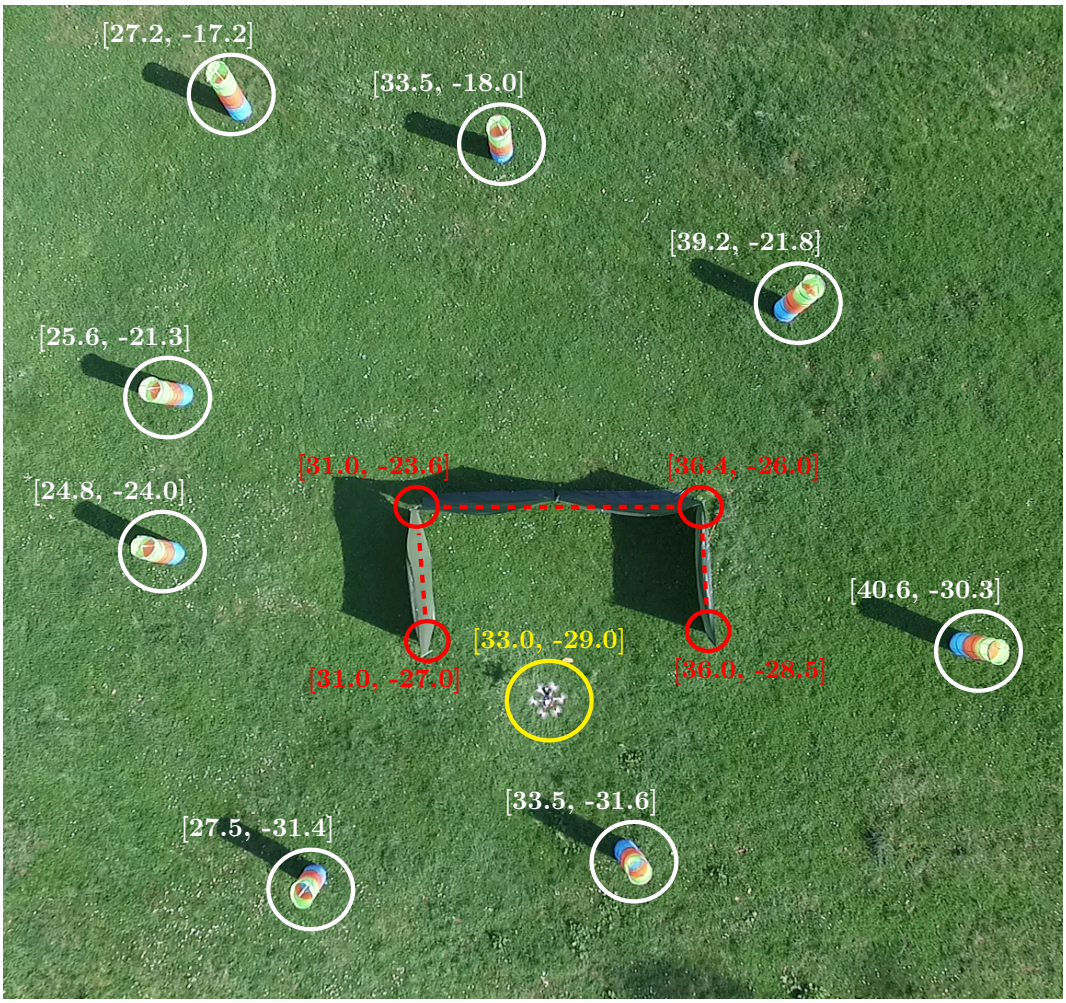


Figure 29: A top view of the experiment setup. The initial position is marked with a yellow circle and the tree and wall type obstacles by white and red respectively.

7.1 Mapping module analysis

The measurement of position in the *local_origin* frame is obtained by matching the current scan into the global map. As such, the map created by the mapping module must be detailed enough to estimate the position with the desired accuracy while simultaneously eliminating positional drift caused by imprecise matching into a too detailed map. Ideally, the mapping module should capture every static obstacle in the environment. Even though the thorough scan preprocessing and outlier rejection during matching can overcome occasional false positive measurements and noise, is is preferred that these unwanted points are not added to the map at all.

The experiment involved a UAV autonomously taking off from the initial position, then flying around through manually assigned setpoints with the controller using the precise UAV pose obtained by fusing PixHawk position estimate from GPS and IMU, with RTK corrections. With only sporadic RTK corrections, the UAV position did not follow the setpoints as accurately, which had no negative effect on this experiment, as the map was built based on the scan alignment instead of the UAV position. The flight trajectory led around all obstacles to integrate them into the map, and after passing the last obstacle, the UAV was manually landed on the takeoff spot.

To successfully build a map it is crucial to remove the points the do not belong into the map. The most notable undesirable points are points coming from the body of the drone or the propellers, the points on the ground, noise, and points external to the flight area. Removing these points is the responsibility of the scan processing module. The statistical evaluation of the number of points before and after processing along with the number of points removed in each removal stage can be seen in Figure 30. The mean number of points before and after processing is 210 and 49 respectively. Consequently, the total number of removed points is 161 on average, out of which 136 was removed by the close points removal subsystem, 5 by the ground removal, 16 points were classified as noise and 4 points were external to the flight area.

The global map is visualized in Figure 31. All eight trees were successfully detected and added to the global map by the mapping module. Inspecting the points corresponding to the U-shaped room, there are no gaps to be seen with all 3 wall segments correctly aligned. The only map inconsistency can be seen in the bottom left corner, where a line of red points is formed. These points appeared as a result of a manual landing in the last part of the experiment, during which the limits imposed on the maximal acceleration and tilt are not active. The combination of low altitude and tilt of the UAV resulted in a few laser beams hitting the ground. Usually, these are filtered out by the ground removal module, which in this specific case was not possible, as the terrain was sloped higher than in the rest of the area.

In the end, 29 out of 308 points of the map were false positives which leads to 93.6% precision. All obstacles were detected in this test flight, so the recall was 100%. Considering the testing conditions were not optimal (nonuniform slope of the terrain, presence of wind with changing gust speeds causing unwanted movements of both the UAV and obstacles), the mapping module performed surprisingly well and produced a consistent global map which is

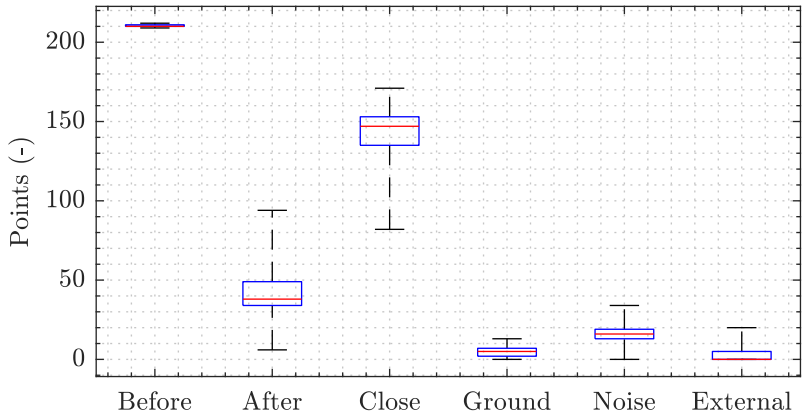


Figure 30: The number of points in the scan before and after scan processing together with the number of points removed by every module. *Close* is the number of points removed by the close points removal module, *ground* – number of points removed by the ground removal module, *noise* – points classified as noise, and *external* – points removed by the external removal module.

suitable for localization.

The mapping capabilities of the system were also verified during a manually controlled flight in the Stará Voda church. The trajectory of the flight led around the walls of the church and the UAV landed on the same spot from which it took off. The globally consistent dense point cloud from the experiment is captured in Figure 32 from various angles.

7.2 Localization verification

When a map is known in advance, the UAV can estimate its position by localizing itself in the map. Since the global map created by the mapping module was accurate and consistent, it can be used for localization. Compared to full SLAM, the isolated localization has the advantage of being able to recover from occasional failures. To allow a safe and smooth flight, the position estimate must fulfill specific requirements which are analyzed by this experiment.

1. For control of the UAV, it is necessary to acquire the position with minimal delay. Large delay can cause unwanted oscillations or even destabilize the UAV.

2. Since the UAV is flying in cluttered surroundings, a precise estimate is needed to avoid obstacles. Additional tasks involving interaction with the environment further increase the required precision.

3. Localizing in a global map can produce abrupt changes in the position when the algorithm suddenly determines it is in another part of the map. Such changes must be minimized as the controller expects a smooth estimate to work reliably.

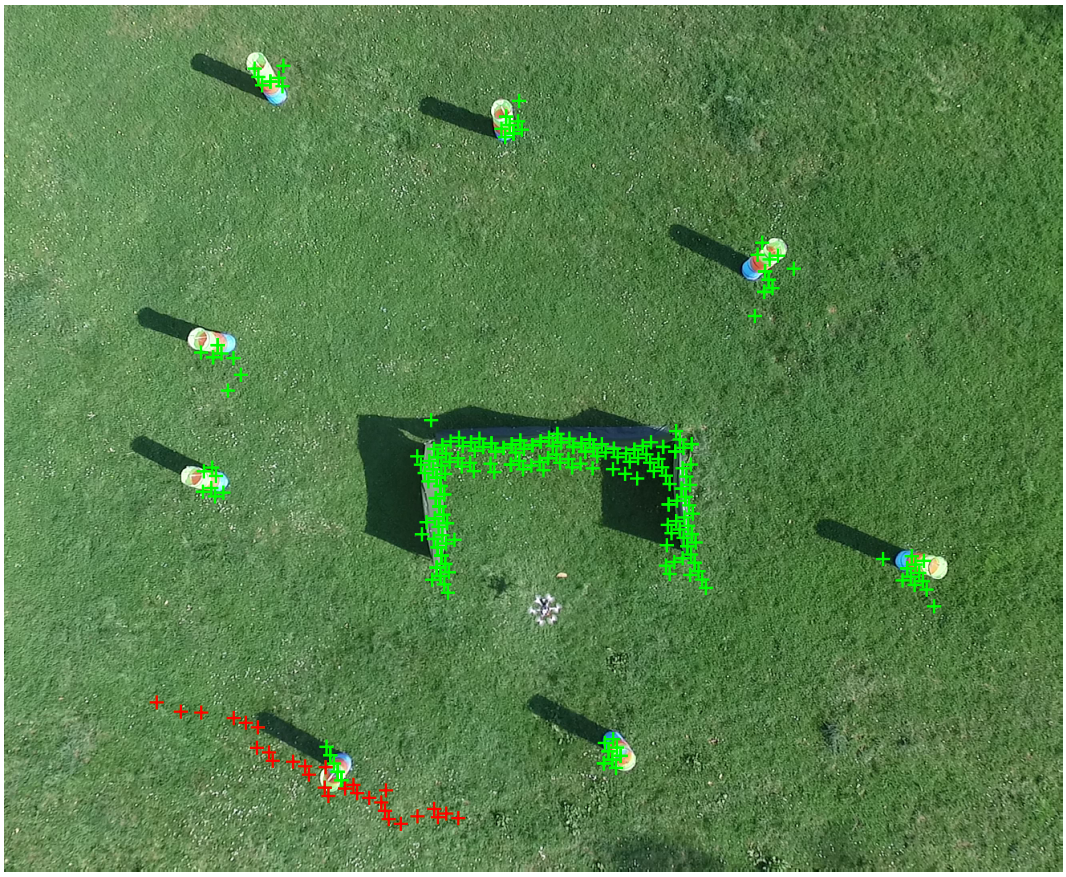


Figure 31: An aerial view of the experiment conducted for mapping module verification. The final map obtained after the whole experiment is visualized by crosses representing the global point cloud. The green color is used for correct points while outliers which should not be present are depicted by red. Some of the points do not seem to be perfectly aligned to the obstacles which is not an error of mapping but a result of the lens and perspective distortion.

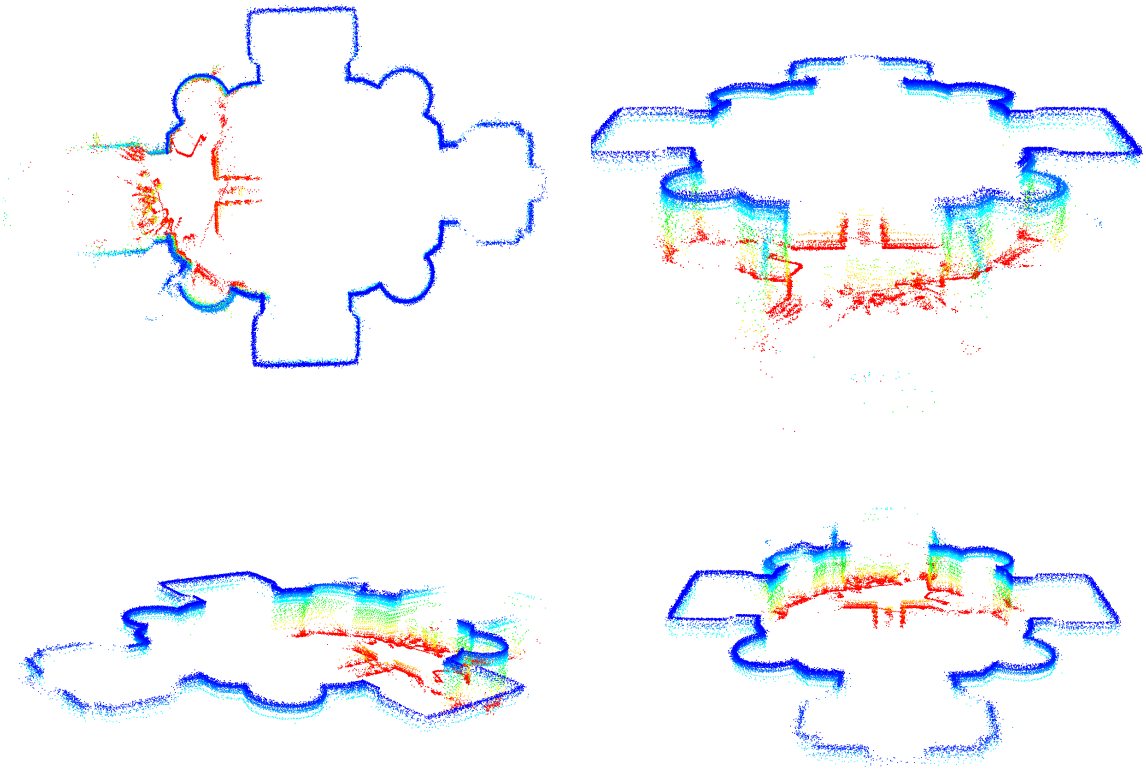


Figure 32: Views from different angles of the 3D dense point cloud obtained by aligning the laser scans from the Stará Voda church dataset. The points are color-coded to distinguish their altitude. The lowest red points were captured by the LIDAR during takeoff and landing. Most of the points are measurements taken from the flight altitude, which was around 4 m (blue color).

In this experiment, the UAV is commanded to fly through setpoints in the previously mapped area. The state estimate fed to the SO(3) controller, unlike in the first experiment, comes from the developed localization pipeline. Matching sequential scans provides an estimate of velocity, while the alignment of the scan into the global map produces a position estimate. Both of them are used as measurements in the Linear Kalman Filter (LKF) together with the orientation estimate from the IMU of PixHawk autopilot. The input to the LKF is the target attitude of the controller.

Unfortunately, the RTK was not receiving corrections from the base station during the experiments so the position estimate cannot be compared to the precise RTK position measurements. Even when the RTK system is not receiving corrections from the base station, it can still operate in Standard Positioning System (SPS) mode which is not as precise as RTK Fix mode. On the other hand, compared to the GPS in PixHawk, it can use GNSS satellites from more providers (GPS, GLONASS, Galileo, BeiDou).

The individual position estimates are plotted in Figure 33 for comparison. With the

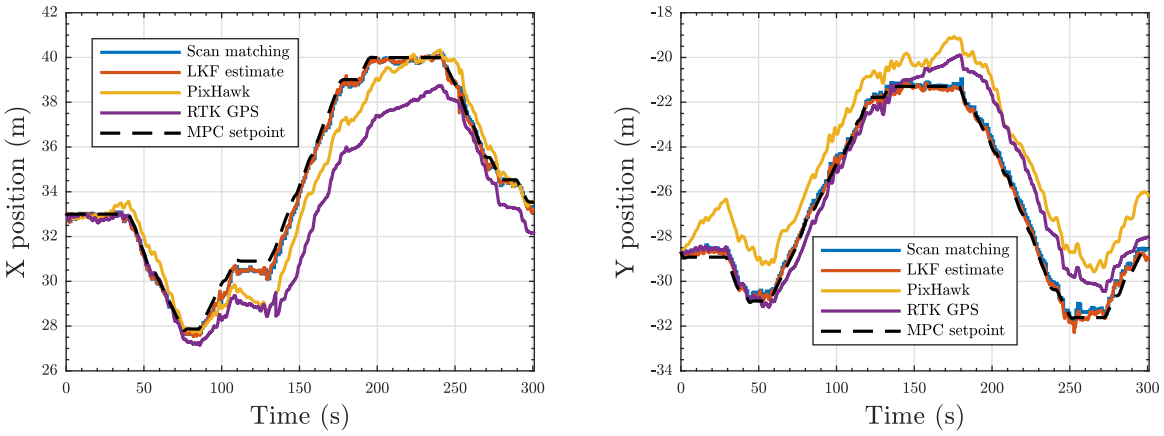


Figure 33: Position estimates from localization verification experiment. Scan matching is the estimate from matching scans to global map which after fusion of additional measurements (velocity and tilt) forms the LKF estimate. PixHawk estimate is a fusion of IMU with GPS. GNSS is the position reported by RTK working in the SPS mode (no corrections from the base station). No ground truth was available during the experiment, so at least MPC setpoint is used for reference.

absence of ground truth position measurements, the setpoints for the SO(3) controller provided by the MPC tracker were used as a reference. However, no conclusions regarding the accuracy of the proposed method can be made, as the position estimate will follow the MPC setpoint even when the estimate does not correspond to the actual position. The scan matching estimate is the raw output of the global matching module which takes a scan and aligns it with the global map. The obtained position changes abruptly after each successful matching which is performed every second. Abrupt change in position measurements are unsuitable for the controller, which needs a more frequent and smoother estimate. An LKF fuses the absolute position with velocities from the sequential matching module and attitude from IMU to provide a smoother estimate which is more responsive to fast changes in both position and velocity. Also, the update rate of the LKF estimate is 100 Hz which makes it much more compatible with the controller compared to the raw output from global matching module coming on 1 Hz, and output from sequential matching coming at 5 Hz. On a big scale, the scan matching and LKF estimates differences cannot be seen, so they are plotted once more in Figure 34 with a shorter time window of 20 seconds for analyzing the details. As seen from the PixHawk estimate, the fusion of GPS and IMU is insufficient for navigating in a tightly constrained environment due to the relatively large unpredictable drift of position over time. The GNSS generally follows the PixHawk estimate, with some deviations caused by more available satellites from multiple positioning services and the absence of IMU.

Figure 34 shows a detailed view of the position estimates to illustrate the difference between scan matching and LKF estimates. The fused LKF estimate is clearly much smoother and reacts to changes in the position more swiftly than the scan matching estimate. Furthermore, the measurements of velocity and tilt angle can even compensate short sequences of global matching errors as can be seen in the Y position plot of Figure 34 between 84 and 86 seconds.

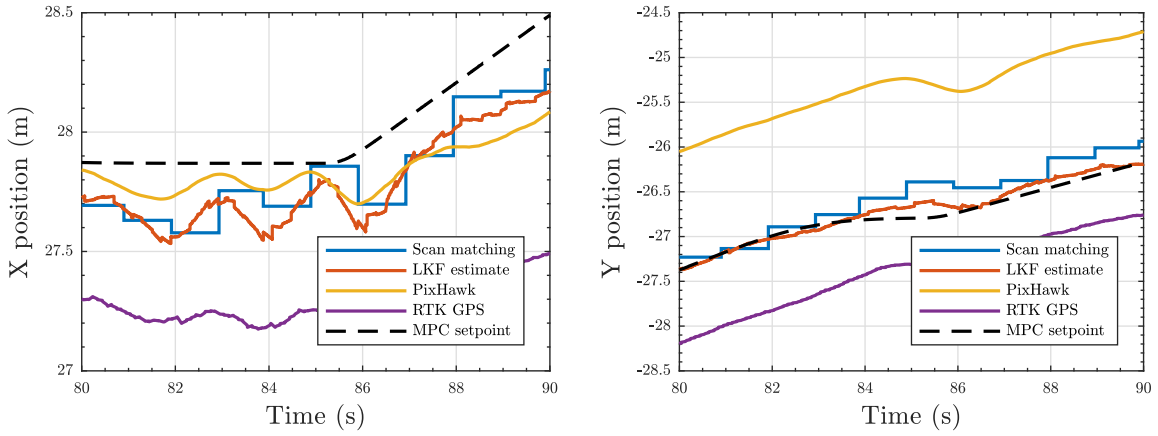


Figure 34: A detailed view of position estimates in a 20 seconds long window in the middle of the localization experiment. The sources of the plotted signals are the same as in Figure 33.

The UAV velocity is shown in Figure 35. Two interesting observations have been made in this figure. First, although the MPC tracker had a velocity constraint set to 0.2 m/s, which it satisfied, the real velocity was at least twice larger, which is most likely the result of the SO(3) controller attempting to follow the velocity reference from MPC. Second, after 250 seconds, a large deviation from the desired velocity occurred. The deviation is not present in the PixHawk estimate, so it was either an external disturbance filtered out by the PixHawk EKF, or the sequential matching module malfunctioned. In either case, the disturbance did not destabilize the system, which proves its robustness.

All depicted estimates are similar in frequent changes, which makes it difficult to analyze the small variations between them in a plot of the whole flight. A zoomed-in Figure 36 shows only 10 seconds from the same flight, but is much more detailed, so subtle differences can be seen. The LKF estimate closely follows the velocity from sequential matching, but smooths out larger peaks, making it more convenient for the controller. There also appears to be a lag of 200 ms after the PixHawk estimate which could be problematic for more aggressive flights with higher gains. Since the lag is caused mostly by the rotation frequency of the sensor, which is constant, it is possible to compensate it. A possibility is to use a Smith predictor or add the PixHawk velocity estimate as a new measurement. The compensation was not implemented within the scope of this thesis as the flight conditions (low velocities and small controller gains) did not require it. It is however planned as a future improvement of the system.

In conclusion, the experiment proved localization pipeline to be a reliable source of the position estimate for the controller. The UAV closely follows the MPC setpoint, and moreover, the localization system can recover even after a short sequence of wrong scan matches thanks to the global map and LKF. Despite the system being primarily designated for indoor and forest environments where GNSS reception is bad or nonexistent, it surpasses the performance of GNNS localization even on a plain with favorable satellite availability. As expected, the estimated velocity has a delay of around 200 ms which has to be compensated when considering more aggressive flying.

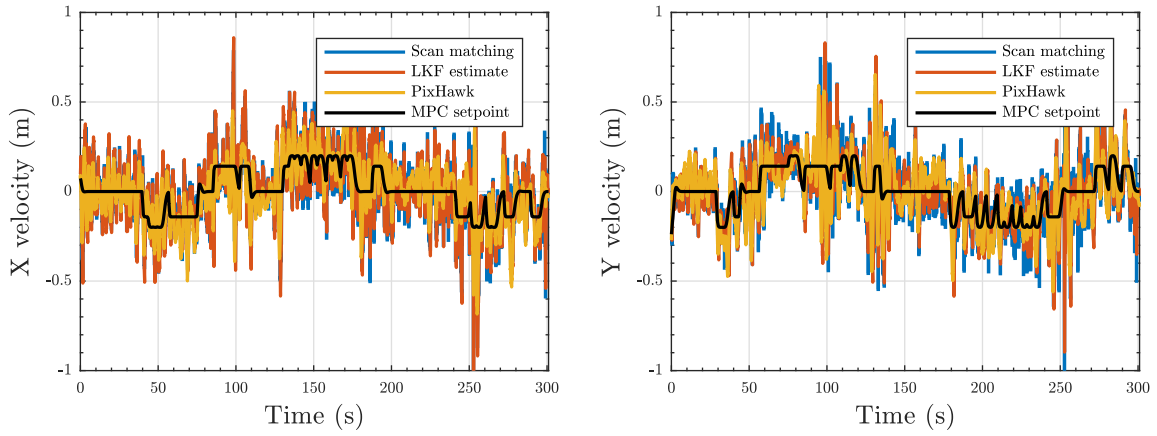


Figure 35: Velocity estimates from the localization verification experiment. Scan matching estimate is obtained from the sequential matcher module, which aligns subsequent scans to estimate relative displacement between them. The LKF estimate is one of the velocity states from LKF. PixHawk estimate is the output of onboard IMU.

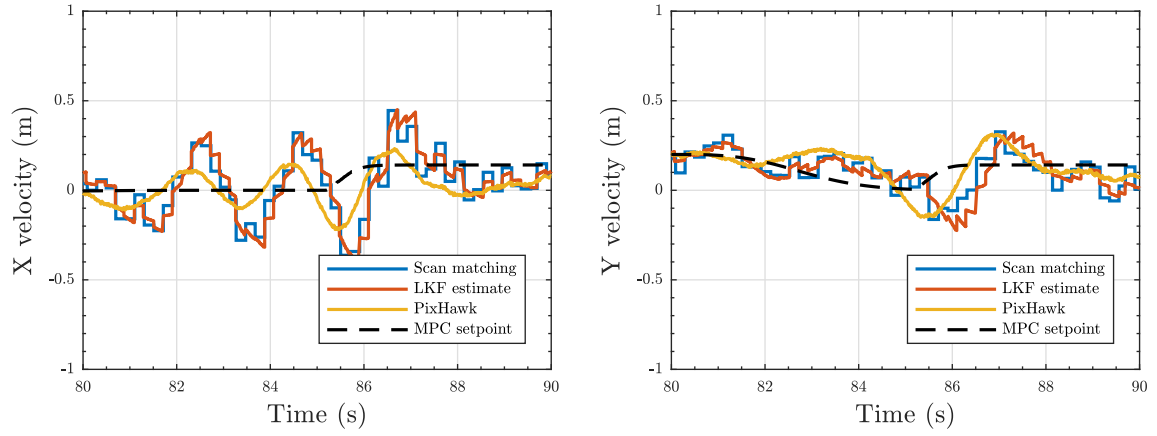


Figure 36: A detailed view of velocity estimates in a 20 seconds long window in the middle of the localization experiment. The sources of the plotted signals are the same as in Figure 35.

Figure 37: A top down view of the experiment. The image is stitched from a video with the UAV position snapshot taken every 10 seconds. The overlay shows, that the position estimates match the actual flight trajectory.

7.3 SLAM experiment

Previous experiments demonstrated the functionality of separate mapping and localization modules. When both systems are combined, a Simultaneous localization and mapping (SLAM) system is formed. SLAM is more challenging since a bad position estimate results in points added to the wrong place of the global map, which then in turn further degrades following localization. Thus, the system can easily get into a state from which it cannot recover.

This experiment takes place in the same environment as the previous two experiments. Even the setup of the experiment including the initial position and system parameters are identical. The only difference is a slightly altered sequence of flight setpoints. No prior knowledge about the environment is available at the start of the experiment, and the global map is empty.

Same as the other two experiments, there were no measurements of precise position

which could have been used as ground truth. To at least roughly show that the position estimate corresponds to the actual flight trajectory, a video from DJI Phantom was combined into a single stitched image. The moving UAV on a static background was captured every 10 seconds and stitched into the image of Figure 37. By plotting the position estimates and the setpoints over the image, it can be seen, that they match the real trajectory of the UAV. Although this verification method is too coarse to allow any qualitative analysis of the estimates, it can still serve as a proof of concept. The whole experiment is video-documented at `http://mrs.felk.cvut.cz/petrlik2018thesis`.

Similarly, as in the localization experiment, the position estimated by PixHawk is much less accurate than the developed method. The LKF position estimate closely follows the reference setpoint and does not seem to drift anywhere. Building the global map during the flight did not present any noticeable detrimental effect on the quality of the position estimate.

This experiment proved that the whole SLAM system is ready for deployment in GNSS denied environments. The environment featured models of obstacles from two expected operation environments - walls representing indoor rooms and trees simulating flying through a forest. Both of them were properly registered by the sensor and provided enough sufficient amount of features to stabilize the position estimate.

8 Conclusion

In this thesis, we have designed, implemented and verified a self-localizing solution for a UAV deployed in an unknown environment with limited access to Global Navigation Satellite Services (GNSS). Only onboard sensors, namely IMU, laser range-finder and rotating LIDAR, were used to obtain an accurate and precise position estimate. The proposed method does not impose any geometric assumptions about the surroundings and is therefore usable in both indoor and outdoor environments containing common obstacles.

We divided the localization system into individual modules by separating independent parts of the solution to facilitate further development and modification. These modules form a pipeline that estimates the current position of the UAV based on the stream of laser scans and IMU measurements. The individual steps of the localization pipeline are scan processing, sequential matching, global matching, mapping, Kalman filtering. A laser scan generated by the LIDAR is first processed in the scan processing module. After that, a transformation between the current and previous laser scans is found by a custom scan matching algorithm, which combines the advantages of different state-of-the-art methods. A velocity estimate is obtained from the transformation and the time gap between acquiring the two scans. The obtained velocity is fused in a linear Kalman filter with IMU measurements and a position estimate from the global matching module that aligns the current scan into the global map. The mapping module is responsible for maintaining a global map that accurately represents the environment, and it has to provide a present state of the map to the global matching module.

According to the thesis assignment, the following tasks have been successfully completed:

- A scan matching algorithm that finds the transformation between successive 2D laser scans was implemented.
- The obtained transformation was fused with data from the IMU and other sensors to provide an estimate of UAV local position.
- The self-localizing technique was integrated into the existing UAV system used in the MRS group.
- Experiments were conducted for establishing the performance and limitations of the designed localization technique in the realistic Gazebo simulator.
- The system was prepared for integration into the position feedback control loop of the UAV and tested in the Gazebo simulator.

The only exception was the last task:

- Test the usability of the complete system on datasets recorded during real-world flights with a precise GPS ground truth.

It was not possible to test the usability on datasets with a precise GPS ground truth since the precise RTK FIX mode of the GPS system was not available during the real-world flights. The unavailability of RTK FIX was caused by frequent communication errors with the base station. Nevertheless, the usability was tested on datasets recorded during real-world flights with regular GPS position and the qualitative evaluation was done on datasets with exact ground truth from the Gazebo simulator instead. The Kalman filtered position estimate from the developed method was found to be much more accurate than the regular GPS, judging from the aerial view of the real-world flight. In combination with less than 0.2 m RMSE in the simulation flight, it is sufficient proof that the system is suitable for deployment even in tight environments. Multimedia material related to this work is available on the website `http://mrs.felk.cvut.cz/petrlik2018thesis`.

In addition to the assigned tasks, the position estimate was used to close the feedback control loop beyond the assignment of this thesis. It is an valuable step towards the autonomy of the UAV. Since now, our UAV platform can be deployed even in locations with poor or without GNSS reception. This presents an opportunity for research of algorithms targeted explicitly to indoor locations. Furthermore, a globally consistent composite point cloud is generated as a by-product of building a global map for localization, which is of particular interest for historians and restorers of heritage buildings.

8.1 Future work

During the work on this thesis, several ideas for improvements emerged. The motivation for the developed solution was to substitute GPS localization in indoor areas. An arbiter that judges the quality of the GPS localization (the number of satellites, the mode of operation of the RTK), and the quality of scan-matching based localization (the number of points in laser scans, the number of outliers and the FRMSD value of correspondences, etc.) should be implemented. The arbiter should then seamlessly switch between the two localization systems based on the quality of their estimates. The platform would then be genuinely universal by allowing flights in combined indoor-outdoor environments, which is also one of the tasks proposed for MBZIRC 2019 [53].

A delay of position estimate is introduced by processing of the laser scans and also by scan matching algorithms. In the case of low-velocity flights with soft controller gains, the delay can be ignored, however, during aggressive flights, the stability of the UAV could be compromised. The solution could be to draw predictions from the already implemented Kalman filter or to implement a Smith predictor to allow higher velocities of the UAV.

The accuracy of localization depends on the number of points in the laser scan. In an environment with only sparse obstacles, it would be useful to have an additional source of measurements. Velocity estimate from optical flow should be fused with the scan matching measurements in the Kalman filter to improve the system robustness in situations with low obstacle density.

References

[1] M. Windolf, N. Götzen, and M. Morlock, "Systematic accuracy and precision analysis of video motion capturing systems—exemplified on the vicon-460 system," *Journal of biomechanics*, vol. 41, no. 12, pp. 2776–2780, 2008.

[2] M. Nitsche, T. Krajník, P. Čížek, M. Mejail, and T. Duckett, "Whycon: An efficent, marker-based localization system," in *IROS Workshop on Open Source Aerial Robotics*, 2015.

[3] T. Krajník, M. Nitsche, J. Faigl, T. Duckett, M. Mejail, and L. Přeučil, "External localization system for mobile robotics," in *16th International Conference on Advanced Robotics (ICAR)*, Nov 2013.

[4] T. Krajník, M. Nitsche, J. Faigl, P. Vaněk, M. Saska, L. Přeučil, T. Duckett, and M. Mejail, "A practical multirobot localization system," *Journal of Intelligent & Robotic Systems*, 2014.

[5] J. Faigl, T. Krajnik, J. Chudoba, L. Preucil, and M. Saska, "Low-cost embedded system for relative localization in robotic swarms," in *International Conference on Robotics and Automation (ICRA)*. IEEE, 2013, pp. 993–998.

[6] M. Saska, "Mav-swarms: Unmanned aerial vehicles stabilized along a given path using onboard relative localization," in *2015 International Conference on Unmanned Aircraft Systems (ICUAS)*, June 2015, pp. 894–903.

[7] V. Walter, M. Saska, and A. Franchi, "Fast mutual relative localization of uavs using ultraviolet led markers," in *Accepted to 2018 International Conference of Unmanned Aircraft System (ICUAS)*, 2018.

[8] D. Scaramuzza and F. Fraundorfer, "Visual odometry [tutorial]," *IEEE Robotics Automation Magazine*, vol. 18, no. 4, pp. 80–92, Dec 2011.

[9] A. Wehr and U. Lohr, "Airborne laser scanning—an introduction and overview," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 54, no. 2, pp. 68 – 82, 1999.

[10] H. C. Lee, S. H. Lee, S. H. Lee, T.-S. Lee, D.-J. Kim, K.-S. Park, K.-W. Lee, and B. H. Lee, "Comparison and analysis of scan matching techniques for cooperative-slam," in *2011 8th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, Nov 2011, pp. 165–168.

[11] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1, pp. 185 – 203, 1981.

[12] D. Fleet and Y. Weiss, *Optical Flow Estimation*. Boston, MA: Springer US, 2006, pp. 237–257.

[13] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679.

[14] 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, Feb 1992.

[15] Scanse, "Sweep v1 LIDAR," 2017, (cited on 2018-05-24). [Online]. Available: https://s3.amazonaws.com/scanse/Sweep_user_manual.pdf

[16] J. S. Gutmann, W. Burgard, D. Fox, and K. Konolige, "An experimental comparison of localization methods," in *Proceedings. 1998 IEEE/RSJ International Conference on Intelligent Robots and Systems. Innovations in Theory, Practice and Applications (Cat. No.98CH36190)*, vol. 2, Oct 1998, pp. 736–743 vol.2.

[17] T. Báča, "Model predictive control of micro aerial vehicle using onboard microcontroller," Master's thesis, Czech technical university in Prague, Czech Republic, 2015.

[18] 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)*, June 2016, pp. 649–656.

[19] F. WANG, J.-Q. CUI, B.-M. CHEN, and T. H. LEE, "A comprehensive uav indoor navigation system based on vision optical flow and laser fastslam," *Acta Automatica Sinica*, vol. 39, no. 11, pp. 1889 – 1899, 2013.

[20] M. Montemerlo and S. Thrun, *FastSLAM: A Scalable Method for the Simultaneous Localization and Mapping Problem in Robotics (Springer Tracts in Advanced Robotics)*. Berlin, Heidelberg: Springer-Verlag, 2007.

[21] J. Q. Cui, S. Lai, X. Dong, P. Liu, B. M. Chen, and T. H. Lee, "Autonomous navigation of uav in forest," in *2014 International Conference on Unmanned Aircraft Systems (ICUAS)*, May 2014, pp. 726–733.

[22] S. Thrun and M. Montemerlo, "The graph slam algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

[23] E. B. Olson, "Real-time correlative scan matching," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 4387–4393.

[24] P. Biber and W. Strasser, "The normal distributions transform: a new approach to laser scan matching," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 3, Oct 2003, pp. 2743–2748 vol.3.

[25] DJI, "E310 Multirotor Propulsion System," 2014, (cited on 2018-05-24). [Online]. Available: http://dl.djicdn.com/downloads/e310/en/E310_User_Manual_v1.0_en.pdf

[26] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys, "Pixhawk: A micro aerial vehicle design for autonomous flight using onboard computer vision," *Autonomous Robots*, vol. 33, no. 1, pp. 21–39, Aug 2012.

[27] Tersus-GNSS, "PRECIS-BX305 GNSS RTK Board," 2017, (cited on 2018-05-24). [Online]. Available: https://www.tersus-gnss.com

[28] Matrix-Vision, "mvBlueFox-MLC," 2016, (cited on 2018-05-24). [Online]. Available: https://www.matrix-vision.com/USB2.0-single-board-camera-mvbluefox-mlc.html?file=tl_files/mv11/support/mvBlueFOX/Datasheets/mvBlueFOX-IGC_technical_details_2016-02.pdf

[29] 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," 2017, (submitted to the special issue on "MBZIRC 2017 - Challenges in Autonomous Field Robotics").

[30] P. Štěpán, T. Krajník, M. Petrlík, and M. Saska, "Vision techniques for on-board detection, following and mapping of moving targets," 2017, (submitted to the special issue on "MBZIRC 2017 - Challenges in Autonomous Field Robotics").

[31] T. Krajnik, M. Nitsche, J. Faigl, P. Vanek, M. Saska, L. Preucil, T. Duckett, and M. Mejail, "A practical multirobot localization system," *Journal of Intelligent & Robotic Systems*, vol. 76, no. 3-4, pp. 539–562, 2014.

[32] M. Saska, V. Krátký, V. Spurný, and T. Báča, "Documentation of dark areas of large historical buildings by a formation of unmanned aerial vehicles using model predictive control," in *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, Sept 2017, pp. 1–8.

[33] Intel, "Realsense R200," 2016, (cited on 2018-05-24). [Online]. Available: https://software.intel.com/sites/default/files/managed/d7/a9/realsense-camera-r200-product-datasheet.pdf

[34] G. Loianno, V. Spurny, T. Baca, J. Thomas, D. Thakur, D. Hert, R. Penicka, T. Krajnik, A. Zhou, A. Cho, M. Saska, and V. Kumar, "Localization, grasping, and transportation of magnetic objects by a team of mavs in challenging desert like environments," *IEEE Robotics and Automation Letters*, 2018, (accepted to RA-L and ICRA).

[35] NicaDrone, "OpenGrab EPM v3 electropermanent magnet," 2017, (cited on 2018-05-24). [Online]. Available: http://nicadrone.com

[36] V. Spurný, T. Báča, M. Saska, R. Pěnička, T. Krajník, G. Loianno, J. Thomas, D. Thakur, and V. Kumar, "Cooperative autonomous search, grasping and delivering in a treasure hunt scenario by a team of uavs," 2017, (submitted to the special issue on "MBZIRC 2017 - Challenges in Autonomous Field Robotics").

[37] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[38] T. Baca, D. Hert, G. Loianno, M. Saska, and V. Kumar, "Model predictive trajectory tracking and collision avoidance for reliable outdoor deployment of unmanned aerial vehicles," 2018, (submitted to IEEE Robotics and Automation Letters).

[39] T. Baca, G. Loianno, and M. Saska, "Embedded model predictive control of unmanned micro aerial vehicles," in *2016 21st International Conference on Methods and Models in Automation and Robotics (MMAR)*, Miedzyzdroje, Poland, 2016.

[40] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, May 2015, pp. 6235–6240.

[41] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *49th IEEE Conference on Decision and Control (CDC)*, Dec 2010, pp. 5420–5425.

[42] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2520–2525.

[43] J. M. Phillips, R. Liu, and C. Tomasi, "Outlier robust ICP for minimizing fractional RMSD," *CoRR*, vol. abs/cs/0606098, 2006.

[44] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: An efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, 2013, software available at http://octomap.github.com.

[45] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 1–4.

[46] M. Brown and D. G. Lowe, "Automatic panoramic image stitching using invariant features," *International Journal of Computer Vision*, vol. 74, no. 1, pp. 59–73, Aug 2007.

[47] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *ACM Comput. Surv.*, vol. 38, no. 4, Dec. 2006.

[48] Y. Ke, X. Tang, and F. Jing, "The design of high-level features for photo quality assessment," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, June 2006, pp. 419–426.

[49] F. Lu and E. Milios, "Robot pose estimation in unknown environments by matching 2d range scans," *Journal of Intelligent and Robotic Systems*, vol. 18, no. 3, pp. 249–275, Mar 1997.

[50] M. Kam, X. Zhu, and P. Kalata, "Sensor fusion for mobile robot navigation," *Proceedings of the IEEE*, vol. 85, no. 1, pp. 108–119, Jan 1997.

[51] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," vol. 3068, 02 1999.

[52] A. Dubeň, "Motion planning with hermite splines for unmanned aerial vehicle in information gathering task," Master's thesis, Czech Technical Univeristy, Czech Republic, 2018.

[53] MBZIRC, "Mohamed Bin Zayed International Robotic Challenge 2019," (cited on 2018-05-24). [Online]. Available: https://www.mbzirc.com/challenge/2019

Appendix A CD Content

In Table 3 are listed names of all root directories on CD.

Directory name	Description
thesis	the thesis in pdf format
thesis_sources	latex source codes
src	source codes of the implemented solution
pcd	pcd files containing point clouds
videos	videos from experiments

Table 3: CD Content

Appendix B List of abbreviations

In Table 4 are listed abbreviations used in this thesis.

Abbreviation	Meaning
UAV	Unmanned Aerial Vehicle
MAV	Micro Aerial Vehicle
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
DGPS	Differential Global Positioning System
RTK	Real Time Kinematic
ESC	Electronic Speed Control
SPS	Standard Positioning Service
LIDAR	Light Detection and Ranging
CPU	Central Processing Unit
IMU	Inertial Measurement Unit
GPU	Graphical Processing Unit
DoF	Degrees of Freedom
PCA	Principal Component Analysis
SLAM	Simultaneous Localization and Mapping
MPC	Model Predictive Control
LTI	Linear Time-Invariant
MRS	Multi-robot Systems
MBZIRC	Mohamed Bin Zayed International Robotics Challenge
FCU	Flight Control Unit
RAM	Random Access Memory
USB	Universal Serial Bus
PC	Personal Computer
SSD	Solid State Drive
ROS	Robotic Operating System
SITL	Simulation In The Loop
IR	Infrared
UV	Ultraviolet
LED	Light-emitting Diode
ICP	Iterative Closest Point
IMRP	Iterative Matching Range Point
IDC	Iterative Dual Correspondence
RMSE	Root Mean Square Error
FRMSD	Fractional Root Mean Squared Distance
LMS	Linear Mean Square
LKF	Linear Kalman Filter
EKF	Extended Kalman Filter
PCL	Point Cloud Library

Table 4: Lists of abbreviations