Research Thesis

A Study of Spatial Mapping

An evaluative study of spatial map building algorithms for indoor use

Dortmund University of Applied Science and Arts

A thesis submitted to the Department of Computer Science study program Embedded Systems Engineering in partial fulfillment of the requirements for the degree of Master of Engineering

by
Aaron Xavier
born 22.01.1998
Matriculation number 7213765

Supervisors:
Prof. Dr.-Ing. Christof Röhrig
M.Sc. Merlin Stampa

Dortmund, August 27, 2023

Abstract

3D mapping may be defined as the process of building a map of an environment for the purpose of inspection, navigation, search and rescue and database archival etc. Modern sensors allow the accurate measurement of shapes and geometries of a space with the help of lasers but the advancement in camera technology has brought down the cost of image sensors substantially and thus they are becoming predominant in the field of perception.

The aim of this research work is to study the various SLAM algorithms available in the industry presently and compare them in terms of their accuracy, application domain, ease of setup and performance on a standard computing platform (ex. Nvidia Jetson TX2) and choose an appropriate algorithm for a specific application, in this case, Factory inspection.

This research focuses on a popular SLAM method, Real-Time Appearance Based Mapping (RTAB-Map), to approach the SLAM problem. It also performs the experiment on the same standard data sets and benchmarks each of the algorithms in terms of the accuracy of the pose, and structural geometry.

This research also visits the possibility of testing these algorithms on a real-world dataset that is generated in-house using a handheld system and comparing the obtained results and the calculated figures.

The whole work is posed as a part solution for a centralized map-building problem, where multiple robots with lesser sensor equipment are used for performing the tasks whereas only a smaller subset of robots with a highly advanced sensor suite carries out the map operation.

Contents

Acronyms

SLAM Simultaneous Localization and Mapping 1

VSLAM Visual SLAM 20

ToF Time of Flight 10

IR Infrared 10

KP Keypoint 11

KF Keyframe 25

STM Short Term Memory 31

OGM Occupancy Grid Map 31

GPU Graphical Processing Unit 35

BA Bundle Adjustment 30

SIFT Scale Invariant Feature Transform 11

SURF Speeded Up Robust Feature 11

FAST Features from Accelerated Segment Test 11

ORB Oriented FAST and Rotated BRIEF 11

BRIEF Binary Robust Independent Elementary Features 12

BRISK Binary Robust Invariant Scalable Keypoints 12

FLANN Fast Library for Approximate Nearest Neighbors 13

1 Introduction

Humans have been using maps to navigate around the world for the longest time. Maps have been used since ancient times, with some of the oldest evidence of maps being used dating back to over 3000 years ago, like the one shown in Figure 1. Today, maps are one of the most fundamental tools for navigation and are used in a plethora of fields like geography, urban planning, and transportation. This concept of maps has been adapted to the world of robotics too, wherein we provide the machines with a spatial awareness of their environment and therefore help the machine to navigate through this environment efficiently and take appropriate decisions.

Figure 1: Turin Papyrus Map drawn at around 1150 BC [FH21]

This is the core idea and the driving force behind mapping in robotics. Mapping is an integral part of robotics which enables the robots to operate autonomously and to perform tasks that would otherwise be impossible without a sense of spatial awareness. By creating accurate maps, the robots can navigate complex environments and interact with them in a near human-like intelligence and cognition. This concept is further expanded into Simultaneous Localization and Mapping (SLAM) which allows the robot to not only map its surroundings but also to localize itself within that map in real time as it moves through this environment.

1.1 Motivation

Most robotic navigation applications that are in use today work based on a map of the environment in which they operate. This map can either be built and then updated in real-time when the robot operates in this environment. But in most cases, building a map of the environment requires a lot of computational resources and once the map is built, updating this map requires much less computational power and can be performed with much lesser sensing modalities. Especially when more than one robot is operating in an environment, recalculating the 3D map of the environment on every robot is very exhaustive in terms of resources and also quite obsolete.

Figure 2: Sample warehouse where numerous robots carry out logistic operations[1]

Collaborative mapping has been taken up as active an research topic in the area of robotics over the last couple of decades. It simply means that a fleet of robots that operate in the same scenario, collaboratively perform map building. It poses numerous advantages, be it in terms of time-saving, coverage, scene dynamics, etc. [Rap]

The one major disadvantage of collaborative systems is that they need to be designed in such a way that each unit of that system is designed robustly. Every robot in the fleet needs to maintain comparably good hardware to ensure the quality of service that the system offers. As good as this works for small to medium systems, one can imagine the complexity and the cost of the system to scale up phenomenally.

Take an example scenario as shown in Figure 2, of a storage warehouse in which the storage container positions are not fixed. For any robot fleet, say an inventory pick and

[1]Image courtesy: INVIA Robotics

place robot, to operate in this scenario, it needs a map of an environment, and if the requirement of mapping the removed, the only navigational sensors it would need would be the ones required for localizing itself in this environment.

Centralized mapping poses the advantage that only a small subset of the fleet is needed to be equipped with the hardware and software resources to perform the computationally expensive map-building and optimization segment. These robots, shown in blue in the fig. 3, *explore* the arena and build maps of the region of interest. The rest of the fleet (shown in black) that is operating in this arena requires only a place recognition framework and the hardware required for this task, which is usually less demanding than the requirement of the optimization. An example of a similar centralized system is explored in [SC19] as this work implements a server for map adjustment and optimization.

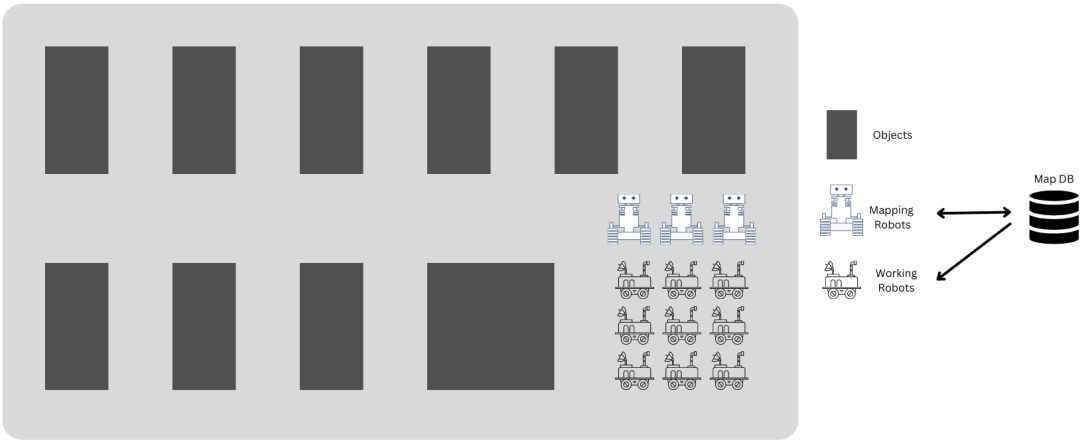


Figure 3: Overview of the scenario in the proposed centralized mapping scheme

With this work, a centralized mapping solution is ideated – a system that can map the environment in high quality and then provide this map for reuse for other robots operating in the same context, thereby taking off the computational load from multiple machines and concentrating it on one particular capable device. This

1.2 Aim of this work

The ultimate aim of this work is to develop a pipeline that can generate a high-quality map that can be then used by other users/robots for navigational purposes, so this project focuses on the map-building part and performing that with an optimum amount of resources.

This thesis is intended to serve as a brief guide to a user who is seeking to set up a map-building pipeline for their robotic solution and therefore contains details on various preexisting methods and processes. Therefore the work covers basic details from the fundamental 3D concepts and methodologies and also explains methods to evaluate the quality of the methods and the results.

The expected result of this work is an end-to-end mapping solution, that can be implemented out of the box for the specified application. The system is fed with data pertaining to the environment in the form of images from the stereo camera. The system processes the information and develops a 3D map of the environment which can then be used for navigation purposes.

1.3 Approach

The problem statement is tackled in a systematic manner as depicted in fig. 4. The first step of the process would be to gather a brief understanding in terms of the current trends in camera based SLAM and map-building technologies. This will help the reader in understanding the history and the core ideas behind various mapping technologies out there. The map building as such is then carried out using a selected method, and then various experiments are carried out with this method.

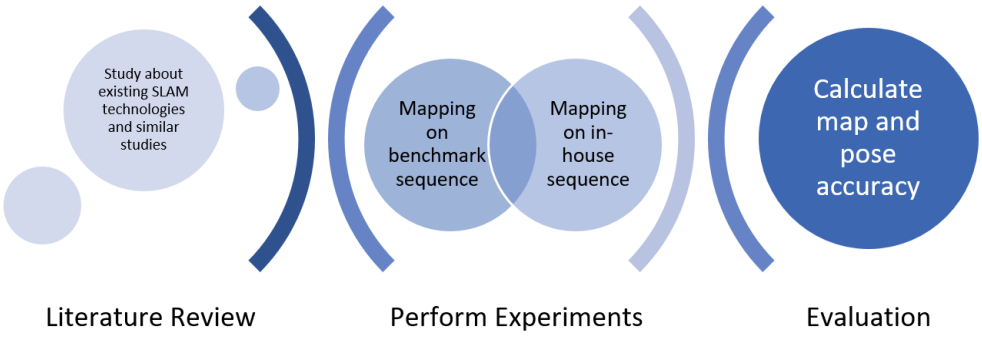


Figure 4: Layout of the thesis approach

This is the exact way in which this thesis is laid out. The methodology section explains the intended use case of the project, and assessing the various application areas and pinpointing on one particular application. Various considerations are made with respect to operating conditions, the requirement of the end user etc. The choice of the method

under study is going to be explained with the help of these conditions and requirements. Subsequent sections covers the literature survey for this work. Past research in this particular topic are studied and also the various different SLAM algorithms that employ similar strategies are also taken up for study. The last section deals with the implementation and the testing of the methods that are arrived at in this work.

Even though this thesis introduces the concept of collaborative maps and the core idea of this work is to lay a background for the map-building part, it is beyond of the scope of this paper to visit any concepts relating to sharing the maps with the system. The overview of the idea is presented, but the sections are only aimed toward the mapping segment, focusing on the quality and accuracy of the maps.

2 Background

As discussed in the introduction, the core idea behind this document is to provide the reader with a comprehensive understanding of the various SLAM systems that are available right now and to arm the reader with sufficient knowledge so that an informed decision can be made regarding a choice of a solution to carry out an appropriate task.

In this chapter, the reader is given an insight into the various concepts and terminologies in 3D measurement and reconstruction to enhance the document's comprehension level. The section then presents a brief overview of one of the most popular methodologies. A subset of these methods is taken up for study and then moves on to the various methods for calculating the accuracy of the aforementioned SLAM pipelines and some significant metrics used to compare the accuracy.

2.1 Fundamentals

Any SLAM system has three main components - Tracking, Estimation, and Optimization. There are even further submodules in different kinds of methods, but these three components are the core of the system. The tracking and estimation part contains the sensor-dependent processing (speaking in context of camera-based methods) and is collectively called the *frontend*, and the optimization (both local and global) contains the sensor-independent processing classified under the *backend* as shown in the fig. 5. The tracking module in the frontend is responsible for the sensor information processing and feature extraction, and the estimation module performs the motion and location estimation. The backend performs computational tasks like calculating the pose and generating pose graphs, performing graph optimization, etc.

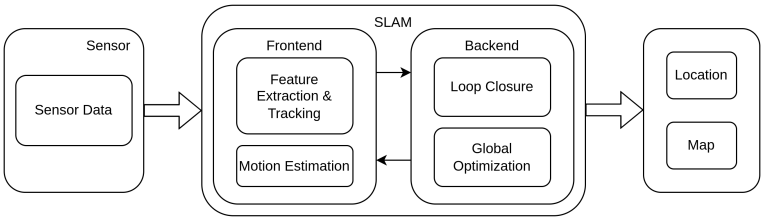


Figure 5: An overview of the constituent modules of a generalized Camera-based SLAM System

In this section, the user is given a brief idea about the fundamental mathematical concepts and tooling that are required to build up a SLAM system. The content that is presented here is largely sourced from the textbooks by Corke [Cor17] and Siciliano and Khatib [SK16]. These textbooks serve as a good starting point for most conventional SLAM concepts and one can then build upon the basic concepts thereafter.

2.1.1 Projective Geometry

This work visits a few imaging and measuring technologies so it is expected of one to gather some basic information about cameras and camera geometries. In computer vision, a popular model to understand the basics of image formation is the central perspective imaging model shown in Figure 6. According to Corke, the image plane is at a distance f in front of the camera origin and an image (non-inverted) is formed on the image plane. The rays converge on the origin of the camera frame C and the image is projected at $z = f$. The z-axis intersects the image plane at the principal point and forms the origin of the image coordinate frame.

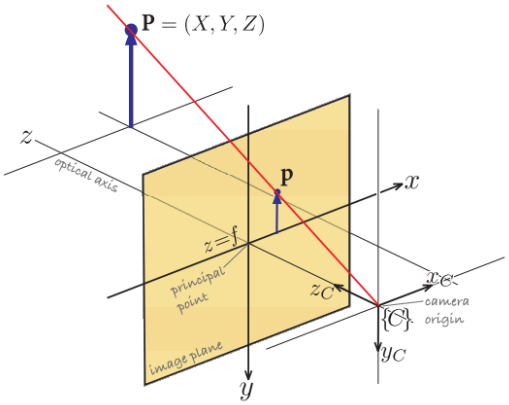


Figure 6: The central projection model [Cor17]

Using the laws of similarity of triangles, we can show that a world point $\mathbf{P} = (X, Y, Z)$is projected onto the image coordinate frame $\mathbf{p} = (x, y)$ as

$$x = f\frac{X}{Y} \qquad\qquad y = f\frac{X}{Y} \tag{2.1}$$

which is also known as the projective transformation or the perspective projection.

These transformations have some properties which appear as geometric effects on the image formed in the 2D image plane.

- A mapping from the 3D space to the 2D image coordinate plane is performed in the form of $\mathcal{P} : \mathbb{R}^3 \mapsto \mathbb{R}^2$.
- Straight lines are preserved as straight lines in the image plane.
- Parallel lines are projected to converging lines that intersect at the vanishing point as shown in fig. 7a, with the exception of the lines that are parallel in a plane that is parallel to the image plane. In this case, the parallelism of the lines is preserved.
- Conics in the world coordinates are projected as conics in the image plane. For example, a circle is transformed as a circle or an ellipse, one of the conic sections in the image plane, See fig. 7b.

(a) Railway lines that are parallel in the world coordinate system seem to converge at the vanishing point

(b) The Ferris wheel, which is actually a circle, is projected as an ellipse in the image plane

Figure 7: Examples illustrating the mapping to image plane

- The size or area of the shape is not preserved as it is a function of the distance.
- The mapping is not one-to-one and therefore no unique inverse is possible. This means that one can only say that the point in the world coordinates lie somewhere on the ray that projects this point onto the image plane.

- The transformation is a non-conformal one and there is no shape preservation.

With the knowledge of various camera parameters, it is possible to derive a mathematical formulation that performs a transformation between the 3D space and 2D image coordinates.

Stereo Camera System

As explained in [Kle14], single-camera systems have one main limitation, that is in terms of scale ambiguity. The 3D point cannot be calculated, but it can only be said that it lies on the ray that falls onto the image plane. This is resolved by adding one more camera that looks at the same scene and performs this 3D to 2D mapping on a different ray. Now we have two non-parallel rays that emanate from the same point and this in turn can be used to solve for the 3D coordinates of the point object. This process is called triangulation from stereo (see fig. 8) and is performed to extract the depth or distance to the point of interest.

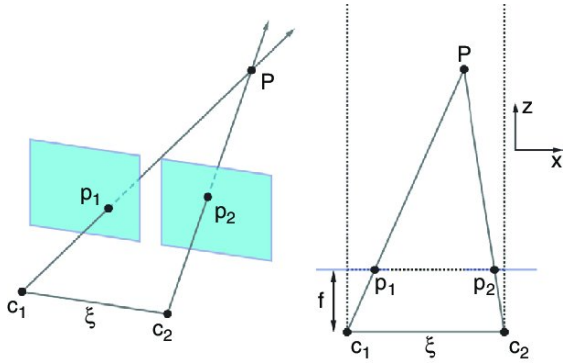


Figure 8: Triangulation from Stereo [WFM19]

The point coordinates P in the world frame are imaged using two cameras C_1 and C_2 and the corresponding image coordinates $p_1 = (u_1, v_1)$ and $p_2 = (u_2, v_2)$ are known. Using the intercept theorem, the location of the 3D point $P = (p_x, p_y, p_z)$ can be calculated as

$$p_z = \frac{f\xi}{\delta} \tag{2.2}$$

where f is the focal length, ξ is the length of the epipolar axis or more commonly referred to as the baseline of the camera, δ is the disparity between the images, which is the pixel difference between a point in the x direction calculated as $\delta = u_1 - u_2$. One can then use

the triangle similarity property to map the 3D coordinates onto the image coordinates as

$$p_x = \frac{p_z}{f} u_1 \qquad\qquad p_y = \frac{p_z}{f} v_1 \tag{2.3}$$

where u_1 and v_1 are the corresponding 2D coordinates on the left image [WFM19].

RGBD Imaging Systems

RGBD imaging systems are a newer family of sensors in the depth perception toolkit. These sensors provide the user with the per-pixel depth along with the color image of the scene. These sensors usually have a precise distance measurement unit, like a Time of Flight (ToF) sensor or a structured Infrared (IR) projector and IR sensor pair. A very good example of an RGBD sensor is the Structure sensor, the one shown in Figure 9.

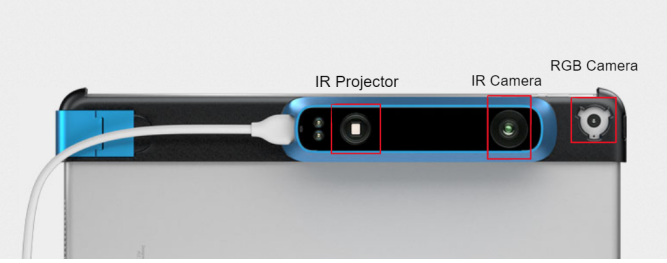


Figure 9: Structure Sensor[1]

Now the user can directly read the depth image from the sensor itself instead of using mathematical calculations like stereo matching to obtain the depth image. Since these sensors use hardware means to calculate the depth, it is extremely accurate in their working range. A denser and continuous depth map can be obtained which in turn enhances the quality of the 3D map.

These sensors find their application in close-range depth sensing, e.g. in the case of indoor scenarios and medical applications, as their depth range is always sensor dependent. Whereas the previously mentioned stereo camera systems are largely preferred in outdoor scenarios because their depth range is dependent on the baseline and this is something that can be varied according to the application. To put things into perspective, very large baseline imaging was also used recently to capture images of the black hole.

[1]Image sourced from `https://structure.io/`

2.2 Image Features

The previous section was intended to give the reader some insight into the various imaging technologies that are used for SLAM. For the system to perform localization, it has to identify *landmarks* in the scene that it is observing. This is the relevance of feature extraction and matching in computer vision. Landmark detection is applicable in various fields of computer vision, e.g. robot navigation, image retrieval and indexing, and 3D object reconstruction.

A feature is simply a piece of information that is contained in the image which can then be used for identification of certain points or characteristics that are contained in the image. Features can be both specific and generic. Specific features are the ones that pinpoint a certain location in the image, e.g. corners, peaks, interesting shapes, etc. Localized features of this kind are also commonly termed Keypoint (KP) or keypoint features. General features are the ones classified by the general neighborhood or a region, e.g. edges, boundaries, patches, etc.

Three main concepts pertaining to image features - feature detection, description, and matching are explained in this section.

Detection This is the process of identifying or locating the feature in the image space. The KPs are usually expressive in nature and are located at object boundaries, corners, and similar regions with sudden intensity changes. It is usually selected and identified based on parameters like prominence. More prominent features are easier to identify and extract. They are robust under perturbations in the image space, i.e., invariant to illumination changes and image operations like affine transformations.

Some examples of classical feature detectors are

- Harris Corner Detector [HS88]
- Scale Invariant Feature Transform (SIFT) [Low99]
- Speeded Up Robust Feature (SURF) [BTV06]
- Features from Accelerated Segment Test (FAST) (See fig. 10) [Vis11]
- Oriented FAST and Rotated BRIEF (ORB) [Rub+11]

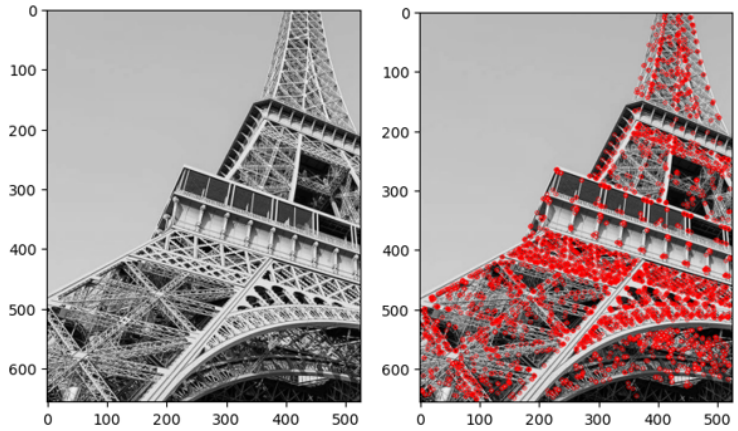


Figure 10: Features extracted using the FAST detector

Descriptors Feature descriptors, also known as feature vectors are used to encode some information within the aforementioned landmarks or features. This is done in order to make the matching process (explained in the subsequent paragraph) easier. Descriptors contain information about the point properties and the properties of the neighborhood. They act like a *fingerprint* for the features and help uniquely identify them. The descriptors are invariant against image transformations [Low99], and are therefore detectable even if the image is transformed geometrically.

A descriptor is always calculated for each of the detected KPs so that these can be correlated across images. As a matter of fact, most feature detectors previously mentioned also include a feature descriptor algorithm embedded into them. Examples of feature descriptors are

- Harris Corner Detector [HS88]
- Scale Invariant Feature Transform (SIFT) [Low99]
- Speeded Up Robust Feature (SURF) [BTV06]
- Binary Robust Invariant Scalable Keypoints (BRISK) [LCS11]
- Binary Robust Independent Elementary Features (BRIEF) [Cal+10]
- Oriented FAST and Rotated BRIEF (ORB) [Rub+11]

Matching Feature matching or generally image matching is an integral part of numerous computer vision algorithms. As the name suggests, a matcher performs the task of finding matching features (called *correspondences*) between two images. It is done by comparing image features from both images, with the help of the calculated descriptors. Some of the most common matchers used in classical problems are

- Brute force matching
- Fast Library for Approximate Nearest Neighbors (FLANN)

An example of the brute force matcher is shown in fig. 11.

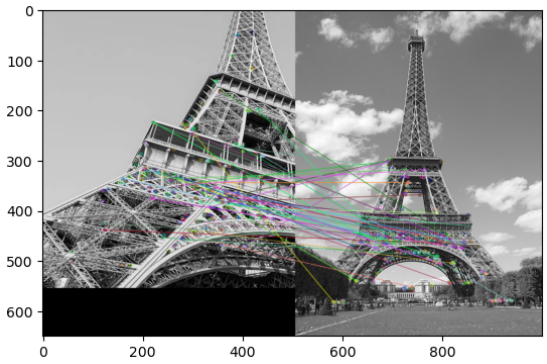


Figure 11: Matching features using the brute-force matcher

2.3 Simultaneous Localization and Mapping

2.3.1 The SLAM Problem

According to Siciliano and Khatib [SK16], the Simultaneous Localization and Mapping problem is ideated as a robot motion model, where the robot motion is uncertain, and the robot makes observations along its motion about the environment with the help of a noisy sensor. The SLAM problem is to construct a map of the environment and localize the robot in this environment, simultaneously.

To put this in mathematical terms, the robot *path* – the sequence of robot positions (denoted by x_t) be X_T where x_t contains the position of the robot. For a 2D case, this is a 3-dimensional vector that will contain the position coordinates and the orientation angle of the robot. The path, X_T of the robot is defined as

$$X_T = \{x_0, x_1, x_2, ..., x_T\} \tag{2.4}$$

Here, x_0 is the initial pose assumption and this observation is important as it serves as the reference for all the further pose calculations. The subsequent positions are calculated using *odometry*. Odometry is defined as the movement of the robot between time $t-1$ and t and is depicted using u_t. The odometry sequence is formulated as

$$U_T = \{u_0, u_1, u_2, ..., u_T\} \tag{2.5}$$

and describes the motion sequence of the robot. In an ideal, noise-free environment, the odometry would be sufficient to estimate poses from the initial pose x_0. But in the real world, this is not usually the case. Neither the odometry nor the initial pose estimate is very accurate. Measurements are always noisy and the calculated pose is erroneous and this error propagates through the calculation. Eventually, this causes the pose estimates to drift gradually and diverge from the actual pose.

Lastly, the robot is deployed in an environment whose true map is denoted by m and is assumed to be static. The environment consists of various objects, surfaces, passages, etc. and the map embeds information regarding them. The measurements, denoted by z_t establish relations between the features in the map and the robot pose x_t, and the generated sequence of measurements is written as

$$Z_T = \{z_0, z_1, z_2, ..., z_T\} \tag{2.6}$$

This is depicted graphically in Figure 12

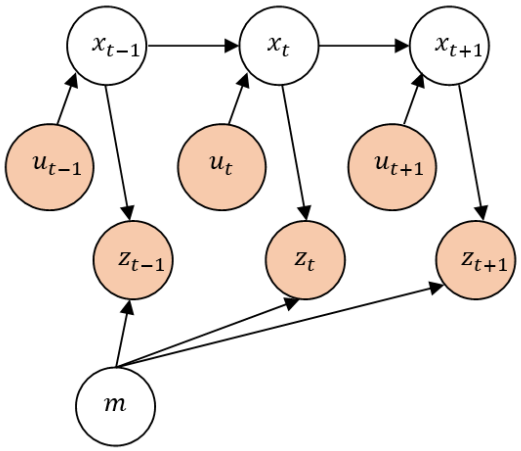


Figure 12: Graphical representation of the SLAM problem [SK16]

Going back to the definition, the SLAM problem is essentially to estimate the map m and the robot location sequence X_T simultaneously, using the odometry U_T and the measurement Z_T. There are generally two approaches to approach this

- Full SLAM, or estimating the complete position sequence and the map using the given odometry and measurements. It is formulated as

$$p(X_T, m \mid Z_T, U_T) \tag{2.7}$$

- Online SLAM, which is in contrast to the previous method, estimating the *current* position of the robot using the present odometry and measurement. This is formulated as

$$p(x_t, m \mid Z_t, U_t) \tag{2.8}$$

  Online SLAM is usually an incremental process unlike the batch process of the full SLAM method and algorithms that process the data sequentially are referred to as *filters*.

To solve the SLAM problem, one needs the odometry information u_t which is a function of robot locations at $t-1$ and t. The measurement information is also needed, which relates the measurements z_t to the robot locations x_t. Since these observations are made with uncertainties, they are modeled as probability distribution functions. The motion model

$$p(x_t \mid x_{t-1}, u_t) \tag{2.9}$$

and the measurement model

$$p(z_t \mid x_t, m) \tag{2.10}$$

are hence formulated. But again, the position estimate and the map are known with some uncertainty, and thus the Bayes rule is employed to structure these relations into a form from which we can reconstruct the probability distributions using the observed data [SK16].

An example can be used to make these mathematical formulations more relevant to the SLAM problem. Consider a map vector m of the 2D world, with N number of landmarks embedded in it. Landmarks may be anything, e.g. doorposts, furniture, corners, etc. projected onto the 2D place as coordinates. Now the *observed* world is a vector of size $2N$, where each landmark is described by a 2 coordinate value. In a common SLAM framework, the robot can sense

- relative distance to the landmarks.
- relative heading (orientation) to these landmarks.
- the *identity* of these landmarks.

Out of the three, the identity of the landmarks can be sensed with the most accuracy.

Moving on to modeling this setup mathematically, a measurement function h is described first. Measurement is a function over robot position x_t and map m and is written as

$$h(x_t, m). \tag{2.11}$$

In the landmark world that is designed in this case, the function h can be computed using triangulation. The probabilistic measurement model is derived from the measurement model by adding a noisy sensor function $\mathcal{N}$. This function is a 2D normal distribution that peaks at $h(x_t, m)$ and incorporates measurement noise as $\mathbf{Q_t}$, the noise covariance matrix.

$$p(z_t \mid x_t, m) = \mathcal{N}(h(x_t, m), \mathbf{Q_t}). \tag{2.12}$$

Similarly, the motion model is derived from the kinematic model for robot motion. The motion model g is defined as

$$g(x_{t-1}, u_t). \tag{2.13}$$

This model may now be transformed into a noisy motion model observation with the normal distribution centered at $g(x_{t-1}, u_t)$ and covariance $\mathbf{R_t}$ as

$$p(x_t \mid x_{t-1}, u_t) = \mathcal{N}(g(x_{t-1}, u_t), \mathbf{R_t}). \tag{2.14}$$

With this, the complete model of the SLAM system in the landmark world with a range-heading measurement system has been derived [SK16].

According to Siciliano and Khatib, the online and offline is just one of the numerous taxonomical classifications that come under the general title of SLAM. Some of the other common distinctions are

- Volumetric and Feature-Based
- Topological and Metric
- Small and Large Uncertainty
- Single-Robot and Multi-Robot

2.3.2 SLAM Paradigms

SLAM being an inherently uncertain problem, dealing with noisy observations and data with extremely weak associations, probabilistic methods are essential in modeling this problem. In the SLAM domain, three major paradigms are used as an approach to counter the uncertainty.

- Extended Kalman Filter (EKF)
- Particle Filter
- Graph-based Optimization Techniques

Extended Kalman Filter

The Extended Kalman Filter (EKF) based method introduced in [SSC88] is one first and the most popular SLAM methods. The robot state and the measurement states are represented as multivariate Gaussian distributions and the unknowns are estimated using recursive Bayesian filtering. This method assumes a metric, feature-based representation of the environment where the robot state and the landmarks form a network of uncertain spatial relationships.

Under EKF SLAM, the problem statement becomes

$$p(x_t, m \mid Z_t, U_t) = \mathcal{N}\left(\mu_{\mathbf{t}}, \Sigma_t\right). \tag{2.15}$$

Here $\mu_{\mathbf{t}}$ contains the position information of the robot itself and the landmarks in the environment. According to the point landmark example that was discussed earlier μ_t is $3 + 2N$ dimensional, the robot location being described using 3 variables and the N landmarks using $2N$ variables. The Σ_t models the uncertainty in the position estimate μ_t and is of size $(3 + 2N) \times (3 + 2N)$ and this matrix is positive definite.

The point-landmark example can easily be adapted into the EKF formulation of the SLAM problem. It works under the assumption that the in a small time interval, the measurement model and the motion model are linear in their domains. In this case, the *vanilla* Kalman filter will be applicable to the problem. What EKF does is it linearizes the function g and h using Taylor series expansion. In the most basic form (and without any data association problem) the EKF SLAM is only the application of the EKF to the online SLAM problem (covered in Section 2.3.1).

Particle Filters

As the name suggests, it represents the robot state (also referred to as the posterior) through a set of *particles* [SK16]. Each particle is representative of the best guess of the true robot state. By collecting many such approximations into a set of good guesses, a set of particles in this context, the particle filter algorithm estimates the robot state [TBF05].

This method is known to operate only under mild and restricted conditions. As the number of features (or landmarks) scales up, it causes the particle filter to scale up

exponentially. The true posterior is approached as the particle set size approaches infinity. It is widely used in 2D laser-based SLAM algorithms, as it contains a limited number of landmarks, and therefore the posterior estimate is arrived at in near-real time.

An example of the particle-based algorithm is the FastSLAM [Mon+02] introduced by Montemerlo et al. The basic FastSLAM can be explained easily using the point-landmark example that was introduced earlier. At any instant, the FastSLAM maintains K particles of type

$$X_t^{[k]}, \quad \mu_{t,1}^{[k]}, \ldots, \mu_{t,N}^{[k]}, \quad \Sigma_{t,1}^{[k]}, \ldots, \Sigma_{t,N}^{[k]} \tag{2.16}$$

Here $[k]$ is the sample index, and the expression contains a sample path $X_t^{[k]}$ and a set of N two-dimensional Gaussians with mean $\mu_{t,n}^{[k]}$ and variances $\Sigma_{t,n}^{[k]}$, one for each landmark in the environment. The variable $n(1 \leq n \leq N)$ is the landmark index.

Initializing FastSLAM is performed by setting each of the particles' robot location to know starting coordinates, and initialize the map. The particle update is then performed as follows:

- When odometry is received, new location variables are generated stochastically for each particle. The distribution is based on the motion model defined as

  $$x_t^{[k]} \sim p\left(x_t \mid x_{t-1}^{[k]}, u^t\right) \tag{2.17}$$

  where $x_{t-1}^{[k]}$ is the previous location. This probabilistic sampling can easily be implemented for any robot whose kinematics can be computed.

- When a measurement z_t is received, the probability of measurement for each particle is measured. This can be expressed as

  $$w_t^{[k]} := \mathcal{N}\left(z_t \mid x_t^{[k]}, \mu_{t,n}^{[k]}, \Sigma_{t,n}^{[k]}\right) \tag{2.18}$$

  where the $w_t^{[k]}$ is called the importance weight as this measure determines the dominance of the particle in the new measurement.

- Next, FastSLAM performs *resampling*. Resampling is the process of drawing replacement from the set of existing particles. The probability of drawing a particle is the normalized importance weight. The intuition behind resampling is that the particles which have a more plausible measurement have a higher chance of surviving the resampling process.

- Finally, FastSLAM updates for the new particle set the mean $\mu_{t,n}^{[k]}$ and the covariance $\Sigma_{t,n}^{[k]}$ based on the measurement z_t.

Summarising the above steps, sampling from the motion model is a straightforward process, since it involves a simple kinematic calculation. Computing the importance for a measurement with Gaussian noise is also is then performed. And finally, this information is used to update a low-dimensional particle filter. These steps make the FastSLAM one of the easiest to implement algorithms to solve the SLAM problem. [SK16]

Graph SLAM Methods

Graph-based SLAM methods, also known as Graph Optimization methods model the robot state and the map as a graph interconnected by the means of observation. The two kinds of nodes in the graph represent the robot pose or a landmark on the map. The edges of the graph represent a constraint bound by observation. This constraint is a function of the measurement and the motion model that was explained in the previous section.

The basic intuition of the SLAM problem is as follows. As mentioned in [SK16], the environment is represented using landmarks, and these landmarks and robot locations can be thought of as nodes in a graph. Every consecutive location x_{t-1}, x_t can be connected by an edge that represents the information provided by odometry u_t. Other soft constraint edges also exist between locations x_t and landmarks m_i (assuming that the landmark m_i was sensed at time t. The construction of the graph (shown in fig. 13) is done in the following manner.

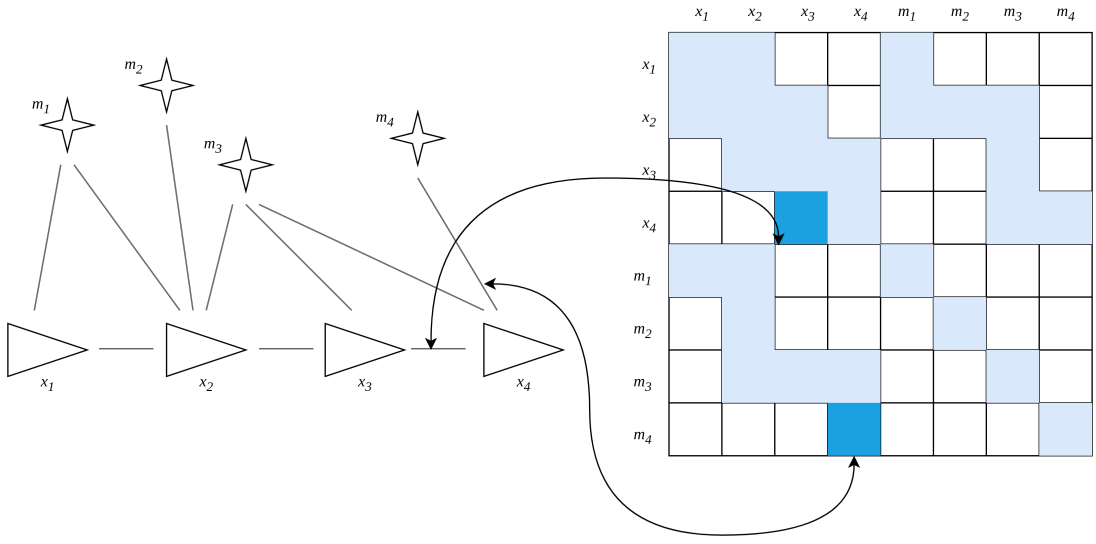


Figure 13: Illustration of the graph construction

At time $t = 1$, the robot senses the landmark m_1 and this builds a relation edge between the two nodes x_1 and m_1. When caching the edged in matrix format, an entry is made into the matrix as shown at the R.H.S. of the fig. 13. Now when the robot moves in the environment to a location x_2, the odometry relation u_2 generates an edge between the nodes x_1 and x_2. At location x_2, the robot is able to sense landmarks m_1, m_2 & m_3, and this leads to edges connecting $x_2 - m_1$, $x_2 - m_2$ and $x_2 - m_3$. Consecutive application of these steps as the robot moves in the environment and senses the landmarks lead to a graph that increases in size. It is to be noted that the graph is *sparse*, as seen in the illustration, as each node is connected to only a limited number of other nodes.

Now that we have a quadratic[2] formulation of the SLAM problem [SK16], computing the solution is equivalent to computing the state of minimal energy of the model.

2.3.3 Visual SLAM

Visual SLAM, or Visual SLAM (VSLAM), according to Siciliano and Khatib, can be summarised as the process of tracking the robot pose in full 6-Degrees of Freedom (DoF) using data from visual sensors like RGB and RGBD cameras. Visual sensors pose the advantage of offering a plethora of information that helps address the complex issues of data association and loop closing using visual appearance-based features and techniques.

2.4 Evaluation Metric

The output of all major SLAM systems is of two types - the estimated camera trajectory (in the form of poses) and a map estimate. It is essential to establish the accuracy of the said outputs, both in terms of map and trajectory accuracy. It is possible to calculate the trajectory accuracy with the help of a prerecorded data stream of poses, or with the help of Global Navigation Satellite System (GNSS) measurements with Real-Time Kinematic (RTK) corrections, or with infrared motion capture methods like VICON[3] and OptiTrack[4] which offer high accuracy. But when it comes to evaluating the quality of the generated maps, it is quite a difficult process, as it is difficult to generate very accurate ground truths.

To counter this pitfall, Sturm et al. came up with a technique in [Stu+12] that can correlate the quality of the estimated trajectory with the input sequence of RGB-Depth (RGBD) images. This technique simplifies the evaluation process, but it has to be noted that a good trajectory does not necessarily imply a good map.

[2]The matrix form happens to correspond to a quadratic equation defining the resulting constraints
[3]`https://www.vicon.com/`
[4]`https://optitrack.com/`

The evaluation is performed using two pose sequences $\mathbf{P} = \{\mathbf{P_1}, \mathbf{P_2}, ..., \mathbf{P_n}\} \in SE(3)$ and $\mathbf{Q} = \{\mathbf{Q_1}, \mathbf{Q_2}, ..., \mathbf{Q_n}\} \in SE(3)$ which represent the estimated and the ground truth trajectory respectively. The two sequences have to be time synchronized, sampled similarly and have the same stream length in order to be compared with each other. But the real world, being far from ideal and the measurement devices that has an inherent margin of error, the measurements of the poses are not identical. They differ in sampling rates, stream lengths and have inconsistent data which cannot be used for measurements. This calls for the requirement of an additional interpolation step with proper data association. The two major metrics that are used very widely in the SLAM research are explained here.

Relative Pose Error (RPE) The relative pose error, as the name suggests, is a measure of the local accuracy of the poses over a fixed time interval Δ. Therefore it is a good indicator of the drift of the trajectory which is one of the major challenges any visual odometry system faces. The relative pose error at time i is formulated as

$$\mathbf{E}_i := \left(\mathbf{Q}_i^{-1}\mathbf{Q}_{i+\Delta}\right)^{-1} \left(\mathbf{P}_i^{-1}\mathbf{P}_{i+\Delta}\right) \tag{2.19}$$

From a sequence of n camera poses, $m = n - \Delta$ individual relative poses are obtained, and the Root Mean Square Error (RMSE) is calculated over the translational component (denoted by $trans(\mathbf{E}_i)$) of the relative pose error as in eq. (2.20).

$$\mathrm{RMSE}\left(\mathbf{E}_{1:n}, \Delta\right) := \left(\frac{1}{m}\sum_{i=1}^{m} \left\|trans\left(\mathbf{E}_i\right)\right\|^2\right)^{1/2} \tag{2.20}$$

Root Mean Square Error (RMSE) is usually the preferred metric so that all the information contained is factored in while calculating the metric. But there are alternatives to this like the mean and the median error. They attribute lesser influence to the outliers and thus they are a stronger indicator of the system's peak performance. Additionally, the rotational error can also be calculated, but as stated in [Stu+12] the rotational error shows up as a translational error when there is camera movement and the evaluation using only the translational component serves as a good estimator for the trajectory of the camera.

Also, the time parameter Δ needs to be chosen accordingly. For example, in the case of a visual odometry system that finds matches between consecutive frames, a good choice is $\Delta = 1$. Another possibility is when the drift per second is to be calculated, the Δ value can be set as the framerate of the capture device (e.g. $\Delta = 30$ for a camera capturing 30 frames per second). For SLAM systems, the error is evaluated over all time intervals [Stu+12] and therefore the equation becomes

$$\mathrm{RMSE}\left(\mathbf{E}_{1:n}\right) := \frac{1}{n}\sum_{\Delta=1}^{n} \mathrm{RMSE}\left(\mathbf{E}_{1:n}, \Delta\right) \tag{2.21}$$

Absolute Trajectory Error (ATE) The global consistency of the estimated poses is an important aspect when the accuracy of SLAM systems are calculated. The most straightforward method to do this is to compare the absolute distances between the estimated and the ground truth trajectory. Since these two trajectories can be in different coordinates, it is necessary to align them before a comparison can be done. This alignment of two coordinate frames only related by a set of measurements is a classical problem in stereophotogrammetry and is discussed by Horn in [Hor87]. This work finds the rigid-body transformations $\mathbf{S}$ corresponding to the least-squares solution that transforms the estimated trajectory onto the ground truth. The ATE at time step i is now formulated as

$$\mathbf{F}_i := \mathbf{Q}_i^{-1}\mathbf{S}\mathbf{P}_i \tag{2.22}$$

The previous concept of calculating the error over all the time steps is also extended to the calculation of the RMSE for the ATE (translational component) as

$$\mathrm{RMSE}\left(\mathbf{F}_{1:n}\right) := \left(\frac{1}{n}\sum_{i=1}^{n}\left\|trans\left(\mathbf{F}_i\right)\right\|^2\right)^{1/2} \tag{2.23}$$

It is also to be noted that the RPE too can be used for the calculation of the global error, by averaging over all possible time indices. The RPE factors in both translational and rotational errors, whereas the ATE considers the translational errors, and this results in the RPE always being slightly larger than ATE. Therefore RPE can be considered a good measurement of the accuracy of the system as it accounts for both kinds of drifts.

3 Literature Review

In this chapter, works of similar interest and objectives are studied. A review of the SLAM algorithms that have been developed over time is visited briefly. A short elaboration of each of the methods is performed, with a focus on the core idea, methodology, and the advantage and disadvantages of each method is performed. This comparison helps the user in setting the context as to why one particular method was chosen to study and then why it is or is not a good candidate for the mapping application.

A review of past works in the context of comparative studies is also carried out to help in setting the context of the evaluation that is carried out in this research manuscript.

3.1 SLAM Methods

The application scenario described in the first chapter poses two main requirements for the system.

- The maps must be comprehensible to a human operator. Therefore the system must be capable of generating maps of the environment with near-centimeter range voxel grid resolution.
- The system must also keep track of its location within an acceptable level of accuracy. Since the system is designed to operate in warehouses and shop floors, the centimeter range accuracy stands as a good candidate for location accuracy of the system.

Since there is a requirement of human-comprehensibility, only VSLAM methods are considered for this study. To narrow down the list further, a selection is made based on the availability of Robot Operating System (ROS) implemntations, as ROS is a popular choice for setting up and implementing robotics systems. With these constraints in place, the list of studied methods are narrowed down to

1. LSD SLAM [ESC14]
2. ORB SLAM 2 [MT17]
3. DTAM [NLD11]

4. RTAB-Map [LM19]

3.1.1 LSD SLAM

LSD SLAM[1], first introduced in [ESC14] is a monocular VSLAM algorithm developed by Engel, Schöps, and Cremers at the Technical University of Munich in Germany. LSD-SLAM stands for Large-Scale Direct Monocular SLAM, and it is built to work with a single camera, thereby monocular. This is a direct SLAM algorithm, which means that it directly estimates the depth of each pixel in the camera image instead of using feature-based methods. Direct methods have several advantages over feature-based methods, including better accuracy, robustness to changes in illumination, and the ability to work with low-texture environments as they work based on scene properties like color, brightness, and intensity gradient [Kud] and track the movement of those pixels from frame to frame. This approach changes the problem being solved from one of minimizing geometric reprojection errors, as in the case of feature-based SLAM, to minimizing photometric errors.

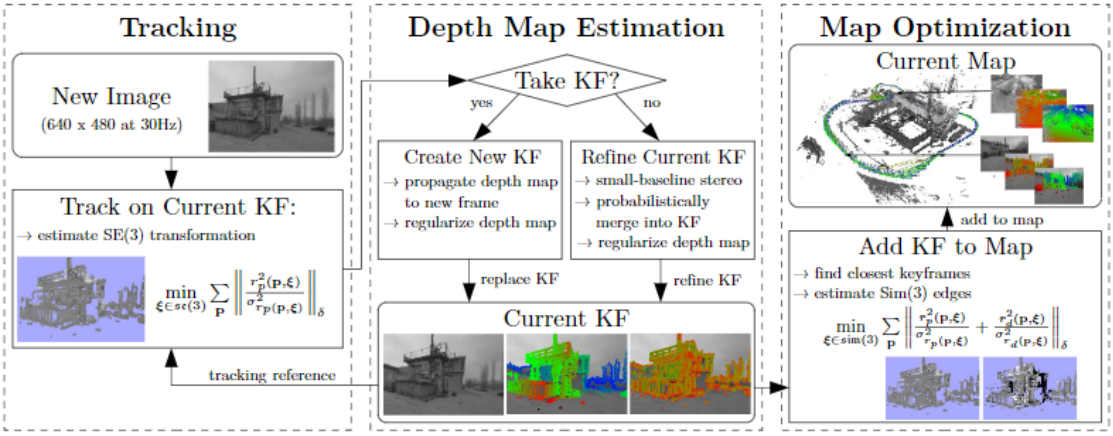


Figure 14: Overview of the LSD SLAM Algorithm [ESC14]

Depth Map Estimation The depth of each pixel in the camera image is calculated by minimizing the photometric error between the observed image and the predicted image. The predicted image is generated by back-projecting the 3D points in the map into the camera frame and warping them to the current frame using the estimated camera pose. The photometric error is calculated as the sum of the absolute differences between the observed image and the predicted image. The algorithm uses a nonlinear optimization

[1] https://github.com/tum-vision/lsd_slam

technique (Gauss-Newton optimization) to minimize the photometric error and estimate the depth of each pixel in the image.

Camera Pose Estimation The camera pose is estimated by minimizing the reprojection error between the observed image and the predicted image. The reprojection error is the distance between the observed key point in the image and the projected key point in the predicted image. The algorithm uses a Keyframe (KF) approach to build the map of the environment. A key frame is a frame that is selected based on the amount of motion and the amount of change in the environment. The KFs are used to initialize the map and to optimize the camera pose and the depth of the pixels in the image.

Map Optimization This method also includes a loop-closure-based map optimization scheme that improves the consistency of the map and reduces drift over time. The loop-closure detection algorithm searches for previously visited locations in the map that are similar to the current location and corrects the camera trajectory accordingly. The loop-closure detection algorithm uses a bag-of-words (BoW) approach to representing the visual vocabulary of the environment. The BoW approach is similar to the approach used in ORB-SLAM2 (visited in section 3.1.2) and represents the frequency of visual words (patches of pixels) in the map.

In summary, LSD-SLAM is a monocular VSLAM algorithm that directly estimates the depth of each pixel in the camera image using a direct photometric error minimization approach. The algorithm uses keyframes to initialize the map and optimize the camera pose and the depth of the pixels in the image. LSD-SLAM also includes a loop-closure detection algorithm that improves map consistency and reduces drift over time. The algorithm has been shown to work in challenging environments such as low-texture or dynamic scenes, e.g. like an outdoor scene shown in fig. 15

3.1.2 ORB SLAM2

ORB-SLAM2[2] introduced in [MT17] is a real-time VSLAM algorithm developed by Mur-Artal and Tardos at the University of Zaragoza in Spain. It is an improved version of ORB-SLAM[3], which was released in 2015. The major improvement over its predecessor that this method offers is the support for stereo and RGBD inputs along with improvements in the SLAM backend.

[2] `https://github.com/raulmur/ORB_SLAM2`
[3] `https://github.com/raulmur/ORB_SLAM`

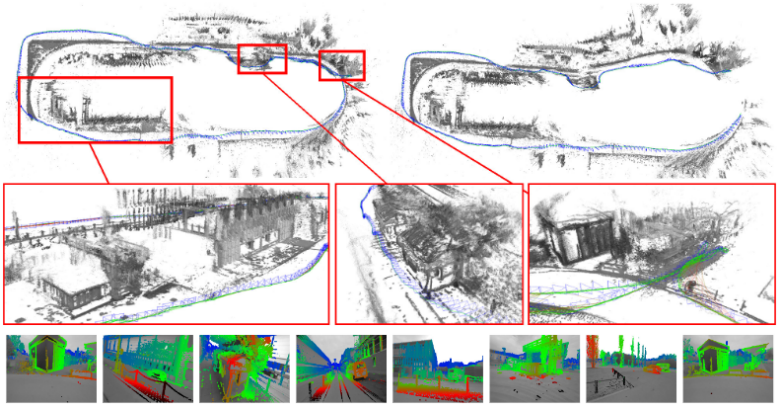


Figure 15: Challenging loop closures on an outdoor dataset [ESC14]

Core Processes ORB-SLAM2 presents 3 main threads for its core processes - tracking, local mapping, and loop closing as shown in fig. 16. The tracking thread receives the sensor input and performs operations like information preprocessing and KF decisions. The local mapping thread receives the KF and perform the insertion and culling of new map points. The loop closing thread performs the loop detection from the local map information and the geometric operation to fuse the calculated local map into the global map.

Feature Extraction & Matching This method uses visual features extracted from the camera images to estimate the camera pose and map the environment. The algorithm uses ORB features, which are a type of binary feature descriptor that is highly efficient and robust to changes in illumination and viewpoint. The ORB feature detector extracts key points from the camera images and calculates a binary descriptor for each key point. The descriptors are used to match key points between frames and to estimate the camera pose.

Pose Estimation The camera pose is estimated using an Extended Kalman Filter (EKF) based approach. The EKF as explained under section 2.3.2 is a recursive algorithm that estimates the state of a dynamic system based on noisy measurements. In the case of ORB-SLAM2, the state of the system is the camera pose, and the measurements are the key point matches between frames.

Loop Closure It also includes a loop-closure detection that improves the consistency of the map and reduces drift over time. The algorithm searches for previously visited locations in the map that are similar to the current location and corrects the camera

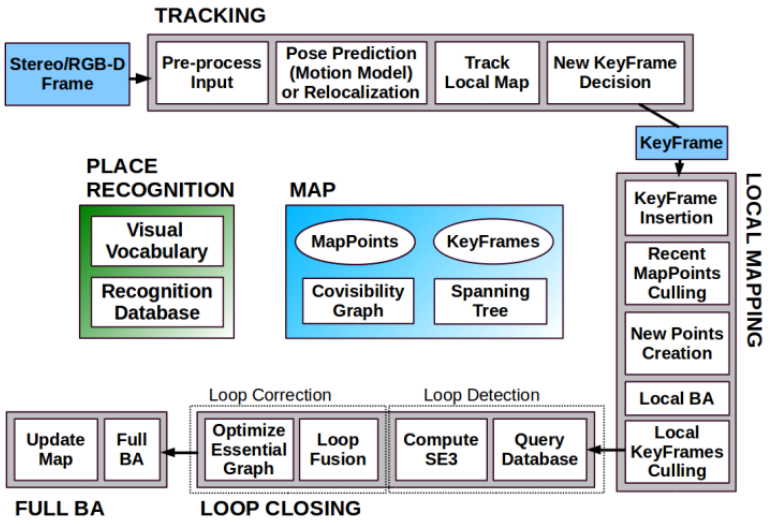


Figure 16: Parallel processes in the ORB-SLAM2 pipeline [MT17]

trajectory accordingly. It uses a bag-of-words (BoW) approach to represent the visual vocabulary of the environment. The BoW approach is a technique used in natural language processing and information retrieval that represents the frequency of words in a document. In the case of ORB-SLAM2, the BoW approach represents the frequency of visual words (ORB descriptors) in the map.

To summarise this method, the paper ORB-SLAM2 [MT17] presents a SLAM algorithm that operates on monocular, stereo and depth inputs and performs re-localization, loop closing, and map reuse. It builds globally consistent and scale-accurate maps for long-term localization which finds good relevance in applications like navigation and Virtual Reality (VR) etc.

3.1.3 DTAM

DTAM (Dense Tracking and Mapping) is a VSLAM algorithm introduced in [NLD11] that uses dense depth maps to estimate the camera pose and the 3D structure of the environment in real-time. It was proposed by Newcombe, Lovegrove, and Davison in 2011 and has since then been improved and extended by many researchers.

DTAM's core idea is to treat the depth map as a 3D grid of voxels, where each voxel stores its own 3D position and a corresponding photoconsistency value. Photoconsistency measures the similarity between the intensity of the voxel's projection in the current frame and the intensity of the same voxel's projection in the reference frame. The algorithm uses these voxels to represent the 3D structure of the environment and updates them

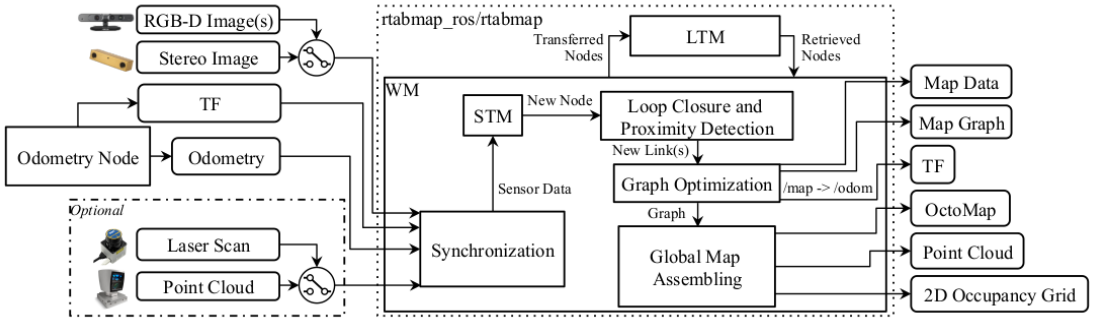


Figure 17: Block Diagram of the *rtabmap* node [LM19]

using a Bayesian filtering approach. The frontend of DTAM tracks the camera pose in real-time by minimizing the photo-consistency error between the current frame and the reference frame. The backend of DTAM fuses the new measurements into the 3D voxel grid and optimizes the camera pose and the 3D structure using a non-linear least-squares optimization.

Advantages & Limitations One of the main advantages of DTAM is that it can operate in real-time, providing dense 3D reconstructions of the environment while the camera is moving. DTAM has been shown to work well in various environments, including indoor and outdoor scenes. However, DTAM also has several limitations. It relies heavily on accurate depth estimation, which can be challenging in some cases, such as reflective or textureless surfaces. Moreover, the accuracy of the algorithm deteriorates when the camera movement is too fast or the scene is highly dynamic like in the case of an outdoor localization problem.

3.1.4 RTAB-Map

Real Time Appearance Based Mapping (RTAB-Map) is an open-source library originally intended for appearance-based loop closure detection approach with long-term memory management [LM13] introduced by Labbé and Michaud in 2013. This was further extended by the authors in [LM19] as an open-source solution for Visual and LiDAR SLAM. The RTAB-Map is a cross-platform standalone C++ and ROS package[4] and packs various features like online processing, low drift odometry, robust localization, map generation and usability, and multi-session mapping.

[4] `http://www.wiki.ros.org/rtabmap_ros`

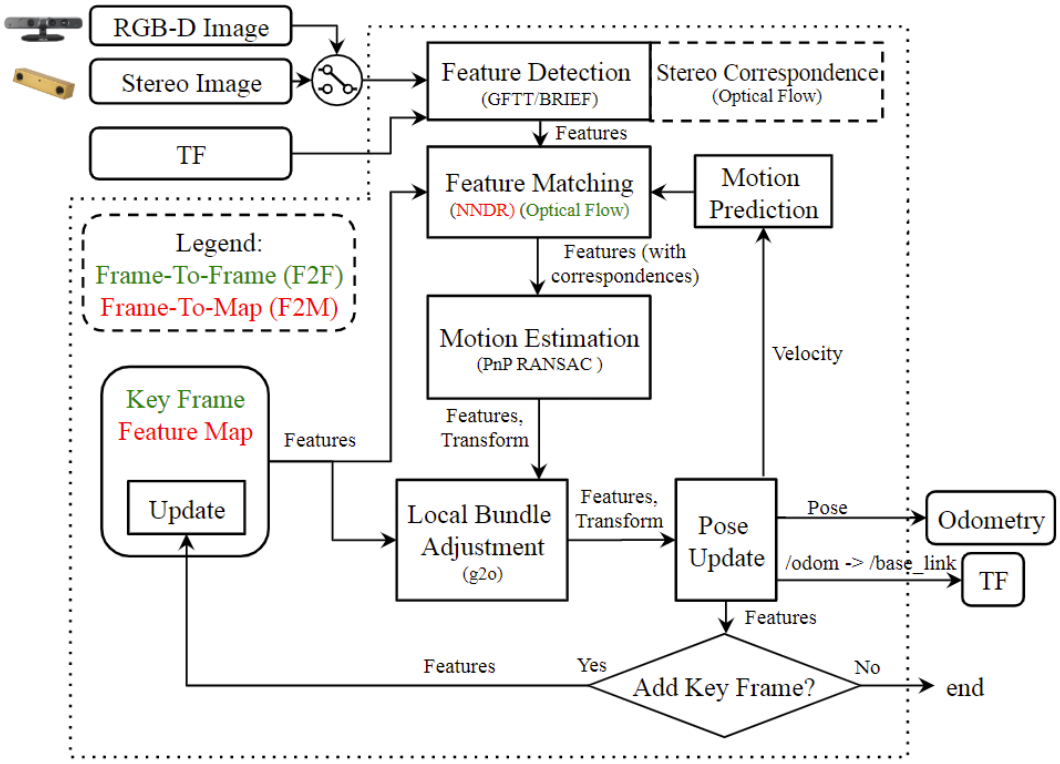


Figure 18: Overview of the RTAB-Map odometry [LM19]

Sensors and Odometry RTAB-Map accepts both range (LiDAR) and visual sensors as inputs to the system. As for the odometry, the system accepts the primitive forms of odometry like wheel odometry and Inertial Measurement Unit (IMU)-based odometry and more complex ones like LiDAR and visual odometry(VO). For Visual Odometry (VO), RTAB-Map uses two standard visual odometry methods Frame-to-Map (F2M) and Frame-to-Frame (F2F) [FS12]. As the name suggests, one method registers the current frame against a keyframe, and the other registers it against the map. When implemented for LiDAR scans, they are called scan-to-scan and scan-to-map registration respectively.

Visual Odometry (VO) VO under RTAB-Map is often used when there is no wheel odometry, or when the wheel odometry is noisy and cannot be used for the computations. The overall process of the visual odometry is represented in fig. 18. The whole odometry workflow consists of various steps that are described here.

- Feature detection is done using Good Features to Track (GFTT) [ST94] by default, and the threshold is set using *Vis/MaxFeatures* parameter. All the feature detectors that are available with OpenCV are supported under RTAB-Map. And the choice

of feature detection is made depending on the scenario that the system is used in. In the case of RGBD images, the depth image is used as a mask for GFTT to avoid extracting features from regions with invalid depth.

- Feature matching is done using a basic nearest neighbour search, using the Nearest Neighbour Distance Ratio (NNDR) test [Low04] using BRIEF descriptors of the extracted features against those in the feature map. NNDR is defined using the parameter *Vis/CorNNDR*.
- Motion prediction is made using a motion model that estimates the position of the features in the current KF based on previous motion transformation. This limit on the window for feature matching facilitates a better and optimised scheme for matching in difficult environments. The window is set using the parameter *Vis/CorGuessWinSize*, and a constant velocity model is used here.
- Motion Estimation is done using Perspective-n-Point (PnP) Random Sampling and Consensus (RANSAC) implementation of OpenCV library is used to compute the transformation of the current frame to that of the Keyframe (KF) or the feature map. The minimum threshold is set by *Vis/MinInliers*.
- Local Bundle Adjustment (BA) is performed on the features of all KFs in the feature map.
- Pose update is performed with the estimated transformation and the *tf* `/odom` → `/base_link` is updated.
- KF and feature map update is updated if the number of inliers computed in the motion estimation step is less than the *Odom/KeyFrameThr* threshold.
- When the feature map is over the threshold *OdomF2M/MaxSize*, the oldest frames which are not matched with the current frame are removed. If a KF does not have features in the feature map anymore, it is discarded.

Synchronization As described earlier, RTAB-Map accepts inputs of various formats (e.g., RGBD, stereo, wheel odometry, 2D and 3D LiDAR laser scans), and combinations of these sensor streams can be used when there are more than one available. As sensors do not publish data at the same rate and at the same time, appropriate synchronization is needed so that the information can be correlated with each other. Under ROS, there are 2 kinds of synchronizations available - approximate and exact. An approximate synchronization correlates data frames with minimal time differences by comparing their timestamps whereas the exact method, as the name suggests, requires that the data frames have the exact same timestamp [LM19].

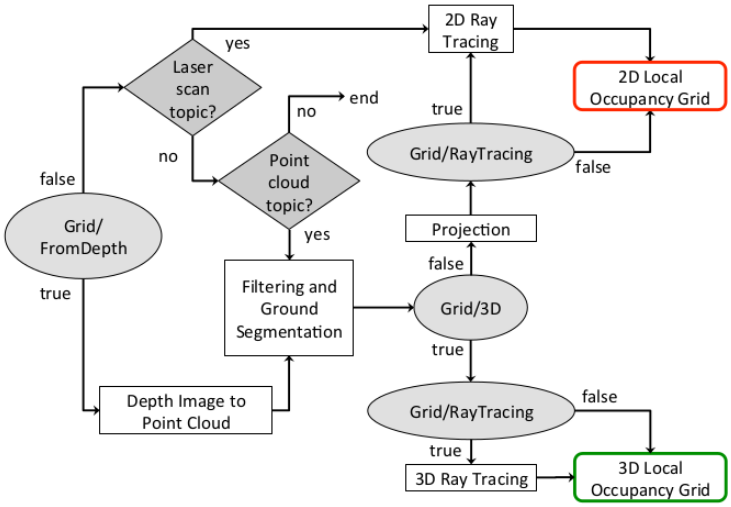


Figure 19: Process overview of the STM's OGM creation

Short Term Memory (STM) and Local Occupancy Grid Creation When a new node is created in STM, a corresponding local Occupancy Grid Map (OGM) is computed from the depth image. The OGM, referred to as local occupancy grid in [LM19], is referenced in the robot frame and consists of empty, obstacle, and ground cells at the cell size set by *Grid/CellSize*. The total size of the local occupancy grid is defined by the sensor that is used to create it. These local occupancy grids are then used to generate global occupancy grids by transforming them based on the poses in the map graph. Depending on the parameters *Grid/FromDepth* and *Grid/3D*, and the input topics, the local occupancy grid is generated differently in 2D or 3D. If the parameter *Grid/FromDepth* is false and `/rtabmap` node is subscribed to a laser scan topic, a local 2D occupancy grid is created. The choice depends on what kind of global map is required for the application and on the processing power available. If *Grid/FromDepth* is false and no laser scan and point cloud topics are subscribed, no grids are computed [LM19].

Loop Closure Loop closure is performed using the BoW approach in this paper. When a new node is created, the STM extracts the visual features and stores them in the form of a visual vocabulary. When visual odometry F2F or F2M is used, it is possible to reuse features already extracted, for odometry for loop closure detection. As loop closure detection does not need as many features as odometry to detect loop closures, and to reduce computation load, only a subset (maximum set by *Kp/MaxFeatures*) of the odometry features are quantized to visual word vocabulary. The other features are still kept in the node when loop closure transformation has to be computed. The created

node is then compared to nodes in WM to detect a loop closure. Locations in STM would be very similar to the last locations and would therefore bias the loop closure hypotheses on them. When a loop closure hypothesis reached the fixed threshold *Rtabmap/LoopThr*, a loop closure is detected and transformation is computed [LM19].

Graph Optimization When a loop closure or a proximity frame is detected or some nodes are retrieved or transferred because of memory management, a graph optimization approach is applied to minimize the errors in the map. It integrates three graph optimization approaches: TORO [BBS08], g2o [Küm+11] and GTSAM [DC22]. g2o and GTSAM converge faster than TORO, but are less robust to multi-session mapping when multiple independent graphs have to be merged together. TORO is also less sensitive to poorly estimated odometry covariance. However, for a single map, based on empirical data, g2o and GTSAM optimization quality is better than TORO, particularly for 6DoF maps. GTSAM is slightly more robust to multi-session than g2o, and thus is the strategy now used by default in RTAB-Map [LM19].

Visual loop closure detection is not error-free, and very similar places can trigger invalid loop closure detections, which would add more errors to the map. To detect invalid loop closure or proximity detections, RTAB-Map now uses a new parameter. If a link's transformation in the graph after optimization has changed more than the factor *RGBD/OptimizeMaxError* of its translational variance, all loop closure and proximity links added by the new node are rejected, keeping the optimized graph as if no loop closure happened [LM19].

Global Map Assembly The global map output options that can be assembled from the local occupancy grid maps are shown in Figure 20. A 3D output in the form of a 3D point cloud is the most desirable output, as the 3D map can be used to generate all other forms of output. When a new node is added to the map, the new local occupancy grid is transformed and combined with the global occupancy grid. When a loop closure occurs, the global map is re-assembled according to the new optimized poses for all nodes in the map graph. This process is required so that obstacles that have been incorrectly cleared before the loop closure can be reincluded. The point cloud outputs assemble all points of the local maps and publish them in the standard sensor `msgs/PointCloud2` ROS format. Voxel grid filtering is done to perform smoothing, which can also be done in post-processing [LM19].

To summarise the discussion, RTAB-Map is proposed as a multi-purpose graph-based SLAM approach that can be used out-of-the-box by users and for prototyping on robot platforms with various sensor configurations and computational capabilities. Sensors required for SLAM have limitations that influence localization accuracy, map quality,

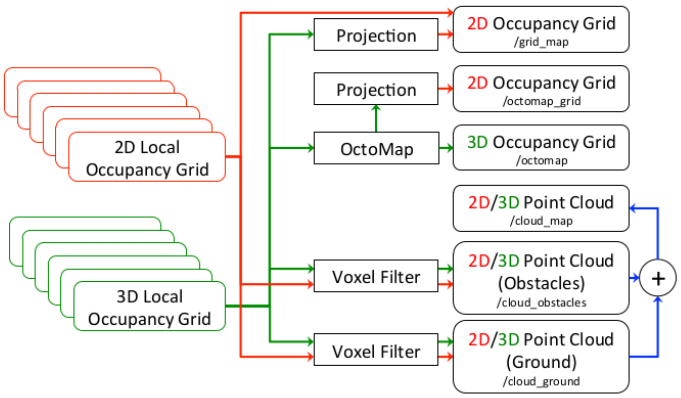


Figure 20: Global Map Assembly procedure under RTAB-Map

Feature	LSD SLAM	ORB SLAM 2	DTAM	RTAB Map
Type	Direct	Feature-based	Direct	Feature-based
Map Density	Semi-dense	Sparse	Dense	Dense
Loop Closure	Yes	Yes	No	Yes
Long-term Localization	No	Yes	No	Yes
Availability[5]	Yes	Yes	No	Yes

Table 1: Comparison of the features offered by the SLAM Methods

and computing resources and this is something that the user has to keep in mind while deploying this method.

The methods discussed in this section are now summarised in Table 1. From the studies, RTAB-Map offers numerous advantages in terms of loop-closure and long-term localization, which are important factors that come in use when performing localization in large areas (like in the case of the proposed use-case). Even though LSD SLAM and DTAM offers good mapping characteristics, it lacks long term localization and is more suited for short range application (like VR). ORB SLAM 2 offers long-term localization but it is a KF based approach and therefore is very much dependent on the availability of the features upon revisiting the same location. RTAB-Map on the other hand offers an advanced memory management scheme which faclitates fast and efficient relocalization and therefore surpasses ORB SLAM 2 in terms of relocalization and loop-closure.

[5]Availability of ROS implementations

3.2 Similar Studies

Mapping of indoor spaces is a relatively active research area, especially since the advancement of indoor service robots. In this section, studies of similar manner are visited and a good understanding of the current trends in the technology and the arguments for the choice of certain methods are derived.

For the purpose of clear understanding, the section is divided into 3 parts. The first part deals with the studies that compare different studies that have come out recently which compare the various SLAM algorithms against each other. Subsequently, research works dealing with robot-based indoor mapping are visited. Lastly, the choice of map-building solution is justified.

Studies comparing various SLAM algorithms

Numerous studies compare the applicability of the available SLAM solutions against each other. One of the most recent work that compares the major SLAM algorithms is the work by Macario Barros et al. [Mac+22], which compares various state-of-the-art algorithms and evaluates them using some of the important criteria. The criteria chosen in this paper are as follows.

- Algorithm type indicates the methodology using which the SLAM system operated. The classification for this criteria (in relevance to this paper[6]) are feature-based, direct and hybrid.
- Map density plays a role in the requirement of the computational resources and the readability of a map. The sparse maps are easier to compute but harder to interpret (for a human) but on the other hand, dense maps are visually comprehensible but computationally very exhaustive.
- Global optimization is a technique used to search and optimize accumulative errors that may have been inflicted during the mapping process.
- Loop closure is the ability of the system to identify previously visited locations during the mapping process and perform optimizations accordingly.
- Availability describes if the work is available for use by the authors as open-source distributions or by third parties to be used in custom applications.
- Embedded implementations tells us if the studied SLAM method is available on an embedded platform, because after all, the algorithms find most relevance in robotic applications and this is an important factor when it comes to the comparison.

[6]This paper visits Visual, Visual-Inertial, and RGB-D methods

Under the classification of RGBD SLAM, it compares KinectFusion [New+11], SLAM++ [Ila+17], RGBDSLAMv2 [End+14], DVO [FPS14][For+17], and ORB-SLAM 2 [MT17]. This work, apart from comparing the current trends in SLAM, also discusses the open problems and the scope of expansion of the problem statement into domains like semantic localization, deep learning-based methods etc. It also compiles various benchmark datasets that were available for the comparison of SLAM algorithms and therefore serves as a good starting point for a comparative study of this manner [Mac+22].

A work from the University of Padoa, Italy, by Giubilato et al. [Giu+19], compares the ROS-compatible stereo-SLAM methods on a specific hardware - The NVIDIA Jetson TX2. It compares methods like ORB-SLAM2 [MT17], SPTAM [Pir+17], and RTAB-Map [LM19]. It focusses these methods on the grounds that are described here.

- Method, describing the kind of approach the algorithm takes (SLAM/Visual Odometry)
- Input type, being monocular/stereo/depth.
- Feature type, describing the feature extraction method used in the algorithm.
- Frontend and backend algorithm, if they support bundle adjustment based global optimization.
- If they perform loop closure or not.

This experiment is also evaluated computationally, by performing Graphical Processing Unit (GPU) based image rectification, thereby providing more processing overhead for the localization processes. This has shown to have only marginal effect on the tracking characteristic. The authors also perform Contrast-Limited Adaptive Histogram Equalization (CLAHE) [K94] on the input images to enhance the feature detection on various parts of the image with high dynamic range and therefore making the tracking more robust. The latter modification shows a significant reduction in the localization characteristic, as the ATE_{avg} for ORB SLAM 2 is improved from 0.28m to 0.1, and for RTAB-Map, the ATE_{final} reduces from 3.75m to 0.06m. The paper also experiments with ZED-VO[7], and exhibits superior tracking characteristics. [Giu+19]

Another work that performs the comparison of ROS compatible methods for mapping and navigation systems is the work by Da Silva, Xavier, and Gonçalves [DXG19]. This paper compares the methods GMapping [GSB07] and RTAB-Map, experimentally, and compares their SLAM accuracy, map quality and map usability. This work focuses on the practical aspects of the algorithms and relate their performance with the internal configurable parameters of the algorithm. This method also performs the experiment on a synthetic scenario, simulated in Gazebo, and also on benchmark datasets. The authors

[7] currently integrated with ZEDfu, the factory SLAM solution for ZED cameras

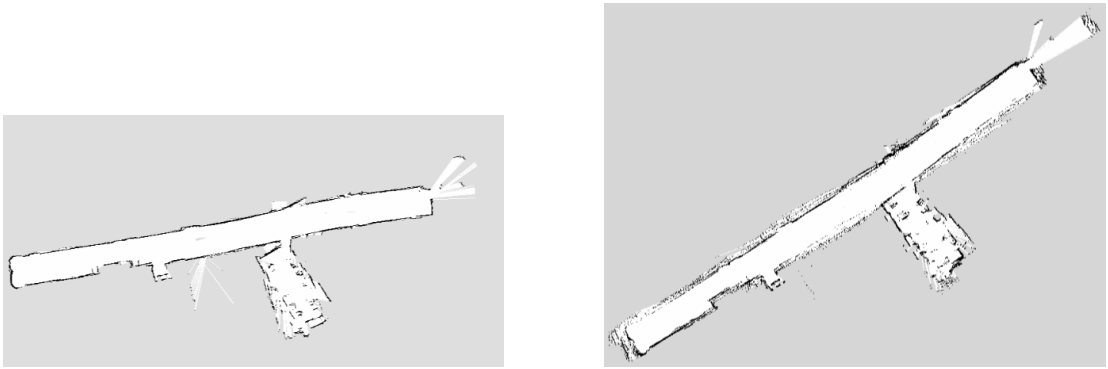
also produce datasets that involves a lot of challenging but relevant scenarios for mobile robots. This work compares the two SLAM algorithms on the following aspects.

- SLAM performance is estimated using the accuracy of the algorithm against a Ground Truth (GT). The ATE and RPE is calculated and serve as an indicator for the performance of the algorithm.
- Mapping performance is visually inspected to see if the generated maps are coherent with the static parts of the environment.
- It also considers the possibility of map reuse, whereby the generated map should be usable for other entities, like a visualization software or a navigation algorithm.

During the simulation experiments, it is observed that GMapping presents a lower error than RTAB-Map because the simulation environment provides the method with noise-free odometry and this is usually not the case in the real world. RTAB-Map on the other hand uses appearance-based matching (visual-odometry) and therefore is prone to noise. When configured to use robot odometry, the latter also presents good-quality mapping characteristics.

When it comes to real-world mapping problem, RTAB-Map is shown to have better mapping characteristics as it offers appearance-based loop closure, which is not the case for GMapping. GMapping uses scan matching in combination with other thresholds to perform loop closure and in this case, the visual words are a much superior method to do so. The RTAB-Map used here is configured to estimate odometry using both robot odometry and the emulated laser scan from the depth camera, and therefore presents comparable results as shown in Figure 21. The research is concluded by selecting RTAB-Map as a preferred and a more complete solution in the context of ground robits with an RGB-D sensor, owing to its mapping quality, robustness of localization etc [DXG19].

The work by Filipenko and Afanasyev [FA18] which compares a few of the SLAM systems for a mobile robot, specifically set in an indoor environment. In this experiment, the prototype is built to perform the comparative study. The prototype, as shown in fig. 22 contains a 2D LiDAR, a monocular camera and a ZED stereo camera. An NVIDIA Jetson TX1 is used as the computer for the system. The experiment is conducted by teleoperating the robot in a closed-loop trajectory of an indoor environment of an office-like scenario. The data is recorded in a rosbag and then later used for each of the SLAM pipelines to calculate the error (ATE) (listed in table 2). The methods that are compared here are Cartographer, LSD SLAM, ORB SLAM (mono), DSO, ZEDfu, RTAB-Map, ORB SLAM (stereo) and S-PTAM. From the tested SLAM methods, RTAB-Map offers the most promising result with the lowest RMSE for mobile robot localization in the indoor environment. Figure 23 shows the obtained trajectory using the methods RTAB-Map,



(a) GMapping (b) RTAB-Map

Figure 21: Comparison of mapping output of the two methods [DXG19]

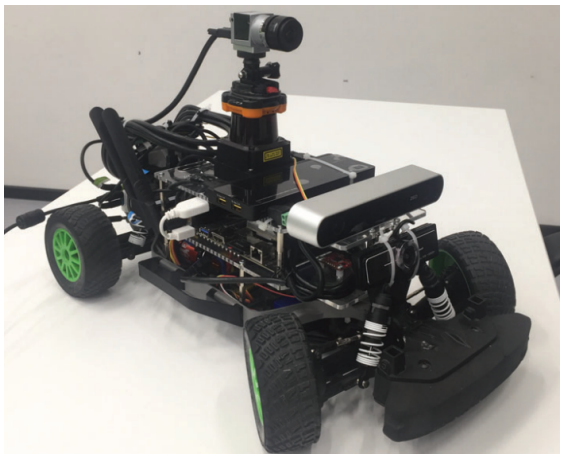
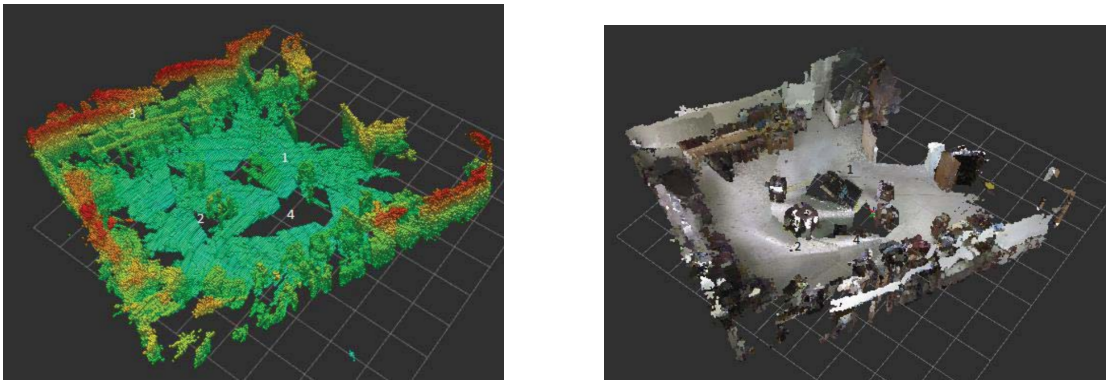


Figure 22: Prototype of test vehicle for evaluation [FA18]

Algorithm	RMSE (m)	Mean (m)	Median (m)	σ(m)	Min (m)	Max (m)
ORB SLAM 2	0.19	0.151	0.102	0.115	0.004	0.414
RTAB-Map	0.163	0.138	0.110	0.085	0.004	0.349
ZEDfu	0.726	0.631	0.692	0.358	0.002	1.323

Table 2: Comparison of ATE for various methods studied [FA18][8]



(a) OctoMap (b) RTAB-Map

Figure 24: Comparison of mapping output of the two methods

ORB SLAM 2 and ZEDfu are shown here.

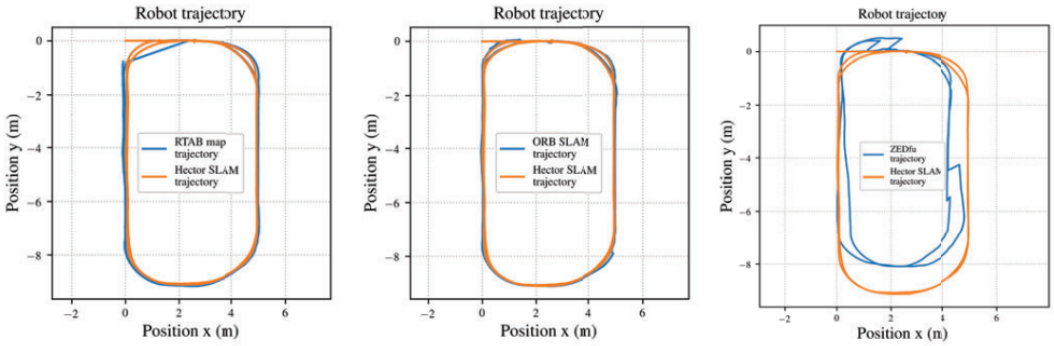


Figure 23: Comparison of generated trajectories using RTAB-Map, ORB SLAM 2 and ZEDfu [FA18]

This work is concluded by selecting ORB SLAM2 and RTAB-Map as desirable candidates for Stereo and Visual SLAM methods owing to their accuracy and robustness. [FA18]

Comparison of Mapping Strategy The paper by De Silva et al. [De +18] compares the 3D mapping strategy and results of RTAB-Map [LM19] and OctoMap [Hor+13]. In this paper, both methods are used to map a dynamically changing environment and this presents a real-world case to assess the suitability of these two methods. The experimentation is done using a Kobuki Mobile Research Base[9]. Figure 24 shows the

[8]Only the algorithms that are relevant to this paper is listed
[9]`http://wiki.ros.org/kobuki`

result of mapping using the two methods. The OctoMap represents the environment as a set of color-coded voxels that can be efficiently stored and does not give importance to the comprehensibility aspect of mapping. RTAB-Map on the other hand, generates a 3D map that is visually accurate, even though being geometrically inaccurate in many instances.

Furthermore, RTAB-Map claims the upper hand in various aspects like odometry being used for trajectory, and ease of integration into robotic applications for navigation and path-planning operations. OctoMap poses advantages in terms of computational ease (and therefore speed) but since the scope if this paper lies in the map building aspect, RTAB-Map is a suitable choice.

Finally, a summary of all the works reviewed in this section is summarised in the following table 3.

The key takeaways from this review of similar studies are as listed here.

- The work by Macario Barros et al. pits the ORB SLAM 2 as a candidate amongst DVO, RGBDSLAM, KinectFusion, and other prominent methods. And according to this study, ORB SLAM 2 offers good performance in terms of localization performance, loop-closing etc. But it falls short in case of map quality (density) which is a requirement.
- The work by Giubilato et al. compares the performance of ORB SLAM 2 and RTAB-Map on a embedded platform and therefore provides a good outlook on how the methods would be implemented on a robotic system. This paper also discussed the possible improements that can be made to the SLAM pipeline, in terms of preprocessing input and parallelising the core processes of the SLAM system.
- The paper by Da Silva, Xavier, and Gonçalves compares RTAB-Map and GMapping in simulation and studies the configurability of both systems and how it can reflect in the overall mapping output of the system. This paper also discusses map reuse, which is a point of interest to the research introduced in this thesis.
- Finally, the paper by Filipenko and Afanasyev compares all 3 proposed methods in this thesis – LSD SLAM, ORB SLAM 2, and RTAB-Map. It studies the indoor localization performance and pits RTAB-Map as a good candidate for the use-case. This is in agreement with the conclusion that was arrived at previously section 3.1.

Work	Source	Year	Compared Algo.
A Comprehensive Survey of VSLAM Algorithms	[Mac+22]	2022	KinectFusion, SLAM++, RGBDSLAMv2, DVO, ORB SLAM 2
An evaluation of ROS-compatible stereo VSLAM methods on a NVIDIA Jetson TX2	[Giu+19]	2019	ORB SLAM 2, SPTAM, RTAB-Map
Mapping and Navigation for Indoor Robots under ROS: An Experimental Analysis	[DXG19]	2019	GMapping, RTAB-Map
Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment	[FA18]	2018	LSD SLAM, ORB SLAM 2, RTAB-Map
Comparative Analysis of Octomap and RTABMap for Multi-robot Disaster Site Mapping	[De +18]	2018	OctoMap, RTAB-Map

Table 3: Summary of the papers reviewed in this section

4 Implementation

In this chapter, the proposed system is designed and modeled using systems engineering tools and the specifics of how the experiment is organized are visited in detail. The list of requirements is formulated first, in order to set the expectations from this project. Subsequently, a set of experiments are devised so that the proposed system can be evaluated against the set requirements so as to establish the suitability of the said designx.

Requirements

- The primary requirement of this system is to produce a 3D map of its environment or the defined work area that can be visualized by a human operator.
- The 3D map is estimated in good quality, with resolution in the range of 1cm voxel grids with an error limit of around 10-15%.
- 2D maps are also to be created, which can then be used by other robots to navigate around the work area.

As shown in the fig. 25, the proposed mapping system interacts with the user, the worker robot, and the environment (modeled as actors here). The user can command the mapping system directly, whereas the worker unit only receives the map information from the planner of the mapping system.

4.1 Experimental Setup

The experiment involves using the stereo camera (or the sensor under study) to record sequences of the environment as the user/robot moves through the environment and a map being created and updated in near-real-time. The software and hardware setup required for the experiment are elaborated here.

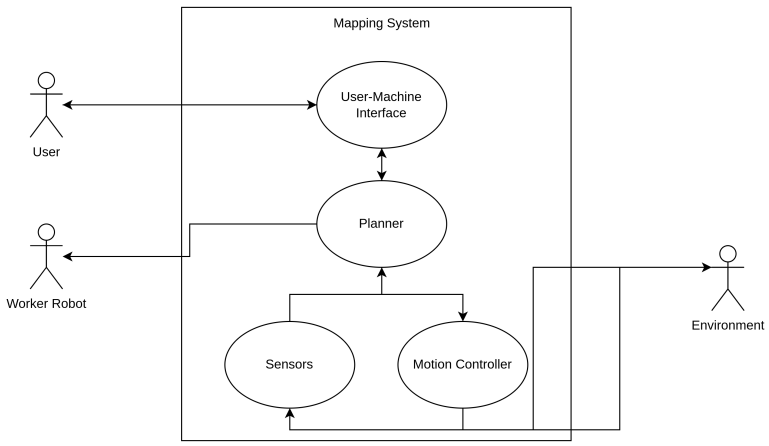


Figure 25: Proposed use-case for the robotic system

4.1.1 Software Setup

This project uses a variety of input data streams coming from various sensors and all of this information is processed on a computing environment. A very popular environment that roboticists use is ROS[1].

Robot Operating System (ROS) ROS is an open-source framework that is used to build and integrate robotic systems and enables intercommunication between the various units like sensors, actuators, and the processing unit of the robot. It provides a common set of tools and conventions that make the process of developing complex robotic systems more easy and streamlined.

ROS also provides users with functionalities to simulate the robot under test. A popular simulation platform that is used with ROS is Gazebo[2]. Gazebo offers features like the simulation of multi-body physics, advanced 3D graphics, and simulations for sensors and actuators and it is even capable of modeling the noise in the systems. It also offers various plugins that enable external functionalities like HIL and SIL. Additionally, Gazebo supports physics simulations so that collisions, forces, and friction effects can be incorporated into these tests. This feature makes it possible to validate control systems and navigation strategies under a wide range of circumstances.

[1] `https://www.ros.org/`
[2] `https://classic.gazebosim.org/`

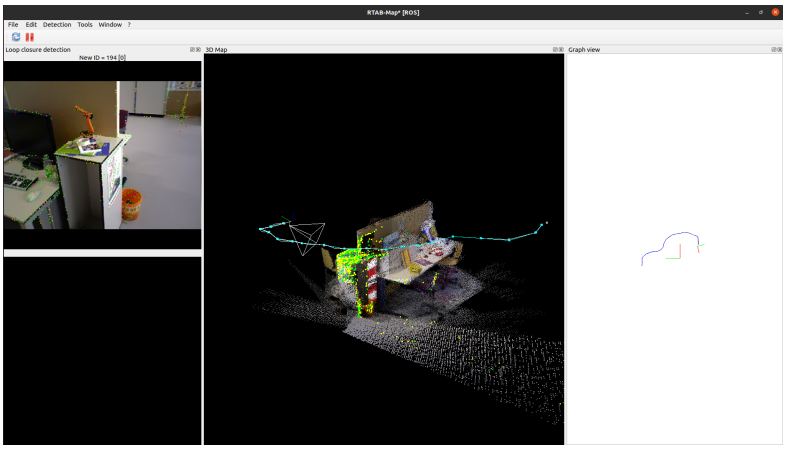


Figure 26: The RTAB-Map Standalone UI

RTAB-Map Another important software that is used very widely in the implementation is the RTAB-Map software[3] made available by the authors Labbé and Michaud as a standalone application for testing and evaluation. This software provides the user with a plethora of interfaces such as the visualizer shown in the fig. 26 to view and debug the SLAM pipeline and tune the parameters[4] in order to tune the algorithm for the specific application.

MeshLab MeshLab[5] is an open-source system for working on 3D data like point clouds and triangular meshes. It provides the user with a set of tools for editing, cleaning, healing, inspecting, rendering, texturing and converting 3D meshes. [**MeshLab**]

In this work, MeshLab is used to perform measurements on the point clouds and use it for evaluation. It also offers the capability for mesh alignment and comparison, and this functionality can also be used for comparing the scans with the synthetic ground truth[6].

4.1.2 Hardware Setup

The experiment is set up on two hardware instances. The first part of the experiment is carried out on a standard computing environment (described in table 4) and this computer was set up with all the necessary software that is required to run the experiments. Apart

[3] `http://introlab.github.io/rtabmap/`
[4] RTAB-Map parameters described in Section 3.1.4
[5] `https://www.meshlab.net`
[6] Explained in Section 4.2.2

Parameter	Configuration
Device Model	Dell Latitude 5490
Processor	Intel i7 8650
Processor Spec	8-core 3.7 GHz
Memory	16 GB
OS	Ubuntu 20.04
ROS Version	Noetic Ninjemys

Table 4: Specifications of the laptop

from the laptop computer, an NVIDIA Jetson is also set up as an embedded computing unit. The NVIDIA Jetson TX2 is a high-performance computer designed for application in robots, drones, and other intelligent machines. It supports a range of operating systems including Linux also includes a variety of hardware and software features that are specifically designed for robotics and other autonomous systems. Another key feature of the platform is its support for real-time processing, with the ability to process multiple high-resolution video streams in parallel. This makes it ideal for applications such as object detection, tracking, and localization, which require real-time processing of large amounts of data.

All these features, with the support for advanced computer vision and deep learning algorithms, make it an excellent choice for this particular research experiment.

4.2 Experimentation

This section visits the 3 proposed experiments in detail. The experiments are divided as

- Experiments on VSLAM Benchmarks
- Handheld Mapping
- Simulating the system in ROS

The firsts experiment is proposed so that the localization/trajectory accuracy of the system can be established. The handheld experiment itself is divided into 2 parts – one to establish the mapping accuracy, and the next one to verify the accuracy of the spatial localization of the system. The last part shown (in simulation) how the system can be implemented on a commercial robot and can be used to perform SLAM.

4.2.1 Visual SLAM Benchmarks

The first experiment involves using the Visual SLAM benchmarks that were introduced by Sturm et al. in [Stu+12] to serve as an evaluation standard for the various RGB-D SLAM algorithms. The work showcases a large set of image sequences recorded using a Microsoft Kinect[7] with highly accurate and synchronized ground truth information. The ground truth is generated with a motion capture system from MotionAnalysis[8] with a millimeter-scale accuracy and with a measured standard deviation of 1.96mm.

The dataset itself composes of recordings in numerous settings and environments, a subset of which is going to be used in the subsequent experiments. The whole dataset is organized into various groups, which are

- Testing and Debugging
- Handheld SLAM
- Robot SLAM
- Structure v/s Texture
- Dynamic Objects

of which, the first three sets are of interest to this experiment. Since the proposed system here is aligned towards indoor mapping for a factory/office environment, the datasets under test are also chosen accordingly. All recordings are available as compressed archives, and also in the form of ROS bags. ROS bags serve as a method for conveniently packaging information from multiple sensors, in this case - camera and depth information.

fr2/xyz This set of recordings is mainly intended to perform the baseline evaluation and testing of the SLAM systems. The recordings in this series are composed of linear motions, not very drastic so that the system does not lose track, and the user can ensure that the basic feature extraction and keeping track of features is performed properly.

fr2/desk For this sequence, a typical office desk with computers and other office equipment like a computer monitor, keyboard, phone, etc. are presented to the sensor. This is an example of a feature-rich scene. The Kinect is moved around the table so that there is a loop closure.

[7] `https://en.wikipedia.org/wiki/Kinect`
[8] `https://www.motionanalysis.com/`

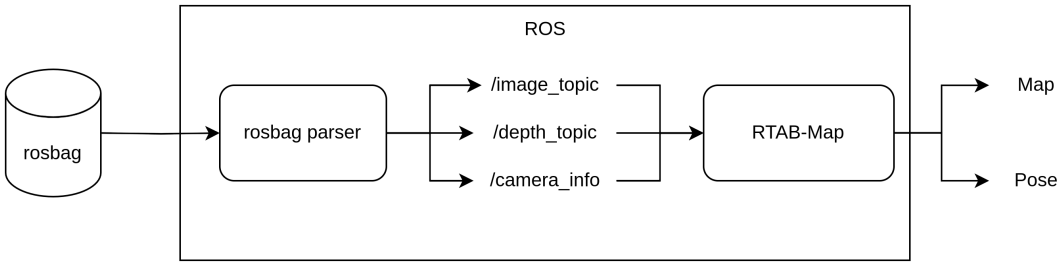


Figure 27: Data flow in the rosbag experiment

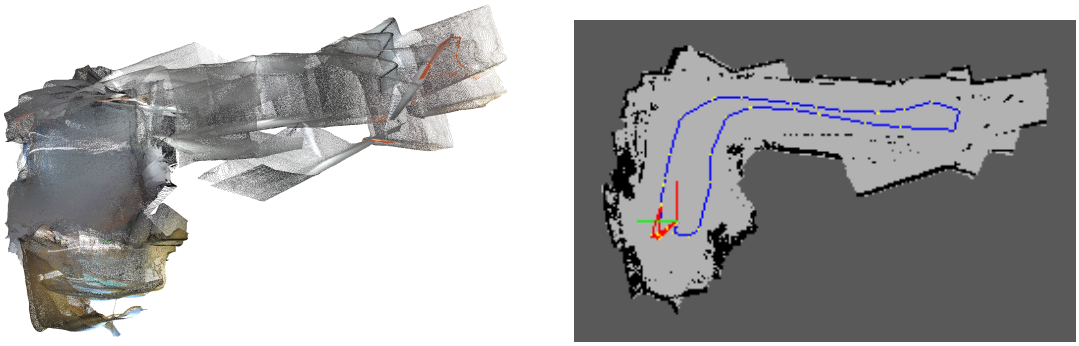
fr3/long_office_household In this sequence, a different sensor, Asus Xtion was used to perform the measurements. The sensor was moved through an office environment similar to the case in the previously visited desk example but with a longer path length. A loop closure is also present in this recording so this also can be evaluated.

fr2/pioneer_360 In the next 2 sequences, the Kinect is set up on a Pioneer robot and the robot is driven through a maze of objects including tables, chairs, and walls. In this recording, the robot turns at its position for more than a 360 degree turn. This can serve as a good example scenario for a scene with large camera movement.

fr2/pioneer_slam In this sequence, the Kinect is set up on a Pioneer robot and the robot is driven through a maze of objects including tables, chairs, and walls. There are several loops in the path where the map is constructed.

The experiment is run by playing the associated rosbags and then using the RTAB-Map software to process the information from the bag files. The data is stored in the bag files in the same way that they are published in the ROS network. The topics that were recorded are published by the parser in the same format and timestamps and the ROS frontend of the RTAB-Map, `rtabmap-ros`[9] subscribes to the appropriate topics as shown in fig. 27. The rtabmap-ros package is responsible for receiving the input topics from the ROS network and feeding them into the RTAB-Map pipeline. The system then performs the mapping and then generates a database file, which can subsequently be used for review, debugging and can also be optimized by editing the parameters.

[9] `http://wiki.ros.org/rtabmap_ros`



(a) 3D Pointcloud (b) 2D Occupancy Grid

Figure 28: A map of a corridor generated with poor mapping parameters

4.2.2 Using a stereo camera for Handheld Mapping

In this experiment, a ZED 2 stereo camera from Stereolabs[10] is the 3D sensor of choice. The previous experiments with the **VLSLAM!** (**VLSLAM!**) benchmarks were intended at establishing the localization accuracy of the system, this experiment is proposed to measure the mapping accuracy (structural) in the case of a handheld mapping experiment and evaluate the dimensional error.

Mapping Accuracy It can be established that a good map of the environment has been created if the 3D map satisfies the following criteria

- The map looks visually comprehensible. The human operator should be able to able to identify locations and landmarks so that the region of interest and the related operations can be specified.
- The map must be usable by the robotic system for navigation.

To put this into perspective, Figure 28 shows a scenario where the quality of the map is inadequate, and the robot cannot use this map for a localization/navigation operation.

The proposed method covers the estimation of the map quality by a series of experiments. All the experiments are carried out with the same sensor, the Stereolabs ZED 2. The ZED 2 camera is a high-performance stereo camera designed for depth sensing and 3D mapping applications. It is equipped with an ultra-high-definition stereo camera, providing up to 60 FPS at 1080p and a maximum depth range of 40 meters. The camera also includes an integrated IMU module, making it ideal for use in cases involving

[10] `https://www.stereolabs.com/`

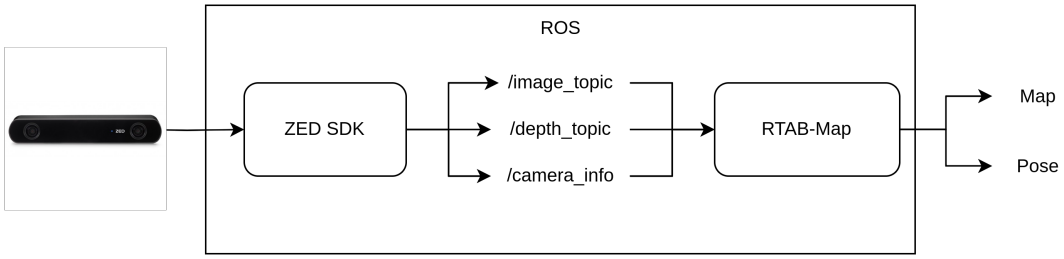


Figure 29: Data flow in handheld mapping

multimodal sensing. One of the key features of the ZED 2 camera is its advanced depth-sensing capabilities. The camera uses a combination of stereo vision and machine learning to provide accurate depth measurements with high precision. The ZED camera is supplemented with a software development kit, the ZED SDK, which provides the software tools for application development and interfacing with the hardware. The SDK also provides ROS wrapper[11] that can employ the full functionality of the SDK within ROS.

The RTAB-Map is available as a ROS package and contains the main node called `rtabmap` and this node is responsible for the management of visual memory and node generation. The odometry computation is performed by `rgbd_odometry`, `stereo_odometry` or `icp_odometry` (based on the choice of odometry) and this information is used by the main node to publish the current map under the topic `/grid_map` or `/cloud_map` in the form of an occupancy grid or a point cloud respectively. In this experiment, only visual odometry is used (stereo) to compute the camera motion.

The experiments are laid out as follows:

1. Mapping of an object of known dimensions
2. Mapping of an area with known dimensions
3. Evaluating the error metrics using GT

Mapping an object/area of known dimensions

This experiment is aimed at measuring the spatial and geometric accuracy of the mapping pipeline. The setup involves using the mapping pipeline to scan an object that is placed in the region that is being mapped. This object is of known dimension and therefore can serve as a viable ground truth for estimating the mapping accuracy.

[11] `https://github.com/stereolabs/zed-ros-wrapper`

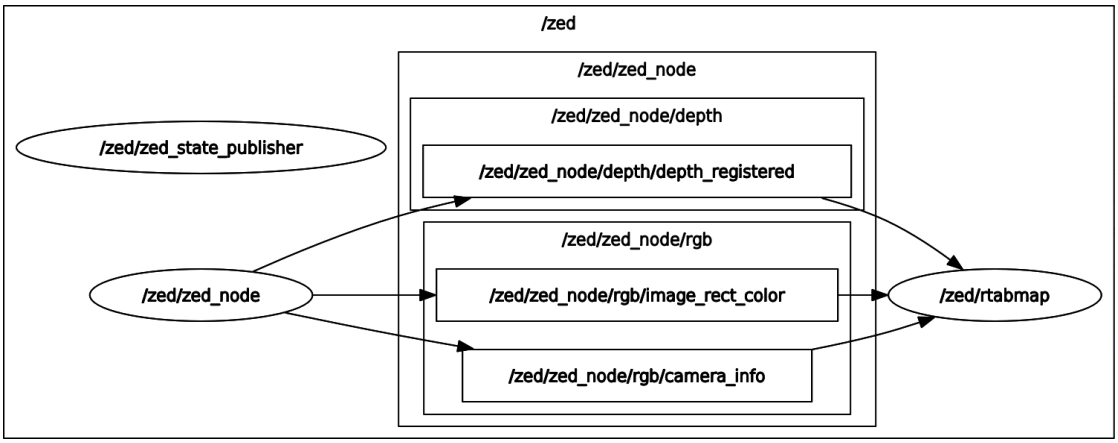


Figure 30: The node graph associated with the ZED-RTABMap example

The RTAB-Map pipeline is started by invoking the ZED-RTABMap example launch file by using the following command:

```
roslaunch zed_rtabmap_example zed_rtabmap.launch
```

This starts the `zed/rtabmap` node and subscribes to the image and camera topics from the `zed/zed_node` (shown in fig. 30). Once the mapping process is completed and exited, the map data is stored in a database and can be further used for post-processing and evaluation.

Initially, the experiment is performed on the laptop with the RTAB-Map (standalone) software running. The stereo camera is used to perform an orbital scan of the object placed on the floor with consistent lighting. The resulting scan is then post-processed to remove outliers, incorrect registrations, and other similar inconsistencies.

This experiment is again repeated using the embedded computer, the Jetson TX2, with the same settings as the laptop instance. This is ensured by using the same configuration file for both executions.

When mapping a closed area (like a room or a hall), the map is visually inspected for consistency with the actual (or measured) floor plan of the structure. The GT measurements here can be made with any range-finding instrument and will serve as a viable GT for the evaluation of the mapping algorithm.

Evaluating the error

As explained in the previous section, the ground truth is essentially the measurements of the physical box of known dimensions. So the most straightforward way of evaluating

the geometric error is to compare the linear distances and the angular measurements of the object. Since a box is used in this experiment, the measurements of the edges in the 3D scan are to be compared with the actual measurements. The corners are also to be checked for right angles.

The scans are loaded into MeshLab and the measurement tooling that is available in the software can be used to make the observations of lengths. This can also be done within the RTAB-Map interface, but MeshLab provides a more streamlined workflow for the 3D measurement and pointcloud processing.

The linear measurements are then performed for each of the boxes' edges and the observations are tabulated. The experiment is repeated with varying distances and objects of different dimesions to also estimate the accuracy of reconstruction at various depth ranges.

4.2.3 Simulating RTAB-Map in Gazebo

As a preliminary to building the actual robotic system, the system is prototyped and demonstrated on a robotic simulation platform called Gazebo. Gazebo serves as a tool to simulate a Clearpath Robotics Husky[12], which is a standard open-source robotic platform used in robotic applications in the industry. Clearpath Robotics also provides the associated ROS packages for the implementation of the system in ROS. This system is then modified to add the appropriate sensors that are required for this solution (here, an RGBD camera) is attached on the robot as explained in the subsequent section.

The Robot As explained above, the Husky is chosen as a platform for the prototype implementation of the system. The Husky offers an advanced suite of highly accurate sensors like wheel odometry and LiDAR, but for the purpose of this experiment, the robot is set up to only use the RGBD camera as the input source to the algorithm. The depth camera plugin is developed using the OpenNI[13] plugin for Gazebo and the camera itself is modeled after the parameters of the Intel Realsense. This means that there is one color image stream and one highly accurate depth stream from the sensor. The sensor is attached to the front of the Husky robot (see fig. 31) and the coordinate transform for the camera center are as described in Unified Robotics Description Format (URDF) file. The appropriate topics for the camera stream and the depth stream can be now published to the ROS network.

[12] `https://clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/`
[13] `https://github.com/OpenNI/OpenNI2`

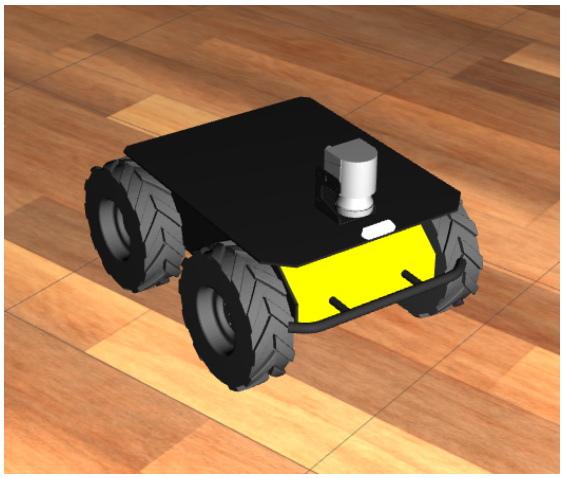


Figure 31: Husky model with the attached RealSense camera

The World The scenario that is used in this experiment is the Office world (shown in fig. 32) that is available from Clearpath Robotics. This is, as the name suggests, an office scenario that contains enclosed rooms with furniture and equipment. This world is a good choice for the simulation of indoor office scenarios and therefore is used in this experiment. This world is launched using the following command:

```
roslaunch cpr_office_gazebo office_world.launch
```

Another example scenario that can be used for experimenting is the Gazebo Factory world. This is very close to the use case that is described in this work, but the downside of this world is that it contains open edges, which is not very suitable for autonomous frontier exploration.

The Prototype Finally, the modified Husky is spawned in the simulated world in Gazebo using the following command:

```
roslaunch cpr_office_gazebo office_world.launch platform:=husky
```

Now RTAB-Map is launched by invoking the following command:

```
roslaunch rtabmap_demos demo_husky.launch slam2d:=true camera:=true
```

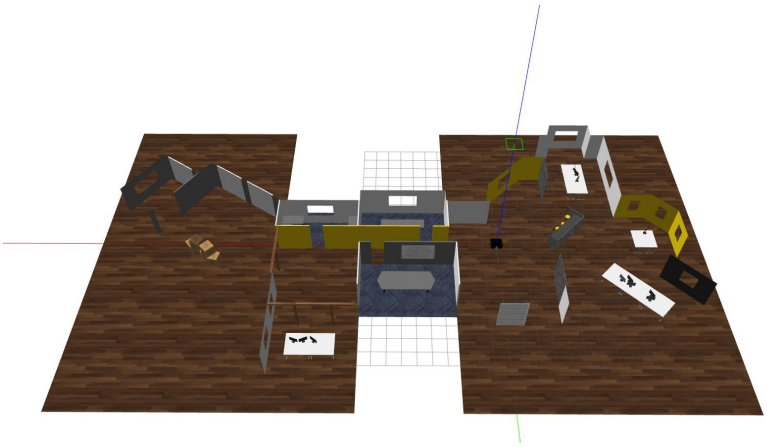


Figure 32: The Office World

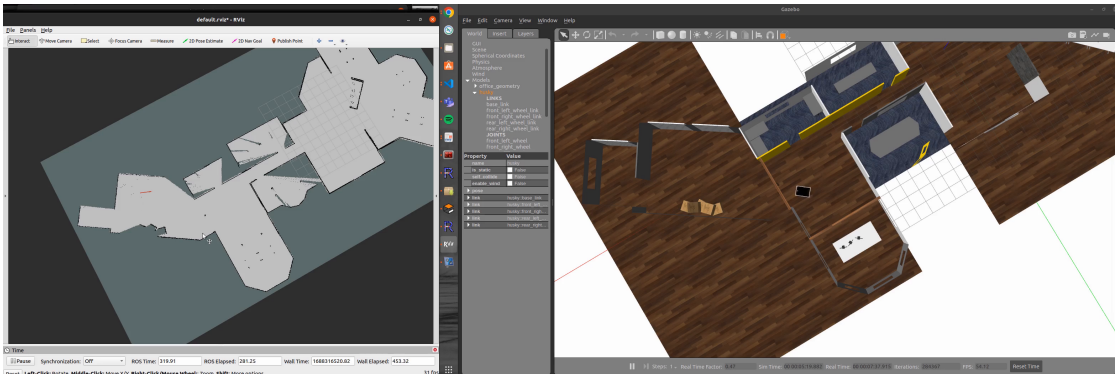


Figure 33: The mapping pipeline set up in Gazebo and RViz

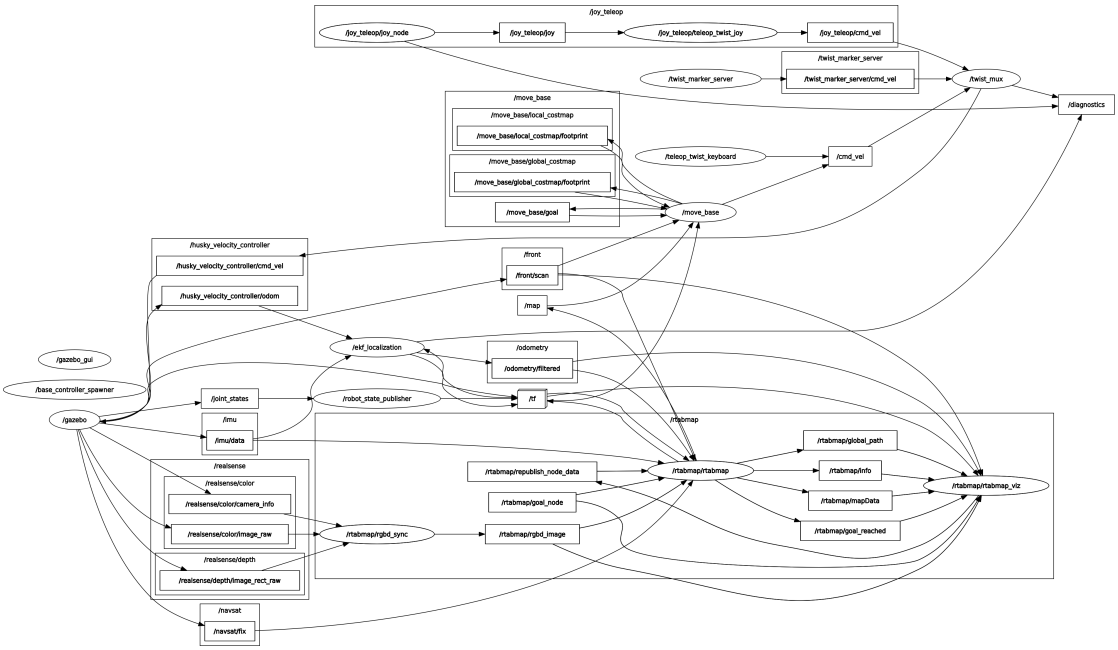


Figure 34: The node graph for the Husky prototype

and the started ROS node subscribes to the image and the depth topic streams available from the robot simulation. The node graph with the associated topics is shown in Figure 34. The robot is now driven in the world using teleoperation and navigation(using 2D nav goal tool in RViz like shown in fig. 33) to build a map of the environment. Once the robot has mapped the entire world, the generated map is saved to the computer using the command

```
rosrun map_server map_saver map_name.yaml
```

It is to be noted that the scope of this prototype is only to serve as a proof of concept for the solution on an actual robot. Real-world conditions like variable illumination, and wheel drift are not accounted into this prototype and therefore it is a very ideal form for the proposed solution.

4.3 Experimental Results

Experiments on the Benchmark Datasets The following data was collected after running the sequences fr2/xyz, fr2/desk, fr2/long_office_household, fr2/pioneer_360 and fr2/pioneer_slam through the RTAB-Map pipeline. The experiment is performed

	RTAB-Map		ORB SLAM 2	
	RMSE (m)	σ(m)	RMSE (m)	σ(m)
fr2/desk	0.029	0.00	0.065	0.05
fr3/long_office_household	0.018	0.01	0.070	0.05
fr2/pioneer_360	0.038	0.00	1.059	0.53
fr2/pioneer_slam2	0.045	0.02	1.709	0.18

Table 5: Comparison of localization error (ATE) from the benchmark sequences

both on the workstation and the Jetson embedded computer and listed in Table 5. It is clear that within the scope of this experiment that RTAB-Map can outperform ORB SLAM in numerous scenarios.

From the experiments, it can be observed that in cases with good features like in the case with the desk sequences, the system is able to keep track and maintain localization in both scenarios. While in the case of pioneer robot instances, the system is presented with featureless scenarios or a scenario with an abrupt change in illumination, the system tends to lose the localization. This explains the abrupt drop in localization accuracy in the last two sequences. But it was observed that RTAB-Map tends to achieve localization even after losing track in these scenarios, and this behaviour was less frequently observed in the case of ORB SLAM 2.

Experiments with Handheld Mapping In the next set of experiments, the system was run exclusively on the Jetson computer. This was because the ZED SDK was written to work only on NVIDIA GPUs and any workaround that was attempted on the host computer would essentially not be using the advanced on-camera depth processing capabilities of the system.

As discussed in Section 4.2.2, the first experiment involves scanning a box and then comparing the measured dimension against the actual one. Two instances of this experiment are performed to evaluate the performance of the depth measurement at different depth ranges.

- The first experiment maps the geometry of Box1 at an approximate distance of 1m from the box.
- In the second experiment, the object (Box2) is placed at a distance of 3m from the camera.

The RTAB-Map pipeline is invoked and the mapping process is carried out. The database is then post-processed to downsample the points and the measurement is exported as a

	Length (m)		Breadth (m)		Height (m)		Error
	Measured	Original	Measured	Original	Measured	Original	%
Box1	0.35	0.34	0.20	0.19	0.45	0.46	2.84
Box2	0.33	0.27	0.22	0.23	0.18	0.16	11.27

Table 6: Comparison of results from the handheld experiments

pointcloud(.ply) file. This is then loaded onto the MeshLab project and the measurement tool is used to measure the edge lengths

The results after performing the mapping experiments are collected in Table 6. As an additional step, the pointcloud is also visually compared against a synthetic 3D mesh of the original box. This is generated within MeshLab itself and aligned with the pointcloud. The visual results of this experiment are shown in fig. 35.

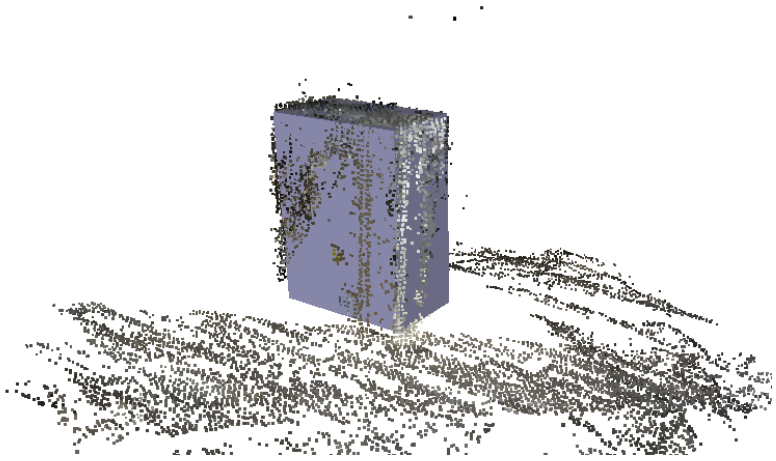


Figure 35: The downsampled points overlayed with the synthetic GT

The next set of experiments includes mapping a closed room and comparing the accuracy of the measurements against the actual dimensions. The GT here is the measurement of the room lengths performed using a Bosch PLR50, an industrial-grade laser rangefinder. The measurements are made and the results are tabulated in table 7. A map is also constructed and is shown in Figure 36.

Simulation Results The simulation world and the modified robot model is loaded in Gazebo and the robot is driven in the environment as the robot builds a map of the

	Length (m)		Breadth (m)		Height (m)		Error
	Measured	Original	Measured	Original	Measured	Original	%
Room1	12.11	12.01	4.31	4.68	2.35	2.45	4.55
Corridoor1	18.43	19.74	1.11	1.58	2.3	2.45	6.64

Table 7: Comparison of results from the area mapping experiments

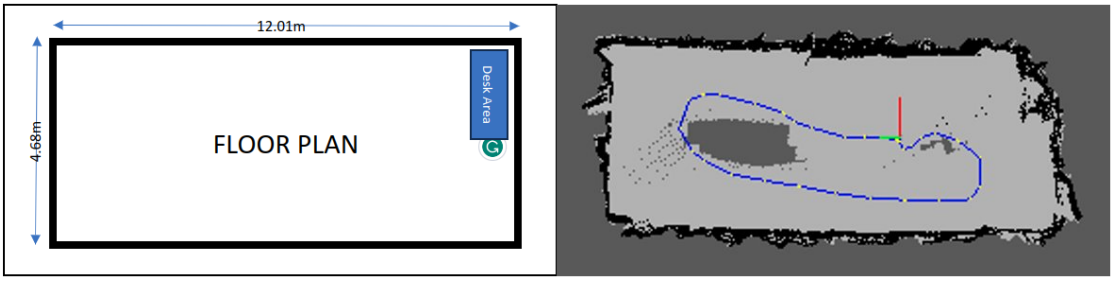


Figure 36: Map generated after mapping Room1

environment. This map is then saved and visually inspected for inconsistencies with the original world.

The experiment is carried out on two fronts.

- The first experiment is performed with accurate odometry by using a LiDAR input to the system.
- The next one is performed with visual odometry only.

As it is known, the LiDAR-odometry is a very accurate form of odometry measurement, and the results shown in fig. 37a is corroborating this prior information. Whereas the map created with visual odometry has clear inconsistencies, as seen in fig. 37b.

(a) LiDAR Odometry (b) Visual Odometry

Figure 37: The generated occupancy grid map in the mapping experiment

5 Conclusion

5.1 Discussions

From the results of the experiments, it can be inferred that when in the context of an indoor mapping solution as proposed by the scope of this research work, RTAB-Map can be considered as a promising algorithm to perform the task with the required accuracy and reliability.

The first set of experiments with the benchmark datasets proves how the system has very good localization capabilities. The best-case localization error is 0.0345, which is in the fr2/long_office_household sequence and this is well inside the limit that we have defined within the scope of this work. The translational error is 0.0292m (RMSE) and the rotational error is 1.09 degree. This indicates that the system has a fairly accurate localization.

The second set of experiments shows the structural accuracy of the generated 3D maps. The measured boxes show a linear dimensional error below 15% and the average error of all the 3 dimensions falls below 10%. This indicates that the structure is also preserved in a fairly accurate manner during mapping and this error is also within the limits set by the scope.

Finally, the prototype puts this together in the form of a boxed solution, which can now be adapted onto standard robots running on ROS with minimal changes in terms of sensor topics and transformations. It is to be kept in mind that the prototype designed here in simulation is a very ideal case as discussed in the previous section. Even though the simulation offers near-real conditions for the camera input, there are certain aspects of the real world that the simulation cannot factor in. Reflections and inconsistent lighting are two such scenarios. Reflective surfaces show up as frontiers in the 3D map (see fig. 38 in the case of a mirror) because the visual sensory input perceives it as another set of image-depth pair and the system cannot differentiate this as a reflection and a real scene. Another glaring problem that can be observed while working with real-world scenarios is the lack of consistent illumination. Light does not shine on an object/surface uniformly and therefore features that are present in one lighting condition cannot be expected to be detected in a subsequent frame of the same data stream. This again causes a mismatch between the correlated features and results in geometrically regular structures showing up as non-conforming irregular artifacts in the scan.

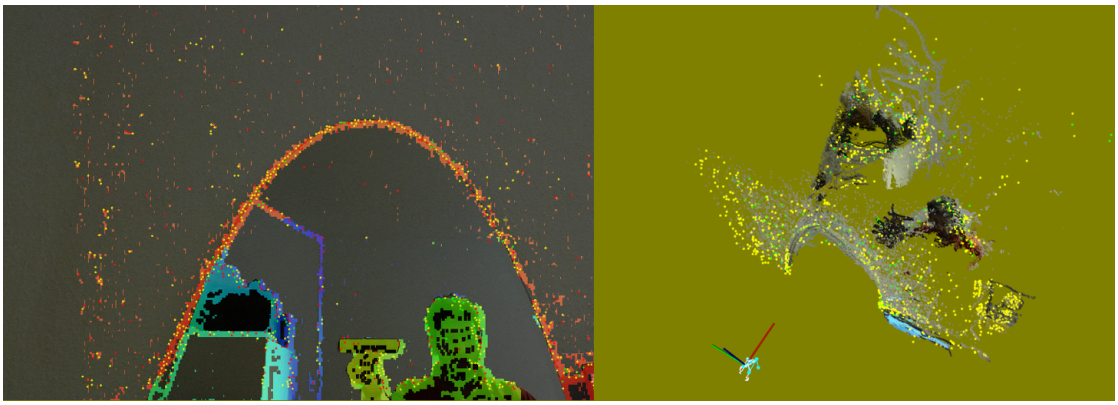


Figure 38: An instance of the reflection being assigned false depth

Issues like these are very common in VSLAM systems as they primarily employ vision-based depth perception and therefore in the event of the absence of features, the system becomes incapable of calculating depth. Many measures are taken in modern VSLAM systems, both in the sensory front (e.g. structured light patterns, on-chip AI) and in the algorithmic side (e.g. sensor fusion, semantic maps) to counter such issues. This work does not go into such details but rather helps the user in setting up a minimal working solution that can reliably scan an environment and provide the user with a geometrically consistent map of the environment.

5.2 Scope of Expansion

As a clear indication of improvement from the mapping experiment of the simulation world, it is evident that accurate odometry can provide very good results. So the first and foremost point for the expansion of this system should be by incorporating an accurate odometry scheme. Accurate odometry as we know aids efficient map building and therefore one can expect the mapping quality to scale up considerably from the state of this prototype.

Another front where this work can be further expanded is by incorporating semantic information into the sensory information. Like in the case of incorrect modeling of reflections in the mapping result, if there were a semantic mask describing the presence of a reflection in the scene, the generated artifacts can be classified as outliers and therefore the mapping quality can be enhanced.

Also to address the problem of irregular surfaces being generated because of the presence of inconsistent illumination in the scene, a hardware-assisted depth camera, like the Kinect and RealSense, projects structured light patterns onto the surface to enhance

depth accuracy. This makes the depth calculation very accurate and may prove as a viable solution for the illumination variation.

5.3 Summary

In this research thesis, originally intended to serve as a guidebook to setting up a mapping solution for an indoor mapping scenario, one can find sufficient resources to gather knowledge in the field of 3D measurement technologies and mapping techniques. Chapter 1 covers an introduction to the work, by providing a background of the problem statement and the motivation to work on the problem. This chapter also gives the reader a first impression of what can be expected from this work and what all are beyond the scope of this research work.

Chapter 2 discusses the background required for the user to dive into the SLAM concepts that are further visited in detail in Chapter 3. This chapter introduces and explains 4 major (or rather conventional) SLAM methods - LSD SLAM, ORB SLAM 2, DTAM, and RTAB-Map, which were considered for this particular application owing to the various features that they offered. A detailed expansion is done for RTAB-Map as this method offers very interesting characteristics like memory management for long-term localization and subsequently was a clear choice for the experiment. From the various works that compare different SLAM algorithms and their performances, it was also noted that input preprocessing showed a positive impact on the localization accuracy of the system. Parallelising some threads on the GPU also showed positive impact on the system performance.

In Chapter 4 the different experiments are devised in order to evaluate the positional and structural accuracy of the mapping system. The datasets provide a good estimate of the tracking capability localization accuracy of the system whereas the handheld mapping experiments estimate the structural accuracy and the quality of the mapped geometries. From the handheld experiments, numerous takeaways have been made in terms of setting up the mapping system as this experiment is the closest one to the real world, and therefore many issues that were not confronted in the experiments with the datasets were faced here.

The observations and inferences from these experiments provide sufficient data to support the argument of RTAB-Map being a good candidate for an indoor mapping application that can be set up and deployed in a small-medium scale environment with minimum effort. When it comes to more complicated layouts, this method may prove to be insufficient and additional hardware as explained in Section 5.2 may be chosen to be adapted into the system, thereby enhancing it's performance. With the learnings from these experiments and the possible scope of expanding this work consolidated in Chapter 5, the research project thesis is concluded.

List of Figures

List of Tables

Bibliography

[BBS08] Wolfram Burgard, Oliver Brock, and Cyrill Stachniss. "A Tree Parameterization for Efficiently Computing Maximum Likelihood Maps using Gradient Descent." In: *Robotics: Science and Systems III*. 2008, pp. 65–72.

[BTV06] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "SURF: Speeded up robust features." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 3951 LNCS (2006), pp. 404–417. ISSN: 03029743. DOI: 10.1007/11744023{_}32. URL: https://www.researchgate.net/publication/225761164_SURF_Speeded_up_robust_features.

[Cal+10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. "BRIEF: Binary robust independent elementary features." In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6314 LNCS.PART 4 (2010), pp. 778–792. ISSN: 16113349. DOI: 10.1007/978-3-642-15561-1{_}56/COVER. URL: https://link.springer.com/chapter/10.1007/978-3-642-15561-1_56.

[Cor17] Peter Corke. "Robotics, Vision and Control." In: Springer Tracts in Advanced Robotics 118 (2017). DOI: 10.1007/978-3-319-54413-7. URL: http://link.springer.com/10.1007/978-3-319-54413-7.

[DC22] Frank Dellaert and GTSAM Contributors. *borglab/gtsam*. Version 4.2a8. May 2022. DOI: 10.5281/zenodo.5794541. URL: https://github.com/borglab/gtsam).

[De +18] K. T. D. S. De Silva, B. P. A. Cooray, J. I. Chinthaka, P. P. Kumara, and S. J. Sooriyaarachchi. "Comparative Analysis of Octomap and RTABMap for Multi-robot Disaster Site Mapping." In: *2018 18th International Conference on Advances in ICT for Emerging Regions (ICTer)*. IEEE, Sept. 2018, pp. 433–438. ISBN: 978-1-5386-7352-2. DOI: 10.1109/ICTER.2018.8615469. URL: https://ieeexplore.ieee.org/document/8615469/.

[DXG19] Bruno M F Da Silva, Rodrigo S Xavier, and Luiz M G Gonçalves. "Mapping and Navigation for Indoor Robots under ROS: An Experimental Analysis." In: (2019). DOI: 10.20944/preprints201907.0035.v1. URL: www.preprints.org.

[End+14] Felix Endres, Jürgen Hess, Jürgen Sturm, Daniel Cremers, and Wolfram Burgard. "3-D Mapping with an RGB-D camera." In: *IEEE Transactions on Robotics* 30.1 (2014), pp. 177–187. ISSN: 15523098. DOI: 10.1109/TRO.2013.2279412. URL: https://www.researchgate.net/publication/260520054_3-D_mapping_with_an_RGB-D_camera.

[ESC14] Jakob Engel, Thomas Schöps, and Daniel Cremers. *LSD-SLAM: Large-Scale Direct Monocular SLAM*. Tech. rep. 2014.

[FA18] Maksim Filipenko and Ilya Afanasyev. "Comparison of Various SLAM Systems for Mobile Robot in an Indoor Environment." In: *9th International Conference on Intelligent Systems 2018: Theory, Research and Innovation in Applications, IS 2018 - Proceedings* (July 2018), pp. 400–407. DOI: 10.1109/IS.2018.8710464.

[FH21] Abdel-Rahman Fowler and Zakaria Hamimi. "Turin Papyrus Map and Historical Background of Geological Work in Egypt." In: 2021, pp. 1–13. DOI: 10.1007/978-3-030-49771-2{_}1. URL: http://link.springer.com/10.1007/978-3-030-49771-2_1.

[For+17] Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems." In: *IEEE Transactions on Robotics* 33.2 (Apr. 2017), pp. 249–265. ISSN: 15523098. DOI: 10.1109/TRO.2016.2623335.

[FPS14] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. "SVO: Fast semi-direct monocular visual odometry." In: *Proceedings - IEEE International Conference on Robotics and Automation* (Sept. 2014), pp. 15–22. ISSN: 10504729. DOI: 10.1109/ICRA.2014.6906584.

[FS12] Friedrich Fraundorfer and Davide Scaramuzza. "Visual Odometry : Part II: Matching, Robustness, Optimization, and Applications." In: *IEEE Robotics & Automation Magazine* 19.2 (June 2012), pp. 78–90. ISSN: 1070-9932. DOI: 10.1109/MRA.2012.2182810. URL: https://ieeexplore.ieee.org/document/6153423/.

[Giu+19] Riccardo Giubilato, Sebastiano Chiodini, Marco Pertile, and Stefano Debei. "An evaluation of ROS-compatible stereo visual SLAM methods on a nVidia Jetson TX2." In: *Measurement* 140 (July 2019), pp. 161–170. ISSN: 02632241. DOI: 10.1016/j.measurement.2019.03.038. URL: https://linkinghub.elsevier.com/retrieve/pii/S0263224119302489.

[GSB07] G. Grisetti, C. Stachniss, and W. Burgard. "Improved Techniques for Grid Mapping with Rao- Blackwellized Particle Filters." In: *IEEE Transactions on Robotics* 23.1 (2007), pp. 34–46. URL: http://ais.informatik.uni-freiburg.de/publications/papers/grisetti07tro.pdf.

[Hor+13] Armin Hornung, Kai M. Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. "OctoMap: An efficient probabilistic 3D mapping framework based on octrees." In: *Autonomous Robots* 34.3 (Apr. 2013), pp. 189–206. ISSN: 09295593. DOI: 10.1007/S10514-012-9321-0.

[Hor87] Berthold K. P. Horn. "Closed-form solution of absolute orientation using unit quaternions." In: *Journal of the Optical Society of America A* 4.4 (Apr. 1987), p. 629. ISSN: 1084-7529. DOI: 10.1364/JOSAA.4.000629.

[HS88] C. Harris and M. Stephens. "A Combined Corner and Edge Detector." In: *Procedings of the Alvey Vision Conference 1988*. Alvey Vision Club, Apr. 1988, pp. 1–23. DOI: 10.5244/C.2.23. URL: http://www.bmva.org/bmvc/1988/avc-88-023.html.

[Ila+17] Viorela Ila, Lukas Polok, Marek Solony, and Pavel Svoboda. "SLAM++1-A highly efficient and temporally scalable incremental SLAM framework." In: *International Journal of Robotics Research* 36.2 (Feb. 2017), pp. 210–230. ISSN: 17413176. DOI: 10.1177/0278364917691110.

[K94] Zuiderveld K. "Contrast Limited Adaptive Histogram Equalization." In: *Graphics Gems* 0 (1994), pp. 474–485. URL: https://cir.nii.ac.jp/crid/1570009751230513024.

[Kle14] Reinhard Klette. *Concise Computer Vision*. 2014. ISBN: 978-1-4471-6319-0. DOI: 10.1007/978-1-4471-6320-6/COVER. URL: http://link.springer.com/10.1007/978-1-4471-6320-6.

[Kud] Kudan. *Understanding how Direct Visual SLAM works*. URL: https://www.kudan.io/blog/direct-visual-slam/.

[Küm+11] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. "g2o: A General Framework for Graph Optimization." In: *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*. Shanghai, China, May 2011.

[LCS11] Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. "BRISK: Binary Robust invariant scalable keypoints." In: *Proceedings of the IEEE International Conference on Computer Vision* (2011), pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.

[LM13] Mathieu Labbé and François Michaud. "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation." In: (2013). DOI: 10.1109/TRO.2013.2242375. URL: http://rtabmap.googlecode.com..

[LM19] Mathieu Labbé and François Michaud. "RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation." In: *Journal of Field Robotics* 36.2 (Mar. 2019), pp. 416–446. ISSN: 15564967. DOI: 10.1002/ROB.21831.

[Low04] David G Lowe. "Distinctive Image Features from Scale-Invariant Keypoints." In: *International Journal of Computer Vision* (2004). URL: https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf.

[Low99] David G. Lowe. *Object recognition from local scale-invariant features*. Vol. 2. IEEE, 1999, pp. 1150–1157. DOI: 10.1109/ICCV.1999.790410. URL: http://www.cs.ubc.ca/~lowe/papers/iccv99.pdf.

[Mac+22] Andréa Macario Barros, Maugan Michel, Yoann Moline, Gwenolé Corre, and Frédérick Carrel. "A Comprehensive Survey of Visual SLAM Algorithms." In: *Robotics 2022, Vol. 11, Page 24* 11.1 (Feb. 2022), p. 24. ISSN: 2218-6581. DOI: 10.3390/ROBOTICS11010024. URL: https://www.mdpi.com/2218-6581/11/1/24/htm%20https://www.mdpi.com/2218-6581/11/1/24.

[Mon+02] Michael Montemerlo, Sebastian Thrun, Daphne Koller, and Ben Wegbreit. "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem." In: *Eighteenth National Conference on Artificial Intelligence*. USA: American Association for Artificial Intelligence, 2002, pp. 593–598. ISBN: 0262511290.

[MT17] Raul Mur-Artal and Juan D. Tardos. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras." In: *IEEE Transactions on Robotics* 33.5 (Oct. 2017), pp. 1255–1262. ISSN: 1552-3098. DOI: 10.1109/TRO.2017.2705103. URL: http://ieeexplore.ieee.org/document/7946260/.

[New+11] Richard A Newcombe, Shahram Izadi, Otmar Hilliges, David Molyneaux, David Kim, Andrew Davison, Pushmeet Kohli, Jamie Shotton, Steve Hodges, and Andrew Fitzgibbon. "KinectFusion: Real-Time Dense Surface Mapping and Tracking." 2011. URL: https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/ismar2011.pdf.

[NLD11] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. "DTAM: Dense tracking and mapping in real-time." In: *Proceedings of the IEEE International Conference on Computer Vision* (2011), pp. 2320–2327. DOI: 10.1109/ICCV.2011.6126513.

[Pir+17] Taihú Pire, Thomas Fischer, Gastón Castro, Pablo De Cristóforis, Javier Civera, and Julio Jacobo Berlles. "S-PTAM: Stereo Parallel Tracking and Mapping." In: (2017).

[Rap] Rapyuta Robotics. *Rapyuta Robotics Wiki*. URL: https://github.com/rapyuta-robotics/rapyuta-mapping/wiki.

[Rub+11] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. "ORB: An efficient alternative to SIFT or SURF." In: *Proceedings of the IEEE International Conference on Computer Vision* (2011), pp. 2564–2571. DOI: 10.1109/ICCV.2011.6126544.

[SC19] Patrik Schmuck and Margarita Chli. "CCM-SLAM: Robust and efficient centralized collaborative monocular simultaneous localization and mapping for robotic teams." In: *Journal of Field Robotics* 36.4 (June 2019), pp. 763–781. ISSN: 1556-4967. DOI: 10.1002/ROB.21854. URL: https://onlinelibrary.wiley.com/doi/10.1002/rob.21854.

[SK16] Bruno Siciliano and Oussama Khatib. *Springer Handbook of Robotics*. Springer Handbooks. Cham: Springer International Publishing, Jan. 2016, pp. 1–2227. ISBN: 978-3-319-32550-7. DOI: 10.1007/978-3-319-32552-1. URL: https://link.springer.com/10.1007/978-3-319-32552-1.

[SSC88] R Smith, M Self, and P Cheeseman. "A stochastic map for uncertain spatial relationships." In: *Proceedings of the 4th international symposium on Robotics Research* 0262022729 (1988), pp. 467–474. URL: http://portal.acm.org/citation.cfm?id=57472.

[ST94] Jianbo Shi and Carlo Tomasi. "Good Features to Track." In: *IEEE Conference on Computer Vision and Pattern Recognition*. Seattle, 1994. URL: http://www.ai.mit.edu/courses/6.891/handouts/shi94good.pdf.

[Stu+12] Jürgen Sturm, Nikolas Engelhard, Felix Endres, Wolfram Burgard, and Daniel Cremers. "A benchmark for the evaluation of RGB-D SLAM systems." In: *IEEE International Conference on Intelligent Robots and Systems*. 2012, pp. 573–580. ISBN: 9781467317375. DOI: 10.1109/IROS.2012.6385773.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. Cambridge, Mass.: MIT Press, 2005. ISBN: 9780262201629. URL: https://mitpress.mit.edu/9780262201629/probabilistic-robotics/.

[Vis11] D. Viswanathan. "Features from Accelerated Segment Test (FAST)." In: (2011).

[WFM19] Ami Woo, Baris Fidan, and William W. Melek. "Localization for Autonomous Driving." In: *Handbook of Position Location*. Wiley, Jan. 2019, pp. 1051–1087. DOI: 10.1002/9781119434610.ch29. URL: https://onlinelibrary.wiley.com/doi/10.1002/9781119434610.ch29.

Declaration of Authorship

I declare that the work presented here is original and the result of my own investigations. Formulations and ideas taken from other sources are cited as such.

It has not been submitted, either in part or whole, for a degree at this or any other university.

Dortmund, August 27, 2023 Aaron Xavier