Simultaneous localisation and mapping with ROS

Haidyn McLeod

April 3, 2018

Abstract

As robots are exploring uncharted areas, they must deal with the unknown. Navigating the unknown can kidnap robots or have critical consequences. Simultaneous localisation and mapping (SLAM) is a powerful concept for mapping and reliably localise the robot in the unknown. This paper tests a ROS based graph-based SLAM multi-session mapping approach that can handle diverse and large environments with under 0.9m error in over 400m of explored environments.

1 Introduction

Localising a robot in an environment is critical for navigating safety. Localisation uses sensory feedback against a mapped background to estimate the robots pose. However, often the environment is unexplored, or the surroundings change resulting in a non-existent or incorrect map. Mapping an unknown environment while simultaneously keeping track of the robot's location within it is an essential area of robotic mapping and navigation. Simultaneous localization and mapping or SLAM is a way of constructing or updating a map of an unknown environment while simultaneously keeping track of the robots location within it [1]. In SLAM, the robot uses sensor measurements and movement controls and constructs a map of the environment while simultaneously localising itself relative to this map.

SLAM is a very powerful concept in all forms of robotics that is tailored to the hardware available. It can be found in self-driving cars, unmanned aerial vehicles and autonomous underwater vehicles [1].

In this paper, I will discuss using the real-time appearance-based mapping or RTAB-map method with the open-source robot operating system (ROS) to perform SLAM. This will be achieved with a wheeled mobile robot platform incorporating an onboard laser and Kinect RGBD sensors. The robot will be tested in two environments using Gazebo. The accompanying code can be downloaded from GitHub at https://github.com/Heych88/slam_project.

2 Background

2.1 SLAM

The SLAM problem has two key features. The first is its form, either online SLAM or full SLAM. Online SLAM estimates the system variables or posterior that occur at time $\boldsymbol{t}$ only. That is, every time data is acquired it attempts to resolve it into a probability distribution. This problem can be modeled with the probability equation $p(x_t, m|z_{1:t}, u_{1:t})$ where we solve the posterior represented by the instantaneous pose x_t and the map $\mathbf{m}$ given the measurements $z_{1:t}$ and controls $u_{1:t}$).

Full SLAM estimates all the system variables that occur throughout the robot travel time. That is, all the data is memorised and after all the data is accumulated, the map is inferred. This problem can be modeled with the probability equation $p(x_{1:t}, m|z_{1:t}, u_{1:t})$ where we solve the posterior represented by the robot's trajectory $x_{1:t}$ and the map $\mathbf{m}$ given the measurements $\mathbf{z1:}\boldsymbol{t}$ and controls $\mathbf{u1:}\boldsymbol{t}$ [2] [3].

The second feature relates to its nature. SLAM problems have a continuous and a discrete element. The continuous component is that a robot continuously collects odometry information to estimate the robot poses and continuously senses the environment to estimate the location of the object or landmark. Thus, both robots pose and object location is continuous aspects of the SLAM problem [3].

The discrete component of the SLAM algorithms has to identify if a relation exists between any newly detected objects and previously detected ones. That is, has the robot been in this same location before. Correspondence or global loop closure [3] is the discrete relation between objects. For loop closure detection, a bag-of-words approach can be used. The bag-of-words uses distinct features such as lines and corners in an image. From these features, descriptors are produced, clustered and then quantised to a visual dictionary. When a new word matches a previous word, the two words associated images are compared and when their words reach a defined threshold, a loop closure is detected [3] [4].

SLAM algorithms' use either of the forms and their approaches generally resides into five main categories, extended Kalman filter, extended information filter, sparse extended information filter, GraphSLAM and FastSLAM [5].

The information presented in this section is only an overview of SLAM and some of its implementations. To obtain more detailed information, review reference [2].

2.1.1 SLAM with Extended Kalman Filter

The extended Kalman filter or EKF SLAM algorithm is an online SLAM form that makes Gaussian noise assumptions for robot motion and perception to map landmark points using maximum likelihood to associate the model's data. EKF-SLAM is subject to approximations, limitations and can be intolerant to errors [2].

2.1.2 SLAM with Extended Information Filter

The extended information filter or EIF SLAM performs the same function as the EKF but uses an information matrix and information vector rather than the Gaussian covariance and mean respectively. The information matrix is the inverse of the covariance matrix, and the information vector is the inverse of the covariance matrix multiplied by the mean [6].

2.1.3 SLAM with Sparse Extended Information Filter

The sparse extended information filter or SEIF provides a solution to the online SLAM problem. It performs the same operation as the EKF algorithm but maintains an information representation of all knowledge. That is, it maintains a posterior over the present robot's pose and map, but represents the posterior in a normalised informational matrix that is naturally sparse. This sparseness make the SEIF more computationally efficient [2] [7].

2.1.4 GraphSLAM

GraphSLAM solves the full SLAM problem where it uses the robots constraints to represent relationships between the robot's poses and the environment. It then tries to optimises all of these constraints to create a maximum likelihood map and a corresponding set of robot poses given the data [2] [5]. Just like the SEIF, GraphSLAM maintains an information representation of all the data and uses nodes as poses, links as odometry and loop closure transformations [4].

2.1.5 FastSLAM

FastSLAM uses a particle filter approach along with a low dimensional extended Kalman filter approach solving the SLAM problem. FastSLAM has the benefit of addressing both the full SLAM and the online SLAM problems. It able to estimate the location of all environment features independent of each other, unlike the SLAMs mentioned above' which use a single Gaussian distribution, FastSLAM has computational benefits over the EKF methods as data associations decisions can be made on a per-particle basis [2].

3 Environment and robot configuration

3.1 Environment

The robot was tested in two environments, a small feature rich kitchen dinner and a large feature rich but consistently similar features building floor plan.

3.1.1 Kitchen diner

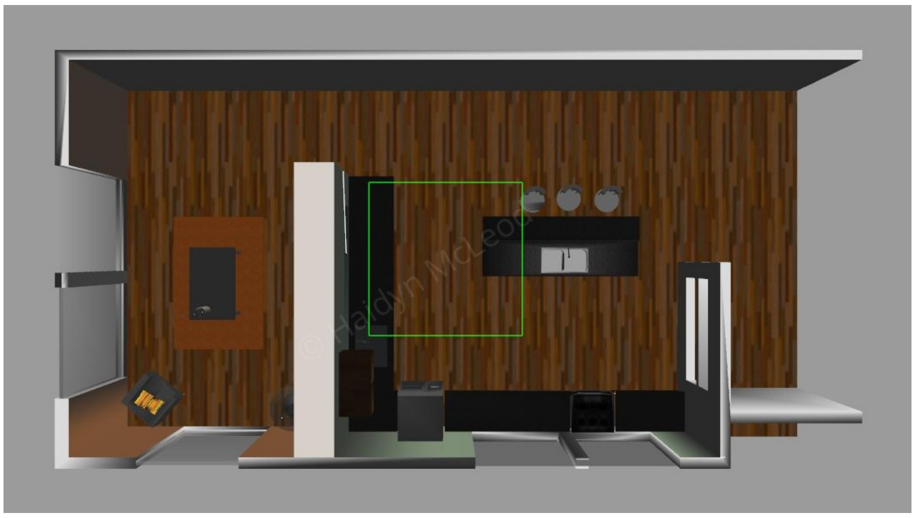


Figure 1: Kitchen diner environment.

The objective of the kitchen diner environment, Figure 1, is to simulate a small home environment. A small home environment can test SLAM over small robot travel distances, thus minimising odometry error, provides commonly observed obstacles such as tables and chairs and providing frequent loop closure opportunities for the algorithm. The environment has not been modified from the supplied Gazebo model.

3.1.2 Office floor plan

The office floor plan environment, Figure 2, provides a large feature dense environment. The office space provides a 46 x 60m area for mapping and over large travel distances. With the possibility of many paths for travel, loop closure opportunities can be infrequent. The environment has not been modified from the supplied Gazebo model.

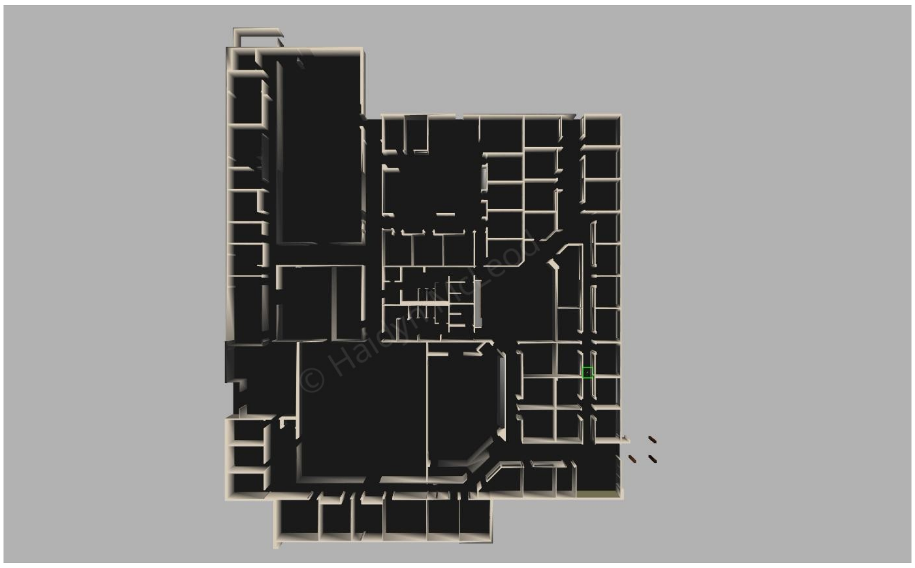


Figure 2: Office building floor plan or Willow Garage office environment.

3.2 Robot configuration

The SLAM robot model is a differential-drive robot consisting of two independent drive wheels with caster wheels at each end for balance. Perception for SLAM is through a *Hokuyo* 180°2D laser scanner, positioned at the front of the robot. The positioning of the laser scanner at mid-height of the robot enables detection of low lying objects. To perform loop closure detection, an RGBD Kinect camera is required. The RGBD camera can also provide a 3D map environment for obstacle detection. The Kinect is positioned on top of the robot above the front caster wheel. This provides excellent depth perception of the image loop closure and 3D mapping while being able to observe objects in proximity to the front of the robot. The Kinect RGB images are used for the appearance-based loop closure detection while the depth images are used to find the 3D position of the visual words. The Hokuyo and Kinect point cloud is used for map visualisation [4].

The model's dimensions are 450 x 350 x 250mm (LxWxH), weight is 30kg, and wheel radius is 200mm. Fig 3 shows the final model design and the model's transform frames can be observed in Appendix A.1.

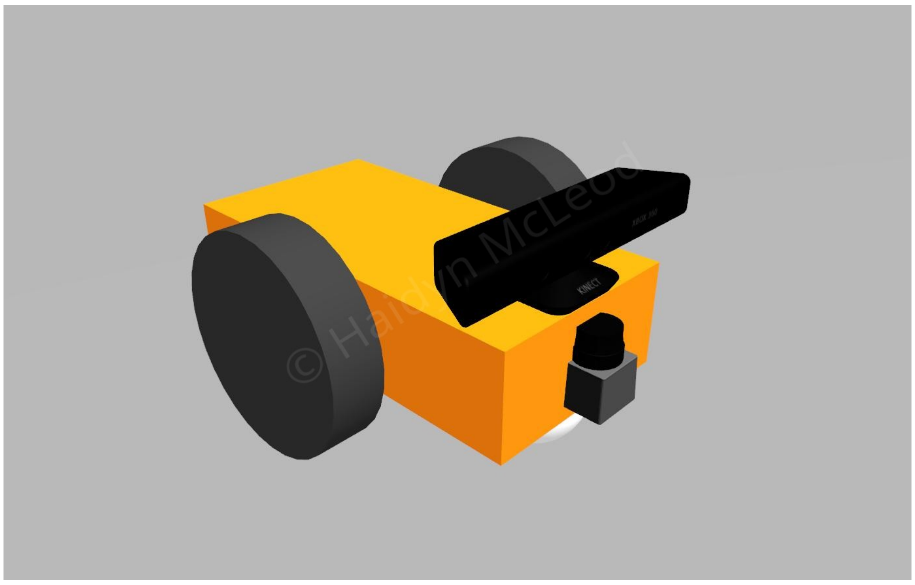


Figure 3: Slam robot model used in testing.

4 SLAM with ROS

There are many packages available in ROS that perform SLAM. The SLAM approach used in this paper is called real-time appearance-based mapping, or RTAB-Map. RTAB-map is a GraphSLAM approach based on a global Bayesian loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous location or a new location. RTAB-Map limits the number of past locations, or nodes, used for loop closure detection and graph optimisation, to restrict computational burden as the map increases over large areas [8] [9]. The final ROS packages and system configuration when mapping the first time can be observed in Appendix B.1 and when localising or multi-session mapping in Appendix B.2.

4.1 Parameters

The RTAB-Map ROS configuration uses the Kinect, odometry and the Hokuyo laser scanner for mapping. Parameter tuning RTAB-Map requires no major adjustment, outside of configuring to the sensors used, to the default parameters. Default parameters enable the supplied code base to function across non-GPU hardware platforms. The tunning parameters available can be observed in RTAB-Map's `Parameters.h` file [10].

The parameters adjusted from the default and used in the simulation are defined in Table 1.

4.2 Navigation of the mapped environment

After SLAM has mapped the environments', the estimated maps were tested first through localising the kidnapped robot in the predicted map, secondly navigating to a goal position and thirdly,

Table 1: RTAB-Map ROS parameters used for simulation.

Parameter	Value	Description
subscribe_depth	true	Subscribe to depth images.
subscribe_scan	true	Subscribe to laser scanner data.
cloud_max_depth	4.0	Max cloud depths added to cloud map.
gen_scan_max_depth	7.5	Max depth of the laser scans generated.
Rtabmap/DetectionRate	1	Rate (Hz) to update nodes in the map.
Reg/Force3DoF	true	3 degrees-of-freedom transform (x,y and yaw).
Kp/DetectorStrategy	0	Loop Closure Detection (0=SURF).
Kp/MaxFeatures	400	Max words per image (bag-of-words).
SURF/HessianThreshold	100	Threshold for key-point detector (SURF).
Reg/Strategy	0	Loop closure (0=Visual).
Vis/MinInliers	20	Min visual inlier's to accept loop closure.
Mem/NotLinkedNodesKept	false	Save data when robot is not moving.

actively performing SLAM while navigating to the goal position.

Localising the kidnapped robot with the database map is achieved through RTAB map directly through the mapping launch file. Adding the parameter `Mem/IncrementalMemory` and setting it to false, as well as removing `Mem/NotLinkedNodesKept`. The launch file `localisation.launch` performs the localisation of the kidnapped robot model.

The navigation planner used RTAB-Map for localisation, `global_planner` for the global planning and `eband_local_planner` as the local planner. Further details of the navigation parameters can be found in my previous paper *ROS Localisation and Navigation using Gazebo* [11].

5 Results

5.1 Kitchen diner

The final robot and mapping calculations are estimated from 379 robot poses over a path length of 104m. The bag-of-words dictionary produced 17331 unique words. The predicted map was produced from 378 neighbors (N), 0 neighbors merged (NM), 91 global loop closures (GL), 664 local loop closure by space (LS), 0 local loop closure by time (LT), 0 user loop closure (U) and 0 prior links (P). The optimisation of the map was performed in 49.9129 ms over 100 iterations, produced a maximum error of 0.251306 m and created a database size of 309MB.

Figures 4 to 6 show the SLAM outputs of the kitchen diner environment after driving the robot around. Figure 6 shows the 3D map of the environment which can be used for obstacle detection.

Figure 7 show an example of loop closure. Figure A is from position 92 with a pose xyz of -1.4878, -1.6655 and 0.1083 respectively with a rotation rpy of 1.3188e-5, -0.0006 and 1.5986. Figure B is from position 254 with a pose xyz of -1.3320, -1.4256 and 0.1083 respectively with a rotation rpy of 1.4055e-6, -0.0014 and 1.5491. The optimised loop closure between the images produced a 4.0563% error between closures.

To observe the generated map and the influence RTAB-Map can have localising a kidnapped robot the robot was driven from the origin until the robot was localised. Over three runs, the robot was able to localise itself under 13 seconds and 3 m. This can be observed in the video https://youtu.be/uhsmxAyt780. Figure 8 shows the robot navigating the localised map. The kitchen diner database can be downloaded from

https://drive.google.com/open?id=1N4_2hQNAKHkBCLbugkAoKEWO-hkwqobJ.

5.2 Office floor plan

The office floor plan was partially mapped over two runs. In the first map, Figure 9, followed the hallways around the office perimeter and produced 675 robot poses over a path length of 230m. The bag-of-words dictionary produced 12296 unique words. The estimated map was calculated from 674 for N, 1 for GL, 686 for LS and zero for the NM, LT, U and P. The optimisation of the map was performed in 83.899 ms over 100 iterations, produced a maximum error of 0.86388 m and

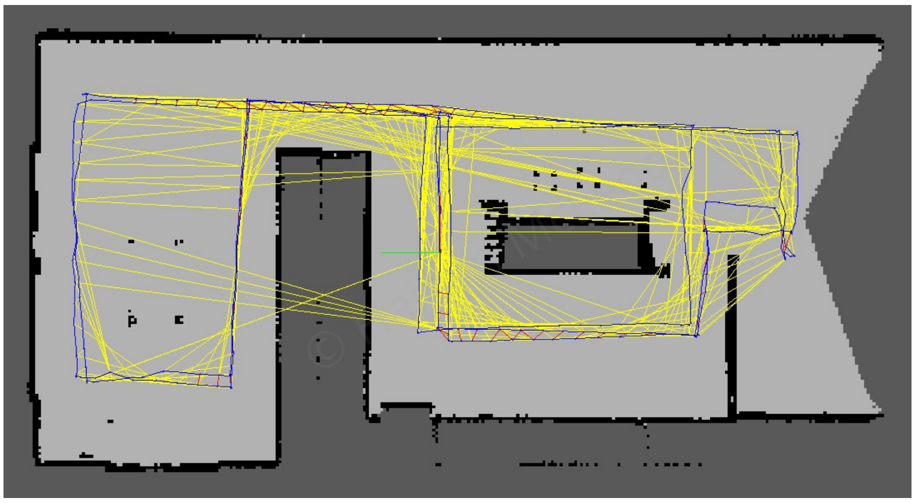


Figure 4: 2D map of the kitchen diner environment with SLAM updated iterations and the path of your robot. The blue line indicates neighbour links between nodes the blue dot are the nodes, yellow is local loop closure, global loop closure between neighbours is red.

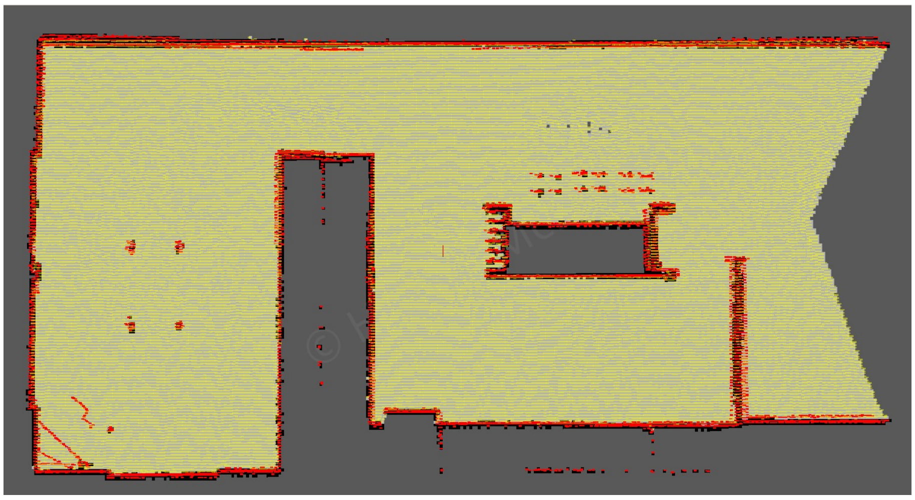


Figure 5: Occupancy grid used to produce the 2D map. Grey is unexplored, yellow is no obstacle explored, and red indicates an obstacle.

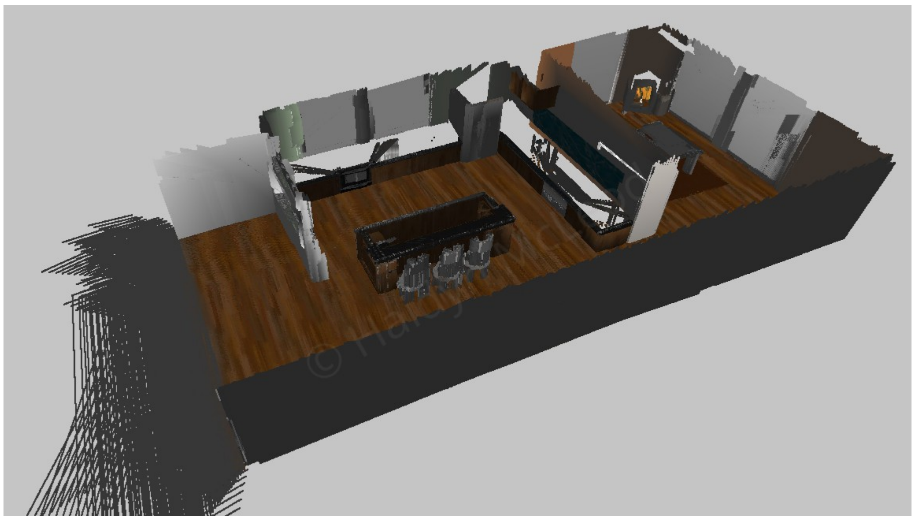


Figure 6: 3D map of the kitchen diner environment.

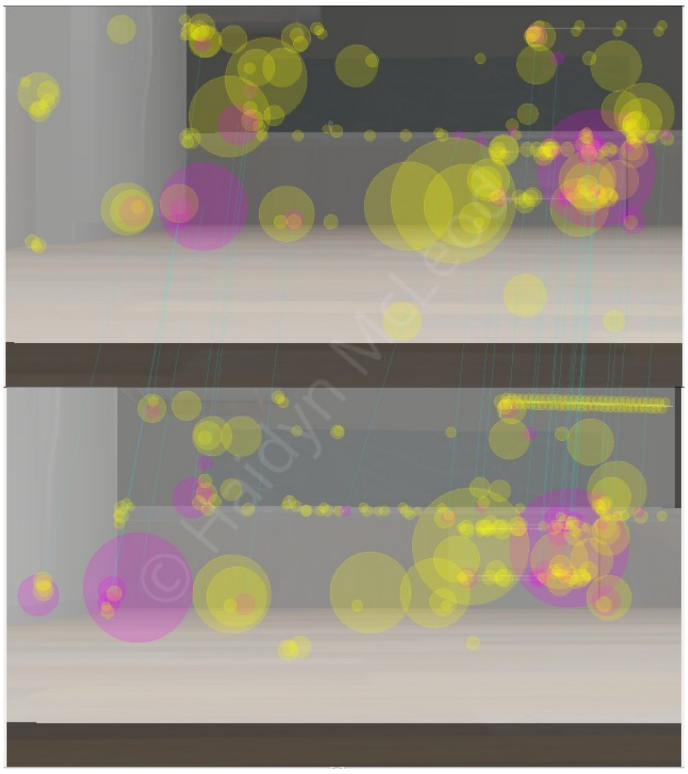


Figure 7: Depth and RGB combined images for loop closure detection between similar frames during the mapping process. A position 92, B position 254. The yellow bubbles indicate image features or words in the bag-of-words. Pink notes common features between two images and is used to create neighbouring links and loop closures. The light blue lines indicate the links between the same feature.

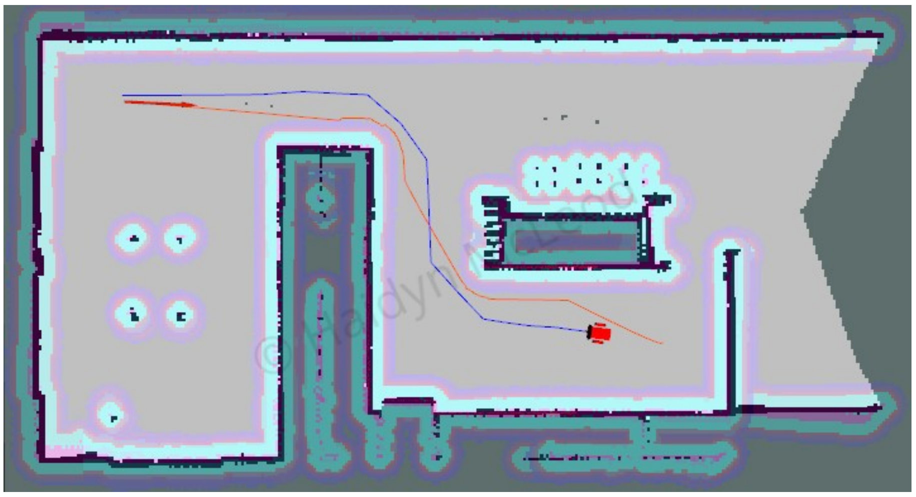


Figure 8: Robot navigating to a goal position after the kidnapped robot has localised itself to the map. Blue indicates the `EBandPlannerROS/global_plan` and red is the `GlobalPlanner/plan`. The transparent blue indicates the obstacles inflation radius of the global cost map.

created a database size of 582MB. The office floor plan map one database can be downloaded from https://drive.google.com/open?id=1N4_2hQNAKHkBCLbugkAoKEWO-hkwqobJ.

Figures 9 show the SLAM output after the first run mapping the environment.

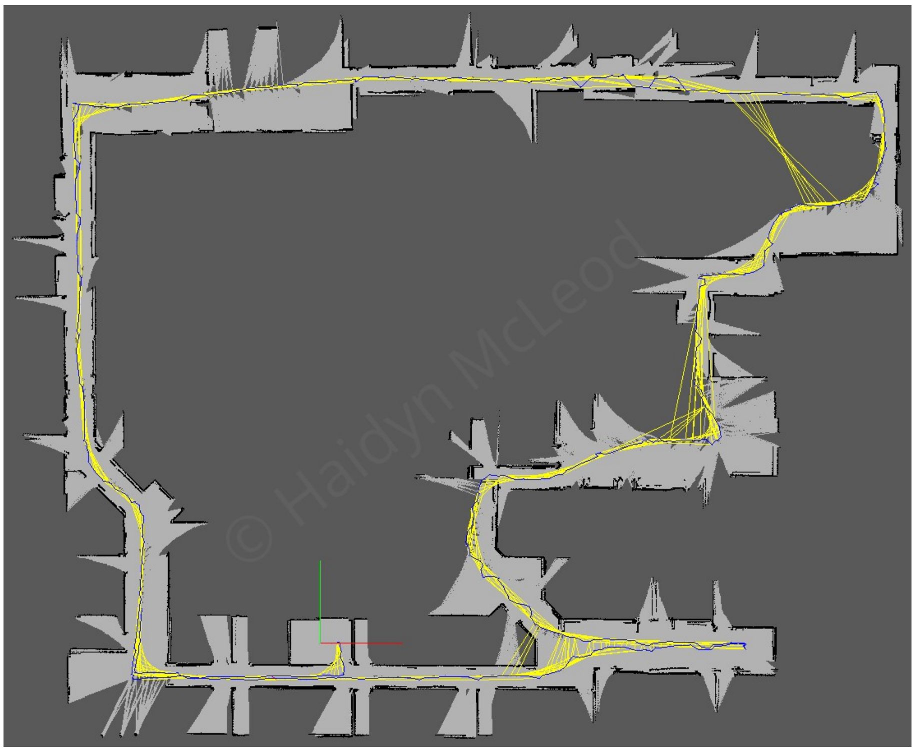


Figure 9: 2D partial map of the office environment with SLAM updated iterations and the path of your robot. The blue line indicates neighbour links between nodes the blue dot are the nodes, yellow is local loop closure, global loop closure between neighbours is red.

The second mapping of the office environment was performed in conjunction with the navigational goal test. The robot was spawned at the origin, and a goal was placed at the top right of map 1. The second map extended the first producing a total of 1419 robot poses over a path length of 403m. The bag-of-words dictionary with 23477 words. The estimated map was calculated from 1415 for N, 61 for GL, 1350 for LS and zero for the NM, LT, U and P. The optimisation of the map was performed in 198.218 ms over 100 iterations, produced a maximum error of 0.92634 m and created a database size of 1257MB. The office floor plan map two database can be downloaded from https://drive.google.com/open?id=1N4_2hQNAKHkBCLbugkAoKEWO-hkwqobJ.

Figures 10 to 12 show the SLAM outputs after extending map 1 with the second map.

Figure 12 show a loop closure. Figure A is from position 691 with a pose xyz of 11.3425, -1.88476 and 0.1 respectively with a rotation rpy of -1.99918e-06, 3.03629e-06 and 2.00904. Figure B is from position 90 with a pose xyz of -11.5068, -1.99777 and 0.0999994 respectively with a rotation rpy of -1.36124e-05, -0.00332582 and 1.5112. The optimised loop closure between the images produced 11.387% error.

6 Discussion

During the testing of various environments and sensor configurations, it was observed that the sensor positioning, in particular, the laser sensor, had an effect not only on obstacle detection but also localisation when `Reg/Strategy` is set to 1 or 2. This is because tables and chairs are observed as small detail-less points when the laser is close to the ground. When it is placed higher, it can detect more details in objects. The tradeoff is that low lying obstacles will not be identified causing potential collisions. To avoid this, If the observed environment includes very low lying obstacles that can restrict movement, the laser scanner can be inverted and lowered closer to the ground surface for detection of these obstacles at the expense of potential false positives and low floating obstacles.

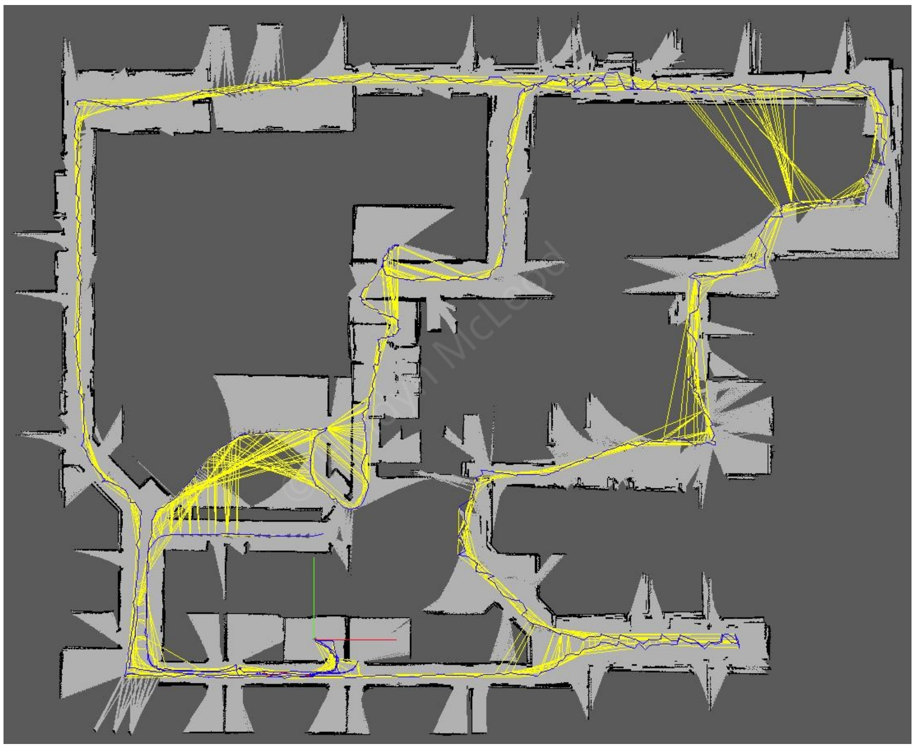


Figure 10: 2D map of the office environment with SLAM updated iterations and the path of your robot. The blue line indicates neighbour links between nodes the blue dot are the nodes, yellow is local loop closure, global loop closure between neighbours is red.

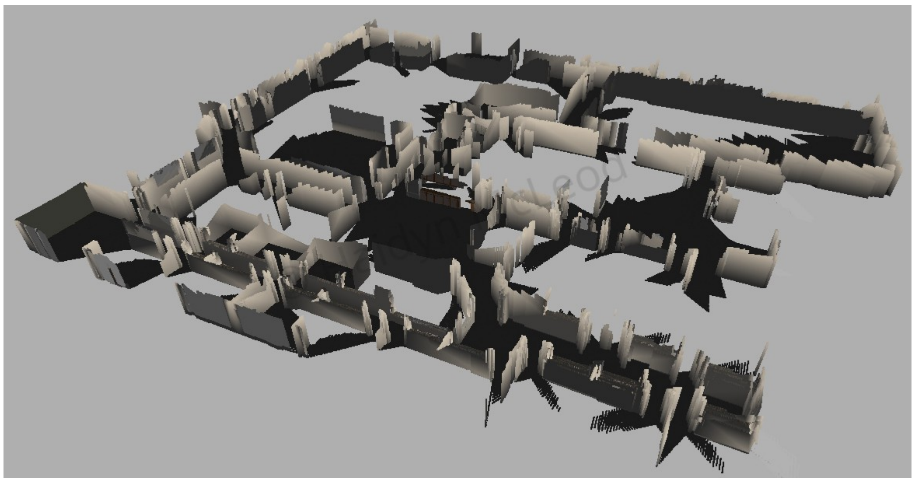


Figure 11: 3D map of the office environment after driving the robot around while performing SLAM.

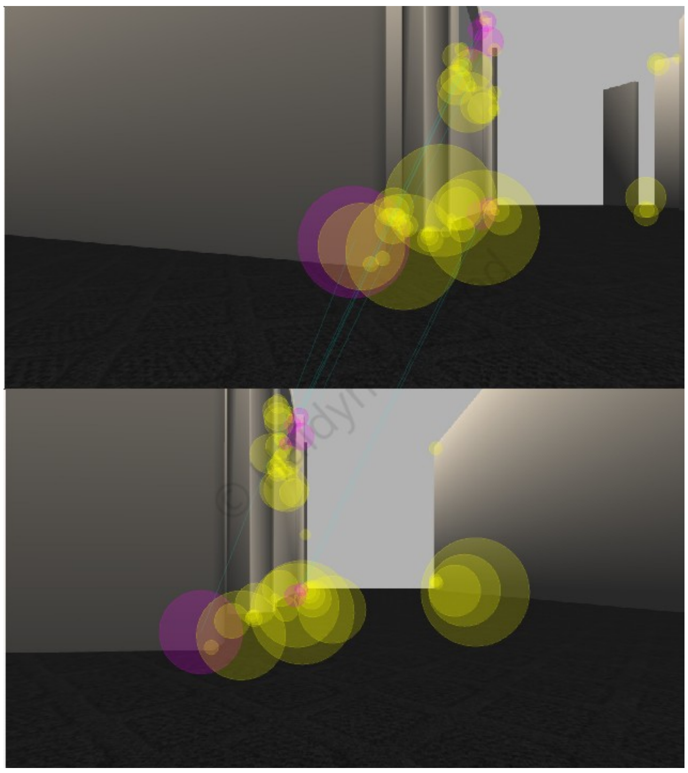


Figure 12: RGB image of the loop closure detection between similar frames during the mapping process. A position 691, B position 90. The yellow bubbles indicate image features or words in the bag-of-words. The pink indicates common features between two images and is used to create neighbouring links and loop closures. The light blue lines indicate the links between the same feature.

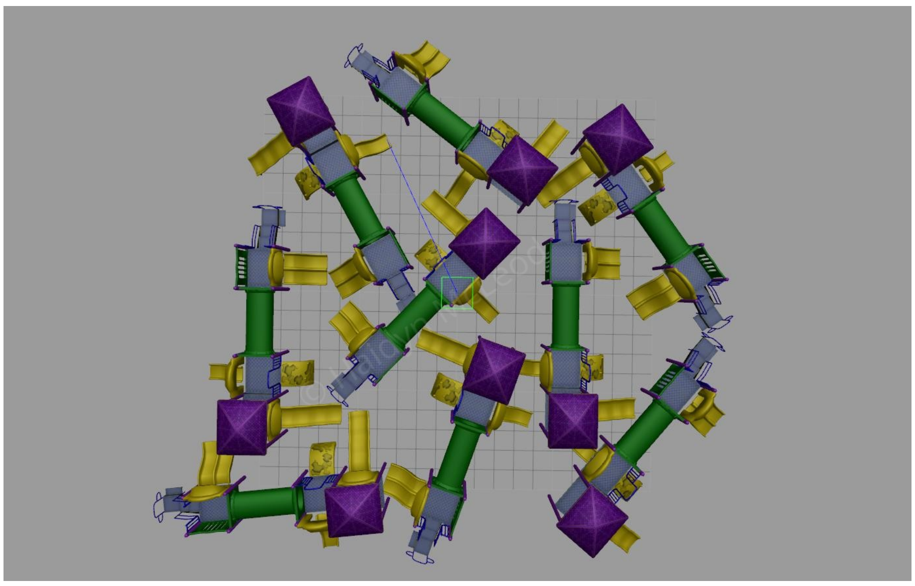


Figure 13: Playground challenge environment with identical featured objects.

A third challenging environment was used for tested which provided no correct map of the environment. Figure 13 shows a playground environment with nine identical models. The loop closure inlier's was raised to 100, the laser scanner parameter, `gen_scan_max_depth` was increased to 10m and `Reg/Strategy` was set to 2. Mapping this environment correctly required parameter fine-tuning and a predefined mapping path that always included laser and vision of two independent models. The environment could be mapped easier with the addition of other models.

RTAB-Map provides a robust, feature-rich and adaptable solution to the SLAM problem. Its ability to limit real-time optimisation to nodes in working memory offers an excellent approach to the full SLAM problem. Overall, RTAB-Map works well in all the tested environments. The provided RTAB-Map tools provide easy visualisation, fine-tuning of parameters and adjustments to the estimated map. The ability to either delete or produce multi-session mapping is beneficial to robots' with hardware limitations or time constraints.

7 Conclusion / Future Work

The paper presented a robust approach to the SLAM problem with ROS. RTAB-Map provides various features that were tested in two different environments. Although SLAM is a complex concept that can be solved with many different algorithms, RTAB-Map graph-based meets large-scale and multi-session needs with less than one-meter error over long distances. It has been shown that with very few parameter adjustments, robot's can map and localise themselves during navigation to a goal location through unmapped areas.

Future extensions of this work could include dynamic obstacles in the test environment. Further fine tuning to work with more complex situations such as the Figure 13. The approaches taken are intended to only show the capabilities and limitations of the RTAB-Map package between different environments.

References

[1] "Simultaneous localization and mapping," in *https://en.wikipedia.org/wiki/Simultaneous_ localization_ and_ mapping*, Wikipedia, March 2018.

[2] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics.* MIT press, 2005.

[3] Udacity, "Grid-based fast-slam," in *https://www.udacity.com/*, Udacity, February 2018.

[4] F. M. Mathieu Labbe´, "Online global loop closure detection for large-scale multi-session graph-based slam," IEEE, 2014.

[5] Udacity, "Introduction to mapping and slam," in *https://www.udacity.com/*, Udacity, February 2018.

[6] C. Stachniss, "Slam course - 07 - extended information filter," in *https://youtu.be/wD6BGkGOW4A?list=PLgnQpQtFTOGQrZ4O5QzbIHgl3b1JHimN_*, Youtube, November 2013.

[7] C. Stachniss, "Slam course - 08 - sparse extended information filter - part 1," in *https://www.youtube.com/watch?v=iQPvH6WgFvM*, Youtube, November 2013.

[8] M. Labbe and F. Michaud, "Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2661–2666, Sept 2014.

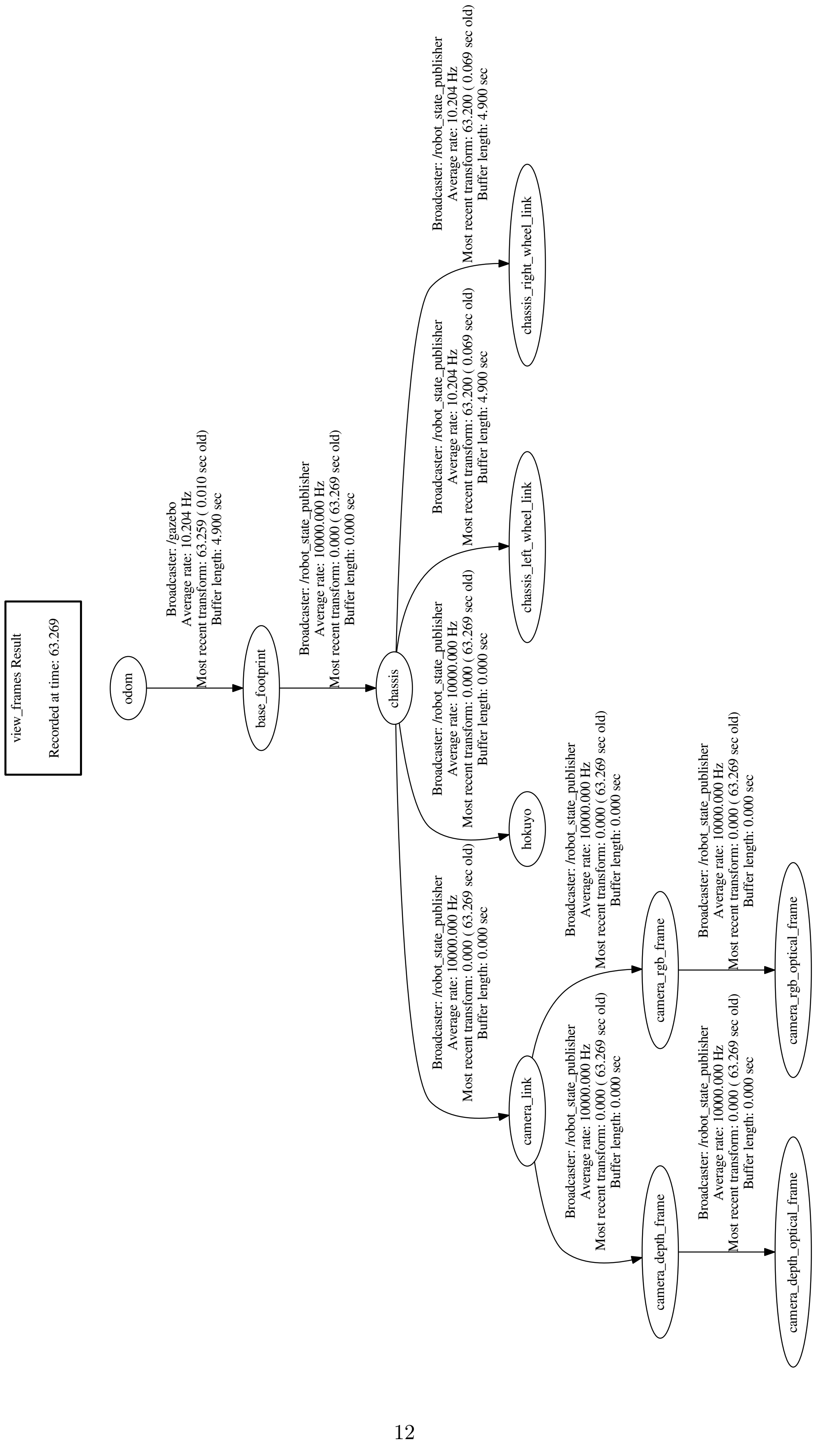
[9] M. Labbe, "rtabmap," in *http://wiki.ros.org/rtabmap*, ROS_wiki, February 2015.

[10] m. IntRoLab, "rtabmap/corelib/include/rtabmap/core/parameters.h," in *https://github.com/introlab/rtabmap/blob/master/corelib/include/rtabmap/core/Parameters.h*, GitHub, March 2018.

[11] H. McLeod, "Ros localisation and navigation using gazebo," in *https://docs.wixstatic.com/ugd/850e97_ 94ec4844112e483d91dc27f7e0604d7d.pdf*, Haidyn McLeod, March 2018.

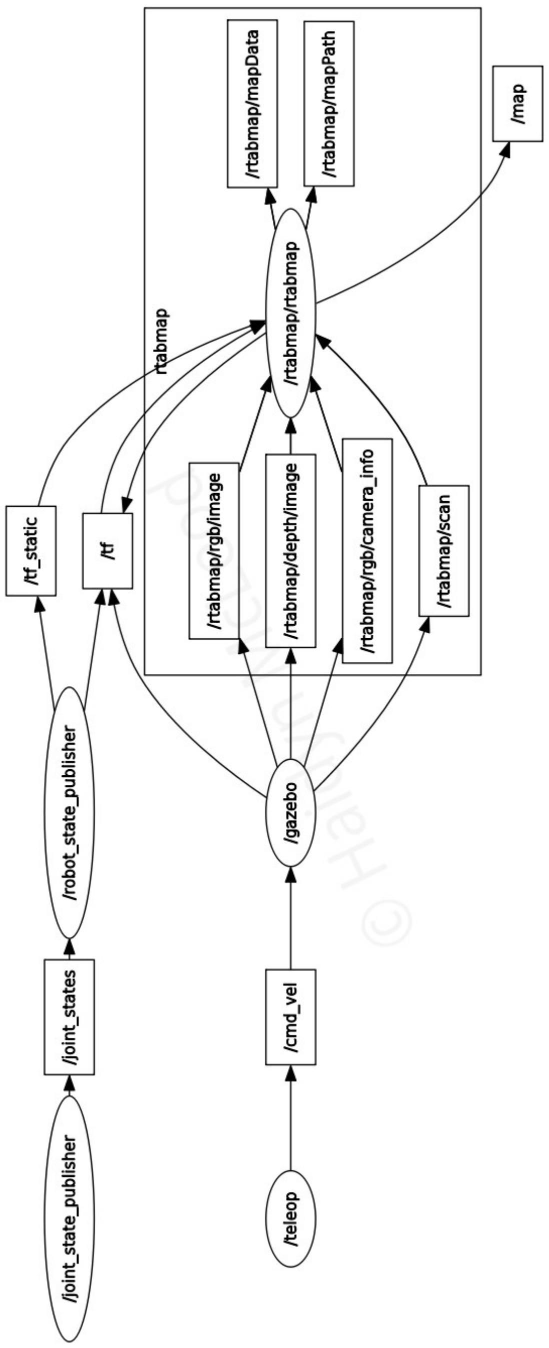
A Appendix

A.1 Robot model transform frames

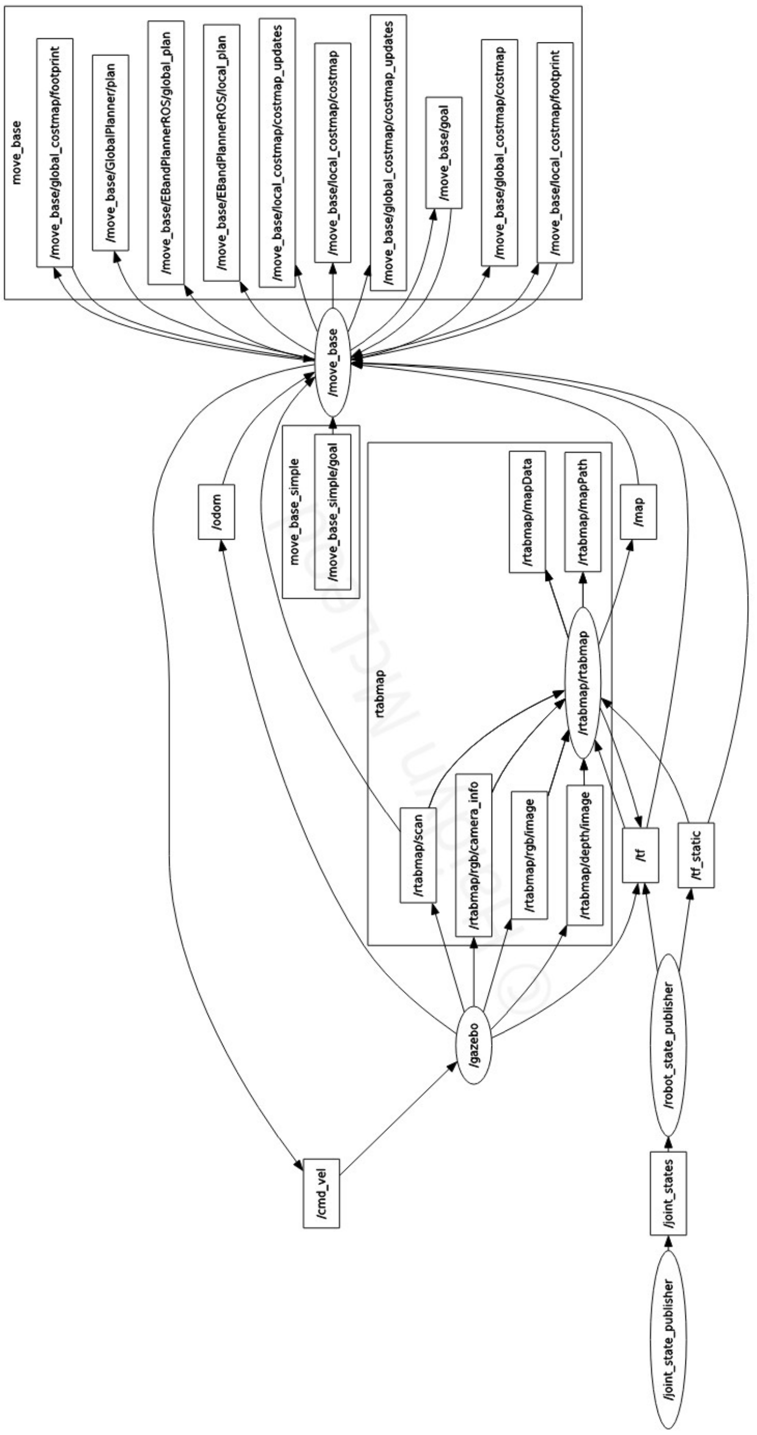

view_frames Result
Recorded at time: 63.269
odom
Broadcaster: /gazebo
Average rate: 10.204 Hz
Most recent transform: 63.259 (0.010 sec old)
Buffer length: 4.900 sec
base_footprint
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (63.269 sec old)
Buffer length: 0.000 sec
chassis
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 63.200 (0.069 sec old)
Buffer length: 4.900 sec
chassis_right_wheel_link
Broadcaster: /robot_state_publisher
Average rate: 10.204 Hz
Most recent transform: 63.200 (0.069 sec old)
Buffer length: 4.900 sec
chassis_left_wheel_link
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (63.269 sec old)
Buffer length: 0.000 sec
hokuyo
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (63.269 sec old)
Buffer length: 0.000 sec
camera_link
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (63.269 sec old)
Buffer length: 0.000 sec
camera_rgb_frame
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (63.269 sec old)
Buffer length: 0.000 sec
camera_rgb_optical_frame
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (63.269 sec old)
Buffer length: 0.000 sec
camera_depth_frame
Broadcaster: /robot_state_publisher
Average rate: 10000.000 Hz
Most recent transform: 0.000 (63.269 sec old)
Buffer length: 0.000 sec
camera_depth_optical_frame

B ROS configuration graph

B.1 ROS configuration graph for new SLAM environment.



B.2 ROS configuration graph for multi-session SLAM.


move_base
/move_base/global_costmap/footprint
/move_base/GlobalPlanner/plan
/move_base/EBandPlannerROS/global_plan
/move_base/EBandPlannerROS/local_plan
/move_base/local_costmap/costmap_updates
/move_base/local_costmap/costmap
/move_base/global_costmap/costmap_updates
/move_base/goal
/move_base/global_costmap/costmap
/move_base/local_costmap/footprint
/move_base
/odom
move_base_simple
/move_base_simple/goal
rtabmap
/rtabmap/mapData
/rtabmap/mapPath
/map
/rtabmap/rtabmap
/rtabmap/scan
/rtabmap/rgb/camera_info
/rtabmap/rgb/image
/rtabmap/depth/image
/tf
/tf_static
/gazebo
/robot_state_publisher
/cmd_vel
/joint_states
/joint_state_publisher