Mário Jorge Pinto Cristóvão

DESIGN OF A MULTI-SENSOR APPARATUS FOR FORESTRY ROBOTICS

A CASE STUDY ON FOREST 3D MAPPING

Dissertação no âmbito do Mestrado em Engenharia Física orientada pelo Doutor David Bina Siassipour Portugal e co-orientada pelo Engenheiro Afonso E. Carvalho e apresentada ao Departamento da Física da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Fevereiro de 2023

Design of a Multi-Sensor Apparatus for Forestry Robotics: A case study on Forest 3D Mapping

Supervisor:

Doutor David Portugal

Co-Supervisor:

Eng. Afonso E. Carvalho

Jury:

Prof. Doutor Rui Paulo Pinto da Rocha

Prof. Doutor Pedro Mariano Simões Neto

Dissertation submitted in partial fulfillment for the degree of Master of Science in Engineering Physics.

Coimbra, February 2023

Acknowledgments

The journey to complete this work was challenging, and the level of dedication and struggle involved cannot be fully captured in these words. Without the support of several individuals, I would not have been able to finish what I started back in 2016.

I am deeply grateful to my supervisor, Dr. David Portugal, whose guidance and support were essential to my success. I also wish to thank Eng. Afonso Carvalho, who, alongside Dr. Portugal, provided valuable feedback and insights that proved fundamental to the successful completion of this dissertation. Special thanks go to Duda Andrada, for helping me settle into ISR and for her willingness to offer assistance with any issue that arose. I am also especially grateful to José Faria, who has been my personal mentor and rubber duck throughout these years.

To all my friends, who are too numerous to name individually, thank you for studying, partying, and debating with me through the years. I cherish your friendship and would be lost without it.

To the friends I didn't choose, my family, thank you for your unwavering support and belief in me. Your presence and encouragement have meant more to me than words can express, and I could not have asked for a better family.

To everyone who has helped me in any way, thank you. Your assistance has earned my eternal gratitude, and I am here for you whenever you need me.

Resumo

Nas últimas décadas, temos observado um aumento na área florestal ardida, uma realidade que poderia ser melhorada com o aumento da manutenção florestal. A robótica tem potencial para contribuir para a resolução deste problema. No entanto, para obter as informações necessárias para desenvolver e testar algoritmos que permitam aos robôs ajudar na manutenção florestal nem sempre é uma tarefa simples.

Neste trabalho, apresenta-se uma solução para este problema na forma de um dispositivo multi-sensorial portátil capaz de adquirir conjuntos de dados relevantes para apoiar atividades florestais, como mapeamento 3D, identificação de material inflamável, planeamento de rotas e limpeza da floresta. O aparelho aproveita várias tecnologias, como *Laser Imaging, Detection and Ranging* (LiDAR), câmera de profundidade, uma câmera estéreo e uma *inertial measurement unit* (IMU) para obter informações sobre o meio envolvente. Adicionalmente, uma aplicação android é desenvolvida para adquirir informações do Sistema de Navegação Global por Satélite (GNSS) durante a gravação de um dataset. Para gravar todas as informações sensoriais, foi desenvolvida uma arquitetura modular em torno do *Robot Operating System* (ROS) e Docker.

O sistema desenvolvido foi validado comparando diferentes implementações estado da arte de localização e mapeamento simultâneo em ambiente florestal, conseguindo resultados que comprovam a robustez do sistema. O trabalho apresentado fornece uma base sólida para melhorias, com a capacidade de suportar futuros trabalhos no campo da robótica florestal.

Palavras-chave: Robótica florestal, Aquisição de dados, Localização e Mapeamento 3D, ROS

Abstract

In recent decades, we have seen an increase in the number of forest areas burned, a circumstance that could be mitigated by increasing forest maintenance. This problem could potentially be solved with the help of robotics. However, obtaining the necessary information to develop and test algorithms that allow robots to help in forest maintenance is not always a simple task.

In this work, a solution to this problem is presented in the form of a portable multi-sensor apparatus capable of acquiring datasets of relevant information to support forestry activities such as 3D metric-semantic mapping, flammable material identification, path planning, and forest clearing. The apparatus takes advantage of several technologies such as Laser Imaging, Detection and Ranging (LiDAR), a depth camera, a stereo camera and an inertial measurement unit (IMU) to obtain information about its surroundings. Additionally, an Android application is also developed to access Global Navigation Satellite System (GNSS) information when recording a dataset. In order to collect all the sensory information, a modular architecture was developed around the Robot Operating System (ROS) and Docker.

The proposed apparatus has been validated through demonstration of its ability to effectively collect the necessary data for simultaneous localization and mapping in a challenging forest environment. The work presented provides a solid foundation for improvement and supports future efforts in the field of forestry robotics.

Keywords: Forestry robotics, Data Acquisition, 3D Localization and Mapping, ROS

"Being challenged in life is inevitable, being defeated is optional."

Roger Crawford

Contents

List of Acronyms

ABS	Acrylonitrile Butadiene Styrene
APE	Absolute Position Error
BW	Bandwidth
CAD	Computer-aided Design
CPU	Central Processing Unit
DBH	Diameter at Breast Height
DEEC	Departamento de Engenharia Eletrotécnica e de Computadores
EKF	Extended Kalman Filter
eMMC	embedded MultiMediaCard
FoV	Field of View
GNSS	Global Navigation Satellite System
GPS	Global Position System
HDMI	High-Definition Multimedia Interface
ICP	Iterative Closest Point
IMU	Inertial Measurement Unit
LiDAR	Laser imaging, Detection, and Ranging
LOAM	LiDAR Odometry And Mapping
LIO-SAM	LiDAR Inertial Odometry via Smoothing and Mapping

MoSCoW	Must, Should, Could, Won't
MVVM	Model View View-Model
ORB	Oriented FAST and Rotated BRIEF
OS	Operating System
PCIe	Peripheral Component Interconnect express
PDF	Probability Dense Function
PTFE	Polytetrafluoroethylene
RAM	Random Access Memory
RGB	Red, Green, Blue
RGBD	RGB and Depth
RMSE	Root Mean Square Error
ROE	Relative Orientation Error
ROS	Robot Operating System
RTAB-Map	Real Time Appearance Based Mapping
RTE	Relative Translation Error
RPE	Relative Position Error
SLAM	Simultaneous Localization And Mapping
SSD	Solid State Drive
UKF	Unscented Kalman Filter
USB	Universal Serial Bus
UV	ultraviolet radiation
UWB	Ultra-Wideband

List of Figures

List of Tables

1 Introduction

Portugal has seen an increase in the area affected by wildfires since 1980, in contrast to other European countries [1, 2]. Effective forest maintenance is essential for preventing wildfires, and the use of mechanization in forestry has significantly increased productivity, leaving the human component as the bottleneck [3]. While the use of forestry robots is not yet widespread, several prototypes have been developed for tasks such as forest monitoring and preservation [4, 5, 6, 7], as well as planting, pruning, and harvesting [8, 9, 10]. These robots often require localization or mapping information to carry out their tasks, but their large size and lack of flexibility can make data acquisition slow.

To address this, the development of a lightweight and portable multi-sensory apparatus with an onboard computer for localization and mapping of forest environments is proposed. This apparatus combines multiple sensing technologies, such as 3D Laser imaging, Detection, and Ranging (LiDAR), depth cameras, infrared spectroscopy and Inertial Measurement Unit (IMU). The apparatus shall be used to collect datasets from forest environments, thus helping to acquire the necessary information to develop and advance the state of the art of several forestry operations, such as metric-semantic 3D mapping, flammable material identification, path planning and forest clearing. Aside from designing the apparatus, this work proposes to integrate, test, and then compare state-of-the-art 3D simultaneous localization and mapping techniques based on the Robot Operating System (ROS) middleware.

1.1 Objectives

The primary objectives of this work are:

- Developing a lightweight and portable multisensory apparatus with an onboard computer for localization and mapping of forest environments;

- Collecting datasets to support forestry robotics research, e.g. metric-semantic 3D mapping, combustible material identification, and forest cleaning;
- Integrating, testing, and comparing state-of-the-art 3D localization techniques based on the ROS middleware using the collected dataset;
- Evaluating the system's performance in localization and mapping in a forest environment;
- Summarizing the work's success and identifying lessons learned and potential improvements.

1.2 System Requirements

A Must, Should, Could, Won't (MoSCoW) analysis is used to clarify the importance of features and prioritize which features to implement in this work. The MoSCoW analysis for this work is presented in Figure 1.1.

Multiple sensors and cameras must be used to achieve the proposed goals. A Xsens Inertial Measurement Unit (IMU) will provide angular velocity and linear acceleration, while an Intel Realsense D435i RGB and Depth camera, which has an integrated IMU, will primarily provide RGB, depth, and infrared information. Both a Livox LiDAR and a Mynt Eye camera are used to extract 3D depth information from the surrounding environment.

1.3 Dissertation Structure

The document is structured as follows. In Chapter 2, essential background concepts and related works are reviewed. In Chapter 3, a detailed description of the design and implementation of the multi-sensor apparatus is provided, including the materials used and the rationale behind design choices. The different steps required to implement the complete system are also outlined. Chapter 4 details the preliminary tests performed as well as the experimental design of the dataset collected in a real-world forest environment. Chapter 5 examines the system's performance while mapping several forest environments.

Finally, in the last Chapter, the work's successes and key lessons learned are summarized, and opportunities for improvement are identified.

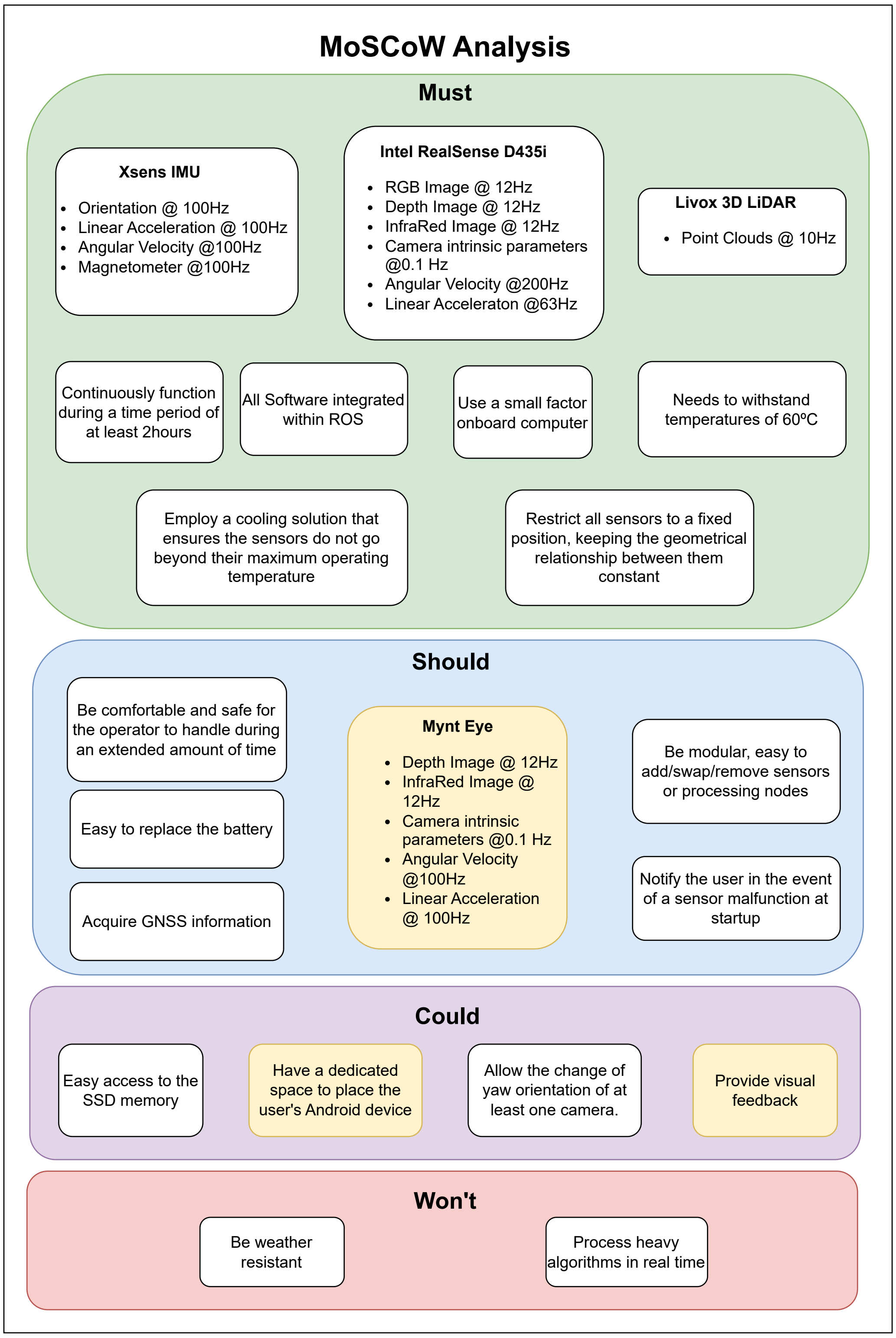

MoSCoW Analysis
Must
Xsens IMU
Orientation @ 100Hz
Linear Acceleration @ 100Hz
Angular Velocity @100Hz
Magnetometer @100Hz
Intel RealSense D435i
RGB Image @ 12Hz
Depth Image @ 12Hz
InfraRed Image @ 12Hz
Camera intrinsic parameters @0.1 Hz
Angular Velocity @200Hz
Linear Acceleraton @63Hz
Livox 3D LiDAR
Point Clouds @ 10Hz
Continuously function during a time period of at least 2hours
All Software integrated within ROS
Use a small factor onboard computer
Needs to withstand temperatures of 60ºC
Employ a cooling solution that ensures the sensors do not go beyond their maximum operating temperature
Restrict all sensors to a fixed position, keeping the geometrical relationship between them constant
Should
Be comfortable and safe for the operator to handle during an extended amount of time
Easy to replace the battery
Acquire GNSS information
Mynt Eye
Depth Image @ 12Hz
InfraRed Image @ 12Hz
Camera intrinsic parameters @0.1 Hz
Angular Velocity @100Hz
Linear Acceleration @ 100Hz
Be modular, easy to add/swap/remove sensors or processing nodes
Notify the user in the event of a sensor malfunction at startup
Could
Easy access to the SSD memory
Have a dedicated space to place the user's Android device
Allow the change of yaw orientation of at least one camera.
Provide visual feedback
Won't
Be weather resistant
Process heavy algorithms in real time

Figure 1.1: MoSCoW analysis of features to implement on the apparatus. Cards in yellow represent the requirements that were not achieved in this work.

2 Background and Related Work

From the beginning of civilization, mapping the surroundings has been a key concept for navigating an environment. In essence, the problem our ancestors faced is being tackled in Robotics in the last decade: How do we map the environment and know our location within it? This is a fundamental problem to tackle and of extreme importance to achieve full robot autonomy. In a simple approach, a depth sensor in a static platform is enough to represent the surroundings in a map, this is the base platform of the taxometry chart presented in Figure 2.1. At this level retrieving localization is trivial, but the amount of uses in real world are narrowed down to a few. The next level presents the localization component, which feeds information into the mapping thus supporting incremental mapping. If the localization information is correct, precise and error free this would be enough to generate an accurate map. However, this scenario is not realistic since every sensor measurement has an error associated with it. The more complex level presents a two way interaction between the mapping and localization module, this is often refered as Simultaneous Localization And Mapping (SLAM). SLAM is the challenge of mapping the local environment perceived by a moving entity (e.g. robot) and updating the map and localization simultaneously and continuously as the entity moves through space.

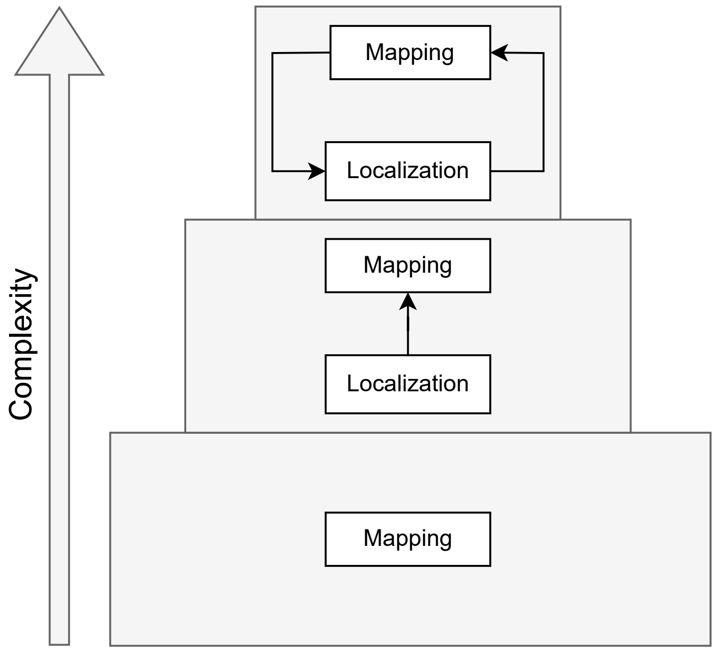
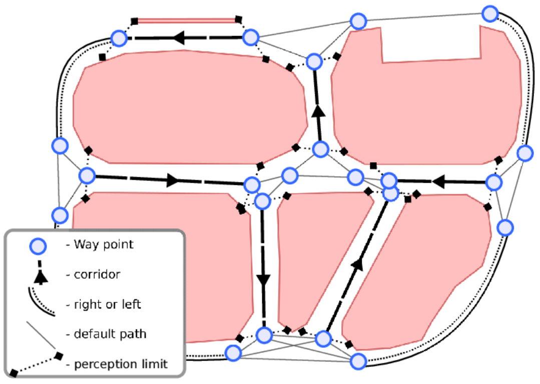


Figure 2.1: Mapping taxonomy graph.

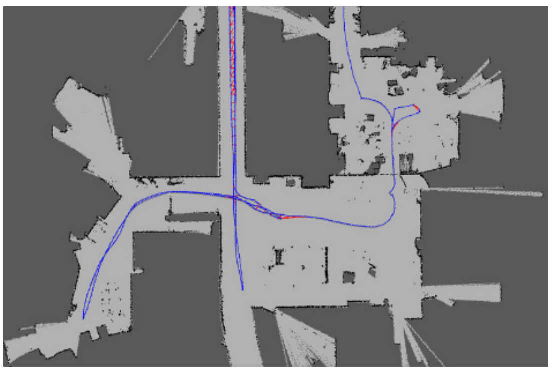
2.1 Mapping

Maps in robotics are usually represented either metrically or topologically [11], both representation are shown in Figure 2.2. The information on topological maps is presented in the form of a graph that contains important landmarks. There are many advantages to topological maps, such as space efficiency, but they are often more difficult to construct in large environments if sensory information is suboptimal and landmarks are hard to identify [11]. A metric map can be represented with raw sensor measurements, or occupancy grids, the latter being more common. In occupancy grids, each cell holds the probability that the space in that cell is occupied. As the cell value increases, the likelihood of that space being occupied also increases. The resolution of the map will be determined by the size of the grid, therefore to generate a precise map, one must have large dedicated memory space. While metric maps are easy to build and mantain, they tend to be space consuming [11]. To combat the large amount of space needed to store metric maps, some alternatives have been proposed in the literature.

Elevation maps, introduced in [16], are a more efficient way to display a 3D metric map by showing only the highest Z value in the grid, as shown in Figure 2.3. This means that any grid point below the highest Z value is unknown. For example, a wall and a bridge have the same representation in elevation mapping although it is possible to walk under



(a) Example of topological map [12].



(b) Example of a 2D metric map [13].

Figure 2.2: Topological vs metric map.

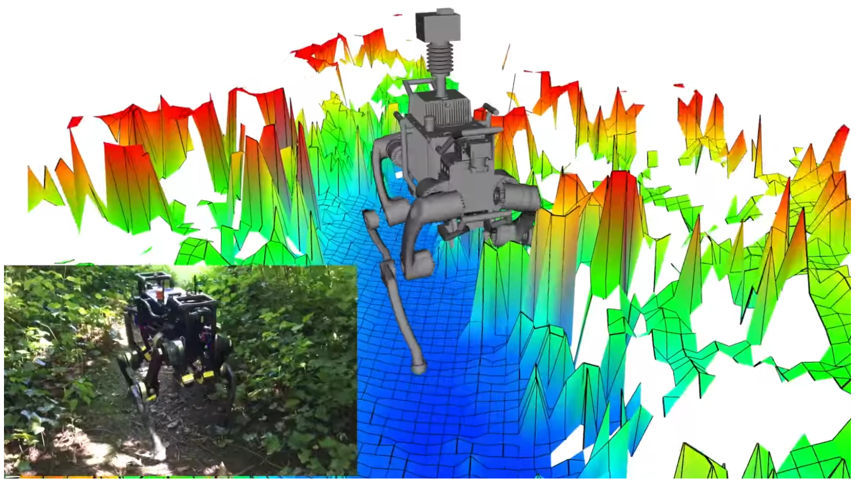


Figure 2.3: Robot mapping the environment with elevation map [14].

the bridge. Octomap is a popular 3D mapping framework to generate occupancy grids based on depth information [15] (see Figure 2.4). Octomap uses octrees[1] to help reduce the memory space of rendering a map since large spaces with equal values represented by a single parent node.

[1]An octree is a tree data struture in which each internal node is subdivided in eight children until the structure resolution is met.

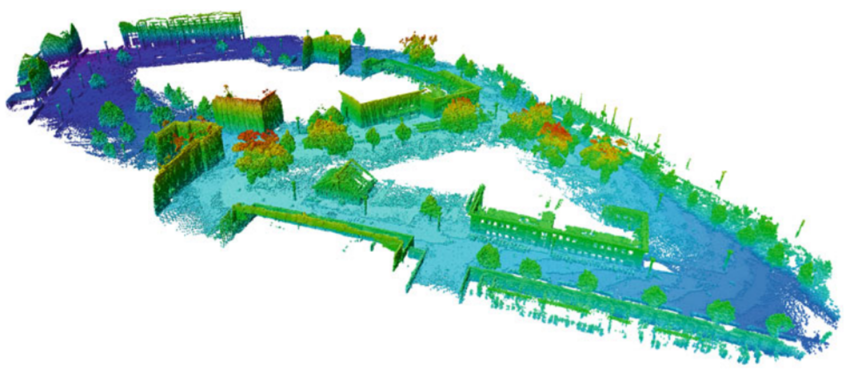


Figure 2.4: Resulting octree maps of an outdoor environments at 0.2 m resolution [15].

2.2 Localization

To fully determine the location of an object in 3D space at a given time, its 6 degrees of freedom (pose) must be known: x, y, z (position) and *roll*, *pitch*, *yaw* (orientation). Localization can take two forms: absolute or relative. On one hand, absolute localization estimates the pose of the agent in a global reference. New estimates provided by absolute localization can be viewed independently of previous poses estimates. Absolute localization can be provided by technologies such as Global Navigation Satellite System (GNSS) or Ultra-Wideband (UWB). On the other hand, relative localization establishes a relative relationship between the current pose and previous poses, usually determined by consecutive states.

The procedure of estimating a change in pose over time by using sensory information is called **odometry**. Classical odometry is computed from motion sensors such as IMU or wheel encoders, but it is also possible to use cameras as an input to acquire odometry, this is called visual odometry. Feature matching algorithms (identifying and relating the same features of an object from different perpectives) such as SURF [17] or ORB [18] are at the core of many visual odometry algorithms. Recently, Liu et al. [19] developed a Visual Odometry algorithm based on a Deep Learning technique.

It is also possible to use LiDAR to acquire odometry, whereas scan matching algorithms like Iterative Closest Point method (ICP) are often used. Scan matching takes two pointclouds or 2D LiDAR scans and finds the transformation that produces the least amount of error between points. Simply put, it tries to compute the transform that leads to the overlap of both scans, the transform is then applied to estimate the robots location,

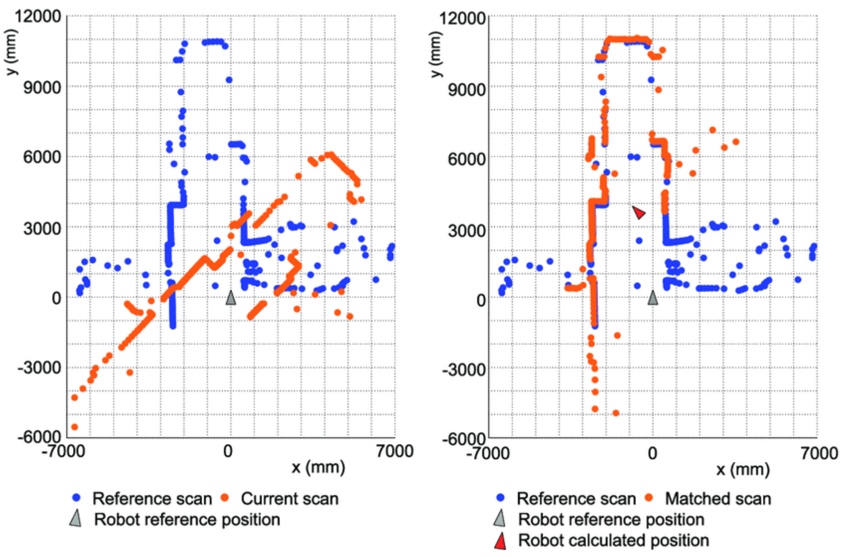


Figure 2.5: 2D scan matching to estimate a robot's pose [20].

as is shown in Figure 2.5. Both scan matching and feature matching can be used to find a transform to update the localization of the robot.

When computing relative localization, in each iteration odometry information is combined with the previous pose, both of which are subject to uncertainty. Over time, the cumulative error of the robot will increase significantly. One way to improve the estimate and reduce the error in measurements is to merge sensory information from different sources, which can be made with Kalman Filters.

Kalman Filter

The Kalman filter is an iterative process based on Bayes Theorem that uses consecutive data input to converge to the true value [21]. It involves calculating three values at each iteration: the Kalman gain, the current estimate, and its uncertainty (see Figure 2.6). The Kalman gain can be thought of as a variable that represents the confidence in the observations and predictions made. One limitation of the Kalman filter is that it assumes the system is linear, so it may not perform well in nonlinear scenarios. The Extended Kalman Filter (EKF) method addresses this by using a first-order Taylor expansion to approximate nonlinear functions as locally linear [22]. A more accurate method for nonlinear systems is the Unscented Kalman Filter (UKF), which uses an Unscented transformation[2]

[2]The Unscented transform picks a few points from the distribution of the input (sigma points), then it passes them through the nonlinear function and computes the mean and standard deviation of the output.

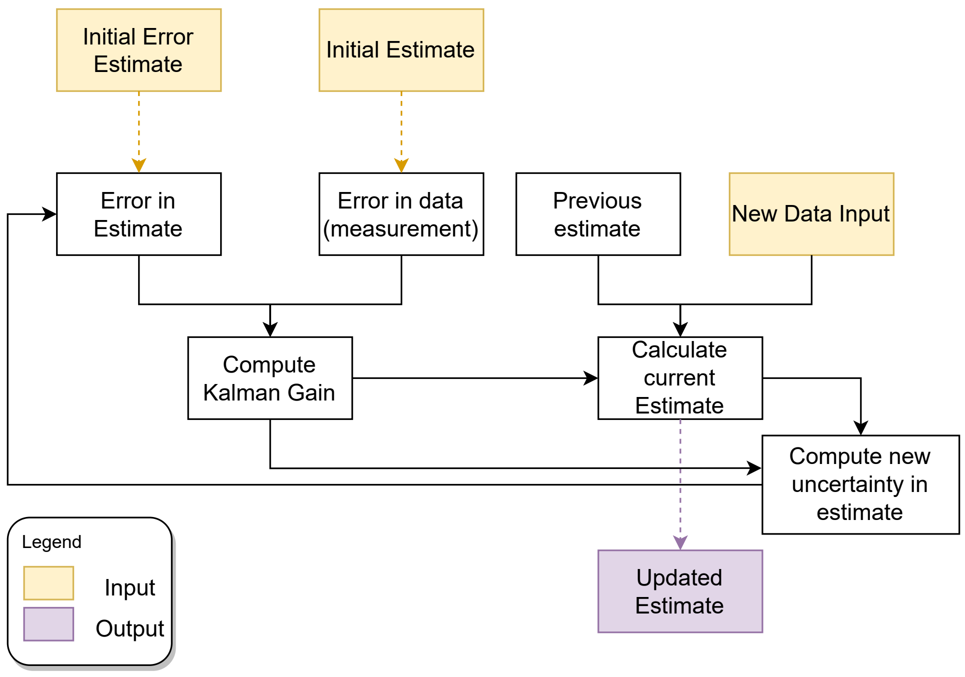


Figure 2.6: Simple flowchart of the Kalman Filter.

instead of locally linearizing the original function. However, all Kalman filter approaches assume that noise follows a Gaussian distribution, which may not be true.

2.3 SLAM techniques

Now that both localization and mapping were reviewed, the techniques to use them together for SLAM can be presented. Figure 2.7 offers a visual representation of a robot performing SLAM. As mentioned before, sensors capture information from the real world with uncertainty associated with each measurement. Even when algorithms are implemented to mitigate these errors, they never vanish entirely. When performing SLAM, a moving robot's perception and performance can be severely affected by these inaccuracies as they accumulate and become increasingly large.

2.3.1 Filter-based SLAM

Filtering based SLAM uses the prevously referred filtering techniques. The EKF SLAM is one way to solve the SLAM problem using the Extended Kalman Filter. As it uses the EKF, each map and pose must be linearly dependent of the previous iteration and this is major shortcome. If the robot moves at a constant speed, the change in pose

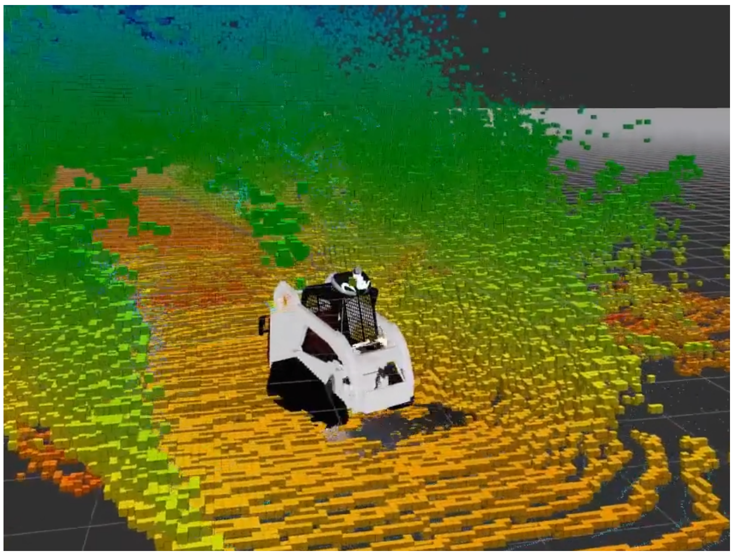


Figure 2.7: Robot moving and mapping the environment [23].

can be assumed linear. This will not be the case as the robot often accelerates. Yet, if the acceleration is not abrupt the nonlinear system will be successfully approximated to linear. The biggest drawback is the implied assumption of a static map. Moving entities in the environment will obviously not produce a close to linear dependency in successful iterations, therefore the first order Taylor expansion will not yield appropriate results.

The FastSLAM algorithm [24] uses particle filters to estimate the change in pose of the robot. By using Particle Filters, FastSLAM can handle nonlinear robot motion but it is more computational intensive. Another advantage of using a Particle Filter to estimate the motion is the multiple system states that are kept in memory through the resampling process, that only discards the low probability states. There are also implementations which replace the EKF with the UKF method to update the landmarks [25], that produce more accurate results but are usually more computational intensive.

2.3.2 Graph-based SLAM

The graph-based SLAM technique, initially proposed by Lu and Milios et al. [26], uses a graph-based approach, in which nodes represent poses or landmarks, while edges represent constraints between nodes, typically obtained from odometry or loop closure techniques, as illustrated in Figure 2.8. Unlike the filtering approach, the graph-based technique maintains knowledge on all gathered data and uncertainties are encoded in the graph. This approach divides SLAM into two main problems: graph construction e.g., building the

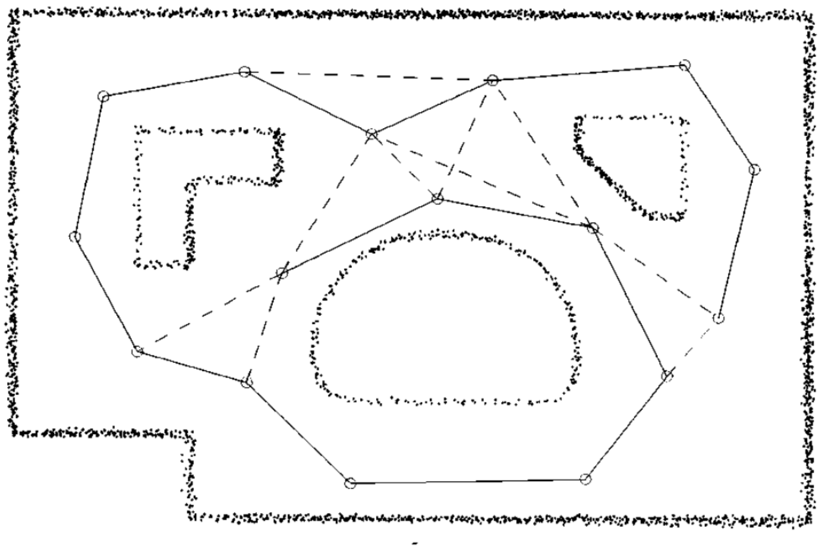


Figure 2.8: A pose-graph representation of SLAM. The circles represent the robot's poses, the solid lines are constrains derived from odometry and the dotted lines are from scan matching constrains [26].

graph from sensory information (commonly known as front-end) and graph optimization, i.e., analysing and estimating the most likely global configuration of poses given the constraints (also known as back-end).

Every new pose introduces a new node into the graph with multiple constraints associated, the constrains can be divided into *weak links* provided by odometry data and *strong links* obtained from scan or feature matching algorithms. These links can be viewed as elastic springs with different elastic constants [26], the global graph optimizer will then find the lowest energy solution to the system. The back-end searches for the best configuration of nodes that minimizes the error of all the constraints in the graph [27]. Some examples of state-of-the art graph optimization software are g2o [28], GTSAM [29] and Ceres [30]. To put it simply, the front-end is responsible for generation of accurate submaps, while the back-end ensures they are consistently correlated.

Loop Closure

The capacity of the system to detect previously visited locations is designated as loop closure. Loop closure identifies a previously mapped landmark and through feature or scan matching algorithms updates the robot location. Loop closure detection can significantly reduce the error in localization, since the cumulative error in pose is revoked. One can think of loop closure as a reset of the pose uncertainty whenever the robot passes through a previously known location. However, an inaccurate detection of a loop closure will introduce a large error in both the mapping and robot's localization, properly identifying the landmarks is of major importance.

2.4 ROS

To materialize an effective robotics project one must have a common framework that integrates sensors and processes. This task can be carried out by the Robot Operating System (ROS) [31].

Although the name suggests that ROS is an operating system, this is not the case. ROS is a middleware for robotic applications. The system provides a communication channel where messages can be subscribed, published and distributed, allowing quick integration between systems and components. Moreover, it provides features such as debugging, visualization, testing, logging, and configuration right out of the box. The basic execution file in ROS is called *rosnode*, via *launch files* it is possible to start multiple *rosnodes* with a single command. It is common to store the information circulating in ROS in a *rosbag*, a file containing all the messages of the selected *topics* in an ordered form, so that the given experiment can be later reproduced. Additionally, ROS includes a number of useful community developed packages for essential basic robotic tasks such as navigation, perception, and manipulation. The ROS community is also constantly evolving with the most recent developments in robotics, so libraries are constantly being added to ROS. Many of these libraries are open-source implementations of previously described SLAM techniques, some of them being reviewed and compared in this dissertation.

2.5 SLAM Implementations in ROS

There are a multitude of open source implementations of SLAM. Since one of the **Must** requirements presented in the MoSCoW analysis of Section 1 is a well integrated system within the ROS platform, only ROS implementations will be reviewed.

One of the most popular implementations of SLAM is Real Time Appearance Based Mapping (RTAB-Map) [13]. RTAB-Map is a graph-based SLAM system that uses a loop closure detector and offers several options for the back-end, namely GTSAM (default)

[29], g2o [28] and TORO [32]. The loop closure detector uses a bag-of-words approach to determine the likelihood that a new image was taken from a previous location or a new location. It supports several modalities, namely LiDAR, RGB-D and stereo cameras. It can estimate odometry from IMU and wheel encoders but it also supports Visual and LiDAR odometry as optional odometry sources. In order to ensure online[3] SLAM, RTAB-Map has a memory manager that keeps control of the amount of locations and landmarks used in the loop closure. Additionally, when executing the loop closure, RTAB-Map will reuse the features that were previously matched in Visual or LiDAR Odometry, this also contributes to improve performance. RTAB-Map can generate both 2D and 3D Occupancy grids with Octomap [15].

Several LiDAR-based method derive from LiDAR Odometry And Mapping, commonly known as LOAM. Implementations based on LOAM usually use LiDAR to obtain both odometry and mapping. Although it can create highly accurate maps, it usually performs poorly in places with few landmarks, like long corridors. LeGO-LOAM [33] add two additional modules to the LOAM technique: pointcloud segmentation and loop closure. These extra components allow an improvement in computing performance and drift reduction over long distances but does not improve the results when used in a featureless enviroment. LeGO-LOAM uses the naive ICP algorithm to perform loop closure, but a more robust approach, based in a point cloud descriptor is implemented in the SC-LeGO-LOAM [33, 34]. To help improving the performance in low features environment, researchers added an IMU to the system in a tighly coupled approach [35, 36, 37]. Due to the computational intensive task of fusing and processing all sensory data, the authors in [38] sugested the LiDAR Inertial Odometry via Smoothing and Mapping (LIO-SAM).

Cartographer is Google's implementation to solve the SLAM challenge [39]. It is another LiDAR based graph SLAM divided into two main components: local SLAM (the front end) and global SLAM (the back-end), shown in Figure 2.9. This approach takes input of range finding sensor, like LiDAR, and applies a bandpass filter to the input data. IMU can also be used to help figuring out the robot rotation and to provide information on gravity direction, that is used in the 3D variant.

[3]In literature, real-time SLAM is referred to as online SLAM and computationally intensive implementations that are not meant to run in real-time are referred as offline SLAM

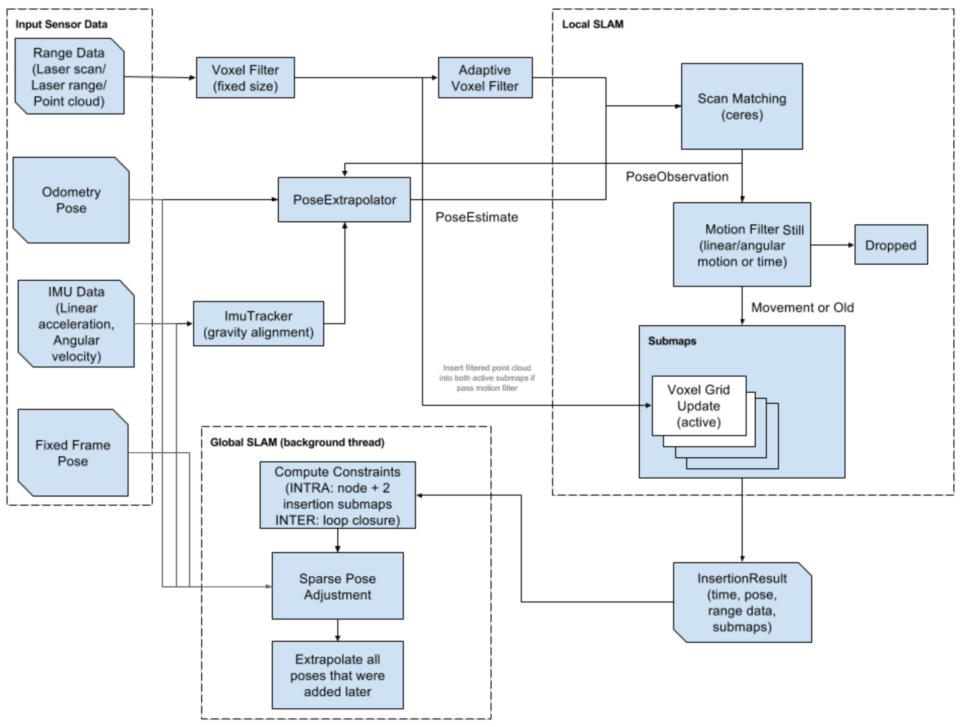

Input Sensor Data
Range Data (Laser scan/ Laser range/ Point cloud)
Odometry Pose
IMU Data (Linear acceleration, Angular velocity)
Fixed Frame Pose
Voxel Filter (fixed size)
Adaptive Voxel Filter
PoseExtrapolator
PoseEstimate
ImuTracker (gravity alignment)
Local SLAM
Scan Matching (ceres)
PoseObservation
Motion Filter Still (linear/angular motion or time)
Dropped
Movement or Old
Submaps
Voxel Grid Update (active)
Insert filtered point cloud into both active submaps if pass motion filter
Global SLAM (background thread)
Compute Constraints (INTRA: node + 2 insertion submaps INTER: loop closure)
Sparse Pose Adjustment
Extrapolate all poses that were added later
InsertionResult (time, pose, range data, submaps)

Figure 2.9: High level system overview of Google's Cartographer.

2.6 Related Systems for Data Acquisition

In the past, some portable and light sensors designed to collect data from the environment around us have already been proposed. There are several proprietary solutions available on the market for gathering sensory information for SLAM, but they tend to be expensive [40, 41]. Additionally, there are some research papers that report similar systems to the one proposed here. Considering that the methodology and results of these studies are readily available, these have been selected for further analysis below.

With the objective of measuring the Diameter at Breast Height (DBH), Oveland et al. [42] compared the performance of three different ground LiDAR scanning implementations, namely, Handeld LiDAR, Backpack LiDAR (composed by two LiDAR perpendicular to each other) and terrestial LiDAR. The authors compared the measurements of DBH to conclude that while the terrestial LiDAR Scanners (a fixed system) is the most accurate, it fails to detect ocult areas. According to the authors the handheld LiDAR scanner has issues in detecting smaller trees. Even though the backpack LiDAR has the highest number of false trees, the authors still aknowledge it as the most efficient method since it presents the highest amount of trees detected and a low Root Mean Square Error

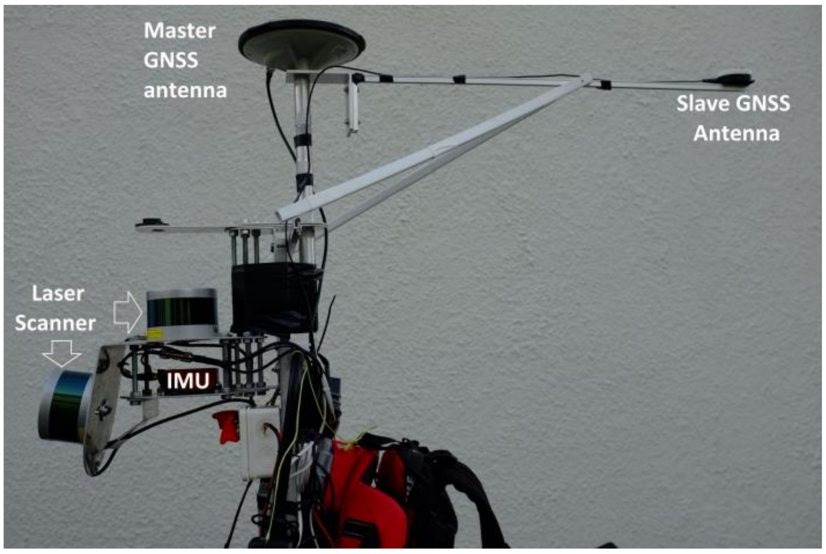


Figure 2.10: Backpack apparatus for dataset collection presented by Oveland et al. [42].

(RMSE) of 0.022m.

Recently, Kui Xao developed a dual LiDARs system, an IMU, and GPS for performing SLAM in indoor and outdoor applications [43]. To increase the vertical Field of View (FoV), one of the LiDARs was placed in the XY plane while the other was positioned with a pitch angle of $-77.94°$. A timestamp synchronization algorithm helped the authors merging the 3D scans from both LiDARs. Secondly, the LiDAR data is tightly coupled with the IMU data in order to reduce noise caused by the inaccuracy of LiDAR-based odometry. In outdoor tests, the IMU calibration is improved by loosely coupling GPS to the IMU. To review the performance of the proposed system, the authors used the Relative Translation Error (RTE) metric. In this instance, RTE gives the relative error between the initial and final position which are assumed to be equal. While their framework outperformed the LeGO-LOAM and LIO-SAM in indoor enviroment it failed to produce any meaningful diference in an outdoor setup. In forest environments, on the other hand, it can be expected to show a slight improvement due to the detection by the vertical LiDAR of the high trees, adding additional features.

Proudman et al. designed a portable system with the purpose of estimating DBH [44]. Based on an Ouster OS0-128 LiDAR and a RealSense D435i, both with an integrated IMU, the system is able to perform online SLAM and generate a map to estimate the DBH with a RMSE of 0.07m not considering outliers and 0.14m with outliers. While their application has the benefit of having a built-in display that allows real-time visualization

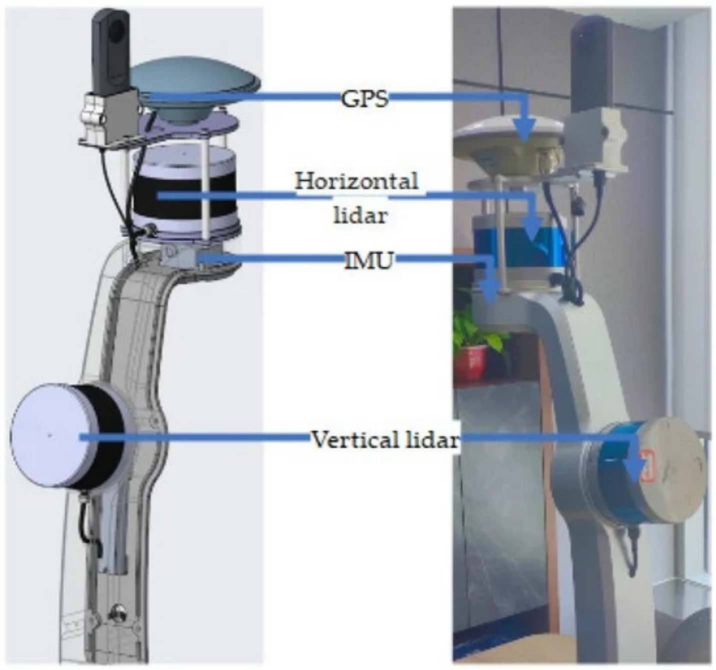


Figure 2.11: Apparatus introduced by Xiao et al. [43].

of data, one of its major drawbacks is the way they build their apparatus, opting for a metal stick rather than a backpack type of design. As the authors acknowledge, user fatigue may lead to excessive variations in stick position, resulting in unintelligible and uncontrolled movements, damaging the performance of the apparatus. This would not be an issue if a more ergonomic structure was used. The errors due to user fatigue can be slightly mitigated by performing SLAM in several sessions instead of one, allowing the user to rest between shorter sessions. After recording multiple sessions, GPS information is used to assemble the multiple sessions' map into a single map.

Yanjun Su et al. [45] developed an accurate backpack with dual LiDAR positioned orthogonally for estimating DBH and tree height. This system consists of two Velodyne Puck VLP-16 LiDARs, an IMU, and a display for viewing the generated point clouds in real time. The system was designed to not be GPS dependent. Using an open-source Python package, the researcher claims to have achieved an RMSE of 0.02m, a value slightly lower than Proudman's. Due to its vertical LiDAR, the system was also able to estimate the height of trees, obtaining a RMSE of 1.9m.

Sier et. al [46] designed a very complete apparatus with the objective to compute ground truth for agent trajectories in GNSS denied environments. The apparatus features six different LiDAR, with both spinning and solid state technologies and a stereo fish eye

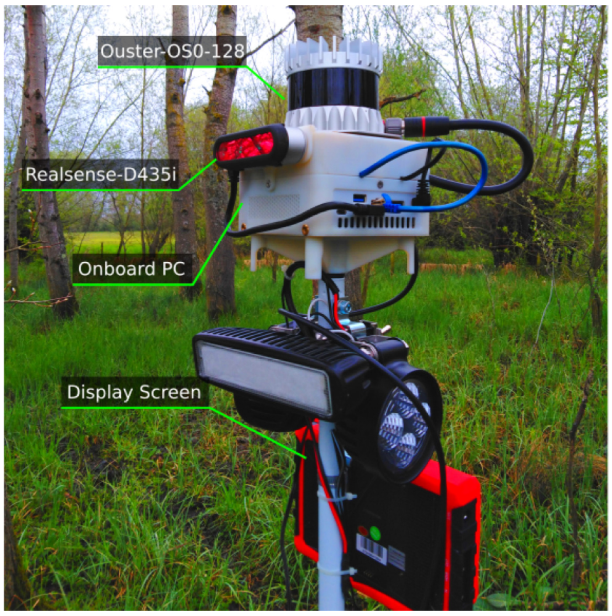


Figure 2.12: Apparatus with a stick design by Proudman et al. [44].

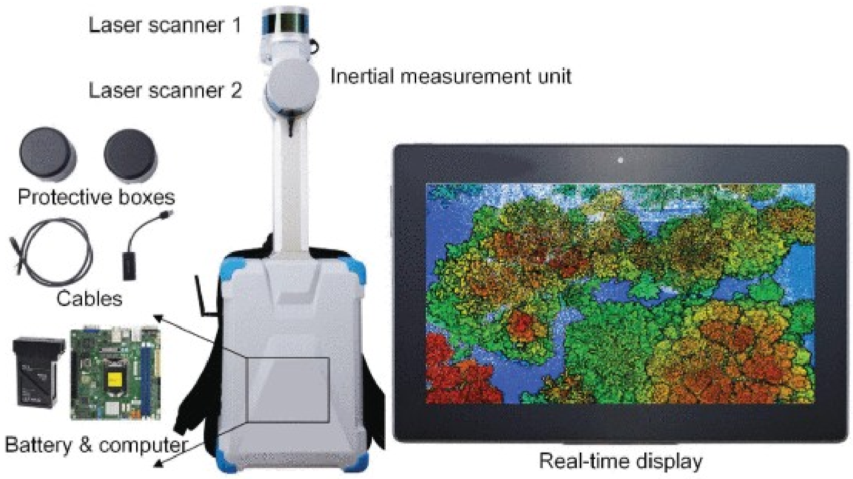


Figure 2.13: Apparatus designed by Yanjun et al. [45].

camera. To calibrate the several LiDAR the authors used the same methodology as in [47]. Essentially, the transformation between LiDARs was obtained by trying to minimize the distance error between projected points. The authors compare different state of the art LiDAR based SLAM with different LiDAR configurations to assess the best overall combination. Using the Absolute Position Error (APE), a metric introduced in [48], the authors concluded that the most robust combination is the FAST-LIO [36] implementation using the more precise OS0 and OS1 spinning LiDARs. Also, on outdoor environments the solid state LiDARs had a similar performance against the more expensive counterparts. This is to be expected as the number of features in outdoor environments is considerably

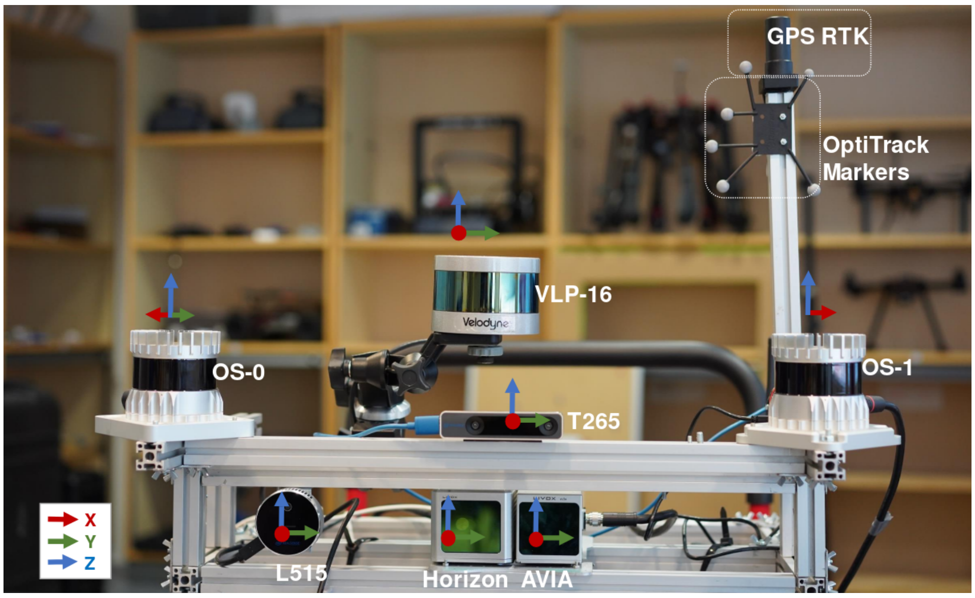


Figure 2.14: Multi-sensor apparatus designed by Sier et al. [46].

larger.

A robot system developed by Jelavic et al. performs precision harvesting missions [5]. The entire workflow begins with a human mapping the desired location with a *sensor module*. Afterwards, a human expert indicates which trees need to be cutdown with the autonomous vehicle. With the sensor module attached to the robot, a planning algorithm determines the most efficient path and pose to reach the next tree. Once the pose proposed in the previous step has been reached, the tree is grabbed and cut. In many ways, their *sensor module* is similar to the one proposed in this dissertation. The system uses a RealSense T265 tracking camera, an Ouster OS-1 LiDAR, an IMU, and a FLIR camera model BFS-U3-04S2M-CS, valued at approximately $10500 dollars. Google's Cartographer was used to create the 3D map of the forest environment, with the LiDAR being used for mapping and cameras for obtaining visual odometry. Only one of the cameras can be used at a time to obtain visual odometry. As the design is similar to [44], the stick module may lead to user fatigue as discussed previously.

Table 2.1 presents a comparison between the several apparatus reviewed. As one is able to see, the apparatus focused in forest aplication inventory and other applications do not compare multiple SLAM implementations, being particularly focused in determing the DBH of trees. Some of the forest focused apparatus send the recorded footage to a cloud base system, such as GeoSLAM Hub[4] to perform offline SLAM. Of all the implementations

[4]GeoSLAM Hub is a SLAM cloud provider, producing a map based on 3D data received. The SLAM

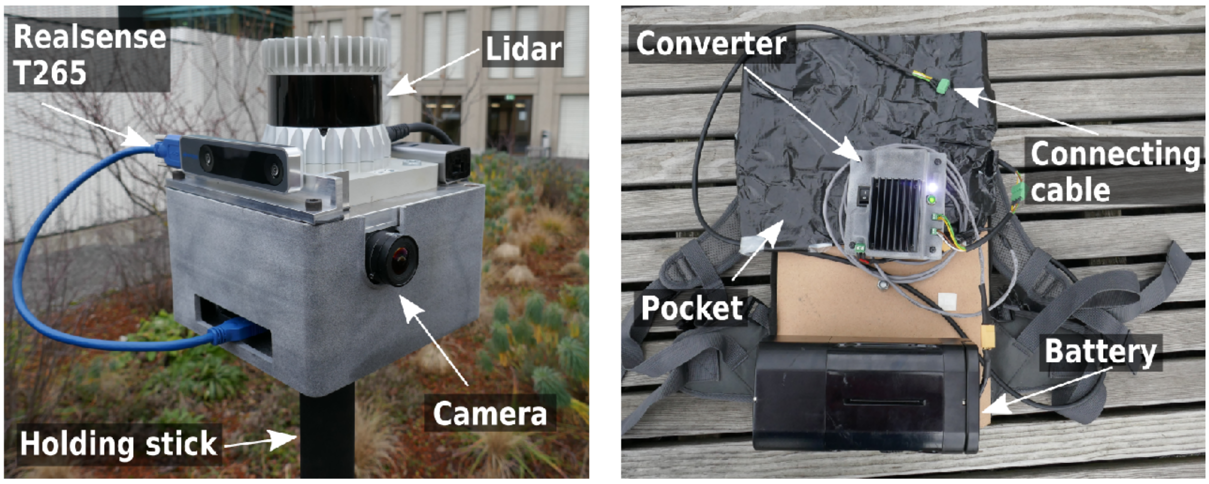


Figure 2.15: Apparatus proposed by Jelavic et al. [5].

discussed only one uses a RGB-D camera, a critical component to perform flammable material identification, as well as trees species and objects identification. Having reviewed the relevant literature, current state of the art of the field and similar solutions to aquire data, the necessary conditions to present the implementation and design chapter were met. In the following chapter, a detailed description of the proposed apparatus will be presented, including the trade-offs and decisions made during the design process.

algorithm used is not described

Table 2.1: Comparison table between previously mentioned systems.

Implementations		**Oveland et al. [42] 2018**	**Xiao et al. [43] 2022**	**Proudman et al. [44] 2021**	**Su et al. [45] 2021**	**Sier et al. [46] 2022**	**Jelavic et al. [5] 2021**
Inputs	Stereo	No	No	No	No	No	Yes
	RGBD	No	No	Yes	No	No	No
	IMU	Yes	Yes	Yes	Yes	Yes	Yes
	LiDAR	Yes	Yes	Yes	Yes	Yes	Yes
	GNSS	Yes	Yes	Yes	No	Yes	No
Display		No	Yes	Yes	Yes	No	No
Structure Type		Backpack	Backpack	Stick	Backpack	Wheel Cart	Stick
SLAM used		GeoSLAM Hub	LeGO-LOAM, LIO-SAM, HSDLIO	Graph-based SLAM	N.A	LeGo-LOAM, FAST-LIO Livox-Mapping, LIO-LIVO	Cartographer
Localization Error (m)		N.A	0.026m (RTE)	N.A	N.A	0.092 (APE)	0.41m
Forestry Application		Yes	No	Yes	Yes	No	Yes

3 Design and Implementation

The purpose of this sytem is to support current and future projects in forest environments. To accomplish this, the apparatus must have sensors that meet the system requirements drawn in Section 1.2, as well as software to ensure that all sensors function and operate as intended. It is also important to have a robust mechanical structure that holds all components securely and is resistant to high ambient temperatures and the heat generated by the components. User safety should also be considered in the design of the mechanical structure. Finally, all the necessary software must be integrated or implemented so that the recording process is as simple as turning on the system and executing a single command.

3.1 Sensors and Components

The apparatus is equipped with multiple sensors that allow it to meet the requirements proposed. Figure 3.1 illustrates the four main sensors included in the proposed apparatus. The Xsens MTi is a IMU (Figure 3.1a), which can measure linear acceleration, gyroscopic and magnetic information. The Mid-70 Livox, shown in Figure 3.1b, is a 3D solid-state LiDAR that can publish point clouds at a frequency of 50 Hz. While it has precision for close objects and a maximum range of 260 m, it has a small FoV of 70.4°. In addition, this LiDAR produces a sparse point cloud (see Figure 3.2).

A Mynt Eye S1030 stereo camera (see Figure 3.1c) is included to increase the FoV degrees for mapping. It has a large FoV of 146° and can also provide linear acceleration and gyroscopic information from an inbuild IMU. The Mynt Eye does not provide RGB image, but two monocolor images able to detect light in the infrared spectrum. The apparauts also includes an Intel Realsense D435i camera, which provides additional depth information and serves as the single source of RGB and infrared information, useful for

(a) Xsens MTi IMU.

(b) Livox LiDAR Mid-70.

(c) Mynt Eye S1030.

(d) Intel Realsense D435i.

Figure 3.1: Collection of the apparatus' sensors.

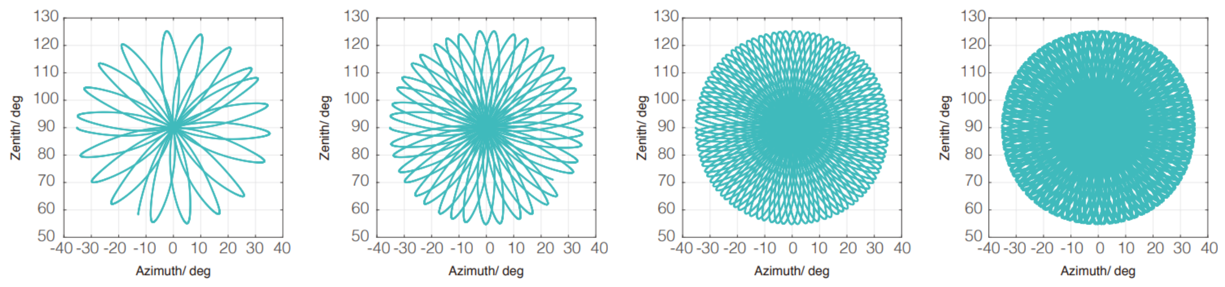


Figure 3.2: Point cloud patterns of the Livox Mid-70 accumulated over an time extended period of time [49].

identifying flammable material.

The Udoo Bolt is the small factor computer chosen, shown in Figure 3.3. Some of Udoo Bolt specifications are outlined on Table 3.1. One of the PCIe slots is used for connecting an SSD with 1TB of storage, which is extremely useful for dataset recording. The Udoo Bolt runs the Linux Operating System with the Ubuntu 20.04 focal distribution. The entire system is powered by a 14.8V Turnigy battery with 10000mAh, which can provide more than 4 hours of continuous system operation. A diagram showing how the various modules are powered can be seen in Figure 3.4.

Figure 3.3: The small factor computer Udoo Bolt.

Table 3.1: Specification table for the Udoo Bolt.

Components	Quantity
AMD Ryzen V1202B dual-core	1x
USB 3.1 Type-A	2x
USB 3.1 Type-C	2x
HDMI ports	2x
32GB eMMC 5.0	1x
Arduino Leonardo	1x
PCIe M.2 slots	3x

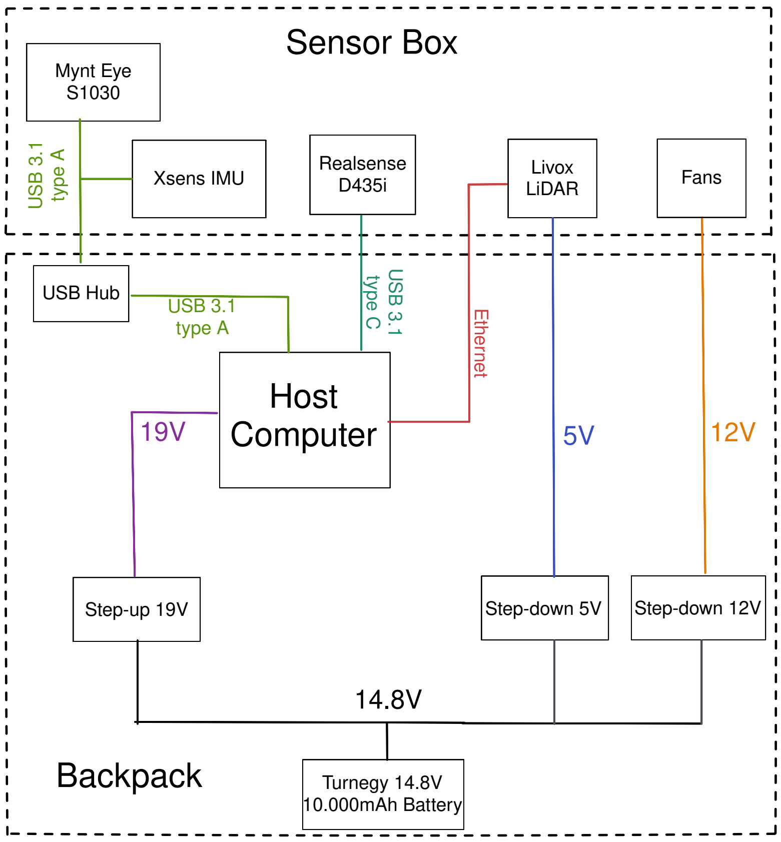


Figure 3.4: Diagram of the connections for the different system components in the proposed apparatus.

The structure was divided into two distinct components: the *sensor box*, which houses all the cameras and sensors, and the *backpack*, which houses the computer, battery, the step-ups and step downs.

3.2 Mechanical Structure

In the literature reviewed, it has been demonstrated that the handheld design of sensor systems can lead to user fatigue. The handheld design offers high maneuverability for the sensors, making it easy for the user to direct it towards specific locations or objects. Despite its advantages, the handheld design imposes a physical strain on the user, as it requires the use of one or both hands to hold the apparatus, thereby making it less practical.

Since the host computer can reach high temperatures, the structure inside the backpack was made of Acrylonitrile Butadiene Styrene (ABS), which can withstand temperatures of about 80°C without significant degradation [50]. Although ABS is heat resistant, it is not resistant to ultraviolet radiation (UV), which may become an issue in the future, as it is expected to operate outdoors for several hours along its lifecycle. The alternative is to use Polytetrafluoroethylene (PTFE), a material that is heat resistant up to 60°C and has better UV resistance, which may be considered in the future. To mitigate potential thermal issues in the *sensor box*, several fans were set up to blow fresh air into the cameras, which tend to heat up after long periods of operation. A heat sink was also attached to back of the Realsense D435i camera to improve heat dissipation. The Livox LiDAR was mounted on top so that it could be easily upgraded to a 360° LiDAR in the future. Since the main purpose of the Mynt Eye is to increase the FoV of the mapping, the mount that holds it in place was designed to allow easy rotation around the yaw axis. In this way, the extent to which the Mynt Eye point cloud overlaps with the rest can be tuned to the user's application. The complete apparatus is shown is Figure 3.6.

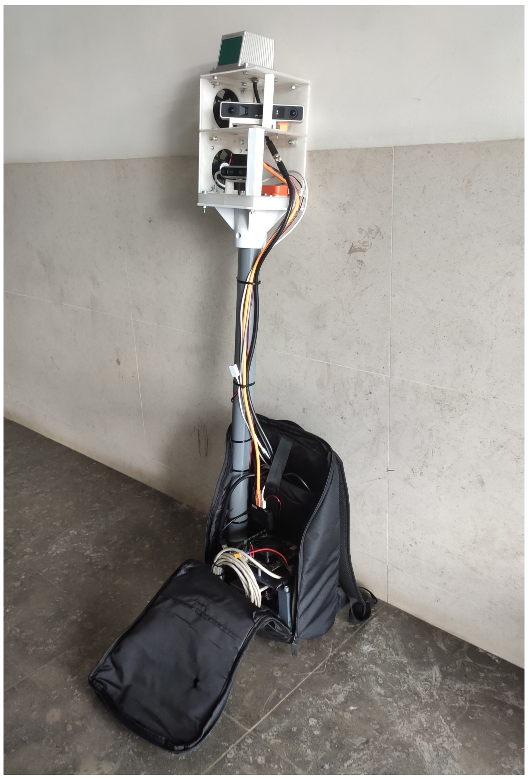
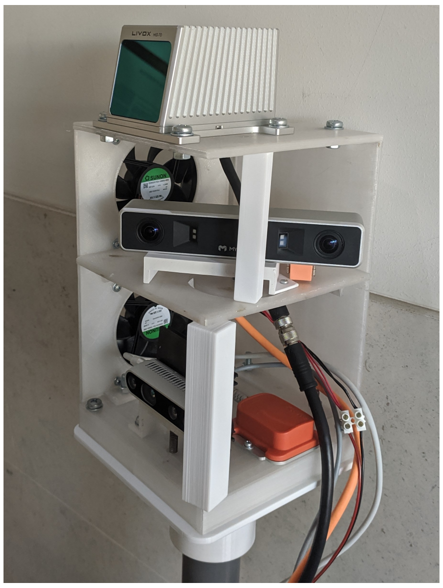
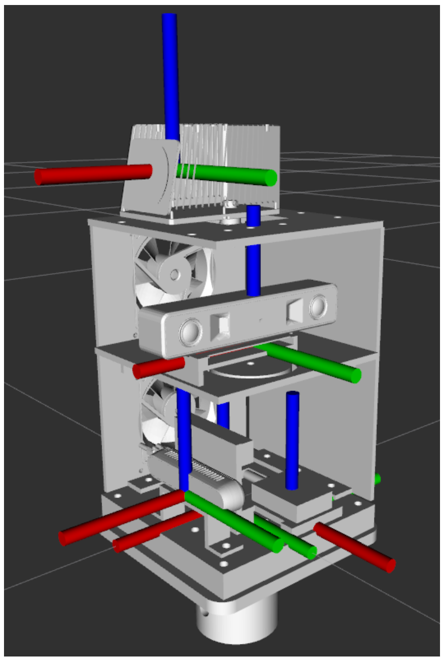
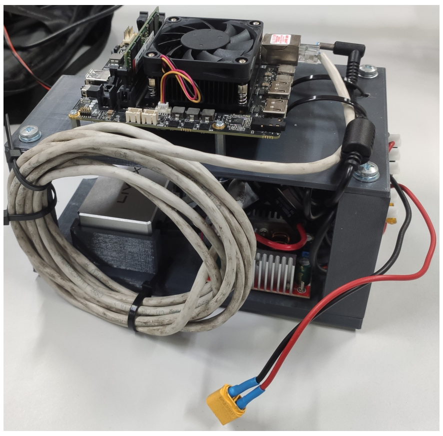


Figure 3.6: Complete apparatus with the backpack open.

The reference frame in the apparatus is set to the bottom shelf, as Figure 3.5 shows, any sensor transformation is referenced to this location, Figure 3.7 shows the complete transform tree for the apparatus. Almost all sensor poses are known from the Computer-aided Design (CAD), with the exception of the Mynt Eye, where the yaw angle is adjustable by the user at will. Sensor registration is done manually using transformations provided by CAD. Once the yaw of the Mynt Eye is set, the pose is manually adjusted by visually comparing the intersection of the different point clouds. Figure 3.8a illustrates the different point clouds in the scenario of Figure 3.8b, as seen by the Realsense D435i RGBD camera. The overlap of the different point clouds appears to be accurate enough for this application. However Figure 3.8c shows that there is room for improvement in extrinsic calibration of sensors.


(a) *Sensor Box*.


(b) CAD design of the *Sensor Box* with references of parent frames for each sensor.


(c) *Backpack* base.

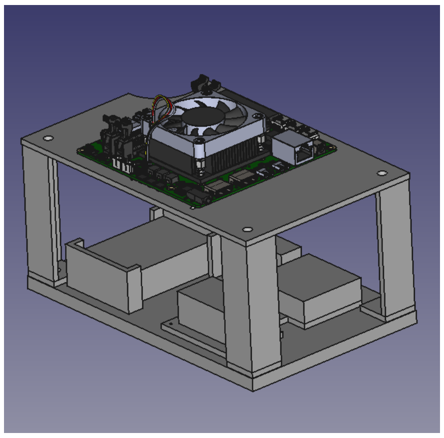

(d) CAD design for the *Backpack* base.

Figure 3.5: Mechanical structure of the apparatus, divided into the *sensor box* and *backpack*.

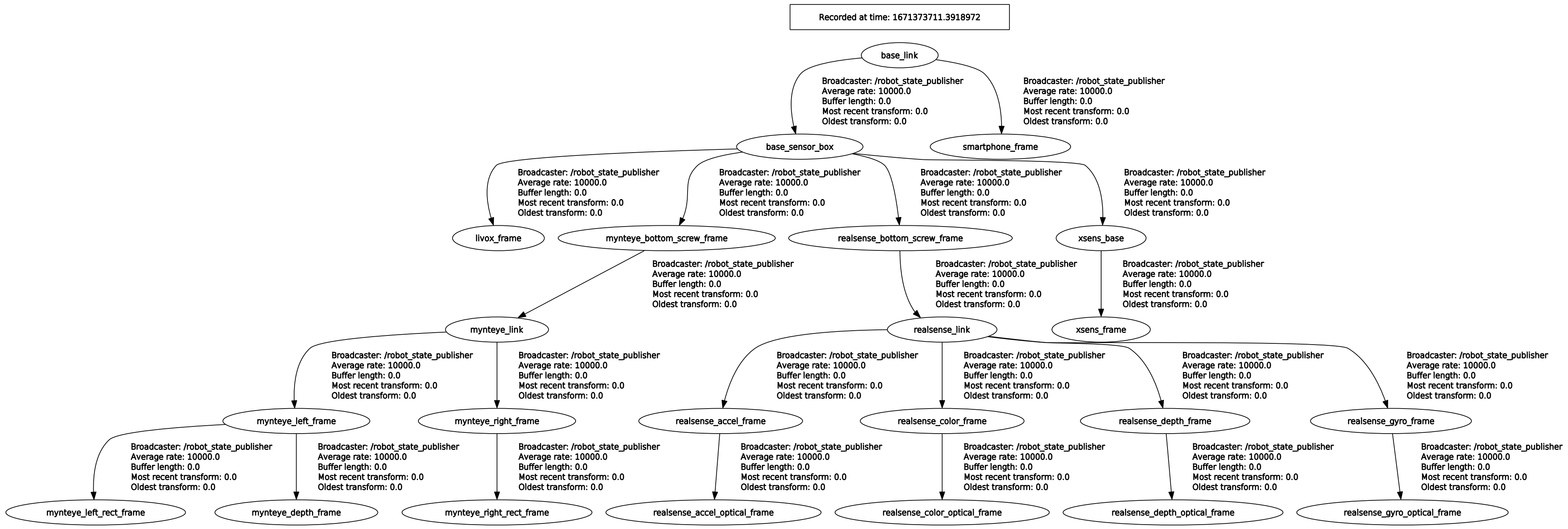
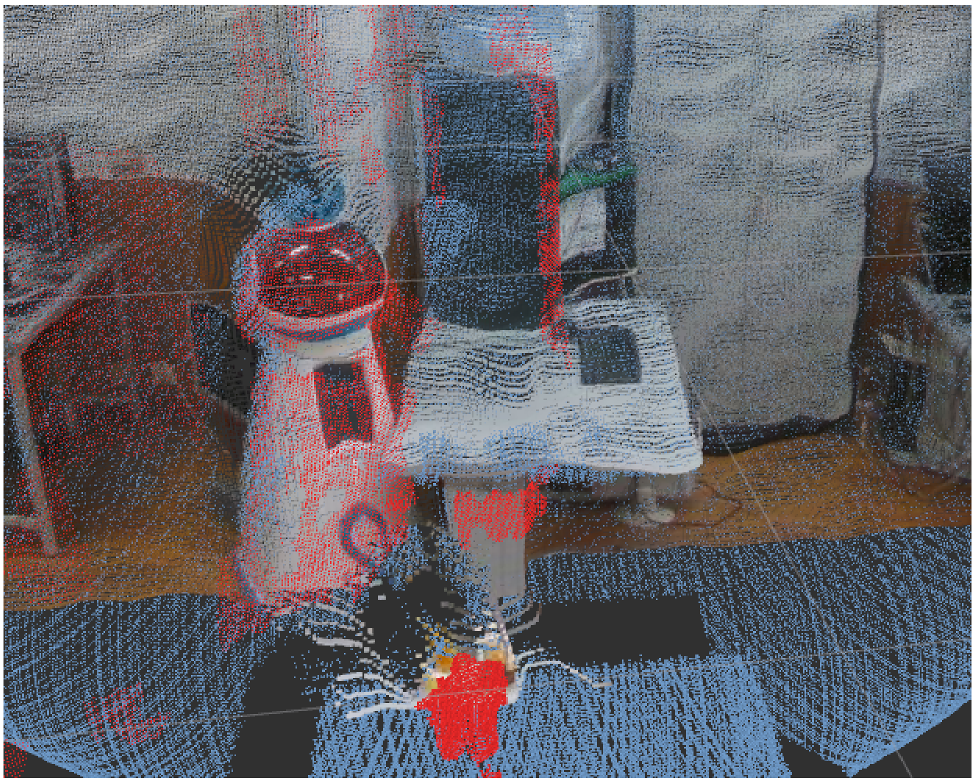
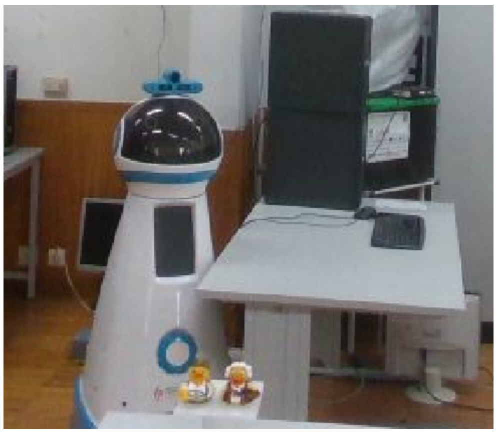

Recorded at time: 1671373711.3918972
base_link
Broadcaster: /robot_state_publisher Average rate: 10000.0 Buffer length: 0.0 Most recent transform: 0.0 Oldest transform: 0.0
base_sensor_box
smartphone_frame
livox_frame
mynteye_bottom_screw_frame
realsense_bottom_screw_frame
xsens_base
mynteye_link
realsense_link
xsens_frame
mynteye_left_frame
mynteye_right_frame
realsense_accel_frame
realsense_color_frame
realsense_depth_frame
realsense_gyro_frame
mynteye_left_rect_frame
mynteye_depth_frame
mynteye_right_rect_frame
realsense_accel_optical_frame
realsense_color_optical_frame
realsense_depth_optical_frame
realsense_gyro_optical_frame

Figure 3.7: Complete transform frame for all components of the apparatus.


(a) Overlap of the different point clouds. LiDAR point cloud in blue, Mynt Eye point cloud in red and D435i point cloud in RGBD.


(b) RGB image of the experimental scenario.

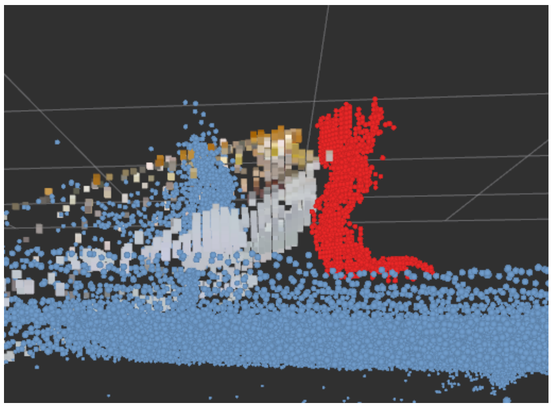

(c) Side view of point clouds of a rubber duck from the several depth sensors.

Figure 3.8: Visual validation of the transformations between sensors.

3.3 Software

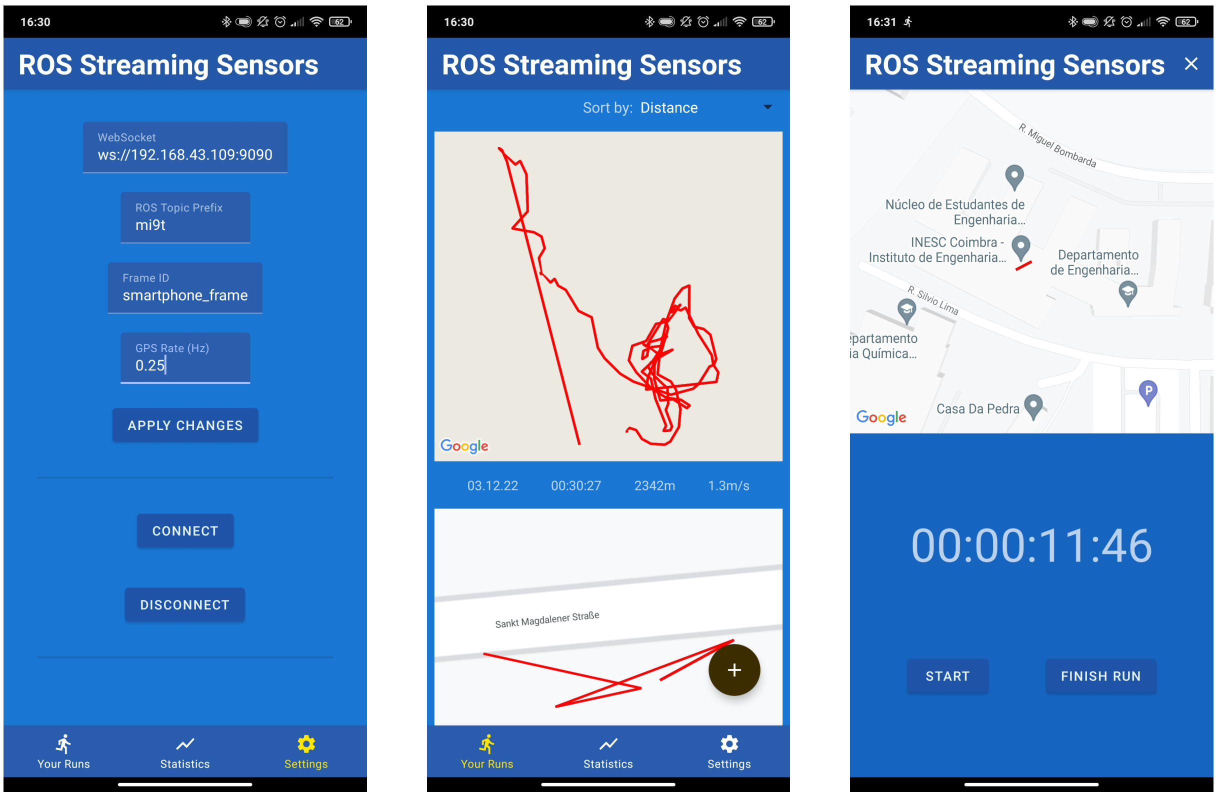
To operate the apparatus effectively, the integration of several software components is required. First and foremost, a software solution must be implemented that can gather GNSS data from Android devices. In addition, a software to launch the system is also necessary to initiate the apparatus and enable data recording and storage. Lastly, it is important to devise an architecture that can compare various SLAM implementations.

3.3.1 Android Application

It is extremely useful to have a source for absolute localization, especially when performing localization in unknown environments. Although there are various solutions on the market, nowadays almost every cell phone receives GNSS information. The possibility of using a smartphone to localize the user was investigated by reviewing some Android applications that establish communication with ROS. Almost all applications currently on the market [51, 52] do not allow transmitting data from Android sensors such as GNSS, gyroscope and accelerometer to the ROS channel. The only application that allows such communication is the Android Sensors Driver [53], which has not been maintained since 2012, meaning that it does not work with current Android versions, as features such as access permission to GNSS information have changed drastically over the years. For this reason, a new Android application was designed and created, named ROS Streaming Sensors[5]. This novel application allows streaming sensor data, such as GNSS to a ROS system. The application follows a Model View View-Model (MVVM) architecture to facilitate maintenance and development of new features. To send messages to ROS, the application establishes websocket communication with a *ROS Bridge* [54] node running on the host computer to which multiple devices can be connected simultaneously.

The application allows the user to configure parameters such as the publication rate, the websocket address, and the ROS topic name (see Figure 3.9a). The user can also view all the previous runs it made and sort them by date, time, distance and average speed as Figure 3.9b illustrates. When starting a new test run, the user is shown a map with

[5]The full application documentation is available at `https://mjpc13.github.io/SensorStreamer/` and an alpha version is available for download at `https://github.com/mjpc13/SensorStreamer/releases`


WebSocket
ws://192.168.43.109:9090

ROS Topic Prefix
mi9t

Frame ID
smartphone_frame

GPS Rate (Hz)
0.25

Sort by: Distance

03.12.22 00:30:27 2342m 1.3m/s

00:00:11:46

(a) Settings option. (b) List of runs. (c) Current Run.

Figure 3.9: Screenshots of the ROS Android Sensor application.

his/her current location and the option to stop and finish the current run (see Figure 3.9c). Lastly, the user has access to global statistics like average speed and total distance travelled.

3.3.2 System Architecture for Dataset Recording

In this dissertation there are advantages in containerizing the multiple software used. Isolating the different components leads to a bigger modularity and Docker is a popular tool that helps containerizing software [55]. A Docker container can be seen as an almost independent computer running inside the host machine. As it is isolated, the software in a container is independent of other softwares in the computer, making version dependency easy to manage. What this isolated container shares with the host computer depends on the configuration, for the work of this dissertation all docker containers share the host networking and are able to access I/O devices. This essentially means that containers are free to comunicate with each other and with the host while also communicating freely with I/O devices. Since Docker allows developers to create containers that include all the software and dependencies that an application needs in order to run, the application can run consistently across different environments, such as different operating systems or across multiple machines, which do not even need to have ROS natively installed. Through the years, it has become popular to pair ROS with Docker [56].

In this work, the use of Docker is critical as some of the sensor drivers were created for different, not fully cross-compatible versions of ROS. Therefore, the full architecture for the recording process, shown in Figure 3.10, is designed around Docker, with six different containers used: a *ROS Master* (running ROS Noetic), *ROS Melodic*, *ROS Noetic*, a *ROS Bridge*, a recording container, and a diagnostic container.

The *ROS Master* is the first container launched, and the remaining containers connect to it. Since all containers use the host networking to receive and send packets, the ROS master running is also accessible to the host computer. Because of the different versions of Python used in ROS Melodic (Python 2.7) and Noetic (Python 3), software developed for one version is usually incompatible with the other. While the drivers for Realsense D435i and the Livox LiDAR run on ROS Noetic, the drivers for the Mynt Eye and Xsens depend on ROS Melodic or older versions. To manage the incompatibility between ROS *Melodic* and *Noetic*, a container is created for each version. Then there is a container running a ROS bridge server that exposes port 9090 on the host computer to receive GNSS information from the Android Sensor ROS application. There is also another container to write the dataset to a rosbag and a ROS debugging node that records a separate rosbag with various debug information, such as topic frequencies and temperatures of the CPU.

3.3.3 System Architecture to perform SLAM

To validate the dataset and compare different implementations, an architecture to perform SLAM, shown in Figure 3.11, was designed. Most implementations of SLAM require odometry as input, so there must be a ROS node that converts the several sensory informations into odometry data. Then the desired odometry must be fused with the IMU measurements and optionally with GNSS in a sensor fusion node. Some of the SLAM implementations used can also accept IMU and GNSS inputs directly. robot_localization [57] is the package used for the sensor fusion node, an approach based on Kalman filters. This package allows both the EKF and UKF options and it can have multiple IMUs and odometry as inputs. For the SLAM node, one of the following packages is used:

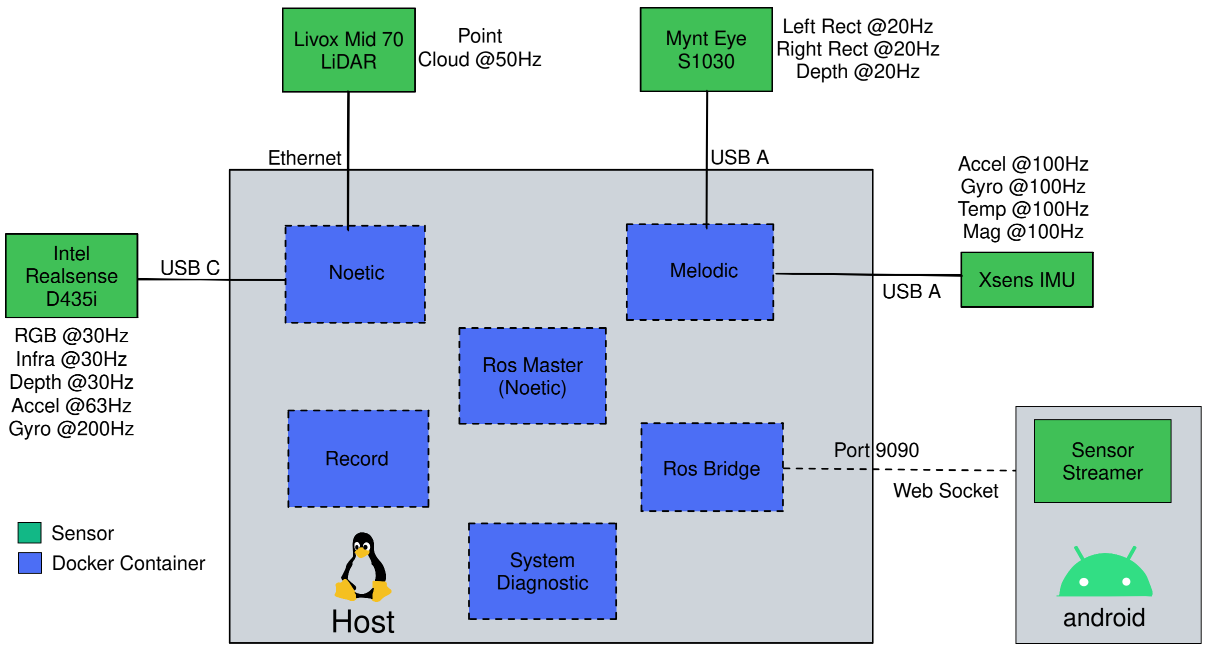


Figure 3.10: Complete system architecture for recording datasets with the proposed apparatus.

- RTAB-Map [13];
- Google's Cartographer [39];
- *livox_mapping*, a LOAM-based implementation[6].

The *livox_mapping* library is chosen due to its optimizations designed specifically for Livox's solid-state LiDARs, considered the most accurate and reliable sensor available in the apparatus. However, this library is based on LOAM and does not incorporate loop closure, which is a crucial aspect of SLAM. On the other hand, Cartographer and RTAB-Map, the two alternative graph-based approaches, both possess the ability to perform loop closures. Despite having similar goals, these two approaches have distinct methodologies and philosophies when it comes to SLAM. Cartographer, for instance, can process multiple point clouds inputs, leveraging the various depth sensors available, but does not utilize the RGB channel of the Realsense D435i. Conversely, RTAB-Map can only only detects loop closures based on images, not on LiDAR. Furthermore, while Cartographer heavily relies on the performance of the IMU, RTAB-Map is not dependent on the performance of the IMU and can operate without one. RTAB-Map also has the

[6]The *livox_mapping* package can be found at `https://github.com/Livox-SDK/livox_mapping`

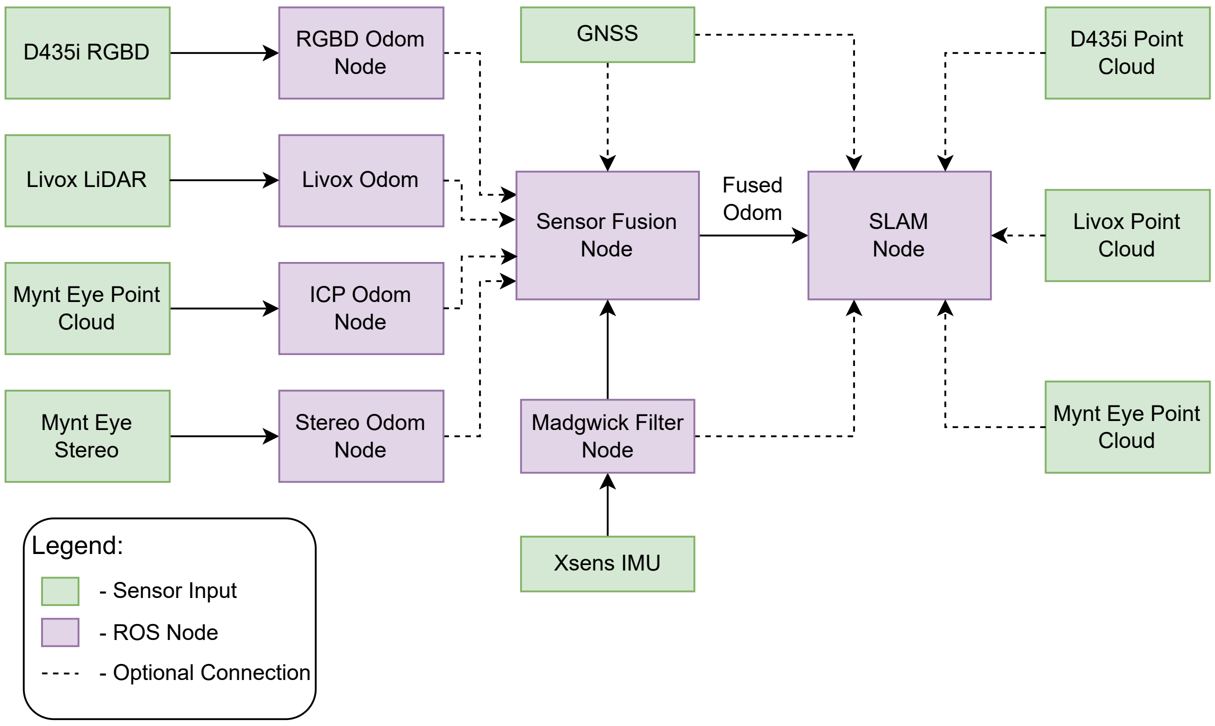


Figure 3.11: Base architecture to perform SLAM with the proposed apparatus.

advantage of displaying the map in three dimensions, using the Octomap package [15]. The differences between the approaches make them worthy of consideration for this work.

4 Preliminary Tests and Experimental Design

This chapter describes preliminary tests that have been conducted and the performance metrics associated with each test. These tests have been defined to verify if the several individual components work as expected before integrating them in a final experiment to generate a robust dataset.

4.1 Android Application Test

This test aims to validate the functionality of an Android application and its ability to publish data to ROS. Additionally, the practicality and effectiveness of the application will be assessed through a comparison of positional estimates generated by multiple Android devices against a ground truth map. The objectives of this study are:

- Validation of the functionality of the Android application and the successful transmission of data from the Android device to ROS.
- Assessment of the practicality and effectiveness of the application through a comparison of the positional estimates generated by multiple Android devices against a known map.

Experimental Design

This experiment was performed in the Choupal National Woods ($40°13'13.3''N$; $8°26'38.1''W$) in Coimbra, Portugal. The user performed two circular loops around the smaller square and a lemniscatory[7] trajectory (see Figure 4.1), amounting to a total distance of around

[7]A lemniscate is any trajectory that resembles the infinite symbol (∞).

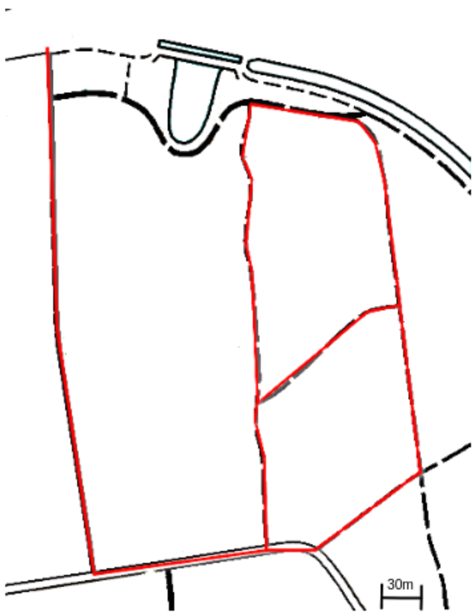


Figure 4.1: Trajectory conducted by the user in red.

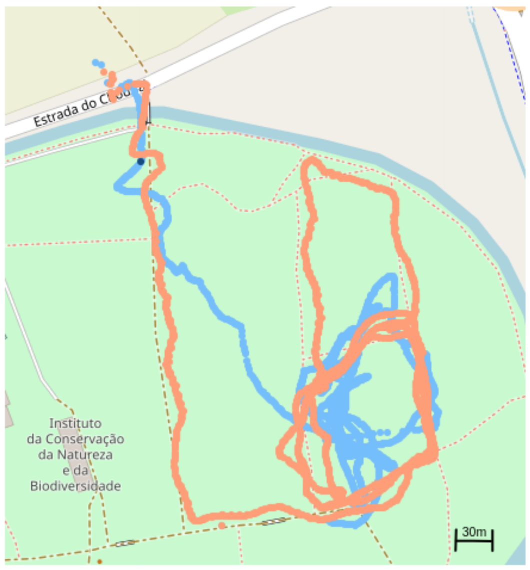


Figure 4.2: GNSS test results. Position tracking of the Xiaomi Mi 9T and the Xiaomi Mi Mix 3 is displayed in blue and orange, respectively.

2000m. During the entire run, the user held two android devices (a Xiaomi Mi 9T and a Xiaomi Mi Mix 3) side by side in their hands. The experiment was performed during a sunny day without clouds in the sky at around 15:30h. The Android application fused GNSS data with cellphone tower localization and broadcasted the information as a *sensor_msgs/NavSatFix* ROS message. The devices were connected to a common ROS Node Bridge, which facilitated the publication of the GNSS messages to a designated topic. The resulting data was recorded in a rosbag and subsequently visualized in Figure 4.2 through plotting.

Results and Discussion

The results of the experiment revealed that the Xiaomi 9T demonstrated inadequate performance and precision, making it unsuitable for this work. The Mi Mix 3 on the other hand offers localization that is sufficiently representative of the actual trajectory, thus it will be used in future testing. Both devices were tested under the same conditions, and the main difference between them, as per their specifications[8], is that the chip in the Xiaomi 9T is older, which could be the reason for its degraded performance compared to the Xiaomi Mi Mix 3.

4.2 DEEC Building Dataset

This study serves as a preliminary step towards the main dataset collection. In this study, the various components of the apparatus described in previous chapters are tested in a challenging environment in order to identify any potential issues. The aims of this experiment are:

- Evaluation of the comfort and usability of the apparatus for the user.
- Assessment of the mechanical durability and stability of the apparatus.
- Confirmation that all topics are being recorded at the desired frequency.
- Acquisition of a preliminary dataset to initiate the integration of the SLAM strategies outlined in the preceding chapter.
- Determination of the feasibility of performing SLAM with the recorded data.
- Identification of any potential malfunctions of the device prior to collecting a dataset in a forest setting.

Performance Metrics

Performance metrics are critical in measuring performance and discussing results. To evaluate the apparatus presented in this dissertation, the following metrics are the ones considered.

Relative Topic Frequency - The ratio between the measured frequency and target frequency:

$$RTF = \frac{f_{measured}}{f_{target}}, \tag{4.1}$$

[8]The specification for the Xiaomi 9T and Xiaomi Mi Mix 3 can be found at `https://xiaomiui.net/smartphones/xiaomi-mi-9t/`, `https://xiaomiui.net/smartphones/xiaomi-mi-mix-3/` respectively

where the *target frequency* is the frequency one should expect from the sensor. This metric highlights the disparity between the anticipated frequency and the actual frequency for a given topic. In the context of this research, values above 0.95 denote an optimally functioning sensor.

Relative Translation Error (RTE) - The relative error between the start and final position:

$$RTE = \sqrt{\sigma_x^2 + \sigma_y^2 + \sigma_z^2}, \tag{4.2}$$

where σ_i is the module of the difference between final and starting value:

$$\sigma_i = |i_{final} - i_{start}|, \tag{4.3}$$

Given that the user begins and concludes his/her path in the same position, the value of RTE should equal 0 and the closer it is to 0, the more optimal it is. However, this metric may not effectively measure drift as loop closures identified between the initial and final positions will reduce the accumulated drift error. The RPE and APE are two commonly used metric in the literature to measure drift, however these metrics demand a ground truth, something that this work lacks.

Relative Orientation Error (ROE) - The relative error between the start and final orientation:

$$ROE = \sqrt{\sigma_{roll}^2 + \sigma_{pitch}^2 + \sigma_{yaw}^2} \tag{4.4}$$

This metric is similar to the previous one, but for the orientation.

Experimental Design

The dataset was collected on the second and third floors of the Departamento de Engenharia Eletrotécnica e de Computadores (DEEC) building at the University of Coimbra. This localization was chosen primarily because it provides both spaces with several of features[9] (see Figure 4.3a) and spaces with few features (see Figure 4.3b) while being close to the development location.

[9]Spaces with several of features generally contain several objects with different colors, textures and shapes.

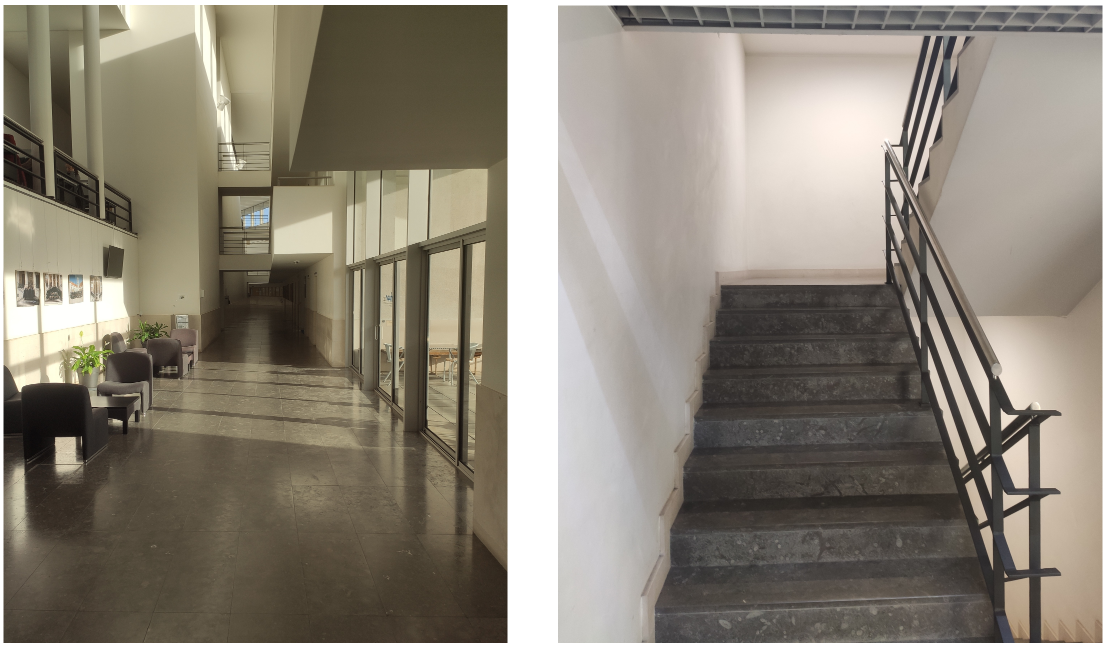

(a) Corridor with features. (b) Featureless staircase.

Figure 4.3: Types of spaces present in the DEEC dataset.

In this experiment, the user performs a circular loop trajectory over two floors of the building with the apparatus on their back in a run of 4:45s. Since the experiment takes place in an indoor environment, the GNSS information was intentionally omitted. All the recorded ROS topics and their corresponding publishing frequency can be seen in Table 4.1.

Table 4.1: Topics collected in the DEEC dataset.

Topic	Message Type (sensor_msgs/*)	Frequencies (Hz)	RTF
/imu/data	IMU	99.86	1.00
/imu/mag	MagneticField	99.86	1.00
/imu/temperature	Temperature	99.86	1.00
/livox/lidar	PointCloud2	50.03	1.00
/mynteye/left/camera_info	CameraInfo	19.83	**0.66**
/mynteye/left/image_raw	CompressedImage	19.83	**0.66**
/mynteye/right/camera_info	CameraInfo	19.83	**0.66**
/mynteye/right/image_raw	CompressedImage	19.83	**0.66**
/realsense/accel/sample	Imu	63.29	1.00
/realsense/gyro/sample	Imu	198.39	0.99
/realsense/aligned_depth/camera_info	CameraInfo	26.54	**0.88**
/realsense/aligned_depth/image_raw	CompressedImage	26.54	**0.88**
/realsense/color/camera_info	CameraInfo	28.88	0.96
/realsense/color/image_raw	CompressedImage	28.88	0.96
/realsense/infra1/camera_info	CameraInfo	16.88	**0.56**
/realsense/infra1/image_rect_raw	CompressedImage	16.88	**0.56**

In this dataset only the RTAB-Map node was used to perform SLAM, but with three different set-ups:

- RGBD odometry from the Realsense D435i fused with the Xsens IMU in *robot_localization*
- Livox odometry odometry fused with the Xsens IMU in *robot_localization*
- RGBD odometry from the Realsense D435i and Livox odometry fused with the Xsens IMU in *robot_localization*

Results and Discussion

The apparatus proved to be robust during the test experiment as the several sensors were firmly secure before and after the experiment. In this 5 minute test run the user also did not report any significant disconfort.

According to Table 4.1, the **relative topic frequency** of the Realsense D435i's infrared and aligned_depth topics are too low. The aligned_depth compression was set too high and the computer did not have the capability of compressing and recording the

Table 4.2: Results for the different set-ups.

SLAM	Sensor Fusion	Input for Sensor Fusion	RTE (m)	ROE (rad)
RTAB-Map	robot_localization	IMU; RGBD odom	52.5	0.32
RTAB-Map	robot_localization	IMU; Livox odom	28.8	3.3
RTAB-Map	robot_localization	IMU, RGBD odom, Livox odom	**0.05**	**0.14**

messages at the desired rate. The drop in frequency of the infrared topic does not share the same explanation, as subsquent tests showed that reducing the compression rate did not affect the topic frequency. An alternative hypothesis for this phenomenon is that the Udoo Bolt CPU board does not have enough air circulation in the backpack, causing it to reach the thermal threshold and start throttling the CPU. This hypothesis is tested in the next preliminary test.

In this experiment, the performance of two individual sensors, the Realsense D435i and the Livox, was evaluated as sources of odometry. The Realsense D435i odometry is computed with an ORB based algorithm and the Livox uses the sensor registration from *livox_mapping* to compute the odometry. Both sensor odometry estimates are then fused with an IMU with an UKF, outputing the fused odometry used in this experience. The results from Table 4.2 revealed that the individually odometry from both sensors performed poorly and produced inconsistent maps, as seen in Figures 4.4 and 4.5. The odometry from Realsense D435i struggled in low-feature environments, such as staircases, leading to an incoherent map. Despite the implementation of a matching planes algorithm in the scan registration, the Livox odometry also failed to produce a coherent map, introducing large errors in pitch caused by poor plane matching, which resulted in the rotation of sections of the map (Figure 4.5) and large error in ROE.

To address these issues, the odometry from both sensors was fused. The integration of the RGBD odometry significantly reduced the large errors in pitch present in the Livox odometry. Meanwhile, the Livox odometry helped to reduce the positional drift from the

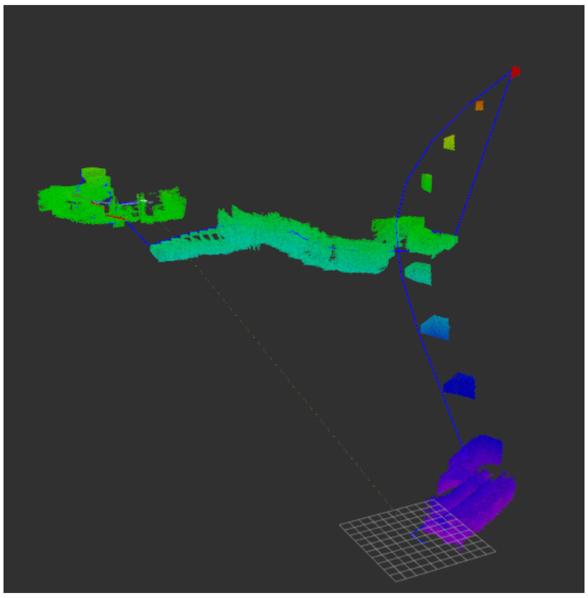


Figure 4.4: Complete map produced by RTAB-Map with only RGBD odometry in the DEEC dataset.

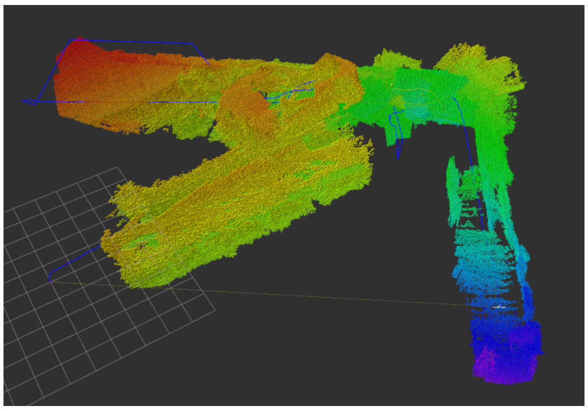


Figure 4.5: Complete map produced by RTAB-Map with only Livox odometry in the DEEC dataset.

RGBD odometry in low-feature sections of the trajectory, resulting in a more consistent map. The improved consistency of the map allowed RTAB-Map to identify both local and global loop closures, as seen in Figure 4.6 and Figure 4.7.

However, the overall quality of the generated map was not optimal, with some sections of the straight corridors appearing distorted, as illustrated by Figure 4.7. This was attributed to the presence of low-feature sections in the trajectory, which still introduced large errors in pose estimation. Despite these limitations, the integration of both sensors demonstrated improved results compared to the use of each sensor odometry individually. In order to obtain a more consistent map, new features need to be added to the staircase or a different sensor should be used, for example, a precise precise 360° LiDAR would improve significantly the plane matching algorithm.

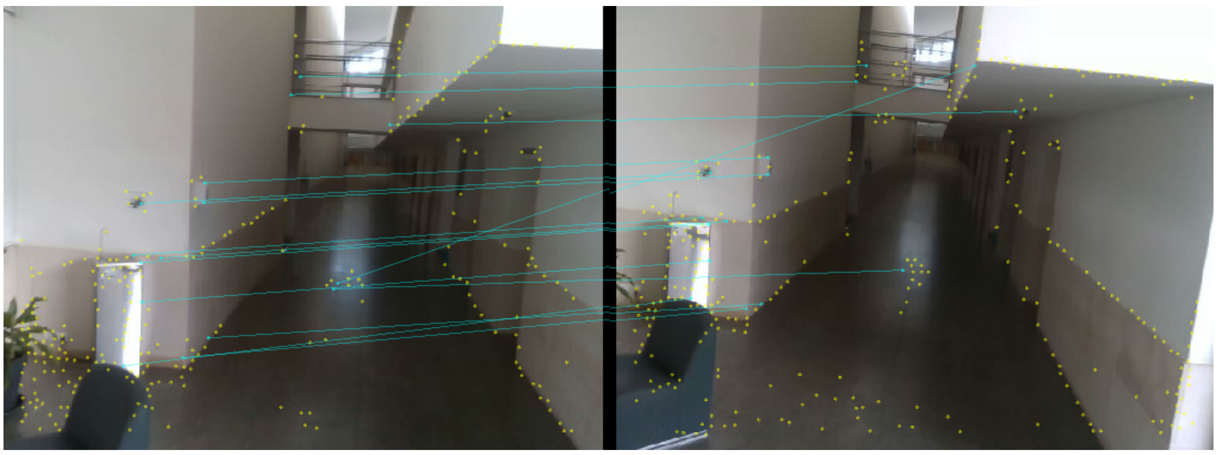


Figure 4.6: Amount of co-related features between initial and final position found in the loop closure detected by RTAB-Map in the DEEC dataset.

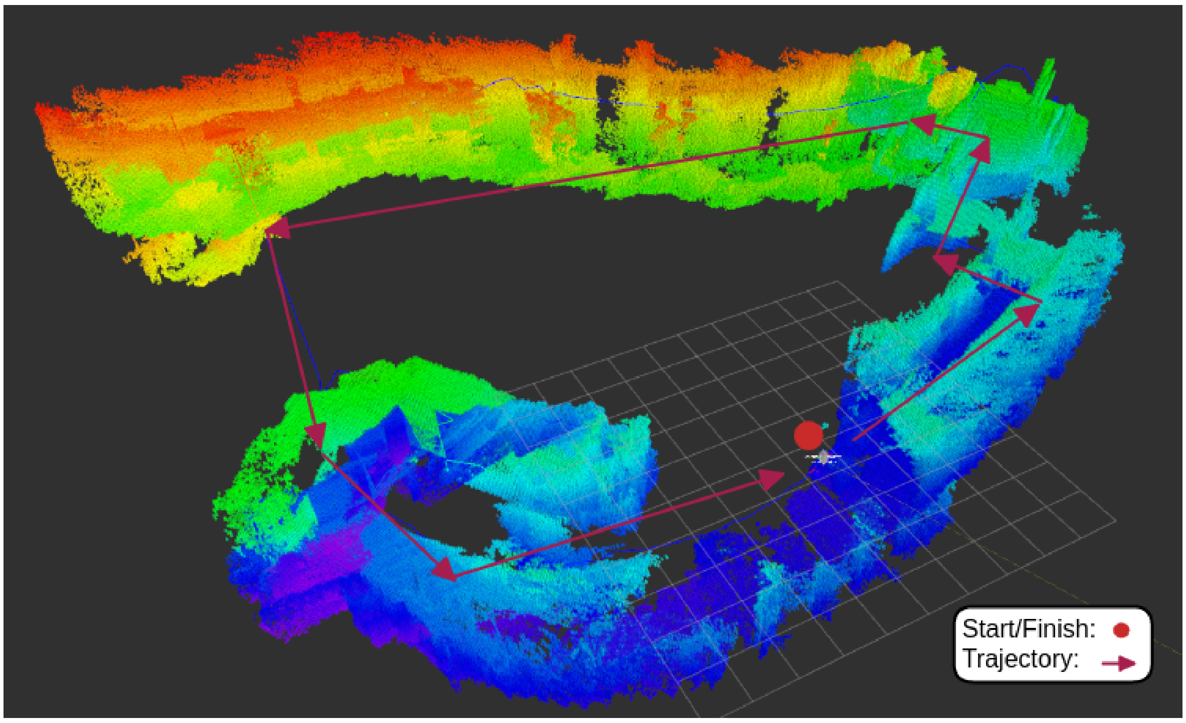


Figure 4.7: Complete map of the DEEC dataset generated with RTAB-Map with fused odometry.

4.3 Heat Test

This experiment was designed to provide further insight of the problematic frequencies discussed in the previous section. The main purpose of this test is:

- Verify if the Udoo Bolt computer starts to thermal throttling and verify if this is the cause of the unsteady acquisition frequencies observed during the preliminary experiences.

Experimental Design

The experiment includes four different scenarios:

- Computer with only the rosnode to record temperatures running
  - Backpack open
  - Backpack closed
- Entire recording architecure running
  - Backpack open
  - Backpack closed

In each scenario, the computer is left running for one hour, with measurements of temperature, CPU frequency, and topics frequencies extracted at 1-second intervals. The two scenarios where the computer is only running the rosnode to check the temperature are the control scenarios of the experiment.

Results and Discussion

The results displayed in Figure 4.8 suggest that the temperature of the Udoo Bolt is higher in the control configuration with a closed backpack, with an average temperature of $39.07°C$ and $39.87°C$ for the open and closed backpack respectively. Both of the temperatures of the control are lower than the configuration with the entire recording architecure running.

The experiment with the open backpack and the recording system running shows that the temperature stabilizes at about 46° Celsius (see Figure 4.8). Although there

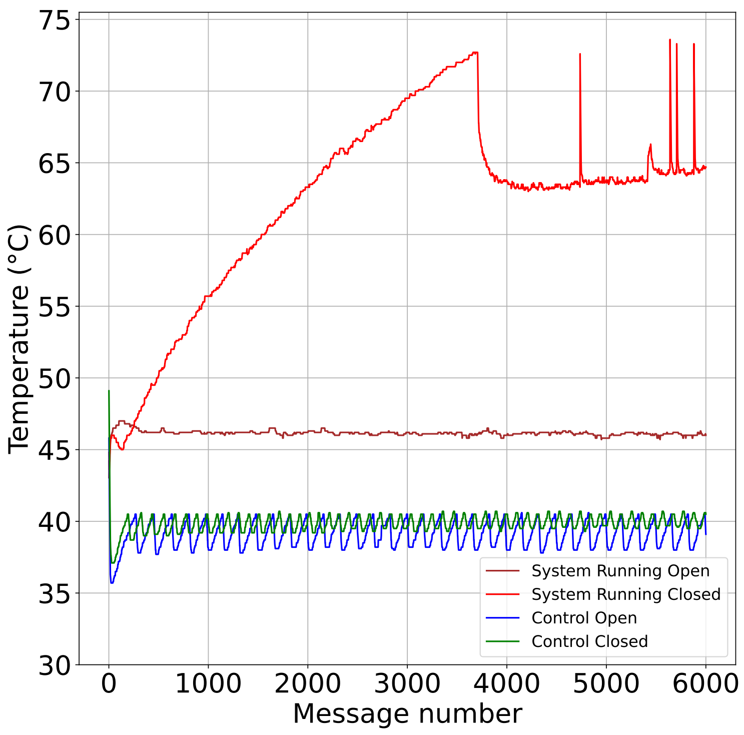


Figure 4.8: Temperature of all the set-ups of the experiment.

is no significant frequency drop throughout the experiment, the infrared image from the Realsense D435i was not published at all and the frequency of publication of the depth image appears to slowly decrease over time (see Figure 4.9). In contrast, in the closed backpack experiment, there does not appear to be enough air circulation to stabilize the temperature, reaching the close to the Udoo Bolt critical temperature of 75° Celsius, causing the CPU to throttle, (as shown in Figure 4.10). There is a significant drop in the frequency of the Mynt Eye streams that appears to correlate with temperature, thus supporting the original hypothesis. However, this hypothesis does not explain the increase in Realsense D435i frequency after the system begins to throttle, or why the infrared channel does publish data regardless of the configuration.

One hypothesis that could explain the results obtained is that the Udoo Bolt does not have enough bandwidth to pass all the information over the bus, since all USB ports share the same bus. This might explain the absence of the Realsense D435i's infrared channel, and why reducing the frequencies of the Mynt Eye camera leads to an increase in the Realsense D435i's depth frequency. To test if the Udoo bolt has enough bandwidth

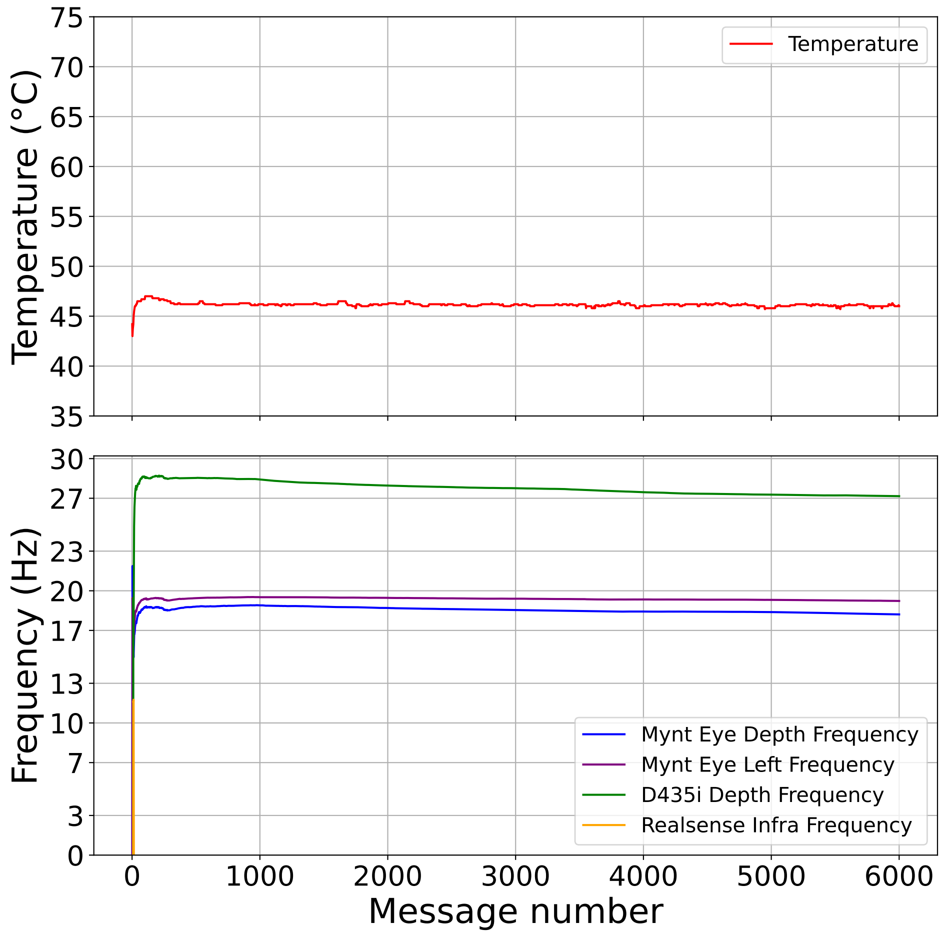


Figure 4.9: Results with entire system running and the backpack open.

(BW) to use all sensors simultaneously, one can make a simple estimate of the amount of information going over the bus and check the maximum bandwidth available. All USB ports in the Udoo Bolt share a single bus, with 16 PCIe lanes (Gen3). This means that, according to the specification of PCIe Gen3, the theoretically available bandwidth of the bus is $15.8GB/s$ in total, $7.877GB/s$ to each direction.

$$BW_{USB\rightarrow CPU} \approx 7.877GB/s \tag{4.5}$$

The sensors that transmit more information are the cameras and LiDAR (note that the LiDAR is connected via Ethernet, which does not use the same bus) and the bandwidth

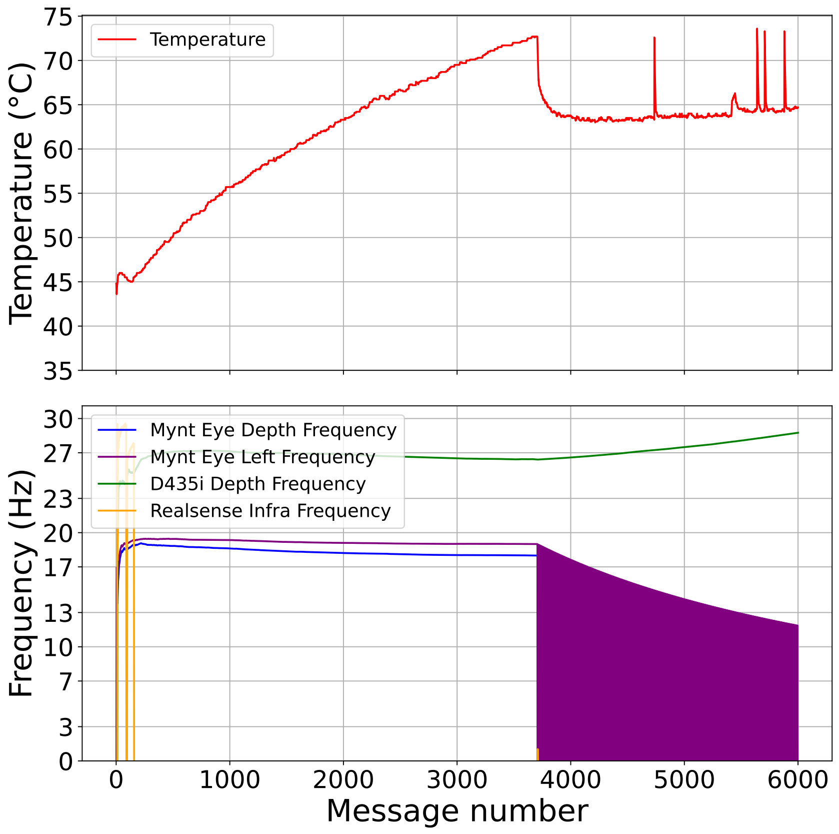


Figure 4.10: Results with entire system running and the backpack closed. The purple area painted under the curve is caused by the Mynt Eye ocasionally not publishing any message during the time windows used to compute the frequency, this led to a value of -1.

that each sensor requires is the size of the data times the sensor frequency:

$$BW = Data_{size} \times f. \tag{4.6}$$

The amount of data in each message from a camera can be estimated as the product of the image array size ($W \times L$) with the size of each element (d_{len}) and the number of channels ($N_{channels}$) in the image:

$$Data_{size} = W \times L \times N_{channels} \times d_{len}. \tag{4.7}$$

The Mynt Eye provides streams such as depth, rect and disparity, but these streams are computed in the host computer, the only images that pass through USB are the left and right images:

$$BW_{\text{Mynt Eye}} = 2 \times (752 \times 480 \times 1 \times 1byte) \times 20Hz \times 10^{(-9)} \approx 0.0144GB/s \tag{4.8}$$

The Realsense D435i does not compute the images on the host computer, so all streams must be considered, namely RGBD, right infrared and left infrared.

$$BW_{D435i} = 6 \times (640 \times 480) \times 30Hz \times 10^{(-9)} \approx 0.0555GB/s \tag{4.9}$$

The total bandwidth required by the cameras is only $69.9MB/s$, which is two orders of magnitude below the theoretical maximum, so it is unlikely that the issue is due to a bandwidth issue.

It has also been hypothesized that the Mynt Eye is taking over the bus controller and starving other devices. Alternatively, it could be that the CPU does not have enough processing power to process all the information and record the topics into a rosbag or that the Udoo Bolt cannot provide the necessary power to operate the two cameras at the same time. These three hypotheses are not tested in this dissertation. However, they are mentioned as **Future Work** in the last chapter, along with suggestions for profiling the problem.

4.4 Choupal National Woods Dataset

This section outlines the main objectives for the main dataset of the dissertation, the performance metrics used to evaluate the apparatus in chapter 5, and the experimental design conditions for recording the dataset.

The objectives of this experiment are:

- Test the confort for the user during long experiments;
- Test the mechanical robustness of the apparatus;
- Produce a dataset that can be used by the community to test several forest operations;
- Have a complete dataset that enables to integrate several SLAM algorithms;
- Validate the performance of the apparatus in a forest environment.

Performance Metrics

The performance metrics utilized in this study will be consistent with those presented in Section 4.2. Given the absence of a highly precise ground truth system, such as a high-precision GNSS system, a set of qualitative metrics will be employed to draw conclusions. These metrics include the overlay of the generated localizations onto maps from O-solutions [58], a professional cartography business, in which the accuracy of the map is not publicly accessible.

Several visual comparisons of the map generated by the SLAM implementations are also reviewed, with details such as global map consistency, amount of features and ground consistency.

Experimental Design

A dataset[10] was collected at the Choupal National Woods ($40°13'13.3''N$; $8°26'38.1''W$). As it shares resemblence to a forest environment (see Figure 4.11) and this is the main dataset used to validate the work.

In this study, a dataset was collected during a partly clouded day in a forest environment, where the user performed two circular loop laps amounting to a distance of approximately 800m (as depicted in Figure 4.11) with a duration of 24 minutes and 26 seconds. The environment is rich in features, including tree trunks, trees, bushes, and leaves. To evaluate the performance of different SLAM approaches, three different methods were compared:

- *livox_mapping*, with LiDAR data as the only input;
- RTAB-Map, odometry from *robot_localization* fusing the IMU, RGBD and Livox odometry together. The only sensor used to map the enviroment was the Realsense D435i camera;
- Cartographer, with raw IMU data and odometry from *robot_localization* by fusing the IMU, RGBD and Livox odometry together. The only input to map was the Livox LiDAR.

[10]The main dataset is available for download at `https://home.mycloud.com/action/share/36a0bad2-a4cc-43f0-b358-7cf97601f30b`

RTAB-Map requires RGB information in order to detect loop closures, making the utilization of the Realsense D435i camera essential. Although it is possible to use multiple depth sensors in RTAB-Map, using the Livox LiDAR did not produce any significant change, and using the Mynt Eye deprecated the map consistency, introducing errors, specially in the time windows where the Mynt Eye was failing to publish data.

Cartographer algorithm can take several point clouds as inputs. However, using either of the Realsense D435i or Mynt Eye cameras with the high precision LiDAR is not feasible due to the sparse point cloud produced by the LiDAR, which would be lost in the dense and noisy data from either the Mynt Eye or the Realsense D435i. The next chapter presents a detailed discussion on the performance of the apparatus in this experiment.

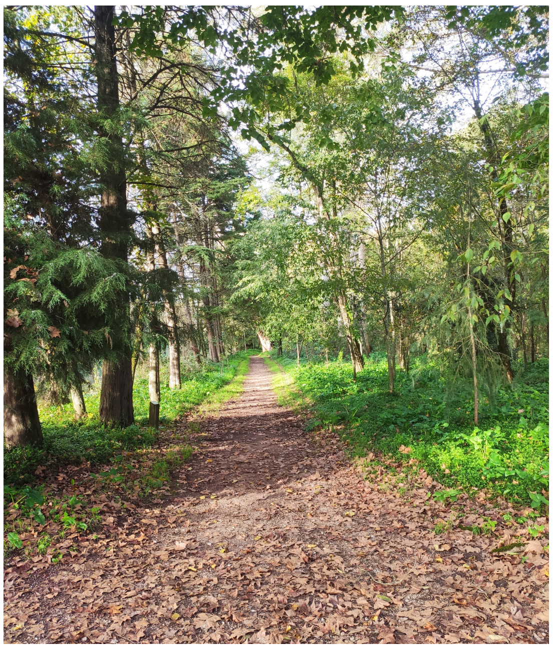


Figure 4.11: An example space in the Choupal National Woods.

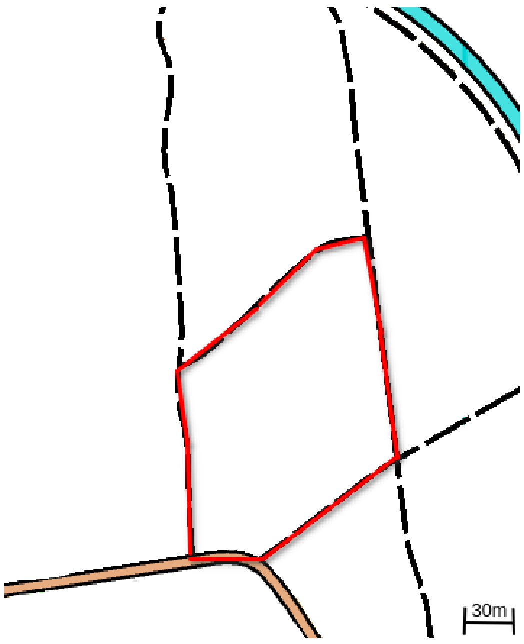


Figure 4.12: Ground truth trajectory in the Choupal Dataset, using the map provided by [58].

5 Results and Discussion

The objectives of this chapter are to analyze and interpret the results obtained from the research study, and to discuss the implications of these findings.

As the experiment progressed, the apparatus demonstrated its robustness as the transformations between the sensors remained relatively unchanged. However, the user reported some discomfort during extended use of the apparatus, which was attributed to the uneven and off-center design of the backpack and its weight. Aligning the pole with the center of the user's back may help to alleviate these issues by improving weight distribution and reducing lateral movement (which reduces the cameras motion blur). Therefore, this should be considered and implemented in an improved version of the backpack.

Unlike previously reported in the preliminary tests, the results show that the GNSS information provided by the Xiaomi Mi Mix 3 device was unreliable, with large errors in the positioning data (as seen in Figure 5.1). The reason for this could be the limited number of satellites available to the device at the time of the experiment. However, Figure 5.2 indicates that eight GNSS satellites were available when the cut-off was set at 45%. It is possible that the decreased precision was a result of the phone being carried in a pocket during the experiment, which could have impacted its performance. This issue will be further addressed in the future work.

In the absence of GNSS data, only a relative sources of localizations are available. The results of the different odometry methods are displayed in Figures 5.3 and 5.4. An analysis of these results reveals that two of the odometry sources, acquired using the Mynt Eye camera, do not provide accurate enough results. The low frame rate of Mynt Eye messages is considered the cause for the inconsistency of the Mynt Eye odometry sources. On the other hand, the Realsense D435i, which performs well in environments rich in features, shows a satisfactory performance, although it still experiences a substantial drift in the Z-axis. Similarly, the sparse but high-precision LiDAR shows adequate results in

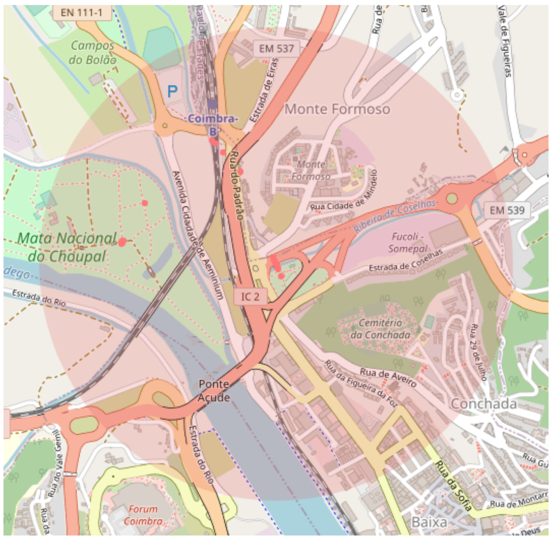


Figure 5.1: GNSS locations for the Choupal dataset. The small red circles are the several positions received during the experiment, and the semi-transparent red circle is the uncertainty radius of one of the positions.

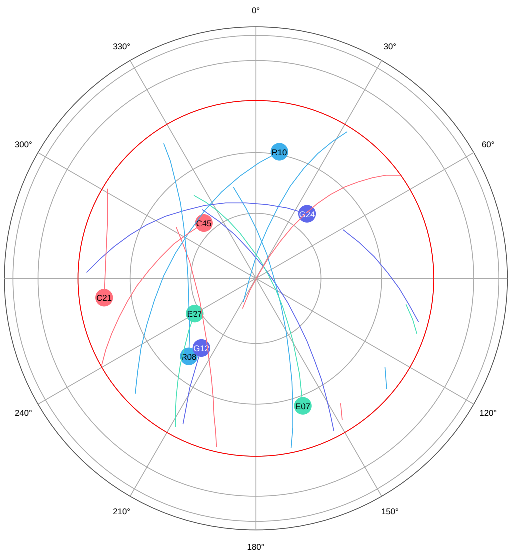


Figure 5.2: Number of GNSS satellites available at the time of the experiment according to [59]. The satellites outside the red circle (a 45° cut-off) are not considered as visible.

the initial stages of the experiment but experiences degradation in performance half way through the second lap.

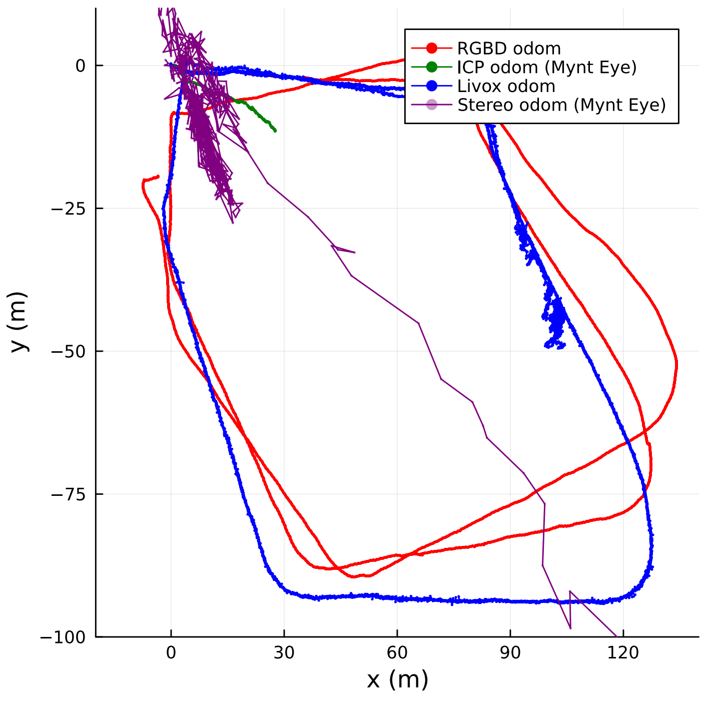


Figure 5.3: 2D plot of the odometry provided by every sensor.

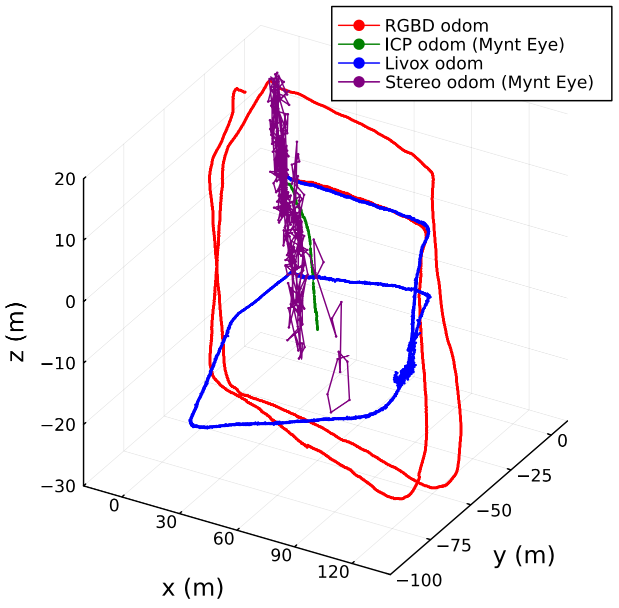


Figure 5.4: 3D plot of the odometry provided by every sensor.

The different localizations for the SLAM approaches are shown in Figure 5.5 and 5.6. Although the *livox_mapping* library, which is based on LOAM, appears to produce

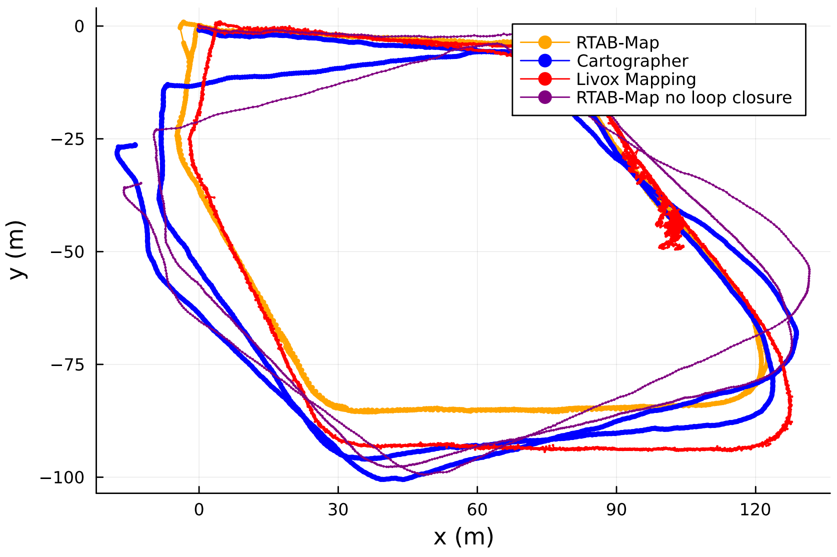


Figure 5.5: 2D plot of the localization estimated by the SLAM approaches.

less drift during the first lap, it cannot perform loop closure. Consequently, the drift that occurs is never corrected, ultimately causing the failure to complete the second lap. Further investigation revealed that the failure of *livox_mapping* was due to the overlapping of points from the first and second lap, leading to incorrect matches and clusters of false points, as demonstrated in Figure 5.7. In order to test this hypothesis, a subsequent experiment was performed using only the second lap, which confirmed the initial claim.

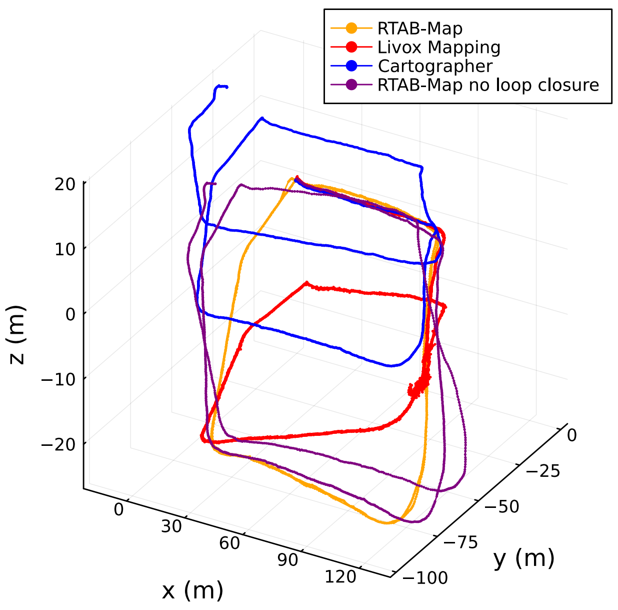


Figure 5.6: 3D plot of the localization estimated by the SLAM approaches.

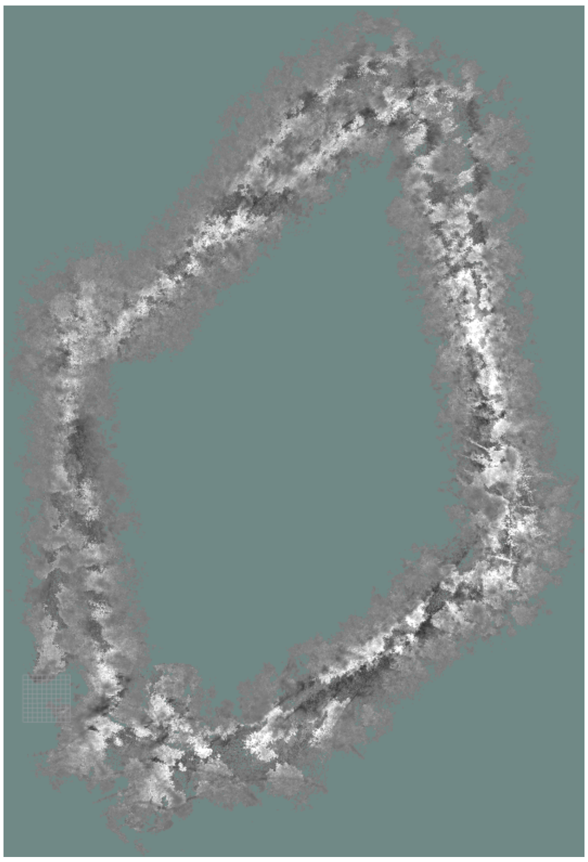


Figure 5.8: The 2D map generated by Cartographer.

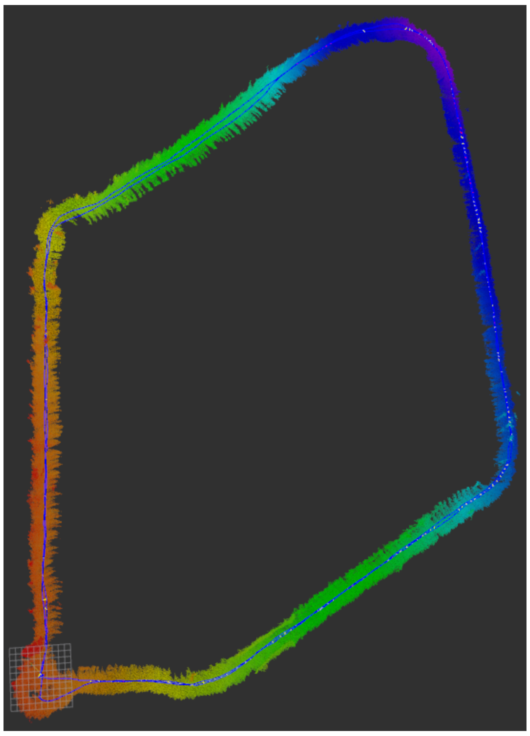


Figure 5.9: 3D map generated by RTAB-Map (top-down view).

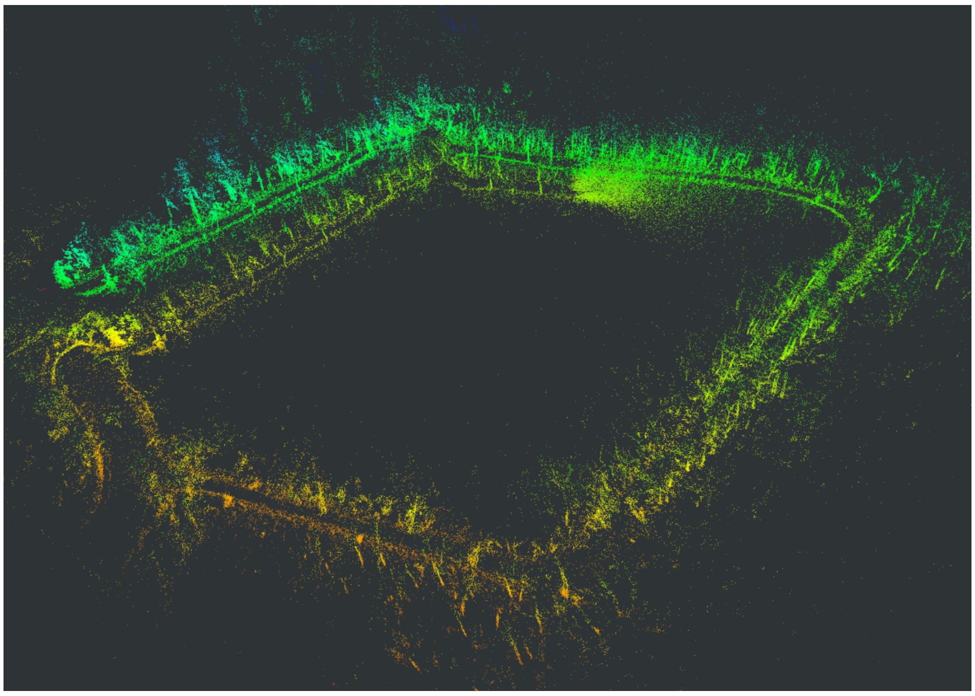


Figure 5.7: 3D point cloud map generated by *livox_mapping*.

In the evaluation of various SLAM implementations, it was found that RTAB-Map was the only method to successfully achieve loop closure detection, as demonstrated in Figure 5.9. The results of this approach, as presented in Table 5.1, indicate a significant reduction in both RTE and ROE errors. To provide a comparison of the drift produced by RTAB-Map and Cartographer, an experiment with loop closure disabled was conducted. The disparity between the results of both RTAB-Map experiments highlights the crucial role of loop closure in correcting drift error. While some discrepancies were observed in the results produced by Cartographer when compared to reality, this can partially be attributed to the metrics used, which do not fully encompass the details of the experiment. However, for the present study, the chosen metrics are considered the most relevant for the proposed apparatus.

Although Cartographer has the ability to add loop closure constraints to the generated map, it was unable to find them in the experiment with the set of parameters used. Fine-tuning of the local SLAM parameters was necessary to achieve a desirable mapping result, as illustrated in Figure 5.8. During the tuning process, the parameter that establishes how much Cartographer can change the orientation of the prior observation (information from odometry and IMU) was significantly decreased as it was causing heavy warping of the map. As the Livox LiDAR was the only sensor utilized for mapping, the number of

Table 5.1: Relative errors for the different SLAM approaches.

	livox_mapping	RTAB-Map	RTAB-Map without loop Closure	Cartographer
σ_x (m)	103.5	**0.14**	12.5	13.8
σ_y (m)	49.2	**0.10**	34.7	25.5
σ_z (m)	13.0	**0.26**	5.8	18.6
RTE (m)	115.4	**0.31**	37.4	34.4
σ_{roll} (rad)	0.38	0.12	0.13	**0.05**
σ_{pitch} (rad)	0.14	**0.01**	0.09	0.08
σ_{yaw} (rad)	1.47	**0.0002**	0.43	0.45
ROE (rad)	1.52	**0.12**	0.45	0.46

accumulated point clouds required for adding a new constraint for the local SLAM was also adjusted. To strike a balance between map consistency and localization update rate, the parameter was set to 20 point clouds. With the Livox operating at its maximum frequency of 50Hz, this means that the localization is updated at most every 0.4 seconds. Decreasing this value would lead to poor results as the point cloud would become too sparse to compute the scan matching algorithm successfully.

RTAB-Map is the approach that yields the best results according to Table 5.1. The *livox_mapping* implementation has the greatest error, as expected since it did not manage to complete the second lap. Table 5.1 also illustrates that the performance of Cartographer is comparable to that of RTAB-Map with loop closure disabled. This finding suggests that further refinement of both local and global SLAM parameters could lead to substantial improvement in the results generated by Cartographer. Cartographer's errors in pitch and roll angles are also minimal, which is beneficial because correcting drift in translation is relatively easy, while correcting large drifts in orientation can be very challenging, reforcing the ideia that a better tunned integration of Cartographer has the potential to yield better results.

Additionally, the validity of the results is also assessed by comparing the localization generated by each method to a known map (refer to Figure 5.10). As the accuracy of the maps is unknown, a quantitative inference regarding their proximity to reality is

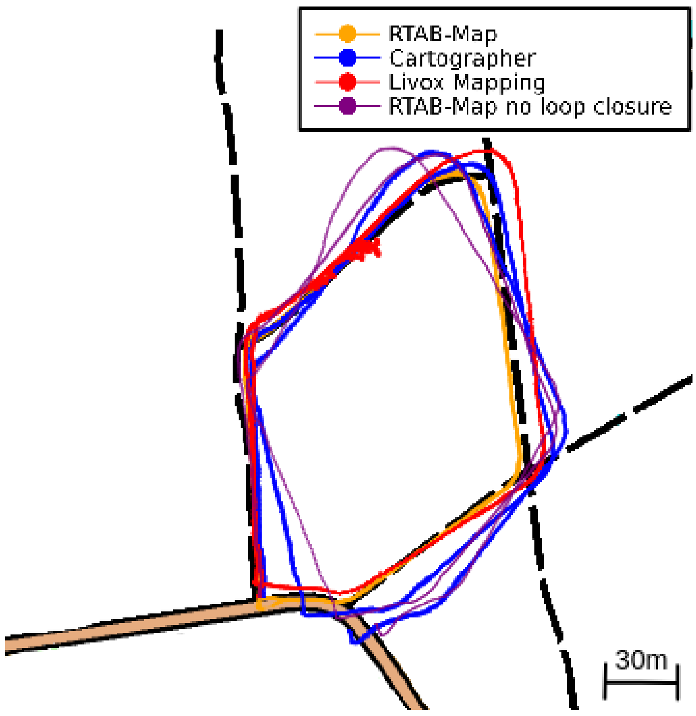
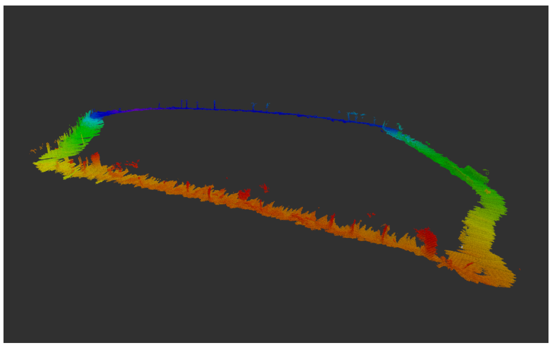


Figure 5.10: Overlaying the path generated by the SLAM on the map adapted from [58].

challenging. However, the localizations generated by each SLAM approach were found to overlap well with the map depicted in Figure 5.10, suggesting a relatively high level of accuracy in the localization obtained by the apparatus, specially by RTAB-Map.

The quality of the maps generated by Octomap using the localization from each SLAM algorithm was analyzed and depicted in Figure 5.11. The Realsense D435i depth camera was utilized to provide depth information to Octomap. Due to the low publishing frequency of the localization provided by Cartographer, the resulting map was found to be less dense, as Octomap was unable to determine the position of the incoming point clouds between locations. This has significant implications on the overall map quality produced by Cartographer. Both localizations were capable of mapping the various trees encountered throughout the trajectory in close proximity to the apparatus. The absence of a functional Mynt Eye resulted in a significantly reduced mapping area in both cases.

Ground consistency and density are crucial characteristics of a map for effective path planning and identification of flammable materials. The ground generated by RTAB-Map was found to be unexpectedly dense and well-connected between successive point cloud measurements, appearing to consistently reside on the same plane, as shown in Figure 5.13a. On the other hand, Cartographer, as illustrated in Figure 5.13b, did not produce


(a) RTAB-Map.

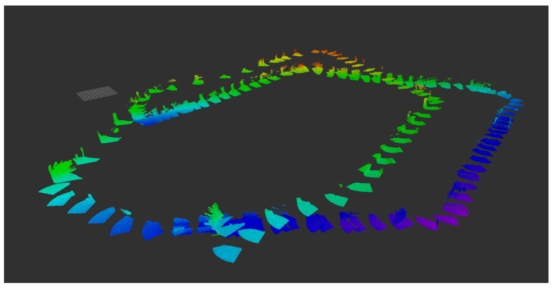
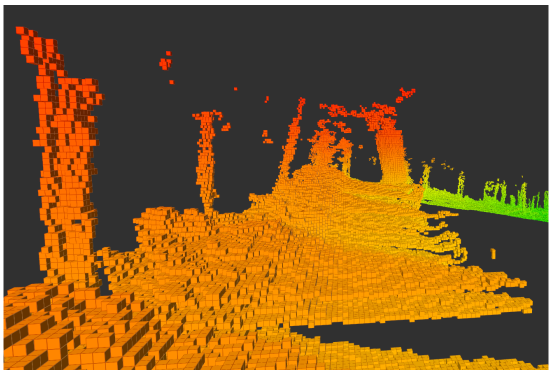

(b) Cartographer.

Figure 5.11: Complete octomap representation using the localization provided by RTAB-Map and Cartographer, with the Realsense D435i as the mapping sensor.


(a) RTAB-Map.

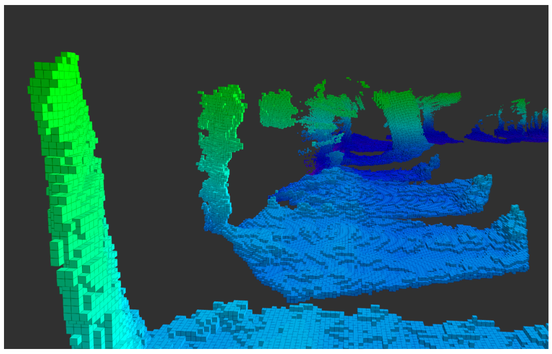
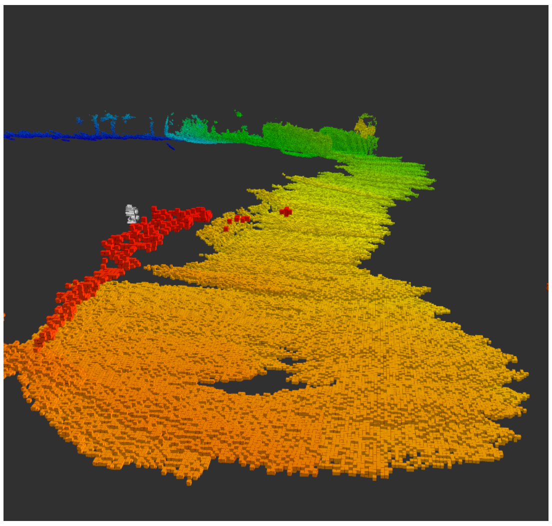

(b) Cartographer.

Figure 5.12: Illustration of relevant details, e.g. trees, using the localization provided by RTAB-Map and Cartographer, with the Realsense D435i as the mapping sensor.

similarly accurate results, which can be attributed to the previously discussed low frequency of localization and consequent low update rate of the map. Figure 5.14 presents a RGBD section of the map produced by RTAB-Map, where several trees are mapped. This is also a good example of the ground consistency obtained with this approach, as it is easy to differentiate the path from the green ground foliage.

In conclusion, the findings of this study offer valuable insights into the potential performance of the proposed apparatus when utilizing different SLAM implementations, as well as the challenges associated with these approaches. The results indicate that the apparatus is robust, as the localization and mapping produced by RTAB-Map are accurate enough to support further experimentation in the field.


(a) RTAB-Map.

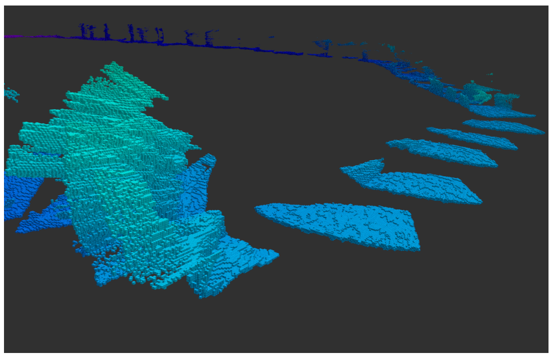

(b) Cartographer.

Figure 5.13: Octomap Map ground consistency using the localization provided by RTAB-Map and Cartographer, with the Realsense D435i as the mapping sensor.

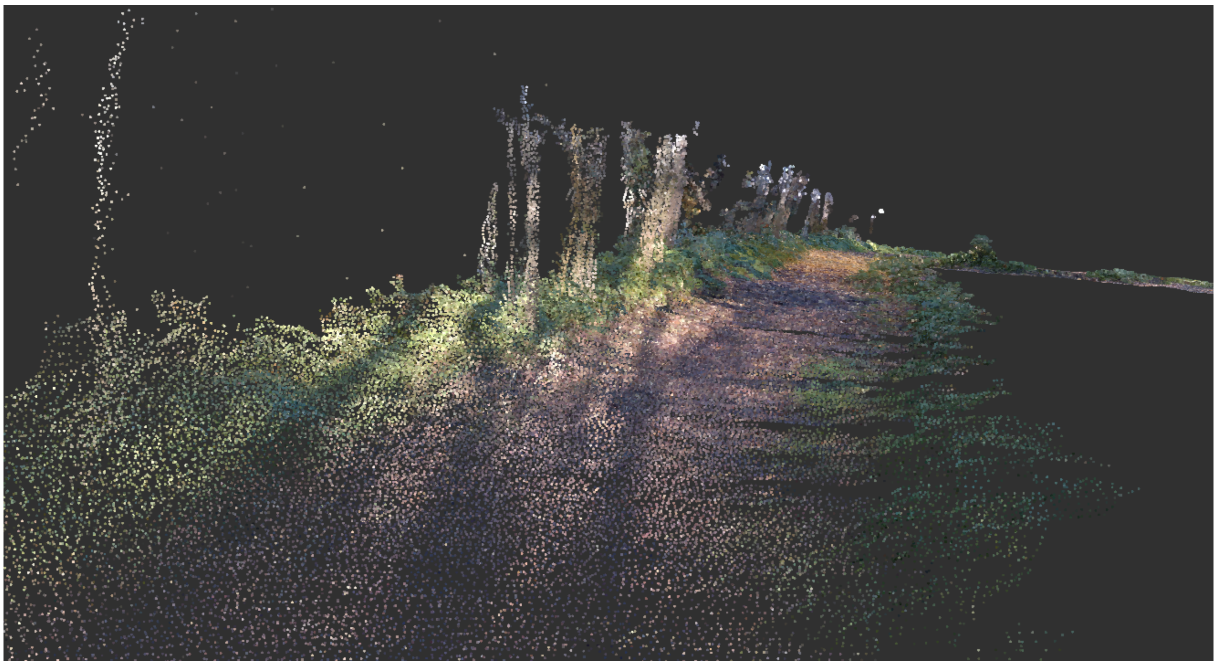


Figure 5.14: RGBD map produced by RTAB-Map in Choupal National Woods.

6 Conclusion

In this dissertation, a new modular and portable multisensory apparatus for the localization and mapping of forest environments has been presented. The apparatus has been implemented and evaluated in real-world conditions, demonstrating its capability to record necessary information to extract localization and generate accurate estimates of its surroundings. This dissertation work produced the following contributions:

- A portable apparatus that can be used to support future research on forestry robotics at the ISR.
- A free, open-source Android application that is publically available. The app fills a gap in the field by transmitting data from Android devices into ROS.
- A multimodal sensing dataset in a forest environment with the potential of helping advancing knowledge in the field of forestry robotics openly shared with the robotics community.
- A comparison of state-of-the-art SLAM implementations using a multi sensor apparatus in a forest environment.

In this work, a mechanical robust structure was developed and paired with a modular recording architecture based in ROS and Docker. An independent Android application was also developed to facilitate communication between a smartphone and the Robot Operating System (ROS). An architecture to implement several SLAM methods in the recorded dataset was also inegrated into the work. Finally, the SLAM methods were compared against each other, acheiving results that supported the viability of the apparatus developed.

Even though, the majority of the requirements outlined in the MoSCoW analysis of the work were successfully fullfield, three requirements were not fully satisfied. The publishing rate of Mynt's Eye relevant topics was insufficient to meet the expected requirements. Although not as important, the lack of a dedicated space for the Android device and live visual feedback were also noted as limitations. The latter was not feasible due to the high CPU utilization required for visual representation.

6.1 Future Work

This study highlights several areas for potential improvement in the current system. Further experimentation is required to fully understand the source of the Mynt Eye performance issue. Initial analysis indicates that a high utilization of CPU may be responsible for this issue, as a significant drop in CPU utilization is observed. To diagnose this issue, an experiment can be designed to run the system without recording *rosbags*, as this operation heavily stresses the CPU. If this is the case, a patch can be implemented in the Mynt Eye driver to reduce computational demands. Another possibility is that the Udoo Bolt is unable to sufficiently power all cameras simultaneously, which can be further investigated using Linux utilities such as *usb-devices* and *powertop.* The possibility of the Mynt Eye requesting the USB controller to itself, starving the Realsense D435i also requires further investigation. A possible solution to this issue is to use an additional board to receive data from the Mynt Eye and transmit it to the Udoo Bolt via Ethernet.

The acquisition of an absolute localization sensor with high-precision GNSS, e.g. a GPS-RTK base station, is also identified as a potential improvement for assessing the robustness of the apparatus, as it could be used as a source for positioning ground truth. Furthermore, the findings of Chapter 3 indicate a need for enhancing the registration of the various sensors and implementing software to determine the sensor transforms automatically. This software should be designed to maintain the modularity of the system, and should be implemented independently of the recording architecture. The Android application developed in this study could also be improved by incorporating the capability to send raw GNSS and orientation data. The experiments done to test the Android Applications together with the apparatus should be repeated to confirm that the decreased precision was due to the positioning of the device inside the user's pocket. To improve SLAM performance, all sensors should be integrated into the mapping process. However, as the messages from the different sensors arrive at different instants of time, a synchronization software should be used to address timing issues. Additionally, since the publishing rate of the multiple sensors can greatly differ and exceed the map update rate, a software to estimate the pose at any given timestamp should be developed, especially for the Livox LiDAR.

From a mechanical perspective, the *sensor box* could be re-engineered to incorporate PTFE material, which is more resistant to UV radiation. Additionally, a designated slot for Android devices could be integrated into the revised design of the *sensor box*. Further experimentation is necessary to confirm the superiority of the backpack design in terms of user comfort. Additionally, a design that offers the user the option of carrying the *sensor box* by hand or using the backpack could also be considered for future improvement. The cooling solution of the backpack should also be re-evaluated.

Bibliography

[1] M. Turco, J. Bedia, F. Di Liberto, P. Fiorucci, J. von Hardenberg, N. Koutsias, M.-C. Llasat, F. Xystrakis, and A. Provenzale, "Decreasing Fires in Mediterranean Europe," *PLOS ONE*, vol. 11, p. e0150663, Mar. 2016.

[2] European Commission. Joint Research Centre., *Forest Fires in Europe, Middle East and North Africa 2020.* LU: Publications Office, 2021.

[3] R. Parker, K. Bayne, and P. Clinton, "Robotics in forestry," *New Zealand Journal of Forestry*, vol. 60, pp. 8–14, Feb. 2016.

[4] M. S. Couceiro, D. Portugal, J. F. Ferreira, and R. P. Rocha, "SEMFIRE: Towards a new generation of forestry maintenance multi-robot systems," in *2019 IEEE/SICE International Symposium on System Integration (SII)*, pp. 270–276, Jan. 2019. ISSN: 2474-2325.

[5] E. Jelavic, D. Jud, P. Egli, and M. Hutter, "Towards Autonomous Robotic Precision Harvesting: Mapping, Localization, Planning and Control for a Legged Tree Harvester," Tech. Rep. arXiv:2104.10110, arXiv, Nov. 2021. arXiv:2104.10110 [cs, eess] type: article.

[6] T. L. Lam and Y. Xu, "A flexible tree climbing robot: Treebot - design and implementation," in *2011 IEEE International Conference on Robotics and Automation*, pp. 5849–5854, May 2011. ISSN: 1050-4729.

[7] G. Notomista, Y. Emam, and M. Egerstedt, "The SlothBot: A Novel Design for a Wire-Traversing Robot," *IEEE Robotics and Automation Letters*, vol. PP, pp. 1–1, Feb. 2019.

[8] Milrem Robotics, “Multiscope Forester Planter.” `https://milremrobotics.com/product/robotic-forester-planter/`.

[9] J. Molina and S. Hirai, “Aerial pruning mechanism, initial real environment test,” *Robotics and Biomimetics*, vol. 4, no. 1, p. 15, 2017.

[10] C. Zhang, L. Yong, Y. Chen, S. Zhang, L. Ge, S. Wang, and W. Li, “A Rubber-Tapping Robot Forest Navigation and Information Collection System Based on 2D LiDAR and a Gyroscope,” *Sensors (Basel, Switzerland)*, vol. 19, p. E2136, May 2019.

[11] S. Thrun, “Learning metric-topological maps for indoor mobile robot navigation,” *Artificial Intelligence*, vol. 99, pp. 21–71, Feb. 1998.

[12] G. Lozenguez, L. Adouane, A. Beynier, A.-I. Mouaddib, and P. Martinet, “Punctual versus continuous auction coordination for multi-robot and multi-task topological navigation,” *Autonomous Robots*, vol. 40, pp. 599–613, Apr. 2016.

[13] M. Labbé and F. Michaud, “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation,” *Journal of Field Robotics*, vol. 36, pp. 416–446, Mar. 2019.

[14] P. Fankhauser, M. Bloesch, and M. Hutter, “Probabilistic terrain mapping for mobile robots with uncertain localization,” *IEEE Robotics and Automation Letters (RA-L)*, vol. 3, no. 4, pp. 3019–3026, 2018.

[15] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, “OctoMap: an efficient probabilistic 3D mapping framework based on octrees,” *Autonomous Robots*, vol. 34, no. 3, pp. 189–206, 2013. Publisher: Springer Science and Business Media LLC.

[16] I.-S. Kweon, M. Hebert, E. Krotkov, and T. Kanade, “Terrain mapping for a roving planetary explorer,” in *IEEE International Conference on Robotics and Automation*, pp. 997–1002, IEEE, 1989.

[17] H. Bay, T. Tuytelaars, and L. Van Gool, “SURF: Speeded Up Robust Features,” in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3951, pp. 404–417, Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. Series Title: Lecture Notes in Computer Science.

[18] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *2011 International Conference on Computer Vision*, pp. 2564–2571, Nov. 2011. ISSN: 2380-7504.

[19] H. Liu, D. D. Huang, and Z. Y. Geng, "Visual Odometry Algorithm Based on Deep Learning," IEEE, 2021.

[20] J. Konecny, P. Kromer, M. Prauzek, and P. Musilek, "Scan Matching by Cross-Correlation and Differential Evolution," *Electronics*, vol. 8, no. 8, p. 856, 2019. Publisher: MDPI AG.

[21] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics.* Intelligent robotics and autonomous agents, MIT Press, 2005.

[22] S. Huang and G. Dissanayake, "Convergence and consistency analysis for extended kalman filter based slam," *IEEE Transactions on robotics*, vol. 23, no. 5, pp. 1036–1049, 2007.

[23] D. Portugal, M. E. Andrada, A. G. Araújo, M. S. Couceiro, and J. F. Ferreira, "ROS Integration of an Instrumented Bobcat T190 for the SEMFIRE Project," in *Robot Operating System (ROS)* (A. Koubaa, ed.), vol. 962, pp. 87–119, Cham: Springer International Publishing, 2021. Series Title: Studies in Computational Intelligence.

[24] S. Thrun, M. Montemerlo, D. Koller, B. Wegbreit, J. Nieto, and E. Nebot, "FastSLAM: An Efficient Solution to the Simultaneous Localization And Mapping Problem with Unknown Data Association," p. 48.

[25] X. Wang and H. Zhang, "A UPF-UKF Framework For SLAM," IEEE, 2007.

[26] F. Lu and E. Milios, "Globally Consistent Range Scan Alignment for Environment Mapping," *Autonomous Robots*, vol. 4, no. 4, pp. 333–349, 1997. Publisher: Springer Science and Business Media LLC.

[27] A. Juric, F. Kendes, I. Markovic, and I. Petrovic, "A Comparison of Graph Optimization Approaches for Pose Estimation in SLAM," in *2021 44th International Convention on Information, Communication and Electronic Technology (MIPRO)*, (Opatija, Croatia), pp. 1113–1118, IEEE, Sept. 2021.

[28] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G^{2}o: A general framework for graph optimization," in *2011 IEEE International Conference on Robotics and Automation*, (Shanghai, China), pp. 3607–3613, IEEE, May 2011.

[29] F. Dellaert, R. Roberts, V. Agrawal, A. Cunningham, C. Beall, D.-N. Ta, F. Jiang, lucacarlone, nikai, J. L. Blanco-Claraco, S. Williams, ydjian, J. Lambert, A. Melim, Z. Lv, A. Krishnan, J. Dong, G. Chen, K. Chande, balderdash devil, DiffDecisionTrees, S. An, mpaluri, E. P. Mendes, M. Bosse, A. Patel, A. Baid, P. Furgale, matthewbroadwaynavenio, and roderick koehle, "borglab/gtsam," May 2022.

[30] S. Agarwal, K. Mierle, and T. C. S. Team, "Ceres Solver," 3 2022.

[31] M. Quigley, "Ros: an open-source robot operating system," in *IEEE International Conference on Robotics and Automation*, 2009.

[32] G. Grisetti, C. Stachniss, and W. Burgard, "Nonlinear Constraint Network Optimization for Efficient Map Learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 10, pp. 428–439, Sept. 2009.

[33] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, IEEE, 2018.

[34] G. Kim and A. Kim, "Scan Context: Egocentric Spatial Descriptor for Place Recognition Within 3D Point Cloud Map," IEEE, 2018.

[35] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3d lidar inertial odometry and mapping," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2019.

[36] W. Xu and F. Zhang, "FAST-LIO: A Fast, Robust LiDAR-Inertial Odometry Package by Tightly-Coupled Iterated Kalman Filter," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3317–3324, 2021. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[37] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast Direct LiDAR-inertial Odometry," Tech. Rep. arXiv:2107.06829, arXiv, July 2021. arXiv:2107.06829 [cs] type: article.

[38] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and R. Daniela, “Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5135–5142, IEEE, 2020.

[39] W. Hess, D. Kohler, H. Rapp, and D. Andor, “Real-Time Loop Closure in 2D LIDAR SLAM,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1271–1278, 2016.

[40] “LiBackpack C50-GreenValley International.”

[41] “LiBackpack DGC50-GreenValley International.”

[42] I. Oveland, M. Hauglin, F. Giannetti, N. Schipper Kjørsvik, and T. Gobakken, “Comparing Three Different Ground Based Laser Scanning Methods for Tree Stem Detection,” *Remote Sensing*, vol. 10, no. 4, p. 538, 2018. Publisher: MDPI AG.

[43] K. Xiao, W. Yu, W. Liu, F. Qu, and Z. Ma, “High-Precision SLAM Based on the Tight Coupling of Dual Lidar Inertial Odometry for Multi-Scene Applications,” *Applied Sciences*, vol. 12, p. 939, Jan. 2022.

[44] A. Proudman, M. Ramezani, and M. Fallon, “Online Estimation of Diameter at Breast Height (DBH) of Forest Trees Using a Handheld LiDAR,” Tech. Rep. arXiv:2108.01552, arXiv, Aug. 2021. arXiv:2108.01552 [eess] type: article.

[45] Y. Su, Q. Guo, S. Jin, H. Guan, X. Sun, Q. Ma, T. Hu, R. Wang, and Y. Li, “The Development and Evaluation of a Backpack LiDAR System for Accurate and Efficient Forest Inventory,” *IEEE Geoscience and Remote Sensing Letters*, vol. 18, no. 9, pp. 1660–1664, 2021. Publisher: Institute of Electrical and Electronics Engineers (IEEE).

[46] H. Sier, L. Qingqing, Y. Xianjia, J. P. Queralta, Z. Zou, and T. Westerlund, “A Benchmark for Multi-Modal Lidar SLAM with Ground Truth in GNSS-Denied Environments,” Tech. Rep. arXiv:2210.00812, arXiv, Oct. 2022. arXiv:2210.00812 [cs] type: article.

[47] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim, "Complex Urban LiDAR Data Set," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, (Brisbane, QLD), pp. 6344–6351, IEEE, May 2018.

[48] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, "A benchmark for the evaluation of RGB-D SLAM systems," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, (Vilamoura-Algarve, Portugal), pp. 573–580, IEEE, Oct. 2012.

[49] Livox, "Mid-70 User Manual." `https://www.livoxtech.com/mid-70/downloads`, 2021.

[50] B. Tiganis, L. Burn, P. Davis, and A. Hill, "Thermal degradation of acrylonitrile–butadiene–styrene (abs) blends," *Polymer degradation and stability*, vol. 76, no. 3, pp. 425–434, 2002.

[51] N. Rottmann, N. Studt, F. Ernst, and E. Rueckert, "Ros-mobile: An android application for the robot operating system," *arXiv preprint arXiv:2011.02781*, 2020.

[52] S. Chitta, E. Marder-Eppstein, W. Meeussen, V. Pradeep, A. Rodríguez Tsouroukdissian, J. Bohren, D. Coleman, B. Magyar, G. Raiola, M. Lüdtke, and E. Fernández Perdomo, "ros_control: A generic and simple control framework for ros," *The Journal of Open Source Software*, 2017.

[53] C. Rockey, "Ros android driver." `https://github.com/ros-android/android_sensors_driver`, 2012.

[54] C. Crick, G. Jay, S. Osentoski, B. Pitzer, and O. C. Jenkins, "Rosbridge: Ros for non-ros users," in *Robotics Research*, pp. 493–504, Springer, 2017.

[55] D. Merkel, "Docker: lightweight linux containers for consistent development and deployment," *Linux journal*, vol. 2014, no. 239, p. 2, 2014.

[56] R. White and H. Christensen, "Ros and docker," in *Robot Operating System (ROS)*, pp. 285–307, Springer, 2017.

[57] T. Moore and D. Stouch, "A generalized extended kalman filter implementation for the robot operating system," in *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*, Springer, July 2014.

[58] Rafael Miguel, “O-Solutions - Choupal.” https://o-solutions.pt/wp-content/uploads/2020/11/17_Choupal-Coimbra-2019_10_05_RM.png, 2019.

[59] Navmatix, “GNSS Mission Planning .” https://www.gnssmissionplanning.com/, 2017.