MULTI-ROBOT EXPLORATION OF UNKNOWN ENVIRONMENTS

Lillian D. Goodwin, Scott B. Nokleby
Mechatronic and Robotic Systems Laboratory
Ontario Tech University
Oshawa, ON
Email: Lillian.Goodwin@ontariotechu.ca; Scott.Nokleby@ontariotechu.ca

ABSTRACT

Frontier navigation is a viable solution for exploring unknown environments in robotic teams. In this work, an implementation of frontier navigation in both simulated and real-world scenarios is investigated. Rapidly exploring randomized trees are utilized for frontier definition and map merging for global exploration. The architecture of choice is a centralized method, assuming connectivity to a central executive. The results show that the implementation works, but there is an opportunity to make the exploration more efficient to make better use of the robots.

Keywords: frontier navigation; multi-robot exploration; Robot Operating System (ROS).

EXPLORATION MULTI-ROBOTS D'ENVIRONNEMENTS INCONNUS

RÉSUMÉ

La navigation de frontières est une solution viable pour l'exploration d'environnements inconnus par des équipes de robots. Dans ce travail, un nouveau cadre pour la mise en œuvre de la navigation de frontières est étudié dans le Robot Operating System (ROS), tant dans des scénarios simulés que réels. L'exploration rapide d'arbres aléatoires est utilisée pour la définition de la frontière, tandis qu'on a recours à la fusion de cartes pour l'exploration globale. L'architecture choisie est centralisée, car les robots Turtlebot3 Burgers utilisés sont tous reliés à un même centre décisionnel. Les résultats montrent que la mise en œuvre fonctionne, mais qu'il demeure des améliorations possibles pour rendre l'exploration plus efficace.

Mots-clés : navigation de frontières, exploration multi-robots, Robot Operating System (ROS).

1. INTRODUCTION

Robotic teams can be a great asset for exploration tasks, however, the control and coordination of these teams proves to be an age old research question. The task starts with the methodology behind achieving successful exploration and continues with the coordination of a team of mobile robots within the exploration process. The objective of this research is to coordinate multiple robot systems (MRS) in exploration of an unknown environment. This will be achieved in simulated scenarios and real-world tests.
The outline for the paper is as follows: Section 2 presents the background and literature review; Section 3 presents the methodology; Section 4 presents the experimental design; Section 5 presents the results and discussion; and Section 6 presents the conclusions and future work.

2. BACKGROUND

A brief review of exploration in robotic teams will be conducted, including frontier exploration algorithms/coordination and map-merging techniques.

2.1. Frontier Exploration

When exploring previously unknown areas different strategies exist to control the team of robots to successfully investigate the area.
Frontier navigation was first proposed by Yamauchi in 1997 [1]. In his work, Yamauchi defined a frontier in terms of an occupancy grid where a frontier cell is an open space adjacent to an unknown space. He defined frontier exploration as exploring the boundary between known and unknown areas. In 1998 Yamuachi [2] furthered his work to MRS. In his design, the exploration was a completely distributed process, using a nearest based allocation method for goal assignment. His design involved having each agent travelling to the nearest frontier according to the agents individual local map. This process was noted to create a lot of overlap in map coverage, as well as the assignment of frontiers.
With the goal of increasing coordination of the exploration task, Simmons, et al. [3] introduced a utility-based method for the assignment of frontiers. This made use of a centralized server or central executive to facilitate a bidding process. The bid was quantified as the net information gain taking into account the expected utility when reaching a goal and subtracting the cost of traversing to the goal. The utility function has also been modified in different scenarios to add different benefits. For example, Rooker and Birk [4] modified their definition of utility to constrain agents to stay within communication range to a central server. Some other researchers modified the definition of cost to further optimize the goal assignments. Burgard, et al. [5] used value iteration to define cost. Pal, et al. [6] take into account, not only the distance, but the energy consumption required to reach a goal point.
Furthering the bidding structure approach, Zlot, et al. [7] eliminated the need for a centralized server by introducing a market-based economy approach to the task allocation scheme. Gerkey and Mataric [8] introduced an auctioneer type of system to manage the assignment of frontiers.
Another approach, as used by Wurm, et al. [9], utilized the Hungarian method created by Kuhn [10] to optimize the task assignments. This method treats the assignment problem as a matrix of costs and uses the algorithm to solve for the lowest cost assignment. However, this method can become quite computationally burdensome, as each agent must determine the cost to each goal.
The proposed system for this research consists of utilizing a frontier-based method for exploration and map-merging using either known or unknown starting locations. The architecture of choice will be a centralized system where each agent communicates with one central executive.

2.2. Map-merging

The next question when dealing with MRS in exploration is how can a team of robots collaboratively build one map together? When merging maps together a common dilemma exists, namely, one must be able to determine the relative positions of each robot's local map to create one global map. In Figure 1 an example of the common map-merging dilemma is shown.

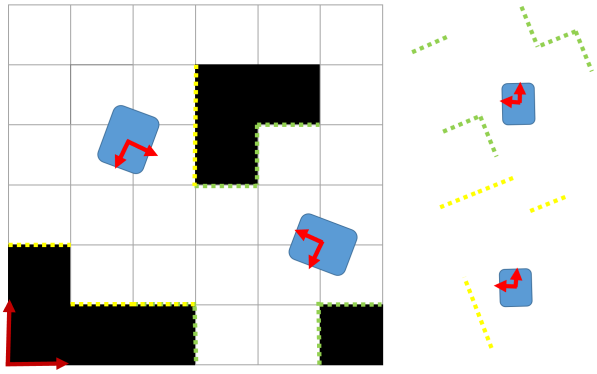


Fig. 1. Map-Merging Dilemma

Many common map-merging methods assume known initial correspondences to merge multiple maps together in a MRS exploration. For instance, in the work by Simmons, et al. [3] each agent builds their own local map and a central server is utilized to build the local maps into a global map, using a maximum likelihood estimate. Other approaches use a distributed map-merging technique like that of Yamuachi [2]. In his work, each robot holds its own local evidence grid. Each time an agent reaches a frontier it broadcasts the map information for the other agents to update their own global map.
Some techniques exist that try to negate the need for known initial correspondences, such as Fox, et al. [11]. They utilized a method where odometry information and laser scan information is exchanged every time robots come within communication range. The maps are then merged using a particle filter based method. Zhou, et al. [12] used a rendezvous solution where similarly relative poses are exchanged among different robots at rendezvous points to merge maps together.

3. METHODOLOGY

In the following section a brief review of the hardware/architecture will be conducted along with a review of the utilized software.

3.1. Hardware

The robots used in this work are open source robots, namely, the Turtlebot3 (TB3) Burger. TB3s have been designed to work with the Robot Operating System (ROS). These robots consist of a Raspberry Pi computer, CR driver board, differential drive motors, and LiDAR range sensor. The footprint for a TB3 can be found below in Figure 2. As mentioned above, TB3s are differential drive robots, consisting of two wheels with separate motors. The type of steering is labelled skid steering. The equations of motion can be

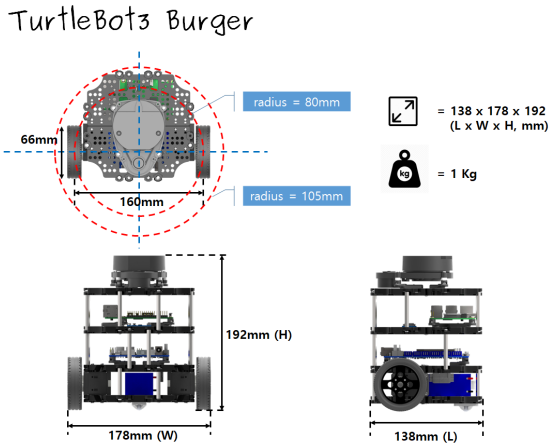


Fig. 2. Turtlebot3 Burger Footprint [13]

defined as:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \frac{\cos(\theta)}{2} & \frac{\cos(\theta)}{2} \\ \frac{\sin(\theta)}{2} & \frac{\sin(\theta)}{2} \\ \frac{1}{l} & \frac{-1}{l} \end{bmatrix} \begin{bmatrix} v_R \\ v_L \end{bmatrix} \tag{1}$$

where $\dot{x}$, $\dot{y}$, and $\dot{\theta}$ are the translation and angular velocities of the robot, θ is the heading angle, and l is the distance between wheels. For the TB3s, l is 160 mm.

3.2. Software

For the scope of this ROS Kinetic Kame was used. ROS is a framework created to facilitate the operation and control of robotic systems. It consists of a collection of packages and libraries developed for simulation and control. After the installation of the basic TB3 packages (http://wiki.ros.org/turtlebot3), many different packages have been utilized and modified as discussed below.

For path planning, the navigation stack is utilized, namely move_base (http://wiki.ros.org/move_base). Move_base consists of local and global path planning techniques for robotic control.

The global planner utilized in this work is navfn (http://wiki.ros.org/navfn). Navfn first assumes a circular robot and aims to find the minimum cost to travel from the start point to the goal point. It uses an implementation of Dijkstra's algorithm to plan the path. Dijkstra's algorithm uses a graphical based method to find the least weighted path by representing the different points on a map using vertices. It visits the current lowest cost vertex and then calculates the distance from that vertex to each unvisited neighbour while updating the lowest cost for each vertex on every iteration [14]. A simple example of Dijkstra's algorithm can be seen in Figure 3.

The local planner utilized is dwa_local_planner (http://wiki.ros.org/dwa_local_planner), which stands for Dynamic Window Approach (DWA) as proposed by Fox, et al. [15]. This planner intakes the global path given and its own local costmap and seeks to control the robot along a global path. The basic algorithm functions by sampling different velocities within the dynamic window, taking into account constraints of what velocities are achievable by the robot within a short period of time. The trajectories within that window that cause collisions are discarded. The optimal path is chosen based upon the trajectory that comes closest to the goal, global path, maximizes speed, and maintains distance from obstacles. A simple example

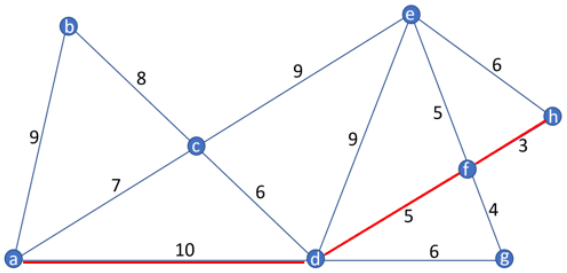


Fig. 3. Simple Example of Dijkstra's Algorithm

of this can be seen in Figure 4, where the illegal trajectories are in red, the possible trajectories are in blue, the green path represents the global path, and the star represents the goal point.

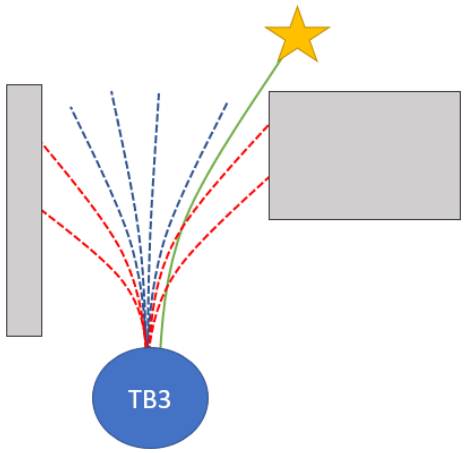


Fig. 4. Simple Example of the DWA Approach for Local Path Planning

Additionally, a localization filter was used to decrease the errors in dead-reckoning. AMCL (http://wiki.ros.org/amcl), which stands for, Adaptive Monte Carlo Localization as proposed by Fox, et al. [16] was used. This package is used to recover for drift in dead-reckoning, by using a particle filter to estimate the pose of the robot against its global map. Each particle represents a possible state that the robot could be, and the belief is represented by a probability density function distributed over the state space.

To create maps of the environments, the gmapping package was utilized (http://wiki.ros.org/gmapping). The gmapping package requires a robot equipped with odometry readings and a laser range finder. The gmapping node takes in the sensor data and converts it to a binary occupancy grid. In order to use this package, a transformation from the laser scan information received to the base link of the robot is needed along with a transformation from the base link to the odometry.

For map-merging the package utilized in this approach is a package created by Hörner [17]. In this package, multiple robots can be used to build one global map. This package contains the option of merging maps based upon known starting locations or unknown starting locations.

With known starting locations, the technique uses a rigid transformation between the different robots to merge the maps together. It takes in multiple occupancy grids and stitches together their maps based upon the transformations provided by the user. This can be problematic when dealing with unknown initial poses or uncertainty in the initial poses.

With unknown starting locations the technique uses a heuristic approach to determine the transformation between the grids of the robots. Features are determined using a feature detector and the algorithm tries to find matches using pairwise matching. If there are enough matches above a threshold value, these matches are then used to find transformations between features. If the confidence is high enough the algorithm will then determine a global transformation.
For coordination of the MRS exploration the Rapidly-exploring Randomized Trees (RRT) package is used as developed by Umari et al. [18]. They break their algorithm into three different modules, as can be seen in Figure 5, which has been modified for the TB3.

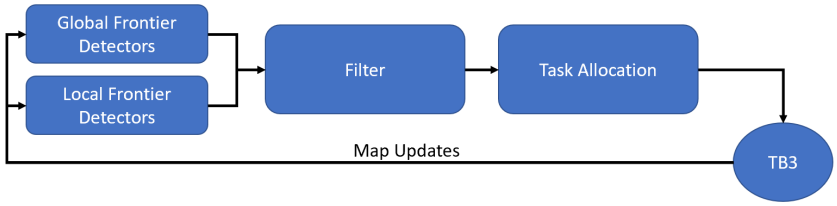


Fig. 5. Modules of RRT Exploration Package for the TB3

Frontier detection is done on both a global and local scale. They define frontiers as points that are reached by RRT. The essence of RRT is to sample surrounding spaces using points generated at random and the points that are generated are used to extend branches of a tree-like structure. In local frontier detection, the tree growth is reset every time a new point is found. The global frontier detection allows the tree to grow to reach the entire map.
The filter module takes a cluster of frontier points and groups them into one centroid. It also deletes old frontier points on each iteration.
This robot task allocation module takes into account the navigation cost and the expected information gain and calculates the total revenue. This also takes into account the hysteresis gain that ensures a robot is more biased to explore frontier points in the region. The revenue equation as defined in [18] is:

$$R(x_f) = \lambda h(x_f, x_r) I(x_f) - N(x_f) \tag{2}$$

$$h(x_f, x_r) = \begin{cases} 1 & if \mid\mid x_r - x_f \mid\mid > h_{rad} \\ h_{gain} & h_{gain} > 1 \end{cases} \tag{3}$$

where λ is a weight in regards to the information gain, x_f is the frontier point, x_r is the radius, h is the hysteresis gain, I is the information gain, and N is the navigation cost.

4. EXPERIMENTAL DESIGN

The experiments were designed for both simulated and real-world environment scenarios. The frontier exploration method of choice is RRT exploration [18]. RRT exploration requires the configuration of multiple robots to utilize the navigation stack and gmapping packages. It is required then, to configure new launch files to allow for the simultaneous launch of many TB3s with their associated navigation stack, gmapping, and AMCL nodes. For each of the robots a local costmap and a single global costmap are published, these costmap files work in conjunction with the TB3 global and local path planners. In Table 1, a summary of the different packages used for the different experiments are provided.

Table 1. Summary of Packages

Parameters	Simulation	Real-World
Environment	- Gazebo	- Basement
Starting Location	- Known	- Unknown
Mapping	- Gmapping	- Gmapping
Navigation	- move_base	- move_base
Goals	- RRT Exploration	- RRT Exploration
Map-Merging	- Multi-map merge	- Multi-map merge

4.1. Simulated Environment

For the simulations, the TB3 house environment from the TB3 packages was chosen, as can be seen in Figure 6. The starting positions are known and given in the launch files for the simulated environment. Map-

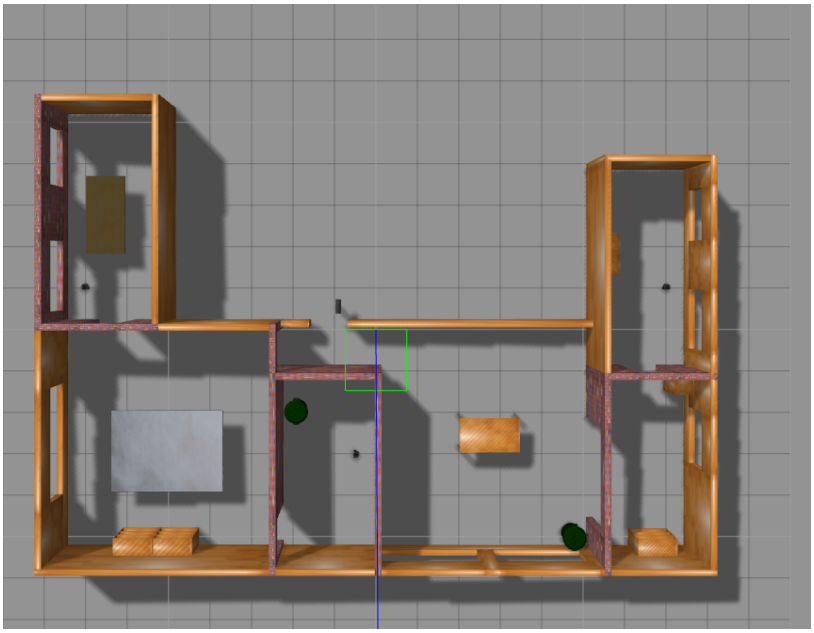
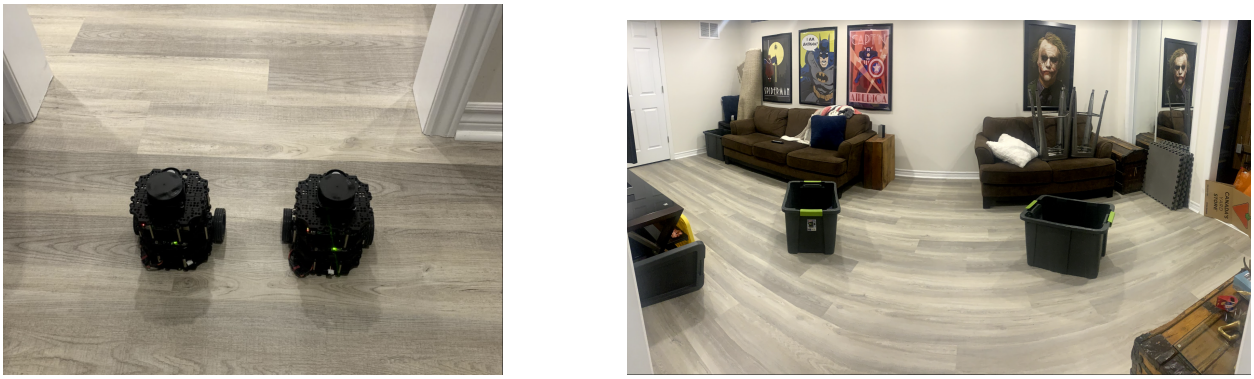


Fig. 6. Gazebo Environment with Multiple TB3s

merging is accomplished with known starting locations. The virtual TB3s are spawned using the Gazebo launch file for the TB3 house. A main launch file was written to start all three TB3s under namespace, along with their individual gmapping and move_base nodes. In the same launch file, two static transformations are published, the map-merging file is launched, and the RViz node is run. The RRT exploration file for three robots is then launched, the exploration window is defined and the robots will begin to autonomously navigate.

4.2. Real-World Environment

For the real-world experiment, the environment was changed to a basement environment as can be seen in Figure 7b. In the real-world experiments, the TB3s are brought up in their starting positions. In this main launch file two TB3s are brought up in namespace, then their individual gmapping, move base, and AMCL files are launched within a grouping. The RViz node is configured to view all of the important topics for the exploration, including the map merge topic, individual map topics, global path planner, local path planner,



(a) (b)

Fig. 7. Environment Setup for Real-World Experiments: (a) Starting Positions and (b) Basement Environment

frontier centroids and RRT trees. After the main file is launched, a modified RRT launch file for two robots is launched. Then the exploration window is defined in RViz and the robots will start to autonomously navigate.

5. RESULTS AND DISCUSSION

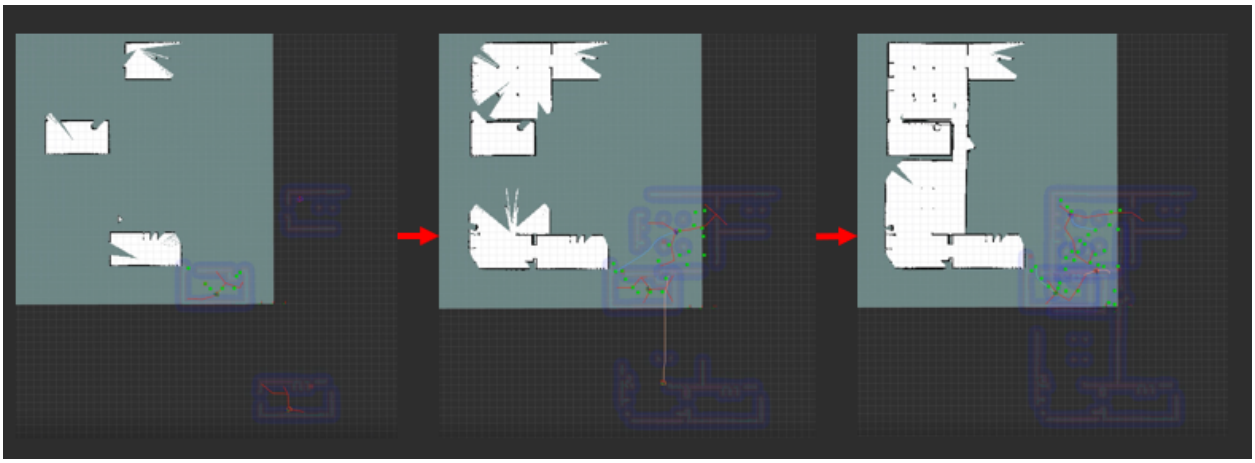
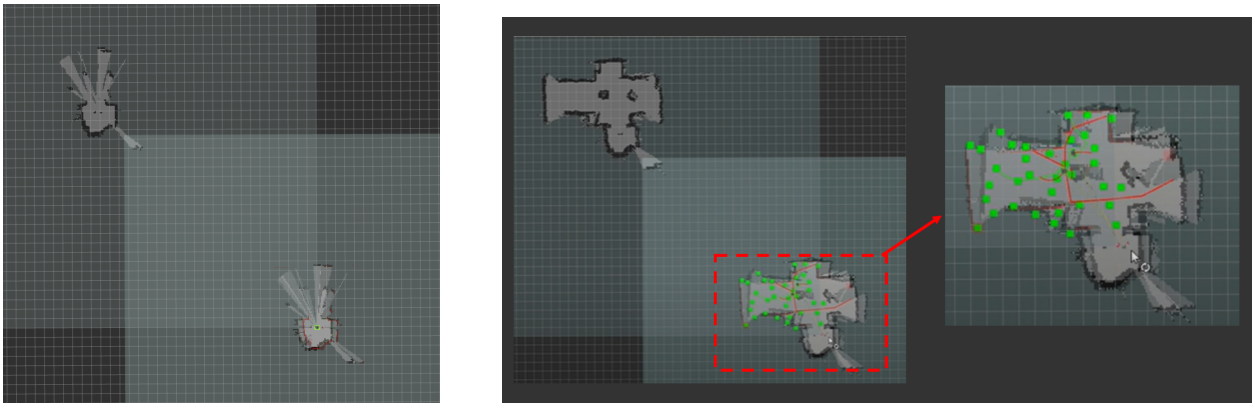


Fig. 8. Progression of Exploration and Map-Merging in Simulated Environment

5.1. Simulation Results

In the simulation, the same Gazebo environment was utilized as seen in Figure 6. The configured main launch file for the exploration was launched. The configured file contained the launch of the gmapping, move_base for the three robots, as well as, an pose initialization file that launched transformations between the robots, and finally the launch of an RViz node. After this was launched the three robot RRT exploration launch files were launched.

In Figure 8, the progressive exploration can be seen as it was being completed in the TB3 Gazebo environment. The top left in each of the grids shows the map merged topic. The bottom right is a visualization of the global costmaps for each of the robots. The other topics that can be seen are the global path plans, the RRT local frontier detection in red for each robot and the centroids of frontier regions in green.



(a) (b)

Fig. 9. RRT Exploration in RViz Using Two TB3s: (a) the Initial Map Topics Before Exploration and in (b) the Final Map Topics After Exploration

5.2. Real-World Results

For the real-world experiment the environment was changed to a basement environment as can be seen in Figure 7b. The robots were initially placed 30 cm apart from each other and were started from their on-board computers. The configured main launch file was launched and the modified RRT exploration file was also launched from two separate terminals. Before the exploration commenced the RViz looked like Figure 9a. The top left shows the map merged topic and the bottom the initial robots map topics overlaid on each other. Nearing the end of the exploration, the map topics appeared like that in figure 9b. In these figures, the green squares are the centroids of frontiers, the red straight branches are the RRT local frontier detection, the small curved red lines are the local path planner, and the long green paths are the global path planner. The final map topics can be seen in Figure 10.

5.3. Discussion

A successful implementation of multi-robot frontier exploration was achieved, along with map-merging. For the simulations, it was seen that the frontier exploration algorithm encouraged exploration of the unknown environment. The map-merging worked seamlessly with the known starting locations option, however, when unknown starting points were assumed the simulation took a long time to provide enough overlap to merge the individual maps together.

In the real-world experiments map-merging was only done assuming unknown initial coordinates. The map merge package worked well in the small basement environment to merge the two maps together. The RRT exploration also managed to allow the robots to explore an unknown environment, however, the assignment of frontier points and the detection of frontier points seemed to be inefficient. The RRT method of determining frontier points detected many frontier points that were not ideal for gaining new map information. The assignment of frontiers also led to a lot of overlap in map coverage.

6. CONCLUSIONS AND FUTURE WORK

A successful implementation of frontier navigation and multi-robot map-merging was achieved. This was realized through implementation of the multi-map merge and RRT exploration package. The multi-map merge package was implemented using both unknown and known starting locations. Known starting locations proved to be more useful in simulations and facilitated faster map-merging. It was found assuming

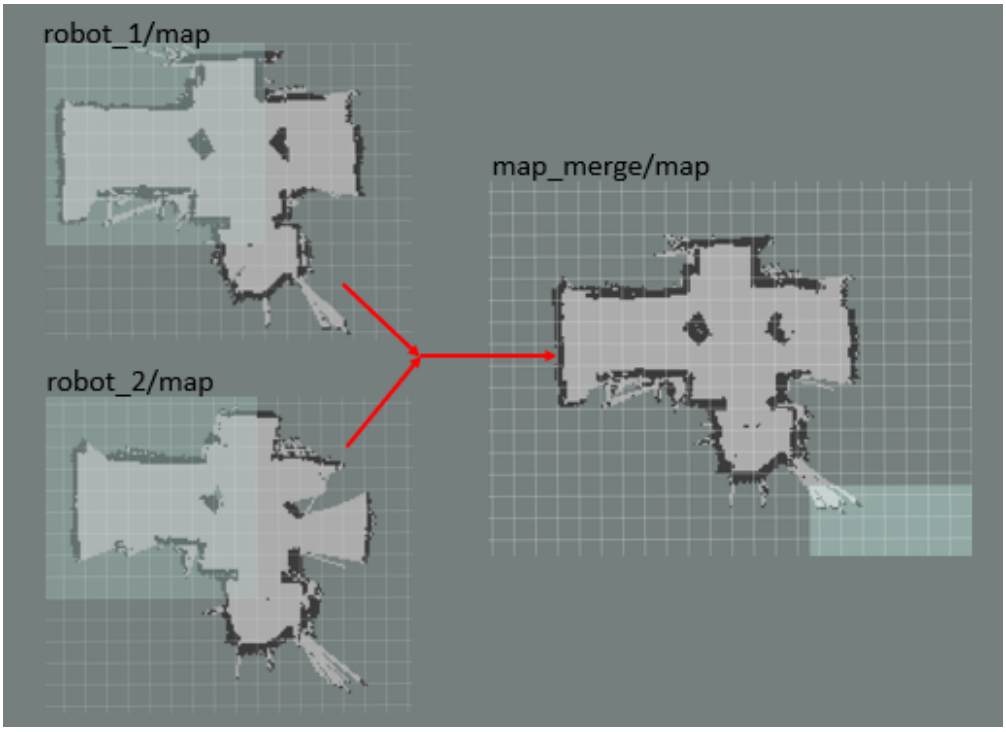


Fig. 10. Real-World Phase 2 Final Map Topics

known starting locations in real-world experiments to be unrealistic. This is because the errors in the odometry caused a lot of uncertainty in the initial starting positions. Unknown starting locations were used in both simulations and real-world applications. However, it was found that to be effective significant overlap in the maps needed to exist.

RRT exploration was found to be successfully implemented in both simulation and in real-world experiments. However, the definition of frontier points using RRTs seemed to provide a lot of overlap detection. This led to a lot more frontier points defined than are actually needed to complete a successful exploration of the environment. It was also found, that the allocation of frontiers to the different robots created a lot of overlap in map coverage. Although, using RRT for frontier point definition could have useful applications in 3-dimensional environments as mentioned by Umari, et al. [18].

In the future, increasing the size of the robotic team to test the map-merging in bigger teams, as well as increasing the environment size would prove useful. Another avenue for future work is in refining the definition of the frontiers, this is to decrease the overlap in map coverage. Also, increasing the coordination of frontier assignments to the individual agents should promote a more efficient exploration.

REFERENCES

1. Yamauchi, B. “A frontier-based approach for autonomous exploration.” In “Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA’97.’Towards New Computational Principles for Robotics and Automation’,” pp. 146–151. IEEE, 1997.
2. Yamauchi, B. “Frontier-based exploration using multiple robots.” In “Proceedings of the Second International Conference on Autonomous agents,” pp. 47–53, 1998.
3. Simmons, R., Apfelbaum, D., Burgard, W., Fox, D., Moors, M., Thrun, S. and Younes, H. “Coordination for multi-robot exploration and mapping.” In “AAAI/IAAI,” pp. 852–858, 2000.

4. Rooker, M.N. and Birk, A. "Multi-robot exploration under the constraints of wireless networking." *Control Engineering Practice*, Vol. 15, No. 4, pp. 435–445, 2007.
5. Burgard, W., Moors, M., Stachniss, C. and Schneider, F.E. "Coordinated multi-robot exploration." *IEEE Transactions on Robotics*, Vol. 21, No. 3, pp. 376–386, 2005.
6. Pal, A., Tiwari, R. and Shukla, A. "Multi-robot exploration in wireless environments." *Cognitive Computation*, Vol. 4, No. 4, pp. 526–542, 2012.
7. Zlot, R., Stentz, A., Dias, M.B. and Thayer, S. "Multi-robot exploration controlled by a market economy." In "Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)," Vol. 3, pp. 3016–3023. IEEE, 2002.
8. Gerkey, B.P. and Mataric, M.J. "Sold!: Auction methods for multirobot coordination." *IEEE Transactions on Robotics and Automation*, Vol. 18, No. 5, pp. 758–768, 2002.
9. Wurm, K.M., Stachniss, C. and Burgard, W. "Coordinated multi-robot exploration using a segmentation of the environment." In "Proceedings 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems," pp. 1160–1165. IEEE, 2008.
10. Kuhn, H.W. "The hungarian method for the assignment problem." *Naval Research Logistics Quarterly*, Vol. 2, No. 1-2, pp. 83–97, 1955.
11. Fox, D., Ko, J., Konolige, K., Limketkai, B., Schulz, D. and Stewart, B. "Distributed multirobot exploration and mapping." *Proceedings of the IEEE*, Vol. 94, No. 7, pp. 1325–1339, 2006.
12. Zhou, X.S. and Roumeliotis, S.I. "Multi-robot slam with unknown initial correspondence: The robot rendezvous case." In "Proceedings 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems," pp. 1785–1792. IEEE, 2006.
13. ROBOTIS. "ROBOTIS E-Manual: Turtlebot3 Burger." [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/specifications/#specifications. [Accessed November 15, 2020].
14. Dijkstra, E.W. et al.. "A note on two problems in connexion with graphs." *Numerische Mathematik*, Vol. 1, No. 1, pp. 269–271, 1959.
15. Fox, D., Burgard, W. and Thrun, S. "The dynamic window approach to collision avoidance." *IEEE Robotics & Automation Magazine*, Vol. 4, No. 1, pp. 23–33, 1997.
16. Fox, D., Burgard, W., Dellaert, F. and Thrun, S. "Monte carlo localization: Efficient position estimation for mobile robots." *AAAI/IAAI*, Vol. 1999, No. 343-349, pp. 2–2, 1999.
17. Hörner, J. *Map-Merging for Multi-Robot System*. Bachelor's thesis, Charles University in Prague, Faculty of Mathematics and Physics, Prague, 2016.
18. Umari, H. and Mukhopadhyay, S. "Autonomous robotic exploration based on multiple rapidly-exploring randomized trees." In "Proceedings 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)," pp. 1396–1402, Sept 2017.