# Map My World

Christopher Ohara

**Abstract**—Simultaneously localization and mapping is a critical field in computer vision and autonomous robotics. Estimating the current pose of a robot and its relative position to the map is a challenging task due to a requirement to solve two unknown conditions in parallel while also attempting to predict the environment in real-time. In this project, a robot is placed in two different environments with a task of mapping and returning its relative location. Utilizing the Real-Time Appearance-Based Mapping algorithm in ROS, a differential-drive robot is equipped with a depth-camera and Hokuyo rangefinder and the proof-of-concept is verified in a simulated environment using Gazebo and RViz. The result is that both the custom robot was able to successfully map two different environments; one benchmark and one custom-built.

**Index Terms**—Robotics, Udacity, ROS, SLAM, RTAB-MAP, teleoperation, Path Planning, Navigation.

## 1 Introduction

SOLVING simultaneous localization and mapping (SLAM) problems is not a trivial task. In an environment that a robotic device is unfamiliar with, the parallel tasks of self-localization and properly interpreting the map needs a starting point. Improper implementation can lead to hazards for the robots and potentially people in the case of human-robot interaction and collaboration. This is usually solved by using traditional computer vision techniques surrounding target recognition (unique waypoints that look like QR codes or line-following) but this only works in environments that have been prepared for autonomous robotic environments that are primarily intended to be static. Once any form of dynamicism appears (obscuring the robot's vision or hampering certain perception/cognition characteristics), using traditional methods will fail. This impacts the safety, performance, and predictability for the robot's behavior.

To combatant these limitations, the robot can be given decentralized tasks of self-localization and mapping by creating its own internal map. In this project, this is completed using sensor fusion data from a hi-res depth camera and a Hokuyo UST-20LX scanning laser rangefinder. These hardware devices are attached to a custom differential-drive robot with the task mapping its given environments while returning its relative position to the newly mapped information. Navigation is completed using a teleoperation script and manual control. The final requirement of completing at least three map closures is also given to ensure validation and verification of the given implementation.

## 2 Background

When considering discrete time-steps, sensor observations are used to estimate the agent's location relative to a map depending on a probabilistic distribution. Using Bayes' rule, the posterior location is sequentially updated based on a probability density function distribution. Furthermore, Kalman filters and particle filters play a major role in these predictions (Monte Carlo Localization) by inputting features from the map to provide an estimation for the posterior probability function related to the pose and map. [1]

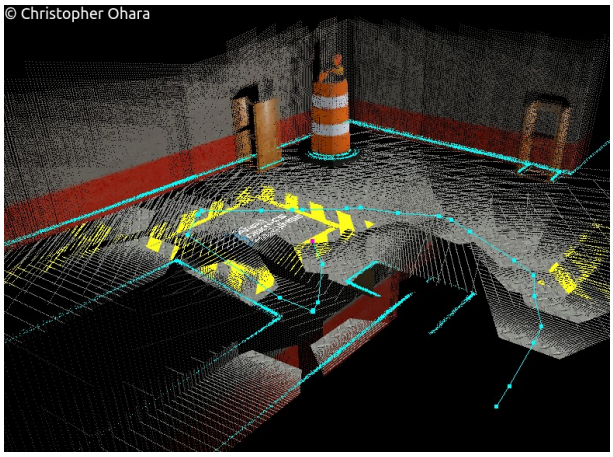


Fig. 1: Custom World as shown in GraphView for RTAB-MAP, demonstrating the resolution.

### 2.1 Kalman Filters

The general Kalman filter is a derivative of the Bayes filter that is used for linear quadratic estimation. The Kalman Filter has two steps. The first of which is the prediction step, where *a priori* state estimates and error covariances are calculated. Next, an update step is completed, in which these values are updated *a posteriori*. Measurements continue to be received and this process estimates the true states recursively. Due to its requirements towards Gaussian (or normal) distributed curves, it has some limitations. While this method is very beneficial for simplistic, linear systems, it fails to deliver high precision in stochastic environments.

Two common flavors of the Kalman Filter are the Extended Kalman Filter (EKF) and Unscented Kalman Filter (UKF). The EFK is relatively easy to implement and can be used in non-linear systems. The EFK's main task is to arrive at a Taylor expansion equivalent that allows a piece-wise linearization around the mean ($\mu$). The UKF, on the other hand, does not require linearization around the mean, by using a transform that maps various points onto a Gaussian and is very suitable for non-linear systems.[2]

*NB: all forms of KF do not assume the noise will be linear.*

### 2.2 Particle Filters

Particle Filters are also a derivative of the Bayes Filter. However, in its vanilla form, it is able to deal with non-linear systems by discretizing random samples into ”particles”. The more samples that are available, the more accurate the estimate will be. However, in practice, a high number of samples can become problematic (as was found during the experiment). The Monte Carlo Localization (MCL) genetic algorithm is very good at the mutation-selection of particles. The samples are taken from the PDF and given a weighted probability. In order to avoid problems in weight disparity, re-sampling is undertaken and particles with negligible weight values are replaced.[3]

Adaptive Monte Carlo Localization (AMCL) is a flavor of the MCL that implements Kullback-Leibler distance-sampling (KLD), which allows for accurate pose tracking of a robot on a known map. The AMCL has takes inputs of a laser-based map, laser scans, and transform messages. It then outputs pose estimates that are used for localization. An important aspect of the AMCL is that it is able to transform inputs (i.e. laser scan data) to the intended frame (odometry, base). This allows for the optimization of techniques like Dead Reckoning, which are generally not optimal for arriving at a goal and only used in non-mission-critical autonomous applications.[4]

### 2.3 SLAM

In general, robot motion is susceptible to errors when making measurements. Using both the Kalman and Particle Filters is also loosely known as ”Monte Carlo Localization.” By extending these in conjunction with a graphical method for visualization, an improved solution for SLAM can be approached (as compared to using the EKF or another algorithm individually). Currently, two popular approaches are GraphSLAM and FastSLAM. GraphSLAM works by using sparse matrices (and usually information filters) to generate a factor graph to analyze inter-dependencies between two observations. If the two observations share the same data about an object, landmark, or waypoint, then this information is updated into the matrix as a ”known” variable.

FastSLAM is an algorithm that utilizes a Rao-Blackwellized representation of the posterior while integrating particle filter and Kalman filter representations based on an inherent conditional independence property. This algorithm has been used in DARPA’s MARS and Grand Challenge missions for autonomous vehicles. FastSLAM requires fewer particles than using a traditional particle filter (e.g. in MCL or AMCL) but has limitations in that landmarks are known (assumption). [5]

One adaptation to GraphSLAM is the Real-Time Appearance-Based Mapping (RTAB-MAP) algorithm. This implementation collects data from the various sensors and compares it to the estimations it has previously made for a map. By doing so, it conducts ”loop closures,” which allow the robot to confirm if it has previously perceived the current environment before. As the name suggests, one major benefit is this can be conducted at (near) real-time. As such, RTAB-MAP has been chosen for this project and has readily available libraries in ROS.

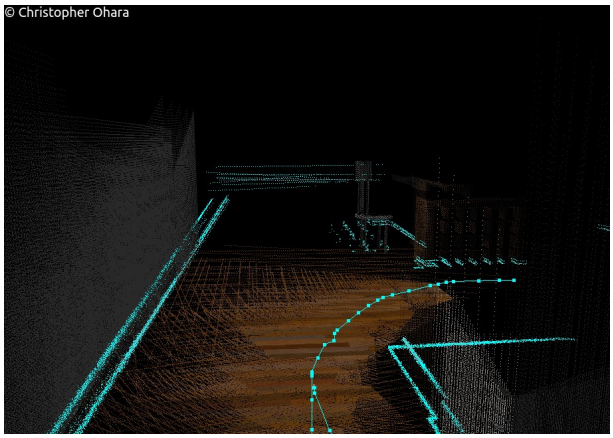


Fig. 2: Example of RTAB-MAP in the Kitchen World.

### 2.4 Previous Work

In a previous project, a custom robot was created to navigate an environment using a hi-res camera, odometry information, and a Hokuyo rangefinder. The goal was to autonomously navigate to a given goal while showing the point cloud distribution of particles (AMCL) while solving the ”kidnapped robot” problem. The robot from that project has made a return here with some slight (negligible) modifications to the design other than no longer being autonomous (navigation via teleoperation).

## 3 Scene and Robot Configuration

The robot was deployed in two different Gazebo environments (.world) for validation. The first environment was provided by Udacity to simulate a kitchen area, typical for a home-service robot (Roomba, iRobot). The second scene is custom made robotics laboratory, modeled similarly to one of the robotic laboratory environments at the Technische Universiteit Eindhoven (TU/e).

### 3.1 Robot Model Design

The robot is a two-wheeled, differential-drive robot. The Mk.II (ohara_bot from a previous project) design has a length of 0.4m, a width of 0.3m and height of 0.1m. It retains the two casters for stability. For sensor hardware, there is an upgraded RGBD camera and a Hokuyo rangefinder.

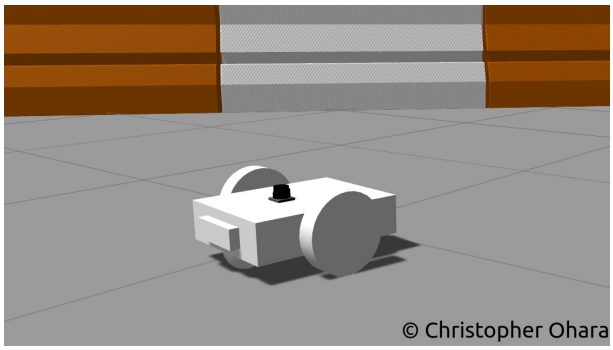


Fig. 3: Mk.II - ohara_bot, custom model in Gazebo from previous work

## 3.2 Scenes

### 3.2.1 Benchmark World

The benchmark world is a simulated kitchen/dining environment. It consists of two primary rooms and contains tables, chairs, as well as other common appliances. The room is rather small, making loop closures easier to achieve. Furthermore, RTAB-MAP requires environments to be rather heterogeneous in design, else the loop closure process will become confused on landmark identification. Nonetheless, the benchmark world has a good distribution of objects and does not have many wide-open spaces, ensuring loop closures can be completed with ease.

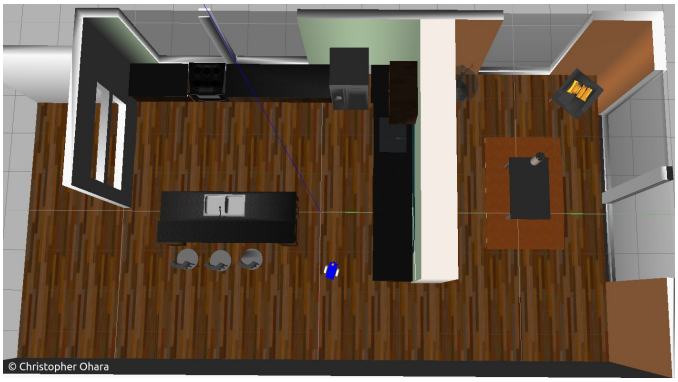


Fig. 4: Benchmark World - Kitchen/Dining Area, as shown in Gazebo.

### 3.2.2 Custom World

The custom-made personal world is a robotics laboratory, meant to be similar to the robotics environments in the *Control Systems Technology* and *Human-Interaction Technology* departments at TU/e. As such, there are several robots spread across the environment, including a PR2, a Robonaut, KUKA mobile bots, and some Parrot Bebop 2s. This is a relatively close representation of common robots in this environment (the only thing missing is an army of NAO robots since TU/e is a world leader in the RoboCup). It might seem as if the world is messy with random robot parts, carts, and boxes strewed across the place, but a visit to these labs will show that it reflects reality.


Fig. 5: Custom World - Robo-Lab Area, as shown in Gazebo.

## 3.3 Simulation Setup

The setup consisted of an Intel Core i5-7400T 2.4GHz processor with 12GB (SODIMM) DDR4 memory and an Intel integrated chipset. Due to the built-in graphics accelerator and lack of an Nvidia GPU, visualizations occasionally crash and are quite intensive for the overhead. This is a known issue and more of an inconvenience than limiting-factor. The build was compiled utilizing the ROS Kinetic distribution on Ubuntu 16.04 LST OS. Modeling and navigation were completed using Gazebo and RViz.

## 3.4 Achievements

The robot was able to successfully complete each map. While the benchmark world was rather simple, the custom world was not trivial in the least. Due to the openness of the world, an additional parameter was needed to be passed into RTAB-MAP to place constraints on the loop closures based on the standard error value. For both maps, a total of three passes was made to ensure proper data collection.

### 3.4.1 Benchmark World

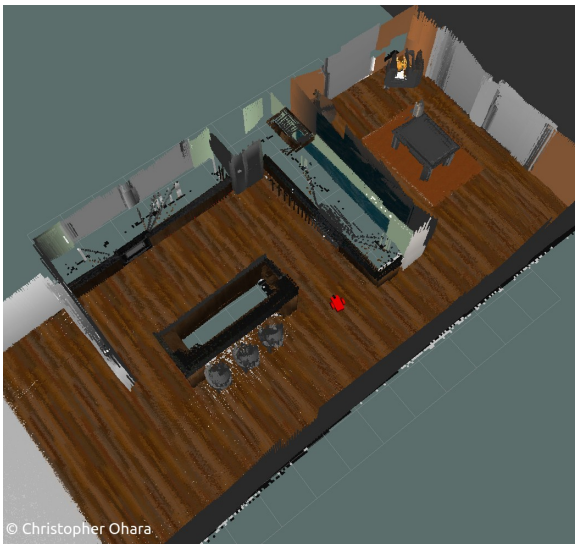


Fig. 6: Kitchen World, shown as completed in RViz.

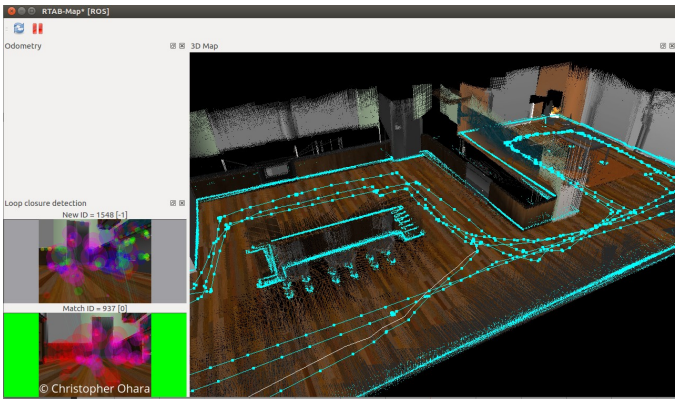


Fig. 7: Kitchen World, shown as completed with three passes in the 3D Map view for RTAB-MAP.

### 3.4.2 Custom World

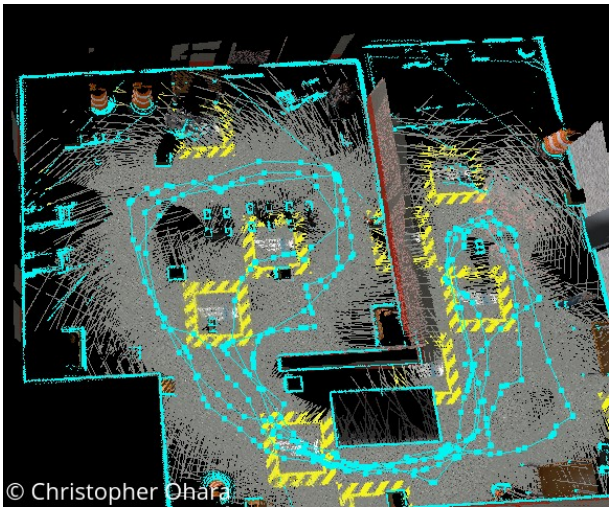


Fig. 8: Robo-Lab World, shown as completed in RViz.

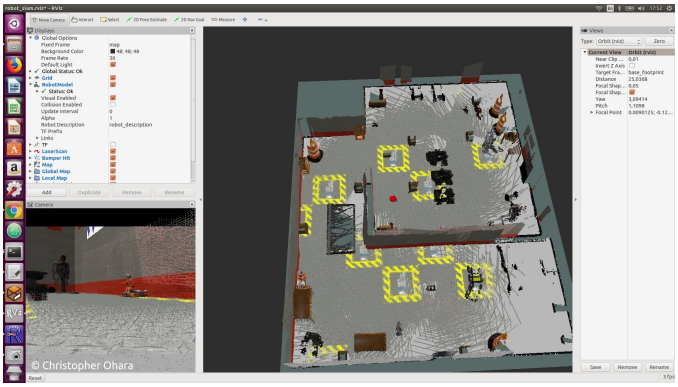


Fig. 9: Robo-Lab World, shown as completed with three passes in the 3D Map view for RTAB-MAP.

### 3.5 Transform Tree and RQT Graph

Below, the transform tree and RQT graph are given. These files are critical when it comes to debugging various errors in ROS. As can be seen, there are many connections (links) involved in the system architecture. The frames can also be manually plotted using RViz for a more intuitive representation, though it can still be difficult to locate where the issues occur.

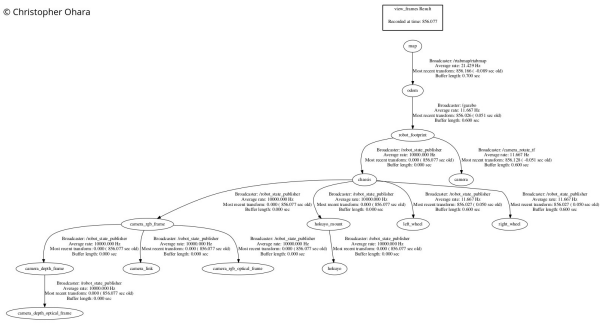


Fig. 10: Transform Tree for Robo-Lab World.

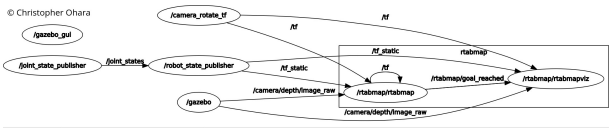


Fig. 11: RQT Graph for Robo-Lab World.

## 4 Results

As can be seen above in the achievements, both maps were filled out properly. Furthermore, the robot was able to meet all function and quality requirements based on the project specifications and criteria. Below, some additional images will be provided to illustrate differences that might not be self-evident by simply observing the completed tasks.

The first set of images shows the initial results of RViz and RTAB-MAP when the robot is initialized.

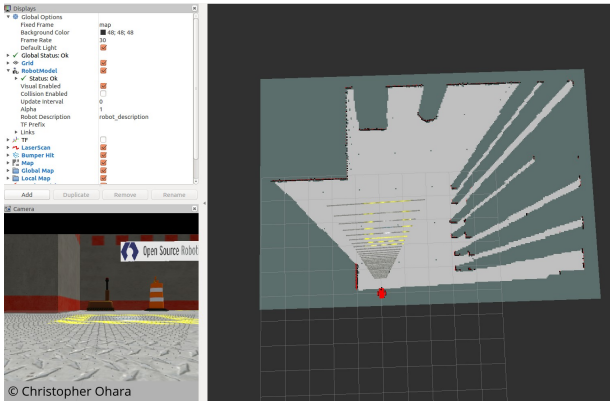


Fig. 12: Robo-Lab World upon initialization, as shown in RViz.

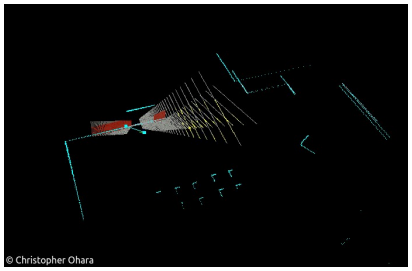


Fig. 13: Robo-Lab World upon initialization, as shown in RTAB-MAP.

The second set of images shows the difference between the RViz and RTAB-MAP 3D views as the experiment is being conducted. RViz is able to map the environment quicker than RTAB-MAP in the same amount of time. This is due to the higher overhead that RTAB-MAP has when conduction loop closures and identifying inter-depended landmarks.

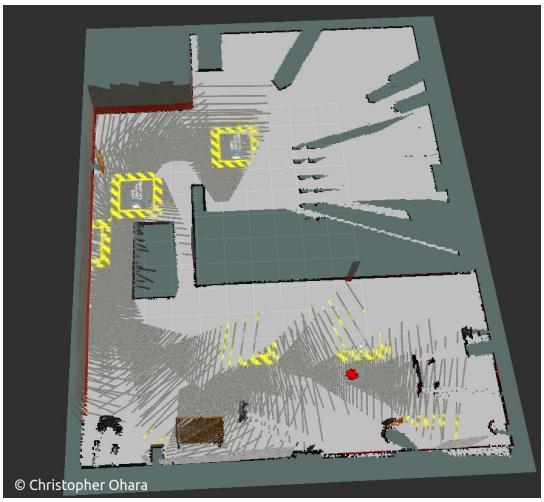


Fig. 14: Robo-Lab World with progress shown in RViz.

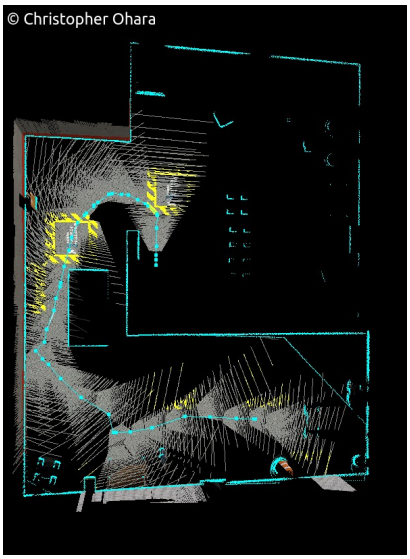


Fig. 15: Robo-Lab World with progress shown in RTAB-MAP.

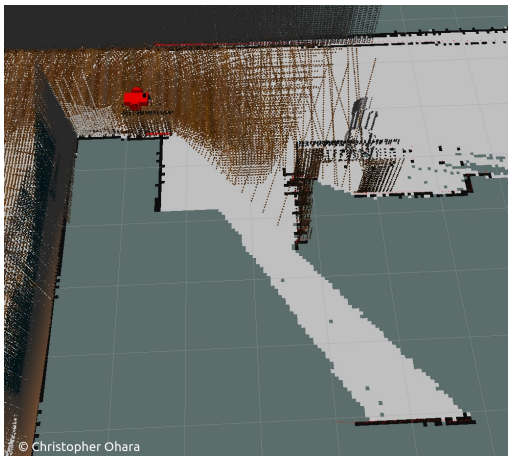


Fig. 16: Kitchen World with progress shown in RViz.

## 5 Discussion

Overall, the benchmark world was much easier to work with. It has the ideal dimensions and distribution of unique objects, allowing for loop closures to be completed with ease. The loop closure process is the primary reason behind performance differences in the project. The custom world was much more problematic, requiring several iterations of adding or moving objects and also requiring an additional parameter to included: *param name="RGBD/OptimizeMaxError" type="double" value="0.001"*

It is known that the multiple-pose formulation has a problem of having the inability to refine the map over multiple passes without dropping vehicle states. As such, the loop closure method limits the map to collecting restrictive geometric primitives.[6] The general process is when the robot has returned to a previous location after having discovered new terrain over a given amount of time. This is also appropriate for addressing the global localization problem, or a "kidnapped robot" scenario (such that the robot becomes moved to a new location or even stuck in a position with the odometer constantly updating the distance traveled through the robot has not moved). Therefore, solving the loop-closure detection problems simultaneously improves SLAM performance and enables additional capabilities to mobile robots. [7]

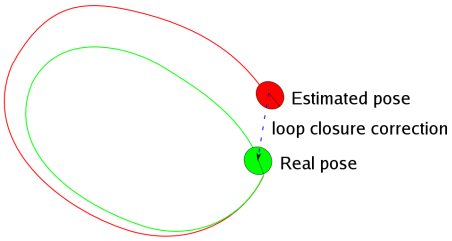


Fig. 17: Visualization of the loop closure process. http://cogrob.ensta-paristech.fr/loopclosure.html

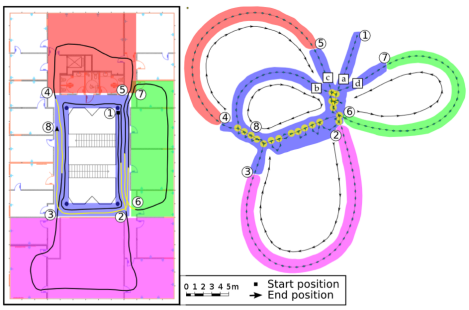


Fig. 18: Example of a topological SLAM map with loop closures. http://cogrob.ensta-paristech.fr/loopclosure.html

As stated above, several objects were added for environmental diversity, as well as a parameter was added. Various values were attempted for optimizing the maximum error (based on standard deviation of error), but the value

above was arrived at by trial-and-error (as the suggested value on the ROS Wiki is 0.5). Two images are provided below to demonstrate the loop hypotheses being rejected as compared to nominal behavior, as well as an in-progress image of successful closures with IDs.

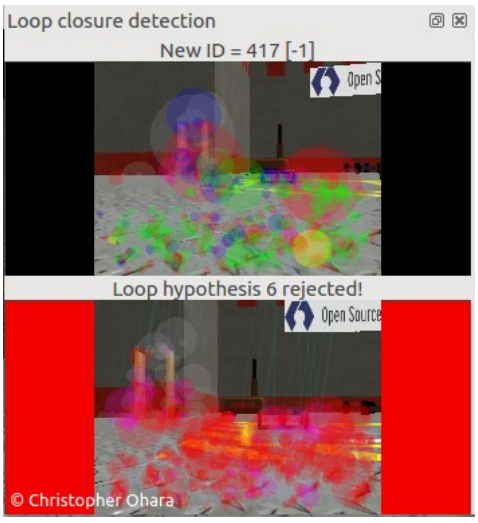


Fig. 19: Poor loop closure behavior due to homogeneous or open environments.

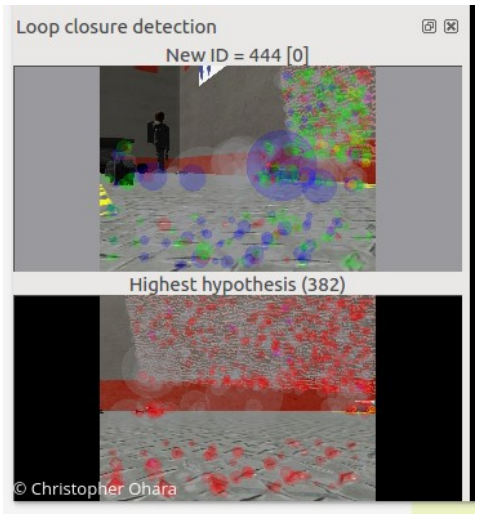


Fig. 20: Nominal loop closure behavior in ideal environments.

## 6 Conclusion / Future work

While the project was successfully completed, there is still much more to investigate. To begin with, this project was completed using a teleoperation script and manually driving around the map to collect data. A logical next step is to implement autonomous behavior where the robot completes the same tasks, but independently.

Future work will include employing these same tactics and libraries at the DAI-Labor (Distributed Artificial Intelligence Laboratory) at Technischen Universität Berlin. A similar project is being completed simultaneously, in which there

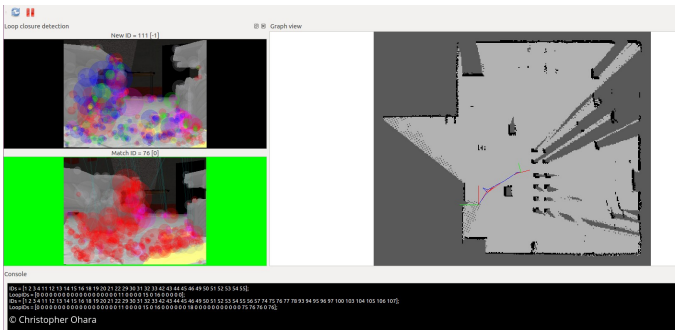


Fig. 21: RTAB-MAP GraphView with Map Closure ID while navigating.

a multi-agent system has shop-floor autonomous robots (small BettyBots) that move objects and packages around an Industry 4.0 Smart Factory environment. The results of this project will be directly beneficial to progress at DAI-Labor.

## References

[1] "Simultaneous localization and mapping: part I, https://ieeexplore.ieee.org/document/1638022

[2] *Fox, Dieter*, KLD-Sampling: Adaptive Particle Filters, papers.nips.cc/paper/1998-kld-sampling-adaptive-particle-filters.pdf.

[3] *Thrun, Sebastian and Burgard, Wolfram and Fox, Dieter*, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), 2005, ISBN (0262201623), The MIT Press.

[4] "ROS Wiki", ros.org: AMCL, wiki.ros.org/amcl, http://wiki.ros.org/amcl

[5] "FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem" , http://robots.stanford.edu/papers/montemerlo.fastslam-tr.pdf

[6] "SLAM- Loop Closing with Visually Salient Features" , http://www.robots.ox.ac.uk/~mobile/Papers/NewmanHoICRA05.pdf

[7] "Loop Closure Detection" , http://cogrob.ensta-paristech.fr/loopclosure.html