Evaluation of 3Di cameras in UGV applications

DIPLOMARBEIT

Ausgeführt zum Zwecke der Erlangung des akademischen Grades eines
Diplom-Ingenieurs (Dipl.-Ing.)

unter der Leitung von

Univ.-Prof. Dr.sc.techn. Georg Schitter
Projektass. Dipl.-Ing. Peter Gsellmann

eingereicht an der
Technischen Universität Wien
Fakultät für Elektrotechnik und Informationstechnik
Institut für Automatisierungs- und Regelungstechnik

von
Karin Egretzberger
Matrikelnummer: 51868030

Wien, im Mai 2025

Technische Universität Wien
Karlsplatz 13, 1040 Wien, Österreich

Abstract

Unmanned ground vehicles (UGVs) enable autonomous operation in challenging and dynamic environments, making them valuable for applications such as search-and-rescue, industrial automation, and exploration. The effectiveness of UGVs depends on robust sensing and mapping capabilities, typically provided by sensor systems such as Light Detection and Ranging (LiDAR) sensors and vision-based cameras. However, these technologies have limitations. LiDAR, while accurate, is expensive and struggles with reflective surfaces, while stereo vision has depth estimation problems in low texture environments. Alternative or complementary technologies are needed to improve UGVs performance.

This study investigates the integration of 3Di cameras as an alternative sensing modality for UGV applications. The 3Di sensor captures high-resolution depth information, which is processed and incorporated into a LiDAR-based Simultaneous Localization and Mapping (SLAM) framework. The primary objective of this study is to assess the capability, accuracy, and repeatability of 3Di-based SLAM applications compared to a state-of-the-art LiDAR system. The data show that LiDAR achieves an average mapping error of 0.5 mm, 80% less than the 3Di camera. However, the camera accuracy is not affected by lighting. Both sensors encounter difficulties with transparent surfaces, but are capable of detecting light absorbing materials. The presence of uneven structures causes inaccuracies, especially with the 3Di camera. Though, the camera outperforms LiDAR in cliff detection, identifying cliffs up to 0.6 m away, while LiDAR does not detect them at all.

With the given SLAM settings, LiDAR provides high accuracy mapping but is not capable of detecting hazards such as cliffs. The 3Di overcomes these drawbacks and thus improves navigation safety, highlighting its potential as a complementary sensor for UGV applications.

Kurzfassung

Unbemannte Bodenfahrzeuge (Unmanned Ground Vehicles, UGVs) ermöglichen einen autonomen Betrieb in anspruchsvollen und dynamischen Umgebungen, was sie für Anwendungen wie Such- und Rettungsdienste, industrielle Automatisierung und Exploration interessant macht. Die Zuverlässigkeit von UGVs hängt von der Robustheit der Sensoren ab, wobei LiDAR und Stereokameras am weitesten verbreitet sind. Diese Technologien haben jedoch ihre Grenzen. LiDAR ist zwar genau, aber teuer und hat Probleme mit spiegelnden Oberflächen, während Stereokameras Probleme mit der Tiefenschätzung in Umgebungen mit geringer Textur haben. Alternative oder ergänzende Technologien sind erforderlich, um die Qualität von UGVs zu verbessern.

In dieser Arbeit wird die Integration einer 3Di Kamera als Sensor für UGV-Anwendungen untersucht. Der 3Di Sensor erfasst hochauflösende Tiefeninformationen, die verarbeitet und in ein LiDAR-basiertes SLAM-System integriert werden. Das Hauptziel ist die Fähigkeit, Genauigkeit und Wiederholbarkeit von 3Di-basiertem SLAM im Vergleich zu einem modernen LiDAR-System zu bewerten. Die Daten zeigen, dass Karten, die mit LiDAR-Daten erstellt werden, einen durchschnittlichen Fehler von 0.5 mm aufweisen, was 80% weniger ist als bei Kameradaten. Die Genauigkeit der Kamera wird jedoch nicht durch unterschiedliche Lichtverhältnisse beeinflusst. Beide Sensoren haben Schwierigkeiten mit transparenten Oberflächen, erkennen aber absorbierende Materialien. Unebene Strukturen führen vor allem bei der Kamera zu Ungenauigkeiten. Außerdem übertrifft die Kamera LiDAR bei der Erkennung von Klippen. Diese können bis zu 0.6 m im Voraus erkannt werden, während der LiDAR sie nicht erkennen kann.

Mit dem verwendeten SLAM Algorithmus bietet LiDAR eine genaue Kartierung. Die Fähigkeit der 3Di Kamera Gefahren wie Klippen zu erkennen, erhöht jedoch die Navigationssicherheit und unterstreicht ihr Potenzial als Sensor für UGV Anwendungen.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

AM	amplitude modulation. 9, 10
API	Application Programming Interface. 26
CMOS	complementary metal-oxide-semiconductor. 9, 10
CW	continuous wave. 9, 10
DBN	dynamic Bayesian network. 18
DDS	Data Distribution System. 25, 26, 32
FM	frequency modulation. 9
FOV	field of view. 29, 33, 37, 45, 48, 52, 67, 69, 70
ICP	Iterative Closest Point. 20, 21
IQR	interquartile range. 57, 59, 62, 65
iToF	indirect Time-of-Flight. v, 3, 4, 10, 13–15, 64, 68–70
LED	light-emitting diode. 10
LiDAR	Light Detection and Ranging. i, ii, v–vii, 2, 3, 8, 20, 21, 23, 24, 27–29, 31, 32, 35, 45, 49–53, 55, 59–70

NIR near infrared. 10

QoS Quality of Service. 26

RANSAC Random Sample Consensus. 38, 39

RCL ROS2 Client Library. 26

RMW ROS2 DDS Middleware. 26

ROS Robot Operation System. 24–26

ROS2 Robot Operation System 2. 24–26, 30, 32, 45, 51, 68

SLAM Simultaneous Localization and Mapping. i, ii, iv, vi, vii, 2–4, 16, 18–21, 23, 24, 27, 29, 30, 32, 33, 35, 38, 41, 44, 45, 48–53, 55, 60–62, 64–70

ToF Time-of-Flight. 5, 8–10, 12, 13, 16, 27–29, 69

UGV unmanned ground vehicle. i, ii, 2–4, 28, 29, 68, 69

CHAPTER 1

Introduction

The rapid development of Industry 4.0 creates novel opportunities and requirements for products and applications [1]. A variety of key technologies are facilitating new possibilities, including big data and augmented reality, as well as robotics applications.

In the contemporary industrial context, a multitude of robotic solutions are being developed and implemented, as displayed in Figure 1.1. Regardless of their function, accuracy class, limb form factor, or dimensions, these robots can be classified into two main categories [2]. Mobile robots, which move freely in inertial space, and fixed robots, which are rigidly attached to a surface. Although stationary robotic and automated systems are more prevalent, advanced, and widely utilized, the potential of mobile robots has yet to be fully recognized, making their development an increasingly promising area of research and innovation.

Mobile robots are utilized predominantly in service applications, such as surveillance, search-and-rescue, cargo transport, and environmental monitoring [4]. Those applications necessitate substantial autonomous mobility. From a mechanical point of

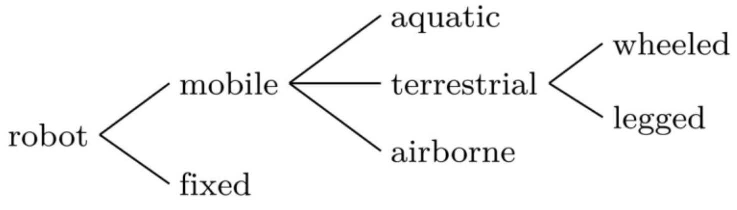


Figure 1.1: Classification of robot types by environmental interaction [3]

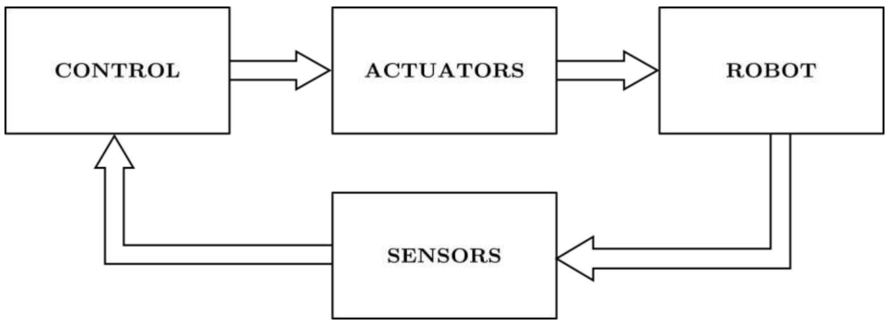


Figure 1.2: Components of a robotic system [5]

view, a mobile robot comprises one or more rigid bodies equipped with a locomotion system [5]. While a distinction can be drawn between aquatic, airborne, and terrestrial systems, ground-based robots, so-called unmanned ground vehicle (UGV)s, are in the focus of current research with regard to their navigation, actuator, and sensor systems.

Despite the presence of design discrepancies, all robotic systems are composed of the same fundamental components. As shown in Figure 1.2 a robotic system consists of three primary elements, additionally to the platform itself: control, sensors, and actuators. The control system functions as the robot intelligence, processing data collected by the sensors and generating commands for the actuators to execute a given task [6]. Actuators enable the robot to move and actually perform that task, while sensors provide information about the robot's surroundings and determine its position.

The sensors play a crucial role in UGV robotic systems, as a considerable proportion of these tasks are conducted in environments that are not known in advance. In contrast to UGVs operating in controlled settings, such as warehouse robots that rely on fixed routes, predefined maps, or external tracking systems, UGVs used in search-and-rescue, exploration, or agricultural applications must be capable of navigating dynamically in unknown environments without requiring modifications of their surroundings [7]. Consequently, sensor systems that do not rely on environmental adaptations are essential to ensure autonomous operation in unknown terrain.

1.1 Motivation

In order to achieve this level of autonomy, UGVs rely on multiple sensing technologies for environmental perception and navigation [8]. Commonly used depth-sensing technologies are LiDAR and stereo cameras. These sensors are fundamental for SLAM, a key technique that enables UGVs to create a map of an unknown environment while simultaneously estimating their own position within it [9]. The necessity for SLAM arises from the absence of external positioning systems such as GPS. In such cases, an UGV must rely solely on onboard sensors to determine its position and the surrounding environment. The absence of SLAM would leave an UGV unable to navigate autonomously in new environments, as it would lack both spatial awareness and the ability to plan efficient paths.

However, the accuracy and reliability of SLAM depend on the quality of the sensor data. Common sensors used in SLAM applications include LiDAR and RGB-D cameras, such as stereo vision or structured light cameras. The presence of errors can be attributed to the limitations of different sensor technologies. LiDAR, despite its high accuracy, is expensive and power-intensive, and can struggle with certain reflective or transparent surfaces [7]. Stereo vision requires high computational resources and performs poorly in low-texture or low-light conditions. These limitations create challenges for UGVs, particularly in dynamic, cluttered, or visually complex environments.

In order to address these challenges, iToF sensors have emerged as a promising alternative for real-time depth perception in UGV applications. iToF sensors measure depth by emitting modulated infrared light and the subsequent analysis of the phase shift of the reflected signal [10]. Compared to LiDAR and stereo vision, iToF technology offers several advantages, including the ability to operate in real-time, a compact form factor, cost-efficiency, and improved performance under low-light conditions. However, limitations such as multipath interference, range constraints, and reduced accuracy on highly reflective or absorptive surfaces affect sensor performance.

1.2 Scope of the Thesis

The aim of this thesis is to evaluate the feasibility of iToF sensors for real-time depth perception in UGVs operating in unknown environments. By utilizing an iToF sensor as the sole sensor for SLAM, this research aims to exploit its ability to provide direct depth measurements while maintaining a compact and cost-effective design. Additionally, this work seeks to combine the advantages of different SLAM approaches by processing the iToF data ready to be used in an existing, reliable, and efficient SLAM framework.

1.3 Outline

The structure of the thesis is as follows: Chapter 2 provides an overview of the state-of-the-art technologies relevant to this research, including depth sensing and SLAM. This chapter establishes the foundation by discussing the strengths and limitations of these technologies in the context of UGV navigation.

In Chapter 3 the system setup is introduced, detailing the hardware components and sensor integration, while Chapter 4 focuses on the software implementation and data processing pipeline. This chapter describes the methods used to process iToF sensor data.

Chapter 5 presents the methodology for system testing, outlining the experimental procedures used to evaluate the proposed approach. This chapter also discusses the results, analyzing the system's accuracy, reliability, and performance under different environmental conditions.

The conclusion of the thesis is provided in the final chapter, which summarizes the key findings, discusses their implications, and provides an outlook on potential future research directions to further improve the use of iToF sensors in SLAM for UGV applications.

CHAPTER 2

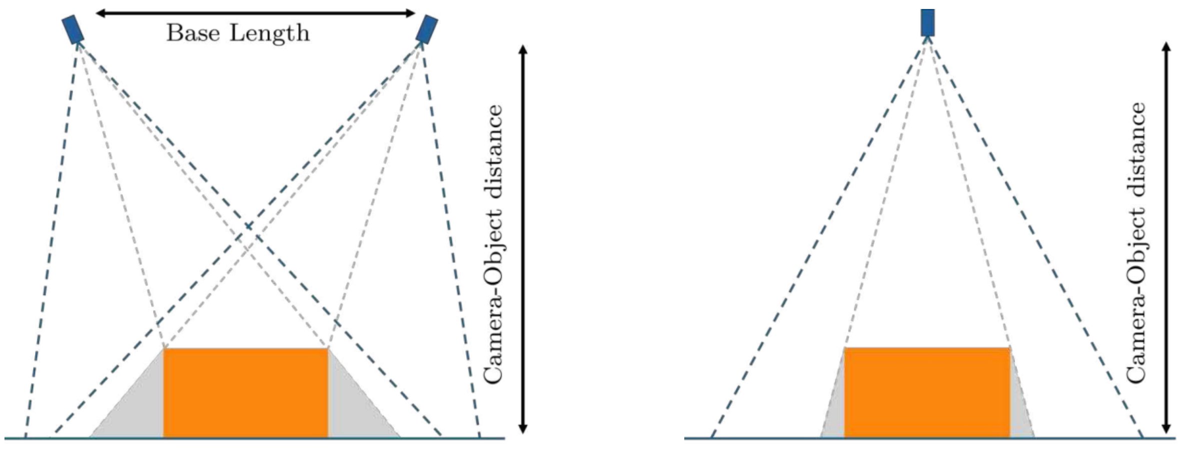
State of the Art

This chapter provides insights into the state-of-the-art technology for navigation applications on mobile robotic platforms. First, depth camera technologies are presented, with a focus on Time-of-Flight cameras. Next, Simultaneous Localization and Mapping is introduced as a key navigation method for autonomous robots. Afterwards, the Robot Operating System is discussed as a framework for integrating these technologies. Finally, the research questions of this thesis are introduced.

2.1 Depth Cameras

Depth cameras, in contrast to color cameras, are sensors for measuring the depth of a scene and provide depth images instead of color images [11]. These cameras are also referred to as 3D cameras, as a 3D point cloud can be calculated from the depth image. There are two main principles for depth calculation. One, known as the geometric approach or triangulation, uses the geometric relations of the sensor and the perceived image to calculate the depth. The other estimates the depth directly using the Time-of-Flight (ToF) principle.

The geometric approach, shown in Figure 2.1a, is achieved by stereoscopy, using two cameras, or with structured-light cameras, using one camera and a structured light projector [11]. Stereo cameras are highly dependent on the appearance of the scene. Visual features are needed to perform triangulation, as distinct points are used to match two images and estimate the depth [12]. Even in scenes with visual features, correspondence problems can occur. Structured light technology reduces these



(a) Triangulation (b) Direct depth estimation

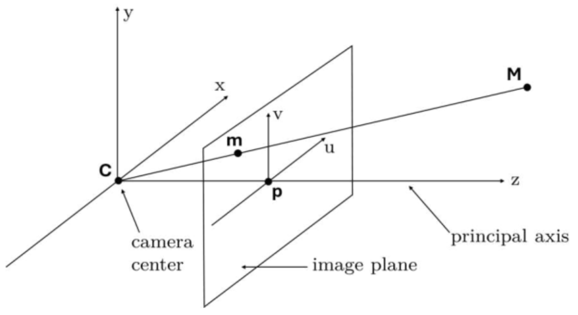
Figure 2.1: Comparison between 3D camera measurement principles from a top view. The 3D shapes of interest are represented as orange boxes, and cameras are represented as blue boxes. In triangulation, both boxes are cameras (stereo) or one is a camera and the other a projector (structured-light). The cameras' field of view is shown with dashed blue lines, and the gray area marks partial shadows [11].

problems by projecting a coded and structured pattern onto the scene, providing an usable reflection for triangulation. Patterns are optimized for various factors, such as scene dynamics, surface texture, and lighting conditions. The selected pattern type plays a crucial role in mitigating correspondence problems and ensuring accurate depth reconstruction. Furthermore, as triangulation requires that the scene has to be viewed from two perspectives, the base length - the relative positions of the two devices - must be known to calculate the scene's depth. Thus, for geometric depth calculation an extrinsic calibration is inevitable. Figure 2.1b shows direct depth estimation arrangement, where only one device and therefore no external information is needed. Consequently, direct depth measurement can be described using simple camera models, as discussed in the following section.

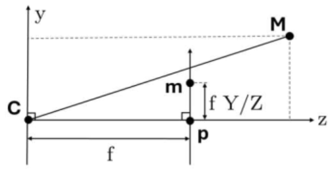
2.1.1 Camera Model

A fundamental camera model is the pinhole model [11]. As shown in Figure 2.2a it is an idealized representation of a camera to showcase the projection of a 3D scene point $\mathbf{M}$ to a 2D image point $\mathbf{m}$. It uses a global coordinate system, x, y, and z, where the image plane is positioned on the z-axis parallel to the global plane (x,y) and is described with (u,v) coordinates. The pinhole model assumes that light rays pass through a single point, the projection origin C, as they travel towards the camera. This perspective projection is displayed in Figure 2.2b and results in

$$\mathbf{M} = \{X, Y, Z\}^T \rightarrow \mathbf{m} = \{u, v\}^T = \left\{ f\frac{X}{Z}, f\frac{Y}{Z} \right\}^T, \qquad (2.1)$$



(a) frontal perspective



(b) trigonometrical representation

Figure 2.2: Pinhole camera model. The frontal perspective illustrates the geometric relationshop between a 3D point **M**, the camera center **C**, and the projected point **m** on the image plane in (u, v) coordinates. The principal axis of the camera aligns with the s-axis. The trigonometric representation provides a side view of the projection process, emphasizing how the image coordinates are computed. The point **M** is projected onto the image plane at **m**, following the principle of similar triangles. The focal length f and the depth Z determine the scaling factor, leading to the projection equations $u = f\frac{X}{Z}$ and $v = f\frac{Y}{Z}$ [11].

where f represents the focal length.

Although the basic pinhole model describes the main principles of a camera, real camera characteristics are not mapped. These need to be obtained by intrinsic camera calibration, extending the pinhole model [11]. The parameters included in intrinsic calibration are the focal length in both directions (f_x, f_y), a principal point (c_x, c_y), and the so-called skew parameter s. The calibration parameters are combined in the calibration matrix

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.2}$$

In comparison to Equation (2.1) the focal length f is now split in order to deal with non-square sensor pixels. Different horizontal and vertical unit vectors, described using m_x and m_y for the number of pixels per length unit along the x- and y-axis, lead to two different focal lengths f_x and f_y

$$f_x = f \cdot m_x, \quad f_y = f \cdot m_y. \tag{2.3}$$

The principal point (c_x, c_y) translates the optical center C to a corner of the image plane. The skewness parameter $s = f\cos(\alpha)$ introduces a correction factor. It can be used if the pixels do not have a rectangular shape with perpendicular sides, where α is the angle between two sides of the pixel. Usually, the correction factor is assumed to be zero $(\alpha = 90°)$.

Moreover, extrinsic camera calibration can be incorporated into the camera model. The parameters define the camera's position to the global coordinate system. Therefore, it includes a rotation matrix $\mathbf{R}$ and a translation vector t. Combining the camera parameters with the projection equation results in

$$\mathbf{m} = \mathbf{K} \cdot [\mathbf{R}|\mathrm{t}] \cdot \mathbf{M}. \tag{2.4}$$

In addition, the camera model can be extended to take lens distortion into account. Since lens imperfections lead to distortion errors, the camera deviates from the model in Equation (2.4) [12]. The imaging system actually measures the distorted coordinates $\hat{\mathbf{m}} = \{\hat{u}, \hat{v}\}$. To find the correct pixels u and v, the anti-distortion models are needed. Hereby, the Heikkila model is widely used and was tested to adequately correct the distortions of most imaging systems [13]. The equation for distortion compensation is

$$\mathbf{m} = \Psi^{-1}(\hat{\mathbf{m}}) = \begin{bmatrix} \hat{u}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + 2d_1\hat{u}\hat{v} + d_2(r^2 + 2\hat{u}^2) \\ \hat{v}(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) + d_1(r^2 + 2\hat{v}^2) + 2d_2\hat{u}\hat{v} \end{bmatrix}, \tag{2.5}$$

where Ψ is the Heikkila model, k_1, k_2, k_3 are distortion parameters for radial distortion, d_1 and d_2 are distortion parameters for tangential distortion, and

$$r = \sqrt{(\hat{u} - c_x)^2 + (\hat{v} - c_y)^2}. \tag{2.6}$$

Whereas these particular camera models demonstrate the formal working principles of 2D projection, in the next section, the necessary components for a ToF camera will be discussed, along with the methodology of depth calculation and the characteristics of this type of sensor.

2.1.2 Time-of-Flight Cameras

For ToF sensors, two measurement principles must be distinguished. Direct ToF technology is the method used in applications such as LiDAR and radar [14]. The depth image is calculated using the stopwatch principle. The sensor measures the time between sending out a light pulse and receiving the reflection. Thus, direct ToF systems rely on precise time measurements. Indirect ToF cameras, also called 3Di cameras, measure the phase shift from the transmitted signal to the received signal. Despite the different measurement principles, all ToF systems share the same key components.

Components of Time-of-Flight Systems

ToF systems are built from two main components, a transmitter (or illuminator) TX and a receiver (or sensor) RX, as shown in Figure 2.3 [12]. The transmitter emits a light signal $s_E(t)$ to a scene at a known time (e.g. $t = 0$). Subsequently, the receiver detects the back-reflection $s_R(t)$ at time τ. When the signal is detected at the receiver, it has traveled to the scene and back, covering a distance of $2d$, where d is the distance

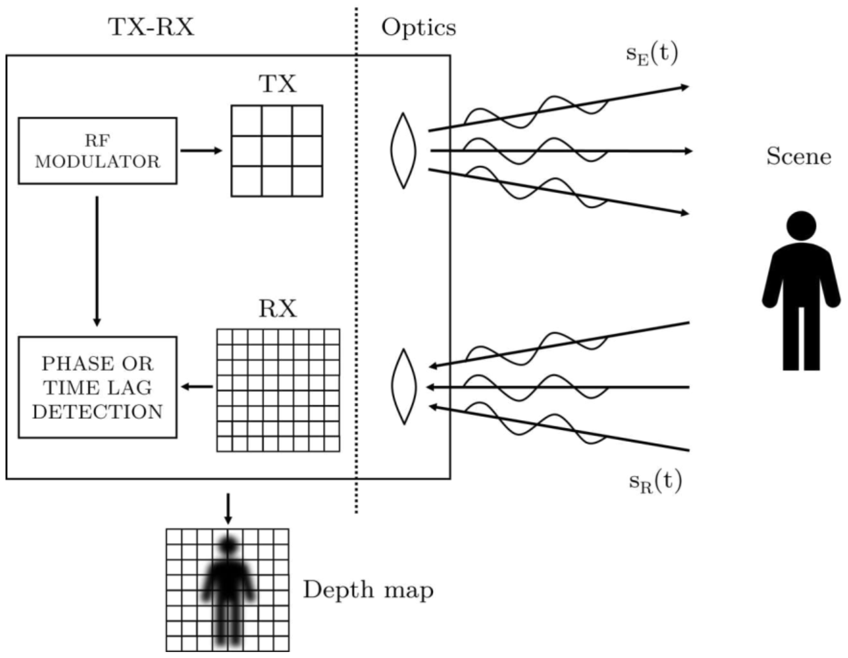


Figure 2.3: Schematic representation of a depth sensing system based on modulated signals. The transmitter (TX) emits modulated signals toward the scene. The receiver (RX) captures the reflection through an optical system. The phase or time lag detection module processes the received signals to compute depth information, generating a depth map that represents the scene's three-dimensional structure [12].

between the camera and the scene. The relationship between time τ and distance d is described with Equation (2.7), where c is the speed of light. This relationship defines the basic principle of direct ToF sensors. Considering Equation (2.7), it can be seen that the depth resolution is limited by the ability to measure small times [15]. For example, a distance of 30 centimeters is traveled out and back in 1 ns. Additionally, the maximum measurable distance is dependent on the intensity of the light source and the characteristics of the sensor.

$$d = \frac{c \cdot \tau}{2} \tag{2.7}$$

The basic functions and structure of the transmitter and the receiver are determined by the modulation method utilized [12]. Most current commercial solutions use homodyne amplitude modulation (AM) with a sinusoidal or square continuous wave (CW) signal. This signal can be effectively implemented with current complementary metal-oxide-semiconductor (CMOS) technology. Moreover, it uses a single modulation frequency f_m and does not require a large bandwidth. A drawback is that the homodyne AM is not robust against multipath and other propagation artifacts. These disturbance sources also appear with other often used signal modulations, such as pseudo-noise modulation and pulse modulation. Furthermore, CW modulation itself could be heterodyne AM or frequency modulation (FM) with chirp signals, instead of homodyne AM.

These modulations would face drawbacks of homodyne AM, but are not yet ready to be implemented in matrix sensor electronics.

The light source at the transmitter is a laser or a light-emitting diode (LED), as they are inexpensive and can be easily modulated by signals within the high-frequency and low very high-frequency bands [12]. The scene is illuminated by a 2D wavefront that is modeled in Equation (2.8) [16]. $s_E(t)$ denotes the emitter near infrared (NIR) signal, f is the NIR frequency, φ is the phase, and $m_E(t)$ the modulating signal. The modulating signal $m_E(t)$ describes an AM of sinusoidal or square wave type.

$$s_E(t) = m_E(t)\sin(2\pi f t + \varphi) \tag{2.8}$$

As shown in Figure 2.3 the signal is diffused to the scene by suitable optics and subsequently the echoed reflection is collected and imaged in the receiver [12]. Here, an optical band-pass filter with center-band is included, tuned to the NIR carrier frequency of the transmitter to increase the signal-to-noise ratio.

The receiver is a 2D sensor array of individual pixels. The receiver design has evolved over the last decades aiming for higher accuracy, and faster and longer range distance measurements [15]. Starting with CMOS and charge-coupled device (CCD) imagers, similar to conventional 2D imagers, receivers are now built using single-photon avalanche diode (SPAD) in CMOS integrated circuits. The ability to detect single photons makes this system suitable for direct and indirect ToF systems. The signal received at each pixel can be described by Equation (2.9), where $m_R(t)$ represents the transformation of the modulating signal $m_E(t)$ as it reaches the receiver, and $n_R(t)$ is the background wideband light noise at the receiver input [12].

$$s_R(t) = m_R \sin(2\pi f t + \varphi') + n_R(t) \tag{2.9}$$

This $N_R \times N_C$ receiver matrix of ToF depth cameras is equivalent to the image plane in the previously discussed camera model (see 2.1.1). It contains the components measuring the relevant values required to calculate the scene depth. As the calculation for direct ToF sensors is fully explained with Equation (2.7), the following chapter focuses on depth calculation for indirect ToF sensors.

Depth Calculation

Indirect ToF faces the problem of measuring small time units that occur in direct ToF technologies, and instead calculates depth via the phase shift [17]. In order to do so, the scene is illuminated with an amplitude modulated CW signal.

A common approach for iToF depth calculation is four-bucket sampling, as shown in Figure 2.4 [18]. The sensor takes four intensity measurements, each sample phase-

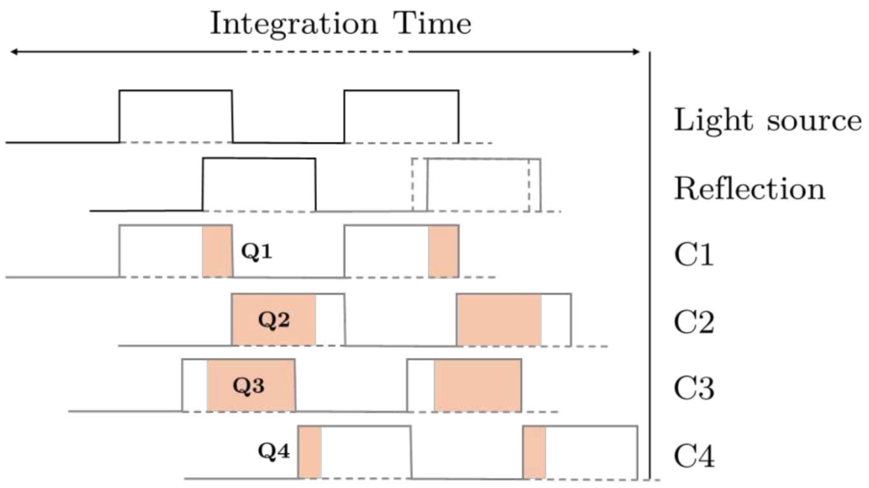


Figure 2.4: Four-bucket sampling for indirect Time-of-Flight depth calculation. The light source emits a modulated signal. The reflected signal is sampled across four channels, C1-C4, at four distinct phase offsets during the integration times. Each channel measures the intensities at its offset, represented by the shaded regions (Q1, Q2, Q3, Q4). The measured intensities are used to determine the phase shift of the reflected signal [18].

shifted by 90°. The phase angle φ is calculated from the intensity measurements Q_i with $i \in \{1 \ldots 4\}$ using

$$\varphi = \arctan\left(\frac{Q_3 - Q_4}{Q_1 - Q_2}\right). \tag{2.10}$$

The distance is then determined by inserting the phase φ into

$$d = \frac{c}{2f} \cdot \frac{\varphi}{2\pi} = \frac{c\,\varphi}{4\pi f}, \tag{2.11}$$

where c denotes the speed of light and f the used modulation frequency. It is evident from Equation (2.10), that this method eliminates any constant offsets by comparing two pairs of measurements, $(Q_1 - Q_2)$ and $(Q_3 - Q_4)$. Additionally, the quotient of these serves to mitigate the impact of any constant gains from the distance measurements, for example circuit amplification and attenuation, or the reflected intensity.

With these four measurements, additional characteristics can be calculated, such as the amplitude A and offset B, shown in Equation (2.12) and Equation (2.13) respectively [19]. The amplitude A functions as a direct indicator of the depth resolution achieved. The offset B facilitates the identification of saturation, which has the potential to introduce distortion in depth measurements.

$$A = \frac{\sqrt{(Q_1 - Q_2)^2 + (Q_3 - Q_4)^2}}{2} \tag{2.12}$$

$$B = \frac{Q_1 + Q_2 + Q_3 + Q_4}{4} \tag{2.13}$$

Whereas in direct ToF systems the measurable range depends on the laser strength, in indirect ToF cameras the maximal distance is defined via the modulation frequency used for the light source [18]. Since the modulation frequency is periodic with 2π, the wavelength defines the maximal measurable distance

$$d_{\max} = \frac{c}{2f}, \tag{2.14}$$

also often referred to as ambiguity distance d_{amb}. Obstacles in the scene at greater distances than d_{f_m} can be detected, assuming that the light source possesses sufficient intensity, but the measured distances are subject to a wrapping-error. The sensor is only capable of indicating position within a single wavelength. It cannot detect how often the signal was repeated from sending to receiving. The true distance is calculated using

$$d_{gt} = d + k\ d_{\max} \quad k \in \mathbb{N}, \tag{2.15}$$

where k points out the phase wrapping.

As indicated by Equation (2.14), a lower modulation frequency allows a wider unambiguous measurement range [11]. However, this increased range is accompanied by an increase in depth uncertainty, as shown in Figure 2.5. The precision of the depth measurement is directly proportional to the uncertainty in phase estimation. Assuming a normal distribution for phase estimation uncertainty, the resulting distance error will also follow a normal distribution, as delineated in Equation (2.11).

An alternative approach to enhance the detection range without reducing the modulation frequency is the multi-frequency technique [20]. This method utilizes two or more frequencies with distinct ambiguity distances $d_{f_x,\max}$, allowing the true position of an object to be determined at the point where these frequencies converge. This concept is illustrated in Figure 2.6, where the true location is found in the third wrap of both modulation frequencies. Adapting Equation (2.15) to the dual-frequency method results to

$$\begin{aligned} d_{gt} &= d_1 + k_1 \cdot d_{f_1,\max} \\ &= d_2 + k_2 \cdot d_{f_2,\max}, \end{aligned} \tag{2.16}$$

where d_{gt} describes the ground truth distance. Indices 1 and 2 describe the two modulation frequencies used, with d as the distance measured by the sensor for one frequency and d_{max} as the belonging ambiguity distance. The factor k outlines the phase wrapping.

To determine the true distance between the sensor and the scene in a dual-frequency

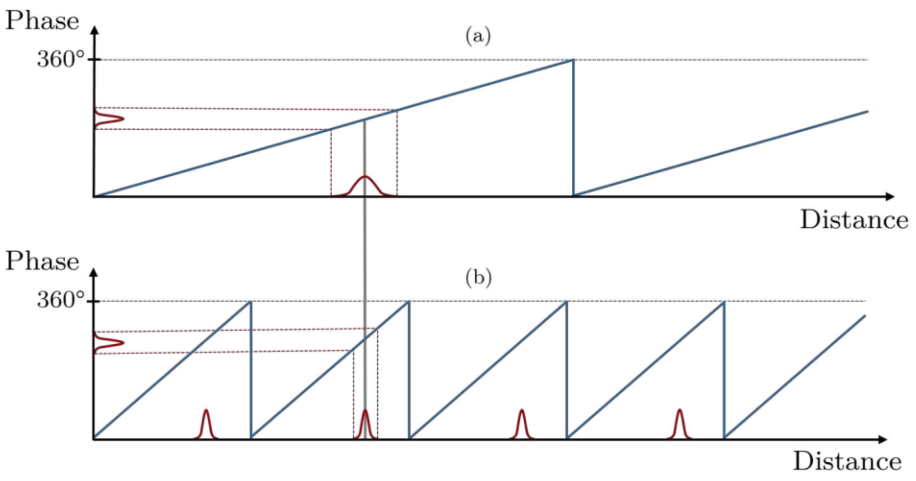


Figure 2.5: Comparison between modulation frequencies on depth measurements in indirect Time-of-Flight cameras. (a) shows depth measurement with a low modulation frequency, where no ambiguity appears, but the depth uncertainty is higher. (b) displays a higher modulation frequency, where the phase wraps multiple times, causing depth ambiguity, but the depth uncertainty is smaller [11].

modulation system,

$$0 = \frac{f_2}{f_{\text{dual}}} \cdot (k_1 + \varphi_1) - \frac{f_1}{f_{\text{dual}}} \cdot (k_2 + \varphi_2) \tag{2.17}$$

must be solved [21]. The unknown integers k_1 and k_2 are resolved by minimizing the function, where φ_1 and φ_2 are normalized phase values, $\varphi_1, \varphi_2 \in [0, 1)$ and $k_1, k_2 \in \mathbf{N}$ with $k_2 \leq k_1$ and $f_2 < f_1$. f_E is defined via the greatest common divisor (gcd) from f_1 and f_2

$$f_{\text{dual}} = \gcd(f_1, f_2), \tag{2.18}$$

the maximal unambiguously measurable distance of the dual-frequency system can be calculated using (2.14) again [22]. Once k_1 and k_2 are found, the distance of the scene can be calculated using one of the Equation (2.16) [20].

In addition to the issues related to maximum distance and phase wrapping, there are other system and environmental characteristics associated with ToF sensors.

System Characteristics and Error Sources

Indirect ToF cameras offer numerous advantages, including the capacity to capture dense depth and intensity images at high frame rates, a compact and lightweight design, and auto-illumination, thus eliminating the need for external light sources [17]. However, these systems are also subject to inherent limitations. The accuracy of the measurements is constrained by the power of the emitted infrared signal, and the amplitude of the

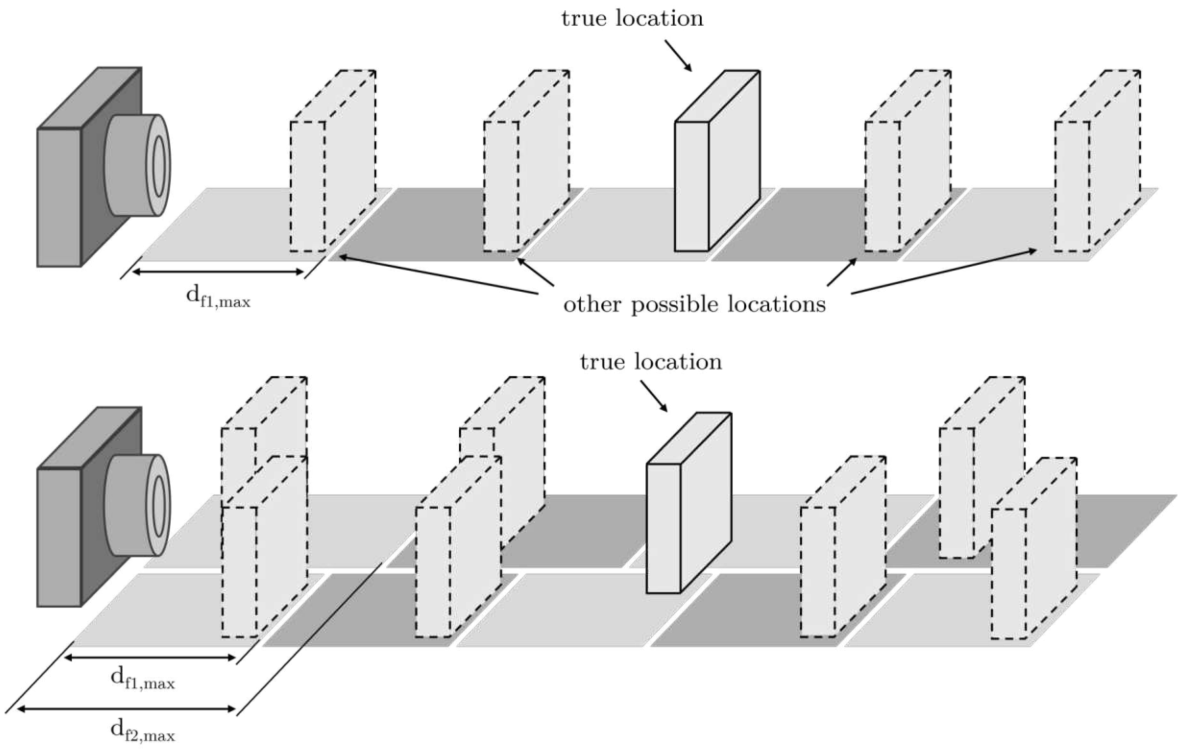


Figure 2.6: Comparison of single-frequency and dual-frequency modulation in indirect Time-of-Flight cameras. The top illustration shows depth ambiguity in single-frequency modulation, where multiple possible locations exist due to phase wrapping. The bottom illustration demonstrates dual-frequency modulation, which combines two different modulation wavelengths d_{amb1} and d_{amb2} to resolve ambiguity and correctly identify the true location [11].

reflected signal varies depending on the material and color of the object surface [23]. Furthermore, depth accuracy and frame rate are constrained by the required integration time, where longer integration times enhance accuracy but limiting the capability to capture moving objects at fixed frame rates.

These limitations lead to various errors that affect the accuracy and reliability of depth measurements [17]. These errors in iToF cameras are categorized into two classes: systematic and non-systematic. Systematic errors are intrinsic to the sensor and can be mitigated through calibration. This includes errors such as depth distortion (wiggle error), integration-time-related errors, receiver-related errors, and temperature-related errors. Non-systematic errors are strongly related to scene content and reduction techniques mainly rely on spatial and temporal averaging [16]. Such errors occur through ambient light, which cannot be completely removed from the signal, multi-camera interference, or multi-path mitigation [10].

Despite the implementation of calibration and filtering, not all errors can be completely mitigated, making it essential to evaluate the overall performance of the sensor.

2.1.3 Sensor Analysis

The accuracy of a sensor is analyzed by measuring its trueness and precision [24]. The trueness of a sensor is defined as the degree to which the arithmetic mean of a large number of test results matches the true or accepted reference value. Precision signifies the level of agreement between the test results themselves.

According to the foregoing definition, trueness μ is calculated using

$$\mu = |d_0 - \mu_d|, \tag{2.19}$$

where d_0 is the true distance. The average depth obtained through the test measurements is

$$\mu_d = \frac{1}{N \cdot n^2} \sum_{i=1}^{N} \sum_{u,v}^{n} I_i(u,v), \tag{2.20}$$

with the number of measurements denoted as N, the size of the measured region as n, and (u, v) as the corresponding coordinate in the 2D depth image I_i.

Precision is declared using

$$\text{precision} = \sqrt{\left(\frac{1}{N \cdot n^2} \sum_{i=1}^{N} \sum_{u,v}^{n} \tilde{I}_i(u,v)^2\right) - \tilde{\mu}_d^2}, \tag{2.21}$$

where $\tilde{I}$ is the corrected fronto-parallel depth image and $\tilde{\mu}_d$ is the average depth of $\tilde{I}$ calculated using Equation (2.20).

The characteristics of ToF sensors discussed in this section allow depth cameras to perform tasks that extend beyond the mere acquisition of spatial data. Thus, ToF sensors enable systems to perceive, interpret, and navigate their surroundings. The subsequent section will therefore focus on SLAM, a state-of-the-art navigation technologies in which these sensors are used.

2.2 Simultaneous Localization and Mapping (SLAM)

The ability to navigate unknown environments represents a significant challenge in the field of mobile robotics [25]. In known environments, where an accurate map is available, navigation is a relatively straightforward process. However, in unknown environments and without external reference systems, such as GPS, map acquisition becomes particularly difficult. Therefore, algorithms that process data from on-board sensors to a map while simultaneously determining the robots position within it are required. This specific problem is called Simultaneous Localization and Mapping problem.

The fundamental concept of SLAM systems is shown in Figure 2.7 [26, 27]. The system is fed with sensor data and returns an estimated robot pose and a map. The implementation of such a system is divided into two distinct algorithms, a front-end algorithm and a back-end algorithm. The front-end is responsible for converting the sensor data into a format that can be utilized by the following back-end algorithm, which solves the original SLAM problem.

In this subchapter, the initial focus is on the SLAM problem and its associated solutions. This is followed by an examination of the processes involved in the preparation of the sensor data. Subsequently, possibilities to display the resulting map are discussed. Finally, common ready-to-use SLAM algorithms are presented.

2.2.1 Back-End Approaches

As illustrated in Figure 2.7 in the back-end section, there are three common algorithms to solve SLAM. Although they can be divided into two different approaches, namely *smoothing* and *online* (also called filtering), they all are based on the same problem.

Probabilistic Formulation of SLAM

A SLAM system uses data from different sensors to estimate position and build a map [25]. Due to sensor measurement noise, SLAM is described in a probabilistic method

$$p(\mathbf{x}_{1:T}, \mathbf{m} \mid \mathbf{z}_{1:T}, \mathbf{u}_{1:T}, \mathbf{x}_0\}, \tag{2.22}$$

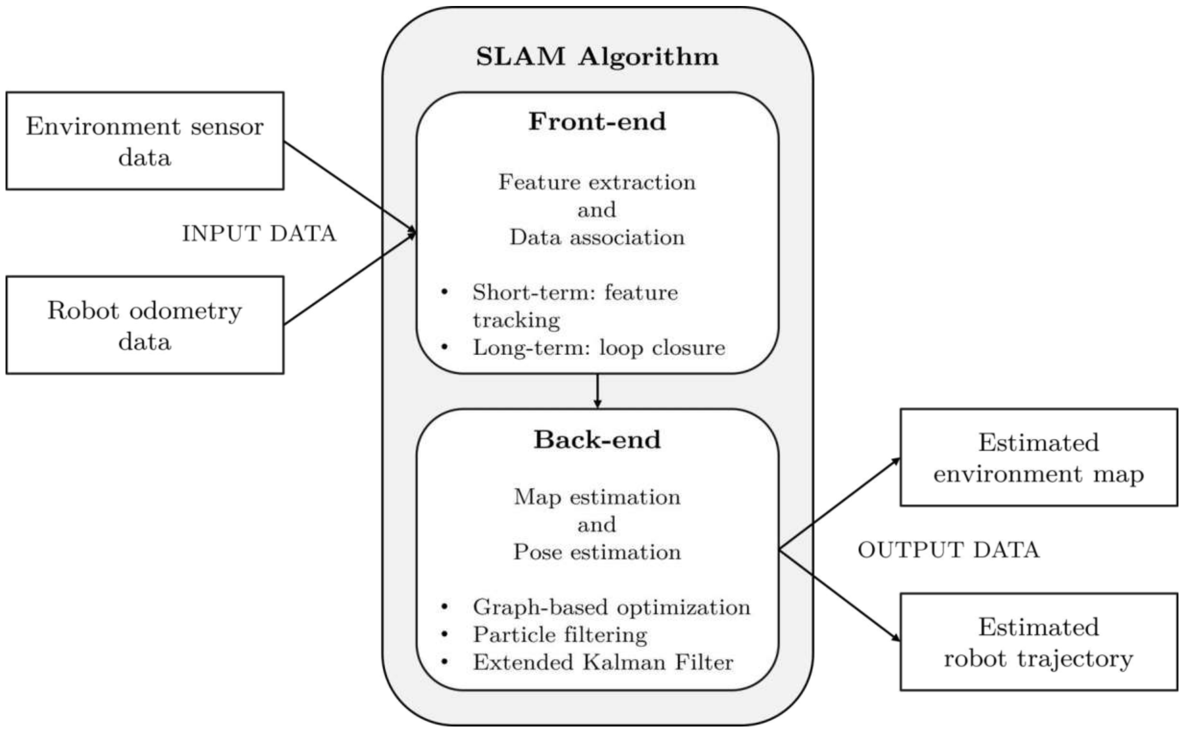

SLAM Algorithm
Environment sensor data
INPUT DATA
Robot odometry data
Front-end
Feature extraction and Data association
• Short-term: feature tracking
• Long-term: loop closure
Back-end
Map estimation and Pose estimation
• Graph-based optimization
• Particle filtering
• Extended Kalman Filter
Estimated environment map
OUTPUT DATA
Estimated robot trajectory

Figure 2.7: Overview of the SLAM algorithm, illustrating its front-end and back-end processing stages. The front-end handles feature extraction and data association, including short-term feature tracking and long-term loop closure. The back-end is responsible for map and pose estimation using techniques such as graph-based optimization, particle filtering and the Extended Kalman Filter, ultimately producing an estimated environment map and robot trajectory [26, 27].

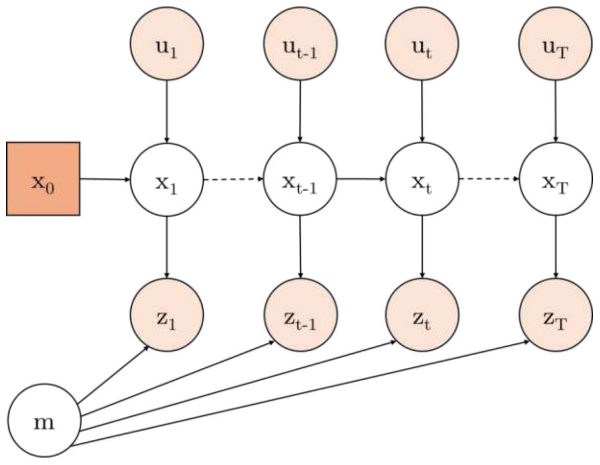


Figure 2.8: Dynamic Bayesian Network of the SLAM process. Observed variables are highlighted in orange, while white nodes represent the estimated robot trajectory $\mathbf{x}_{1:T}$ and the environment map $\mathbf{m}$. The recursive pattern is governed by the state transition model, shown by arrows leading to $\mathbf{x}_t$, which represents the probability of transitioning from $\mathbf{x}_{t-1}$ to $\mathbf{x}_t$ based on the odometry $\mathbf{u}_t$ and the observation $\mathbf{z}_t$ [25].

where $\mathbf{p}$ describes the probability of a certain trajectory of the robot. Since the robot moves in an unknown environment, the trajectory is given as a sequence of $\mathbf{x}_{1:T} = \{\mathbf{x}_1, ..., \mathbf{x}_T\}$, the hereby acquired odometry measurements are named $\mathbf{u}_{1:T} = \{\mathbf{u}_1, ..., \mathbf{u}_T\}$, and the perceptions from the environment are declared as $\mathbf{z}_{1:T} = \{\mathbf{z}_1, ..., \mathbf{z}_T\}$. $\mathbf{x}_0$ describes the initial position of the map and can be chosen arbitrarily.

Estimating the probability $\mathbf{p}$ in Equation (2.22) involves high-dimensional state spaces that can lead to an intractable formulation [25]. Therefore, certain assumptions are needed. The first one is the static world assumption and the second one is the Markov assumption, which assumes that the following step of a process is entirely dependent on the current state. Considering those, the problem can be described with the structure of a dynamic Bayesian network (DBN), as displayed in Figure 2.8. The observed variables are colored orange, while white nodes show the estimated trajectory of the robot $\mathbf{x}_{1:T}$ and the map of the environment $\mathbf{m}$. The recursive pattern of the DBN is characterized by the state transition model and the observation model. The state transition model, see Equation (2.23), shown by arrows leading to $\mathbf{x}_t$ represents the probability of the robot moving from $\mathbf{x}_{t_1}$ to $\mathbf{x}_t$ given the odometry information $\mathbf{u}_t$.

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \tag{2.23}$$

The observation model, see Equation (2.24), shown by arrows leading to $\mathbf{z}_t$, represents the probability of an observation $\mathbf{z}_t$ given the robot's position $\mathbf{x}_t$ and the map $\mathbf{m}$.

$$p(\mathbf{z}_t|\mathbf{x}_t, \mathbf{m}) \tag{2.24}$$

Since the DBN representation of SLAM captures the system's temporal dependencies, it inherently admits sequential estimation methods. Such methods are called filter-based algorithms, they are complemented by another main method called smoothing or graph-based algorithms.

Filter-Based Approach

Filter-based approaches, of which the Extended Kalman Filter (EKF) SLAM and Particle Filter-based SLAM are examples, maintain a probabilistic belief about the robot's state and update it recursively as new observations are made [28]. Earlier approaches aimed to improve the performance of EKF-SLAM while preserving its core linear Gaussian assumptions. In contrast, FastSLAM introduced a novel approach based on particle filtering, also common as recursive Monte Carlo sampling, condensation, or survival of the fittest [29]. Thus, FastSLAM became the first method to explicitly represent both the nonlinear process model and the non-Gaussian pose distribution [28]. Although the observation model is still linearized, this simplification is generally a reasonable approximation for range-bearing measurements when the robot's pose is known. However, the efficacy of these methods is optimized primarily for small-scale environments. Mapping in large-scale environments encounters challenges due to the substantial increase in computational complexity that accompanies the number of landmarks and observations. This behavior not only results in scalability issues, but also in reduced accuracy in long-term operation.

Graph-Based Approach

Graph-based approach, in contrast, constructs a graph representation of the environment [25]. Within this graph, each node represents a robot's position along with the measurement taken at that location. An edge between two nodes defines a probability distribution over the spatial constraint between the corresponding robot poses. As the observation model in Equation (2.24) suggests, one observation z_t can lead to multiple edges, each edge denoting a distinct potential connection between different poses. It is evident that the consideration of all possible models in the estimation process would result in substantial combinatorial complexity. To maintain computability, the most practical approaches of graph-based SLAM restrict the model to the most likely graph topology. Once the graph has been constructed, the next step is to resolve the error minimization problem using techniques such as nonlinear least squares, and thereby minimizing the overall error in the graph. This approach requires significant computational resources to handle optimization processes, but allows efficient corrections and making it more robust in large-scale environments.

While the back-end operates on abstract representation of the data solving the optimization problem, the front-end's task is to select the most probable constraint from observation.

2.2.2 Front-End Approaches

As Figure 2.7 displays, the front-end tasks involve feature extraction from the incoming data and data association from new data with existing data [27]. Data association is split into short-term and long-term association. Short-term association includes feature tracking and comparing the current frame to the last few frames, while long-term association is about loop closure algorithms. The front-end of the algorithm directly deals with the sensor input data. Therefore, it is heavily dependent on the type of sensors used. To consider the characteristics with different sensor types, visual-based SLAM and LiDAR-based SLAM is distinguished.

Visual-based SLAM

Visual SLAM is an active research area that focuses on SLAM solutions that use different camera technologies, such as RGB cameras, RGB-D sensors [30]. There are two common techniques: direct SLAM and feature-based SLAM. Direct SLAM estimates pixel movement and camera pose at the same time [31]. Therefore, images are aligned and the photometric error is minimized. With this approach, raw image data is processed directly, while feature-based SLAM relies on detecting and tracking key features across frames first. The position is estimated afterward by minimizing the geometric error between the features. Both principles are commonly used with RGB and RGB-D cameras. These approaches have gained popularity due to their cost effectiveness, ability to collect large amounts of information, and wide measurement range [30]. Nonetheless, state-of-the-art visual SLAM algorithms face challenges, particularly in real-time applications [31]. Theoretically, the frame rate of the camera limits the processing speed, but in practice, issues often arise from frame losses caused by peaks in processing time. Even when an algorithm operates at the camera frame rate, those peaks suggest that information is lost in real-time forced conditions, where the most recently received image is processed at each time.

LiDAR-based SLAM

LiDAR-based SLAM has become a widely adopted mapping technology due to its simplicity and accuracy [32]. Compared to visual-based SLAM, LiDAR-based SLAM achieves low-drift motion estimation while maintaining an acceptable level of computational complexity. Tracking is done using scan matching methods, where successive LiDAR scans are aligned to estimate motion. A prevalent technique used for this purpose is the Iterative Closest Point (ICP) algorithm.

The ICP algorithm is used to align newly acquired LiDAR scans with previously mapped LiDAR data [30]. The method is shown on two consecutive scans, scan i and scan $i + 1$, as shown in Figure 2.9a and Figure 2.9b. First, a guess about the initial transformation is made, followed by a nearest-neighbors search, see Figure 2.9c.

Table 2.1: Comparison: Visual-based vs. LiDAR-based SLAM

Feature	**Visual-based SLAM**	**LiDAR-based SLAM**
Sensor Cost	Low	High
Accuracy	Moderate	High
Illumination Sensitivity	High	Low
Robustness in Textureless Areas	Low	High

The algorithm then refines the transformation to minimize the distances between the corresponding points. This leads to aligned scans at the end of the algorithm, as shown in Figure 2.9d. However, ICP has several notable limitations. Firstly, the algorithm requires a computationally expensive search to establish point correspondences between scans. Secondly, its accuracy and convergence are highly dependent on a good initial transformation guess. Due to these constraints, ICP alone is not sufficient for SLAM and therefore is combined with one of the previously described back-end algorithms.

Table 2.1 summarizes the key differences between visual-based SLAM and LiDAR-based SLAM, underscoring the strengths and limitations of each approach. The comparison includes factors such as sensor cost, accuracy, sensitivity to illumination changes, and performance in texture-less environments. The classification is based on the properties discussed above. Although visual-based SLAM is generally more cost-effective, it is highly dependent on lighting conditions and struggles in texture-less areas. In contrast, LiDAR-based SLAM offers greater accuracy and robustness in diverse environments, but comes at a higher sensor cost.

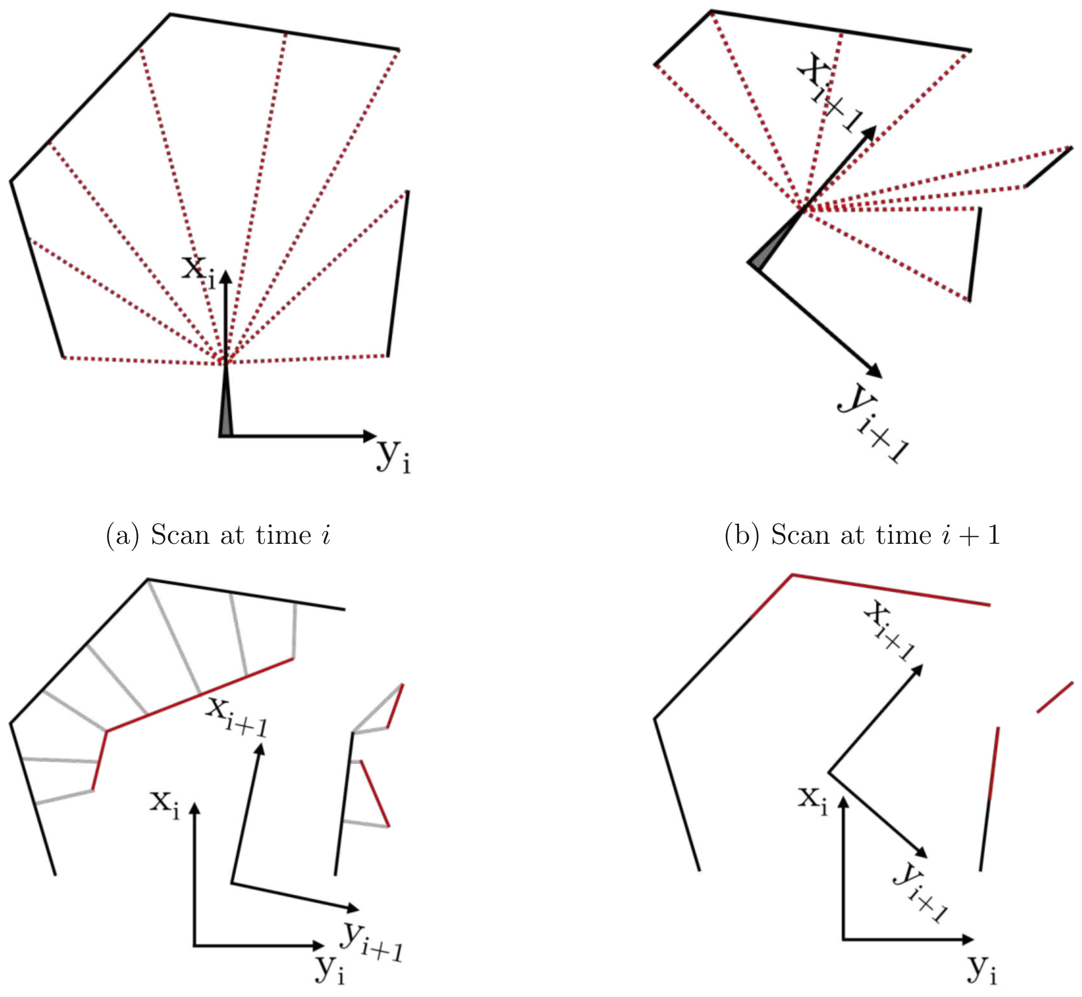
Following a comprehensive exposition of the SLAM algorithms, the subsequent stage is to examine the storage and structuring of the obtained spatial data.

2.2.3 Mapping Techniques

The output map can be structured in many different ways depending on the sensor data and the used algorithm. The most common map representation is a set of sparse 2D or 3D landmarks [26]. Here, the stored 2D or 3D landmarks correspond to distinctive features determined in the front-end algorithm. This approach, widely used in visual-based environments SLAM, has been observed to encounter difficulties in low-texture environments.

Dense map representation, on the other hand, captures unstructured point clouds. As this comes with significant storage demand, geometric primitives, such as planes or surfaces, can be extracted from point clouds to optimize performance. This makes it useful for obstacle avoidance, if sufficient computational power is provided.

Grid-based representation is a method of dividing space into 2D grids or 3D voxels, with each cell assigned a state of *free*, *occupied* or *unknown* based on probabilistic



(a) Scan at time i

(b) Scan at time $i + 1$

(c) First guess of transformation and nearest neighbors search

(d) Transformation after the last distance minimization

Figure 2.9: Principle of the Iterative Closest Point algorithm used for aligning two scans. (a) shows a scan taken at time i, followed by (b) with another scan at time $i + 1$. In (c), an initial transformation guess is applied and nearest neighbor correspondences between the scans are identified. The transformation is refined iteratively by distance minimization until convergence is reached. As a result, the scans are aligned in (d) [30].

models. This method, returning the so-called Occupancy Grids, has been shown to be effective for path planning and obstacle avoidance.

After exploring both front- and back-end SLAM algorithms, as well as different mapping representations, the final step involves examining SLAM solutions that integrate these components into practical and deployable systems.

2.2.4 Ready-to-use SLAM Algorithms

This section introduces common open-source implementations for SLAM with a special focus on SLAM Toolbox, as it is one of the most recently developed algorithms.

ORB-SLAM

Of these, ORB-SLAM is the only implementation that processes data from RGB and RGB-D cameras [33]. It provides accurate and real-time camera trajectory and spare 3D reconstructions of various environments. However, it is important to note its reliance on trained vocabulary for feature matching and loop closure detection.

Hector SLAM

Examining LiDAR-based SLAM algorithms, Hector SLAM is a fully optimized filter-based SLAM algorithm designed exclusively for LiDAR information. It functions utilizing solely LiDAR data and, due to its efficiency, is particularly well-suited for real-time applications. However, due to the absence of odometry information, it is entirely dependent on scan matching, which can result in accumulated drift over time and inaccurate long-term mapping.

Gmapping

Conversely, Gmapping, a 2D filter-based SLAM, is heavily reliant on odometry information and is capable of efficiently constructing grid-based maps. Unfortunately, due to the high impact of odometry, it tends to drift, reducing accuracy in dynamic environments.

Karto SLAM

In comparison to Gmapping, Karto SLAM, which also returns grid-based maps, is a graph-based method that uses the current pose and the complete map to process the current data. Consequently, the Karto SLAM system demonstrates enhanced map consistency in large environments. Nonetheless, the system's computational complexity is proportional to the number of landmarks, which can compromise its real-time performance in large-scale environments.

SLAM Toolbox

SLAM Toolbox, which is based on Karto SLAM, has been developed for the purpose of accurate and scalable mapping in large and dynamic environments [34]. It improves upon previous SLAM solutions by offering multiple mapping modes, real-time localization, multi-session mapping, and improved graph optimization.

SLAM Toolbox provides two principal mapping modes: synchronous and asynchronous [35]. In synchronous mode, all sensor measurements are processed, ensuring high accuracy but potentially causing delays in large environments if computational resources are limited. This mode is typically used in offline mapping for optimal map quality. Conversely, asynchronous mode functions on a best-effort basis, processing data as computational capacity allows. This mode is suited for online mapping, enabling real-time navigation on systems with limited processing power.

However, its reliance on 2D LiDAR data may restrict its applicability in highly unstructured 3D environments [34]. Furthermore, although it offers improved computational efficiency, handling extremely large-scale or highly dynamic environments with continuous map updates remains a challenge.

SLAM Toolbox is an integral component of the Robot Operation System (ROS), a framework designed for the development and management of robotic applications. The following section will provide a comprehensive exploration of ROS and its architectural framework.

2.3 Robot Operating System

Robot Operation System (ROS) is an open-source software that facilitates the development of robotic applications [36]. Initially developed as ROS, it has become a widely adopted software solution within the robotics community. Despite its success, ROS has several limitations, particularly in real-time applications and distributed multi-robot systems [37]. Furthermore, the absence of built-in security mechanisms leaves systems vulnerable to cyber threats. To overcome these limitations, Robot Operation System 2 (ROS2) was developed with the aim of real-time performance and enhanced security [36]. The next section outlines which changes in system architecture were necessary to make ROS2 more suitable for the latest system requirements.

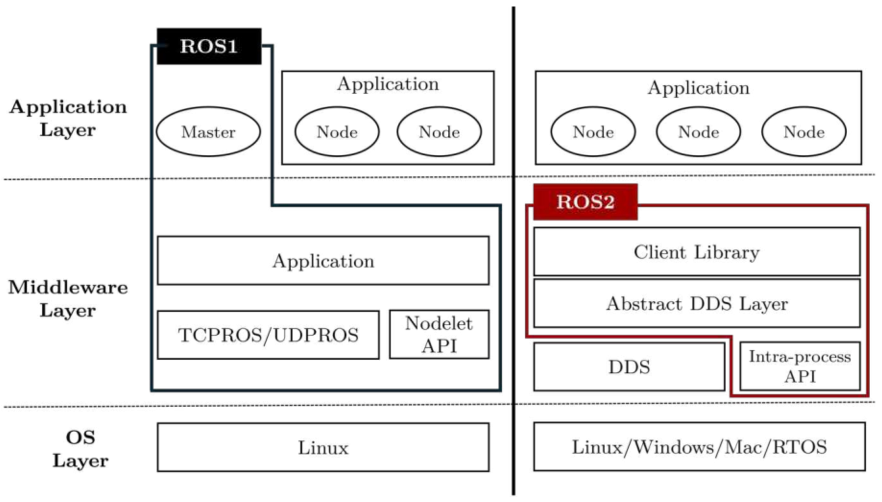


Figure 2.10: Comparison between ROS1 and ROS2 architecture. ROS1 relies on a master node for communication and uses TCPROS/UDPROS for transport. In contrast, ROS2 replaces the master with DDS, an abstract DDS layer, and an intra-process API for improved efficiency. ROS2 also supports multiple operating systems, including Linux, Windows, Mac, and RTOS, improving cross-platform compatibility [36].

2.3.1 System Architecture

Figure 2.10 displays the differences in the architecture of ROS and ROS2 [36]. As shown in the application layer, both ROS applications are composed of independent computing processes known as nodes. The nodes utilize a publisher-subscriber model, in which messages are exchanged through designated topics. Each topic serves as an identifier for the type of message that is transmitted. When a node publishes a message on a topic, any subscribing node receives and processes it. This publisher-subscriber model promotes modularity and is well-suited for distributed systems, since it enhances fault isolation, accelerates development, and supports code re-usability. Another method of communication between nodes is services [38]. While topics enable nodes to publish continuous updates and receive data streams, services use a call-and-response model. Thus, servers only provide data when specifically invoked by a client.

Although the applications have a similar structure in ROS and ROS2, there are major differences in the communication system. As shown in Figure 2.10 on the left, ROS has an additional node in the application layer, the so-called ROS master, which is a centralized process for managing communication. In addition, ROS utilities TCP and UDP (TCPROS/UDPROS) as a middleware for communication. The Subscriber and publisher nodes first interact with the master node, which manages the topics. Subsequently, subscribers establish direct connections with publishers. The ROS master is important for connecting the nodes, but is not involved in data transmission. This setup using a ROS master is a major drawback of the system, as the master is a single-point-of-failure in the system and limits scalability. This architectural constraint is inherent in the design of ROS. In contrast, ROS2 is built on top of Data Distribution

Table 2.2: Explanation of common QoS policies [39]

Policy	Description
`Deadline`	detects if a data is written or read within a set amount of time, if `period` $\neq \infty$
`Ownership`	specifies if there is one publisher to one topic (`exclusive`) or more than one (`shared`)
`Reliability`	controls whether samples can be dropped under specific circumstances (`best_effort`) or every value has to be delivered (`reliable`)
`Transport_Priority`	prioritizes data in the transport layer, if `value` > 0

System (DDS), eliminating the need for a master process, as illustrated on the right side of Figure 2.10. This separate DDS abstraction layer enables high-level configurations and has a significant impact on scalability and fault tolerance.

2.3.2 Data Distribution System

DDS is a mature middleware protocol [37]. It employs the Data-Centric Publish-Subscribe (DCPS) model, wherein a global data space contains DDS topics, analogous to ROS topics, see Figure 2.11. Processes that publish or subscribe to these topics are designated as participants, and their communication is managed through configurable Quality of Service (QoS) parameters [39]. These QoS parameters are a form of service contract and are used to define the needed reliability or timing requirements. Two parties can only communicate if they show compatible QoS policies. Hereby aspects of data transmission, such as reliability, durability, latency, and deadline constraints, are controlled. The most common QoS parameters are listed in Table 2.2.

Moreover, Figure 2.11 shows the interaction between ROS2 and the DDS via abstract DDS Application Programming Interfaces (APIs) [37]. The user-defined application logic is processed by the ROS2 Client Library (RCL) to establish node-based communication, which is then translated by the ROS2 DDS Middleware (RMW) into DDS structures and configurations. At runtime, ROS2 nodes map directly to DDS participants, converting node actions into DDS API calls for communication.

The improvements from ROS to ROS2, such as real-time capabilities and enhanced security, further establish ROS2 as a state-of-the-art technology for diverse robotic applications. With its extensive set of software libraries and development tools, ROS2 provides a robust framework for building and deploying modern robotic systems [40].

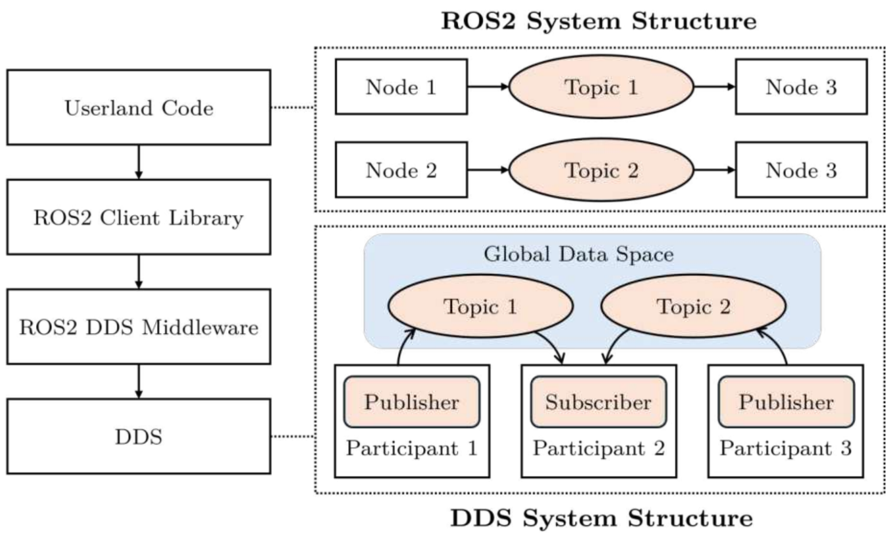


Figure 2.11: ROS2 DDS architecture with DCPS protocol. Nodes publish and subscribe to topics within ROS2 System Structure, while DDS manages data exchange through a global data space for efficient communication [37].

2.4 Research Questions

The state-of-the-art analysis indicates that Simultaneous Localization and Mapping (SLAM) is imperative for mobile robotics, as it facilitates autonomous navigation and interaction in uncharted environments. Nevertheless, attaining precise and dependable SLAM algorithms remains a challenge due to limitations in sensor technology, computational constraints, and the necessity for robustness in dynamic conditions. A critical aspect of SLAM research concerns the selection of sensor modality, with vision-based and LiDAR-based approaches being the most prevalent. While vision-based SLAM offers affordability and compact sensors, it struggles with lighting variations and scale estimation. Conversely, LiDAR-based SLAM provides high accuracy and robustness, but at a higher cost and complexity. To address this trade-off, the use of indirect Time-of-Flight cameras, which provide direct depth measurements while remaining compact and cost-effective, in SLAM algorithms is investigated. This raises the first research question:

Research Question 1

Is it feasible to run SLAM algorithms for mobile robot platforms using solely the depth information of a ToF camera?

In the context of exploring the applicability of an indirect ToF camera in SLAM applications, it is imperative to consider the reliability of depth measurements when confronted with variations in environmental conditions. As indirect ToF cameras depend on modulated light signals, their performance is prone to being influenced by different lighting conditions and surface materials. This leads to the second research question:

Research Question 2

Is the chosen ToF camera more robust against different light conditions and materials in SLAM applications compared to an UGV setup equipped with a state-of-the-art LiDAR sensor?

CHAPTER 3

System Setup

This chapter presents the system setup for the integration of a 3Di camera into a mobile robot platform for SLAM. First, the used 3Di camera is introduced, followed by a description of the mobile robotic platform and the LiDAR sensor, which serves as a comparative reference in the experiments. Subsequently, the communication framework connecting these components is presented. Finally, sensor placement is examined, emphasizing the impact of the field of view (FOV) on collision avoidance and cliff detection.

3.1 3Di Camera

Key considerations for sensors in SLAM are depth accuracy, resistance to ambient light variations, frame rate, and integration capability. Hence, a suitable candidate is the 3Di camera from Infineon, namely the model 2877C, and thus is used in this setup. Due to its compact size and practical form factor, it is well-suited for UGV applications. The Infineon 2877C sensor is an indirect ToF single-chip sensor with illumination control and digital data output [41] as shown in Figure 3.1. The sensor has a FOV of 68° horizontally and 50° vertically. Its main advantage is the small form factor with high pixel resolution. It features a VGA system resolution achieved by a 640 x 240 twin pixel array within a compact 4 mm image circle. In addition, the sensor incorporates a patented Suppression of Background Illumination circuitry in every pixel, therefore enhancing robustness against strong sunlight. Interference with other ToF systems is prevented using Spread Spectrum Clock where the clock frequency - which controls the timing of light pulses and signal processing - is slightly varied over time. This reduces

Table 3.1: Use cases specification of Infineon iToF sensor 2877C

Parameter	**use case 0**	**use case 1**	**use case 2**	**use case 3**
$f_{mod,1}$ [MHz]	80.32	60.24	80.32	80.32
$f_{mod,2}$ [MHz]	60.24	-	-	60.24
$f_{mod,dual}$ [MHz]	20.08	-	-	20.08
Max Distance [m]	7.47	2.49	1.87	7.47
Max Exposure [μs]	1200	1000	1000	640

the likelihood of multiple sensors operating at the exact same frequency, minimizing crosstalk, and ensuring accurate measurements.

Furthermore, the sensor is equipped with four predefined use cases, as indicated in Table 3.1. These use cases vary in terms of modulation frequency and exposure time. Two of these use cases (1 and 2) are optimized for short-range applications and use a single modulation frequency. The remaining two use cases (0 and 3) use dual-frequency modulation, enabling longer-range measurements. The maximal exposure times for these use cases are different and are necessary to maintain compliance with eye-safety regulations. Within these constraints, the exposure time and the frame rate can be modified to ensure optimal sensor performance. Following the completion of each frame, these settings and the current use case selected can be adapted, facilitating the sensor for real-time adjustments. The Infineon 2877C is already used in various applications, such as secure face authentication, and automotive applications. In cars it is used for driver monitoring, smart airbags, and short-range car exterior, not for driving-related tasks.

In terms of connection, the sensor supports integration with either a Raspberry Pi or an NVIDIA Jetson platform. In the proposed use in a SLAM application, this sensor is used with a Raspberry Pi 4B, which is a high-performance single-board computer [42]. The device supports dual-band Wi-Fi (2.4 GHz/5.0 GHz), Bluetooth 5.0, and Gigabit Ethernet for reliable communications. The 8GB RAM version is used to ensure sufficient memory, high efficiency, and low power consumption - critical for battery-powered operation on the mobile robotic platform.

3.2 Mobile Robotic Platform

The used mobile robotic platform is the TurtleBot 4 Lite, a compact state-of-the-art robot platform designed for SLAM applications, in the following just referred to as TurtleBot [43]. The platform is built upon the iRobot Create 3 mobile base, which provides mobility and sensor integration, and operates on ROS2 enabling modular software development and integration with additional sensors and actuators.

The TurtleBot is composed of two on-board computing units: the Create 3 and

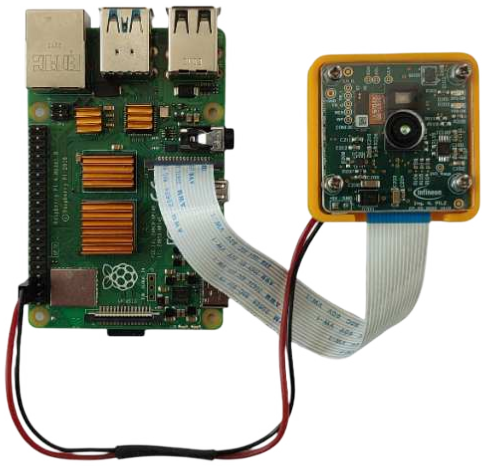


Figure 3.1: Infineon iToF sensor 2877C connected to Raspberry Pi 4B

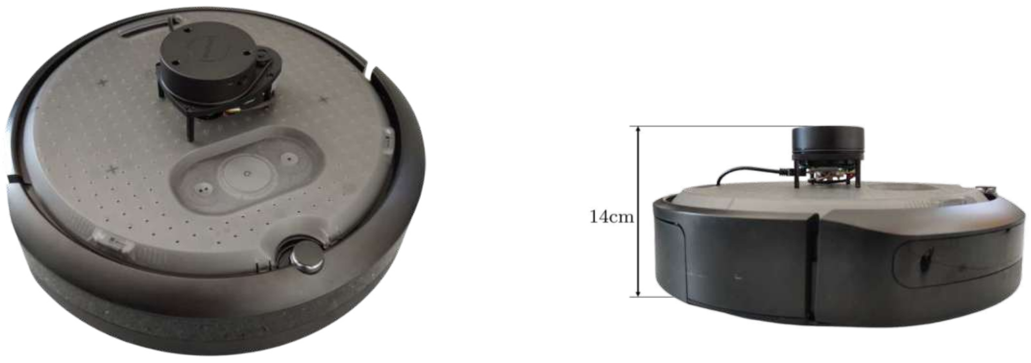
Figure 3.2: Slamtec LiDAR sensor RPLIDAR A1M8

the Raspberry Pi 4B. These are connected via an USB-C cable, which serves the dual purpose of powering the Raspberry Pi and establishing an Ethernet connection between the units. Both units are equipped with Wi-Fi cards. The Create 3 connects to 2.4 GHz networks, while the Raspberry Pi connects to 2.4 GHz and 5 GHz networks. For best performance, a router capable to support both bands is used.

While Create 3 is seen as the integrated control unit of the TurtleBot, the Raspberry Pi comes into use, if additional sensors are connected to the platform. The TurtleBot 4 Lite is equipped with an additional sensor, namely the RPLIDAR A1M8 LiDAR sensor, shown in Figure 3.2, which is used as the state-of-the-art technology for comparison with the Infineon 2877C iToF camera.

The RPLIDAR A1M8 sensor is connected to the Raspberry Pi via micro USB cable for power supply and data transmission. The sensor has a scan rate ranging from 2 Hz to 10 Hz adjustable via the motor signal [44]. Performance characteristics depend on the selected scan rate. At 5.5 Hz the sensor provides a maximum range of 12 meters, with 8000 samples per second, and an angular resolution of 1°.

In order to establish data transmission between the presented systems components, a communication network is required, which will be elaborated in the following section.



(a) Robot Platform with mounting plate (b) Side-view to outline setup height

Figure 3.3: Turtlebot 4 Lite mobile robot platform by iRobot equipped with RPLIDAR A1M8

3.3 Communication Network

The communication network ensures data exchange between the components. Figure 3.4 displays all the system components and their connection.

The camera setup, which provides the depth image, is shown on the left. The depth image stream is provided to the PC via TCP/IP over Wi-Fi, as indicated by the dashed arrow. The PC processes the incoming depth images and functions as an execution environment for the utilized SLAM. Therefore, it is connected via Wi-Fi with the robot platform using ROS2.

For ROS2-based communication, a network configuration has to be chosen. The setup with Turtlebot 4 provides two possible configurations: Simple Discovery and Discovery Server [45]. Simple Discovery relies on multicast communication, allowing nodes to automatically detect each other on the same network. This method is generally effective in straightforward setups. However, issues can arise in dual-band networks, potentially affecting communication reliability. Alternatively, the Discovery Server method centralizes node discovery, reducing network overhead and improving stability in more complex or resource-constrained environments. This approach is particularly useful when multicast traffic is restricted or when a more controlled discovery process is required.

Simple Discovery mode is recommended for systems with one to two robots and should match the requirements. In this mode ROS2 combined with Turtlebot 4 works with two different DDS vendors, FastDDS, and CycloneDDS. Here, FastDDS, the default vendor for the used ROS2 distribution, ROS2 Humble, is used. Furthermore, Simple Discovery allows to controll the Turtlebot 4 soley via Create 3. Therefore, the Raspberry Pi integrated in the Turtlebot 4 platform will only be used for experiments with the comparison technology, as it is used to integrate the LiDAR into the system.

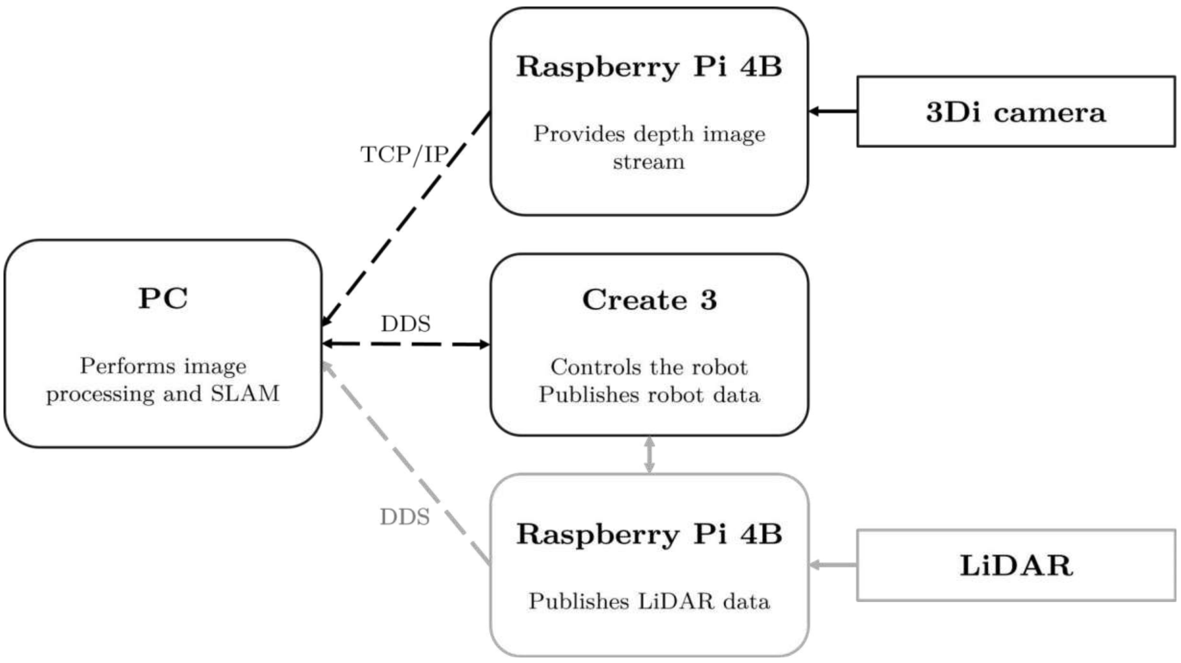


Figure 3.4: System setup components and connections. Solid lines represent wired connections, while dashed lines represent Wi-Fi communications. Sensors are shown in square boxes and computing units are shown with rounded edges. Gray elements represent additional devices for comparison technology.

3.4 Sensor Placement

A crucial point in the deployment of the presented Infineon iToF camera within a SLAM system is the considered sensor placement on the mobile robot platform. In order to make a trade-off between a wide range and near-field detection capability, the sensor is mounted on the front-top section of the mobile robot platform. The system now reaches a high of 14 cm, with the lens positioned at 11.5 cm. With the vertical FOV of the camera of 50° obstacles higher in the scene enter the camera's view at a distance of 5 cm from the sensor. Due to the high mounting point, the floor becomes visible from a distance of 25 cm. The floor detection range is particularly relevant for identifying potential cliffs preemptively, such as staircases, which is critical for safe navigation in various applications.

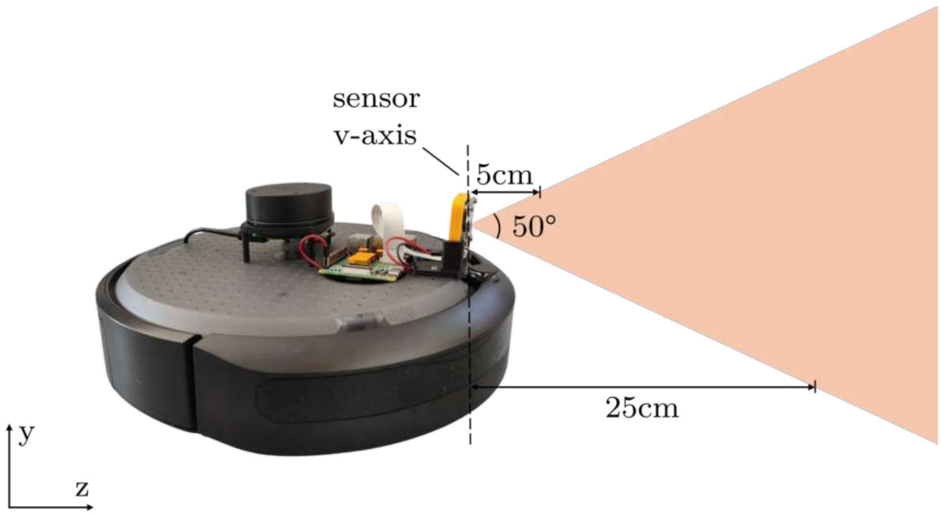


Figure 3.5: Sensor placement on the Turtlebot 4 with the resulting minimum distances for detecting head collisions or cliffs

CHAPTER 4

Software Implementation

This chapter outlines the software implementation for processing depth images into input data for the SLAM algorithm. The steps are outlined in Figure 4.1. The image processing steps ensure that all relevant environmental features contribute to the mapping task, which provides the estimated environment map.

In the preprocessing steps, the raw depth image data is reduced by removing unnecessary information with the goal of keeping only obstacles relevant for SLAM tasks. This includes a plane segmentation algorithm to remove the floor plane from the image. To enhance safety, a preemptive cliff detection mechanism is incorporated, utilizing points from the removed floor plane to detect negative obstacles. Finally, the image data are processed using a histogram-based filter heuristic in order to remove noise. The gained data are then ready to be transferred into a laser-scan-like representation. This transformation facilitates integration with SLAM algorithms that rely on LiDAR-based input, ensuring compatibility and robust environmental mapping.

4.1 Preprocessing

The preprocessing steps, comprising downsampling, point cloud transformation, and floor detection, apply filters to the raw depth image data to keep only the pixels relevant for obstacle detection.

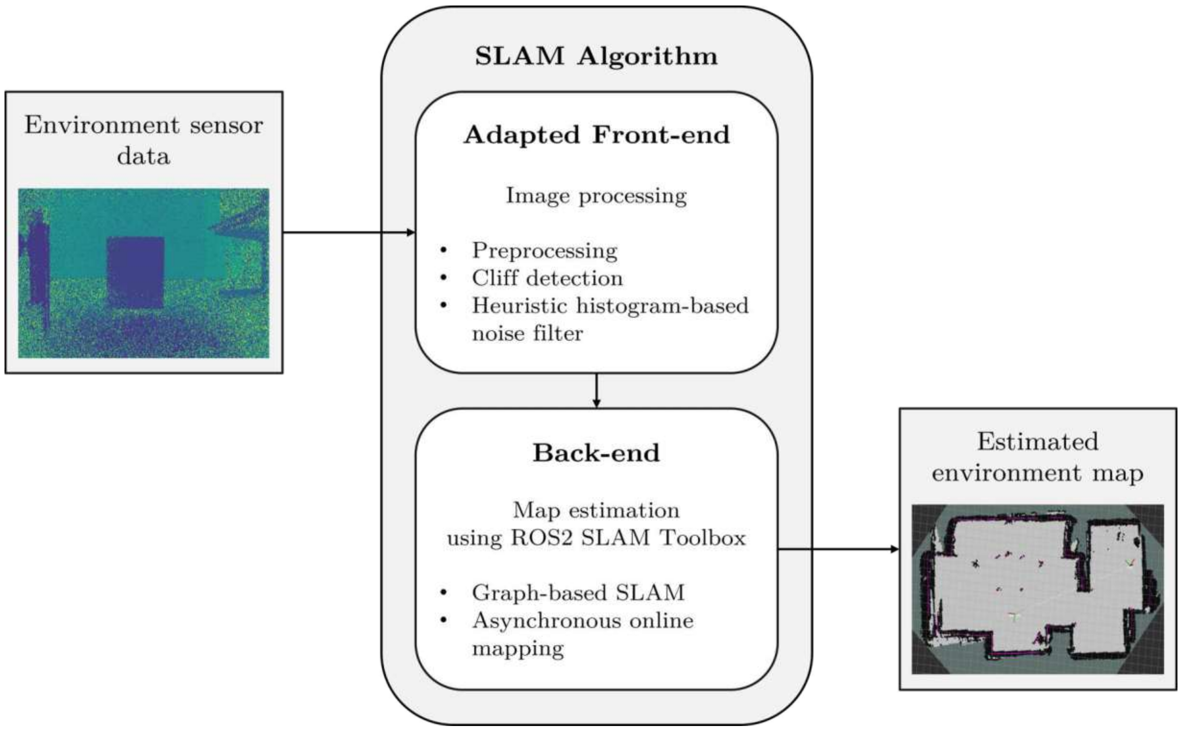

SLAM Algorithm
Environment sensor data
Adapted Front-end
Image processing
• Preprocessing
• Cliff detection
• Heuristic histogram-based noise filter
Back-end
Map estimation using ROS2 SLAM Toolbox
• Graph-based SLAM
• Asynchronous online mapping
Estimated environment map

Figure 4.1: Data flow and software steps in the SLAM process, starting with data input from the 3Di sensor. The front-end processes the images, applying preprocessing, cliff detection and noise filtering. The back-end estimates the map using the ROS2 SLAM Toolbox to generate an estimated environment map.

4.1.1 Downsampling and Point Cloud Transformation

The initial image processing steps serve to reduce computational effort while preserving essential information. Therefore, the input depth image undergoes a downsampling process, where only every fifth column is retained. The selection of every fifth column was empirically determined to achieve optimal balance between depth information and computational efficiency. The downsampling is set to delete columns instead of rows, as the horizontal resolution is higher than the vertical resolution. Moreover, the last step of the software implementation, where the 3D image data is transformed to a 2D array, scans the image along the horizontal FOV. By downsampling the columns, a higher computational advantage is achieved while losing less information. Subsequently, a filter is applied in order to remove all points with z-values that fall outside the camera's valid depth range.

In order to disregard points that are irrelevant for the purpose of obstacle detection, such as the floor or points that are outside the range of the robot, the downsampled depth image is transformed into a point cloud. This is achieved by inverting the projection equation from the 2D image depth values in order return the 3D scene points

$$\mathbf{M} = \mathbf{K}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{4.1}$$

where $D(u, v)$ is the depth value in the image in position (u, v). The inverse intrinsic matrix $\mathbf{K}^{-1}$ removes the effects of the intrinsic properties of the camera and leaves a normalized direction vector, which is scaled by depth. $\mathbf{K}$ is defined in Equation (2.2). The intrinsic parameters necessary for the matrix are calculated from the FOV and the image size in pixels of the Infineon 2877C. The intrinsic parameters in x-direction are calculated using the horizontal field of view FOV_h of 68°, and the image width of 640 pixels, which results in a principal position c_x and a focal length f_x of

$$c_x = \frac{N_C}{2}, \text{ and} \tag{4.2}$$

$$f_x = c_x \frac{1}{\tan(\mathrm{FOV}_h/2)}. \tag{4.3}$$

The intrinsic parameters in y-direction are calculated using the vertical field of view FOV_v of 50°, and the image height of 240 pixels, resulting in a principal position c_y and a focal length f_y of

$$c_y = \frac{N_R}{2}, \text{ and} \tag{4.4}$$

$$f_x = c_y \frac{1}{\tan(\mathrm{FOV}_v/2)}. \tag{4.5}$$

With the calculated parameters, the projection matrix for the Infineon 2877C is

$$\mathbf{K} = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 474.42 & 1 & 320 \\ 0 & 257.34 & 120 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4.6}$$

As the cameras extrinsic parameters are not considered in Equation (4.1), the resulting 3D points are perceived in a 3D coordinate system with its origin in the camera center.

Since the point cloud still contains the entire scene, more filtering is necessary in order to focus on the obstacle data for SLAM.

4.1.2 Floor Detection

The next step is to delete the points corresponding to the floor plane from the downsampled point cloud. As this setup operates in indoor applications, a level floor is assumed. Therefore, a Random Sample Consensus (RANSAC) algorithm is used to detect the floor plane in the point cloud.

RANSAC is an iterative method for estimating model parameters from a dataset [46]. It selects random subsets, constructs a model based on this subset, and evaluates their fit to the dataset. Widely used in computer vision and geometry processing, RANSAC detects geometric primitives such as planes, spheres, and cylinders in point clouds. This implementation uses Open3D's built-in RANSAC function for floor plane detection.

Since RANSAC only returns the plane model that best fits the majority of points in the scene, this can be a problem, for example, if the robot starts in front of a wall. Therefore, the basic RANSAC algorithm is adjusted to only accept a detected plane model if it matches the floor. The adapted RANSAC algorithm is outlined in Algorithm 1, where θ defines the geometric form. For plane detection θ includes the plane parameters $[a,\ b,\ c,\ d]$ from the plane equation

$$0 = ax + by + cz + d. \tag{4.7}$$

To check if the detected plane model matches the floor, it is analyzed at a reference point, as the position of the floor is evident at the camera's position. This reference point is then utilized to validate the detected plane by leveraging the known mounting height of the camera and the fact that the point cloud originates from the camera's coordinate system (x, y, z), as illustrated in Figure 2.2. Consequently, the tilt of the plane is not predetermined, but rather its offset d, which is used for verification. The algorithm is thus modified to accept the detected plane model only if it aligns with the expected parameters.

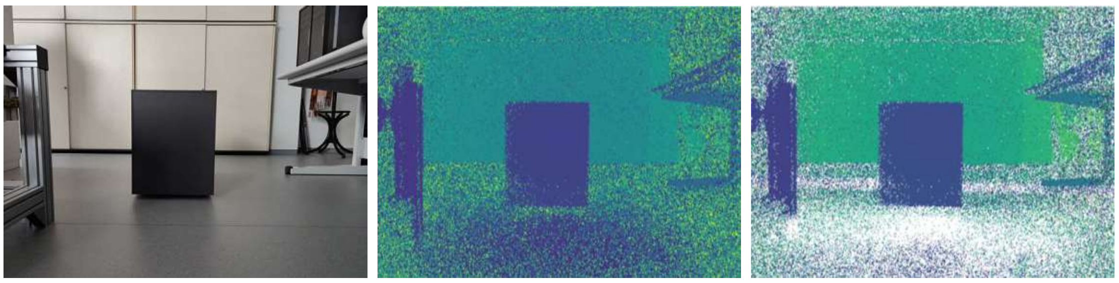
In the event that the initial run does not meet this criterion, it is probable that a wall has been detected instead of the floor. To enhance floor detection, the point cloud is cropped at a height of $y = 10\,\text{cm}$, only the points below remain, and the plane estimation is performed again. This increases the likelihood that more floor points than wall points remain in the scene, allowing for a more accurate estimation of the floor plane model.

Algorithm 1: Adapted RANSAC Algorithm

```
Input: - Dataset $P$
       - Model parameters $\theta$
       - Max iterations $N$
       - Inlier threshold $\epsilon$
Output: best model parameters $\theta^*$
Initialize best model $\theta^* \leftarrow \emptyset$
Best inlier count $n^* \leftarrow 0$
Initialize $floor\ model\ not\ found$
Run RANSAC:
while floor model not found do
    for $i \leftarrow 0$ to $N$ do
        Select a random subset of $k$ points from $P$
        Estimate model parameters $\theta$ from the selected points
        Compute the number of inliers in $P$ that fit $\theta$ within threshold $\epsilon$
        if $n > n^*$ then
            Update best model $\theta^* \leftarrow \theta$
            Update best inlier count $n^* \leftarrow n$
        end
    end
    Check if the detected plane matches floor:
    if plane is floor then
        $floor\ model\ found$
    end
    if plane not floor then
        Cut point cloud at height of 10 cm over the expected floor
        Initialize best model $\theta^* \leftarrow \emptyset$
        Best inlier count $n^* \leftarrow 0$
    end
end
return best model $\theta^*$
```

To limit the computational effort of the plane estimation, the RANSAC algorithm is only performed at the beginning of the program. This implies that the floor is even and that no ramps occur as the robot moves through the scene. If this is not the case, plane estimation has to be executed in the program for every new input image or every few input images.

Once the ground plane is identified, it is used to separate the points associated with



(a) Original scene captured by an RGB camera (b) Raw depth image of the whole scene showing noise (c) Depth image after preprocessing

Figure 4.2: Floor detection and invalid pixel removal on an example scene with a gray box 1.5m in front of the camera, and a wall 3.5m in front of the camera. On the left of the scene is a aluminium pillar, and on the right of the scene is a table.

the ground from the image. Therefore, the distance from every point in the point cloud to the floor plane is calculated and every point in a threshold of ±1.5 cm is marked as a floor point and deleted from the point cloud. This threshold serves to counteract the uncertainty of ground points that do not perfectly match the plane model.

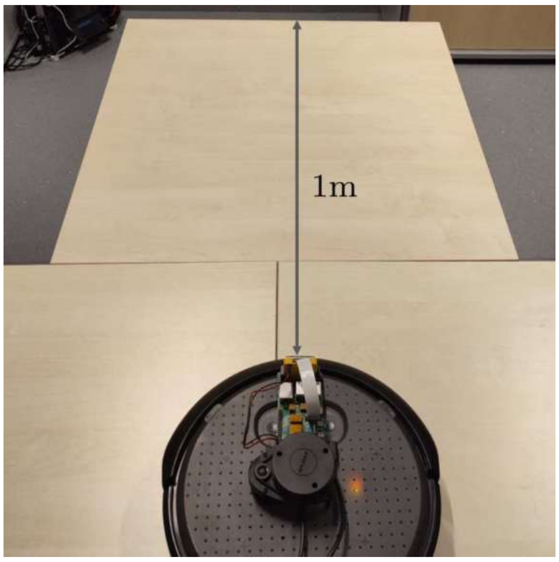
The preprocessing steps are shown on a basic scene example, displayed in Figure 4.2a. A gray box is positioned at a distance of $d_b = 1.5\,\text{m}$ in front of the camera, whereas the wall is at a distance of $d_w = 3.5\,\text{m}$. On the left an aluminum pillar is placed, and on the right a part of a table is visible. This scene results in a depth image shown in Figure 4.2b. After the processing steps, pixels outside of the camera range and pixels irrelevant for obstacle detection and mapping are deleted. This results in Figure 4.2c, where clearly the floor and some noise are removed.

If the environment is known and free of stairs or other significant elevation changes, the processed point cloud can be forwarded directly to noise filtering and laser scan generation. However, if the presence of such obstacles is uncertain, a cliff detection algorithm is implemented.

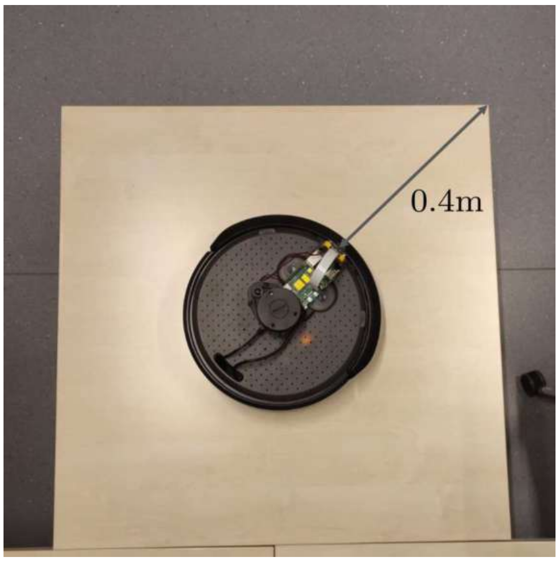
4.2 Cliff Detection

Mobile robot platforms, such as the Turtlebot, are equipped with cliff detection sensors. These sensors are usually mounted on the bottom of the platform in front of the first wheel. With these infrared sensors, the robot detects if no floor is in front and stops driving. Unless the robot is exactly in front of the cliff, it is not observed with any sensor. To enhance planning for autonomous driving applications, a preemptive cliff detection algorithm is implemented in this setup.

The proposed algorithm is based on the assumption that a cliff can be inferred by the absence of floor points in the recorded depth image. Due to the camera's low



(a) Scene 1 with a cliff in 1 m distance from the camera origin



(b) Scene 2 with a cliff in 0.4 m distance from the camera origin

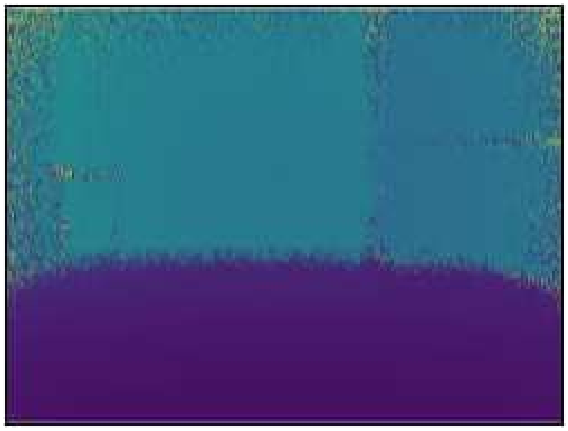
Figure 4.3: Example scenes for cliff detection algorithm with cliffs at different distances from the camera position

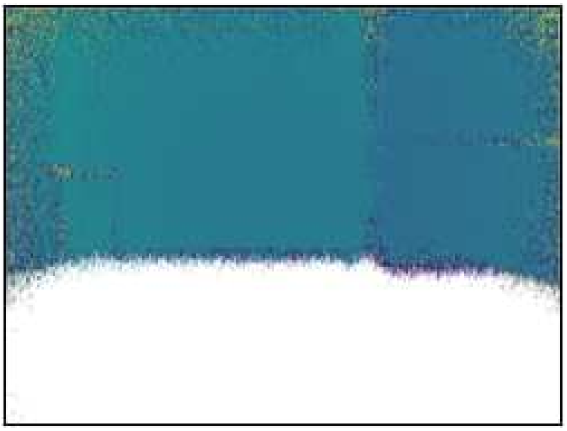
mounting position and its perpendicular orientation to the floor, according to Figure 3.5, the observed floor area is limited. To ensure reliable cliff detection, a dense distribution of floor points must be present within the analyzed region. In the event that the density of detected floor points is deemed insufficient, a cliff is assumed in that area. For this setup, the analyzed region for cliff detection extends from 0.3 m to 0.6 m on the z-axis of the camera origin, and from −0.2 m to 0.2 m on the x-axis. This area is expected to contain a sufficiently dense distribution of floor points.

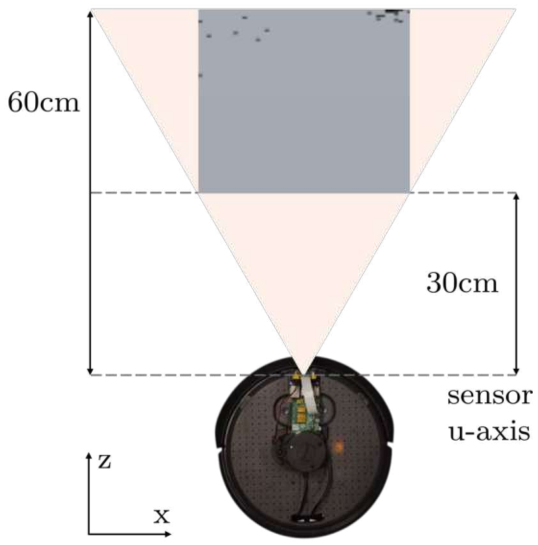
The algorithm for preemptive cliff detection is explained using two examples, where scene 1 represents a scenario with a cliff 1 m in front of the robot, as shown in Figure 4.3a, and scene 2 represents a scenario with a cliff 0.4 m in front of the robot, as displayed in Figure 4.3b. The depth images from the scenes are shown in Figure 4.4a and Figure 4.5a, where violet areas depict the ground plane, blue areas represent the walls at a distance of 2 m and 2.5 m, respectively, and yellow pixels represent noise.

The idea of the method is to add information about detected cliffs to the preprocessed depth image, which at this step contains only information about obstacles in the scene and noise. By adding a barrier to the region of the point cloud where no floor could be found, an obstacle will be detected in the next step of the algorithm and therefore the cliff is mapped as obstacle in the resulting map from SLAM. Subsequently, the area behind the cliff is avoided in path planning for autonomous driving.

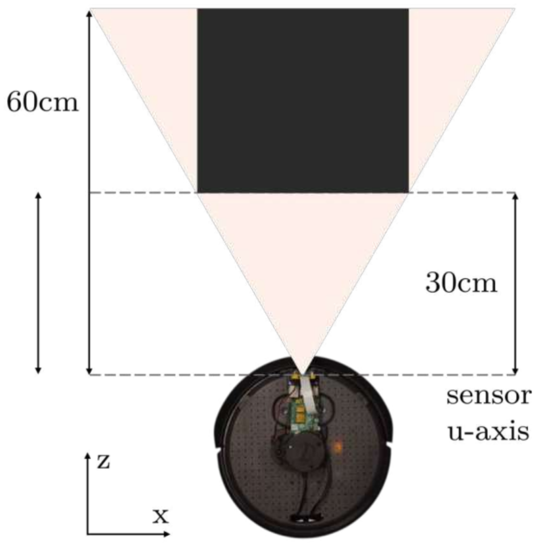
In the first step of the algorithm, the designated floor region is transformed into a top-down view depth image. This facilitates a structured analysis of floor coverage. As the region of interest is chosen in distances where dense floor points can be expected, the resulting floor image should appear as a continuous square for the analyzed region,


(a) Original depth image of scene 1 with floor in front


(b) Resulting depth image after floor removal and cliff detection

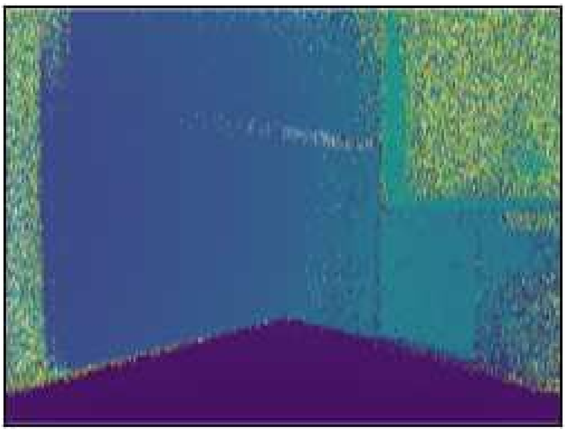


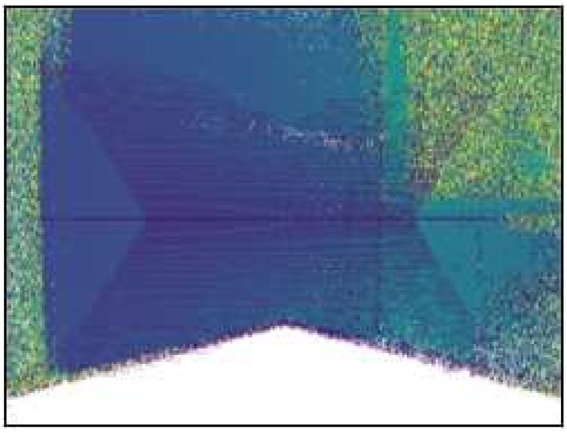
(c) Detected floor plane shows dense floor points (light gray) in the area 0.3-0.6m in front of the camera in scene 1, top-view

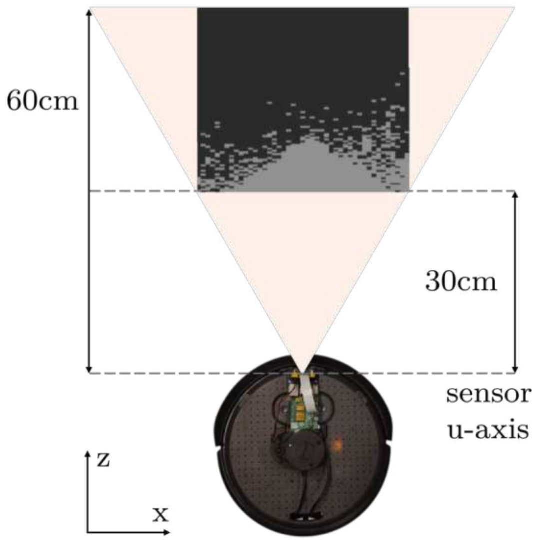


(d) Detected cliff plane shows no cliff (no light gray pixels) in the area 0.3-0.6m in front of the camera in scene 1, top-view

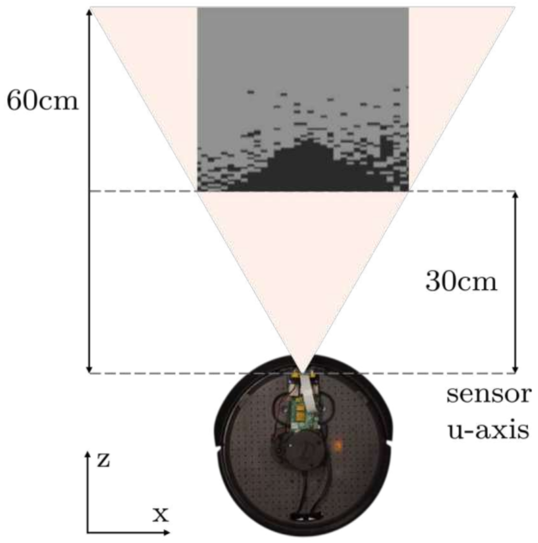
Figure 4.4: Cliff detection algorithm applied on scene 1, with original depth image, intermediate steps and the resulting depth image


(a) Original depth image of scene 2 with floor in front, yellow points determine noise


(b) Resulting depth image after floor removal and cliff detection, violet points in the middle display the barrier at cliff distance



(c) Detected floor plane shows only floor points (light gray) in the area 0.3-0.4m in front of the camera in scene 2, top-view



(d) Detected cliff plane shows a cliff (light gray) in 0.2m distance from the camera in scene 2, top-view

Figure 4.5: Cliff detection algorithm applied on scene 2, with original depth image, intermediate steps and the resulting depth image

if no cliff is in the designated area.

For scene 1, where the cliff is 1 m ahead, the transformed floor image is displayed in Figure 4.4c. The image data displays a top-down view. The robot is added to the figure to capture the context. Floor points detected using plane segmentation are displayed in light gray, while dark gray pixels indicate areas where no floor point is detected. In scene 2, where the cliff is at a distance of 0.4 m, the resulting top-down view of the floor is displayed in Figure 4.5c. The light gray pixels indicate the area up to 0.4 m away from the robot, where floor points are detected. Furthermore, on the right on the cliff noise is detected at floor level, leading to a less sharp edge detected.

It is important to note that missing pixels may appear in the ground plane. These gaps are analyzed, and if they are identified as outliers within an otherwise dense region, they are removed to preserve the continuity of the detected floor plane. The processed image is then analyzed in order to determine whether it forms a complete square or whether there are missing floor pixels. In regions where floor pixels are absent, the presence of a cliff is inferred.

The goal is to add a barrier to the point cloud data at the position of the detected cliff. Therefore, the image is inverted so that missing floor areas appear as distinct pixels, while detected floor points are removed. This step is displayed in Figure 4.4d for scene 1. The figure shows that for this scenario, no cliff is detected, as all points are deleted. In contrast, the result for scene 2 is presented in Figure 4.5d. The cliff is displayed in light gray points, while the floor points are deleted, as they are not necessary for mapping.

In the last step, the identified cliff points are projected back into the three-dimensional space using the known transformation parameters and incorporated into the existing point cloud, which is the result of the preprocessing method. For scene 1 the result after the preemptive cliff detection algorithm is shown in Figure 4.4b. Compared to the raw input image of the scene only the floor is removed, as no cliff is detected, and thus the algorithm is not adding any data points to the image. The result for scene 2 is shown in Figure 4.5b. In this case a cliff is detected, and the added data points are displayed in the middle of the image, representing a barrier at the cliff distance of the robot.

Up to this point in the analysis, the processing has focused on identifying the floor, detecting cliffs, and accounting for the camera's limited range. However, the resulting point cloud still contains noise, which has the potential to interfere with further processing. Therefore, in the next step, a noise filter is applied, and the filtered data is immediately transformed into the scan information necessary for SLAM.

4.3 Heuristic Histogram-based Noise Filter

The transformation of depth data into a ROS2 *scan messages*, suitable for the LiDAR-based SLAM algorithm, necessitates further extraction of the relevant depth information from the point cloud. The 3D depth information is transformed to a 2D scan information sufficient to solve the SLAM problem. As the preprocessed data is still affected by noise, a filtering step is required to ensure that the generated scan accurately represents the nearest obstacles.

For scan messages, a specified message type exists in ROS2 [47]. Additional to the header information, which includes information, such as the timestamp, which is necessary for every ROS2 message, a scan message has to contain the following information:

Listing 4.1: Laser Scan Parameters

```
float32 angle_min
float32 angle_max
float32 angle_increment
float32 range_min
float32 range_max
float32[] ranges
```

The angle parameters `angle_min` and `angle_max` result from the horizontal FOV, while the variable `angle_increment` is a result of the number of range values over the FOV. The range minimum and maximum `range_min` and `range_max` is necessary to filter invalid scan points and are identically to the sensor range. The `ranges` array includes the scan data. This data is extracted from the depth image after the cliff detection is performed.

In order to realize this, a heuristic histogram-based filter with an adaptive threshold is applied to filter noise and simultaneously extract the relevant depth values for the 2D scan message for the SLAM algorithm. As the filter processes depth images, the preprocessed point cloud is transformed to a depth image again.

The scan information is selected by analyzing the transformed depth image along the horizontal FOV. The concept is shown in Figure 4.6. The depth image on the left represents the input image for the filter algorithm. The filter determines the range values along the horizontal FOV and stores them in a scan message array, as emphasized below the depth image. The method for obtaining the range value for an area is based on a histogram, as shown on the right. The function in the histogram represents the threshold function. As soon as one histogram bin shows values higher than the threshold function, an obstacle is detected at this distance, and the belonging distance is assigned to the scan array. The ordinate section defines the number of values at a direct distance from the camera ε_{max}. As distance increases, fewer values are required, at least ε_{min} at the maximum camera range.

Algorithm 2: Heuristic Histogram Filter

```
Input: Depth image $image$
Output: Filtered depth scan $scan$
set scan variables:
    coefficient of scan values over image width $scan\_index$
    $num\_steps \leftarrow width/scan\_index$
    $sample\_size$
Initialize histogram-based filtering parameters:
    histogram bin-size $resolution$
    y-offset of the threshold function $\varepsilon_{max}$
    gradient of the threshold function $\varepsilon_k$
for $i \leftarrow 0$ to $num\_steps$ do
    $sample\_image \leftarrow$ columns around position i to achieve sample_size
    $sample \leftarrow$ valid points from sample_image
    Check if sample includes valid points, otherwise continue with the next sample

    Compute histogram bins:
        calculate $num\_bins \leftarrow \mathrm{int}\left(\frac{\max(sample)-\min(sample)}{resolution}\right)$
    Compute histogram:
    $hist \leftarrow \mathrm{histogram}(sample, num\_bins)$
    Compute adaptive threshold for the distances of each bin:
        $\varepsilon \leftarrow \mathrm{int}(\varepsilon_{max} - dist_{hist} \cdot \varepsilon_k)$
    Update scan value at position $i$ with first distance-value where histogram
      value $> \varepsilon$
end
return $scan$
```

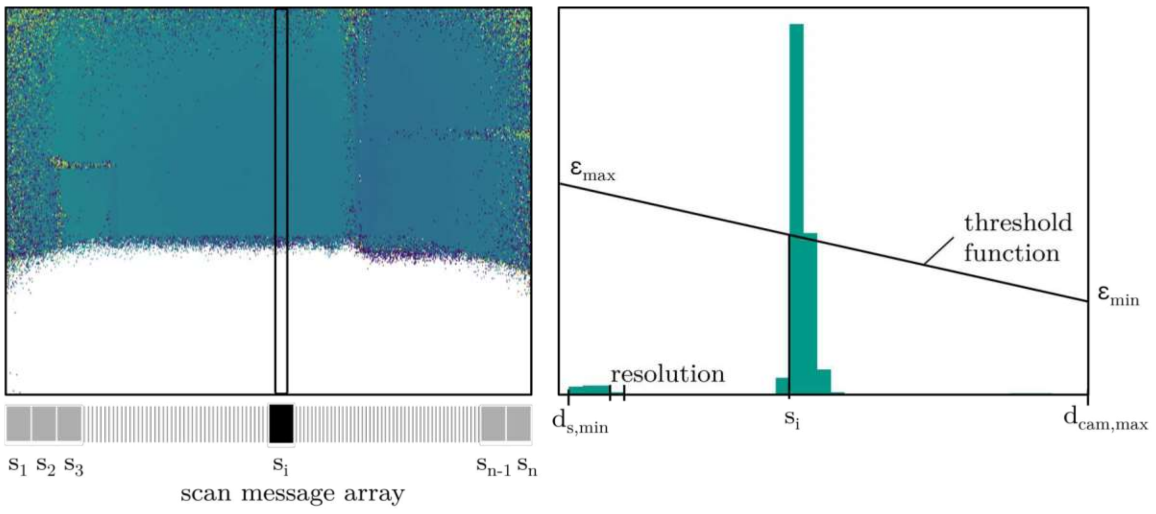


Figure 4.6: 2D scan information extracted from a preprocessed depth image, one sample is abstracted at a time (left). The sample is analyzed with a histogram-based filter approach (right), where a threshold function is used to identify the closest obstacle. All histogram bins that remain below the function are considered noise. The obstacle distance is stored in a scan message array.

In the algorithm for histogram-based noise filtering, as displayed in Algorithm 2, the image is divided into areas, the so-called samples. In Figure 4.6 one sample image column i is marked. The scan array of size n stores the determined depth value from the histogram at position i.

The resolution of the scan array is defined via two parameters, the scan index and the sample size. The scan index defines how many scan values are extracted from the depth image, while the sample size defines how wide the analyzed area is around the actual position of the scan value. Depending on the scenes, the sample size is an important value to ensure the detection of obstacles that are stretched in width, not in height. In this application, the sample size is set to 5 columns per sample. In order to half the computational effort, the scan index is set to 2.

With the parameters set, the local depth sample data is extracted for one area after the other and a histogram is generated. In order to remove noise while ensuring accurate depth estimation, an adaptive threshold ε is applied

$$\varepsilon = \varepsilon_{max} - d_{hist} \cdot \varepsilon_k = \varepsilon_{max} - d_{hist} \cdot \frac{\varepsilon_{max} - \varepsilon_{min}}{d_{cam,max}}, \tag{4.8}$$

where ε_{max} and ε_{min} determine the threshold at maximum and minimum camera range respectively. d_{hist} is the distance on the x-axis of the histogram, and $d_{cam,max}$ is the maximum camera range. While the camera range is set for each use case, the minimum and maximum values depend on the resolution of the scan. The threshold decreases with increasing depth. This is due to the fact that a greater number of valid depth measurements are available at closer distances. If the histogram contains values exceeding this threshold, the closest valid depth value is selected and stored in the output array *scan*.

The heuristic histogram filter is used to remove noise and determine the scan information in one step. The resulting scan message is expected to reliably capture the closest obstacle, whether a physical barrier or a cliff, across the horizontal FOV, ensuring the robot avoids hazardous areas. This scan information is sent to the SLAM algorithm to perform the environment mapping.

CHAPTER 5

Experiments and Results

In order to evaluate 3Di cameras in SLAM applications, the accuracy and applicability of the sensor are tested in various experiments. At first a sole sensor analysis is done to further analyze its characteristics in all available use case settings. Thereafter, the sensor accuracy and repeatability is analyzed in the context of SLAM, measuring the error in the SLAM output maps. The last experiment focuses on the obstacle and cliff detection. For comparison, the SLAM related tests are additionally performed with the state-of-the-art LiDAR technology, the RPLIDAR A1M8.

5.1 Methodology

At first, the different conditions and settings used for the experiments regarding accuracy and repeatability, as well as the method to evaluate the accuracy from the generated data, are outlined.

5.1.1 Testing the 3Di Camera under various Environmental Conditions

The accuracy and the repeatability of the sensor is evaluated under different lighting conditions and on different surfaces commonly found in indoor environments. The analyzed lighting conditions are:

- closed blinds, with no artificial lighting turned on to achieve a dark room surrounding,
- office lighting, with closed blinds and artificial lights turned on, and
- daylight, where the blinds are opened, but no additional artificial lighting is turned on.

The analyzed surface materials are:

- black fabric, to test the sensor on light absorbing material,
- foam material, to test the sensor on uneven surfaces, and
- glass, to test the sensor on transparent surfaces.

The measurements under different light conditions are performed against white walls, whereas the measurements on different surface materials are performed with closed blinds. All tests are performed moderate outdoor lighting conditions, without direct sunlight incidence on the device under test.

5.1.2 Data Analysis

To measure the accuracy of the Infineon 2877C sensor, the trueness μ is calculated, as introduced in Equation (2.19). The trueness represents the absolute mean error on a dataset. The closer the value is to zero, the higher is the accuracy of the dataset. One dataset consists of one or more measurements with the 3Di camera performed under the same conditions, from the same distance, on the same surface, with the same lighting conditions.

To get a more general representation of the data accuracy, the mean trueness $\overline{\mu}$ is introduced as

$$\overline{\mu} = \frac{1}{S} \sum_{s=0}^{S} \mu_s, \tag{5.1}$$

where S defines the total number of datasets, and μ_s represents the trueness of one dataset.

To analyze the accuracy and repeatability of the 3Di camera and the LiDAR sensor in SLAM applications, the error is calculated by comparing the occupancy grids returned after performing the SLAM algorithm with the ground truth map, as shown in Figure 5.1. The occupancy grid map is color coded with three colors, where white represents free space, black represents obstacles, and gray represents unknown terrain. The resulting occupancy grid map is scattered, while the ground truth is of a sharp square shape. For data analysis, only pixels relevant for collision avoidance are considered. This is done by keeping only the interior outline of the found obstacle pixels, which are used to

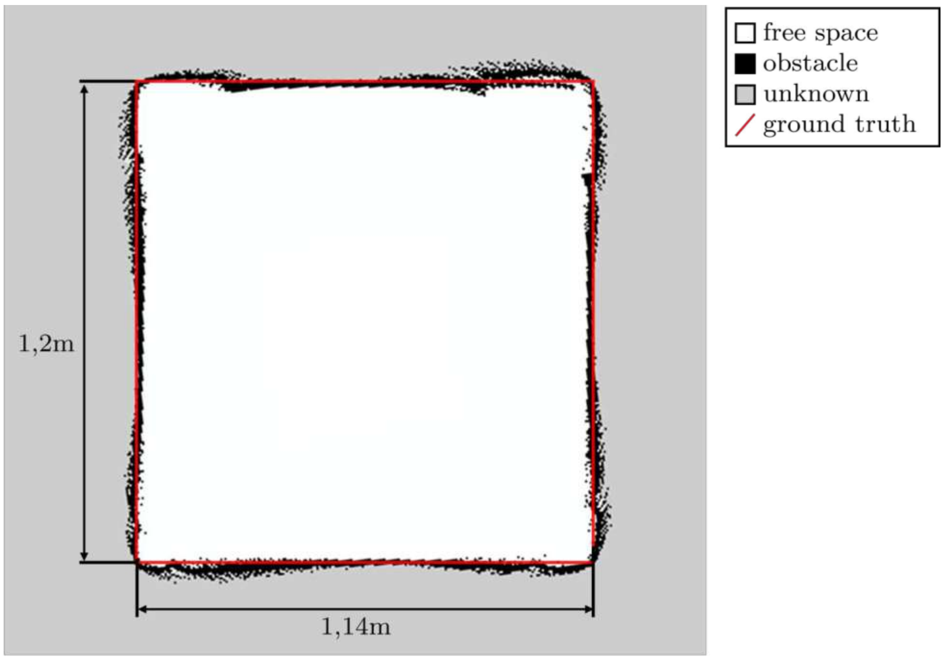


Figure 5.1: Comparison between an occupancy grid map and the ground truth to evaluate the accuracy of 3Di camera and LiDAR sensors. Free space, obstacles, and unknown terrain are color-coded, while the red outline represents the ground truth. Only the interior obstacle boundaries from the scattered occupancy grid for error analysis.

calculate the error between the obstacles found and the ground truth distances, and thus results in the accuracy value $\overline{\mu}$.

To analyze the repeatability, the standard deviation on the datasets is calculated by comparing the trueness μ_i of one frame i to the trueness of one dataset μ with n frames in total. This results in the repeatability

$$\sigma = \sqrt{\frac{\sum_{i=1}^{n}(\mu_i - \mu)^2}{n}}. \tag{5.2}$$

5.1.3 SLAM Parameters

SLAM parameters are used to tune the SLAM characteristics of the used ROS2 SLAM Toolbox. The parameters are divided into correlation parameters and map and general SLAM parameters. Correlation parameters define scan matching characteristics, while the map and general slam parameters specify variables such as a coefficient of trust in odometry and map update intervals.

The map parameters are especially relevant for the following experiments, as they define the accuracy of the occupancy grid map. As the map is the basis for the analysis, the resolution defined in the SLAM settings represents a bottleneck for accuracy analysis.

For the accuracy and repeatability experiments, the map resolution is set to 0.005 m. With the angle increment between depth values of the incoming scan measurements and the measurement ranges of the used sensors, a map resolution of 0.005 m provides maps without a lack of information on the sensors FOV. For the experiments on obstacle and cliff detection, the resolution is set to 0.05 m. As the scene for this test is wider, a higher resolution ensures a consistent mapping. For all tests, scan matching is deactivated, to achieve better comparison between the 3Di camera and the LiDAR. This is relevant, since LiDAR has a horizontal FOV of 360° and therefore provides more information than the camera with a horizontal FOV of 68°, which will result in better scan matching and more accurate maps under uneven conditions.

5.1.4 Experimental Setup for Static Sensor Analysis

The objective of this experiment is to analyze the performance of the 3Di camera under various conditions to determine the optimal sensor settings for a subsequent SLAM algorithm. The sensor is tested with all use cases (cf. Table 3.1) at different distances, light conditions, and surface materials to evaluate its accuracy in terms of trueness and precision.

For each measurement procedure, lighting and surface material are specified. One procedure consists of several measurements at predefined distances from the surface. The sensor is analyzed from 0.05 m to 5 m. With the sensor specifications in Table 3.1 only the maximal unambiguous range can be calculated, but information regarding a minimum range is not known. Therefore, the spatial increments at closer distances to the analyzed surface are set smaller. The test points are at 0.05 m, 0.1 m, 0.15 m, 0.2 m, 0.25 m, 0.5 m, 1 m, 1.5 m, 2 m, 2.5 m, 3 m, 3.5 m, 4 m, 4.5 m, and 5 m. At each distance, ten consecutive measurements are recorded for each of the four camera use cases to improve statistical repeatability. Afterward, a 20×20 pixel sample is extracted from the center of each image. The narrow sample region facilitates the reduction of potential distortions from lens effects and edge artifacts. This sample serves as the basis for further analysis.

At first, a comparison of all use cases over the distance is provided. The measurement is compared by its trueness μ at each measurement position. From these data, a conclusion about the reliable measurement range of each use case is made. Subsequently, the mean error across all measurements in the reliable range is determined by calculating the mean trueness $\overline{\mu}$ to ensure a robust comparison of all use cases.

While this experiment serves to analyze the sensor itself, the following focus on the sensor in SLAM applications.

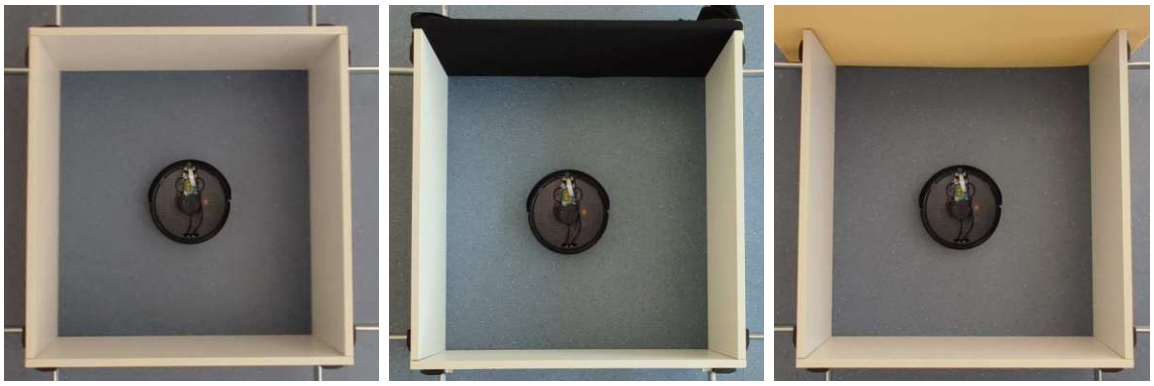

(a) white walls (b) black fabric (c) foam material

Figure 5.2: Experiment setup to test the accuracy and repeatability of the 3Di camera and the LiDAR in SLAM. To compare the results with different surfaces involved, one wall at a time is replaced with the material of interest.

5.1.5 Experimental Setup for Accuracy and Repeatability Tests in SLAM

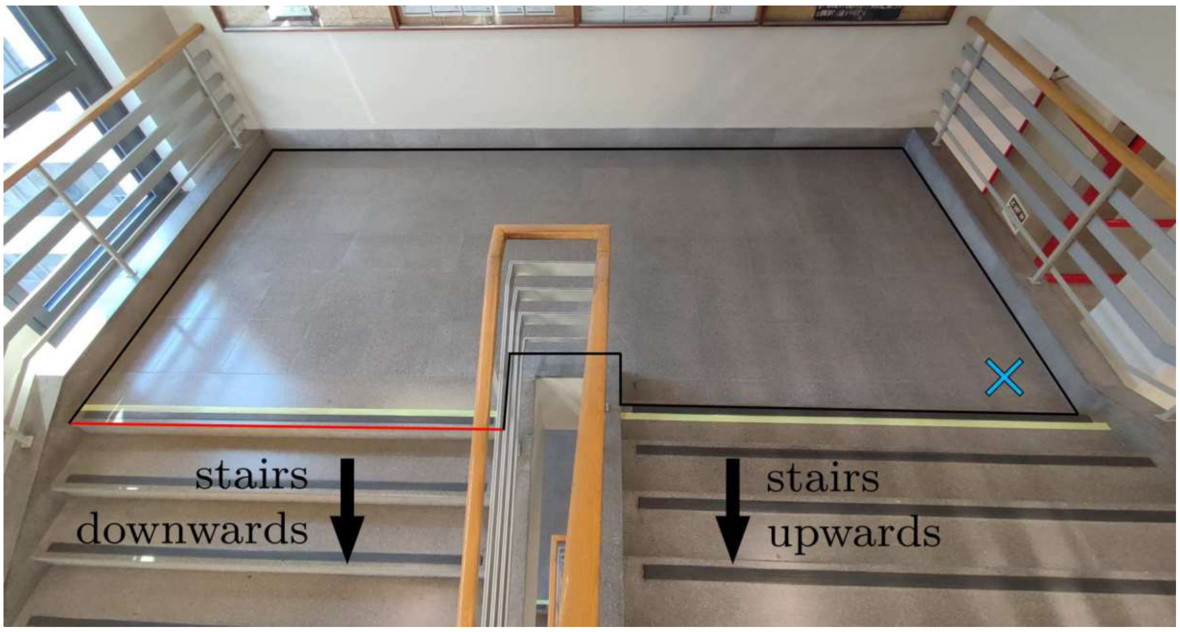
For this experiment the setup is installed as in Section 3.4 with the depth camera mounted on the Turtlebot. The robot is positioned in a 1.2 m × 1.14 m square space surrounded by four walls, ensuring a controlled test environment. The goal is to analyze sensor accuracy and repeatability under the introduced light conditions and on the different surface materials. The camera operates in use case 0, see Table 3.1 for specifications. The LiDAR sensor is used to perform the same tests with a state-of-the-art sensor for comparison.

For each scenario, five datasets are collected using SLAM. For map generation the robot rotates 360° with a rotation velocity of 0.15 rad/s. The resulting map from one frame can look like Figure 5.1. By comparing the received data with the ground truth map, accuracy and repeatability can be calculated for every scenario. The setup for testing the different light conditions is shown in Figure 5.2a. The impact of different surface materials is assessed by replacing one of the white walls with black fabric, see Figure 5.2b, or foam, see Figure 5.2c to evaluate surface effects.

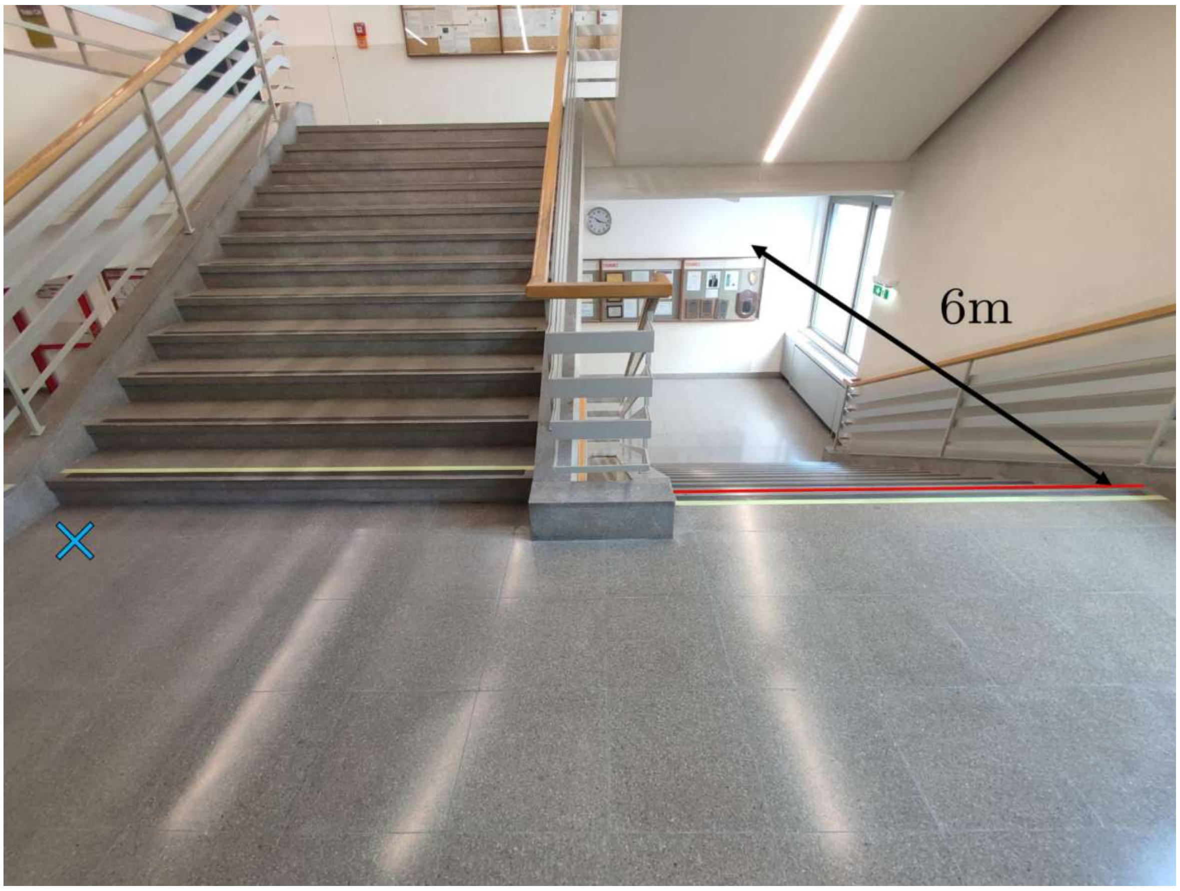
5.1.6 Experimental Setup for Obstacle and Cliff Detection Accuracy Tests

To test setup accuracy in a practical context, SLAM is performed in an environment with some difficulties, such as obstacles of small height, and a cliff area.

The experiment is carried out on the landing of a staircase, as illustrated in Figure 5.3a and Figure 5.3b. This environment is selected to assess the system's capacity to detect both obstacles, such as stairs leading upward, and cliffs, such as stairs



(a) View on the scene from the staircase with the ground truth plane marked in black and red



(b) View on the scene from the white wall with the cliff marked in red.

Figure 5.3: Scene for evaluating the system's ability to detect obstacles and cliffs. The red and black lines indicate the ground truth of the floor plane, with the red marking the cliff section. The blue marker is for orientation in the scene and in the output map.

leading downward. The robot is positioned on the landing platform, which measures 13.7 m^2 and is surrounded by a white wall. The floor and the lower part of the wall are covered with gray tiles. As illustrated in Figure 5.3a, a sketch of the floor plane is included, representing the ground truth that is the anticipated result of the SLAM algorithm. The red border delineates the cliff section of the environment. The blue marker is for orientation in the scene and in the output maps.

The test is performed once with the 3Di camera and with the LiDAR scanner. The robot drives with an angular speed of 0.15 rad s^{-1} and a linear speed of 0.2 m s^{-1}. The sensor is analyzed using the error between the ground truth map and the resulting map. As cliff detection from LiDAR data is a known challenge, the analysis is carried out separately. First, all map borders except the cliff, represented by black lines in Figure 5.3a, are analyzed in one patch. Then, the data belonging to the cliff, marked in red in Figure 5.3a, are analyzed separately.

5.2 Results

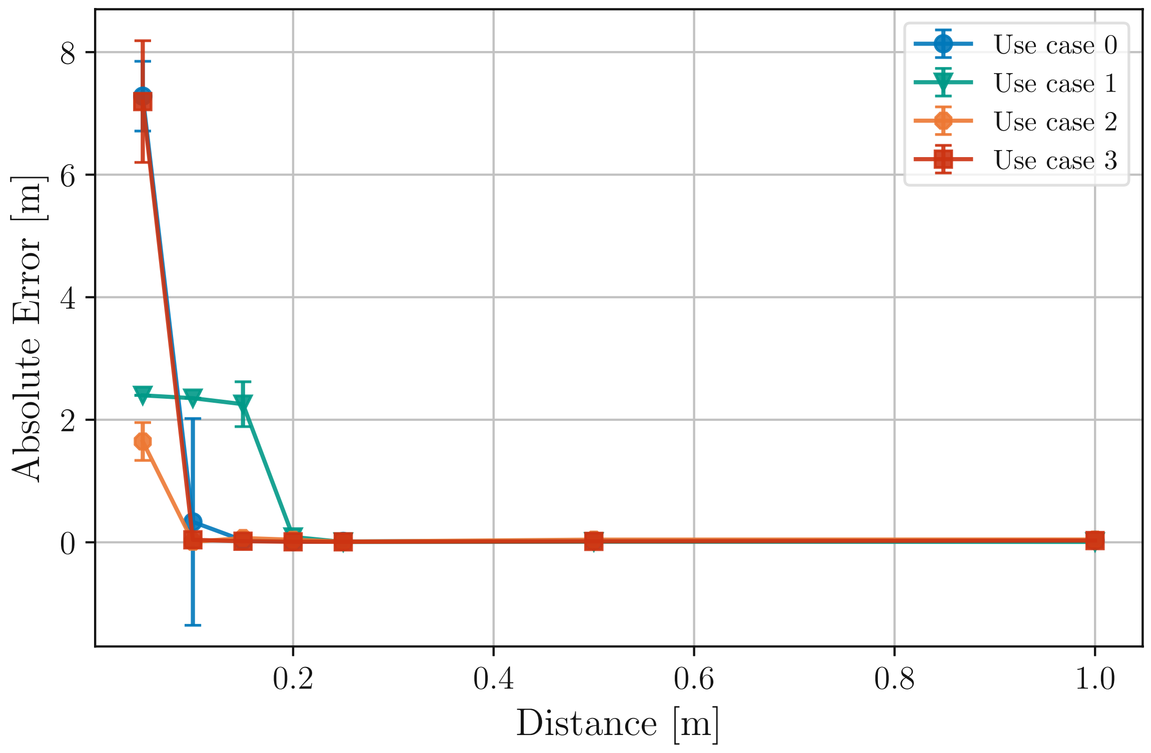
In this section, the results of the experiments are presented and analyzed. At first, the results of the static sensor analysis are presented, followed by the results on the accuracy of the 3Di sensor in SLAM applications.

5.2.1 Static Sensor Analysis

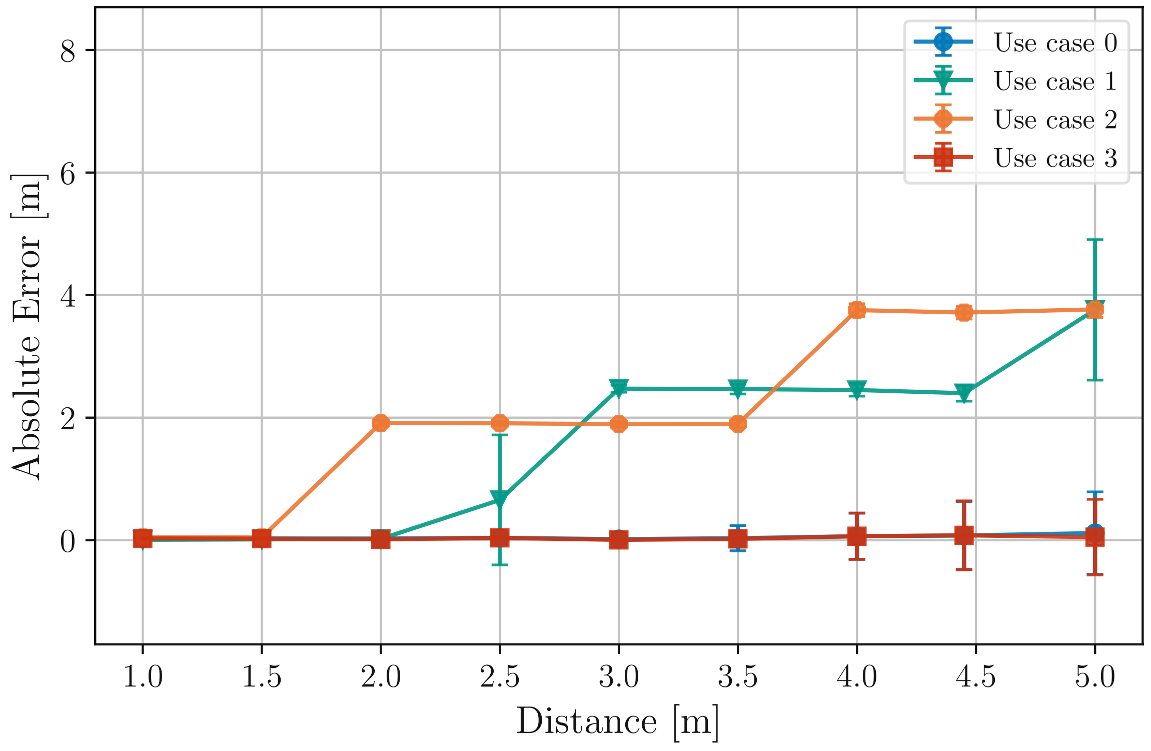
In Figure 5.4a and Figure 5.4b the trueness μ of the sensor is illustrated for the four different use cases measured in a room with closed blinds. The x-axis represents the measurement distances in meters, while the y-axis displays the trueness, also called absolute error, in meters. The data is displayed as an error bar, where each data point represents the trueness at a specific position, and the bars represent the standard deviation of the measurement data at this position.

For the first measurement at distance 0.05 m, see Figure 5.4a, all use cases show an error in their maximum ambiguity range according to Table 3.1. Use case 1 and use case 3 show the minimum errors from distance 0.1 m on, while use case 2 has inaccuracies until a distance of 0.2 m with errors in the range of 2.45 m to 2.4 m. Use case 0 and use case 3, which have an unambiguous range of more than 7 m, show a trueness of 0.03 m over the distance from 0.15 m to 5 m. Nevertheless, in both use cases, use case 0 and use case 3, the standard deviation increases with larger distances reaching 0.67 m and 0.62 m respectively at a distance of 5 m between the surface and the camera, see Figure 5.4b. At distance 3.5 m use case 3 shows a standard deviation 68% smaller than the standard deviation of use case 0, and at 5 m the standard deviation is again 8% smaller compared to use case 0.

Use case 1, which has an unambiguous range at 2.5 m, shows a high trueness from



(a) Depth error for each use case over a distance from 0.05 m to 1 m



(b) Depth error for each use case over a distance from 1 m to 5 m

Figure 5.4: Depth error progression over the distance. For each use case the absolute mean error and the standard deviation is observed at distances from 0.05 m to 5 m.

0.25 m to 2 m. At 2.5 m the error rises to 0.66 m, and stagnates again at 2.5 m at the distance from 3 m to 4.5 m, see Figure 5.4b. While the standard deviation in the measurements remains zero for the whole distance, these two data points, where the measurement distance between the surface and the camera is a multiply of the unambiguous range, the standard deviation is at 1.1 m. Use case 2 has a calculated unambiguous range of 1.9 m. As soon as the unambiguous range is exceeded the error rises to 1.9 m. At 4 m, when the unambiguous range reaches twice the distance, the error rises to 3.8 m.

The mean trueness of the sensor demonstrated for different lighting conditions and each designated use case is shown in Figure 5.5. The x-axis is divided in three sections to represent the different scenarios: closed blinds, office lighting, and daylight. The first section of the figure presents the results obtained in a darkened room with closed blinds, the middle section presents the results under office lighting conditions, and the final section on the right presents the results when the sensor is utilized in daylight conditions. The y-axis displays the mean trueness in meters with values ranging from −0.1 m to 0.1 m. The data is visualized using box plots, where the horizontal line represents the median error, and the box shows the interquartile range (IQR), covering the middle 50% of data points. Whiskers extend up to 1.5 times the IQR, with values beyond this range considered outliers and excluded from the plot.

In the context of use case 0 in Figure 5.5, measurements at distances ranging from 0.1 m to 5 m are regarded. The median error demonstrated stability at −0.024 m, consistent under both closed blinds and office lighting conditions. However, in daylight conditions, the error increased to −0.043 m. The uniformity of the measurement spread remained consistent across all lighting conditions, as evidenced by the consistent dimensions of the box plots with an IQR of 0.03 m.

Furthermore, Figure 5.5 displays the trueness of use case 1, where measurements in a range from 0.2 m to 2.0 m are considered. The absolute median error measured in the scenarios with closed blinds and office lighting is 0.006 m. At daylight conditions the absolute median error rises to 0.014 m. This dataset demonstrates the highest degree of accuracy, exhibiting consistent performance across all lighting environments.

For use case 2, the measurement range is from 0.1 m to 1.5 m. Figure 5.5 indicates a high degree of measurement inaccuracy, especially in scenarios with closed blinds and office lighting. In these cases, 50% of all recorded errors fall within the range of −0.04 m to 0.04 m. However, in daylight conditions, the variability in measurements is reduced by 57%, yet the median error shifts downward to −0.058 m.

The data from use case 3 features measurements ranging from 0.1 m to 5 m, exhibiting a pattern analogous to that observed in the dataset of use case 0 in Figure 5.5. The median error remains constant at 0.015 m for both, closed blinds and office lighting conditions. However, in daylight exposure, the median error increases by 130%, while the variance in the measurement values remains constant, as demonstrated by the unaffected box size of 0.03 m.

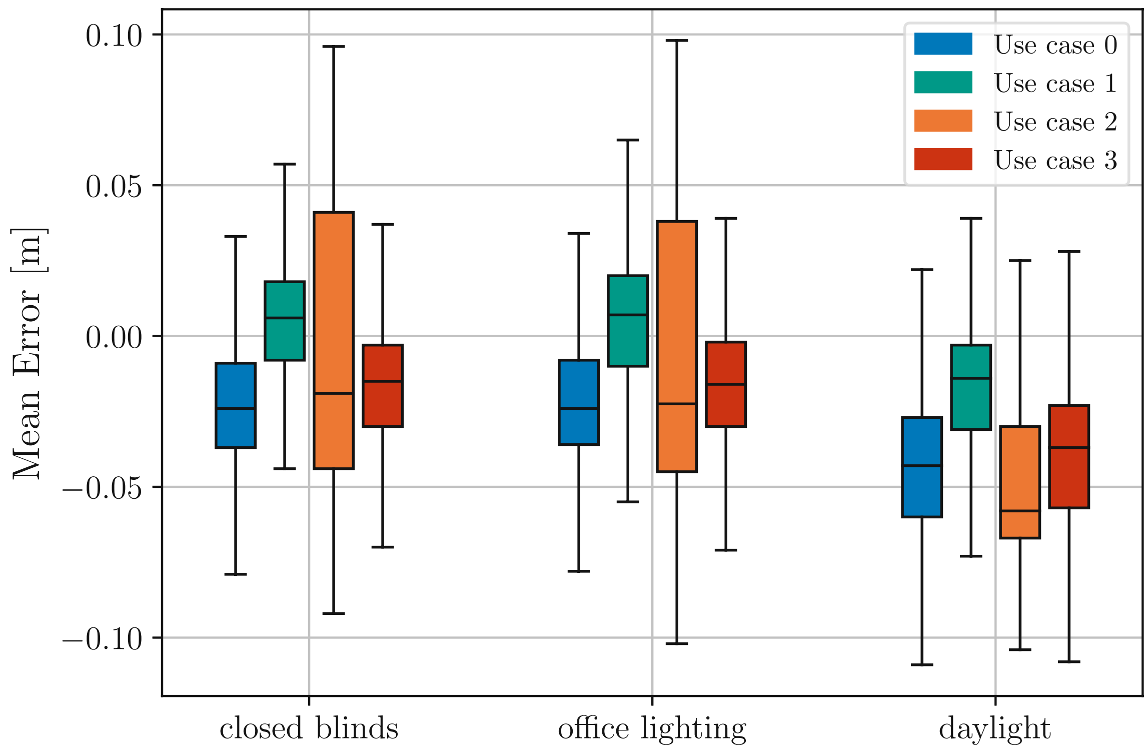


Figure 5.5: Mean depth error at three different light conditions, closed blinds, office lighting, and daylight, for each use case over its range

The mean trueness of the sensor varies across different surface materials for each use case, as shown in Figure 5.6. The x-axis is split into three sections to outline the different surface materials. The initial section of the figure depicts measurements taken on black fabric, the middle section corresponds to foam, and the last section displays results for glass. On the y-axis the mean error is represented with a range from −2.6 m to 0.8 m.

The results demonstrate similar behavior for both use cases 0 and 3, which utilize dual-frequency modulation for distance estimation. For black fabric and foam, 50% of the data points have an absolute trueness between 0.035 m and 0.88 m. In the scenario on glass, the median error increases to 0.24 m, with 50% of the data points falling between 0.07 m and 0.34 m. An error of 0.4 m suggests that the sensor detected the surface behind the glass rather than the glass itself, as this is the distance between the wall and the glass surface in the setup.

In the context of use case 1 in Figure 5.6, the median error on black fabric is observed to be −0.035 m, which is 45% lower than the median error of use case 0. However, the median error on foam for use case 1 is found to be twice that of use case 1, though the variance in the primary data points is reduced by 48%. When measuring glass, a median error of 0.18 m is observed. In this scenario, 50% of the data points around the median exhibit an error range from −1.99 m to 0.34 m, representing the widest range of deviations among all use cases.

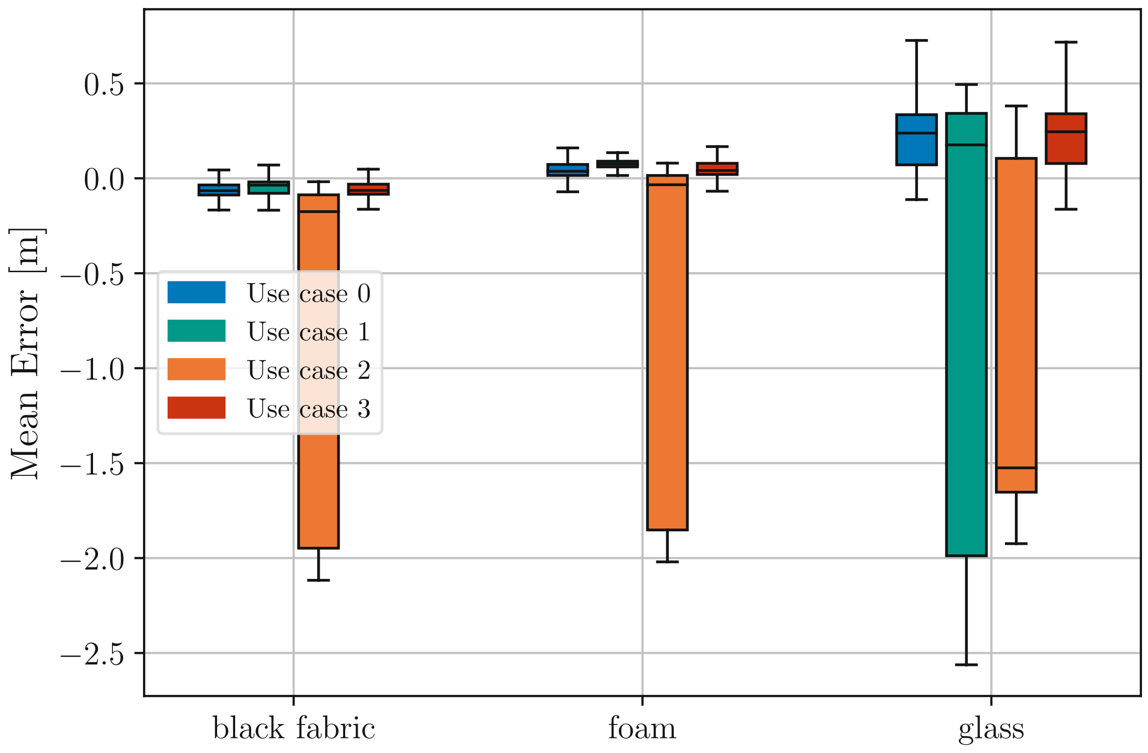


Figure 5.6: Mean depth error on surface of black fabric, foam, and glass material for each use case over its range

Use case 2 proves an error range independent of the targeted surface material, as can be seen in Figure 5.6. Across all tested materials, the measurement error spans a range greater than 2.2 m. The median error is −1.9 m for black fabric, −1.8 m for foam, and −1.55 m for glass.

5.2.2 Accuracy and Repeatability in SLAM

Figure 5.7 shows the error distribution of the sensor under three different lighting conditions, daylight, closed blinds, and office lighting. The data is visualized in box plots, where red boxes represent the data generated with the 3Di camera as used sensor and blue represents the data from the LiDAR sensor. The y-axis shows the mean error for all datasets. The axis spans an error range from −3.5 mm to 0.5 mm. In addition to the accuracy data, the repeatability of all scenarios is displayed in Table 5.1.

The camera shows identical accuracy under all tested light settings. The error ranges from −2.0 mm to −3.5 mm with an IQR from −2.5 mm to −3.0 mm.

The LiDAR performs worst in the dark room, where the measured errors range from −1.5 mm to 0.5 mm, while the error at office lighting and during daylight conditions is exactly −0.5 mm for a minimum of 50% of the data points over all created maps. Therefore, no whiskers appear in the box plot. All errors unequal to −0.5 mm are

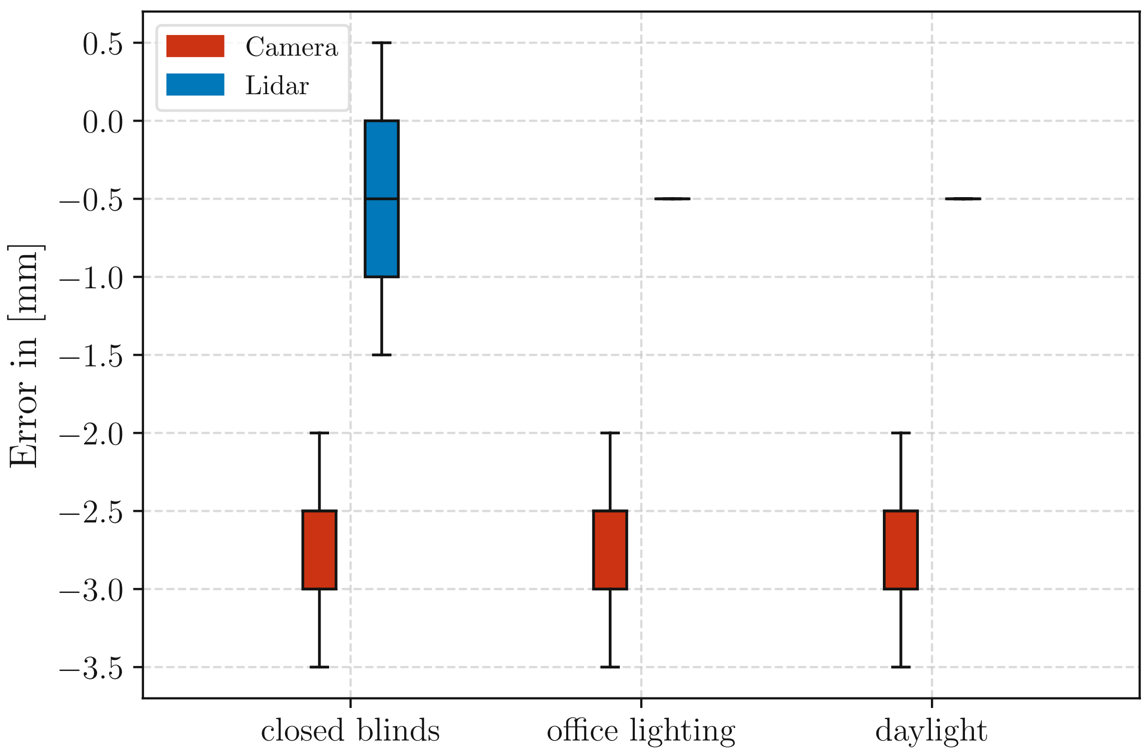


Figure 5.7: Comparison of the mapping error of the 3Di camera and the LiDAR sensor in SLAM at three different light conditions, closed blinds, office lighting, and daylight

Table 5.1: Comparison of the mapping repeatability of the 3Di camera and the LiDAR sensor in SLAM at different light conditions

	$\sigma_{\text{closed blinds}}$ [mm]	$\sigma_{\text{office lighting}}$ [mm]	σ_{daylight} [mm]
3Di Camera	0.0501	0.0887	0.0532
LiDAR	0.0289	0.0325	0.0376

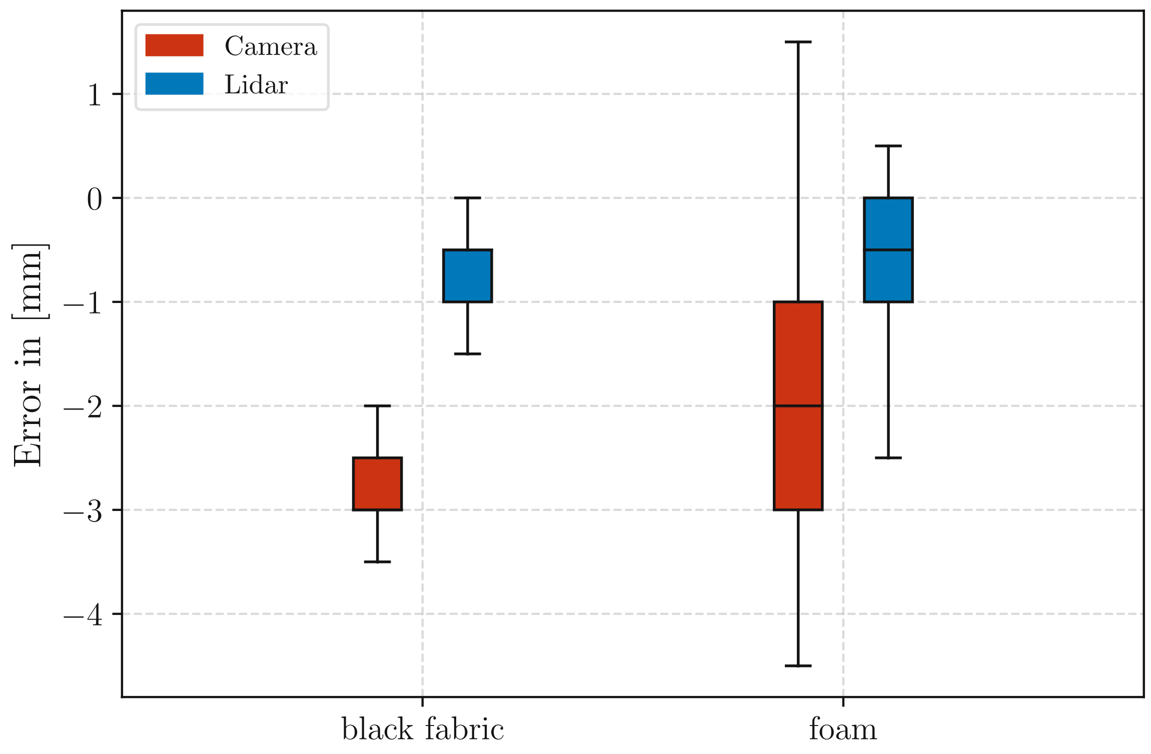


Figure 5.8: Comparison of the mapping error of the 3Di camera and the LiDAR sensor in SLAM on scenes with different surface materials involved

considered outliers.

Additionally, over all three experiments the repeatability of LiDAR data is 95% better than the repeatability of camera data. According to Table 5.1, the worst repeatability in the test data appears in the camera data from the tests at office lighting, while the LiDAR data from the tests with closed blinds show the best repeatability values. The highest discrepancy in repeatability is with office lighting conditions, where the repeatability value of the camera is 2.73 times higher than the repeatability of LiDAR data.

The test results for the accuracy on different surface materials are displayed in Figure 5.8. The data is again displayed in box plots. The y-axis represents the mean error reaching from −4.5 mm to 1.5 mm. The x-axis is divided in two sections, each representing one material, black fabric on the left, foam on the right. As the tests on glass exhibit a median depth error of minimum 0.24 m in the static analysis, it will not be further investigated on accuracy in SLAM context. The repeatability for each case is presented in Table 5.2.

The camera accuracy in SLAM mapping with different materials is displayed in red. The accuracy of the camera data with black fabric involved in the scene delivers exactly the same result as with different light conditions. For the tests with foam included in the scene, the median error decreased from −2.5 mm to −2.0 mm, while the accuracy

Table 5.2: Comparison of the mapping repeatability of the 3Di camera and the LiDAR sensor in SLAM with different surface materials involved in the scene

	$\sigma_{\text{black fabric}}$ [mm]	σ_{foam} [mm]
3Di Camera	0.0789	0.0800
LiDAR	0.0216	0.1400

shows a four time higher variance of 2 mm. Moreover, the repeatability of the measured datasets is 10% smaller than the dataset with office lighting conditions, which represents the worst repeatability of the camera data.

The LiDAR data increased their IQR for both scenarios over the measurements at different light conditions. With black fabric involved, the errors range from −1.5 mm to 0 mm. With foam material, the inaccuracy increased to errors from −2.5 mm to 0.5 mm. This scenario shows the least repeatable results, while the LiDAR data tested on black fabric shows the best repeatability.

5.2.3 Obstacle and Cliff Detection Accuracy

Figure 5.9 displays the map generated using the 3Di camera under test. The camera map is displayed in red, the ground truth is shown in green, and overlapping areas in a mix of both. As it is commonly done in occupancy grid maps, white represents free space and gray represents unknown areas. For orientation, the blue marker is displayed on the same positions as in Figure 5.3a. Additionally, the area of the stairs upward and downward is labeled.

In areas where the ground is not enclosed by stairs, the camera detects the barriers at 0.05 m to 0.1 m closer than the ground truth distance. The highest inaccuracies are observed in the corner regions. For the cliff area, where the stairs lead downward, the detection error remains within ±0.05 m.

In Figure 5.10 the map result measured with the LiDAR sensor is shown. LiDAR data is displayed in blue, the ground truth is green and overlapping areas are in a mix of both. Again, white represents free space and gray represents unknown areas. For orientation, the blue marker is displayed on the same positions as in Figure 5.3a and the area of the stairs upward and downward is labeled.

In comparison to the camera data in Figure 5.9, it can be seen that SLAM fed with LiDAR data cannot detect the stairs downwards as barrier. The sensor detects the wall over 6 m ahead and measures the stair railing to the left and to the right. Additionally, the map generated from LiDAR data shows higher inaccuracies in the corners.

These maps lead to the trueness values for each scenario, listed in Table 5.3. The map built from camera data achieves 24.1% higher accuracy with obstacle detection in

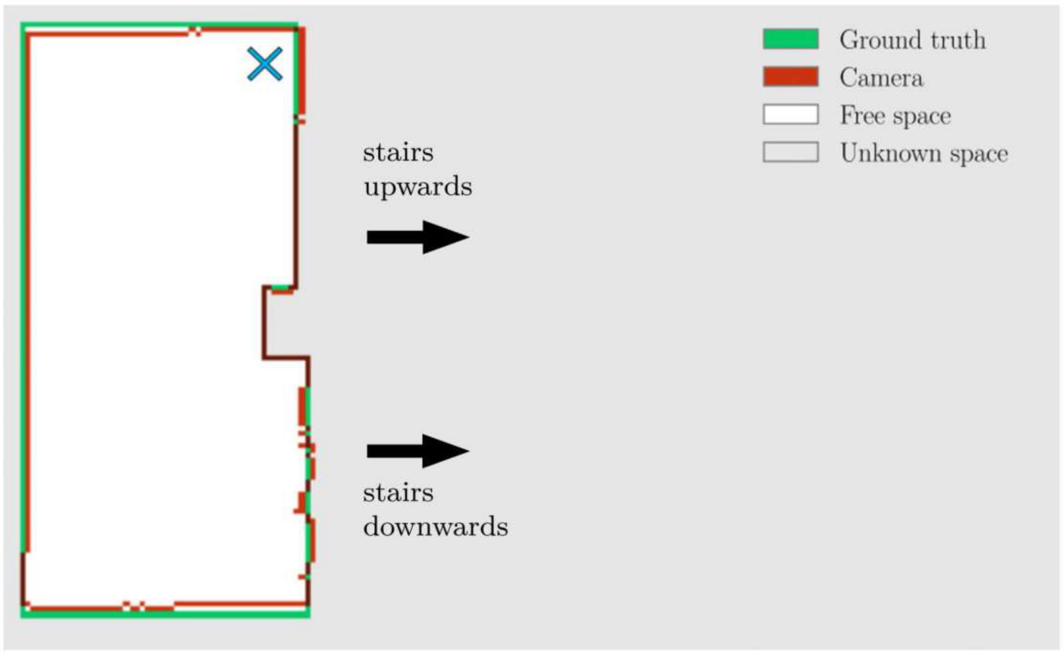


Figure 5.9: Comparison of the map generated using 3Di camera data (red) compared to the ground truth obstacles (green)

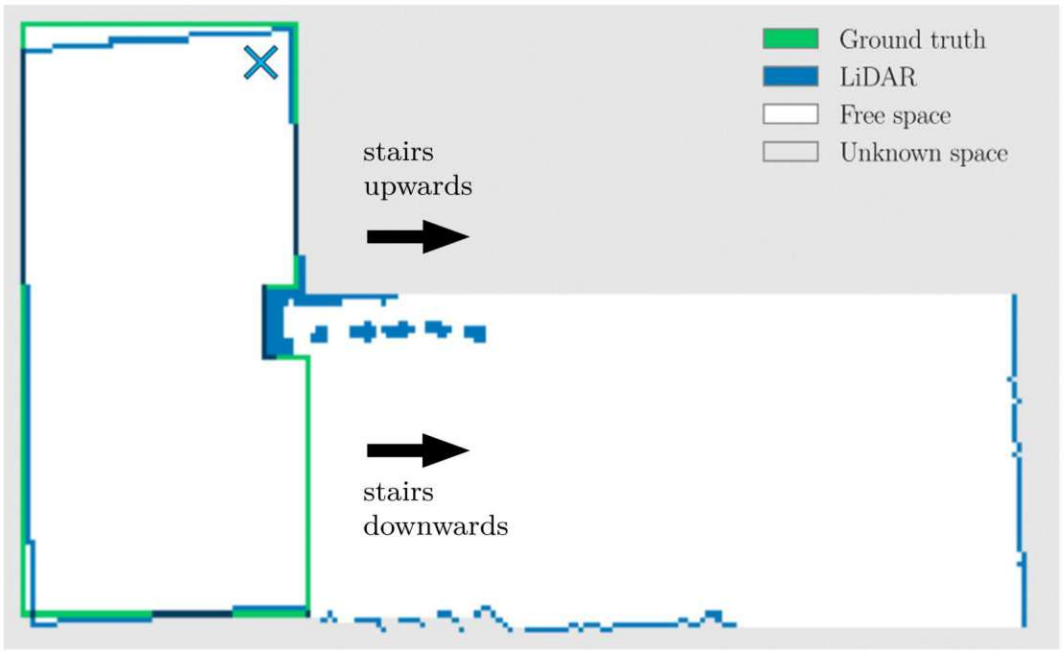


Figure 5.10: Comparison of the map generated using LiDAR data (blue) compared to the ground truth obstacles (green)

Table 5.3: Comparison of the mapping trueness of SLAM in a practice context with 3Di camera data and LiDAR data

	3Di Camera	LiDAR
trueness μ for obstacle detection [m]	0.063	0.083
trueness μ for cliff detection [m]	0.052	6.250
total trueness μ_{total} [m]	0.062	1.478

this scenario. For cliff detection the error within LiDAR data is 6.35 m, which, as seen in Figure 5.10, is because the LiDAR scanner detects the wall ahead while ignoring the level-change in the floor plane. The 3Di camera shows a trueness of 0.052 m. Summing this up in a weighted total trueness, the camera has a accuracy 95.8% better than the LiDAR.

5.3 Discussion

In this section, the previous results are discussed, outlining the suitable sensor settings for the Infineon 2877C iToF camera for SLAM applications and the achievable accuracy.

5.3.1 Sensor Accuracy

While the used 3Di camera shows high accuracy over all use cases and distances, some data points in use case 1 stand out. Not only that the minimum range for this use case is double to the minimum measurable range of the other use cases, additionally the error increases at distance 2.5 m and 5 m. As this is very close to the sensors maximum unambiguously measurable distance of 2.49 m, according to Table 3.1, it is likely that the sensor receives ambiguous signals. For some samples, the reflection represented the beginning of the wave, which leads to measurements close to zero, while for other samples the reflection represented the end of one wave over the wavelength resulting in distance measurements close to 2.49 m.

Analyzing Figure 5.4a and Figure 5.4b over the distances allows a more precise use case specification. The minimal distance is added to the specifications in Table 3.1 resulting in Table 5.4.

In order to determine the most suitable sensor for different lighting conditions and materials, the mean absolute median error for the 3Di camera and the LiDAR is listed in Table 5.5. This value serves as a coefficient for robustness against variations in lighting and material properties. This table shows use case 1 to achieve the highest trueness, followed by use case 3 and use case 0.

A comparative analysis of the use cases reveals that use case 0 and use case 3 are

Table 5.4: Adapted use case specification of Infineon iToF sensor 2877C

Parameter	**use case 0**	**use case 1**	**use case 2**	**use case 3**
$f_{mod,1}$ [MHz]	80.32	60.24	80.32	80.32
$f_{mod,2}$ [MHz]	60.24	-	-	60.24
$f_{mod,dual}$ [MHz]	20.08	-	-	20.08
Max Distance [m]	7.47	2.49	1.87	7.47
Min Distance [m]	**0.10**	**0.20**	**0.10**	**0.10**
Max Exposure [μs]	1200	1000	1000	640

Table 5.5: Results on trueness of Infineon iToF sensor 2877C

	use case 0	**use case 1**	**use case 2**	**use case 3**
trueness μ under different light conditions [m]	0.0303	0.0090	0.0332	0.0227
trueness μ on different materials [m]	0.1133	0.0957	0.5782	0.1167
trueness μ on different materials without glass [m]	0.0510	0.0555	0.1047	0.0525
mean overall robustness [m]	0.0465	0.0282	0.0690	0.0376

the most suitable for SLAM applications. However, in scenarios where the presence of an obstacle is known to be within a short range of maximum 2.5 m, use case 1 can be employed to obtain more reliable data, as it has been shown to demonstrate the best performance under varying lighting conditions and with different materials.

5.3.2 Accuracy and Repeatability in SLAM

For the measurements with foam, both, the LiDAR and the 3Di camera exhibited their least performance. Moreover, while all other measurements predominantly show negative errors, the measurements on foam include errors closer to zero. This means that larger distances were measured and suggests that multipath interference problems occur with this material.

For all other scenarios except foam, the accuracy in SLAM shows no sensitivity to different light conditions or black fabric. Since the IQR box for camera data starts at -2.5 mm for all tests, this shows a constant offset. The offset can be explained by inaccuracies in the transformation from the camera position to the robot base. The same phenomenon is seen in LiDAR data with a shorter offset of -0.5 mm.

However, the repeatability suggests that a higher sample number for camera data would be necessary to allow more accurate conclusions.

5.3.3 Obstacle and Cliff Detection Accuracy

The findings of the experiments regarding the obstacle and cliff detection accuracy demonstrate significant discrepancies between camera-based and LiDAR-based mapping systems, particularly with regard to their capability for cliff detection.

The findings demonstrate that in environments where floor structures are unknown, the camera-based approach exhibits significantly superior performance. The camera is able to detect cliffs 0.6 m preemptively, while the LiDAR does not detect them at all. Although this is traced back to the working principle of this state-of-the-art LiDAR, this shows the benefit of using a 3Di camera for SLAM applications. For such cases, the Turtlebot is equipped with a cliff sensor, which is mounted in the front area of the robot pointing downward to check for cliffs and stops driving as soon as a cliff is detected. Although these sensors detect cliffs only at 20 mm in advance, the camera setup detects them 20 times earlier, thereby improving predictive planning methods.

The second observation concerns measurement inaccuracies in corners. This phenomenon is not detected in the previous experiment regarding the map accuracy and repeatability. A fundamental distinction between the present experiment and the previous lies in the manner in which the robot traverses the environment. In the previous experiment the robot remained on the same position only executing a 360° rotation. However, the incorporation of continuous movement in the present experiment introduces novel sources of errors that may be attributed to data overload and inaccuracies in odometry. Furthermore, the misalignment of sensors due to movement, in combination with delays in data processing, has the potential to amplify the observed inaccuracies, particularly in areas characterized by corners. The absence of scan matching in this configuration precludes the possibility of correcting odometry drift, thereby propagating errors in localization. This finding indicates that the SLAM algorithm, when fed with LiDAR data, exhibits a higher reliance on scan matching as with camera data.

5.4 Summary of Results

The analysis of static sensor data indicates that use case 3 is the most suitable for SLAM applications. Use case 3 and use case 0 exhibit the widest measurement range, while the mean error in use case 3 is 0.038 m, 20% less than with use case 0. Use case 1 can be considered for short-range measurements up to 2.5 m, as it demonstrates the best trueness of 0.028 m under varying lighting conditions and different materials, further decreasing the error by 25%.

Errors observed in SLAM accuracy tests differ from those seen in static sensor tests. In static tests, inaccuracies are primarily attributed to variations in lighting conditions and material properties. However, during the SLAM accuracy tests, additional errors emerge and which can be attributed to factors such as transformation inaccuracies between the sensor and the robot base, or delays in data processing. These issues contribute to greater inaccuracies for the 3Di camera and the LiDAR sensor, especially when scanning larger environments.

The accuracy of mapping is known to increase with larger environments, which require greater robot movement for scanning. If the robot remains stationary and performs a 360° rotation, errors for obstacles scanned with the 3Di camera are at a mean of $-2.5\,\text{mm}$, and at $-0.5\,\text{mm}$ with the LiDAR scanner. However, in practical movement tests, where the robot navigates the environment, errors increase due to accumulation of odometry drift and the absence of scan matching. Under consideration of the different map resolutions, the error increases from the first to the second test by at least 26% for the camera data, and by at least 66% for the LiDAR tests. This is particularly evident in corner areas, where movement amplifies misalignment and data overload further affects accuracy. The LiDAR-based SLAM system typically relies heavily on scan matching. Conversely, the LiDAR-based SLAM system supplied with camera data is unable to perform scan matching effectively due to its constrainted field of view.

In the context of cliff identification, the approach with the camera as sensor has been demonstrated to be significantly more effective than the LiDAR scanner. The camera detects cliffs preemptively from a maximal distance of 0.6 m, while LiDAR does not detect the cliff at all. As a fallback technology for LiDAR sensors, the Turtlebot is equipped with a front-mounted cliff sensor that detects cliffs 20 mm in advance. However, this fallback technology does not contribute to preemptive navigation and path planning.

With the employed SLAM settings, LiDAR offers enhanced accuracy for scanning small environments with minimal robot movement involved in the scanning process. However, the 3Di camera's capacity to preemptively detect hazardous areas, such as cliffs, justifies its utilization in SLAM applications, thus enhancing navigation safety.

CHAPTER 6

Conclusion & Outlook

This thesis evaluates the feasibility of iToF sensors for unmanned ground vehicle (UGV) applications. The primary objective is to utilize the iToF camera as the sole sensor for SLAM, exploiting its ability to provide direct depth measurements while maintaining a compact and cost-effective design. A review of state-of-the-art methods highlights key challenges in SLAM: vision-based algorithms are affordable but struggle with lighting variations and scale estimation, whereas LiDAR-based SLAM offers high accuracy at a higher cost. This work aims to bridge the gap between these approaches by integrating iToF sensor data into a LiDAR-based SLAM algorithm.

The proposed setup for the integration approach is a Turtlebot 4 Lite mobile robot platform combined with an Infineon 3Di depth sensor. The RPLIDAR A1M8 is used as a comparison technology. The used SLAM algorithm is the ROS2 SLAM Toolbox. To prepare the depth image from the Infineon sensor for the LiDAR-based SLAM algorithm, processing steps filter the image to keep only the obstacles relevant for collision avoidance. Additionally, a dedicated processing step analyzes the iToF data to detect cliffs and prevent drop-offs, enabling the system to identify cliffs in advance, an ability typically not possible with conventional 2D LiDAR-based SLAM approaches that rely solely on LiDAR data.

With this setup, various experiments are performed to test the 3Di camera in SLAM applications, with a special focus on the accuracy under different light conditions and on different surface materials. With the gained information, the following statements regarding the research questions are made.

6.1 Research Questions

With the known challenges in SLAM applications the first research question reads as follows:

Research Question 1

Is it feasible to run SLAM algorithms for mobile robot platforms using solely the depth information of a ToF camera?

Using an indirect ToF camera as a single sensor in SLAM applications is feasible, but reliable odometry information about the robot is essential. As a 2D LiDAR-based SLAM approach is deployed in the proposed integration approach, the narrow horizontal FOV of the iToF camera limits the amount of information available to the SLAM algorithm after the transformation from 3D to 2D space. This makes the processed iToF data alone insufficient for the odometry correction. However, iToF camera data can be modified for use in LiDAR-based SLAM in general. The thesis suggests that a wider horizontal camera FOV is required to accurately perform SLAM functions additional to environment mapping, such as feature matching and loop closure, and thus correct the robot's odometry.

In order to evaluate the suitability of an iToF camera for SLAM, the reliability of depth measurements under various environmental conditions is crucial. IToF cameras depend on modulated light signals, which implies that their performance may be influenced by lighting conditions and the properties of the surface materials. This leads to the second research question:

Research Question 2

Is the chosen ToF camera more robust against different light conditions and materials in SLAM applications compared to an UGV setup equipped with a state-of-the-art LiDAR sensor?

The used iToF camera shows consistent behavior under different lighting conditions in both stand-alone and SLAM applications, but cannot match the performance of LiDAR data. It effectively handles light absorbing materials such as black fabric. However, with uneven materials the results do not match the behavior observed in the other scenarios, since multipath interference problems occur. Additionally, while LiDAR can detect objects behind glass, the iToF camera provides various ambiguous depth measurements on glass surfaces. Nevertheless, with the implemented algorithm, the camera is capable of identifying glass by detecting the transition where it meets another visible material, as it analyzes the entire 3D scene within the FOV. Conversely, the LiDAR system is only capable of detecting obstacles that are precisely at its mounting height.

6.2 Outlook

Although the system can be used to build environment maps with SLAM, further improvements could enhance robot pose estimation and overall mapping accuracy.

One potential enhancement to consider is the utilization of multiple iToF sensors mounted to achieve a wider horizontal FOV. This approach could facilitate scan matching in a manner conventional to LiDAR-based SLAM, thereby enhancing odometry correction and data association. Alternatively, the implementation of a scan matching algorithm during the preprocessing stage, incorporating a comparison between depth images, could increase the accuracy while preserving the benefits of LiDAR-based SLAM.

Alternatively, if the key advantages of LiDAR-based SLAM, such as fast processing and occupancy grid mapping, are not a priority, the iToF sensor can be investigated for visual-based SLAM. Typically, RGB-D cameras are used in this approach, and further analysis is needed to determine whether iToF data is sufficient for similar applications.

Bibliography

[1] J. Pistorius, *Industrie 4.0 – Schlüsseltechnologien für die Produktion: Grundlagen • Potenziale • Anwendungen*. Berlin, Heidelberg: Springer, 2020.

[2] O. Pavlovskyi, "Walking small unmanned vehicles," in *Advanced System Development Technologies II*, M. Bezuglyi, N. Bouraou, V. Mykytenko, G. Tymchyk, and A. Zaporozhets, Eds., Cham: Springer Nature Switzerland, 2025, pp. 393–429.

[3] Mordechai Ben-Ari and F. Mondada, *Elements of Robotics | SpringerLink*. Springer, 2018.

[4] T. Umetani, Y. Kondo, and T. Tokuda, "Rapid development of a mobile robot for the nakanoshima challenge using a robot for intelligent environments," *Journal of Robotics and Mechatronics*, vol. 32, pp. 1211–1218, Dec. 20, 2020.

[5] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics* (Advanced Textbooks in Control and Signal Processing), red. by M. J. Grimble and M. A. Johnson. London: Springer London, 2009.

[6] L. Jaulin, *Mobile Robotics*. Elsevier, Jan. 1, 2015.

[7] J. Huang, S. Junginger, H. Liu, and K. Thurow, "Indoor positioning systems of mobile robots: A review," *Robotics*, vol. 12, no. 2, p. 47, Apr. 2023, Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[8] H. Wang *et al.*, "Mobile robot indoor positioning system based on k-ELM," *Journal of Sensors*, vol. 2019, e7547648, Feb. 14, 2019, Publisher: Hindawi.

[9] A. B. S. H. M. Saman and A. H. Lotfy, "An implementation of SLAM with extended kalman filter," in *2016 6th International Conference on Intelligent and Advanced Systems (ICIAS)*, Aug. 2016, pp. 1–4.

[10] C. Bamji *et al.*, "A review of indirect time-of-flight technologies," *IEEE Transactions on Electron Devices*, vol. 69, no. 6, pp. 2779–2793, Jun. 2022, Conference Name: IEEE Transactions on Electron Devices.

[11] S. Giancola, M. Valenti, and R. Sala, *A Survey on 3D Cameras: Metrological Comparison of Time-of-Flight, Structured-Light and Active Stereoscopy Technologies* (SpringerBriefs in Computer Science). Cham: Springer International Publishing, 2018.

[12] P. Zanuttigh, G. Marin, C. Dal Mutto, F. Dominio, L. Minto, and G. M. Cortelazzo, *Time-Of-Flight and Structured Light Depth Cameras: Technology and Applications*, 1st ed. Cham: Springer International Publishing AG, 2016.

[13] J. Heikkila and O. Silven, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, ISSN: 1063-6919, Jun. 1997, pp. 1106–1112.

[14] E. Hering and G. Schönfelder, *Sensors in Science and Technology : Functionality and Application Areas*. Springer, 2022.

[15] F. Piron, D. Morrison, M. R. Yuce, and J.-M. Redouté, "A review of single-photon avalanche diode time-of-flight imaging sensor arrays," *IEEE Sensors Journal*, vol. 21, no. 11, pp. 12 654–12 666, Jun. 2021, Conference Name: IEEE Sensors Journal.

[16] S. Gokturk, H. Yalcin, and C. Bamji, "A time-of-flight depth sensor - system description, issues and solutions," in *2004 Conference on Computer Vision and Pattern Recognition Workshop*, Jun. 2004, pp. 35–35.

[17] S. Foix, G. Alenya, and C. Torras, "Lock-in time-of-flight (ToF) cameras: A survey," *IEEE Sensors Journal*, vol. 11, no. 9, pp. 1917–1926, Sep. 2011, Conference Name: IEEE Sensors Journal.

[18] L. Li, "Time-of-flight camera - an introduction," 2014.

[19] X.-L. Jin and S.-Q. Zeng, "Design of 3d TOF camera system based on CW modulation technique," in *2016 13th IEEE International Conference on Solid-State and Integrated Circuit Technology (ICSICT)*, Oct. 2016, pp. 373–375.

[20] A. Payne, "Multiple frequency range imaging to remove measurement ambiguity,"

[21] J. Wang, P. Liu, F. Wen, R. Ying, and W. Wang, "Phase unwrapping for time-of-flight sensor based on image segmentation," *IEEE Sensors Journal*, vol. 21, no. 19, pp. 21 600–21 611, Oct. 2021, Conference Name: IEEE Sensors Journal.

[22] A. P. P. Jongenelen, D. G. Bailey, A. D. Payne, A. A. Dorrington, and D. A. Carnegie, "Analysis of errors in ToF range imaging with dual-frequency modulation," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 5, pp. 1861–1868, May 2011, Conference Name: IEEE Transactions on Instrumentation and Measurement.

[23] M. Hansard, S. Lee, O. Choi, and R. Horaud, *Time of Flight Cameras: Principles, Methods, and Applications*. Springer, May 22, 2021.

[24] D. ISO, *Accuracy (trueness and precision) of measurement methods and results*, Nov. 1997.

[25] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard, “A tutorial on graph-based SLAM,” *IEEE Intelligent Transportation Systems Magazine*, vol. 2, no. 4, pp. 31–43, 2010, Conference Name: IEEE Intelligent Transportation Systems Magazine.

[26] B. Alsadik and S. Karam, “The simultaneous localization and mapping (SLAM)-an overview,” *Journal of Applied Science and Technology Trends*, vol. 2, no. 2, pp. 147–158, Nov. 2, 2021, Number: 02.

[27] C. Cadena *et al.*, “Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age,” *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, Dec. 2016, Conference Name: IEEE Transactions on Robotics.

[28] H. Durrant-Whyte and T. Bailey, “Simultaneous localization and mapping: Part i,” *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 99–110, Jun. 2006, Conference Name: IEEE Robotics & Automation Magazine.

[29] K. Murphy and S. Russell, “Rao-blackwellised particle filtering for dynamic bayesian networks,” in *Sequential Monte Carlo Methods in Practice*, A. Doucet, N. Freitas, and N. Gordon, Eds., New York, NY: Springer New York, 2001, pp. 499–515.

[30] C. Debeunne and D. Vivet, “A review of visual-LiDAR fusion based simultaneous localization and mapping,” *Sensors*, vol. 20, no. 7, p. 2068, Jan. 2020, Number: 7 Publisher: Multidisciplinary Digital Publishing Institute.

[31] M. Ferrera, A. Eudes, J. Moras, M. Sanfourche, and G. Le Besnerais, “OV^{2}SLAM: A fully online and versatile visual SLAM for real-time applications,” *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1399–1406, Apr. 2021, Conference Name: IEEE Robotics and Automation Letters.

[32] J. Zhang and S. Singh, “Low-drift and real-time lidar odometry and mapping,” *Autonomous Robots*, vol. 41, no. 2, pp. 401–416, Feb. 2017.

[33] A. Manar, I. Haouala, and A. Raisov, *Low cost mobile navigation using 2D-SLAM in complex environments*. Jul. 28, 2023.

[34] S. Macenski and I. Jambrecic, “SLAM toolbox: SLAM for the dynamic world,” *Journal of Open Source Software*, vol. 6, no. 61, p. 2783, May 13, 2021.

[35] S. Macenski, “On use of the SLAM toolbox,” 2019.

[36] Y. Maruyama, S. Kato, and T. Azumi, “Exploring the performance of ROS2,” in *Proceedings of the 13th International Conference on Embedded Software*, ser. EMSOFT ’16, New York, NY, USA: Association for Computing Machinery, 2016, pp. 1–10.

[37] G. Deng, G. Xu, Y. Zhou, T. Zhang, and Y. Liu, “On the (in)security of secure ROS2,” in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS ’22, New York, NY, USA: Association for Computing Machinery, Nov. 7, 2022, pp. 739–753.

[38] O. Robotics. “Understanding nodes — ROS 2 documentation: Humble documentation.” (), [Online]. Available: https://docs.ros.org/en/humble/Tutorials/Beginner-CLI-Tools/Understanding-ROS2-Nodes/Understanding-ROS2-Nodes.html (visited on 03/11/2025).

[39] S. Kugele, D. Hettler, and J. Peter, “Data-centric communication and containerization for future automotive software architectures,” in *2018 IEEE International Conference on Software Architecture (ICSA)*, Apr. 2018, pp. 65–6509.

[40] H. Teper, M. Günzel, N. Ueter, G. von der Brüggen, and J.-J. Chen, “End-to-end timing analysis in ROS2,” in *2022 IEEE Real-Time Systems Symposium (RTSS)*, ISSN: 2576-3172, Dec. 2022, pp. 53–65.

[41] I. T. AG, “REAL3 image sensor IRS2877a(s) - 3d time-of-flight single chip for automotive,” Jun. 2022.

[42] R. P. Ltd, “Raspberry pi 4 model b - product brief,” Apr. 2024.

[43] R. Automation. “User manual · turtlebot4 user manual.” (Feb. 8, 2022), [Online]. Available: https://turtlebot.github.io/turtlebot4-user-manual/ (visited on 03/14/2025).

[44] SLAMTEC. “RPLIDAR-a1 360°laser range scanner _ domestic laser range scanner.” (), [Online]. Available: https://www.slamtec.com/en/Lidar/A1 (visited on 03/14/2025).

[45] R. Automation. “Networking · user manual.” (Feb. 8, 2022), [Online]. Available: https://turtlebot.github.io/turtlebot4-user-manual/setup/networking.html (visited on 03/15/2025).

[46] R. Schnabel, R. Wahl, and R. Klein, “Efficient RANSAC for point-cloud shape detection,” *Computer Graphics Forum*, vol. 26, no. 2, pp. 214–226, Jun. 2007.

[47] ROS2. “Sensor_msgs/LaserScan documentation.” (), [Online]. Available: https://docs.ros.org/en/noetic/api/sensor_msgs/html/msg/LaserScan.html (visited on 03/21/2025).

Eigenständigkeitserklärung

Hiermit erkläre ich, dass die vorliegende Arbeit gemäß dem Code of Conduct, insbesondere ohne unzulässige Hilfe Dritter und ohne Benutzung anderer als der angegebenen Hilfsmittel, angefertigt wurde. Die aus anderen Quellen direkt oder indirekt übernommenen Daten und Konzepte sind unter Angabe der Quelle gekennzeichnet. Die Arbeit wurde bisher weder im In– noch im Ausland in gleicher oder in ähnlicher Form in anderen Prüfungsverfahren vorgelegt.

Datum	Karin Egretzberger