POLITECNICO DI TORINO

Master's Degree in Mechatronic Engineering

Master's Degree Thesis

LiDAR SLAM for indoor exploration using Unmanned Aerial Vehicles

Supervisors

Prof. Alessandro RIZZO

Dr. Phillipp FANTA-JENDE

Candidate

Francesco VULTAGGIO

July 2023

Abstract

The role of Unmanned Aerial Vehicles, *UAVs*, in the context of disaster response has usually been to extend the senses of the responders by providing a bird's eye view of the affected area. In contrast, the role of indoor exploration has often been the purview of Unmanned Ground Vehicles, *UGVs*, since their greater payload allowed them to use the heavier sensors needed for such tasks. The recent development of lighter, cheaper, and more accurate sensors has opened the possibility of deploying small UAVs to map indoor spaces. UAVs offer many advantages over UGVs, they are more agile and quicker, they can fly over obstacles that may be insurmountable for UGVs, and reach structures inaccessible from the ground.

The goal of this work has been to identify the best hardware and software setup to perform Simultaneous Localization And Mapping, *SLAM*, in indoor spaces using a small UAV. To this end we performed an extensive battery of tests in simulated environments using a custom plugin we developed for this purpose. Having selected the most promising solution we then implemented drift correction strategies to increase the robustness and accuracy of the solution.

Acknowledgements

Working on this project has been the most enriching experience of my academic career thus far, and its success owes itself to the invaluable guidance and support of several individuals and organizations. I would like to express my deepest gratitude to my primary supervisors, Dr. Phillipp Fanta-Jende and Prof. Alessandro Rizzo, whose profound expertise in the field of computer vision and robotics played a pivotal role in shaping the direction of my research. Their mentorship not only helped me make informed decisions but also encouraged me to explore every research question in great depth.

I am indebted to the entire team at the AIT for their collaborative spirit and support. Dr. Francesco D'Apolito, Dr. Felix Bruckmüller, and Dr. Christoph Sulzbachner, in particular, have been instrumental in this journey. Their willingness to assist, provide valuable insights, and engage in discussions enriched my research experience. I must also acknowledge the exceptional teamwork of Wolfgang Boltz and Noah Maikisch, who patiently accommodated my requests for new data and served as valuable sounding boards for my ideas.

A special thanks goes to my dear friend, Marco Cella, who worked on the path planning aspects of our project. The countless late nights spent together in the office and our shared triumphs and challenges have not only forged a strong professional partnership but also a lasting friendship.

Beyond my academic network, my family has been an unwavering source of inspiration and motivation throughout this long and rewarding journey. My parents, in particular, have been a constant pillar of support, encouraging me to pursue my passions, from Palermo to Turin and now Vienna. While many people shaped my academic journey they made me the person I am.

As I look forward to the next phase of my research I carry with me the lessons, support, and inspiration from these remarkable individuals and organizations.

"Wir müssen wissen, wir werden wissen"
D. Hilbert

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Robotics for Disaster Response

In times of natural or man-made disasters, the ability to quickly and effectively respond to the crisis can make a significant difference in saving lives and minimizing damage. Traditional disaster response efforts often face challenges such as limited resources, time constraints, and hazardous conditions that put human responders at risk. However, with the rapid advancement of robotics technology, a new wave of innovative solutions has emerged to augment and improve disaster response operations.

By utilizing robotics in disaster response, response teams can enhance their capabilities, improve efficiency, and mitigate risks. Robots can operate in hazardous conditions without endangering human lives, overcome physical limitations, and provide real-time data for informed decision-making. The integration of artificial intelligence and further enhances their autonomy and adaptability, enabling them to learn and respond dynamically to changing circumstances.

According to [1], as of 2012, rescue robots have been used in the aftermath of at least 28 disasters worldwide. The first ever deployment of robots in the context of disaster response dates back to 1986 in the aftermath of the Chernobyl nuclear disaster[2]. Some robots were remote operated machinery used to remove debris while other were smaller in size and used for remote sensing through cameras and radiation dosimeters 1.1.

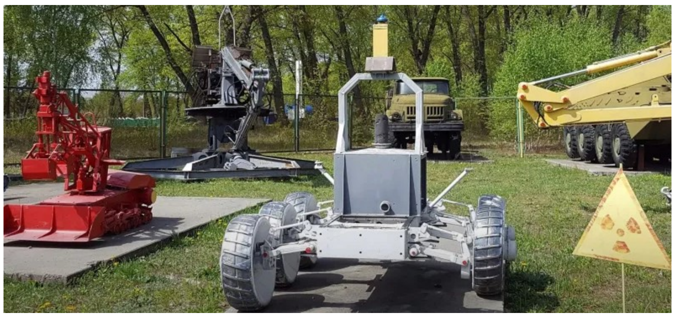


Figure 1.1: Some of the robots used in Chernobyl, displayed at the Chernobyl Museum in Kyiv

Another prominent use of robots, often cited as the first ever, occurred following the 2001 World Trade Center attacks, where small UGV (Unmanned Ground Vehicle) were used to explore portions of the fallen structure too narrow for humans to traverse.

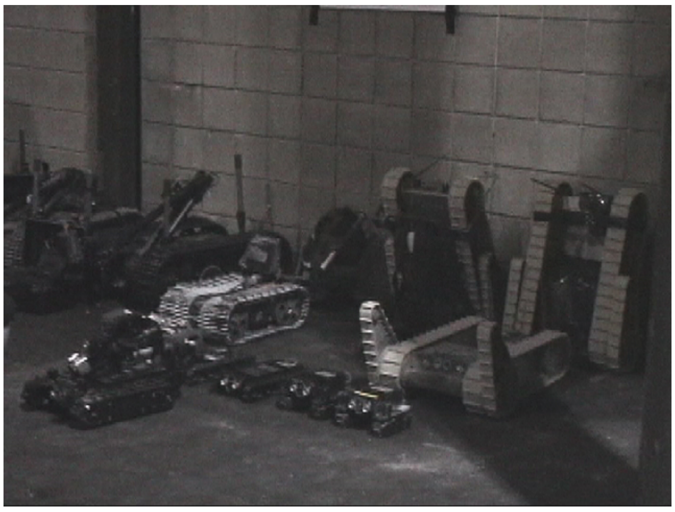
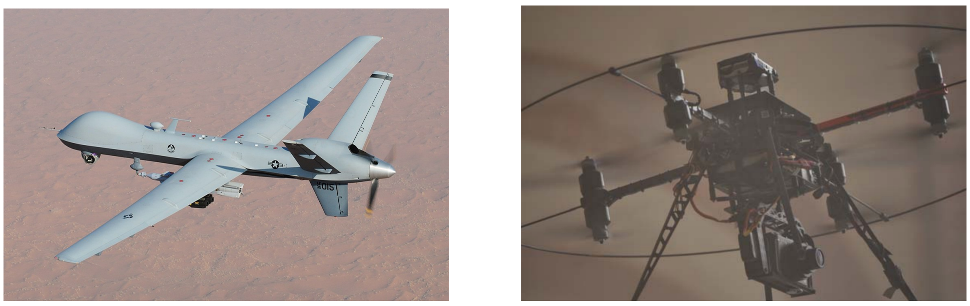


Figure 1.2: UGVs used after the World Trade Center collapse

UGVs are designed to operate in various terrains and environments, including rugged landscapes, collapsed structures, and contaminated areas. UGVs are able to carry heavier payloads such as manipulator arms, heavy sensors and powerful processing units. Historically, this has allowed UGVs to perform tasks such as delivering supplies or conducting remote inspections.

Whilst UGVs offer many advantages one key limitation stems from their inability to explore areas inaccessible from the ground and to offer a bird's eye view of an area. To tackle this limitation UAV (Unmanned Aerial Vehicle)s are often deployed alongside them.

UAV is a broad term used to describe aerial platforms with drastically different capabilities. Figure 1.3 shows two UAVs used for disaster response[3, 4]. UAVs are often categorised based on their configuration, i.e. fixed wing and multi rotor. However in this work UAVs will be divided based on their main domain of of operation: *outdoor* and *indoor* UAVs.



(a) Reaper UAV **(b)** Micro UAV

Figure 1.3: Example of UAV used in disaster response

Outdoor UAV have been used for decades for mapping and remote sensing applications[5, 6]. Recent advancements have led to the development of more compact UAVs suitable for indoor exploration of precarious structures in the aftermath of disasters.

Researchers in the field of disaster response have stressed the importance that autonomy plays in reducing the cognitive load for first responders. The disaster robotics community has primarily focused on three domains [7], namely:

- Exploration, planning, and path execution
- Object recognition and scene interpretation
- Localization
- Mapping

A fully autonomous platform capable of operating in disaster scenarios should excel in all of these domains, accomplishing goals with minimal human interaction through high-level commands. Outdoor UAVs have already reached a high level autonomy, being able to fly and create a bird's eye view map to first responders with minimal human input[8, 9]. The same cannot be said for indoor UAVs which so far have relied on human guidance, mediated by a live video feed, to traverse the inherently hostile and unstructured environments typical of disaster areas.

Indoor UAVs trade flight time and payload for maneuverability. To achieve a high degree of maneuverability indoor UAV often rely on multi-rotor configurations. Due to their reduced payload such systems have historically been reliant on low weight sensors, mainly IMU (Inertial Measurement Unit) and cameras.

This work will focus on improving the localization and real-time mapping capabilities of autonomous UAVs in indoor environments. While indoor localization and mapping have already reached a high degree of maturity[10, 11] most solution don't focus on negotiating with harsh environments. The design choices will be motivated by the assumption of an uncooperative environments, i.e.:

- No prior map of the environment
- Absent or inconsistent lighting
- Fog or other suspended particulate
- Aggressive motion profiles
- Limited to no access to external positioning system

1.1.1 The challenges of indoor localization

GNSS (Global Navigation Satellite System) signals, such as those provided by the American GPS (Global Positioning System) or the European Galileo constellations, are instrumental in facilitating autonomous mapping for outdoor UAVs. These signals allow outdoor UAVs to accurately determine their precise location and orientation in real-time. By combining GNSS data with onboard sensors, such as IMUs and cameras, UAVs can create detailed maps of the environment while navigating autonomously. GNSS signals provide a reliable and globally available source of positioning information, allowing UAVs to efficiently localize themselves even under adverse conditions.

The availability of GNSS signals simplifies the process of mapping for outdoor UAVs in disaster scenarios. Instead of relying solely on visual information, which can be challenging in chaotic and dynamically changing environments, UAVs can utilize GNSS signals to establish a reference frame and anchor their mapping efforts. This helps overcome the limitations posed by adverse weather conditions, low lighting, or obscured visual cues. GNSS signals provide a robust and independent source of positioning information, allowing UAVs to adapt to various environmental challenges and maintain accurate mapping capabilities.

Unfortunately, GNSS data is not a reliable source of positioning for indoor platforms [12]. We can identify many reasons for why this may be the case, see Figure 1.4.

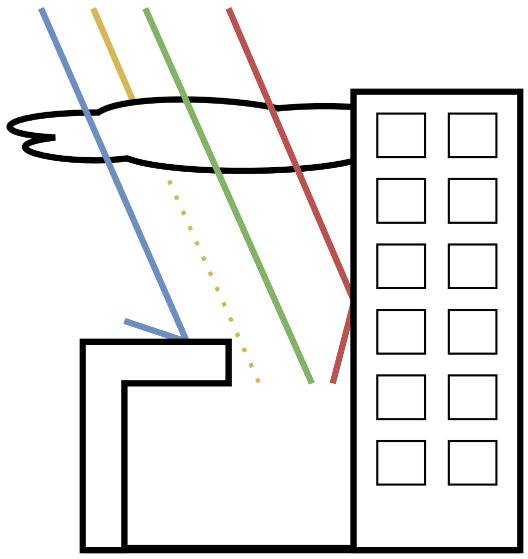


Figure 1.4: From left to right, reflection, attenuation, correct signal, multi-path

Firstly, indoor environments typically have limited or no direct line-of-sight with GNSS satellites this leads to reflection of the signal. Moreover, the signals from these satellites are easily attenuated or completely blocked by structures, walls, and ceilings, resulting in weak or no reception. As a result, the UAV cannot receive accurate GNSS signals indoors, leading to significant errors or complete failure in positioning.

Secondly, the multi-path effect, where GNSS signals reflect off surfaces before reaching the receiver, is more pronounced indoors. In indoor environments with reflective surfaces, such as glass windows or metallic objects, the UAV may receive multiple signal reflections, causing interference and leading to inaccurate position calculations. This multi-path effect further deteriorates the reliability of GNSS data for indoor positioning. These phenomenons are often encountered also outdoor, especially in urban canyons, but they are never as pronounced as indoor.

To address these challenges, alternative positioning technologies are used for indoor UAVs. These technologies include but are not limited to: IPS (Indoor Positioning Systems) [13] based on Wi-Fi, Bluetooth, or other wireless signals. These techniques rely on sensors and data sources that are specifically designed for indoor environments, offering higher accuracy and reliability compared to GNSS in such settings. IPS systems can offer sub centimeter accuracy when properly deployed and are often used to to generate ground truth positions in benchmark efforts[14].

However, there are several limitations that make IPS less suitable for disaster response. IPS heavily relies on infrastructure such as wireless beacons, which may become damaged or inaccessible during a disaster, rendering the system ineffective. Moreover, IPS accuracy can still be affected by factors like: signal interference, multi-path effects, and signal attenuation in complex indoor environments. These challenges hinder the reliability and robustness of IPS during critical situations. In the last decade a new avenue of research sitting at the crossroad between robotics and computer vision has proposed a software solution to the localization and mapping problem: SLAM (Simultaneous Localization And Mapping).

SLAM offers a more promising approach for localization in indoor environments during disasters. SLAM algorithms enable autonomous systems to simultaneously construct a map of the environment and estimate their own position within it, without relying on pre-existing infrastructure. This self-contained nature of SLAM makes it highly adaptable to dynamic and challenging scenarios, allowing robots to navigate and localize themselves even in the absence of reliable external references. SLAM's ability to handle uncertainties and adapt to changing environments makes it a better-suited approach for the localization problem in indoor settings during disasters. Moreover, SLAM algorithms are able to provide a map of the environment in real time, Figure 1.5. Such maps can be highly beneficial in the decision making process of the first-responders[7].

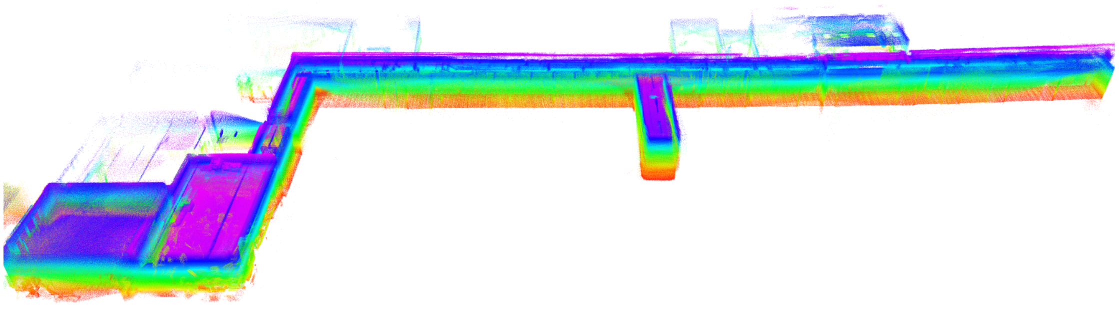


Figure 1.5: Example of 3D map generated in real-time by our system

Chapter 2

Foundations for SLAM

2.1 Formulating the problem

The SLAM problem has been described as "*deceptively easy to state*"[15]. As the name suggest, SLAM algorithms aim to acquire a spatial model of the environment and localize the sensing apparatus within said model in real time. Starting from a position x_0 the sensing apparatus roams the environment, if the motion is known we assume this information to be affected by noise. While it moves, the apparatus collects noisy data regarding the environment, leveraging these data the SLAM framework estimates its position and incrementally builds a map of the environment.

We can provide a mathematical formulation of the SLAM problem using a probabilistic framework. We denote the sequence of locations as:

$$X(T) = \{x_0, x_1, ..., x_T\}$$

We denote with T some terminal time at which we assume our observations to end. The location x_0 is assumed to be known by arbitrarily fixing it as the origin of our local frame of reference.

The set of measurements and controls can be similarly be represented with the vectors $Z(T)$ and $U(T)$ respectively. Denoting with m the map itself we can then formulate the SLAM problem as:

$$p(x_t, m|z_{1:t}, u_{1:t})$$

Borrowing the terminology used in [16] we can distinguish between two versions of the SLAM problem. The previous formulation is called *online* SLAM. Given that we only seek to estimate the pose at time t. On the other hand, we could formulate the SLAM problem also as:

$$p(x_{1:t}, m|z_{1:t}, u_{1:t})$$

This second formulation can be denoted as the *full* SLAM problem, since at each iteration t we seek to estimate the full trajectory spanning $\{0,1,...t\}$. Tackling the full SLAM problem is more challenging than the online SLAM problem due to several reasons.

First, in the online SLAM formulation, the objective is to estimate the pose at a specific time, which simplifies the problem. In contrast, the full SLAM problem aims to estimate the entire trajectory spanning from the initial time step to the current time step, encompassing all intermediate poses. This extended scope increases the complexity of the problem as it requires considering and integrating a larger amount of information.

Second, in the online SLAM problem, the estimation can be updated incrementally as new measurements and controls are obtained, allowing for a more efficient and real-time processing. The full SLAM problem, however, requires maintaining and updating the entire trajectory history as new data becomes available, leading to a growing computational burden and memory requirements. The accumulation of errors over time can also impact the accuracy of the estimates in the full SLAM formulation, as any errors in previous poses can propagate and affect subsequent estimations.

Furthermore, the full SLAM problem poses additional challenges when it comes to data association. Data association refers to the task of correctly associating measurements with the corresponding features or landmarks in the environment. In the online SLAM formulation, the associations can be resolved incrementally, relying on the previously estimated poses. In the full SLAM problem, however, the associations need to be consistently maintained throughout the entire trajectory, considering all past and future measurements. This requires addressing the data association problem in a global and coherent manner, which can be more difficult due to the increased complexity and potential ambiguities.

We will now present two different paradigms to tackle the SLAM problem. Many SLAM algorithms fall into one of these paradigms which represent two ends of the spectrum of how to tackle the problem.

2.2 Filter-based approaches

EKF (Extended Kalman Filter) based approaches have historically been the first paradigm explored by the robotics community to tackle the online SLAM problem [17, 18]. Later, more complex filtering approaches, such as those based on particle filtering [19, 20], gained popularity due to their ability to handle more extreme motion profiles and localization ambiguities.

2.2.1 Extended Kalman Filter SLAM

In the EKF formulation the map is assumed to be comprised of distinct features, which can be imagined as one dimensional points and are often referred to as *landmarks*. The *belief* we are trying to estimate will consequently be constituted by the coordinates of the sensing apparatus and the coordinates of each landmark in the map. If we assume our system to move and sense in a 3D space our model will be:

$$y_t = [x_t, y_t, z_t, r_t, p_t, y_t, m1_x, m1_y, m1_z, ..., mN_x, mN_y, mN_z]$$

The EKF paradigm represents the belief as a multivariate Gaussian distribution:

$$p(y_t|Z(T), U(T)) = \mathcal{N}(\mu_t, \Sigma_t)$$

Indicating by N the number of landmarks constituting our map, the size of μ_t is $3N + 6$ while the size of the covariance matrix Σ_t would be its square. The covariance matrix quantifies the uncertainty about the position of the robot and each landmark in the space, the off-diagonal elements capture the cross correlations in the estimate of the different variables.

As the platform moves, the uncertainty represented by the covariance matrix Σ_t will grow over time. However, when the platform observes again a previously mapped landmark the overall uncertainty will decrease. This is due to the correlation expressed in the covariance matrix of the Gaussian. This phenomenon is illustrated in Figure 2.1.

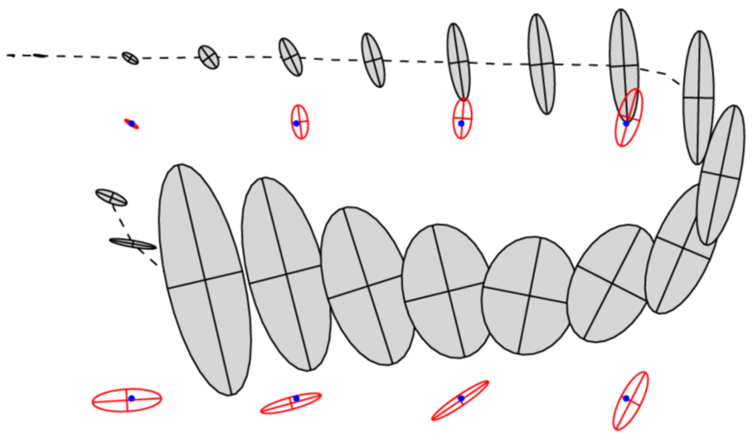


Figure 2.1: The dotted line shows the path of the platform and the dots represent the landmarks position. The shaded ellipses represent the position estimate of the platform while the red ones the estimate of the landmark. As the platform moves the uncertainty grows until, after re-observing the original landmark, the EKF corrects the accumulated drift[16]

The main limitation of the EKF formulation of the SLAM problem is its inherent poor scalability which stems from the quadratic growth of the covariance matrix. Over the years many approaches have been deployed to tackle this problem, some researchers formulated the problem using the inverse of the covariance matrix, called *information matrix*, which by it's very nature is sparse and thus amenable to efficient inversion algorithms[21, 22]. While others still have been able to define a measurement model which expresses the geometric constraints between the landmark poses without having to include the landmark position in the state vector, resulting in linear complexity over the number of features[23] but with cubic complexity over the number of old poses.

While filtering based approaches have been at the core of many early seminal SLAM works, alternative SLAM frameworks have gained in popularity to the point that in recent reviews the period of EKF dominance has been called "*the classical age*" in contrast to the *modern age* where we can see a prevalence of optimization-based approaches amongst state of the art frameworks.[24]

2.3 Optimization-based approaches

Optimization-based techniques utilize a graph-like representation of the SLAM problem. This approach breaks down the comprehensive probability distribution inherent to the SLAM issue into a sequence of conditional probabilities. These probabilities encompass only specific subsets of poses, control inputs, and landmark measurements. The inherent graph structure in this formulation facilitates intuitive visual representations, as shown in Figure 2.2.

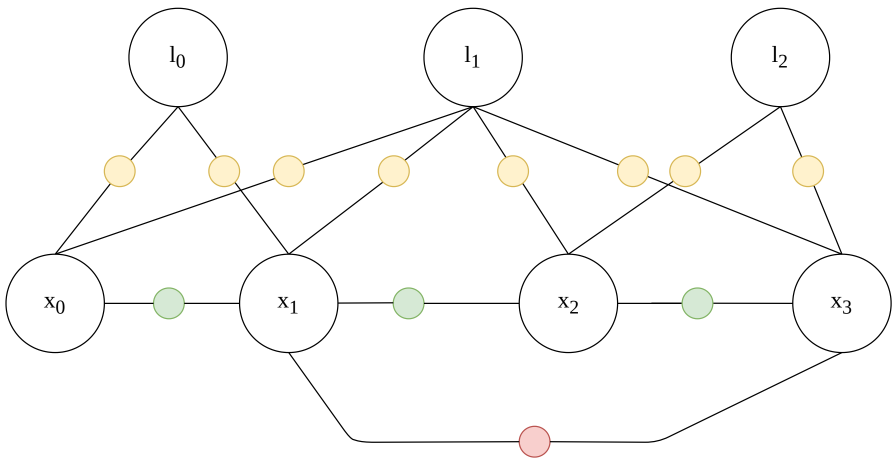


Figure 2.2: Structure of a general graph SLAM

Such graphs are termed *factor graphs* [25]. The nodes symbolize the estimable variables, while the edges, or factors ϕ_i, connect them. Factors in these graphs can represent any function, distinguishing them from *Bayes nets*. It's a common practice to model these factors as non-linear functions, $h(.)$, linked to nodes, and accompanied by Gaussian noise, Σ_i. Within factor graph terminology, these non-linear functions are known as *measurement prediction functions*. Notably, estimating a factor graph's nodes, X, equates to determining the MAP (Maximum A Posteriori) of the joint probability distribution.

$$\begin{aligned} X^{MAP} &= \operatorname*{argmax}_X \Phi(X) \\ &= \operatorname*{argmax}_X \prod_i \phi_i(X_i) \end{aligned}$$

considering that

$$\phi_i(X_i) \propto exp\{-\frac{1}{2}\|h_i(X_i) - z_i\|^2_{\Sigma_i}\}$$

we can rewrite the X^{MAP} expression as:

$$X^{MAP} = \operatorname*{argmax}_X \prod_i exp\{-\frac{1}{2}\|h_i(X_i) - z_i\|^2_{\Sigma_i}\}$$

If we take the negative log of this expression it is clear how this problem becomes the minimization of nonlinear least-squares:

$$X^{MAP} = \underset{X}{\operatorname{argmin}} \sum_i \|h_i(X_i) - z_i\|^2_{\Sigma_i}$$

Optimization within the manifold where our nodes reside is intricate, given that the manifold's dimension often surpasses that of our measurements. Therefore, amassing multiple observations becomes essential for discerning a solution, and even more so for a robust one. While delving into the complexities of such optimization problems goes beyond the scope of this work and most SLAM practitioners, there are recognized strategies to approach them. A prevalent method linearizes the non-linear function around a point X_i, converting the issue into a more tractable linear least square problem. Yet, several algorithms directly minimize the problem's non-linear formulation [26], including: Steepest Descent, Gauss-Newton, Levenberg-Marquardt [27], and the Dogleg [28]. Machine learning —a domain necessitating robust non-linear optimizers— has also birthed algorithms adept at minimizing notably non-convex cost functions, with Adam [29] standing out prominently.

One might observe that the earlier factor-graph-based optimization description implies the need for complete data availability. While continually optimizing with each new data-point's arrival is feasible, it's computationally redundant. Without computational constraints, this would be an ideal problem-solving strategy. Nonetheless, in most scenarios, newly acquired data rarely affects older nodes. Hence, the ability to execute incremental updates to our estimates—capitalizing on previous computations—would be invaluable. Recent innovations, like iSAM2[30], accomplish this by translating factor graphs into a Bayes tree[31]. This allows for seamless inclusion of new leaf nodes, capitalizing on earlier calculations, and retaining precision comparable to comprehensive batch optimization.

Pivotal to understanding factor-graph-based optimization in SLAM is distinguishing between full factor graphs and pose graphs. While both are graph-based representations of the SLAM problem, pose graphs are a condensed version, predominantly capturing the robot's trajectory or poses over time. In pose graphs, nodes represent robot poses, and edges depict spatial constraints between these poses, often arising from motion models or loop closure detections. Contrarily, full factor graphs encapsulate a richer set of information, including individual landmark measurements alongside poses. As such, nodes in full factor graphs denote both robot poses and observed landmarks, with edges linking nodes based on measurements and motion controls. The simplification offered by pose graphs often results in a reduction of computational complexity, making them a favorable choice for large-scale environments or when real-time processing is paramount.

2.4 Filtering versus Optimization

The question of which paradigm is best to tackle the SLAM challenge often emerged amongst roboticists [32, 33, 34]. Some like [33] have mathematically proven that filtering techniques which marginalize out past poses will always be outperformed by optimization techniques which operate on a selection of most relevant past data. It is generally accepted that filtering based techniques are able to run faster and are thus often preferred for embedded systems where real time operation is crucial. On the other hand, optimization based techniques are often preferred when accuracy is the main concern. This should make intuitive sense to those familiar with tasks such as SfM (Structure from Motion) [35].

Choosing which paradigm to subscribe to is not only a matter of raw performance and many *soft-factors* play a role in the choice of paradigm. State of the art optimization-based SLAM libraries[30, 36, 37, 38] often unparalleled modularity and a high degree of abstraction for the end user which is tasked to mainly work on the *measurement prediction function* and many are already provided out of the box.
The next section, providing a sensor-agnostic overview of modern SLAM solutions, highlights how the choice between filtering and optimization is not a binary one and often these can work in tandem to achieve the best possible results.

2.5 Frontend, backend, odometry, and SLAM

The previous overview of the SLAM problem proceeded under the assumption that our system had a preexisting high-level representation of its surrounding. In practice no sensor is able to measure *landmarks* themselves: visual sensors output pixel intensities, LiDAR the distance to the closest obstacle, IMUs accelerations, and so on. Landmarks are a higher level idea and exist only once identified and, most importantly, observed over time.

Ever since the seminal PTAM (Parallel Tracking And Mapping) paper[39] the SLAM community has embraced a general paradigm where the role of landmark detection and positioning updates is left to the *frontend* while that of landmark management and drift correction, also known as loop correction, is left to a *backend.* Finding reliable landmarks for the frontend to recognize over time is arguably the hardest challenge limiting the robustness of SLAM systems and highly platform dependent. On the other hand the backend is usually highly generalizable across many robotics systems and is often left to a general purpose SLAM library.

Since the frontend role is to provide an incremental positional update, they are often referred to in the literature as "*odometries*". The use of the term odometry in this context is likely to have originated in works from the early 2000 where we can first find the term *visual-odometry* [40, 41]. This term was selected in analogy to the term used for other sensors like wheel encoders,*wheel-odometry*, or IMUs, *inertial-odometry*. All these sources of positioning share the limitation of not being able to correct the accumulated drift.

Modern frontends are also able to incrementally build a map of their environment, thus blurring the line between odometry and SLAM. For the remainder of this work a SLAM framework will be defined as any localization and mapping system able to detect and correct its accumulated drift without the use of GNSS data. Conversely, any localization and and mapping system unable to do the same will be referred to with the term odometry.

Chapter 3

Hardware Selection

The first attempts to tackle the SLAM problem in the context of autonomous navigation date back to the early days of the Navlab group [42, 43] from Carnegie Mellon University. The Navlab1 platform, see Figure 3.1, incorporated wheel encoders, IMUs, RGB cameras, sonars, and 3D laser scanners.

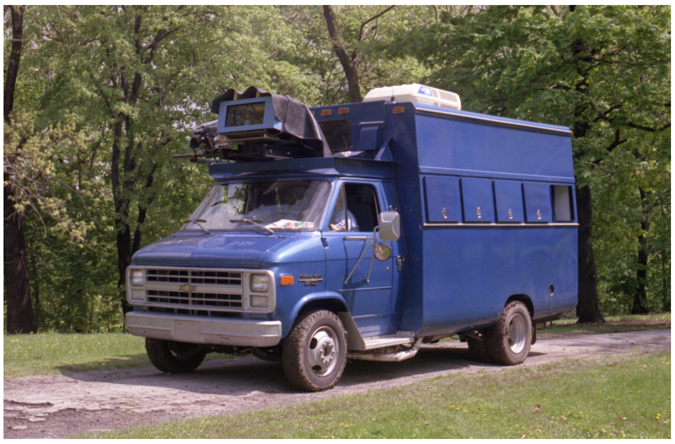


Figure 3.1: Navlab1 development platform from 1986

Sensors can be separated into two families, exteroceptive and interoceptive ones. Interoceptive sensors measure some aspects of the internal state of the platform. A wheel encoder measures the rotational position of wheels and by knowing their geometry, it is possible to estimate the motion of the platform. Wheel odometry is subjected to drift due mainly to slippage between the wheel and the ground. Interoceptive sensors, being blind to their surroundings, are inherently unable to recognize the inevitable drift any localization system is subjected to. This problem can be mitigated by the use of external positioning systems, IMU-based navigation systems, INS (Inertial Navigation System), are for example often equipped with GNSS receivers to periodically correct the accumulated drift.

On the other hand, an exteroceptive sensor captures information regarding the environment surrounding the platform itself. The most popular forms of exteroceptive sensors used for SLAM are cameras[44], LiDARs[45], sonars[46] and radars[47]. Being able to observe the external world is key to any SLAM framework which makes these sensors crucial for SLAM applications.

Table 3.1 shows the general characteristics of many popular sensor used for SLAM. In practice, it is often beneficial to pair exteroceptive and interoceptive sensors together in order to increase the robustness and accuracy of the SLAM pipeline. Often the former output data at a much lower rate compared to the latter, having access to high frequency data is useful to preserve the high frequency motion of the platform.

	Family	*Frequency*	*Resolution*	*Accuracy*	*SWAP*
Camera	Exteroceptive	30-60 Hz	High	Medium	Low
LiDAR	Exteroceptive	10-20 Hz	High	High	High
IMU	Interoceptive	100-1000 Hz	Medium	Varying	Low
Sonar	Exteroceptive	10-100 Hz	Low	Low	Low
Radar	Exteroceptive	10-20 Hz	Low	Low	High

Table 3.1: Sensor characteristics for UAV SLAM

State of the art SLAM solutions for UGVs use all possible sensors to achieve the best possible state estimate. UAVs on the other hand are much more constrained by the maximum payload they can carry while still maintaining airworthiness. For this reason the vast majority of UAVs rely on cameras as their main exteroceptive sensor.

Figure 3.2 compiles the localization performances of the top algorithms tested to date[1] on the KITTI benchmark dataset[48]. From the graph we can see that while camera based algorithms are more popular, LiDAR based ones tend to provide a lower translational error.

[1]Rankings as of 2023

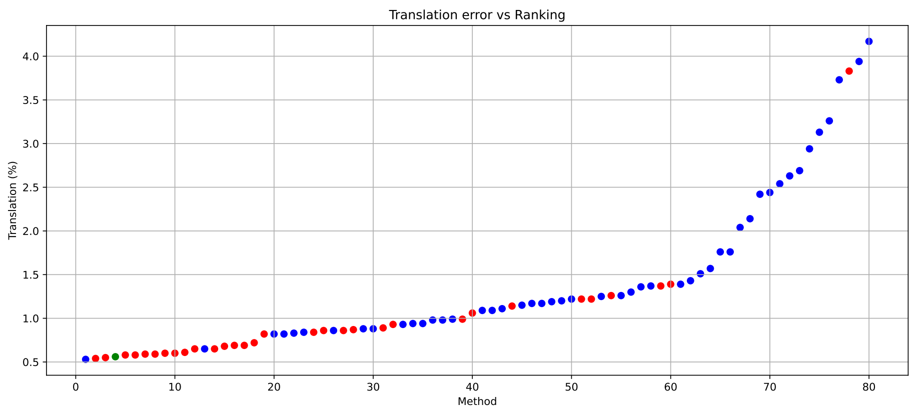


Figure 3.2: Performances of various SLAM algorithms on the KITTI dataset In red LiDAR based ones, in blue camera based, in green Camera and LiDAR based ones.

While in the previous chapter we explained how SLAM work in an agnostic fashion, to explain the difference in performances between LiDAR and visual SLAM we need to explain how their frontends work and what are the inherent limitations to each of these sensor modalities.

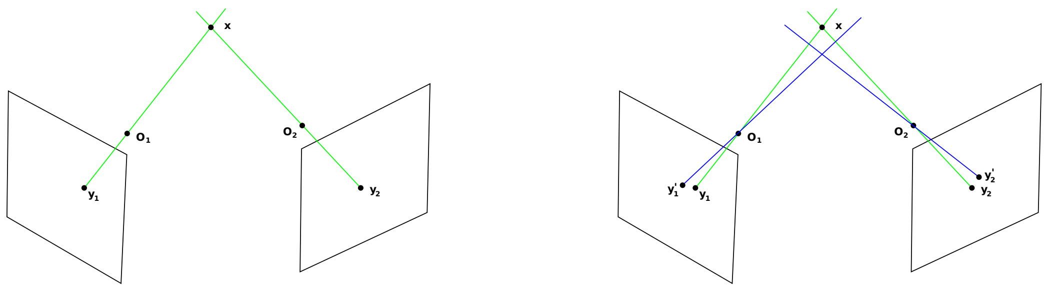
3.1 Visual SLAM

Visual SLAM relies on visual data captured by the camera or set of cameras. These visual data are translated in 3D data using many different strategies but at the core of most of them lies the idea of 3D point triangulation from sets of 2D features. Figure 3.3 provides an intuitive visual understanding of feature triangulation. In an *ideal* scenario, Figure 3.3a, we have a perfect camera calibration matrix P, this matrix establishes a relationship between the 2D image coordinates, $\boldsymbol{y_1}$, and 3D world coordinates,$\boldsymbol{x}$, up to a scale factor k:

$$\boldsymbol{y} = k\boldsymbol{P}\boldsymbol{x}$$

We can imagine this relationship as casting a ray passing through the y_1 and x the scalar factor k represents the fact that we can't gauge depth from a single image. Using another image observing the same feature we can then find the point in 3D space in which these points intersect and use this information to triangulate the position of the point in 3D space.

There are many limitations to this technique, the first is represented in Figure 3.3b, the projection matrix $\boldsymbol{P}$ cannot be estimated perfectly during the camera calibration phase[49]. This leads to imperfect triangulation which gets worse the further the point we are trying to triangulate is from the camera.



(a) Ideal feature triangulation **(b)** Realistic feature triangulation

Figure 3.3: Feature triangulation

Another limitation stems from the problem of recognizing the same 2D point across images in the first place. The usual pipeline first detects points of interest in each images and for each computes a visual descriptor. These descriptors act as identifiers for each point of interest that can be used to then find visually similar points across images. The most well known are probably SIFT[50], ORB[51], and SURF[52]. These early *hand-crafted* descriptors are still very popular in the field of visual-SLAM but can often fail dramatically if the environment is subjected to poor lighting conditions, fog or is lacking textured areas.

Recently, so called, *learned* descriptors have been proposed[53, 54, 55] to address the shortcomings of older methods. These descriptors are derived from machine learning models trained on datasets of image pairs and can address most of the shortcomings of older descriptors[56]. However those descriptors are difficult to compute at the rate at which cameras generate frames making them too expensive for many robotics applications.

Moreover, since point triangulation is an expensive process in practice most SLAM approaches try and triangulate as fewer points as possible while still maintaining accurate localization performances. This results in maps low point density which, while detailed enough for a robot to localize itself, are hard to understand for human operators. Triangulating more points would, in principle, be possible as demonstrated by SfM techniques which can be imagined as an offline version of the visual SLAM problem.

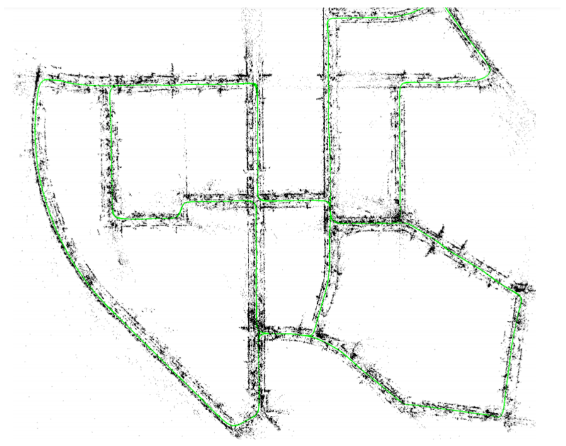


Figure 3.4: Mapping results of State of The Art Visual SLAM[57] on the KITTI[48] dataset

In practice, Visual SLAM can provide accurate localization with moderate computational resources. This, however, comes at the cost of sparse real time mapping performances, see Figure 3.4. Moreover, visual data can be effected by poor visibility conditions making the entire pipeline more fragile and less than ideal to operate in the aftermath of a disaster scenario.

3.2 LiDAR SLAM

LiDAR provide direct 3D measurements of the environment with respect to the sensor itself. These point clouds can be extremely dense, a typical LiDAR outputs hundreds of thousands of points per second and many go up to millions of points per second allowing for accurate mapping without the need for triangulation. This technology is extensively employed for the acquisition of precise 3D spatial information about the surroundings, making it indispensable in various applications, including autonomous navigation, environmental mapping, and object recognition. In this section, we delve into the fundamental principles and operational mechanisms that underlie LiDAR systems.

At its core, LiDAR operates on the principle of emitting laser pulses and measuring the time it takes for these pulses to return after interacting with objects within the sensor's field of view. LiDAR systems employ lasers, typically in the near-infrared spectrum, to generate brief, high-energy pulses of light. These pulses are directed into the environment in a controlled manner, an optomechanical mechanism, allowing them to cover a wide range of angles and distances.

LiDAR SLAM works by aligning subsequent scans using the ICP (Iterative Closest Point) algorithm, first proposed in [58]. The ICP algorithm is a widely used method for registering two 3D point clouds, $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t-1)$, by iteratively refining the transformation that aligns them. The goal of ICP is to find the roto-translation $\boldsymbol{T(t)}$ (a 3D translation and rotation) that minimizes the distance between corresponding points in $\boldsymbol{P}(t)$ and $\boldsymbol{Q}(t-1)$. The general steps in the ICP algorithms are:

- Start with an initial estimate of the transformation $\boldsymbol{T}$, often recovered assuming constant velocity between time t and $t-1$ or assumed to be the identity matrix.
- For each point $p_i \in \boldsymbol{P}(t)$ find the closest $q_i \in \boldsymbol{Q}(t)$.
- Minimize the error $E(\boldsymbol{T})$ between corresponding pairs by optimizing the transformation $\boldsymbol{T}$
$$\underset{\boldsymbol{\Delta}t, \boldsymbol{\Delta}R}{\operatorname{argmin}} E(\boldsymbol{T}) = \Sigma_i ||q_i - \boldsymbol{T} p_i||^2$$
- Update $\boldsymbol{T}$ using the newly found translation vector $\boldsymbol{\Delta}t$ and rotation matrix $\boldsymbol{\Delta}R$
- Iterate until either a maximum number of step is reached or the error falls below a certain threshold.

LiDAR frontend are thus able to recover the ego-motion of the platform between each scan and produce an odometry output which can be fed to a pose graph optimization framework to correct the accumulated drift. The ICP algorithm derives its robustness from two main assumptions, the fact that the initial alignment between incoming scans is already close to being correct and that there will be a great number of points to be observed, thus averaging the noise out over hundreds of thousands of points.

LiDARs are extremely precise sensors and can provide millions of points per seconds to centimeter level precision. However, their accuracy is bound by an inevitable physical phenomenon: *diffraction*, often referred to as *beam divergence* to encompass every aspect which can increase it. The beam divergence angle θ is typically defined as the angle within which the majority of the laser energy is concentrated. It can be mathematically expressed as:

$$\theta = \frac{\lambda}{\pi D}$$

Where λ is the wavelength of the light pulse and D is the diameter of the light beam at its narrowest point.

This ideal value of represents the best possible beam divergence value for any given light pulse, a beam approaching this value is said to be *diffraction limited.* In real commercial systems this lower bound is never reached and the quality of the optics plays a much greater role in the actual performances of the LiDAR. However, while beam divergence plays a pivotal role for systems expected to capture long range data we expect to operate in indoor environments (10-40$\boldsymbol{m}$ range) where the beam divergence characteristics of a LiDAR are not mission critical.

LiDARs, as any sensor, can be affected by many disturbances. Since they work by measuring the time taken by a light pulse to return to the receiver they are affected by factors which prevent this process to work. In practice we found that heavy particulate can impede some of the reflection and provide false returns as can be seen from Figure 3.5. The experiments performed by my colleagues [59] showed that the LiDAR data, while still effected by smoke are still usable for SLAM.

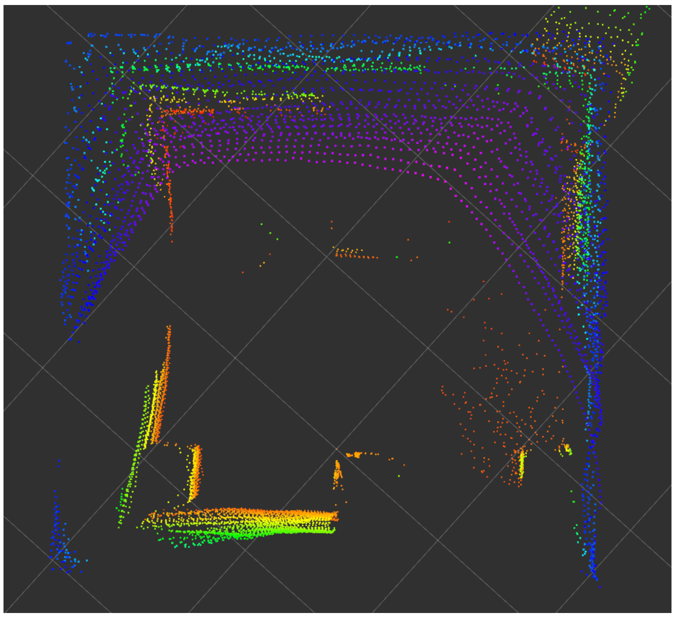


Figure 3.5: Effects of heavy particulate concentrations on LiDAR point cloud

Except under specific edge cases LiDAR sensor offer an incredible level of precision and reliability under conditions spanning from fog to total darkness, such scenarios would prevent RGB-cameras to operate. Moreover, even under ideal conditions LiDARs can outperform cameras in SLAM applications, see Figure 3.2.

In applications where accuracy is paramount, such as surveying, LiDARs have been the de facto standard for decades and have proven themselves to be an essential tool for autonomous applications ever since the 2007 DARPA challenge[60]. The reader might then ask why not always use LiDARs in place of cameras? We argue that the main barriers to LiDAR adoption have been SWAP (Size Weight And Power).

To illustrate this point, the LiDAR sensor used for the KITTI dataset in 2012 is a Velodyne HDL-64e[2], at the time one of the best sensors on the market. The point cloud density and accuracy would still make the HDL-64e useful for LiDAR-SLAM today. However, the size weight and power consumption being 13kg and 60W respectively, make it unfeasible for indoor UAV platforms.

3.3 LiDARs for indoor UAVs

The last ten years have seen the development of ever smaller and more efficient LiDAR sensors[61, 62, 63, 64]. Figure 3.6, shows the different kinds of LiDARs based on their design.

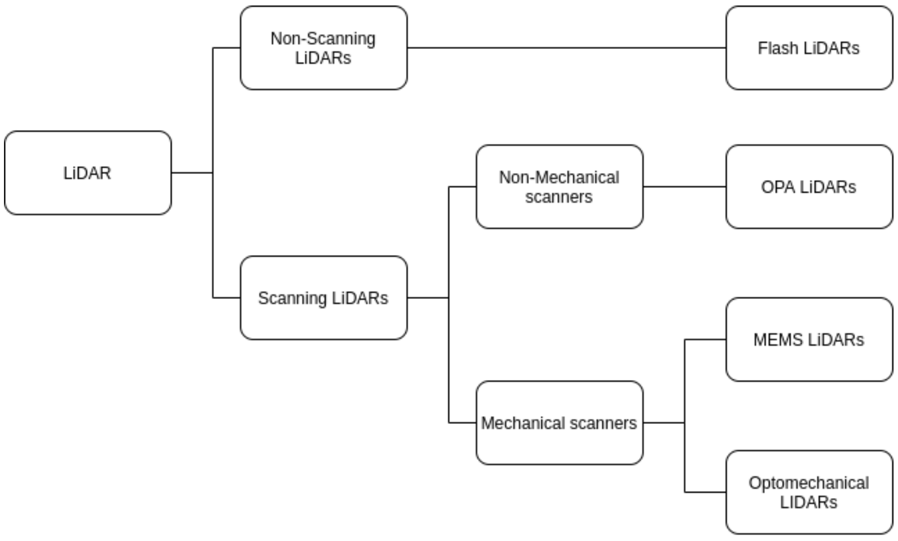


Figure 3.6: Scheme of popular LiDARs technologies available on the market

Optomechanical LiDARs, also known as rotating LiDARs, are designed to capture detailed 3D spatial information about their surroundings. Optomechanical LiDARs

[2]HDL-64e datashet

rely on a rotating mechanism for beam scanning. One of the key features of optomechanical LiDARs is their mechanical scanning mechanism. This mechanism includes a motor that rotates a thin circuit board about its vertical axis. Attached to this circuit board are laser emitters and receivers pairs.

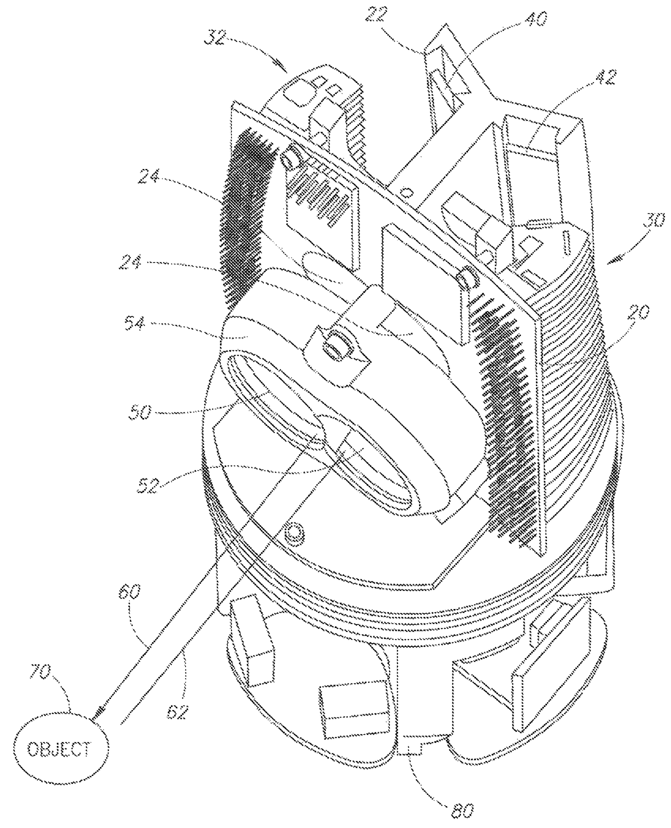


Figure 3.7: Typical design of an Optomechanical LiDAR[65]

Figure 3.7 shows the design of such sensors. Each pair constitutes a beam and a single sensor can have between 15 and 128 such beams. Due to their rotating mechanism, they offer a complete 360-degreeFoV (Field Of View), making them ideal for applications requiring omnidirectional perception. The main limitation of these technologies stems from SWAP considerations: optomechanical LiDARs tend to be larger and heavier compared to solid-state LiDARs, which may limit their suitability for certain UAV applications where SWAP constraints are critical.

On the totally opposite side we have solid state LiDARs, these sensors rely on technologies like OPA (Optical Phased Arrays)[64]. OPA LiDARs, operate by leveraging the principles of optical phase manipulation and beam steering. These

LiDAR systems replace traditional optomechanical scanning mechanisms with an array of tiny optical emitters, often lasers, and controlling phase shifts in the emitted light. By carefully adjusting the phase of the light from each emitter, these LiDARs can precisely steer and combine multiple laser beams, allowing for rapid and precise scanning of the environment without any moving parts. As these laser beams hit objects in the surroundings, they bounce back and are detected by an array of receivers.OPA LiDARs offer the potential for high-resolution 3D mapping, fast scanning speeds. Unfortunately, while promising these sensors are still in the research and development stage and are not yet commercially available.

MEMS LiDARs offer the best compromise between availability on the market and SWAP considerations. These sensors use tiny, highly precise mechanical components constructed at the microscale. In MEMS LiDAR systems, a micro-mirror or an array of micro-mirrors, typically etched onto a silicon substrate, plays a pivotal role. These micro-mirrors can tilt or rotate rapidly, directing laser beams with extraordinary precision. As the laser pulses are emitted, these micro-mirrors adjust their angles to steer the beams towards specific points in the environment and then capture the reflections. MEMS LiDARs excel in terms of their compact size, low power consumption, and high reliability.

In practice, the majority of MEMS LiDAR sensors available on the market suffer from a limitation in their FoV when compared to traditional optomechanical LiDAR sensors, which offer a complete 360° FoV. This reduced FoV poses challenges for the robustness of the ICP algorithm and necessitates specific SLAM algorithms to accommodate them[66]. For your reference, Table 3.2 presents the specifications of some of the most popular MEMS LiDAR sensors currently widely available.

	Ouster Os0-128	**Blickfeld QB1**	**Livox Mid70**
H-FOV (°)	360	70	70
V-FOV (°)	90	30	70
Points (hz)	2.4M	70k	100k
Range (m)	100	70	130
Weight (g)	500	275	580
Power (W)	15	12	8
Price (Eur)	> 10k	not available	1K
Release date	2021	2021	2020

Table 3.2: MEMS based LiDARs specifications

The primary advantage of MEMS LiDARs lies in their lower power consumption, reduced weight, and cost-effectiveness. Notably, at the commencement of this project, Livox introduced a groundbreaking sensor: the Livox Mid360, the details of which can be found in Table 3.3. This innovative sensor represents a significant milestone in the world of LiDARs as it offers a full 360-degree FoV while still maintaining the low SWAP, characteristic typical of MEMS LiDARs.

	Livox Mid360
H-FOV (°)	360
V-FOV (°)	59
Points (hz)	200k
Range (m)	40
Weight (g)	265
Power (W)	8
Price (Eur)	< 1k
Release date	2022

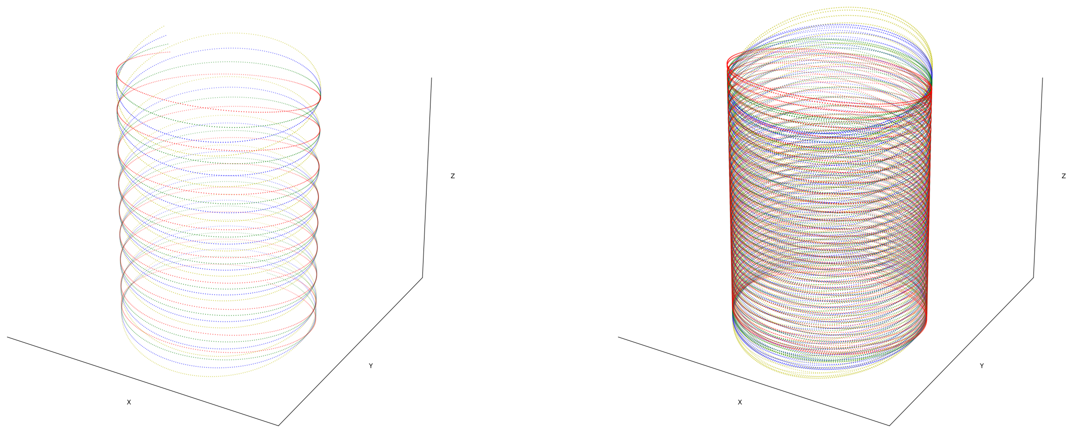
Table 3.3: Livox Mid-360 specifications

The Livox Mid360 sensor achieves the performance metrics detailed in Table 3.3 through a novel approach. It employs a helicoidal non-repeating pattern to guide the light beams. You can observe the resulting point clouds in Figure 3.8, showcasing data generated by the sensor after 0.1 seconds and 2 seconds. To produce these visualizations, we utilized the data provided by the manufacturer, which can be found in the official pattern file[3].

Our analysis of this data, coupled with subsequent experiments, led us to a significant insight. The Livox Mid360 sensor consists of four diodes strategically oriented towards a MEMS mirror, responsible for deflecting their beams. This unique configuration results in a pattern that efficiently saturates the surrounding space over time.

After careful evaluation of available LiDARs options, we determined that the Livox Mid360 was the most suitable choice for our project. The primary factor influencing our decision was its exceptional low SWAP characteristics, which made it an ideal candidate for mounting on a small indoor UAV. This feature was crucial

[3]official pattern



(a) Mid360 pattern after 0.1 seconds **(b)** Mid360 pattern after 2 seconds

Figure 3.8: Example of the non repeating pattern of the Livox Mid360. Points from different diodes are coloured in red, green, blue and yellow respectively.

for our specific application, where the UAV's agility and maneuverability were of paramount importance.

Moreover, the Livox Mid-360 exhibited performance metrics that we deemed sufficient for our SLAM requirements. However, it is important to note that the Livox Mid360 does have a limitation when compared to top-of-the-line optomechanical LiDARs. It offers a lower point cloud density, which can impact the precision of our mapping and navigation tasks. Moreover it is unclear to what extent the unique acquisition pattern may effect the LiDAR SLAM developed with traditional sensors in mind.

To comprehensively assess the implications of this limitation, we conducted a rigorous benchmarking process. This involved comparing the performance of the Livox Mid-360 against, arguably, the best LiDAR sensor currently available on the market: the Ouster Os0-128. While its SWAP characteristics make it a poor candidate for integration in an indoor-UAV its pointcloud make it an ideal to act as a baseline for our comparisons. The results of this benchmarking exercise will be discussed in detail in the following chapter, shedding light on how the Livox Mid-360's point cloud density affects its performance in various SLAM scenarios

Chapter 4

Analysis of state of the art SLAM algorithms

In the previous chapter, we discussed the challenges arising from the SWAP characteristics of traditional optomechanical LiDARs. These challenges have impeded their incorporation into indoor UAVs, leading to a research gap in SLAM algorithms specifically designed for indoor platforms with LiDAR sensors. While LiDARs are commonly used on outdoor UAVs for tasks such as surveying[67, 45] and on ground vehicles including UGVs and automobiles, the majority of research has been directed towards SLAM algorithms suited for these contexts. We addressed this disparity in our publication [68], detailing our testing campaign in the following section."

In our pursuit of advancing indoor UAV-based mapping and navigation, we established a set of criteria for selecting suitable SLAM algorithms for our experiments. These criteria encompassed the following key considerations:

- **Dual Sensor Support**: Central to our objectives was the evaluation of the Livox Mid360's performance in comparison to traditional LiDAR sensors. Consequently we only tested SLAM pipelines which supported both the Livox Mid360 and the Ouster Os0-128. This dual-sensor support allowed us to conduct a comprehensive and fair comparisons between the Livox Mid360 and traditional LiDAR sensors, taking into account the distinct characteristics of each sensor. Notably, we excluded popular frameworks such as F-LOAM[69] and LeGo-LOAM[70] from our testing campaign due to their incompatibility with the point clouds generated by the Mid360. Conversely Livox-LOAM[66] was excluded as it was incapable of process the pointcloud generated by the Os0-128.

- **Real-time performances**: Given the dynamic and real-time nature of indoor UAV mapping scenarios, the ability to operate in real-time was an imperative criterion. This ensured that our chosen SLAM algorithms could deliver timely and responsive localization and mapping results to first responders and enable effective UAV navigation and mission execution. For this reason we excluded PUMA[71] because, although it could process both set of data it was unable to do so in real time.

- **ROS compatibility**: Recognizing the ubiquity and versatility of the Robot Operating System (ROS)[72] framework in in robotics research and development, compatibility with ROS was deemed essential. This compatibility streamlined the integration of our selected SLAM algorithms with the broader robotic system, ensuring seamless communication and data exchange. We excluded for this reason MULLS[73] since while able to process data from both data it was not integrated with ROS.

- **Open-Source Availability**: An implicit but critical requirement was the availability of SLAM algorithms as open-source software. This stipulation was essential to enable thorough testing and evaluation, as access to the algorithm's source code is indispensable for a comprehensive understanding of its functionality and performance. Consequently, the many commercial LiDAR SLAM solutions that lacked open-source availability were excluded from consideration.

The subsequent sections of this chapter will delve into the specific SLAM algorithms selected based on these criteria and their comprehensive performance evaluations under these rigorous conditions. Through this research, we aim to shed light on the capabilities of the Livox Mid360 and its potential to advance indoor UAV-based mapping and navigation.

4.1 Presentation of the selected Algorithms

4.1.1 CT-ICP

The core of the CT-ICP [74] algorithm is its novel approach to the scan-to-map registration step. One of the limitations associated with employing the naive ICP[58] algorithm for matching successive scans stems from the distortion experienced by each new scan as the sensor undergoes movement during the acquisition process[75]. This work is not the first to implement a de-skewing strategy prior to the scan matching step. However, the approach taken here is novel in that it does not rely on a constant velocity model for the sensor motion or IMU data. This system proposes an elastic formulation of the trajectory with continuity of start and end pose in the scan acquisition step while allowing for discontinuity between adjacent scans. In particular, during the de-skewing step no assumption is made about the fact that the pose of the sensor at the beginning of scan $T_b(n)$ is equal to that of at the end of the previous one $T_e(n-1)$. This approach preserves high-frequency motion while still correcting for the scan skew. In the same paper, the authors also introduce a back end pose graph optimizer based on g2o [37] used to implement Loop Closure. However, it has not been integrated with the ROS framework and can only be used to process datasets off-line and we consequently did not integrated it in our tests.

4.1.2 KISS-ICP

The KISS-ICP framework [76] aims to be a general approach applicable to any LiDAR sensor with minimal to no tuning of its parameters. It does so by having no assumption on the kind of sensor being used and avoiding point cloud descriptors to aid in the scan matching step. To this end KISS-ICP forgoes most of the sophisticated optimization techniques and reduces the odometry estimation loop to four steps, namely:

- De-skewing the incoming scan using the constant velocity model,
- Sub-sampling of the deskewed point cloud to bound complexity,
- Scan-to-map matching to recover the incremental odometry,
- Updating of the local map stored using a voxel grid using the previously subsampled point cloud.

During the scan-to-map matching step, the regular ICP algorithm is utilized. However, instead of establishing a maximum limit for iterations, an adaptive threshold is employed, using the point-to-point distance between the local map and the incoming scans as metric.

4.1.3 LIO-SAM

LIO-SAM [77] is an integrated inertial-LiDAR odometry and mapping pipeline.The utilization of IMU data serves a dual purpose within this framework. First, it plays a crucial role in the de-skewing step, aligning LiDAR data accurately. Second, the IMU provides an initial estimation for the scan registration step.

The integrated IMU data and the LiDAR odometry data, together with optionally loop closure and GNSS signals, are merged into a factor graph which is optimized using GTSAM [31]. The scan registration step generates the LiDAR odometry factor, i.e., each scan is registered to a local map by first extracting edge and planar features based on the local roughness [78].

The factor graph formulation allows for the integration of loop closure in the LIO-SAM framework. Loop detection is initialized based on the Euclidean distance between the current pose and previous key poses in the previous trajectory. Upon triggering, the current scan is aligned with the one corresponding to the pose captured by the key pose which activated the loop closure event, the retrieved rototranslation is used as a constraint for the pose graph. Like depicted in Figure 2.2.

4.1.4 FAST-LIO2

Fast-LIO2 [79] is a SLAM system that relies on both LiDAR and IMU data. In its previous iteration [80], the system utilized an iterated EKF for point cloud deskewing and scan matching initialization. However, one notable evolution in the latest version is the elimination of the need for a feature extraction step in the scan matching process.

The standout feature of this updated formulation lies in its ability to handle dense maps effectively, which consequently enables efficient scan-to-map matching. Unlike many other LiDAR SLAM systems that either rely on sub-sampling or require feature extraction to make real-time scan matching feasible, FAST-LIO2 introduces a specialized implementation of a k-d tree for storing and managing map data. K-d trees are data structures designed for fast point matching using kNN (k-Nearest Neighbor) search.

One challenge with traditional k-d trees is that inserting new points into the map requires re-balancing the entire tree, potentially causing non-real-time operation, especially when incremental map generation is necessary. To overcome this issue, the authors implemented a self-balancing k-d tree based on the *Scapegoat Tree* concept proposed in [81]. This innovative approach ensures that the final algorithm can handle extremely dense maps while maintaining real-time performance.

4.2 Benchmarking methodology and analysis

Having selected a representative sample of LiDAR SLAM algorithms we moved onto the testing phase. Many datasets have been proposed in the literature and are usually used as a benchmark for SLAMs. However, most of these datasets have been collected from sensors mounted on cars[48, 82, 83, 84] or other UGVs [85, 86, 87]. The comparatively slow and smooth trajectories of ground platforms prevents them from capturing data challenging enough to tease the differences between different

The trajectories of ground vehicles typically follow slow and smooth paths, which limit the complexity of the data they capture. Using LiDAR data collected from ground vehicles may not provide a reliable basis for predicting which algorithm will excel when the sensor is mounted on a drone. Some recent datasets[88, 89] have incorporated hand held sequences using MEMS LiDARs. These datasets could potentially offer intriguing insights, as hand-held sequences often exhibit greater motion variation compared to those captured from a vehicle. However, these datasets do not include data from the specific LiDAR models we aimed to evaluate.

Recognizing the need for relevant and tailored data, we made the decision to develop a custom dataset within a simulated environment. This approach allowed us to precisely replicate the characteristics of both the Mid360 and Os0-128 sensors, ensuring that the data generated closely resembled the real-world scenarios our research aimed to address.

Furthermore, the simulation environment presented us with an invaluable opportunity: the ability to equip the drone with both virtual sensors simultaneously. This capability allowed us to capture both point clouds during each flight. This is impossible to achieve in the real world, given the considerable weight of the Os0-128 sensor for an indoor drone. During this phase, our primary objective was to assess whether the Mid360 could reliably generate actionable point clouds, particularly when subjected to the complex motion patterns typical of indoor UAVs. This investigation was fundamental in the context of our exploration of SLAM techniques for indoor UAVs.

By creating this custom dataset, we aimed to bridge the gap between existing datasets and the unique demands of indoor UAV-based mapping and navigation, enabling a more accurate and comprehensive evaluation of SLAM algorithms and the performances of the Mid360.

4.2.1 Simulation Pipeline

We choose to integrate our simulation withing the Gazebo simulation environment[90]. Compared to other popular simulation environments such as AirSim[91], Carla[92], or Nvidia Omniverse[1], Gazebo offers increased efficiency. This comes at the cost of decreased visual fidelity, however, since it still supports high quality

[1]Nvidia Omniverse presentation page

meshes this will not effect the simulation of *lidar*s. The other simulation pipelines are be better suited for advanced visual SLAM simulations.

In order to generate the point clouds for the Mid360 sensor we elected to use the official pattern released by the manufacturer. However the official plugin it is a part of introduces distortion in the point cloud that unacceptably degraded the SLAM performances. An example of such distortion can be seen in Figure 4.1. Both point clouds were generated by placing the virtual sensor in the center of a cylinder, the left point cloud shows a noticeable distortion. Analysis of the original code showed that this was caused by improper handling of the zenithal coordinate when computing the distance to the closest mesh.

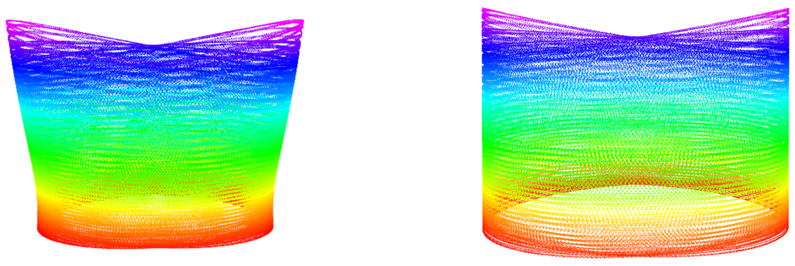


Figure 4.1: Comparison of point cloud generated by the original plugin, on the left, and ours, on the right.

We further improved on the original plugin by adding the timestamp of each point in the point cloud a behaviour present in the data outputted by the original sensor but not present in the data generated by the official plugin. Lastly, our implementation is able to work with the latest version of ROS and Gazebo, this is not the case for the official plugin. We released our version of the plugin[2] to the benefit of the robotic community as part of our previous publication[68].

To the best of our knowledge only one other work tackling the simulation of the Mid360 is the recently released MARSIM[93]. MARSIM implementation is a standalone LiDAR simulation framework which forgoes entirely the use of meshes and only relies on dense point clouds to represent the environment. Their strategy yields them higher performances compared even to the already fast Gazebo but requires highly detailed, and manually refined, point cloud maps. We decided to

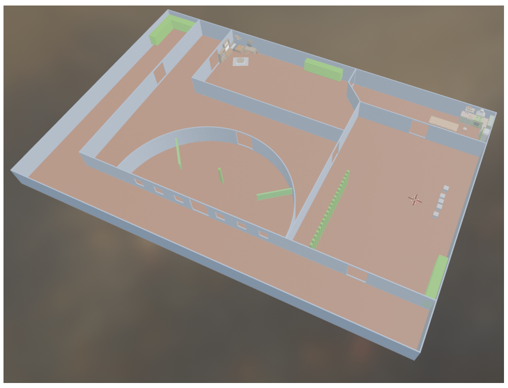
[2]Link to the plugin

work in as a plugin to gazebo for its deep ROS integration and possibility to use meshes as maps.

Testing Environments

Being able integrate maps in Gazebo we decided to create two maps as testing environments. Figure 4.2 shows the two maps we used to run our tests in.

The initial map, referred to as the *house*, exhibits a relatively compact layout in comparison to the second map, the *cave*. Notably, the house map features a greater number of open spaces, strategically chosen to facilitate the execution of more aggressive flight maneuvers by our simulated drone. In contrast, the cave map is characterized substantially larger, maze-like configuration, rendering aggressive drone piloting unfeasible. However, the extensive network of elongated corridors within the cave map posed a significant challenge to the overall robustness and consistency of our SLAM algorithms.


(a) House map

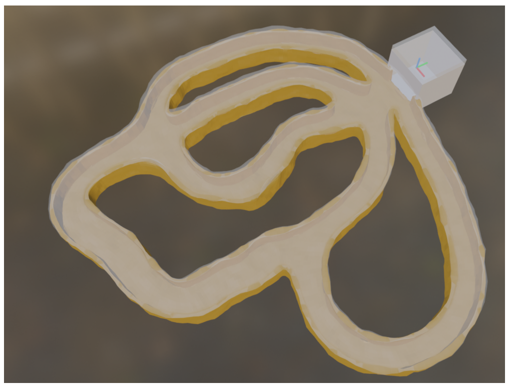

(b) Cave map

Figure 4.2: Maps used to run our tests in.

4.2.2 Evaluation procedure

Most benchmarking efforts evaluate only the localization performances of different SLAM algorithms. This is partially due to the fact that capturing ground truth data of the environment is a hard and time consuming endeavour and partially due to the fact that most of the slam research has focused on improving the localization accuracy rather than the mapping quality. The metrics most often used to benchmark the performances of a SLAM algorithm are the ATE (Absolute Trajectory Error) and RPE (Relative Pose Error)[94], to measure the global consistency and the local accuracy respectively.

To compute the RPE, the reference trajectory is divided into uniformly spaced intervals, of 1 m in our case. The relative transformation $\Delta_{i,j}$ between the pose at the beginning P_i and at the end P_j of each segment is then computed for both trajectories. Finally, using the inverse compositional operator [95], denoted as $\ominus$, we can compute the RPE between each transformation in the reference and relative trajectory. Using the RPE is possible to quantify the drift per meter in each run and in doing so have a metric to represent the local accuracy of each SLAM pipeline.

$$RPE_{i,j} = \Delta_{est_{i,j}} \ominus \Delta_{ref_{i,j}} = (P_{ref,i}^{-1} P_{ref,j})^{-1} (P_{est,i}^{-1} P_{est,j})$$

Conversely, the ATE measures the overall consistency of the final trajectory by aligning the estimated one and the ground truth one, optionally the scale error is also estimated by computing the overall Umeyama[96] transformation between the two sets of poses. Once aligned the distance between each estimated pose and its closest neighbour amongst the ground truth ones is computed.

In our tests we decided to use the RPE measurement using[97] to compute the local drift of each sensor in each run but we used the final point cloud to measure the overall consistency of our SLAM. In analogy to the ATE we computed the global consistency by:

- Aligning the point cloud with the ground truth mesh.
- Computing the distance between each point in the point cloud and the closest vertex in the mesh
- Computing the RMSE (Root Mean Squared Error) over all the point-to-mesh distances.

In our analysis of the data we identified as outliers all the points whose distance from the mesh was greater than 20 cm. These points severely effect the quality of the final map and present themselves as *ghosting* artifacts[98] in the point cloud. An example of these artifacts can be seen in Figure 4.3.

Local Accuracy Results

Table 4.1 presents the local accuracy of the selected algorithms on two different maps: *House* and *Cave*. It is notable that most algorithms found the *House* map more challenging than the *Cave*. One possible explanation is the constrained environment of the *Cave*, which restricts the range of potential motions available to the UAV during flight. In contrast, the *House* offers open spaces, allowing

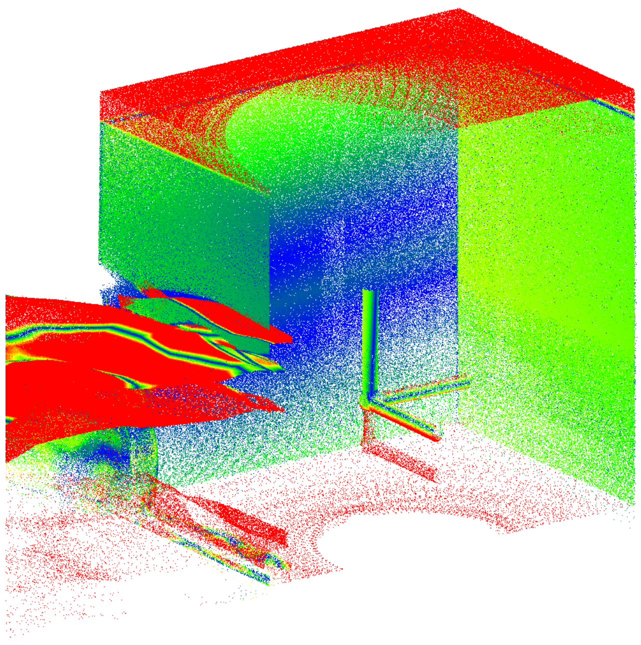


Figure 4.3: Section of the final point cloud captured in the cave. The points in red are those whose distance to the original mesh was greater than 20cm

for abrupt changes in both speed and heading. This poses challenges for LiDAR-based odometry, as this sensor family typically has a low scan acquisition rate. While conventional cameras capture images at frequencies ranging from 30-60Hz, LiDAR sensors commonly operate at a scan acquisition rate of 10Hz, making them especially sensitive to high-speed motion.

High-speed motion coupled with a low scan acquisition rate results in poorly overlapping scans, regardless of accurate de-skewing. This adversely affects the scan-matching process, a critical component in any SLAM algorithm, thereby compromising the overall state estimation performance. The impact of a low data acquisition rate on odometry estimation has been extensively examined within the visual SLAM community [99].

CT-ICP demonstrates remarkable performance, particularly given its non-reliance on IMU data. The high local accuracy can be attributed to its scan registration method, which effectively preserves the sensor's high-frequency motion components. This is particularly evident in its performance on the *House* map, where it achieves

Cave					
	CT ICP	KISS ICP	LIO SAM lc	LIO SAM	Fast LIO2
Mid-360	1.78	**3.34**	2.50	2.61	1.08
Os0-128	**1.53**	8.17	**1.26**	**1.24**	**0.49**

House					
	CT ICP	KISS ICP	LIO SAM lc	LIO SAM	Fast LIO2
Mid-360	**2.72**	**6.88**	3.54	2.7	1.61
Os0-128	2.79	7.1	**1.65**	**1.51**	**1.11**

Table 4.1: RMSE of the drift in cm per 1 m interval.
In **bold** the best-performing sensor for each algorithm, underlined the best-performing algorithm for each sensor

accuracy levels comparable to LIO-SAM.

Examining KISS-ICP we can see the only instance where the local accuracy decreased going from the *Cave* to the *House* map. Nonetheless, it performed worse than other algorithms. It can be observed that also the other purely LiDAR-based algorithm outperforms KISS-ICP, as the constant velocity model, employed to provide the initial guess for scan-to-map registration, frequently experiences significant violations during our tests where the sensor was mounted on a UAV subjected to high speed motion.

LIO-SAM, both with and without loop closure, has been able to preserve high local accuracy in both maps. We attribute this to its use of the IMU pre-integration used to provide a first guess to the scan alignment step. The local accuracy doesn't show a significant improvement by activating the loop closure as loop closure can only correct the accumulated drift and not the instantaneous one, we will discuss its impact on global consistency in the next section.

	LIO-SAM	Fast-LIO2
Mid-360	+3.3%	+32.5 %
Os0-128	+17.9%	+55.8%

Table 4.2: Percentage of drift increase from *Cave* to *House* map

The performances of Fast-LIO2 are a testament to the impact that dense matching can have on LiDAR SLAM. Fast-LIO2 shows the lowest drift of all the algorithms in either map and using either sensor. While both Fast-LIO2 and LIO-SAM rely on IMU data to remain accurate at high-speed, the use of dense matching improves the accuracy of each registration yielding more accurate odometry. However, computing the drift increase going from the *Cave* to the *House* map, see Table 4.2, suggests the pose graph strategy implemented by LIO-SAM[31] may be a comparatively more robust method than the iterative EKF implemented by Fast-LIO 2, albeit less precise in absolute terms.

Looking at the impact of the sensor itself, it is evident that the CT-ICP algorithm's ability to perform odometry estimation in an indoor environment is largely unaffected by which sensor is being used. By storing the local map in a sparsified voxel structure CT-ICP, is able to reduce its computational footprint while at the same time having the side effect of uniformly processing point clouds with different densities. Conversely, LIO-SAM shows a clear preference for higher-density point clouds, this is due to the fact that in order to extract the LAOM[78] features it needs to perform scan matching, and higher-density point clouds are greatly beneficial in detecting the features in the first place. Overall the Mid360 measurably decreases the local accuracy of most SLAM algorithms but, while measurable, it's not significant. Its average being below three centimeter per meter in the most challenging map.

Global consistency

Looking at the Table 4.3 we can see that the *Cave* map has been the most difficult to maintain global consistency in. All algorithms succeeded in generating an accurate final map of the *House* with both sensors. This is likely due to the fact that the many openings present in the map, combined with its smaller size, effectively turned any local map into an equivalent *global* one, this can be seen in Figure 4.4. Registering points to what amounts to a *global* map rather than to a *local* one improves the overall consistency of the final map.

Looking at the point-to-mesh histograms in Figure 4.5 we can see that, while CT-ICP did not show a strong preference for either of the two sensors when looking at the local accuracy, the global consistency decreases significantly using the data provided by the Mid360. In particular, Table 4.4 shows a tenfold increase in the number of outliers. As the authors themselves have highlighted, local maps maintained on the distance d to the last registered scan, and not on a sliding window of the n most recent frames, are particularly susceptible to degradation of the global consistency following bad scan insertion. CT-ICP counteracts this

Cave					
	CT ICP	KISS ICP	LIO SAM lc	LIO SAM	Fast LIO2
Mid-360	20.06	**22.58**	9.99	**14.27**	19.98
Os0-128	**7.68**	51.47	**6.26**	17.19	**5.52**

House					
	CT ICP	KISS ICP	LIO SAM lc	LIO SAM	FAST LIO2
Mid-360	6.45	6.86	6.95	5.90	**5.27**
Os0-128	**4.33**	**4.88**	**3.65**	**4.19**	5.52

Table 4.3: RMSE in cm compued from the point-to-mesh distance.
In **bold** the best-performing sensor for each algorithm, underlined the best-performing algorithm for each sensor

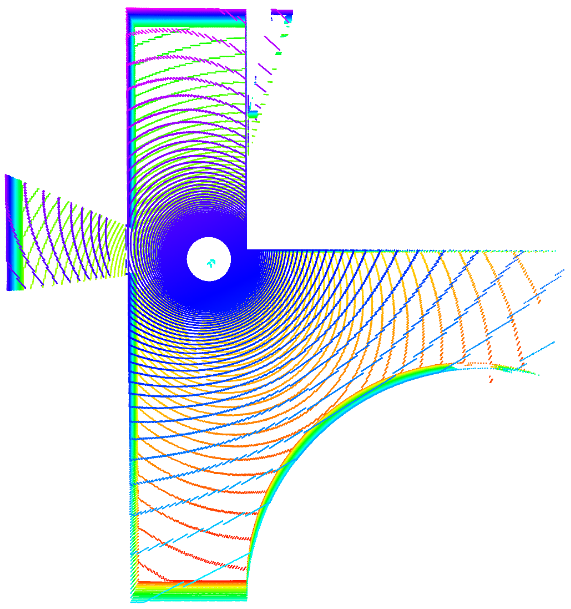


Figure 4.4: Scan from the Os0-128 captured in the *House* map

by adopting a more cautious approach to scan-to-map registration during abrupt pose changes. Notably, sensors with longer-range capabilities are more sensitive to minor yet rapid motion changes, aiding early detection. The limited range of Mid360's point cloud, however, makes it less conducive to triggering this preventive mechanism, resulting in ghosting artifacts.

Cave					
	CT ICP	KISS ICP	LIO SAM lc	LIO SAM	Fast LIO2
Mid-360	14.95	**31.91**	<u>3.68</u>	**10.68**	15.59
Os0-128	**1.40**	58.25	**<u>0.17</u>**	21.78	**0.18**

House					
	CT ICP	KISS ICP	LIO SAM lc	LIO SAM	Fast LIO2
Mid-360	0.13	3.17	4.24	2.28	**<u>5.92e-3</u>**
Os0-128	**0.02**	**0.45**	**9.54e-4**	**<u>9.34e-4</u>**	0.02

Table 4.4: Percentages of outliers in each final point cloud.
In **bold** the best-performing sensor for each algorithm, <u>underlined</u> the best-performing algorithm for each sensor

Figure 4.5 underscores the efficiency of the optimizations present in CT-ICP. While KISS-ICP also employs a voxel structure for local map maintenance, it apparently lacks strategies to counteract poor scan registration. Consequently, point clouds from the Os0-128 produce amplified ghosting artifacts in the resulting map.

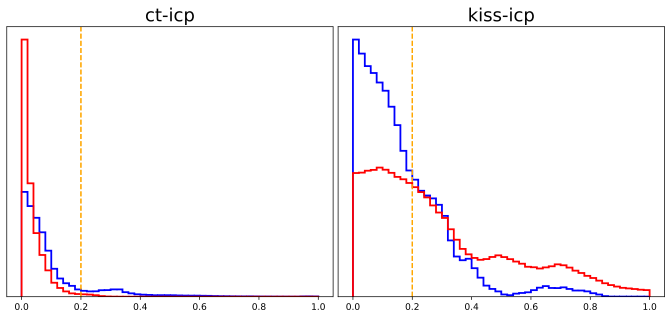


Figure 4.5: Point-to-mesh error distribution for CT-ICP and Kiss-ICP
In blue with the Mid360 and in red with os0-128

The maps generated by LIO-SAM demonstrate the effects that loop closure has on maintain global consistency. From Figure 4.6 it is possible to see a pronounced bulge in the point-to-mesh distance histogram of the final map generated by LIO-SAM without the loop closure module active. These are due to entirely misaligned portions of the final map. Activating the loop closure realigns the trajectory and the map indirectly. These misalignments occur infrequently but are difficult to predict, loop closure is the best tool to counteract this phenomenon. However, finding the right set of parameters for loop detection in LIO-SAM is a trade-off between real-time operation and the global consistency of the map. As the authors themselves wrote, the loop detection module, being based on the Euclidean distance between the current pose and the trajectory, is *naive but effective* and more advanced methods are present in the literature [100, 101].

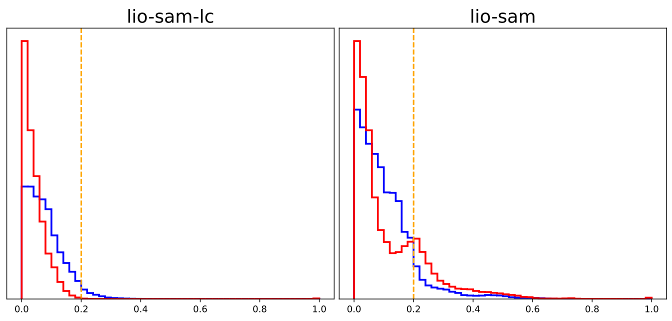


Figure 4.6: Point-to-mesh error distribution for LIO-SAM with and without loop closure active
In blue with the Mid360 and in red with os0-128

Fast-LIO 2 produces admirable results in both the *House* and the *Cave* environments. The inconsistent proportion of outliers across each of the four tests, see Table 4.4 is indicative of how the dense scan-to-map registration approach is susceptible to drift accumulation which is never able to correct. Figure 4.3 is indicative of the results that can be expected in a worst-case scenario.

However, preserving global consistency using a dense global map comes with the caveat of considerable memory consumption. The i-k-d tree implemented in Fast-LIO 2 allows for real-time interaction with a dense global map, as far as CPU utilization is concerned. However, it does not reduce its size. Figure 4.7 shows the memory utilization of the algorithm over time and it is evident that the memory utilization grows linearly with time by a factor proportional to the size of the point cloud being used. In our tests, after a few hundred seconds the map has grown to occupy a few gigabytes. The algorithm starts to truncate the global map only when the sensor is detected as being outside a predetermined distance from the first scan which by default is set to 1 km. This approach is inherently memory unsafe, not because of the size of the default distance, but because it is possible for a sensor to remain static and saturate the system RAM.

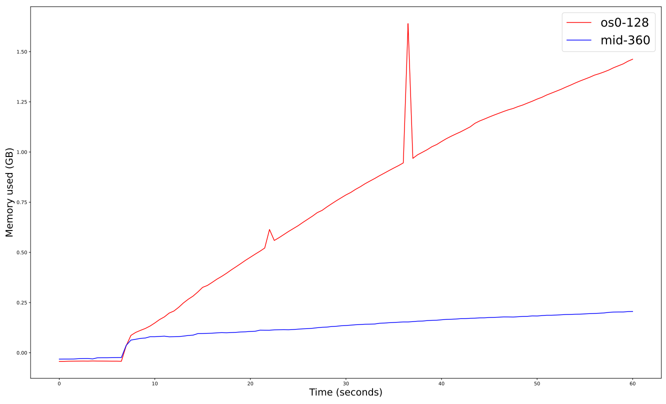


Figure 4.7: Memory usage of Fast-LIO2.
In blue with the Mid360 and in red with os0-128

4.3 Selection of the SLAM algorithm

Upon analysis of the data from each algorithm, we opted for CT-ICP [74] for our subsequent tests on real hardware. This decision was informed by several factors:

- FAST-LIO2 [79], despite showcasing the best results, exhibited high memory consumption, rendering it suboptimal for a SLAM framework operating on power-constrained embedded hardware.
- Kiss-ICP [76], although the simplest to calibrate and execute, performed the least efficiently among the SLAM systems examined. Notably, while its constant velocity model suffices for data from ground platforms, it falls short with data from agile UAV.
- LIO-SAM [77] didn't present a substantial benefit over CT-ICP. Moreover, its backend relies on GTSAM [36]. Our team's path planner [102] incorporates its backend based on Voxgraph [38] for sub-map maintenance and exploration decision-making. Hence, to avoid concurrent operation of two backends, we decided to select CT-ICP and integrate it into the Voxgraph-provided backend.

4.3.1 Performance on real data

The testing platform is depicted in Figure 4.8. Given the minimal size and power requirements of our sensor, it was integrated into a compact UAV suitable for indoor laboratory piloting. The drone's dimensions are 40X40X30cm (Width, Length, and Height), with a flight autonomy approximated at 10 minutes.

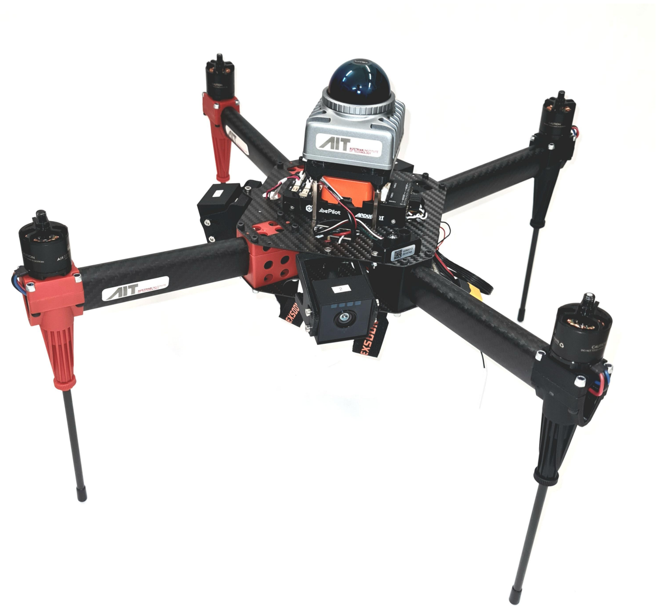


Figure 4.8: UAV mounting the Livox Mid360

The final map, constructed using LiDAR data captured by the Mid360 affixed to the drone, is displayed in Figures 1.5 and 4.9. This map results from a 3:54s mission during which the drone traversed an estimated 120 meters, as determined by the odometry data from CT-ICP. Notably, we lacked access to a tracking system for ground truth position validation during testing. Thus, neither local nor global error quantification is feasible. Yet, a qualitative analysis suggests that the map's reconstruction is precise.

Figure 4.10 shows the plot of the estimated vertical position estimated by CT-ICP. As we took off and landed from the ground we can quantify the overall accumulated error in the vertical direction at the end of the run. Using the last 20 position estimates we have a final height estimate of -4.5cm. Assuming the floor to be level and flat, an overall error of less than 5 centimeters makes us confident that even assuming degraded performances when deployed in a disaster scenario the overall consistency should be satisfactory.

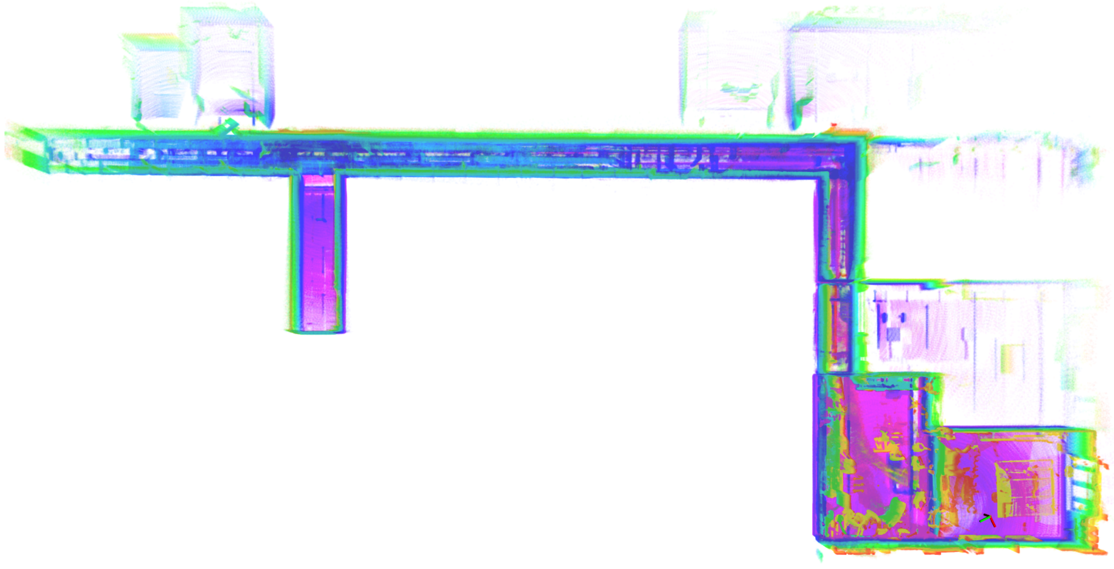


Figure 4.9: Top down rendering of the point cloud generated by CT-ICP from data collected from our flying platform

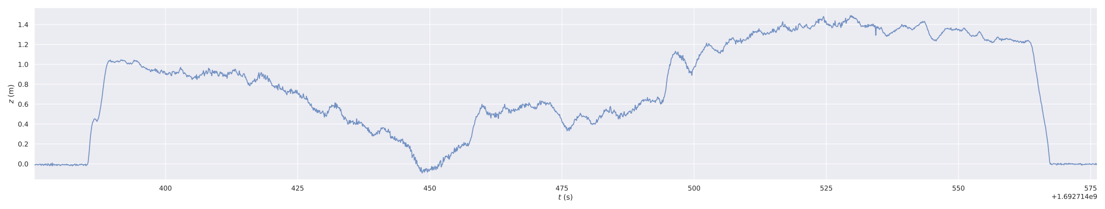


Figure 4.10: Plot of the height during real flight as elaborated by CT-ICP

Chapter 5

Loop Closure implementation

As described in section 2 a true SLAM framework must incorporate both a way to detect the accumulated drift and correct it. Amongst the slam framework we presented in the previous chapter most[74, 79, 76] should be better characterized as *odometries* rather tham real *slam*. The only exception is LIO-SAM which keeps track of backtracking to take advantage of possible loop closures which it maintains using GTSAM[36].

In our analysis of the algorithms we selected CT-ICP[74], which does not naively support loop closure. To mitigate this limitation in this chapter we will present a possible strategy to integrate true loop detection and correction in the context of LiDAR SLAM, Figure 5.1 shows the general architecture.

5.1 Loop detection

Many strategies for detecting loop closures with LiDAR data have emerged in the literature; however, numerous approaches grapple with challenges. Some struggle to operate in real-time [103], while others cannot recover the full 6 degrees of freedom correction [100, 74]. To overcome these limitations, our research integrated a recently proposed methodology: BoW3D [101].

BoW3D is an innovative method that leverages the LinK3D[104] feature descriptor to offer fast and accurate loop closure detection within 3D point cloud data. The underlying principle behind this approach is inspired by traditional 2D image features, such as SIFT[105] and ORB[51], and the method seeks to represent 3D keypoints using surrounding neighborhood information.

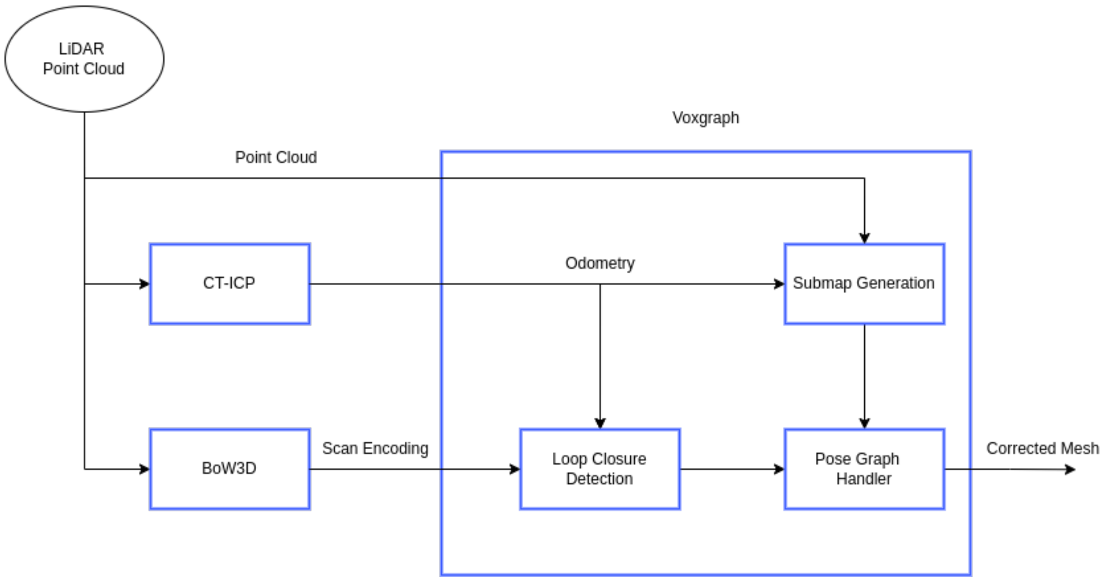


Figure 5.1: Architecture of how we integrated loop closure detection and integration in a pose graph framework

BoW3D predominantly consists of a vocabulary and associated place sets. The vocabulary is a compilation of words. Each word represents a unique combination of dimension values from the LinK3D descriptor. These words encapsulate the different recognized patterns within the 3D LiDAR data. In essence, they serve as a summarized representation of the key features found across different frames of point cloud data.

Accompanying each word in the vocabulary is a respective place set. A place set enumerates the frames (or scans) in which a particular word (or pattern) has been identified. The data structure ensures that for any given word in the vocabulary, all its occurrences across different frames can be quickly ascertained by referencing its associated place set.

These Binary Words are stored in a hash table allowing for retrieval in constant time O(1). This speed makes it possible for BoW3D to quickly check each incoming point clouds against the previously stored binary words to check for possible loop closure candidates. Once a loop closure candidate is detected the current scan is matched against the one corresponding to the loop closure position using the Link3D features to recover the relative transformation as a loop closure constraint.

In our experiment we corroborated the speed of this technique being able to operate in concert with CT-ICP with little to no overhead. However we also noticed an abundance of spurious matches. Since the only filtering step implemented by BoW3D is based on the ratio of distinctive features present in each scan we also added the following constraints to mitigate this problem:

- Minimum time distance between current and retrieved scan
- Maximum recovered distance during loop closure

The first constraint is instituted to preclude matches with scans acquired in close temporal proximity. A threshold of 40 seconds was established, premised on our observation that such a duration was insufficient to manifest any drift. Imposing loop corrections within this interval only exacerbated computational demands without enhancing the system's precision. The second constraint serves as a safeguard against false matches in map regions exhibiting self-similarity. Given that our trials without loop closure never exhibited drifts beyond 5 meters, we instituted this value as the threshold. Any loop correction exceeding this range was inherently deemed dubious.

5.2 Loop Correction

To act on the loop closure constraints retrieved by BoW3D, we utilized Voxgraph [38] as our optimization backend. This tool is adept at transforming sparse point clouds, commonly sourced from LiDAR SLAM, into dense, consistent, and detailed meshes of the environment.

At its core, Voxgraph operates on a voxelized truncated signed distance field (TSDF)[106]. In this grid-like structure, each voxel encodes the shortest distance to the nearest surface. Depending on the side of the surface the voxel is positioned, distances are either marked as positive ('empty' side) or negative ('occupied' side). By truncating distances beyond a specific threshold, computational efficiency is achieved, while also curbing the influence of noise.

The environment is modularly represented through submaps in Voxgraph. A submap is initiated when the robot traverses beyond a designated distance from the inception of the prevailing submap or after integrating a predetermined number of measurements. Each submap maintains its individual TSDF, allowing for localized dense reconstructions.

Concurrently, Voxgraph crafts a pose graph where nodes represent submaps and edges signify spatial constraints, often stemming from sensor readings or detected loop closures. The optimization of this graph refines the relative poses of submaps, ensuring a better alignment with the collected data.

In our tests we wanted to isolate the effects of BoW3D on global consistency so we disabled any pose graph optimization step in Voxgraph except for the loop closure constraints provided by BoW3D. The full pose graph at the end of the tests in the cave map can be seen in Figure 5.2.

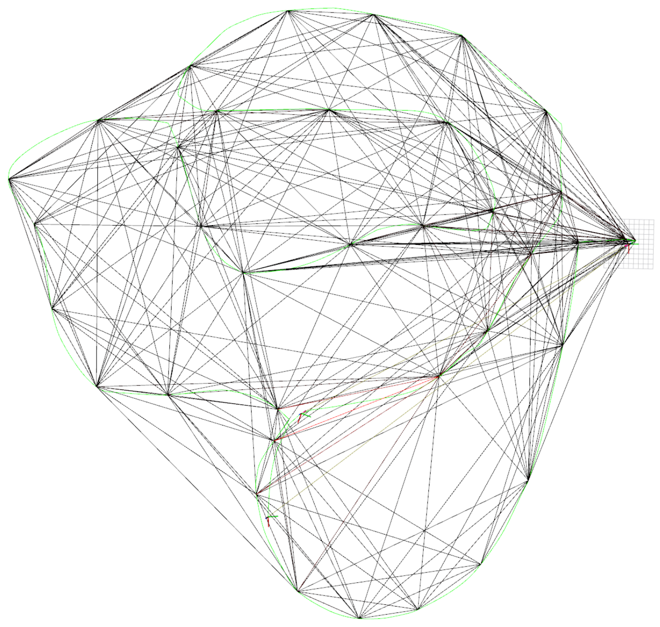
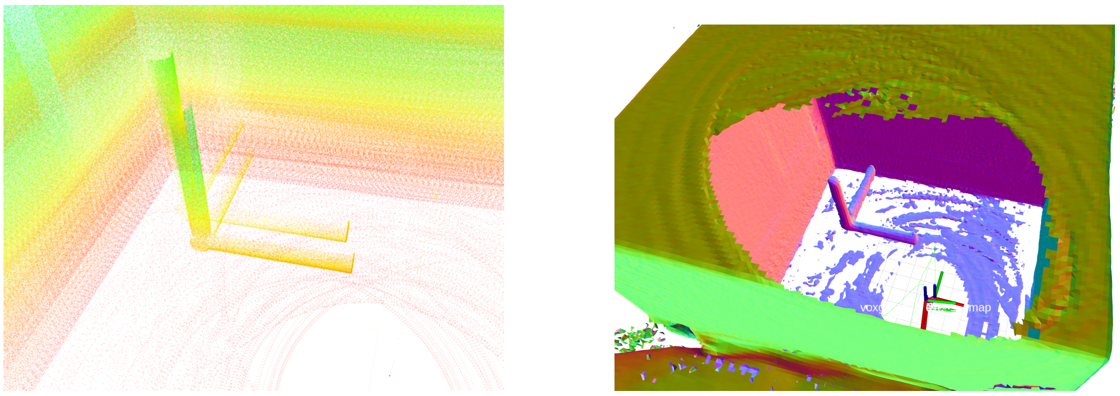


Figure 5.2: Pose graph generated by Voxgraph in the cave environment.
In black odometry constraints and in red the loop closure constraints. The green line is the trajectory of the drone computed by CT-ICP

5.3 Results

We performed our tests to validate the effectiveness of BoW3D in the cave map as it was the map where we saw the highest level of global inconsistency in the final point cloud and consequent ghosting artifacts. We used the same data collected in our previous tests but we now used our pipeline detailed in Figure 5.1 to process them.

The main limitation of using Voxgraph as our backend is that it is only able to apply the loop closure corrections to the dense submaps it reconstructs. Consequently we evaluated the global consistency on the final mesh generated by Voxgraph, with and without loop closure active. Figure 5.3 shows qualitatively how the loop closure is able to correct the ghosting artifacts present in the original point cloud.



(a) Ghosting artifacts from accumulated drift

(b) Loop closure effects on the final reconstructed mesh

Figure 5.3: BoW3D in concert with Voxgraph is capable of effectively correct the accumulated drift in the most challenging map.

Figure 5.4 shows the point-to-mesh error distribution of the final map reconstructed by Voxgraph with and without loop closure. From our analysis we measured a 22.8% improvement in the number of outliers by activating the loop closure strategy detailed in Figure 5.1. However, for Voxgraph to run in real time we had to set the size of the voxels to 0.2m. Such a coarse reconstruction generates noisy meshes and consequently the outlier percentage without loop closure active is equal to 41.59%, a substantial increase compared to the overall consistency of the unprocessed point cloud obtained by CT-ICP.

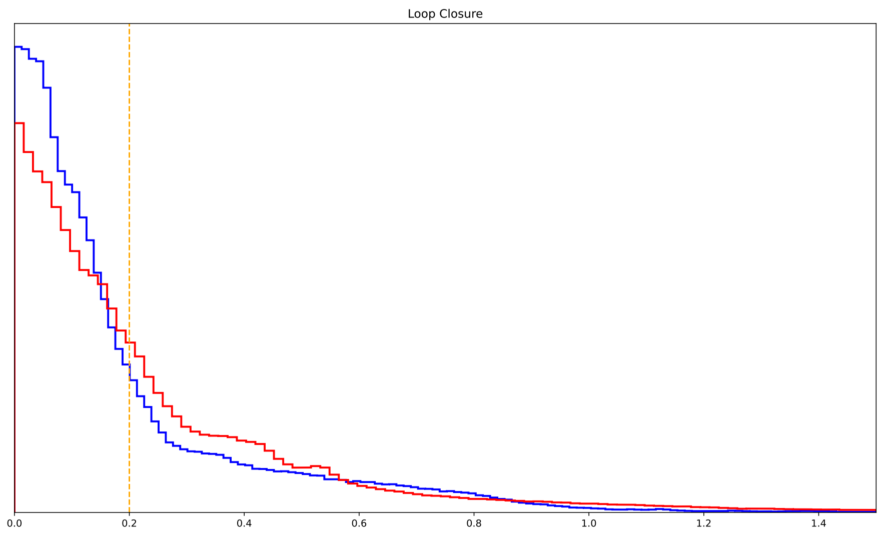


Figure 5.4: Point-to-mesh error distribution in our Loop Closure test
In red without loop closure active and in blue with loop closure active

Conclusions and future work

In our research, we underscored the significant advantages LiDAR technology can offer in disaster management. Notably, while the majority of UAVs utilized for indoor exploration predominantly depended on human pilots due to a deficiency in autonomy, our findings demonstrate that contemporary MEMS-based LiDARs enhance localization precision. This facilitates real-time three-dimensional reconstructions of indoor settings for first responders. Given the pronounced accuracy and robustness of LiDAR, its value is further evident considering its extensive application in sectors like surveying and autonomous transportation. Additionally, the maturity of these sensors now permits their integration into compact aerial platforms.

Nevertheless, our evaluation revealed that LiDAR SLAM systems, initially designed for terrestrial platforms, face challenges when adapted to agile UAVs. Our trials pinpointed CT-ICP as the superior LiDAR SLAM method. We subsequently refined it by embedding it within a comprehensive framework capable of real-time loop closure detection and rectification using BoW3D and Voxgraph. This framework emphasized the imperative for precise dense mesh reconstructions, as existing pipelines compromise the original point cloud's accuracy.

Our endeavor underscored the vital role of effective simulation pipelines. The plugin developed for the Livox Mid360 simulation was pivotal in our data acquisition, enabling sensor comparisons that would have been impractical on our compact indoor UAV. We aspire to further enhance simulation efficiency in our subsequent work through GPU acceleration.

In conclusion, our research has illuminated the transformative potential of LiDAR technology, especially in the realm of disaster management and indoor exploration. The enhancements in precision, autonomy, and real-time response underscore the pivotal role of advanced sensor technology in driving forward the capabilities of UAVs. As technology continues to evolve, it is imperative for research to stay aligned with these advancements. Our work contributes to this ongoing dialogue, providing insights and avenues for further exploration in the field of UAV technologies and their applications.

Bibliography

[1] Robin R Murphy. *Disaster robotics.* MIT press, 2014 (cit. on p. 1).

[2] J. Abouaf. «Trial by fire: teleoperated robot targets Chernobyl». In: *IEEE Comput. Graphics Appl.* 18.4 (July 1998), pp. 10–14. DOI: 10.1109/38.689654 (cit. on p. 1).

[3] Geert-Jan M Kruijff et al. «Rescue robots at earthquake-hit Mirandola, Italy: A field report». In: *2012 IEEE international symposium on safety, security, and rescue robotics (SSRR).* IEEE. 2012, pp. 1–8 (cit. on p. 3).

[4] Daniel P Stormont and Vicki H Allan. «Managing risk in disaster scenarios with autonomous robots». In: *Journal of Systemics, Cybernetics and Informatics* 7 (2009), pp. 66–71 (cit. on p. 3).

[5] Jurgen Everaerts et al. «The use of unmanned aerial vehicles (UAVs) for remote sensing and mapping». In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 37.2008 (2008), pp. 1187–1192 (cit. on p. 3).

[6] Stuart M Adams and Carol J Friedland. «A survey of unmanned aerial vehicle (UAV) usage for imagery collection in disaster research and management». In: *9th international workshop on remote sensing for disaster response.* Vol. 8. 2011, pp. 1–8 (cit. on p. 3).

[7] Robin R Murphy, Satoshi Tadokoro, and Alexander Kleiner. «Disaster robotics». In: *Springer handbook of robotics* (2016), pp. 1577–1604 (cit. on pp. 3, 6).

[8] Phillipp Jende, Francesco Nex, Markus Gerke, and George Vosselman. «A fully automatic approach to register mobile mapping and airborne imagery to support the correction of platform trajectories in GNSS-denied urban areas». In: *ISPRS J. Photogramm. Remote Sens.* 141 (July 2018), pp. 86–99. ISSN: 0924-2716. DOI: 10.1016/j.isprsjprs.2018.04.017 (cit. on p. 4).

[9] Sudipta Chowdhury, Adindu Emelogu, Mohammad Marufuzzaman, Sarah G. Nurre, and Linkan Bian. «Drones for disaster response and relief operations: A continuous approximation model». In: *Int. J. Prod. Econ.* 188 (June 2017), pp. 167–184. ISSN: 0925-5273. DOI: `10.1016/j.ijpe.2017.03.024` (cit. on p. 4).

[10] Heajung Min, Kyung Min Han, and Young J Kim. «OctoMap-RT: Fast Probabilistic Volumetric Mapping Using Ray-Tracing GPUs». In: *IEEE Robotics and Automation Letters* (2023) (cit. on p. 4).

[11] Elia Kaufmann, Leonard Bauersfeld, Antonio Loquercio, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. «Champion-level drone racing using deep reinforcement learning». In: *Nature* 620 (Aug. 2023), pp. 982–987. ISSN: 1476-4687. DOI: `10.1038/s41586-023-06419-4` (cit. on p. 4).

[12] G. Lachapelle. «GNSS Indoor Location Technologies». In: *Journal of Global Positioning Systems* (2004) (cit. on p. 5).

[13] Rainer Mautz. «Overview of current indoor positioning systems». In: *Geodezija ir kartografija* 35.1 (2009), pp. 18–22. DOI: `10.3846/1392-1541.2009.35.18-22` (cit. on p. 6).

[14] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. «The EuRoC micro aerial vehicle datasets». In: *The International Journal of Robotics Research* (2016). DOI: `10.1177/0278364915620033` (cit. on p. 6).

[15] Cyrill Stachniss, John J Leonard, and Sebastian Thrun. «Simultaneous localization and mapping». In: *Springer Handbook of Robotics* (2016), pp. 1153–1176 (cit. on p. 8).

[16] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics.* Intelligent Robotics and Autonomous Agents series. MIT Press, 2005. ISBN: 9780262201629 (cit. on pp. 8, 11).

[17] Randall Smith, Matthew Self, and Peter Cheeseman. «Estimating Uncertain Spatial Relationships in Robotics». In: *Autonomous Robot Vehicles.* New York, NY, USA: Springer, New York, NY, 1990, pp. 167–193. DOI: `10.1007/978-1-4613-8997-2_14` (cit. on p. 10).

[18] Randall C Smith and Peter Cheeseman. «On the representation and estimation of spatial uncertainty». In: *The international journal of Robotics Research* 5.4 (1986), pp. 56–68 (cit. on p. 10).

[19] Kevin Murphy and Stuart Russell. «Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks». In: *Sequential Monte Carlo Methods in Practice.* New York, NY, USA: Springer, New York, NY, 2001, pp. 499–515. DOI: `10.1007/978-1-4757-3437-9_24` (cit. on p. 10).

[20] Mark Pupilli and Andrew Calway. «Real-Time Camera Tracking Using a Particle Filter». In: *British Machine Vision Conference.* 2005 (cit. on p. 10).

[21] Sebastian Thrun, Yufeng Liu, Daphne Koller, Andrew Y. Ng, Zoubin Ghahramani, and Hugh Durrant-Whyte. «Simultaneous Localization and Mapping with Sparse Extended Information Filters». In: *Int. J. Rob. Res.* 23.7-8 (Aug. 2004), pp. 693–716. ISSN: 0278-3649. DOI: 10.1177/0278364904045479 (cit. on p. 11).

[22] Viorela Ila, Josep M. Porta, and Juan Andrade-Cetto. «Information-Based Compact Pose SLAM». In: *IEEE Trans. Rob.* 26.1 (Nov. 2009), pp. 78–93. ISSN: 1941-0468. DOI: 10.1109/TRO.2009.2034435 (cit. on p. 11).

[23] Anastasios I. Mourikis and Stergios I. Roumeliotis. «A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation». In: *Proceedings 2007 IEEE International Conference on Robotics and Automation.* IEEE, Apr. 2007, pp. 3565–3572. DOI: 10.1109/ROBOT.2007.364024 (cit. on p. 11).

[24] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J. Leonard. «Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age». In: *IEEE Trans. Rob.* 32.6 (Dec. 2016), pp. 1309–1332. ISSN: 1941-0468. DOI: 10.1109/TRO.2016.2624754 (cit. on p. 11).

[25] Frank Dellaert and Michael Kaess. *Factor Graphs for Robot Perception.* Now Publishers Inc., Aug. 2017 (cit. on p. 12).

[26] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization.* New York, NY, USA: Springer. ISBN: 978-0-387-40065-5 (cit. on p. 13).

[27] Donald W. Marquardt. «An Algorithm for Least-Squares Estimation of Nonlinear Parameters». In: *Journal of the Society for Industrial and Applied Mathematics* (July 2006) (cit. on p. 13).

[28] M. J. D. Powell. «A New Algorithm for Unconstrained Optimization». In: *Nonlinear Programming.* Cambridge, MA, USA: Academic Press, Jan. 1970, pp. 31–65. ISBN: 978-0-12-597050-1. DOI: 10.1016/B978-0-12-597050-1.50006-3 (cit. on p. 13).

[29] Diederik P. Kingma and Jimmy Ba. «Adam: A Method for Stochastic Optimization». In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings.* Ed. by Yoshua Bengio and Yann LeCun. 2015 (cit. on p. 13).

[30] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John Leonard, and Frank Dellaert. «iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering». In: *2011 IEEE International Conference on Robotics and Automation.* IEEE, May 2011, pp. 3281–3288. DOI: `10.1109/ICRA.2011.5979641` (cit. on pp. 13, 14).

[31] Michael Kaess, Viorela Ila, Richard Roberts, and Frank Dellaert. «The Bayes Tree: An Algorithmic Foundation for Probabilistic Robot Mapping». In: *Algorithmic Foundations of Robotics IX.* Berlin, Germany: Springer, 2010, pp. 157–173. DOI: `10.1007/978-3-642-17452-0_10` (cit. on pp. 13, 30, 38).

[32] Chang Chen, Hua Zhu, Menggang Li, and Shaoze You. «A Review of Visual-Inertial Simultaneous Localization and Mapping from Filtering-Based and Optimization-Based Perspectives». In: *Robotics* 7.3 (Aug. 2018), p. 45. ISSN: 2218-6581. DOI: `10.3390/robotics7030045` (cit. on p. 14).

[33] Hauke Strasdat, J. M. M. Montiel, and Andrew J. Davison. «Real-time monocular SLAM: Why filter?» In: *2010 IEEE International Conference on Robotics and Automation.* IEEE, May 2010, pp. 2657–2664. DOI: `10.1109/ROBOT.2010.5509636` (cit. on p. 14).

[34] Amay Saxena, Chih-Yuan Chiu, Ritika Shrivastava, Joseph Menke, and Shankar Sastry. «Simultaneous Localization and Mapping: Through the Lens of Nonlinear Optimization». In: *IEEE Rob. Autom. Lett.* 7.3 (June 2022), pp. 7148–7155. ISSN: 2377-3766. DOI: `10.1109/LRA.2022.3181409` (cit. on p. 14).

[35] Johannes Lutz Schönberger and Jan-Michael Frahm. «Structure-from-Motion Revisited». In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2016 (cit. on p. 14).

[36] Frank Dellaert. «Factor Graphs and GTSAM: A Hands-on Introduction». In: *Georgia Institute of Technology* (Sept. 2012) (cit. on pp. 14, 42, 45).

[37] Rainer Kümmerle, Giorgio Grisetti, Hauke Strasdat, Kurt Konolige, and Wolfram Burgard. «G2o: A general framework for graph optimization». In: *2011 IEEE International Conference on Robotics and Automation.* IEEE, May 2011, pp. 3607–3613. DOI: `10.1109/ICRA.2011.5979949` (cit. on pp. 14, 30).

[38] Victor Reijgwart, Alexander Millane, Helen Oleynikova, Roland Siegwart, Cesar Cadena, and Juan Nieto. «Voxgraph: Globally consistent, volumetric mapping using signed distance function submaps». In: *IEEE Robotics and Automation Letters* 5.1 (2019), pp. 227–234 (cit. on pp. 14, 42, 47).

[39] Georg Klein and David Murray. «Parallel Tracking and Mapping for Small AR Workspaces». In: *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*. IEEE, Nov. 2007, pp. 225–234. DOI: 10.1109/ISMAR.2007.4538852 (cit. on p. 14).

[40] D. Nister, O. Naroditsky, and J. Bergen. «Visual odometry». In: *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.* Vol. 1. IEEE, June 2004, p. I. DOI: 10.1109/CVPR.2004.1315094 (cit. on p. 15).

[41] R. Bunschoten and B. Krose. «Visual odometry from an omnidirectional vision system». In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*. Vol. 1. IEEE, Sept. 2003, 577–583vol.1. DOI: 10.1109/ROBOT.2003.1241656 (cit. on p. 15).

[42] Takeo Kanade, Chuck Thorpe, and William Whittaker. «Autonomous land vehicle project at CMU». In: *CSC '86: Proceedings of the 1986 ACM fourteenth annual conference on Computer science*. New York, NY, USA: Association for Computing Machinery, Feb. 1986, pp. 71–80. DOI: 10.1145/324634.325197 (cit. on p. 16).

[43] C. Thorpe, M. Herbert, T. Kanade, and S. Shafer. «Toward autonomous driving: the CMU Navlab. I. Perception». In: *IEEE Expert* 6.4 (1991), pp. 31–42. DOI: 10.1109/64.85919 (cit. on p. 16).

[44] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. «A Comprehensive Survey of Visual SLAM Algorithms». In: *Robotics* 11.1 (Feb. 2022), p. 24. ISSN: 2218-6581. DOI: 10.3390/robotics11010024 (cit. on p. 17).

[45] Leyao Huang. «Review on LiDAR-based SLAM Techniques». In: *2021 International Conference on Signal Processing and Machine Learning (CONF-SPML)*. IEEE, Nov. 2021, pp. 163–168. DOI: 10.1109/CONF-SPML54095.2021.00040 (cit. on pp. 17, 28).

[46] Franco Hidalgo and Thomas Bräunl. «Review of underwater SLAM techniques». In: *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*. IEEE, Feb. 2015, pp. 306–311. DOI: 10.1109/ICARA.2015.7081165 (cit. on p. 17).

[47] Ziyang Hong, Yvan Petillot, and Sen Wang. «RadarSLAM: Radar based Large-Scale SLAM in All Weathers». In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, pp. 2020–24. DOI: 10.1109/IROS45743.2020.9341287 (cit. on p. 17).

[48] Andreas Geiger, Philip Lenz, and Raquel Urtasun. «Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite». In: *Conference on Computer Vision and Pattern Recognition (CVPR).* 2012 (cit. on pp. 17, 20, 31).

[49] Thomas Schöps, Viktor Larsson, Marc Pollefeys, and Torsten Sattler. «Why Having 10,000 Parameters in Your Camera Model is Better Than Twelve». In: *arXiv* (Dec. 2019). DOI: `10.48550/arXiv.1912.02908`. eprint: `1912.02908` (cit. on p. 19).

[50] David G. Lowe. «Distinctive Image Features from Scale-Invariant Keypoints». In: *Int. J. Comput. Vision* 60.2 (Nov. 2004), pp. 91–110. ISSN: 1573-1405. DOI: `10.1023/B:VISI.0000029664.99615.94` (cit. on p. 19).

[51] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. «ORB: An efficient alternative to SIFT or SURF». In: *2011 International Conference on Computer Vision.* IEEE, pp. 06–13. DOI: `10.1109/ICCV.2011.6126544` (cit. on pp. 19, 45).

[52] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. «Speeded-Up Robust Features (SURF)». In: *Comput. Vision Image Understanding* 110.3 (June 2008), pp. 346–359. ISSN: 1077-3142. DOI: `10.1016/j.cviu.2007.09.014` (cit. on p. 19).

[53] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. «Superpoint: Self-supervised interest point detection and description». In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops.* 2018, pp. 224–236 (cit. on p. 19).

[54] Michał Tyszkiewicz, Pascal Fua, and Eduard Trulls. «DISK: Learning local features with policy gradient». In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 14254–14265 (cit. on p. 19).

[55] Jerome Revaud, Cesar De Souza, Martin Humenberger, and Philippe Weinzaepfel. «R2d2: Reliable and repeatable detector and descriptor». In: *Advances in neural information processing systems* 32 (2019) (cit. on p. 19).

[56] Yuhe Jin, Dmytro Mishkin, Anastasiia Mishchuk, Jiri Matas, Pascal Fua, Kwang Moo Yi, and Eduard Trulls. «Image matching across wide baselines: From paper to practice». In: *International Journal of Computer Vision* 129.2 (2021), pp. 517–547 (cit. on p. 19).

[57] Carlos Campos, Richard Elvira, Juan J Gómez Rodriguez, José MM Montiel, and Juan D Tardós. «Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam». In: *IEEE Transactions on Robotics* 37.6 (2021), pp. 1874–1890 (cit. on p. 20).

[58] Paul J Besl and Neil D McKay. «Method for registration of 3-D shapes». In: *Sensor fusion IV: control paradigms and data structures*. Vol. 1611. Spie. 1992, pp. 586–606 (cit. on pp. 21, 29).

[59] Bolz Wolfgang, Phillipp Fanta-Jende, and Simon Schwaiger. «Sensor Selection for Unmanned Aerial Vehicles in Uncooperative Indoor Environments». 2023 (cit. on p. 22).

[60] Chris Urmson et al. «Tartan racing: A multi-modal approach to the darpa urban challenge». In: (2007) (cit. on p. 23).

[61] Behnam Behroozpour, Phillip AM Sandborn, Ming C Wu, and Bernhard E Boser. «Lidar system architectures and circuits». In: *IEEE Communications Magazine* 55.10 (2017), pp. 135–142 (cit. on p. 23).

[62] Dingkang Wang, Connor Watkins, and Huikai Xie. «MEMS mirrors for LiDAR: A review». In: *Micromachines* 11.5 (2020), p. 456 (cit. on p. 23).

[63] Xiaosheng Zhang. «Laser Chirp Linearization and Phase Noise Compensation for Frequency-modulated Continuouswave LiDAR». In: *University of California*. 2021 (cit. on p. 23).

[64] Yong Liu and Hao Hu. «Silicon optical phased array with a 180-degree field of view for 2D optical beam steering». In: *Optica* 9.8 (2022), pp. 903–907 (cit. on pp. 23, 24).

[65] David S. Hall. *High definition LiDAR system*. Apr. 2020. URL: `https://uspto.report/patent/grant/RE47,942` (cit. on p. 24).

[66] Jiarong Lin and Fu Zhang. «Loam livox: A fast, robust, high-precision LiDAR odometry and mapping package for LiDARs of small FoV». In: *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2020, pp. 3126–3131 (cit. on pp. 25, 28).

[67] Nina Varney, Vijayan K Asari, and Quinn Graehling. «DALES: A large-scale aerial LiDAR data set for semantic segmentation». In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*. 2020, pp. 186–187 (cit. on p. 28).

[68] F Vultaggio, F d'Apolito, C Sulzbachner, and P Fanta-Jende. «SIMULATION OF LOW-COST MEMS-LIDAR AND ANALYSIS OF ITS EFFECT ON THE PERFORMANCES OF STATE-OF-THE-ART SLAMS». In: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 48 (2023), pp. 539–545 (cit. on pp. 28, 33).

[69] Han Wang, Chen Wang, Chun-Lin Chen, and Lihua Xie. «F-loam: Fast lidar odometry and mapping». In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 4390–4396 (cit. on p. 28).

[70] Tixiao Shan and Brendan Englot. «Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain». In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 4758–4765 (cit. on p. 28).

[71] Ignacio Vizzo, Xieyuanli Chen, Nived Chebrolu, Jens Behley, and Cyrill Stachniss. «Poisson surface reconstruction for LiDAR odometry and mapping». In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 5624–5630 (cit. on p. 29).

[72] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. «ROS: an open-source Robot Operating System». In: *ICRA workshop on open source software*. Vol. 3. 3.2. Kobe, Japan. 2009, p. 5 (cit. on p. 29).

[73] Yue Pan, Pengchuan Xiao, Yujie He, Zhenlei Shao, and Zesong Li. «MULLS: Versatile LiDAR SLAM via multi-metric linear least square». In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2021, pp. 11633–11640 (cit. on p. 29).

[74] Pierre Dellenbach, Jean-Emmanuel Deschaud, Bastien Jacquet, and François Goulette. «CT-ICP: Real-time elastic LiDAR odometry with loop closure». In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 5580–5586 (cit. on pp. 29, 42, 45).

[75] Anas Al-Nuaimi, Wilder Lopes, Paul Zeller, Adrian Garcea, Cassio Lopes, and Eckehard Steinbach. «Analyzing LiDAR scan skewing and its impact on scan matching». In: *2016 International Conference on Indoor Positioning and Indoor Navigation (IPIN)*. IEEE, pp. 04–07. DOI: `10.1109/IPIN.2016.7743598` (cit. on p. 29).

[76] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. «Kiss-icp: In defense of point-to-point icp–simple, accurate, and robust registration if done the right way». In: *IEEE Robotics and Automation Letters* 8.2 (2023), pp. 1029–1036 (cit. on pp. 30, 42, 45).

[77] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. «Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping». In: *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE. 2020, pp. 5135–5142 (cit. on pp. 30, 42).

[78] Ji Zhang and Sanjiv Singh. «LOAM: Lidar odometry and mapping in real-time.» In: *Robotics: Science and systems*. Vol. 2. 9. Berkeley, CA. 2014, pp. 1–9 (cit. on pp. 31, 38).

[79] Wei Xu, Yixi Cai, Dongjiao He, Jiarong Lin, and Fu Zhang. «Fast-lio2: Fast direct lidar-inertial odometry». In: *IEEE Transactions on Robotics* 38.4 (2022), pp. 2053–2073 (cit. on pp. 31, 42, 45).

[80] Wei Xu and Fu Zhang. «Fast-lio: A fast, robust lidar-inertial odometry package by tightly-coupled iterated kalman filter». In: *IEEE Robotics and Automation Letters* 6.2 (2021), pp. 3317–3324 (cit. on p. 31).

[81] Igal Galperin and Ronald L Rivest. «Scapegoat trees». In: *Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms*. 1993, pp. 165–174 (cit. on p. 31).

[82] Jinyong Jeong, Younggun Cho, Young-Sik Shin, Hyunchul Roh, and Ayoung Kim. «Complex Urban Dataset with Multi-level Sensors from Highly Diverse Urban Environments». In: *International Journal of Robotics Research* 38.6 (2019), pp. 642–657 (cit. on p. 31).

[83] Albert S Huang, Matthew Antone, Edwin Olson, Luke Fletcher, David Moore, Seth Teller, and John Leonard. «A high-rate, heterogeneous data set from the darpa urban challenge». In: *The International Journal of Robotics Research* 29.13 (2010), pp. 1595–1601 (cit. on p. 31).

[84] Xinyu Huang, Xinjing Cheng, Qichuan Geng, Binbin Cao, Dingfu Zhou, Peng Wang, Yuanqing Lin, and Ruigang Yang. *The ApolloScape Dataset for Autonomous Driving*. [Online; accessed 9. Oct. 2023]. 2018. URL: `https://openaccess.thecvf.com/content_cvpr_2018_workshops/w14/html/Huang_The_ApolloScape_Dataset_CVPR_2018_paper.html` (cit. on p. 31).

[85] Nicholas Carlevaris-Bianco, Arash K. Ushani, and Ryan M. Eustice. «University of Michigan North Campus long-term vision and lidar dataset». In: *International Journal of Robotics Research* 35.9 (2015), pp. 1023–1035 (cit. on p. 31).

[86] Thierry Peynot, Steve Scheding, and Sami Terho. «The Marulan Data Sets: Multi-sensor Perception in a Natural Environment with Challenging Conditions». In: *Int. J. Rob. Res.* 29.13 (Nov. 2010), pp. 1602–1607. ISSN: 0278-3649. DOI: `10.1177/0278364910384638` (cit. on p. 31).

[87] Maurice Fallon, Hordur Johannsson, Michael Kaess, and John J Leonard. «The mit stata center dataset». In: *The International Journal of Robotics Research* 32.14 (2013), pp. 1695–1699 (cit. on p. 31).

[88] Riccardo Giubilato, Wolfgang Stürzl, Armin Wedler, and Rudolph Triebel. «Challenges of SLAM in extremely unstructured environments: the DLR Planetary Stereo, Solid-State LiDAR, Inertial Dataset». In: *IEEE Robotics and Automation Letters* (2022), pp. 1–8. DOI: `10.1109/LRA.2022.3188118` (cit. on p. 32).

[89] Li Qingqing, Yu Xianjia, Jorge Pena Queralta, and Tomi Westerlund. «Multi-modal lidar dataset for benchmarking general-purpose localization and mapping algorithms». In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2022, pp. 3837–3844 (cit. on p. 32).

[90] N. Koenig and A. Howard. «Design and use paradigms for Gazebo, an open-source multi-robot simulator». In: *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*. Vol. 3. IEEE, Sept. 2004, 2149–2154vol.3. DOI: `10.1109/IROS.2004.1389727` (cit. on p. 32).

[91] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. «AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles». In: *Field and Service Robotics*. Cham, Switzerland: Springer, Nov. 2017, pp. 621–635. DOI: `10.1007/978-3-319-67361-5_40` (cit. on p. 32).

[92] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. «CARLA: An Open Urban Driving Simulator». In: *Conference on Robot Learning*. PMLR, Oct. 2017, pp. 1–16. URL: `https://proceedings.mlr.press/v78/dosovitskiy17a.html` (cit. on p. 32).

[93] Fanze Kong, Xiyuan Liu, Benxu Tang, et al. «MARSIM: A Light-Weight Point-Realistic Simulator for LiDAR-Based UAVs». In: *IEEE Rob. Autom. Lett.* 8.5 (Apr. 2023), pp. 2954–2961. ISSN: 2377-3766. DOI: `10.1109/LRA.2023.3264163` (cit. on p. 33).

[94] David Prokhorov, Dmitry Zhukov, Olga Barinova, Konushin Anton, and Anna Vorontsova. «Measuring robustness of Visual SLAM». In: *2019 16th International Conference on Machine Vision Applications (MVA)*. IEEE, May 2019, pp. 1–6. DOI: `10.23919/MVA.2019.8758020` (cit. on p. 34).

[95] F. Lu and E. Milios. «Globally Consistent Range Scan Alignment for Environment Mapping». In: *Autonomous Robots* 4.4 (Oct. 1997), pp. 333–349. ISSN: 1573-7527. DOI: `10.1023/A:1008854305733` (cit. on p. 35).

[96] Shinji Umeyama. «Least-squares estimation of transformation parameters between two point patterns». In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 13.04 (1991), pp. 376–380 (cit. on p. 35).

[97] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM*. `https://github.com/MichaelGrupp/evo` (cit. on p. 35).

[98] Mahdi Chamseddine, Jason Rambach, Didier Stricker, and Oliver Wasenmuller. «Ghost Target Detection in 3D Radar Data using Point Cloud based Deep Neural Network». In: *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, Jan. 2021, pp. 10398–10403. DOI: `10.1109/ICPR48806.2021.9413247` (cit. on p. 35).

[99] Guillermo Gallego, Tobi Delbrück, Garrick Orchard, et al. «Event-Based Vision: A Survey». In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.1 (July 2020), pp. 154–180. ISSN: 1939-3539. DOI: `10.1109/TPAMI.2020.3008413` (cit. on p. 36).

[100] Giseop Kim and Ayoung Kim. «Scan Context: Egocentric Spatial Descriptor for Place Recognition Within 3D Point Cloud Map». In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, Oct. 2018, pp. 4802–4809. DOI: `10.1109/IROS.2018.8593953` (cit. on pp. 41, 45).

[101] Yunge Cui, Xieyuanli Chen, Yinlong Zhang, Jiahua Dong, Qingxiao Wu, and Feng Zhu. «Bow3d: Bag of words for real-time loop closing in 3d lidar slam». In: *IEEE Robotics and Automation Letters* 8.5 (2022), pp. 2828–2835 (cit. on pp. 41, 45).

[102] M. Cella, F. D'Apolito, P. Fanta-Jende, and C. Sulzbachner. «FUELING GLOCAL: OPTIMIZATION-BASED PATH PLANNING FOR INDOOR UAVS IN AN AUTONOMOUS EXPLORATION FRAMEWORK». In: *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* XLVIII-1-W1-2023 (May 2023), pp. 85–91. ISSN: 1682-1750. DOI: `10.5194/isprs-archives-XLVIII-1-W1-2023-85-2023` (cit. on p. 42).

[103] Lin Li, Xin Kong, Xiangrui Zhao, Tianxin Huang, Wanlong Li, Feng Wen, Hongbo Zhang, and Yong Liu. «SSC: Semantic scan context for large-scale place recognition». In: *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2021, pp. 2092–2099 (cit. on p. 45).

[104] Yunge Cui, Yinlong Zhang, Jiahua Dong, Haibo Sun, and Feng Zhu. «Link3d: Linear keypoints representation for 3d lidar point cloud». In: *arXiv preprint arXiv:2206.05927* (2022) (cit. on p. 45).

[105] David G Lowe. «Distinctive image features from scale-invariant keypoints». In: *International journal of computer vision* 60 (2004), pp. 91–110 (cit. on p. 45).

[106] Helen Oleynikova, Zachary Taylor, Marius Fehr, Roland Siegwart, and Juan Nieto. «Voxblox: Incremental 3d euclidean signed distance fields for on-board mav planning». In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 1366–1373 (cit. on p. 47).

Acronyms

ATE Absolute Trajectory Error

EKF Extended Kalman Filter

FoV Field Of View

GNSS Global Navigation Satellite System

GPS Global Positioning System

ICP Iterative Closest Point

IMU Inertial Measurement Unit

INS Inertial Navigation System

IPS Indoor Positioning Systems

kNN k-Nearest Neighbor

LiDAR Light Detection and Ranging

MAP Maximum A Posteriori

MEMS Micro Electro-Mechanical Systems

OPA Optical Phased Arrays

PTAM Parallel Tracking And Mapping

RMSE Root Mean Squared Error

RPE Relative Pose Error

SfM	Structure from Motion
SLAM	Simultaneous Localization And Mapping
SWAP	Size Weight And Power
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle