Article

Mapping and Navigation for Indoor Robots under ROS: An Experimental Analysis

Bruno M. F. da Silva [1,2], **Rodrigo S. Xavier** [1,2] **and Luiz M. G. Gonçalves** [1,3]*

1 Natalnet Associate Laboratories, Federal University of Rio Grande do Norte, Campus Universitário, 59078-970, Natal, Brazil;
2 School of Sciences and Technology, Federal University of Rio Grande do Norte, Campus Universitário, 59078-970, Natal, Brazil; bruno.silva@ect.ufrn.br, rodrigosarmentox@gmail.com
3 Department of Computer Engineering and Automation, Federal University of Rio Grande do Norte, Campus Universitário, 59078-970, Natal, Brazil lmarcos@dca.ufrn.br
* Correspondence: lmarcos@dca.ufrn.br; Tel.: +55-84-3215-3771

Abstract: Since it was proposed, the Robot Operating System (ROS) has fostered solutions for various problems in robotics in the form of ROS packages. One of these problems is Simultaneous Localization and Mapping (SLAM), a problem solved by computing the robot pose and a map of its environment of operation at the same time. The increasingly availability of robot kits ready to be programmed and also of RGB-D sensors often pose the question of which SLAM package should be used given the application requirements. When the SLAM subsystem must deliver estimates for robot navigation, as is the case of applications involving autonomous navigation, this question is even more relevant. This work presents an experimental analysis of GMapping and RTAB-Map, two ROS compatible SLAM packages, regarding their SLAM accuracy, quality of produced maps and use of produced maps in navigation tasks. Our analysis aims ground robots equipped with RGB-D sensors for indoor environments and is supported by experiments conducted on datasets from simulation, benchmarks and from our own robot.

Keywords: ROS; SLAM; RGB-D sensors; experimental analysis

1. Introduction

With technological advances in sensing devices and computing the problem known as Simultaneous Localization and Mapping (SLAM) [1–4] has been one of the most worked problems by the Robotics research community in the last decade, which has produced several contributions [5–9]. To solve the SLAM problem, a map of the surroundings of a robot should be incrementally constructed at the same time that the robot pose (localization and orientation) is continuously estimated with respect to this map. Since the navigation of the robot in an environment involves knowing its own location and those of its goals, determining the best solutions for SLAM is considered an important step towards enabling mobile robots to operate without human intervention.

The interest for SLAM has increased due in large part to the Robot Operating System (ROS) [10], which has been proposed to facilitate the development of software for robotics and has become the *de facto* standard robotics middleware. Facilities like offering inter process/inter machine message passing mechanisms and hardware abstraction for handling data from robot sensors and controllers explains why ROS has been widely adopted in robot research and industry [11]. ROS is open source as well, which makes its development reach more visibility.

Nevertheless, estimates for the pose of a robot and the mapping of its environment of operation, alone, are not sufficient for allowing the robot navigation. Besides offering SLAM estimates with the lowest possible errors, a robot must plan a trajectory from its location to goal locations using a suitable map representation, such as occupancy grid [1] and then execute the necessary commands to traverse the planned route.

The broad adoption of ROS by the community has contributed substantially to the development of novel approaches on both SLAM [5–9,12–17] and robot navigation [18]. The solutions are available as ROS packages, a high level of abstraction software ready to be used on any ROS compatible robot.

It is often the case that roboticists and mainly non roboticists are posed with the choice of which SLAM algorithm should be employed on a given application in view of their requirements. Choices like this have become common because of the increasingly availability of programmable robot kits like the Turtlebot [1] and RGB-D cameras, which are a class of cameras that has enough accuracy for indoor, small scale applications and include, for example, the Microsoft Kinect, Intel RealSense, ZED stereo camera [19] and the recently launched SVPRO 3D VR. Methods for rapidly and easy calibrating and using these cameras have also become available [20].

Considering the vast availability of ROS packages for SLAM, our work aims to to provide a preliminary guidance in order to assist beginners and experienced roboticists on the selection of an adequate ROS compatible SLAM package. We aim to support the development of applications for indoor environments using ground, mobile robots equipped with a RGB-D camera solution. For this, we experimentally analyze the GMapping [12] and RTAB-Map [9] ROS packages taking into consideration the SLAM accuracy of the algorithms, the quality of the grid maps produced as outputs and how well these grid maps are used in navigation tasks. In our analysis, we focus on practical aspects of the algorithms and relate their performance with their inner workings and configurable parameters. To support our analysis, we conduct experiments on datasets from simulation, SLAM benchmarks [21] and on real robot sensory data collected at the facilities of our university. So, the main contribution of this work can be understood as being twofold. The first one is to analyze the main ROS packages that deal with simultaneous localization and mapping and the second contribution relies on the production pf our own dataset that can be used for this kind of evaluation, for existing or further algorithms.

The remaining of this paper is organized as follows. Section 2 lists works related to SLAM and ROS implementation of SLAM algorithms, along with other experimental evaluation of ROS SLAM packages. Section 3 briefly describes the analyzed SLAM algorithms, whereas Section 4 provides the details on how the evaluation was carried out. Section 5 explains our analysis in light of the results of the performed experiments. Finally, we conclude the work with Section 6.

2. Related Works

Since its proposition, 10 years ago, the wide adoption of ROS has boosted research and development of robotics applications. Researches related to ROS include, to name a few, using robots as a service [22–24], easing development of applications using robots [25], leveraging robotics for education [26], developing robots for telepresence [27] and enabling knowledge transfer between robots [28].

Variations of SLAM are used in the above applications and it is considered nowadays, actually, as a well-studied problem. Its theoretical background, foundations and sensor agnostic formulations are available in tutorial papers [2,3,29] and text books [1,4]. In general, SLAM algorithms must be able to associate sensor data gathered from distinct time steps and also current sensor data with current map data. This last ability is known in the literature as *loop closing* [2,3], and is of uttermost importance for any robotic platform to operate.

Algorithms relying on sensor modalities such as lasers [30], monocular cameras [31,32], stereo cameras [33,34] and RGB-D cameras [35] have been designed and validated for indoor, outdoor,

[1] https://www.turtlebot.com/

structured and non-structured scenarios. Nonetheless, SLAM still has various open issues [36], mainly because a general purpose solution that works with all sensors and all types of environments is not available, yet.

As a consequence, a considerable number of ROS compatible SLAM systems is present in the robotics literature [5–9,12–17]. Each of these solutions is designed based on its own mathematical formulation that allows it to work in a specific combination of application requirements, such as number and type of sensors, degrees of freedom (DoF) of the robot movement and characteristics of the operation environment. In line with this and according to the used sensor type, we can divide ROS SLAM systems in two main categories: lidar and vision based.

Among the lidar based ROS systems, we can cite GMapping [12], tinySLAM [13], Karto SLAM [5], Lago SLAM [6], Hector SLAM [14] and Google Cartographer [8]. In practice, lidar based SLAM algorithms work by fusing laser scans with robot odometry [5,6,8,12,13] or with IMU [14]. The underlying formulation of these approaches assumes a planar lidar mounted on a robot moving on a 2D plane, which limits the estimated robot poses to 3 DoF (2D position and an orientation angle). Albeit, an algorithm as the Hector SLAM is able to estimate 6 DoF poses (3D position and three rotation angles around the X,Y and Z axis) if an IMU is available. GMapping and tinySLAM solve SLAM calculating the probability density function of the robot pose and perceived map given laser scans and robot odometry. The computation is implemented with a particle filter [1], which maintains a set of particles that are representations of robot pose and environment map. These particles are continuously sampled during the process. Lago SLAM, Karto SLAM and Google Cartographer estimate the same probability density function, but solves an optimization problem on graphs in which the vertices are robot poses and the edges are rigid transformations relating pairs of robot poses [29]. The Hector SLAM follows an alternative approach which optimizes the robot pose that best fits the laser scan endpoints within the map using gradient descent for error minimization.

RGB-D SLAM [7], LSD SLAM [15], ORB SLAM [16], ORB SLAM 2 [17] and RTAB-Map [9] can be cited as relevant works on vision based ROS systems. Systems from this class are based on a real-time 3D reconstruction framework known as visual odometry [37], which in turn was originated from the projective geometry theory [38]. To compute 6 DoF robot poses and environment maps, the algorithms use images from a single camera [15–17], stereo camera [9,17] or from RGB-D camera [7,9,17]. Camera poses are then estimated from image keypoints such as SIFT [39], SURF [40] or ORB [41]. As more recently proposed, they can also be estimated directly from image intensity [15,42]. Together with loop closing (detecting when a previously mapped location is being revisited) and the optimization back-end [9,16,17,43], visual odometry is an important component in vision based SLAM. Hence it is worth mentioning that there are also visual odometry systems compatible with ROS [42,44,45].

A clear advantage of lidar based approaches is the fact that ROS packages of this class produce a 2D representation of the environment known as occupancy grid [1], which in general is not available from the visual SLAM counterpart. Occupancy grid is essential for robot navigation, since path planning algorithms use it as one of the input data along with the starting and final positions. Because of the prohibitive costs of lidars, especially for low cost applications, one solution is to emulate a laser scanner with an RGB-D sensor. GMapping, Hector SLAM and RTAB-Map are examples of ROS systems that support an RGB-D sensor emulating a laser scanner, although with varied rates of success [46].

Empirical comparisons of ROS SLAM systems have been proposed in the robotics literature [46–49]. In general, these works aim to provide guidance for roboticists that seek for the best SLAM solution according to different application requirements. For example, Santos et al. [47] evaluate lidar based SLAM algorithms for robot search and rescue competition. Monocular visual SLAM solutions are evaluated in the work of Buyval et al. [48] whereas the work of Ibragimov and Afanasyev [49] experiments with monocular, stereo and RGB-D algorithms.

In our previous work [46], we experimentally evaluatetd some ROS SLAM algorithms that can be used with RGB-D sensors in indoor scenarios. Here we extend this work in the following ways.

Firstly, we focus our analysis on the best two algorithms: GMapping, a lidar based solution that can be employed with a RGB-D sensor and the RTAB-Map, which is a visual solution supporting various configurations. Secondly, the SLAM accuracy of the selected systems is assessed on a commonly used benchmark [21]. Thirdly, the two solutions are evaluated on both simulation datasets, from Gazebo [50] and on our own real data. In comparison to the data used in our previous work, the operation environment of the experiments in this work is larger in size, distance travelled and has different data capturing conditions. Lastly, since the two selected algorithms produce occupancy grids, we evaluate the resulting grids and whether the quality of produced grids are suited for robot navigation.

3. Robot Mapping with ROS

ROS [10,51] is a middleware for robotics that offers hardware abstraction and inter-process communication mechanisms. Modules from robot software which are commonly used in different applications can be reused thanks to its underlying architecture.

More specifically, each running process in the ROS is called a node. Each node is able to communicate with another node by a publish/subscribe message passing scheme: nodes that subscribe on a topic receive the corresponding message whenever any node on the system publishes a message on that topic. This mechanism allows the design of robot software composed of independent, asynchronous and distributed programs that talk to each other, as is the case of software involved in autonomous mapping for robotics. Also, ROS keeps all transformations between the reference frames involved in the distributed system in a tree structure called transformation tree (tf tree).

For an algorithm to compute a solution for SLAM, it must estimate the robot pose $\mathbf{x}_t$, parameterized by a 2D or 3D position and one or three rotation angles, according to the DoF of the robot movement. It must also estimate a map $\mathbf{m}_t$, which is generally given as an occupancy grid but may also be represented by a 2D or 3D point cloud. This is performed by processing sensor measurements $\mathbf{z}_t$ and odometry readings $\mathbf{u}_t$, with all of these quantities indexed by a time step t.

In a ROS system solving the SLAM problem, a node computes the current robot pose and the map by subscribing to topics related to the required sensor measurements $\mathbf{z}_t$ and odometry readings $\mathbf{u}_t$. Each time a new sensor measurement or odometry reading is published by the respective sensor nodes, the subscribed node gets notified and thus it receives a message consisting of sensor data or odometry displacement. These messages are in turn processed by the algorithm, producing the most recent estimate for the robot localization $\mathbf{x}_t$ and map $\mathbf{m}_t$ of its environment.

The ROS subsystem responsible for robot navigation [18] takes as input the map produced by SLAM, the robot location and a goal location. Then, a planned trajectory is forwarded to the ROS nodes that are responsible for the robot controllers, which then drive the robot towards its goal location. For this to be possible, the map must be a in a suitable representation for ground robot navigation called occupancy grid [1]. An occupancy grid is a planar 2D grid formed by a set of square cells. Each cell of the grid has a probability value informing if the cell is navigable or not by a robot. Additionally, reactive nodes perceive the environment using data from the available sensors in order to detect dynamic objects that are not present in the grid maps. In doing so, routes can be re-planned whenever is the case that dynamic objects appear in front of the robot while it is moving to its goal location.

3.1. GMapping

GMapping [12] is a planar (2D) SLAM algorithm that relies on the robot odometry and on measurements that come from range sensors (e.g. sonars and lasers) to estimate both the robot pose and the map in which the robot is operating, which is represented as an occupancy grid.

Following the well known Bayesian approach [1] to fuse pose transitions and sensor measurements in a probabilistic framework, the algorithm solves SLAM by utilizing a Rao-Blackwellized Particle Filter. Equation 1 shows the sought posterior probability $p(\mathbf{x}_t, \mathbf{m}_t | \mathbf{z}_t, \mathbf{u}_t)$ after being factorized by a process called Rao-Blackwellization [1,12].

$$p(\mathbf{x}_t, \mathbf{m}_t | \mathbf{z}_t, \mathbf{u}_t) = p(\mathbf{x}_t | \mathbf{z}_t, \mathbf{u}_t) p(\mathbf{m}_t | \mathbf{x}_t, \mathbf{z}_t) \quad (1)$$

In Equation 1, the posterior probability of a robot pose $\mathbf{x}_t$ and map $\mathbf{m}_t$ given sensor measurements $\mathbf{z}_t$ and odometry $\mathbf{u}_t$ is shown as a product of two other probabilities: the probability $p(\mathbf{x}_t|\mathbf{z}_t, \mathbf{u}_t)$ of having pose $\mathbf{x}_t$ given measurements $\mathbf{z}_t$ and odometry $\mathbf{u}_t$ times the probability $p(\mathbf{m}_t|\mathbf{x}_t, \mathbf{z}_t)$ of having a map $\mathbf{m}_t$ given robot pose $\mathbf{x}_t$ and measurements $\mathbf{z}_t$.

In a particle filter, hypothesis for the robot pose and the map of its environment are kept in a set of particles. Each particle $< \mathbf{x}^i, \mathbf{m}^i, \mathbf{w}^i >$ of the set $\{< \mathbf{x}^1, \mathbf{m}^1, \mathbf{w}^1 >, ..., < \mathbf{x}^N, \mathbf{m}^N, \mathbf{w}^N >\}$ has a history of past robot poses $\mathbf{x}^i$, a map computed from this history $\mathbf{m}^i$ and a weight $\mathbf{w}^i$. The particle weight is a probability value that informs how likely the particle represents the pose history and map given odometry and sensor measurements.

The challenge is then how to sample each particle $< \mathbf{x}^i, \mathbf{m}^i, \mathbf{w}^i >$ in a way that truly represents the robot belief about its pose and computed map. For this, GMapping relies on robot odometry and scan matching (registration between consecutive laser scans) to estimate the parameters of a probability density function from which the particle set will be generated in each iteration of the filter. In doing so, the number of particles N needed for accurate SLAM estimation is drastically reduced, which in turn also reduces the computational cost involved in the process. The particle depletion problem [12], a situation inherent to particle filters in which particles with a high weight can be discarded, is avoided by the algorithm. To comply with this, particles are resampled only if an estimated number of effective particles N_{eff} is below a prespecified threshold.

3.1.1. GMapping ROS Implementation

The GMapping ROS implementation is composed of a single ROS node called `slam_gmapping`. This node subscribes to a `scan` topic with sensor data and `tf` topic with odometry data. Whenever the node receives a message from these topics, it publishes messages in the topics `map` (which publishes the updated occupancy grid) and updates the transformation from the `map` reference frame to the `odom` (odometry) reference frame in the robot `tf_tree`. Table 1 lists the relevant topics published or subscribed by the `slam_gmapping` ROS node.

Table 1. List of relevant published/subscribed topics for the GMapping ROS package.

Topic	Message	Type
tf	`tf/tfMessage`	Subscribed
scan	`sensor_msgs/LaserScan`	Subscribed
map	`nav_msgs/OccupancyGrid`	Published

ROS allows online parameter tuning of the algorithms. For GMapping, a not exhaustive list from the available parameters[2] is given on Table 2, along with the values used in our experiments.

[2] http://wiki.ros.org/gmapping

Table 2. List of relevant parameters for the GMapping ROS package.

Parameter	Value	Description
`particles`	30	Number of particles in the filter
`linearUpdate`	1.0	Travelled distance that triggers a scan processing
`angularUpdate`	0.5	Rotated angle that triggers a scan processing
`resampleThreshold`	0.5	N_{eff} resampling threshold
`delta`	0.05	Occupancy grid resolution (m)
`occ_thresh`	0.25	Threshold to consider a cell as occupied

3.2. RTAB-Map

RTAB-Map (Real Time Appearance Based Mapping) [9] is a ROS compatible visual SLAM algorithm that works with a stereo or RGB-D camera. The robot pose estimated by RTAB-Map has 6 DoF, which makes it suitable for non-planar (e.g. drones) robots. Originally designed as visual loop closing algorithm based on a memory management mechanism [52], the ROS package has evolved to incorporate various functionalities related to SLAM. With these functionalities, RTAB-Map can combine visual SLAM with input data from other sensors such as IMU, lidar, robot odometry and an emulated laser scanner (from RGB-D data).

The visual processing algorithm of RTAB-Map is a feature based implementation of a visual odometry [37] front-end with a pose graph optimization [29] back-end. This architecture has been used in a number of visual SLAM algorithms [7,16,17,32].

Visual odometry work as follows. In each input image, features (configurable as GoodFeaturesToTrack [GFTT] [53], SURF [40], BRIEF [54], etc.) are extracted and matched against those in the previous image. For this, RTAB-Map can use optical flow or matching by distance minimization between descriptor vectors. Then, the global 3D coordinates associated with the tracked features are used as input in a perspective-n-point algorithm [37], which allows to estimate the camera pose in the reference frame of the 3D coordinates. Whenever the number of keypoint matches falls below a fixed threshold, an image is detected as a keyframe and a node is added to a graph of keyframe poses. The estimated camera pose is continuously optimized by bundle adjustment [38] along with the camera pose of the last keyframes and the 3D coordinates of keypoints visible in these keyframes. Camera pose estimation is robust to dynamic scenes, since a feature prediction scheme is employed on the process.

A visual memory formed by quantized keypoint descriptors (bag of words) [52] is managed by the algorithm to allow detection of revisited places. For this to be executed with real-time performance, the memory is partitioned in short term memory (STM), working memory (WM) and long term memory (LTM). The STM is composed by the last keyframes and bag of words descriptor associated with each of them; the WM is formed by recent keyframes excluding those of the STM; finally, the LTM is formed by keyframes that correspond to locations not recently visited. Each memory partition has a fixed size. The algorithm manages which keyframes should be moved in and out the STM, WM and LTM according to loop detection.

To detect loop closures, the bag of words of the current image is compared to those of the nodes in WM. If the algorithm detects a loop closure, a graph optimization routine is triggered, optimizing the node with the current camera position and those kept in the WM.

RTAB-Map is flexible and configurable in the sense that it allows the integration of robot odometry and laser odometry to replace visual odometry. If it is the case that a lidar is available, scan matching transformation is estimated between pairs of point clouds by the ICP algorithm [55] or between the scan and the map.

As the representation for the produced map, the algorithm outputs dense 3D point clouds of the mapped area, a 2D occupancy grid and also a 3D occupancy grid.

3.2.1. RTAB-Map ROS Implementation

RTAB-Map is implemented on ROS with a main node called `rtabmap`. The main node handles node creation and visual memory management. Odometry is computed by the nodes `rgbd_odometry`, `stereo_odometry` or `icp_odometry`, depending on the category of desired odometry computation. The configured odometry also imposes which topics (Table 3) will be required by the main node. The main node publishes the current map in the topic `grid_map` (occupancy grid map) or in the topic `cloud_map` (dense point cloud). In addition, the estimated pose is set as the relative transformation from the `map` reference frame to the `odom` reference frame in the `tf_tree` of the robot.

Table 3. List of relevant published/subscribed topics for the RTAB-Mapping ROS package.

Topic	Message	Type
odom	`nav_msgs/Odometry`	Subscribed
rgb/image	`sensor_msgs/Image`	Subscribed
rgb/camera_info	`sensor_msgs/CameraInfo`	Subscribed
depth/image	`sensor_msgs/Image`	Subscribed
scan	`sensor_msgs/Image`	Subscribed
grid_map	`nav_msgs/OccupancyGrid`	Published
cloud_map	`sensor_msgs/PointCloud2`	Published

We configure RTAB-Map to use visual odometry only. Given its flexible nature, RTAB-Map has a considerable list of adjustable parameters[3]. Table 4 shows a list of relevant parameters and the values used in the experiments according to our description of the algorithm.

Table 4. List of relevant parameters for the RTAB-Mapping ROS package.

Topic	Message	Type
`Grid/Cellsize`	0.05	Occupancy grid resolution (m)
`Mem/STMsize`	10	Size of STM
`Rtabmap/MemoryThr`	0	Size of WM (0 = Inf.)
`Rtabmap/DetectionRate`	1	Detection rate (Hz)
`Vis/FeatureType`	6	Type of visual feature (6 = GFTT/BRIEF)
`Vis/Cortype`	0	Correspondence estimation (0 = feature matching)

4. Analysis Methodology

In this section, we explain the methodology employed to analyze the selected ROS packages for the task of indoor mapping and navigation. A description of the used materials and datasets utilized in the experiments is given along with the reasons that motivate each of them. Also, we describe the metrics used to evaluate the selected algorithms quantitatively and qualitatively. Finally, we list which functionalities we find important for the task of robotic mapping and navigation and the experiments carried out to assess whether and how each algorithm implements these features.

4.1. Datasets

Among many other tools, ROS provides the functionality to record multiple streams of data in a file format called ROS bag. A ROS bag is a synchronized collection of sensor data (odometry, color and depth images, etc.) that can be played as if the recorded data streams were being collected online.

In the following, we list each category of ROS bag used in the experiments.

[3] http://wiki.ros.org/rtabmap_ros

4.1.1. Gazebo Simulation Datasets

A common procedure in robotics is to evaluate algorithms in simulators before employing robotic platforms to operate in real world scenarios. Taking advantage of the simulation tools available on ROS, we evaluate the selected algorithms on 3D virtual environments generated by computer graphics. This evaluation can make explicit certain properties related to the mapping process that we would not have access otherwise (e.g. the exact distance travelled by a robot).

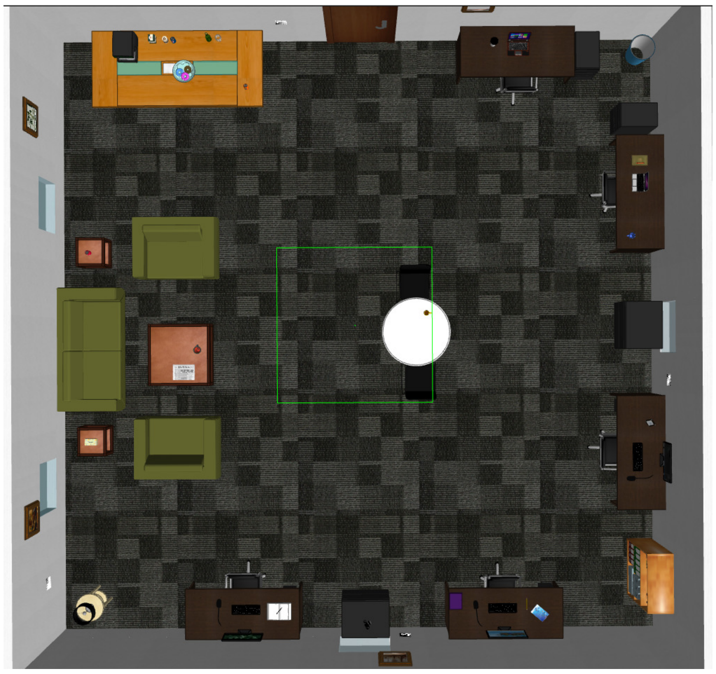


Figure 1. Top view of the virtual environment in Gazebo simulator from which we generated two datasets.

Hence, we use the Gazebo simulator [50] and a publicly available virtual environment [56] to analyze the selected mapping algorithms.

Two different datasets were generated in the virtual environment shown on Figure 1. The selected virtual environment is a single room mimicking an office, in which there are textured 3D objects such as couches, desks, bookshelves, etc.

A simulated model of the Turtlebot 2 robot was teleoperated to travel the virtual environment performing a short trajectory, which generated the dataset *sim_short* and a long trajectory, which generated the dataset: *sim_long*.

4.1.2. TUM RGB-D Datasets

The well known TUM RGB-D datasets [21] offers RGB-D data synchronized with ground truth along with assessment tools in a complete benchmark package. It allows a quantitative evaluation of RGB-D SLAM algorithms, since statistics of error metrics can be calculated when using the benchmark.

We select the four following sequences from the available datasets: *fr2/pioneer_360*, *fr2/pioneer_slam*, *fr2/pioneer_slam2* and *fr2/pioneer_slam3*. The version of these datasets in ROS bag format has data streams of a Pioneer mobile robot equipped with a Kinect RGB-D camera and a SICK laser scanner. The data was captured with the robot being tele-operated to navigate an open warehouse having some scattered objects. For this reason, these datasets are compatible with RTAB-Map and with the GMapping algorithm, which assumes a robot performing only 2D planar motions as is the case of the discussed image sequences. However, it should be noted that the ROS bag files are not compatible with RTAB-Map because the recorded transformation tree is incomplete. As a consequence, we had to use the standalone C++ library (outside of ROS) version of RTAB-Map to process the datasets. This technical issue does not change the algorithm and is mentioned here to possibly guide users having this same problem.

4.1.3. ROS Bags Collected at Natalnet-UFRN

We are interested in analyzing the result of the GMapping and RTAB-Map ROS packages for datasets collected at facilities of the Natalnet Laboratory from Federal University of Rio Grande do Norte (UFRN).

For this, we operate a Turtlebot 2 equipped with a Kinect RGB-D camera with a joystick. The robot moves around the facilities recording in a ROS bag file data streams with registered color/depth images (RGB-D data), robot odometry, an emulated laser scan (computed from depth data) and the robot *tf_tree*

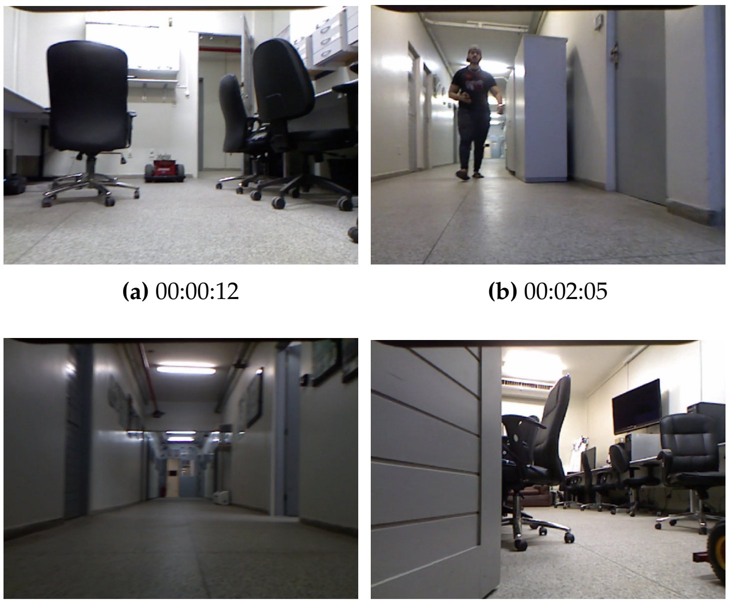

(a) 00:00:12 **(b)** 00:02:05 **(c)** 00:03:03 **(d)** 00:04:31

Figure 2. Images from the *DCA_corridor* dataset. Also indicated is the correspondent timestamp in which each image was captured.

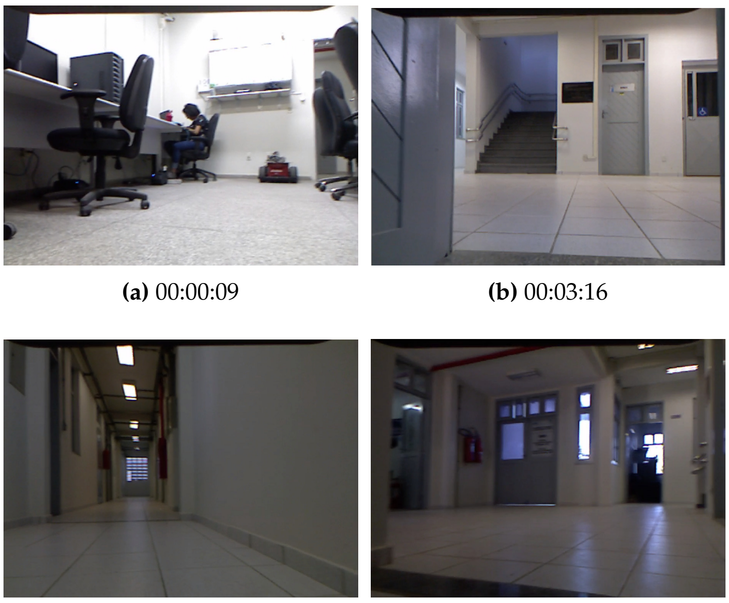

(a) 00:00:09 (b) 00:03:16 (c) 00:03:55 (d) 00:05:38

Figure 3. Images from the *DCA_complete* dataset. Also indicated is the correspondent timestamp in which each image was captured.

Two datasets were collected: *Natalnet-DCA_corridor* and *Natalnet-DCA_complete*. Images depicting the mapped environment are shown in Figure 2 and Figure 3 respectively. In both datasets, the robot starts from our research laboratory and travels throughout the department corridors, finishing its trajectory at our laboratory.

During the data collection process, the scenes captured have different illuminations conditions, mixed amount of texture and also persons walking in front of the robot. All these conditions impose difficulties that the mapping algorithms must overcome.

4.2. SLAM Performance

From Gazebo simulation and TUM RGB-D datasets, a quantitative evaluation of the accuracy of SLAM algorithms can be carried out. This is possible because the estimated data can be compared against a ground truth.

Accordingly, the maximum and root mean squared (RMSE) Absolute Trajectory Error (ATE), a standard metric for SLAM accuracy [21], are computed for each dataset of these simulations and TUM RGB-D categories. The ATE metric measures the misalignment between the ground truth trajectory and the robot trajectory computed by a SLAM algorithm.

4.3. Mapping Performance

Both GMapping and RTAB-Map produces as output a map that is suitable for robot navigation known as an occupancy grid [1]. We evaluate qualitatively the quality of the maps produced by each algorithm by visually inspecting if the results are consistent with the static portion of the environment.

4.4. Reuse of Grid Maps in Localization Only Mode

Our evaluation of the selected ROS algorithms for SLAM also involves how the produced grid maps can be employed in robot navigation. Hence, we conduct an experiment assessing whether the computed grid maps can be loaded by the robot to be reused in another mapping session. There are two configurations for this scenario: 1. a SLAM algorithm loads a previously computed map, modifying it if it detects this is necessary e.g. to correct wrong estimates or 2. a SLAM algorithm loads a previously computed map in localization only mode i.e. the robot uses its sensors to localize itself within the map with mapping capability disabled.

The advantage of the first configuration is the capability to extend the original map to include non explored areas although with the risk of overwriting the map with wrong estimates. The second configuration offers the possibility of robots with different sensors than those used on the map construction to localize themselves within the map but they are only allowed to navigate in the original mapped area.

4.5. System Configuration

The experiments were carried out using a laptop computer with an Intel Core i7-7500U 2.70Ghz processor and 8 GB of RAM running Ubuntu 16.04 with ROS Kinectic. This computer was used to run the ROS nodes responsible to collect the ROS bags of our datasets, process the operations commands for the robot, execute the mapping algorithms and show the current state of the mapping process.

5. Experimental Analysis

This section shows results of the experiments carried out to evaluate GMapping and RTAB-Map SLAM algorithms. An analysis of the algorithms is drawn based on their performance in relation to SLAM accuracy, quality of the produced maps and aspects important to how the produced maps are reused in navigation. We make available video sessions[4] of the system in execution, offering another resource to follow our analysis.

5.1. SLAM Accuracy in Simulation

The SLAM accuracy of GMapping and RTAB-Map represented by the RMSE of the ATE for the datasets from Gazebo simulation are shown on Table 5.

Table 5. Absolute Trajectory Error (ATE) of the SLAM algorithms on datasets from simulation (maximum error is shown enclosed in parentheses).

	RTAB-Map	**GMapping**
sim_short	0.24 m (0.447 m)	0.037 m (0.174 m)
sim_long	0.071 m (0.170 m)	0.047 m (0.211 m)

In these datasets, GMapping presents a lower RMSE than RTAB-Map in both *sim_short* and *sim_long* datasets. We verified that this reflects the fact that GMapping uses an error free odometry: Gazebo does not add noise to the actual synthetic robot trajectory when simulating this data. Also, since RTAB-Map relies on appearance data rather than geometry data (as does GMapping), the computer generated texture and lighting directly influence its performance. RTAB-Map has a better SLAM performance on *sim_long* than it has on *sim_short* because the robot navigates closer to objects that it does in *sim_short*.

If configured to run the experiments with robot odometry instead of visual odometry, the results are very similar to those of GMapping.

5.2. SLAM Accuracy in Real Datasets

Table 6 shows the SLAM accuracy of the selected algorithms for the TUM RGB-D datasets.

[4] https://www.youtube.com/watch?v=Q8Lk59yEfvA&list=PL3-5cPiG5Gky80ZXGX8KJGyExF_dzxcb0

Table 6. Absolute Trajectory Error (ATE) of the SLAM algorithms on TUM RGB-D datasets (maximum error is shown enclosed in parentheses).

	RTAB-Map	GMapping
fr2/pioneer_360	0.190 m (0.823 m)	0.161 m (0.444 m)
fr2/pioneer_slam1	0.514 m (1.084 m)	0.224 m (0.565 m)
fr2/pioneer_slam2	0.115 m (0.312 m)	0.418 m (0.988 m)
fr2/pioneer_slam3	0.393 m (0.719 m)	0.228 m (0.838 m)

The results show that GMapping has a marginally lower RMSE ATE for the datasets *fr2/pioneer_360* and *fr2/pioneer_slam3* than RTAB-Map. It has a significantly lower RMSE for the dataset *fr2/pioneer_slam1*. These results clearly show the advantage of lidar based SLAM, since in the selected TUM datasets of these experiments GMapping uses a SICK laser as its sensor instead of the emulated scan from RGB-D data. However, RTAB-Map has lower error than GMapping for the dataset *fr2/pioneer_slam2*. This can be explained by the fact that in most part of the mapping session, RTAB-Map is not able to compute visual odometry estimates. This in turn biases the ATE metric since the error is being computed for an incomplete trajectory. In the other hand, the same result shows that RTAB-Map is able to recover its pose (compute a valid robot pose after being considered lost).

5.3. Mapping Quality: Simulation Datasets

Occupancy grid maps produced by GMapping and RTAB-Map after processing the simulated datasets can be viewed in Figure 4 (*sim_short*) sequence and Figure 5 (*sim_large*) sequence.

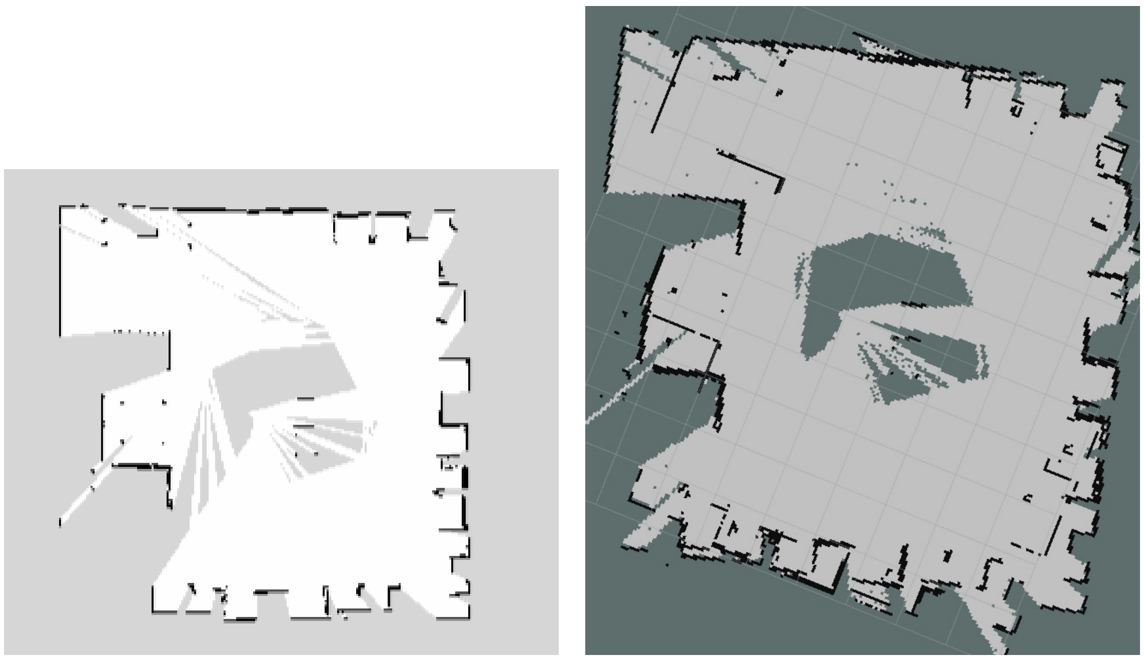

(a) GMapping **(b)** RTAB-Map

Figure 4. Results on the *sim_short* simulation dataset.

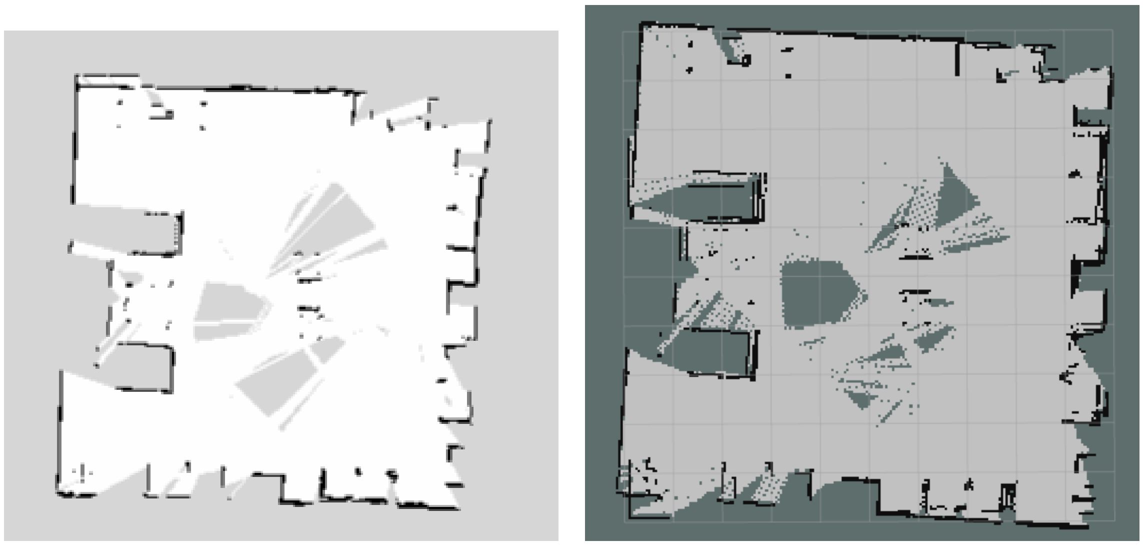

(a) GMapping **(b)** RTAB-Map

Figure 5. Results on the *sim_long* simulation dataset.

Comparing the computed grids with the top view of the mapped environment (Figure 1), it can be seen that GMapping reconstructs a correct representation of the environment, while the grid produced by RTAB-Map has visible misalignments.

Because RTAB-Map does not use odometry, the computed grid by this algorithm shows more proeminent misalignments than those present in the results of GMapping, which is using error free odometry estimates.

The unmapped portion of the grids (center of the map) are present because the robot trajectory did not performed sensor measurements in these areas.

5.4. Mapping Quality: TUM Datasets

The occupancy grid maps computed by GMapping for each dataset of the TUM set are shown on Figure 9. Occupancy grid maps cannot be computed by RTAB-Map for the TUM datasets because these experiments were run on the standalone version of RTAB-Map (as explained in Section 4.1.2). Instead, as the result of RTAB-Map we compute top view projections of the point clouds.

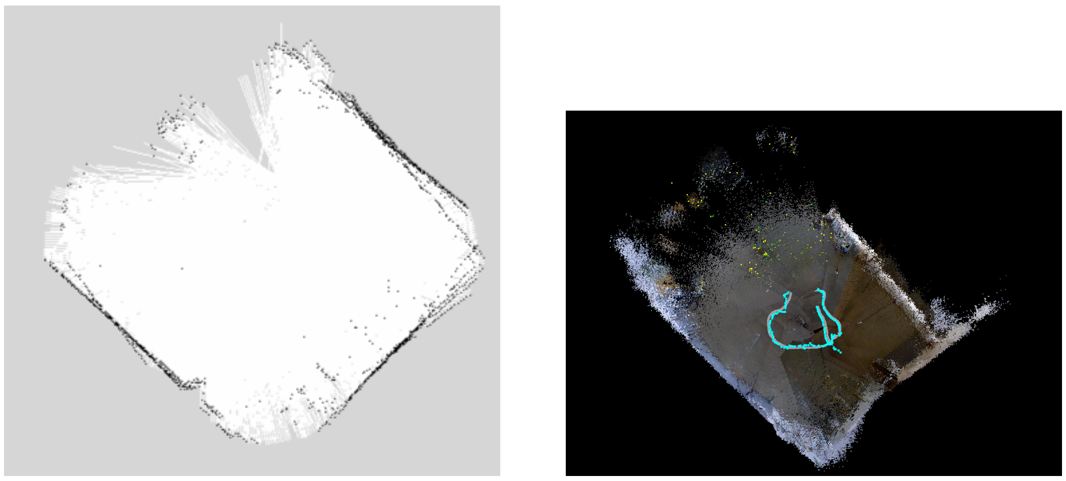
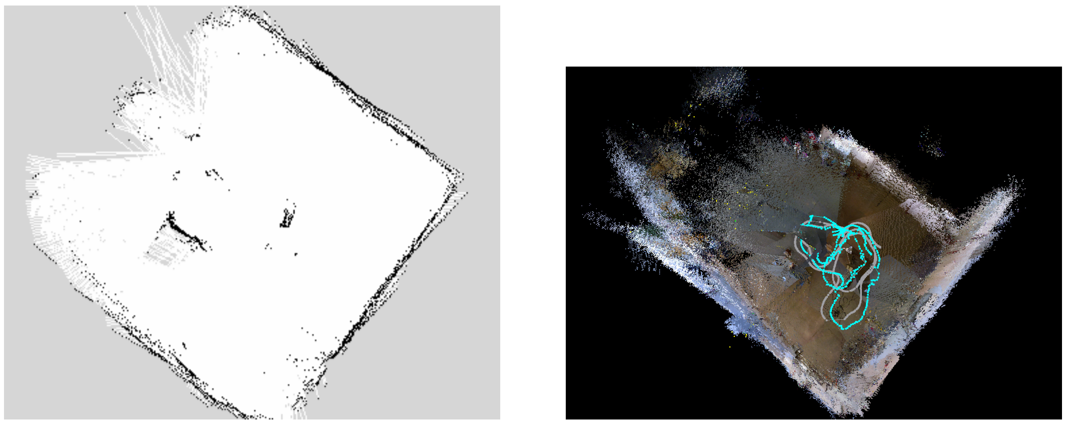
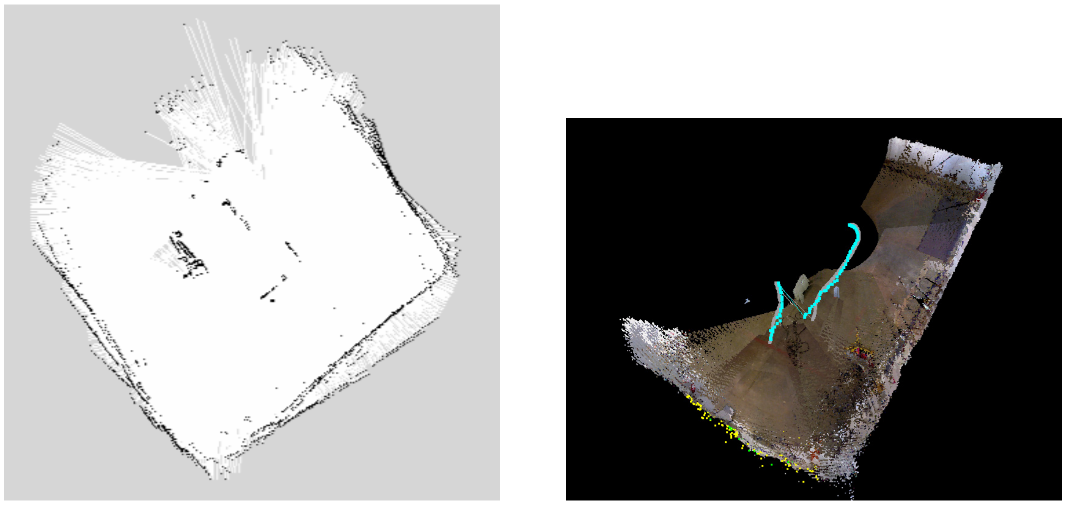

(a) GMapping **(b)** RTAB-Map

Figure 6. Results on the TUM *fr2/pioneer_360* dataset.



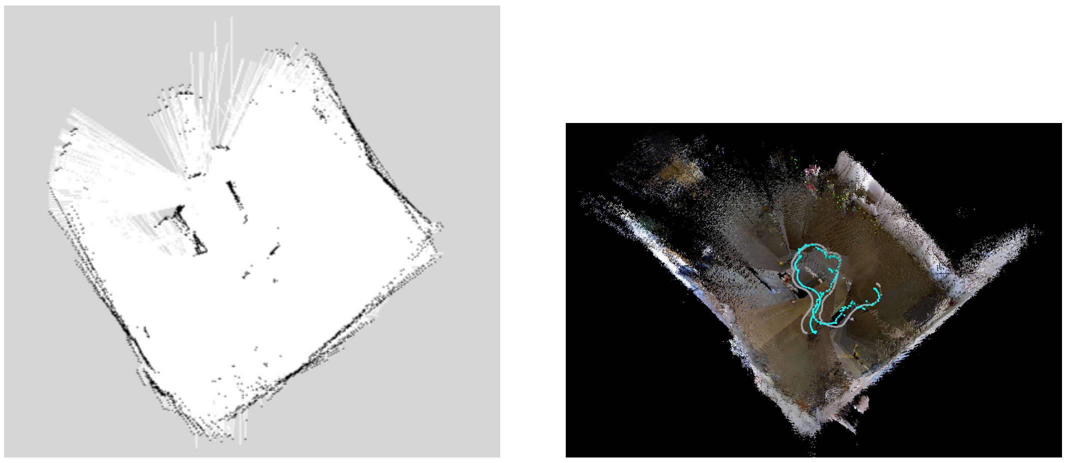
(a) GMapping **(b)** RTAB-Map

Figure 7. Results on the TUM *fr2/pioneer_slam1* dataset.



(a) GMapping **(b)** RTAB-Map

Figure 8. Results on the TUM *fr2/pioneer_slam2* dataset.



(a) GMapping **(b)** RTAB-Map

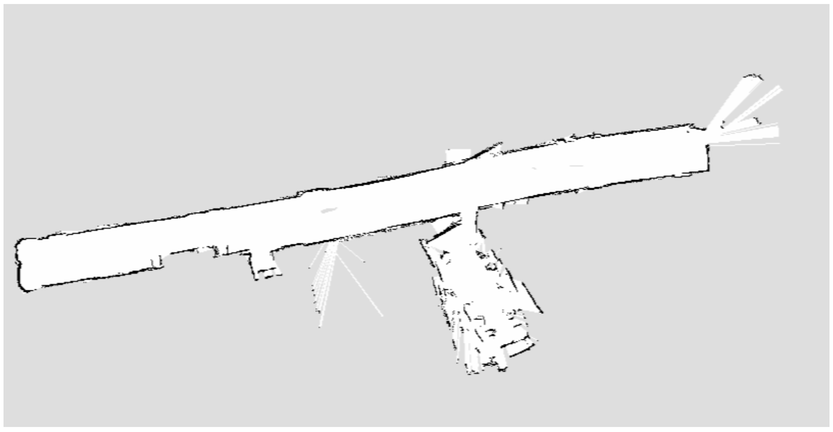
Figure 9. Results on the TUM *fr2/pioneer_slam3* dataset.

GMapping computes occupancy grid in the form of a large squared area predominantly composed of free spaces in all TUM datasets. This results is in accordance with the large warehouse mapped in these images. Note however that the maps were computed processing laser data instead of the RGB-D data from Kinect. Nevertheless, duplicated walls (i.e. accumulated error) can be seen in the maps of *fr2/pioneer_slam2* and *fr2/pioneer_slam3* datasets.

RTAB-Map computes 3D point clouds coherent with the mapped environment albeit some maps are clearly incomplete. During the mapping process, RTAB-Map estimates incorrect robot poses that are later corrected (*fr2/pioneer_360* e *fr2/pioneer_slam*). It is able to detect a large loop closure (*fr2/pioneer_slam2*) as well. As already discussed on Section 5.2, the robot does not compute odometry estimates for a long period of time because it visits an area without any visual features within the range of the camera. Duplicated walls are present on the maps of *fr2/pioneer_slam1* and non orthogonal walls on the maps of *fr2/pioneer_360* and *fr2/pioneer_slam2*.

5.5. Mapping Quality: Datasets Collected at UFRN

The computed occupancy grids by GMapping and RTAB-Map for the data collected at our university are shown on Figure 10 (*DCA_corridor* dataset).


(a) GMapping

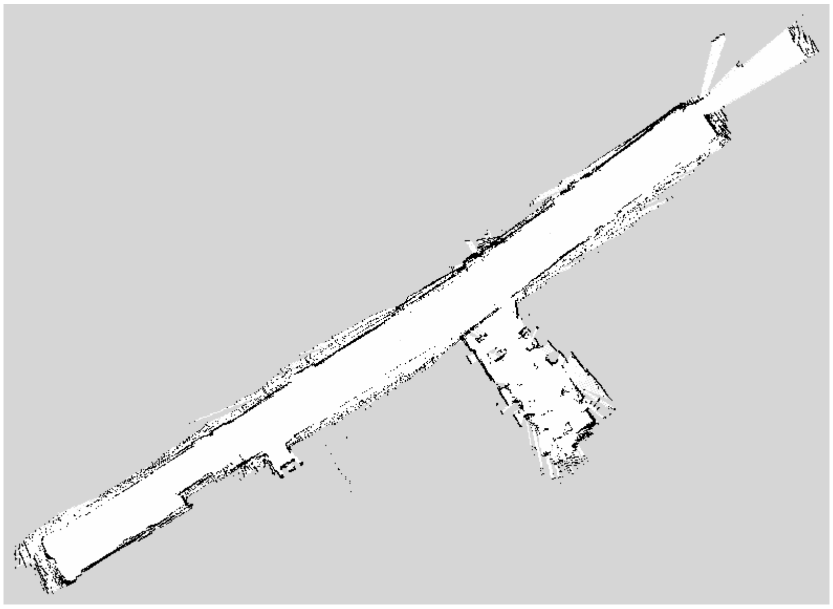
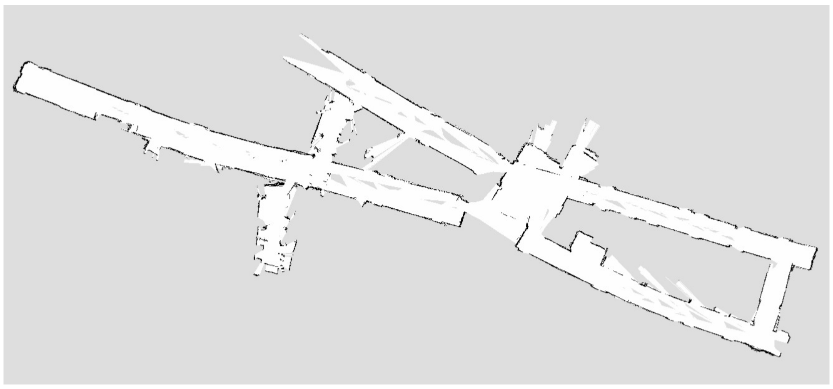

(b) RTAB-Map

Figure 10. Results on the *DCA_corridor* dataset.


(a) GMapping

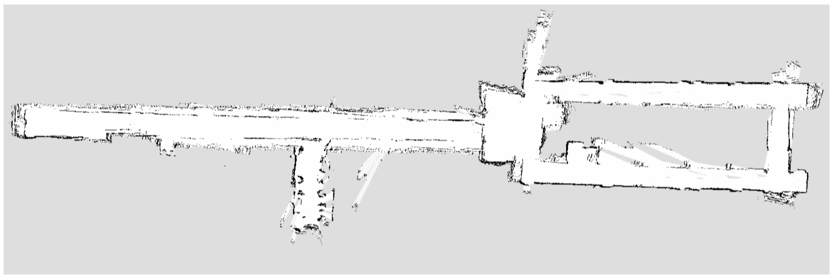

(b) RTAB-Map

Figure 11. Results on the *DCA_complete* dataset.

When mapping the dataset *DCA_corridor*, GMapping estimates the main corridor in a non perpendicular way in relation to the laboratory from which the robot starts the mapping session. The walls of the corridor are coherent with the true structure of the environment. Importantly, moving persons appearing in the images are not added to the grid map. This is correct, since the grid should capture the static structure of the environment. The robot arrives back in the laboratory, resulting in duplicated estimates of the room.

For the larger dataset *DCA_complete*, the same non perpendicular alignment between the laboratory and the corridor is estimated. During the session, the robot enters an area of the department in which there is a fork joining two secondary corridors (left corridor and right corridor). It then proceeds with the mapping through the left corridor, completes a loop around the building and arrives in the same location of the fork. However, this time the robot enters the area with the fork arriving from the right corridor, when it is clearly seen that it has accumulated large mapping errors (a duplicate of the main corridor and laboratory is estimated on the map).

If used with the configuration stated in Section 3.2.1, RTAB-Map computes SLAM estimates of very bad quality. This happens because of the low availability of visual features on the mapped area, which are not sufficient for visual odometry and consequently for the pose optimized pose graph. For this reason, we configured RTAB-Map to estimate odometry using both robot odometry and emulated laser scan from the RGB-D camera.

RTAB-Map computes an estimate of the map on *DCA_corridor* that is coherent with the mapped structure, despite some duplicated walls along the main corridor. Even with these errors, the main portion of the corridor is accurately mapped, which is relevant for route planning and navigation in general. Unlike GMapping, the perpendicular relation between the laboratory and the corridor is represented on the grid. Between 3:11s and 3:15s of the video sequence, we can see that loop closures are correctly detected by the algorithm, which in turn triggers a map optimization.

The complete area of the mapped building is correctly reconstructed by RTAB-Map after processing the dataset *DCA_complete*. However, this result is only available at the end of the dataset, when the algorithm matches images of the laboratory captured at the end of the mapping session with those in the visual memory captured at the beginning. This loop closure detection triggers an optimization that corrects the accumulated error by the algorithm. Before arriving back at the laboratory, the error can be seen by incorrect duplicates of the main corridor, although with a lesser magnitude than those of GMapping.

5.6. Reuse of Maps by the Algorithms

Our evaluation of GMapping of RTAB-Map also involves the reuse of grid maps produced by these algorithms, as explained in Section 4.4. To evaluate this capability, we reused the grid map computed with the best quality (Figure 11.b). GMapping does not have a localization only mode as does RTAB-Map.

Hence, we loaded the map on RTAB-Map and then, supply an approximate estimate of the initial robot pose within the grid map using RVIZ, a ROS tool that offers the possibility to insert the pose via a drag and drop interface.

We then navigate the robot through waypoint locations within the map, for which we use RVIZ to indicate goal locations for robot navigation. A total of five waypoints is given to the robot. The ROS navigation system is not able to compute a valid path for the first waypoint, since it was indicated on a small area with various scattered (and dynamic) objects. For the rest of the waypoints, the robot performed navigation without problems, avoiding moving persons that passed in front of it.

Although GMapping does not offer localization only mode, the ROS package for Adaptive Monte Carlo Localization (AMCL) [57], which works similarly in principle to GMapping, can also be used for this task. We performed the same experiment with a prebuilt map for navigation and confirmed the applicability of AMCL.

5.7. Discussion

When autonomously mapping an environment, it is interesting for the robots to have a number of capabilities. We draw the following considerations based on the results of our experiments and used setup (indoor planar robot with a RGB-D camera). The discussion further assumes that a decent processing power is available for both algorithms.

Regarding SLAM, the algorithms should compute correct pose estimates, detect previously visited places and minimize accumulated error.

The results of our experiments show that both GMapping and RTAB-Map have low error estimates, as evidenced by the RMSE ATE metric. However, GMapping meets these requirements only partially, since it is not able minimize the accumulated error if it grows substantially (as happens in the experiments with our datasets). RTAB-Map recovers robot pose when revisiting places even in cases where visual odometry is not computed for a long period of time.

In relation to the maps produced by SLAM algorithms, the results of experiments and our previous experiences [46] emphasize that the maps computed by SLAM can be useful for robot navigation if this subsystem offers them in the form of a occupancy grid map, as is the case of GMapping and RTAB-Map. The estimated grids must capture the static structure of the mapped environment, since dynamic objects can be avoided by the reactive behavior of path planning algorithms. In this aspect, RTAB-Map performs better than GMapping in scenarios where the robot is equipped with a low cost RGB-D camera as is the case of our experiments.

It is important to note that these findings are not obvious from simulation experiments, since the simulated data are not corrupted with actual noise as those found in the experiments with real robots.

6. Conclusion

This work proposed an experimental analysis of two widely used ROS compatible SLAM packages: GMapping and RTAB-Map. The experiments were carried out by executing SLAM and navigation sessions on datasets from Gazebo simulation, TUM RGB-D benchmarks [21] and using those collected in our university Lab. A discussion is drawn relating practical aspects of the algorithms with the inner workings and main parameters of the respective ROS implementation. Specifically, we found that RTAB-Map is a more complete solution for ground robots equipped with a RGB-D sensor, mainly because of capabilities like accuracy, quality of produced maps and configuration flexibility, which allows it to reuse maps in localization only mode.

Besides the detailed analysis performed, our dataset is also another contribution to the community that can use it in further works. This dataset will be made available as soon as another report will be written or it can be asked by requesting to us. So we believe that the analysis reported on this work should shed light for researchers involved with ROS application design in choosing a SLAM package that computes estimates suitable for ground robots equipped with a RGB-D sensor to autonomously navigate an environment.

Author Contributions: Conceptualization, B.S. and R.X.; methodology, B.S.; software, R.X.; investigation, B.S. and R.X.; resources, B.S. and L.G.; data curation, R.X.; writing—original draft preparation, B.S. and R.X.; writing—review and editing, B.S. and L.G.; supervision, B.S. and L.G.; project administration, B.S. and L.G.; funding acquisition, B.S. and L.G.

Funding: We thank to Brazilian sponsoring agencies for research CNPq and Capes (under grant number 001) for the financial support.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Thrun, S.; Burgard, W.; Fox, D. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*; The MIT Press, 2005.
2. Durrant-Whyte, H.; Bailey, T. Simultaneous localization and mapping: part I. *IEEE Robotics and Automation Magazine* **2006**, *13*, 99–110. doi:10.1109/MRA.2006.1638022.
3. Bailey, T.; Durrant-Whyte, H. Simultaneous localization and mapping (SLAM): part II. *IEEE Robotics and Automation Magazine* **2006**, *13*, 108–117. doi:10.1109/MRA.2006.1678144.
4. Siciliano, B.; Khatib, O. *Springer Handbook of Robotics*; Springer-Verlag: Berlin, Heidelberg, 2007.
5. Vincent, R.; Limketkai, B.; Eriksen, M. Comparison of indoor robot localization techniques in the absence of GPS. *Proceedings of SPIE - The International Society for Optical Engineering* **2010**, pp. 1–8. doi:10.1117/12.849593.
6. Carlone, L.; Aragues, R.; Castellanos, J.; Bona, B. A Linear Approximation for Graph-based Simultaneous Localization and Mapping. Robotics: Science and Systems (RSS), 2012. doi:10.15607/RSS.2011.VII.006.
7. Endres, F.; Hess, J.; Sturm, J.; Cremers, D.; Burgard, W. 3-D Mapping With an RGB-D Camera. *IEEE Transactions on Robotics* **2014**, *30*, 177–187. doi:10.1109/TRO.2013.2279412.
8. Hess, W.; Kohler, D.; Rapp, H.; Andor, D. Real-time loop closure in 2D LIDAR SLAM. 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 1271–1278. doi:10.1109/ICRA.2016.7487258.
9. Labbé, M.; Michaud, F. RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. *Journal of Field Robotics* **2019**, *36*, 416–446. doi:10.1002/rob.21831.
10. Quigley, M.; Conley, K.; Gerkey, B.; Faust, J.; Foote, T.; Leibs, J.; Wheeler, R.; Ng, A.Y. ROS: an open-source Robot Operating System. ICRA Workshop on Open Source Software, 2009, p. 5.
11. Brian Gerkey. ROS, the Robot Operating System, Is Growing Faster Than Ever, Celebrates 8 Years. https://spectrum.ieee.org/automaton/robotics/robotics-software/ros-robot-operating-system-celebrates-8-years, 2015.
12. Grisetti, G.; Stachniss, C.; Burgard, W. Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters. *IEEE Transactions on Robotics* **2007**, *23*, 34–46. doi:10.1109/TRO.2006.889486.
13. Steux, B.; Hamzaoui, O.E. tinySLAM: A SLAM algorithm in less than 200 lines C-language program. Proceedings IEEE International Conference on Control Automation Robotics and Vision,pages (ICARCV). IEEE, 2010, pp. 1975–1979.

14. Kohlbrecher, S.; von Stryk, O.; Meyer, J.; Klingauf, U. A flexible and scalable SLAM system with full 3D motion estimation. 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, 2011, pp. 155–160. doi:10.1109/SSRR.2011.6106777.
15. Engel, J.; Schöps, T.; Cremers, D. LSD-SLAM: Large-Scale Direct Monocular SLAM. Computer Vision – ECCV 2014; Fleet, D.; Pajdla, T.; Schiele, B.; Tuytelaars, T., Eds.; Springer International Publishing: Cham, 2014; pp. 834–849.
16. Mur-Artal, R.; Montiel, J.M.M.; Tardós, J.D. ORB-SLAM: a Versatile and Accurate Monocular SLAM System. *IEEE Transactions on Robotics* **2015**, *31*, 1147–1163. doi:10.1109/TRO.2015.2463671.
17. Mur-Artal, R.; Tardós, J.D. ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras. *IEEE Transactions on Robotics* **2017**, *33*, 1255–1262. doi:10.1109/TRO.2017.2705103.
18. Marder-Eppstein, E.; Berger, E.; Foote, T.; Gerkey, B.; Konolige, K. The Office Marathon: Robust navigation in an indoor office environment. 2010 IEEE International Conference on Robotics and Automation (ICRA), 2010, pp. 300–307. doi:10.1109/ROBOT.2010.5509725.
19. Ortiz, L.; Cabrera, V.; Goncalves, L. Depth Data Error Modeling of the ZED 3D Vision Sensor from Stereolabs. *ELCVIA Electronic Letters on Computer Vision and Image Analysis* **2018**, *17*, 1–15.
20. Cabrera, E.V.; Ortiz, L.E.; Silva, B.M.F.d.; Clua, E.W.G.; Gonçalves, L.M.G. A Versatile Method for Depth Data Error Estimation in RGB-D Sensors. *Sensors* **2018**, *18*. doi:10.3390/s18093122.
21. Sturm, J.; Engelhard, N.; Endres, F.; Burgard, W.; Cremers, D. A benchmark for the evaluation of RGB-D SLAM systems. 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2012, pp. 573–580. doi:10.1109/IROS.2012.6385773.
22. Arumugam, R.; Enti, V.R.; Bingbing, L.; Xiaojun, W.; Baskaran, K.; Kong, F.F.; Kumar, A.S.; Meng, K.D.; Kit, G.W. DAvinCi: A cloud computing framework for service robots. 2010 IEEE International Conference on Robotics and Automation, 2010, pp. 3084–3089. doi:10.1109/ROBOT.2010.5509469.
23. Koubaa, A. ROS As a Service: Web Services for Robot Operating System. *Journal of Software Engineering for Robotics* **2015**, *1*, 1.
24. Costa, L.F.; Gonçalves, L.M.G. RoboServ: A ROS Based Approach towards Providing Heterogeneous Robots as a Service. 2016 XIII Latin American Robotics Symposium and IV Brazilian Robotics Symposium (LARS/SBR), 2016, pp. 169–174. doi:10.1109/LARS-SBR.2016.35.
25. Crick, C.; Jay, G.; Osentoski, S.; Pitzer, B.; Jenkins, O.C. Rosbridge: ROS for Non-ROS Users. International Symposium on Robotics Research (ISRR), 2011.
26. Alisher, K.; Alexander, K.; Alexandr, B. Control of the Mobile Robots with ROS in Robotics Courses. *Procedia Engineering* **2015**, *100*, 1475 – 1484. doi:http://dx.doi.org/10.1016/j.proeng.2015.01.519.
27. Rezeck, P.A.F.; Vieira, M.A.M.; Chaimowicz, L.; Campos, M.F.M. On the Development of a Robotic System for Telepresence. 2013 Latin American Robotics Symposium and Competition, 2013, pp. 8–13. doi:10.1109/LARS.2013.49.
28. Costa, L.F.S.; Nascimento, T.P.; Maia, R.d.S.; Gonçalves, L.M.G. N-learning: An Approach for Learning and Teaching Skills in Multirobot Teams. *Robotica* **2019**, p. 1–21. doi:10.1017/S0263574719000468.
29. Grisetti, G.; Kummerle, R.; Stachniss, C.; Burgard, W. A Tutorial on Graph-Based SLAM. *IEEE Intelligent Transportation Systems Magazine* **2010**, *2*, 31–43. doi:10.1109/MITS.2010.939925.
30. Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. Eighteenth National Conference on Artificial Intelligence; American Association for Artificial Intelligence: Menlo Park, CA, USA, 2002; pp. 593–598.
31. Davison, A.J. Real-Time Simultaneous Localisation and Mapping with a Single Camera. IEEE International Conference on Computer Vision (ICCV). IEEE, 2003, p. 1403.
32. Klein, G.; Murray, D. Parallel Tracking and Mapping for Small AR Workspaces. IEEE International Symposium on Mixed and Augmented Reality (ISMAR). IEEE, 2007, pp. 1–10. doi:http://dx.doi.org/10.1109/ISMAR.2007.4538852.
33. Se, S.; Lowe, D.; Little, J. Mobile Robot Localization and Mapping with Uncertainty using Scale-Invariant Visual Landmarks. *The International Journal of Robotics Research* **2002**, *21*, 735–758, [https://doi.org/10.1177/027836402761412467]. doi:10.1177/027836402761412467.
34. Geiger, A.; Ziegler, J.; Stiller, C. StereoScan: Dense 3d reconstruction in real-time. 2011 IEEE Intelligent Vehicles Symposium (IV), 2011, pp. 963–968. doi:10.1109/IVS.2011.5940405.

35. Endres, F.; Hess, J.; Engelhard, N.; Sturm, J.; Cremers, D.; Burgard, W. An Evaluation of the RGB-D SLAM System. IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2012, pp. 1691–1696.
36. Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age. *IEEE Transactions on Robotics* **2016**, *32*, 1309–1332. doi:10.1109/TRO.2016.2624754.
37. Scaramuzza, D.; Fraundorfer, F. Visual Odometry [Tutorial]. *IEEE Robotics Automation Magazine* **2011**, *18*, 80–92. doi:10.1109/MRA.2011.943233.
38. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*, 2 ed.; Cambridge University Press: New York, NY, USA, 2003.
39. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision* **2004**, *60*, 91–110. doi:http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.
40. Bay, H.; Ess, A.; Tuytelaars, T.; Gool, L.V. Speeded-Up Robust Features (SURF). *Computer Vision and Image Understanding* **2008**, *110*, 346–359. doi:http://dx.doi.org/10.1016/j.cviu.2007.09.014.
41. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An efficient alternative to SIFT or SURF. IEEE International Conference on Computer Vision (ICCV). IEEE, 2011, pp. 2564–2571. doi:10.1109/ICCV.2011.6126544.
42. Forster, C.; Pizzoli, M.; Scaramuzza, D. SVO: Fast semi-direct monocular visual odometry. 2014 IEEE International Conference on Robotics and Automation (ICRA), 2014, pp. 15–22. doi:10.1109/ICRA.2014.6906584.
43. Gálvez-López, D.; Tardós, J.D. Bags of Binary Words for Fast Place Recognition in Image Sequences. *IEEE Transactions on Robotics* **2012**, *28*, 1188–1197. doi:10.1109/TRO.2012.2197158.
44. Huang, A.S.; Bachrach, A.; Henry, P.; Krainin, M.; Fox, D.; Roy, N. Visual odometry and mapping for autonomous flight using an RGB-D camera. In Proc. of the Intl. Sym. of Robot. Research (ISRR), 2011.
45. Dryanovski, I.; Valenti, R.G.; Xiao, J. Fast visual odometry and mapping from RGB-D data. 2013 IEEE International Conference on Robotics and Automation, 2013, pp. 2305–2310. doi:10.1109/ICRA.2013.6630889.
46. Silva, B.M.F.; Xavier, R.S.; do Nascimento, T.P.; Gonçalves, L.M.G. Experimental evaluation of ROS compatible SLAM algorithms for RGB-D sensors. 2017 Latin American Robotics Symposium (LARS) and 2017 Brazilian Symposium on Robotics (SBR), 2017, pp. 1–6. doi:10.1109/SBR-LARS-R.2017.8215331.
47. Santos, J.M.; Portugal, D.; Rocha, R.P. An evaluation of 2D SLAM techniques available in Robot Operating System. 2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2013, pp. 1–6. doi:10.1109/SSRR.2013.6719348.
48. Alexander Buyval, Ilya Afanasyev, E.M. Comparative analysis of ROS-based monocular SLAM methods for indoor navigation. 2016 International Conference on Machine Vision (ICMV), 2016, Vol. 10341. doi:10.1117/12.2268809.
49. Ibragimov, I.Z.; Afanasyev, I.M. Comparison of ROS-based visual SLAM methods in homogeneous indoor environment. 2017 14th Workshop on Positioning, Navigation and Communications (WPNC), 2017, pp. 1–6. doi:10.1109/WPNC.2017.8250081.
50. Koenig, N.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), 2004, Vol. 3, pp. 2149–2154 vol.3. doi:10.1109/IROS.2004.1389727.
51. Quigley, M.; Gerkey, B.; Smart, W.D. *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*, 1st ed.; O'Reilly Media, Inc., 2015.
52. Labbé, M.; Michaud, F. Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation. *IEEE Transactions on Robotics* **2013**, *29*, 734–745. doi:10.1109/TRO.2013.2242375.
53. Shi, J.; Tomasi, C. Good Features to Track. IEEE Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 1994, pp. 593–600.
54. Calonder, M.; Lepetit, V.; Strecha, C.; Fua, P. BRIEF: Binary Robust Independent Elementary Features. Computer Vision – ECCV 2010; Daniilidis, K.; Maragos, P.; Paragios, N., Eds.; Springer Berlin Heidelberg: Berlin, Heidelberg, 2010; pp. 778–792.
55. Segal, A.; Hähnel, D.; Thrun, S. Generalized-ICP. Robotics: Science and Systems (RSS), 2009. doi:10.15607/RSS.2009.V.021.

56. Rasouli, A.; Tsotsos, J.K. The Effect of Color Space Selection on Detectability and Discriminability of Colored Objects. *CoRR* **2017**, *abs/1702.05421*, [1702.05421].
57. Fox, D.; Burgard, W.; Dellaert, F.; Thrun, S. Monte Carlo Localization: Efficient Position Estimation for Mobile Robots. Proceedings of the Sixteenth National Conference on Artificial Intelligence and the Eleventh Innovative Applications of Artificial Intelligence Conference Innovative Applications of Artificial Intelligence; American Association for Artificial Intelligence: Menlo Park, CA, USA, 1999; AAAI '99/IAAI '99, pp. 343–349.