A realistic Simulator in Search and Rescue

by

Yuhan Zhu

Supervisor: Prof. Goldie Nejat

April 2022

B.A.Sc. Thesis

Division of Engineering Science
UNIVERSITY OF TORONTO

Abstract

To date, the robotics application in Urban Search and Rescue (USAR) has been a promising research area. It provides a safe solution to assist rescue workers in reducing potential risk for workers to enter unsafe structures and increasing the speed of response by increasing robot quantities in rescue scenes. A high-leveled autonomous robot team requires efficient collaboration, which requires reliable communication with their teammates, and centralized control center. However, in USAR scenarios, communications are frequently interrupted. Thus, a multi-agent Deep Reinforcement learning model was recently presented to handle poor communication issues in an unknown cluttered environment. To test the performance of the DRL model, integration with the real-world is required. While extracting large datasets from physical robots is expensive and unsafe, a real-world simulator was designed to accelerate the training process. Therefore, in my thesis project, a 3D realistic simulator in search and rescue is proposed for future deep learning research. The 3D simulator was tested on sixty trials with three different initial positions, five robot team sizes, and four environment sizes. The results demonstrate that the robot fleet could improve exploration efficiency with the increment of robots, which met the expectation. The 3D realistic simulator has excellent potential for further improvement to mimic real search and rescue scenes.

Acknowledgements

Throughout the writing of this thesis project, I have received a great deal of support and assistance.

I would first like to thank my supervisor, Professor Goldie Nejat, whose expertise was invaluable in formulating the research questions and methodology. Your insightful feedback pushed me to sharpen my thinking and brought my work to a higher level.

I gratefully recognize the help of Ph.D. student Aaron Tan whose valuable guidance throughout my studies and detailed feedback was influential in shaping my experiment methods and results.

My appreciation also goes out to my family and friends for their encouragement and support throughout my studies.

Table of Contents

List of Figures

List of Tables

1. Background

The application of robots in Urban Search and Rescue (USAR) has gained significant attention in recent years. Due to their replaceability, repairability, and upgradable feature, a large number of collaborative robots are dispatched in a coordinated manner to explore unknown cluttered scenes and search for victims [1]. The methodologies can be categorized as centralized or decentralized in multi-robot system (MRS), each with its advantages and drawbacks. The centralized systems can optimize the performance with a global knowledge of the system, but they suffer from single point failure of the server [2]. The decentralized system requires strong understanding of the environment to pre-design strategies for exploration [3], but depends on information exchange locally with neighboring robots [4]. However, in USAR missions, communication with a centralized computer is often unreliable, and local communication with teammates is limited by both robots' transmission range and environment (e.g., obstacles) [5]. As a result, both traditional methodologies are hard to achieve satisfactory performance with respect to mission time, energy consumption, etc. To address these performance degradations, robots should learn to predict other robots' behaviors [6]. A multi-agent DRL method known as MADE-Net (Macro Action Decentralized Exploration Network) was recently presented to handle the poor communication issue in the unknown cluttered environment [7]. It learns the teammates' intentions during centralized training and executes exploration tasks in decentralized manner.

To further validate the performance of MADE-Net in addressing the communication dropout challenges, integrating the architecture into real-world environments with physical robots is needed. However, DRL methods require considerable training data, and extracting training samples directly from physical robots is time-consuming and unsafe [8]. A real-world simulator for multi-robot exploration will accelerate the learning process. Previous works have been done to construct USAR customizable environments with Gazebo 3D simulator and perform frontier exploration in the environment [9] using Clearpath Husky robot with a stereo camera and a thermal camera. However, there is no simulator for interfacing the USAR 3D simulator with deep learning frameworks to support multi-thread training [10].

2. Objectives

This project aims to build a 3D simulator to support deep reinforcement learning in USAR scenarios. The following requirements should be met.

2.1 Simulation Configuration

To further evaluate the performance of deep learning model regarding the environment exploration, the simulator should be flexible in terms of spatial configurations (e.g., obstacle densities, terrain steepness, etc.), robot configuration, robot team size (from 1 to 6), and the environment size (20m x 20m, 30m x 30m, 40m x 40m, etc.) based on the multi-thread training requirements.

2.2 Robot Simulation

The USAR 3D simulator should be capable of including any robot configuration in simulations by editing the URDF file and control parameters. As for this particular application of the simulator, given that the physical robots used for real-world testing are Clearpath Jackal robots with ZED2 stereo camera [11] and Velodyne lidar [12], the USAR 3D simulator will need to be modified to accommodate the specified Jackal configurations [13].

2.2.1 Path Planning

The simulated robots should be capable of performing their navigation tasks (i.e., navigation between arbitrary initial and goal locations) based on the tasks allocated by the deep learning model (MADE-Net DRL model in this application) in a decentralized manner. Herein, a task is defined as an exploration goal, which is spatially distributed within the unknown environment. Each robot is assigned a unique task to navigate simultaneously.

2.2.2 SLAM

The simulated robots should be equipped with SLAM (Simultaneous Localization and Mapping) to construct maps of the unknown environment and track robots' localization while navigating [14]. The generated map will be taken as high-level observations by the proposed DRL model.

2.2.3 Robot Communication

The simulated robots explore the environment in a decentralized manner. Thus, no centralized communication is needed; but the robots should be capable of exchanging information and performing map merging operations within a predefined sensing range.

3. Literature Review

Due to the existence of previous work on USAR environment simulator [9], the 3D simulator platform will not be reviewed in this section. The main components reviewed in detail is SLAM [15] [16] [17] [18] [19] [20] [21] [22] [23], path planning [24] [25] [26] [27] [28] [29] [30] and map merging [31] [32] [33].

3.1 SLAM Algorithms

Since urban search and rescue environments are significantly cluttered and unstructured [1], the simultaneous localization and mapping (SLAM) algorithms should be capable of acquiring a 3D map of a USAR environment and tracking robots' location [34]. 2D SLAM methods such as Gmapping [35], HectorSLAM [36] are not suitable in this scenario because they are limited by their ability to retrieve 3D information and construct 3D map. Currently, the sensors deployed in the SLAM system are mainly Light-Detection, Ranging (LiDAR), and cameras [15]. The SLAM methods reviewed are categorized by the type of sensors deployed in the SLAM system [16]. The popular SLAM methods reviewed are visual-based SLAM [17] and Lidar-Based SLAM [15].

3.1.1 Visual-Based SLAM

Visual-based SLAM uses different types of visual sensors, including monocular [37], stereo [38], and depth (RGB-D) cameras [39], to capture images of the environment [15]. These images are used to extract features and depth information to estimate robot's motion and position and generate 2D or 3D maps [17]. Considering existing hardware installed in physical Jackal robots, SLAM methods that can use stereo camera as input source. In this section, papers introducing visual-based SLAM methods are discussed in detail, including ORB-SLAM2 [18], S-PTAM [20], and RTAB-Map [21].

The method proposed in [18] introduces a feature-based SLAM method, ORB-SLAM2. It can utilize monocular, stereo, and RGB-D cameras for image extraction. The system preprocesses input to extract ORB features [40] at the salient point for tracking, mapping, and place recognition, which achieves good performance regardless of camera autogain and autoexposure,

and illumination changes. More importantly, these features are fast to track and match, allowing real-time operation with good precision. The system is embedded with a place recognition module based on DBoW2 [41] for relocalization and performs motion-only BA (Bundle Adjustment) to compute the optimal structure and motion solution. ORB-SLAM2 was tested on KITTI [42], EuRoC [43] datasets for stereo method, and TUM RGB-D [44] dataset for RGB-D method and achieves the highest accuracy in most cases compared to stereo method LSD-SLAM [45], and ElasticFusion [46], Kintinuous [47], DVO-SLAM [48] and RGB-D SLAM [49].

S-PTAM[20] is a real-time keyframe-based SLAM method with stereo camera. The features extracted from images are binary features describing visual point-landmarks to reduce the storage requirements and the matching cost. Real-time loop detection and correction are included in the system. The generated map is refined with bundle adjustment in a local co-visible area to improve global consistency. S-PTAM was compared with ORB-SLAM2 and S-LSD-SLAM on the KITTI dataset [42] and it presented the least amount of rotation error among these three methods. It was tested on Level 7 s-block dataset [50] with ORB-SLAM2, showing comparable accuracy and similar error peaks.

RTAB-Map (Real-Time Appearance-Based Mapping) [21] was proposed for long-term and large-scale environment mapping. It utilizes Stereo, RGB-D, and multi-camera as input and extracts GFTT (GoodFeaturesToTrack) [51] features for matching. The features detected are matched by nearest neighbor search with nearest neighbor distance ratio (NNDR) test [52] for F2M (Frame-To-Map). For F2F (Frame-To-Frame) approach, the optical flow is done directly on GFTT features without descriptor extraction. The system is able to provide visual odometry using Frame-To-Map (F2M) and Frame-To-Frame (F2F) approaches and estimate the robot's position by computing transformation of the current frame to feature in keyframe or feature map with Perspective-n-Point (PnP) RANSAC [53]. A block matching algorithm [54] is applied to compute dense disparity images and convert them to point clouds for stereo images. A loop closure detection is introduced using the bag-of-words approach [55] for optimization. The performance of RTAB-Map was evaluated on KITTI [42], EuRoC [43], TUM RGB-D [44], and PR2 MIT Stata Center [56] dataset. Compared to ORB-SLAM2 [18] and LSD-SLAM [45] methods, RTAB-Map has 0.09% larger translation error and 0.0001 deg/m better rotation

performance on the KITTI dataset. For EuRoC and TUM datasets, ORB-SLAM2 performs better than RTAB-Map, but it is more computationally expensive.

3.1.2 Lidar-Based SLAM

LiDAR-Based SLAM system obtains accurate point clouds of the environment [15] using a laser sensor and generate the map. In this section, three different LiDAR-based SLAM methods are reviewed: RTAP-Map [21], Google Cartographer [22], and SegMatch [23].

RTAB-Map also includes lidar as input source [21]. The pose estimation is updated when ICP (iterative-closest-point) [57] is successfully done using Point to Point (P2P) or Point to Plane (P2N) correspondence. The pose estimation requires valid motion prediction from a previous registration or from external odometry transformation. The pose and map optimization are done the same way as visual SLAM. The evaluation was done with the same datasets as mentioned above in RTAB-Map visual SLAM approach in comparison to Google Cartographer [22], Karto SLAM [58], Hector SLAM [36] and GMapping [35]. For short-range lidar, RTAB-Map is the best for both sequences during evaluation. For long-range lidar, RTAB-Map performs equally well to Hector SLAM.

Google Cartographer involves two processes, called local SLAM and global SLAM [59]. During local SLAM, new scans are matched against the current submap with a Ceres-based scan matcher. The submap is expressed as a probability grid where each discrete grid point shows the obstacle occupancy. In global SLAM, generated submaps and scans are included for loop closing and errors accumulated in submaps are removed. The algorithm was evaluated based on Radish [60] dataset benchmarking and tested in the real-world [61]. Compared with Graph Mapping, Google Cartographer performs better on 5 out of 7 datasets. And in real-world experiments, the error results are roughly in the expected order of magnitude compared to ground truth.

In [23], a real-time laser-based 3D SLAM method, SegMatch, was introduced. The point clouds obtained from 3D LiDAR are first segmented into unique point clusters for matching using the “Cluster-All Method” [62]. Next, feature extraction is performed based on eigenvalue and ensemble of shape histograms. A random forest for classification and timing performances was deployed, the correspondences are then matched. RANSAC (random sample consensus) [63]

algorithm is applied to verify the potential matches. The proposed algorithm was tested using the KITTI dataset [64] and can successfully localize its vehicle within 35 meters 95% of the time.

3.1.3 Limitations and Selection

A summary of limitations for all SLAM methods mentioned above is presented in Table 1. For visual-based SLAM, the localization and mapping are performed by matching features extracted from images [19]. While requiring unobstructed cameras and adequate features for matching, the images' quality suffers from illumination and visibility changes in real life. However, cameras are cheaper, have lower power consumption and they can provide more detailed environmental information than other sensors, such as lidar and Radar [65].

For LiDAR-based SLAM, the localization and mapping are achieved by matching point cloud data [23]. The information extraction is limited to geometry information, ignoring semantic information. LiDAR can provide high-precision distance measurements but is not able to provide as finely detailed point clouds as images in terms of density. The performance of LiDAR-based SLAM is not affected by illumination changes compared with visual-based SLAM. In places with few obstacles, the algorithm suffers from aligning the point clouds, resulting in losing track of the pose prediction [15].

All methods mentioned in previous sections have packages implemented in ROS (Robot Operating System) except for SegMatch [23]. Considering all aspects, RTAB-Map is selected for simulation because it supports both visual-based and LiDAR-based SLAM, allowing changing sensors based on specific needs. Bothe methods in RTAB-Map are capable of generating 2D occupancy grids, which is suitable for most path planners [66].

Method Name	**Summary Description**	**Limitation**
ORB-SLAM2 [18]	• Feature-based SLAM method • Utilizes monocular, stereo, and RGB-D cameras • ORB features are fast to track and match, allowing real-time operation with good precision	Trajectory estimation not as accurate as RTAB-Map [67]

S-PTAM [20]	• Real-time keyframe-based SLAM method with stereo camera • Binary features describing visual point-landmarks	Support only stereo camera
RTAB-Map [21]	• Support both visual-based and lidar-based SLAM • Features detected by F2M (Frame-To-Map) and F2F (Frame-To-Frame) approach • Provide visual odometry	Images' quality suffers from illumination and visibility changes
Google Cartographer [22]	• Local SLAM and global SLAM • Local SLAM generates submaps expressed in probability grids with new scans • Global SLAM includes submaps and scans for loop closure and error removal	Information extraction is limited to geometry information
SegMatch [23]	• "Cluster-All Method" segments point clouds into unique point clusters • RANSAC is used for verifying the potential matches	No integration with ROS

Table 1: A summary of all SLAM methods and their limitations

3.2 Path Planning Algorithms

Path planning is a crucial task in autonomous robotics. It involves defining a path from the initial position to the goal position avoiding obstacles in the environment [68]. In this section, papers pertaining to path planning are reviewed. These papers are categorized as global path planners [69] and local path planners [70].

3.2.1 Global Path Planner

Global path planners reviewed in this section are categorized as grid-based approaches [25], evolutionary approach [71], and probability-based approaches [66].

3.2.1.1 Grid-Based Algorithm

Grid-based path planning utilizes 2D occupancy grid. The optimal path is calculated based on the property of each grid cell.

A* algorithm [72] finds the shortest path from initial node q_{init} to goal node q_{goal}. It employs a function f(q) = g(q) +h(q), where g(q) estimates the cost of the shortest path from initial node q_{init} to q, and h(q) estimates the cost of the shortest path from q to $q_{goal.}$ The costs are calculated based on either Euclidian distance or Manhattan distance. The experimental result shows that the best result in path length corresponds to A* algorithm, but the processing time is more than 30% higher compared to probabilistic Roadmap and genetic algorithms [68].

D* algorithm [25] is a modified dynamic version of the A* algorithm, where cost functions are updated once new obstacles are detected. A complete map is not required to compute the cost function. Instead of using g(q) to represent the cost from current node q to goal node q_{goal}, it calculates the cost backward from the destination cell. Recomputations are required every time obstacles are detected. It suits the dynamic and unknown environment. In the 8x8 grid test, D* algorithm has 0.003 second longer execution time than A* algorithm.

3.2.1.2 Evolutionary Algorithm

Genetic Algorithm is a stochastic search technique that mimics the natural evolution process inspired by Darwin [71]. Potential solutions to a problem are encoded as chromosomes, which form a population [73]. In path planning, chromosomes are used to represent the grid cells of the mobile robot environment. GA generates a population of possible paths iteratively, and the population is evaluated by the predefined fitness function. The crossover strategy prevents the solution from converging to local minima. GA was tested in 10x10 and 100x100 grid world and it was able to find the shortest path in different environment setups within 100 generations [74].

3.2.1.3 Probabilistic-based Algorithm

Rapidly-Exploring Random Tree (RRT) [27] is a probabilistic-based algorithm, solving path planning without nonholonomic constraints. The tree incrementally grows from the starting point q_{init} to a random number of new nodes q_{new} until it hits the endgoal q_{goal}. And two different functions, creation and expansion of the tree, are used. The expansion of the tree is one way from starting point to end point whereas the expansion of the tress is bidirectional to improve the time efficiency. RRT algorithm performs better in execution time compared to RRT*, but worse than

A* and HPA* algorithms. In addition, the path length computed by RRT is the longest among all algorithms, including A*, HPA*, MEA*, etc [74].

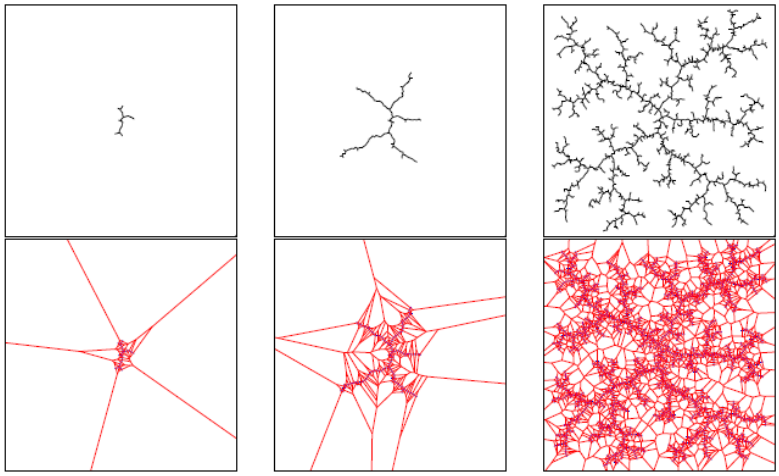


Figure 1: RRT exploration process [74]

3.2.1.4 Limitation and Selection

A summary of all SLAM methods discussed, and their limitations are shown in Table 2. The computation cost of grid-based algorithm and RRT algorithm increases considerably for large scale and high-dimensional environment because of the usage of grid map [74]. Parameter tunning is one disadvantage of evolutionary algorithms in addition to huge computation cost increment. RRT requires a large number of iterations and samples to avoid local minima, which increases the need for memory [71]. However, robots are not equipped with high-power computers with enough computer memory, which limits the potential of using the RRT algorithm.

Considering difficulties in parameter tuning and computation power optimization, the evolutionary approach is not considered an appropriate approach in the USAR scenario. Since the chosen SLAM method RTAB-Map can generate a 2D occupancy grid for both visual-based and LiDAR-based approaches, the grid-based methods are chosen. In addition, A* algorithm exists in ROS as one of the main global planners. Thus, the A* algorithm is chosen for future simulation development.

Method	Summary	Limitations
A* algorithm [72]	• Grid-based path planning • Cost function f(q) = g(q) +h(q)	The processing time is more than 30% higher compared to the probabilistic Roadmap and genetic algorithms.
D* algorithm [25]	• Modified dynamic version of the A* algorithm • Cost functions are updated once new obstacles are detected	Computation cost increase greatly for large scale and high-dimensional environment.
Genetic Algorithm [73]	• Generates a population of possible paths • Population is evaluated by the predefined fitness function	Parameter tunning is hard
Rapidly-Exploring Random Tree (RRT) [66]	• Probabilistic-based algorithm • Tree incrementally grows from the starting point until it hits the end-goal	Requires a large number of iterations and samples to avoid local minima, increasing memory usage.

Table 2: A summary of global path planning algorithms and their limitations

3.2.2 Local Path Planner

In this section, local path planners available in ROS are reviewed, including Dynamic Window Approach (DWA) local planner [28], Time Elastic Band (TEB) local planner[29] and EBAND local planner [30].

3.2.2.1 Dynamic Window Approach (DWA) local planner

The DWA local planner deals with the local path planning problem in the 2D occupancy grid [28]. The DWA planner discretely proposes linear velocity (dx, dy) and angular velocity (dϴ). For each velocity sample, forward simulation from the current robot's position is performed to

predict the position of the robot after the proposed velocity is applied for some short period of time (Figure 2). The results from forwarding simulation are evaluated and unsuccessful trajectories are forbidden. The best trajectory is chosen, and the corresponding velocity is sent to a mobile base. The algorithm is tested in 2D corridor with static obstacles. The experiments showed that the DWA planner is very robust and successfully handled collision avoidance tasks, with a safe maximum driving speed up to 95 cm/s, while the computation time is within 0.25 sec.

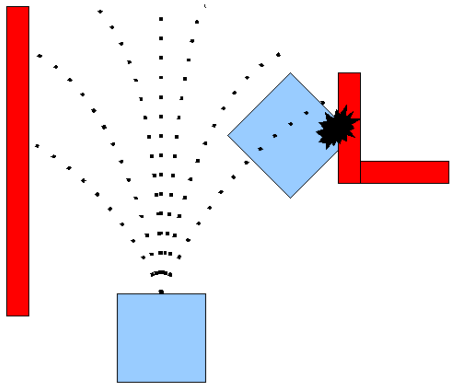


Figure 2: Example of forward prediction of DWA local planner [75]

3.2.2.2 Time Elastic Band (TEB) local planner

TEB local planner is introduced in [75]. It accounts for the velocity and acceleration constraints as well as surrounding obstacle distance, with a weight assigned to each. It generates a sequence of poses for discrete time intervals. Similar to the DWA planner, the linear and angular velocity will be fed into a mobile base controller for navigation. The results from simulations and experiments illustrates that the approach is robust and provides a smooth path [76]. In addition, TEB local planner takes into consideration of geometric constraints. In recent version of *teb_local_planner*, it supports dynamic obstacles. However, the performance depends on the obstacle tracking and state estimation accuracy. As shown in figure 3, the TEB local planner collides as its preferred trajectory is elongated by the motion of human.

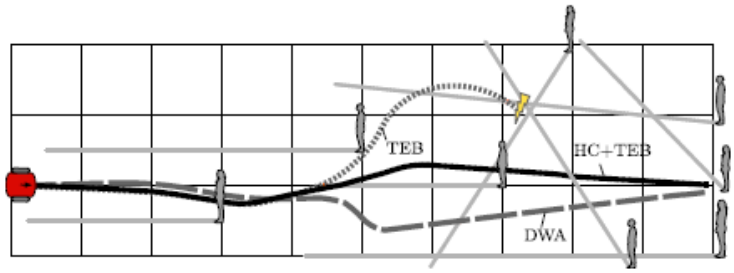


Figure 3: Scenario 1: Traces of robot and human's movement [78]

3.2.2.3 EBAND local planner

EBAND local planner [30] produces a collision-free path by two artificial forces: contraction force, and repulsion force. The contraction force helps plan a smooth path, while the repulsion force keeps the robot away from obstacles. The global trajectory is modified such that the robot can smoothly follow the path. Results showed that the EBAND planner was less smooth compared to DWA and TEB planner and experiences similar test results as TEB planner in terms of execution time. This algorithm was implemented in ROS as eband_local_panner package [77].

3.2.2.4 Limitations and Selection

DWA local planner does not consider the robot's geometric constraints, which means the robot will not keep a safe distance away from obstacles. This may cause collisions [78]. For EBAND and TEB local planner, the direction of dynamic obstacles' movement may make the path planning worse. In scenario 2 (Figure 4) when a human ignoring the approaching robot, the initial optimal trajectory passes between the wall and the human becomes infeasible. The updated trajectory will be elongated to avoid the human, and eventually cause collision with the wall [30].

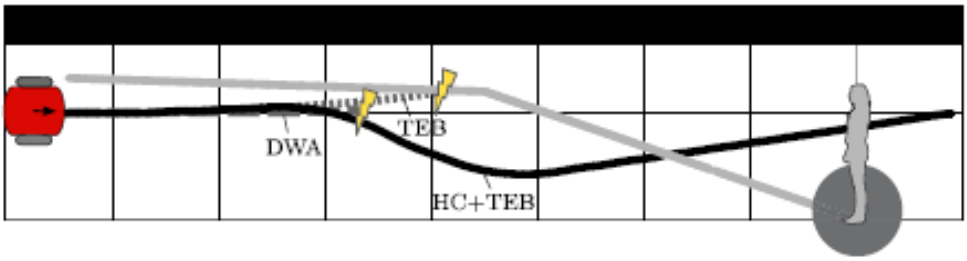


Figure 4: Scenario 2: Traces of robot and human's movement [78]

Due to the requirement of multiple robots to work collaboratively in the same environment, we need to consider the effect of one robot crossing the other's planned path. Thus, DWA planner was chosen regardless of the geometric constraints of the robots. In addition, DWA planner is the default local planner implemented in Jackal.

3.3 Map Merging Algorithms

To construct a complete global map of the environment, the data collected by different robots need to be integrated into a single map. Moreover, the integration should be done as fast as possible to facilitate the environment exploration process [31]. If the initial position of robots is known, map merging is a straightforward extension of the single robot mapping [32], [33]. This is because the transformation between local maps is known; map merging is a simple addition operation. However, the integration of maps when robots do not know their relative position is more complicated. With few correspondences in local maps, feature extraction will fail, and transformation between different local maps can be erroneous.

The method proposed in [33] allows keeping multiple disconnected maps in memory. The map merging is done by computing a transformation between two overlapping regions by scan matching. One of the maps is then transformed such that two overlapping areas fit. A loop closing operation is applied to refit the two maps to improve the accuracy of merging. The algorithm was tested with RoboCup Rescue Virtual Robots Competition maps [79] and the Cogniron dataset [80].

The approach in [31] describes an adapted particle filter in combination with a predictive model of the environment in addition to estimation of overlaps. The most likely hypothesis is determined at each iteration of the particle filter. The algorithm was tested with a sequence of data collected in three environments. The experiment showed that the map-merging algorithm generated accurate, consistent maps in all 12 runs and actively verified the relative location of robots.

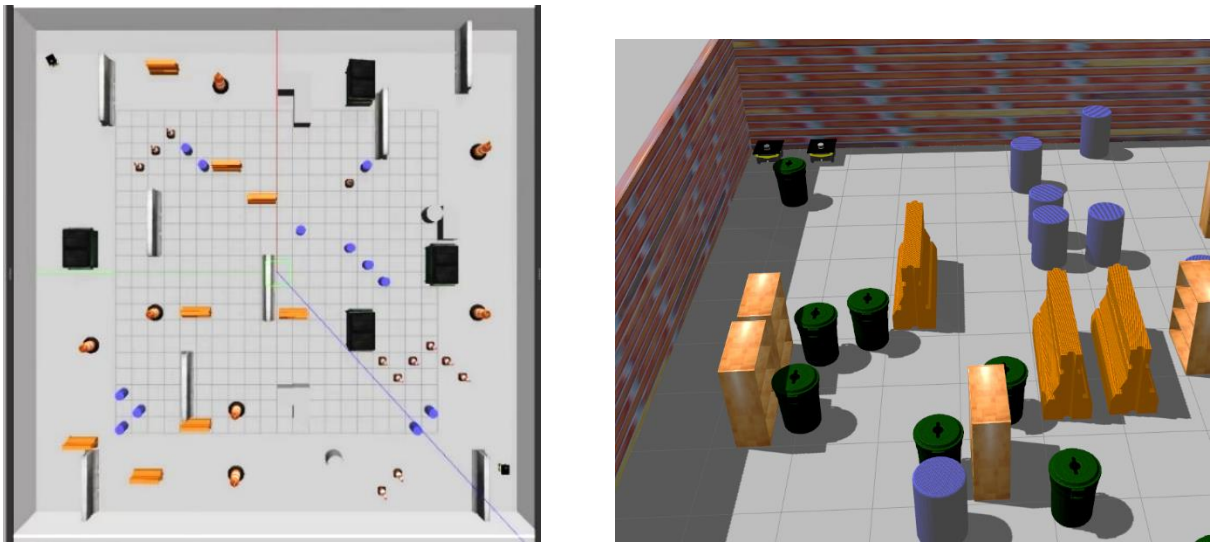
In our environment exploration experiment, the initial positions of the robots are known. Thus, straight forward map merging algorithm is more than enough to handle this scenario. In ROS, map_merge node was implemented in m_explore package and selected for future simulations.

4. Methods

This section will describe the final simulator design in detail, which is divided into four components: Environment Simulation, Multi-Robot Simulation, Map Merging, Integration with MADE-Net

4.1 Environment Simulation

The simulator should be flexible enough to provide customized cluttered and unstructured environment by using various types of obstacles and placing them randomly in the scene, Figure 5. The environment configurations are saved in .world files, where the size of the workspace, type of obstacles, and their placement can be customized through gazebo environment configuration.



(a) *(b)*

Figure 5: a) Bird-eye view of the simulated environment, b) closer view of the environment

4.1.1 Exploration Boundary

The boundary of the environment used is wall models built with gazebo building editor. The exploration environment is located from $-\frac{env\_size}{2}$ to $\frac{env\_size}{2}$ for both x and y axes. The wall texture was modified to red brick for SLAM to extract features more easily.

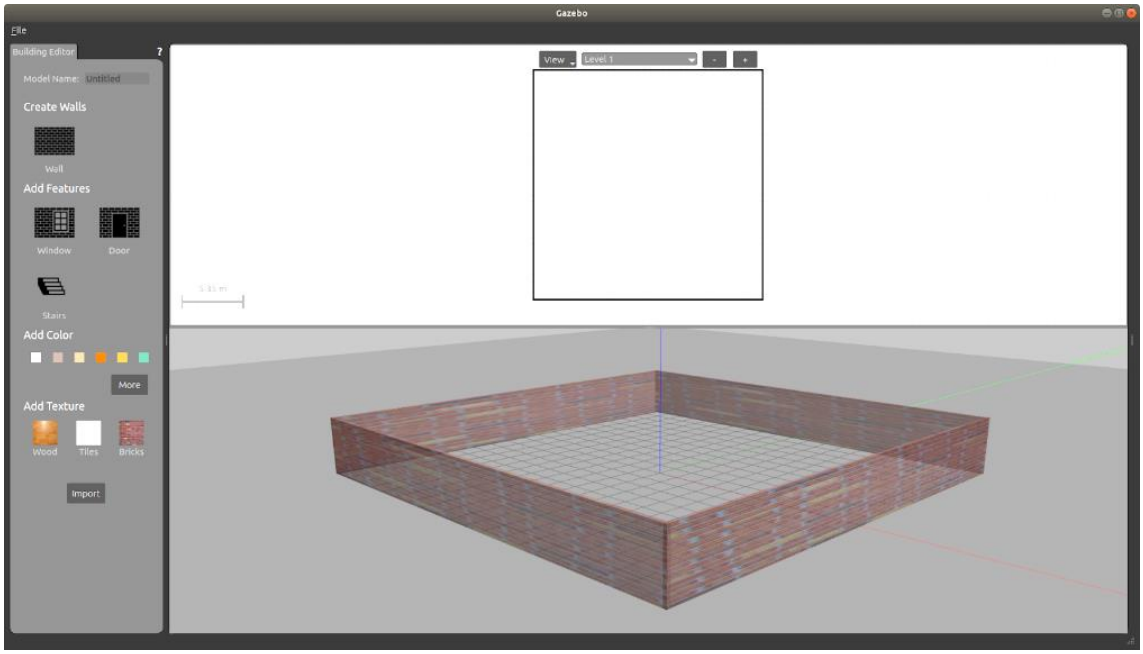


Figure 6: 20x20 exploration boundary built with Gazebo Building Editor

4.1.2 Obstacles

Gazebo provides various obstacle models in the online library, such as construction cones, garbage bins, dumpsters, bookshelves, etc., Figure 7.

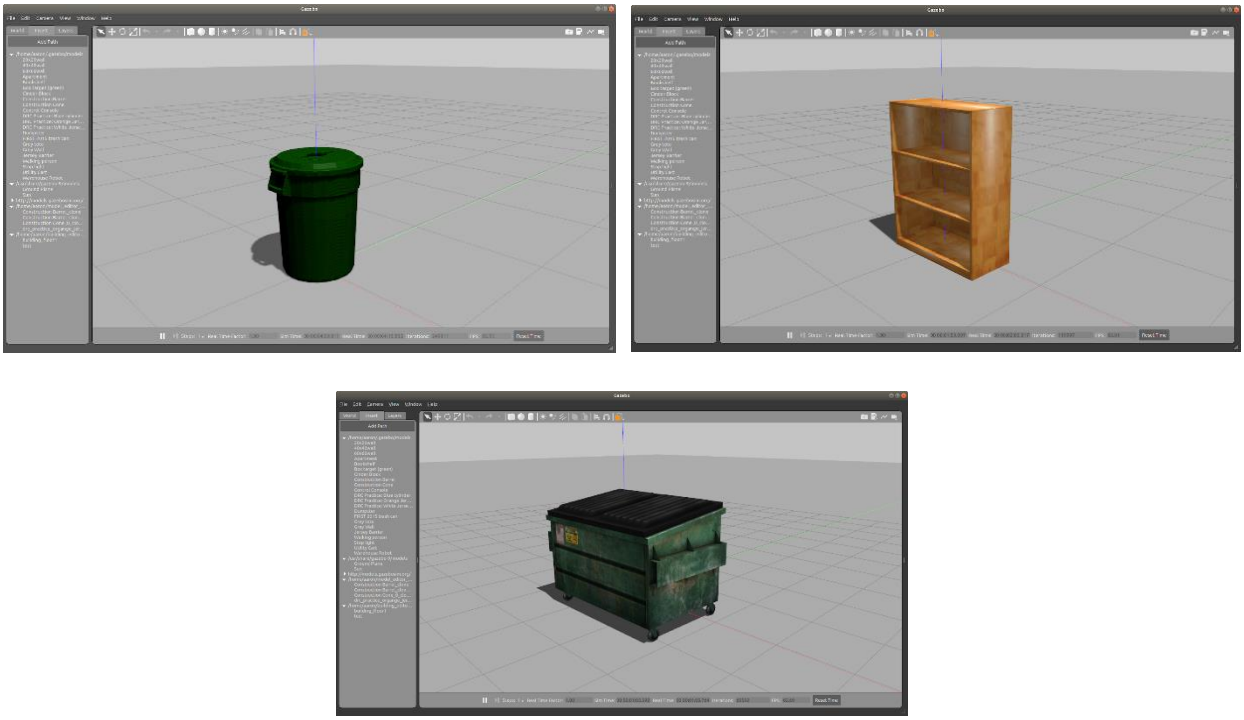


Figure 7: Garbage bin, bookshelf, and dumpster obstacle model

4.2 Multi-Robot Simulation

As mentioned above, the robot in simulation is designed with one ZED2 camera and one VLD-16 lidar in Figure 8 to allow the application of both visual and LiDAR SLAM methods.

(a) (b)

Figure 8: a) 3D view of Jackal with ZED2 camera and VLP-16 lidar[13], b) simulated 3D view of Jackal with ZED2 camera and VLP-16 lidar

Jackal is integrated with high torque 4x4 drivetrain for rugged all-terrain operations [13]. It has an onboard computer, GPS, and IMU, which is suitable for research in all terrains. It is already integrated with ROS as ros-melodic-jackal-simulator package [81].

Jackal robots can be spawned in any world in Gazebo by *spawn_node* in modified *spawn_jackal.launch* file in *jackal_gazebo* package to account for spawning multiple robots. By default, a *robot_state_publisher* node is initialized for each robot to calculate the robot's forward kinematics and publishes transformation through *tf*.

The robot configuration, including customized sensors, will be discussed in Experiment section.

4.2.2 RTAB-Map SLAM

RTAB-Map was selected for robot localization and mapping. Visual SLAM mode was used because it can collect more information than lidar SLAM mode. A ZED2 camera ROS plugin is not available at the moment; therefore, a customized stereo-camera plugin was used. *libgazebo_ros_multicamera.so* was installed and added to the Jackal robot URDF (Unified Robot Description Format) file [82] to simulate the stereo camera. It will create two camera topics (from the left and right cameras), and the output image is shown in Figure 9.

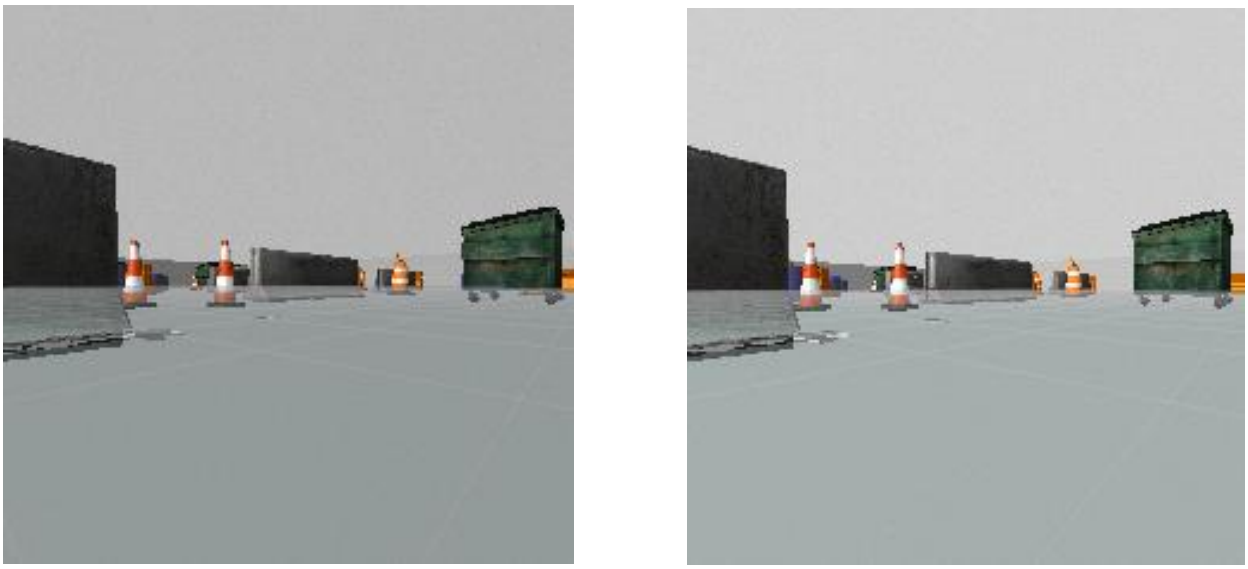
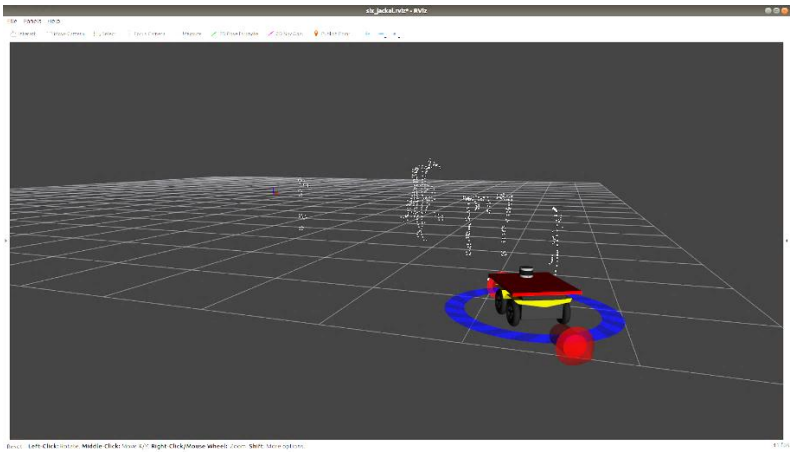
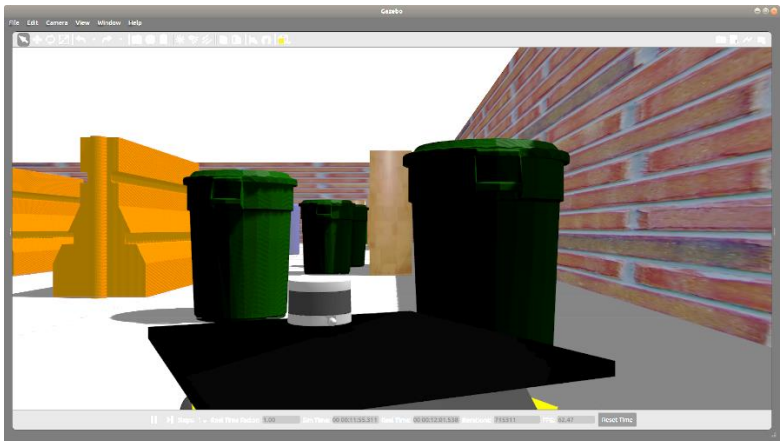


Figure 9: Left and right raw image obtained from stereo camera.

To simplify the matching problem, the raw images are passed through stereo_image_proc node to perform rectification [83] which warps the images taken by stereo camera such that they appear equivalent to images taken with only horizontal displacement, and de-mosaicing which reconstructs a full-color image [84]. A 3D point cloud map is then created by matching features extracted from these images, shown in Figure 10



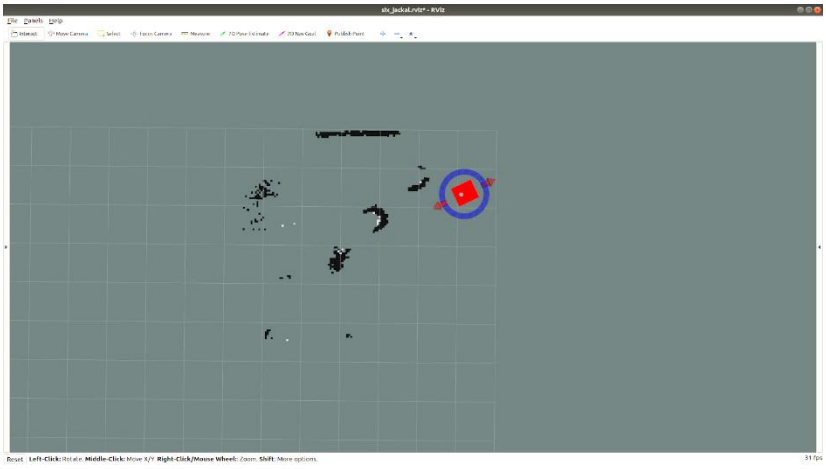
a) Point cloud extracted from stereo images



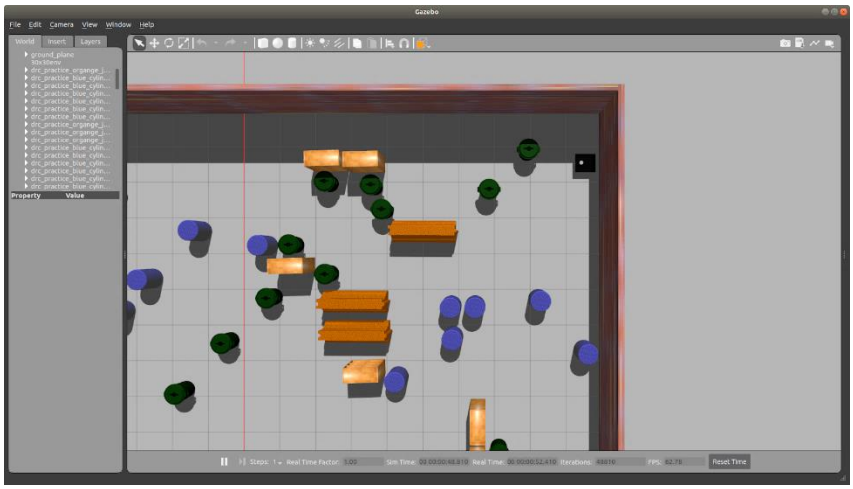
b) Actual environment

Figure 10: An 3D point cloud generated from stereo images

A 2D occupancy grid (shown in Figure 11) is created by projecting the 3D point clouds on the ground plane (e.g., x-y plane). The black pixels are space occupied by obstacles. And the red modules represent Jackal robots.



a) Occupancy grid generated from point cloud data in Figure 10



b) Actual environment bird eye view

Figure 11: An occupancy grid generated by RTAB-Map

4.2.3 Navigation

After constructing the 2D occupancy map and localizing themselves on the map, the robots can then navigate to arbitrary locations. The *move_base* package is implemented to achieve global navigation in the simulated environment [85]. This package is commonly used for robot navigation, and it is integrated with Jackal official package, which can be directly used. Some modifications should be made to consider different namespaces for each robot. With the chosen global planner (A*) and local planner (DWA), robots can complete navigation goals assigned by the proposed DRL model (MADE-Net in this application) [7]. Figure 12 shows the recommended navigation stack by ROS. In general, the robot should subscribe to sensor readings and occupancy maps to construct global and local costmaps, which will be used by global and local planners to generate a feasible path. Velocity commands are calculated and sent by *local_planner* to move the robot.

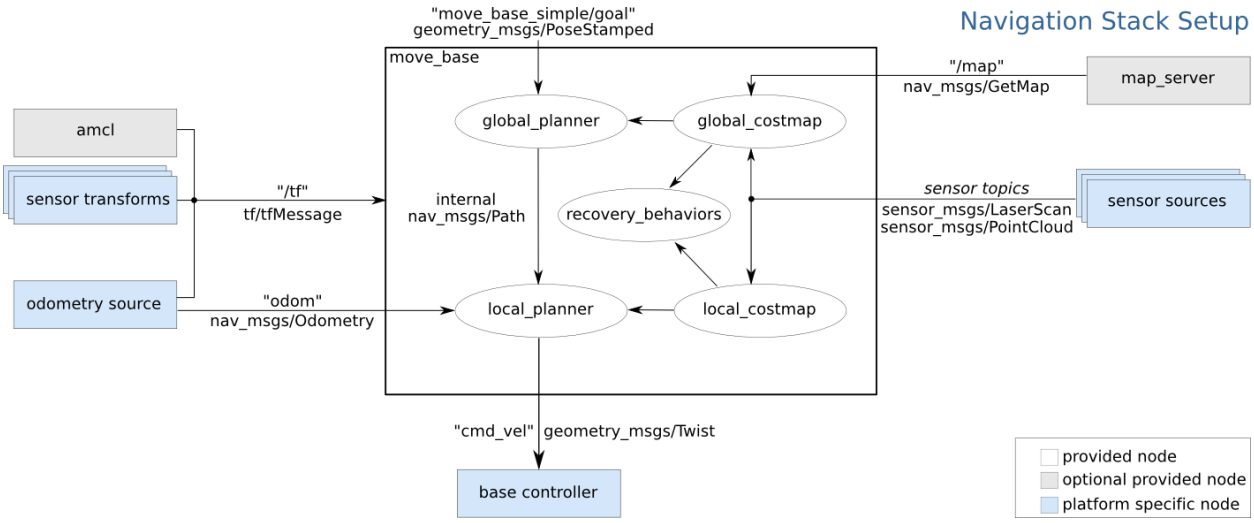


Figure 12: Navigation stack setup required for move_base node [87]

The move_base package utilizes 2D occupancy map to calculate a global trajectory and local trajectory when a navigation goal is set, Figure 13. The green line and orange line are the global and local trajectories that one robot should follow, respectively.

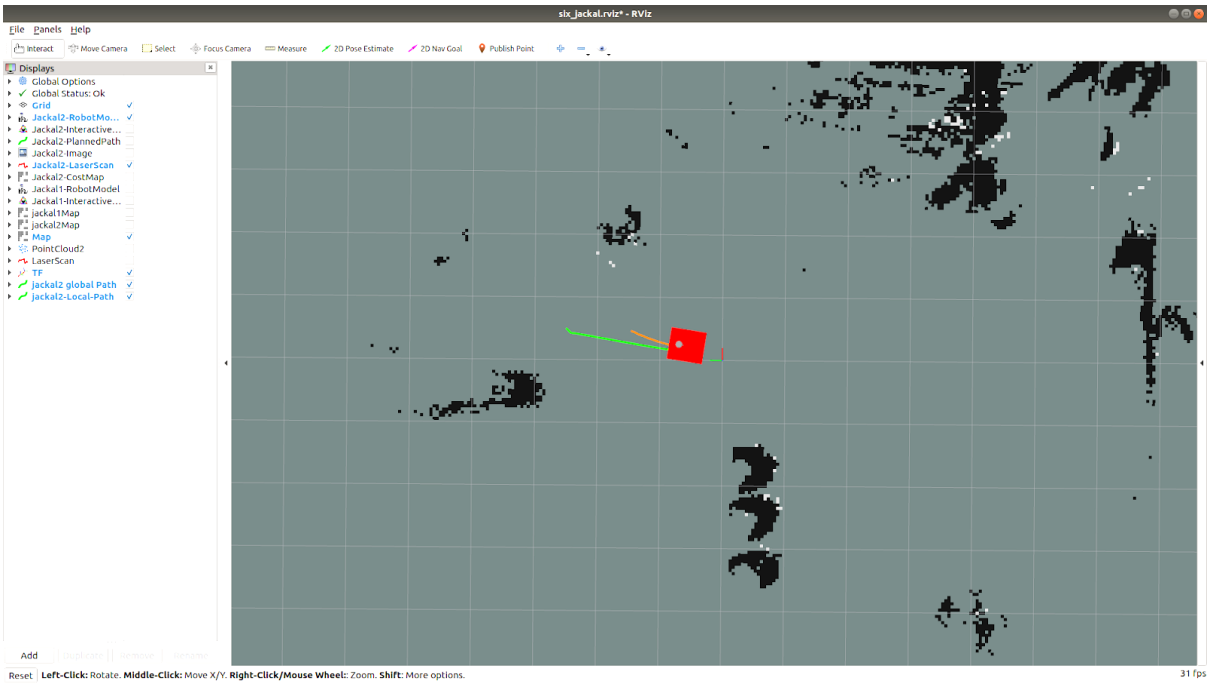


Figure 13: Global Path (green) and Local Path (orange)

4.3.3 Map Merge

In this USAR simulator, the communication focuses on the map exchange when two robots satisfy the conditions of map exchange: 1) two of the robots are within a predefined communication range, and 2) there is no obstacles or walls in their line of sight.

Now that each robot generates its 2D occupancy map with its own RTAB-Map node, a map merging node is introduced such that they can exchange their map information and develop a merged map. The map exchange process of two robots is shown in Figure 14.

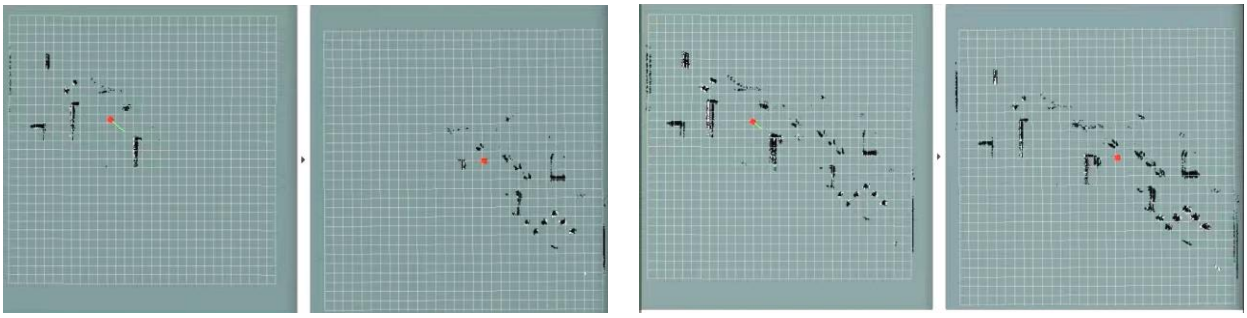


Figure 14: a) Two robots' local maps before map exchange b) robots' local maps after map exchange

The previous map merging algorithm used was *map_merge* node implemented in m_explore package [86]. However, this node introduced some additional errors in map processing, which resulted in misalignment between the local map and the merged map, in Figure 15.

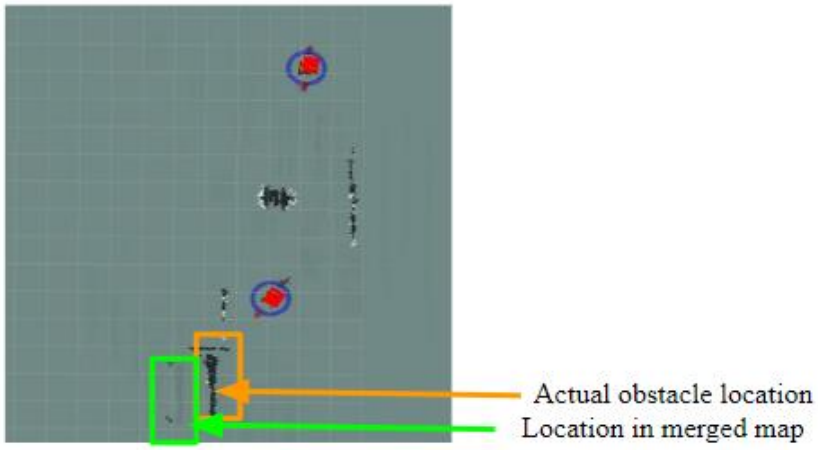
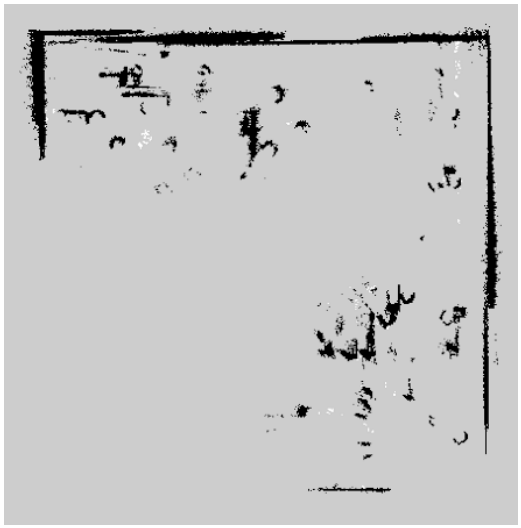
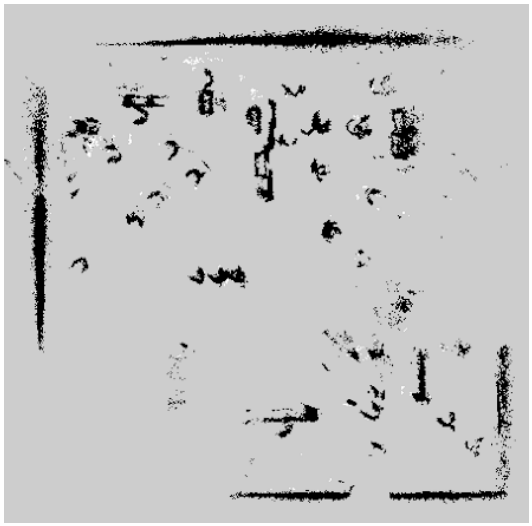


Figure 15: Misalignment of map using existing m_explore package

Therefore, a customized python script was built that deals with occupancy grid directly utilizing NumPy array [87]. The customized script subscribes to local map topics generated by SLAM algorithm and greedily combines the 2D occupancy grids. For faster computation speed, each occupancy grid is converted from tuple to NumPy array. The algorithm works for an arbitrary number of robots. A demonstration of the map merging process with two robots is shown in Figure 16.



a) 2d occupancy map – jackal1



b) 2d occupancy map – jackal2

c) Merged map

Figure 16: Map merging demo

The map merging part of the algorithm rqt_graph is shown in Figure 17. The “base_map_server” nodes are initially publishing empty maps, making “jackal1/map_merge” and “jackal2/map_merge” nodes only to publish their local 2D occupancy map generated by rtabmap SLAM. Topics “jackal1/map” and “jackal2/map” are subscribed by “map_merge” node [88], which publishes the global “map”. The merged map of two robots is running in the background at the beginning. Once two robots meet the communication criteria, the global map will be saved by “map_saver” node and the “base_map_server”s will start to publish the saved map [88].

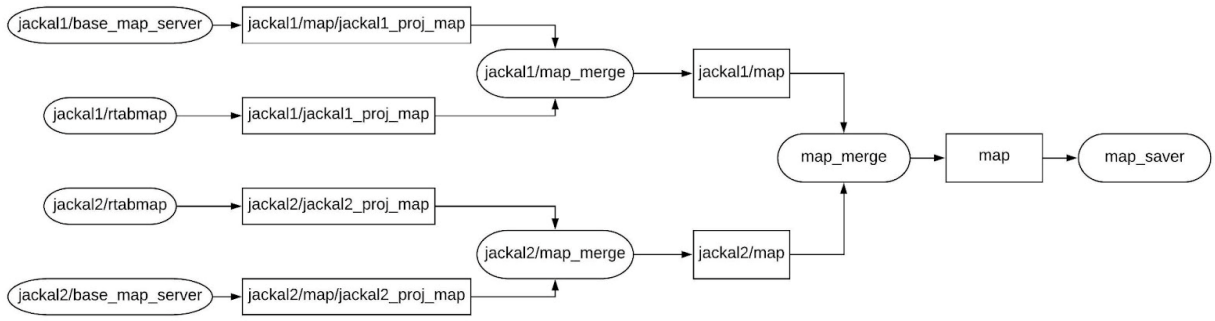


Figure 17: Partial rqt_graph of the system focusing on map merge

4.3.3 Integration with MADE-Net

The integration with MADE-Net is accomplished by jackal_waypoint.launch launch file, which takes in desired goal location, orientation and wait time after reaching the goal for each robot and publishes command to move_base_simple/goal for move_base node to execute. An example of a set of waypoints is shown in Figure 18.

```
Robot name and following lines have coordinates x,y,z roll,pitch,yaw (in radians) and time which is
the time the robot holds the position before moving to the next waypoint
jackal1
12 13 0 0 0 3.14 1
10 12 0 0 0 3.14 1
8 10 0 0 0 3.14 1
6 9 0 0 0 5 1
5 6 0 0 0 3.6 1
4 4 0 0 0 3.6 1
2 2 0 0 0 3.14 1
jackal2
-11 -14 0 0 0 0 1
-9 -12 0 0 0 0 1.5
-7 -10 0 0 0 0 3
-5 -9 0 0 0 0 3
-3 -7 0 0 0 0 2
0 -4 0 0 0 0 4S
```

Figure 18: Example waypoints used in navigation

5. Experiments and Results

5.1 Experiments

To test the scalability of the simulator and integration with deep reinforcement model, the following experiment setups are proposed.

5.1.1 Computer Specification

The computer specifications are shown in the table below. In total, two computers are used during the experiments.

	Computer 1	**Computer 2**
CPU	Intel i7 11800H 8-core	AMD Ryzen threadripper 1950x 16-core processor x32
RAM	16GB	110GB
GPU	NVDIA GeForce 3060	NVDIA GeForce 2070

Table 3: The computer specifications in experiment simulations

5.1.2 Robot Setup

The jackal robot is modified to be equipped with VLD-16 Lidar and ZED2 stereo Camera, by importing the *velodyne_gazebo_plugins* and *stereo_camera_plugin.* The Velodyne lidar is located (x = 0.120m, y = 0m, z = 0.05m) with respect to the robot's origin. The stereo camera is located at the (x = 0.25m, y = 0m, z = 0.132m) with respect to the robot's origin.

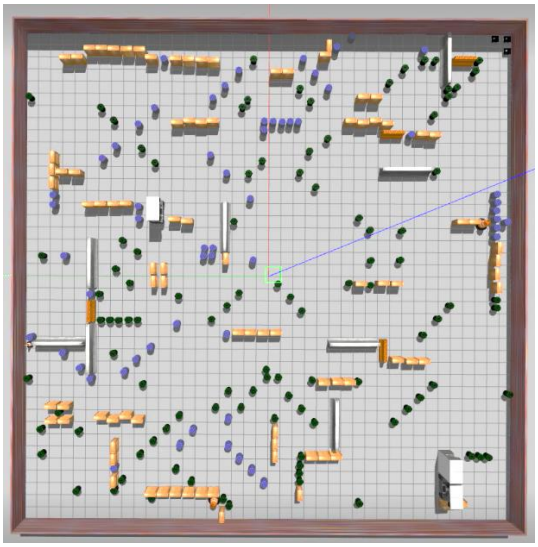
The robots used in the experiment were equipped with RTAB-Map SLAM for constructing maps and localization. A* algorithm and DWA local planner were used for global and local path planning.

Each robot has a communication range of 5 meters, within which the robots can exchange their information, including local maps, robot positions, etc.

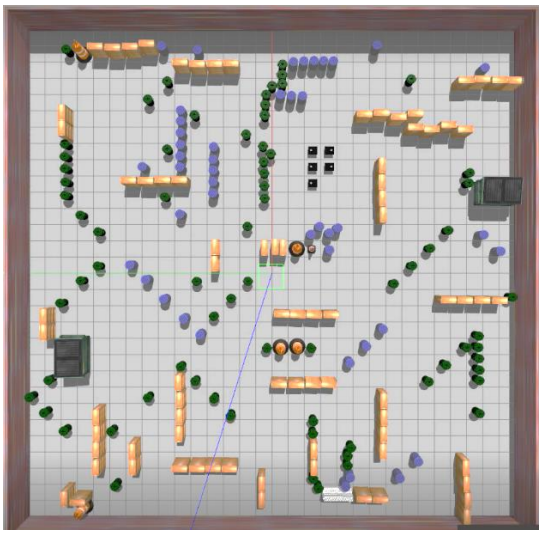
5.1.3 Validation Environment Setup

The current environments generated for simulation are 20m x 20m, 30m x 30m, 40m x 40m, and 60m x 60m, which various densities and randomly placed obstacles, in Figure 19. The spacing between diagonally placed obstacles was larger than the robot size, to allow robot to go through. The environment setup can be expanded to any arbitrary size according to different training needs.

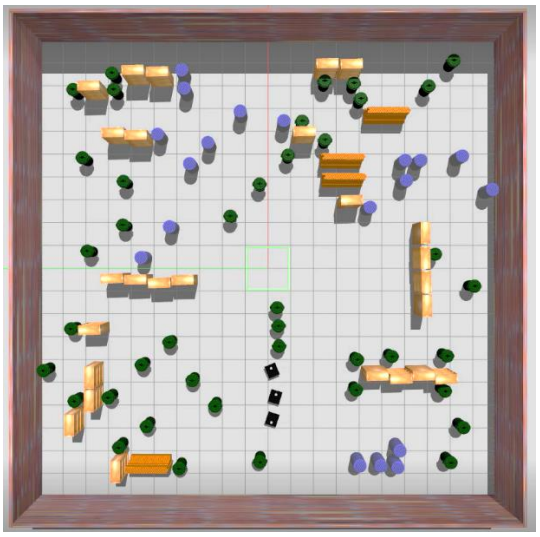
a) 60 x 60 m^2



b) 40 x 40 m^2



c) 30 x 30 m^2

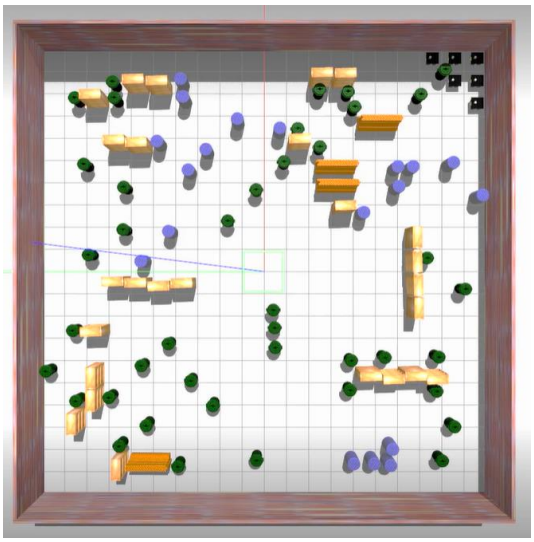


d) 20 x 20 m^2

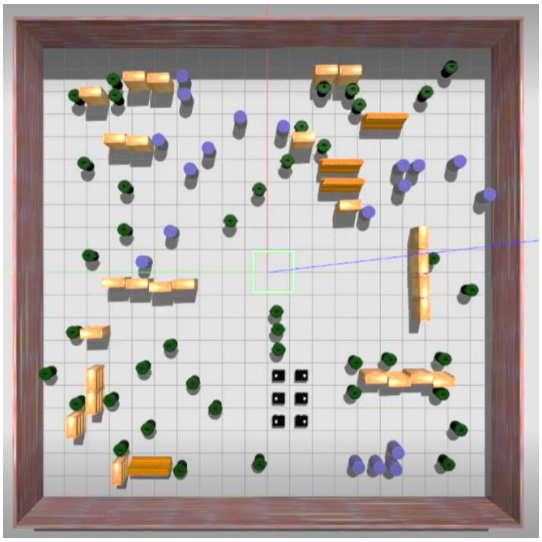
Figure 19: Simulated environments for validation process

5.1.4 Experiment Procedure

Once the robots and environment are set up properly in Gazebo, the next step is to run the navigation tasks assigned by MADE-Net. The robots were spawned at three different robot team starting positions, bottom left corner, top right corner, and a randomly selected position from the middle of the map, shown in Figure 20. In bottom left corner and top right corner trials, the robots localized further from the corner started navigation first, whereas in the third trials, the robots started navigation at the same time. Each trial was repeated in four different environment sizes with five different robot team sizes.



a) Top right corner initial position



b) Middle initial position

c) Bottom left corner initial position

Figure 20: Three different initial position for robot team

The test was terminated when 95% of the environment was globally explored by the team. For each experiment, the total exploration time was recorded based on the simulation time shown in gazebo.

5.2 Results and Discussion

The total exploration time with different team sizes and environment sizes is averaged across 3 cases with different team starting positions, shown in Figure 21, Table 4. In general, the exploration time experienced a decreasing trend with the increment of robot team size. However, there was barely any improvement in total exploration time in 20x20, 30x30, and 40x40 environments with 6 robots. Contrariwise, the teams significantly improved exploration time, even with 6 robots in a 60x60 environment. This is due to more space for robots to navigate without interference with other robots.

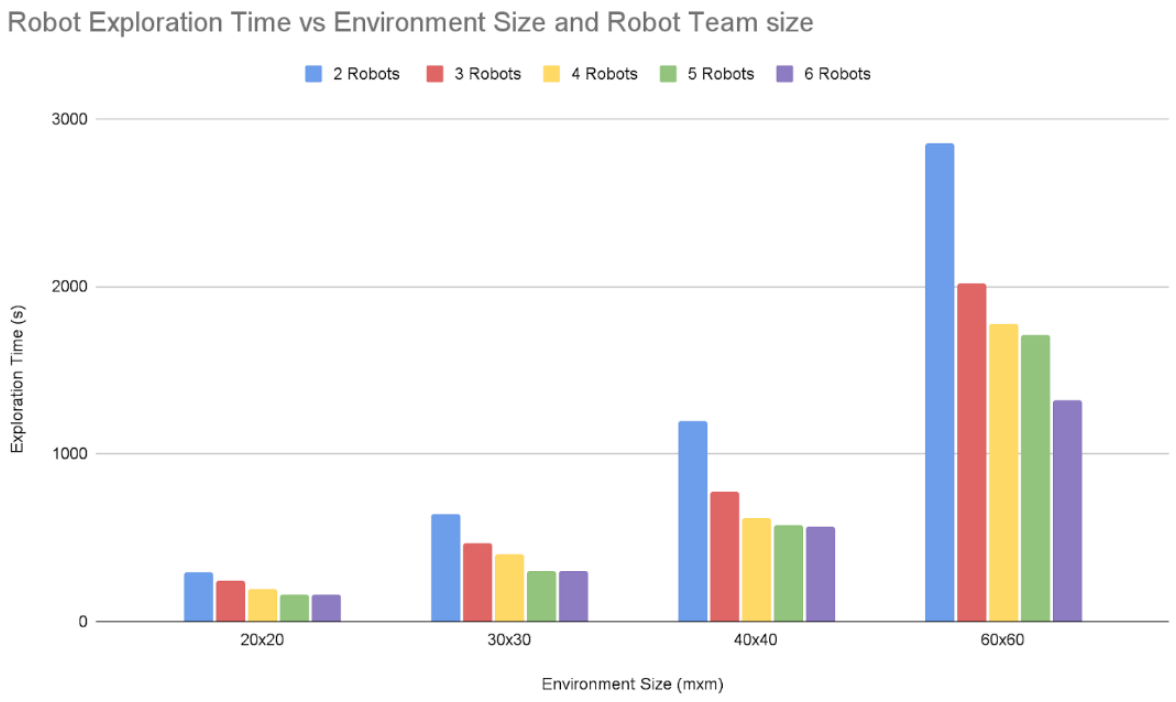


Figure 21: The total exploration time for each environment size and robot size combinations

Exploration Time (s)		Robot Team Size				
		2 Robots	3 Robots	4 Robots	5 Robots	6 Robots
Environment Size	20x20	294	241.666667	195.33333	159.66667	161.333333
	30x30	643	470.6666667	400.666667	300.666667	299
	40x40	1198.66667	774	617.66667	579.33333	565
	60x60	2854	2016.666667	1777	1709	1319

Table 4: The total exploration time for each environment size and robot size combinations

The widespread distribution of robots improved exploration efficiency when more robots were added to the scene. However, dispatching more robots into the scene also results in more time for each robot to navigate between waypoints and avoid collisions with teammates. Especially at the beginning of exploration, when robots were gathered in one corner, the time for each robot to navigate significantly increased as the team size increased. This can be validated by the shorter exploration time in the case where robots started in the middle of the environment and spread out to distinct orientations, shown in Table 5.

Exploration Time (s)	2 robots	3 robots	4 robots	5 robots	6 robots
BL	720	407	473	327	360
TR	667	530	342	286	313
MID	445	445	377	277	265

Table 5: The exploration time for each starting position in 30x30 environment

6. Conclusion and Future Work

In conclusion, the 3D realistic simulator design can simulate environment exploration with multi-robots for deep reinforcement training. A 3D unstructured world that resembles the 2D grid world for training can be customized in Gazebo simulator. An arbitrary number of robots with different initial positions can be included in the world to perform navigation tasks assigned by MADE-Net and generate macro-observations maps required using RTAB-Map SLAM algorithm. In addition, the communications between robots are represented by simulating map merging processes. A merged map can be produced when two robots meet.

The design was tested with 60 trials with four different environment sizes, five robot team sizes, and three different initial positions. In 95% of the trials, the environments were explored completely. The simulated results met the expectation that the exploration time improves with the increment of team size [7].

However, there are some limitations in this design that can be improved in the future. The current simulation only supports visual-based SLAM. Thus, it is impossible to extract observations at the back of the robot, which is one of the reasons why the environments were not explored completely. If LiDAR-based SLAM is introduced into the simulator, it will allow users to get 360 degree observations, resulting in more flexibility in customizing robot exploration configurations. Also, a more robust and fast map merging algorithm with the ability to clear moving obstacles can be designed to make corrections to the merged map. In addition, rough terrain and unstructured obstacles can be included in the simulation environment to further validate the performance reinforcement learning in more realistic urban search and rescue scenes.

Reference:

[1] Y. Liu and G. Nejat, “Robotic Urban Search and Rescue: A Survey from the Control Perspective,” *J. Intell. Robot. Syst.*, vol. 72, no. 2, pp. 147–165, Nov. 2013, doi: 10.1007/s10846-013-9822-x.

[2] S. Saravanan, K. C. Ramanathan, R. MM, and M. N. Janardhanan, “Review on state-of-the-art dynamic task allocation strategies for multiple-robot systems,” *Ind. Robot*, Sep. 2020, doi: 10.1108/IR-04-2020-0073.

[3] M. Geng, K. Xu, X. Zhou, B. Ding, H. Wang, and L. Zhang, “Learning to Cooperate via an Attention-Based Communication Neural Network in Decentralized Multi-Robot Exploration,” *Entropy*, vol. 21, p. 294, Mar. 2019, doi: 10.3390/e21030294.

[4] M. Geng, X. Zhou, B. Ding, H. Wang, and L. Zhang, “Learning to Cooperate in Decentralized Multi-robot Exploration of Dynamic Environments: 25th International Conference, ICONIP 2018, Siem Reap, Cambodia, December 13–16, 2018, Proceedings, Part VII,” 2018, pp. 40–51. doi: 10.1007/978-3-030-04239-4_4.

[5] F. Amigoni, J. Banfi, and N. Basilico, “Multirobot Exploration of Communication-Restricted Environments: A Survey,” *IEEE Intell. Syst.*, vol. 32, pp. 48–57, Nov. 2017, doi: 10.1109/MIS.2017.4531226.

[6] A. Bautin, O. Simonin, and F. Charpillet, “Towards a communication free coordination for multi-robot exploration,” May 2012.

[7] A. H. Tan, F. P. Bejarano, and G. Nejat, “Deep Reinforcement Learning for Decentralized Multi-Robot Exploration with Macro Actions,” *ArXiv211002181 Cs*, Oct. 2021, Accessed: Jan. 23, 2022. [Online]. Available: http://arxiv.org/abs/2110.02181

[8] N. Liu, Y. Cai, T. Lu, R. Wang, and S. Wang, “Real–Sim–Real Transfer for Real-World Robot Control Policy Learning with Deep Reinforcement Learning,” *Appl. Sci.*, vol. 10, p. 1555, Feb. 2020, doi: 10.3390/app10051555.

[9] “Aronlin4458/USAR,” *GitHub*. https://github.com/Aronlin4458/USAR (accessed Jan. 23, 2022).

[10] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, “USARSim: a robot simulator for research and education,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, Apr. 2007, pp. 1400–1405. doi: 10.1109/ROBOT.2007.363180.

[11] “ZED 2 - AI Stereo Camera.” https://www.stereolabs.com/zed-2/ (accessed Jan. 27, 2022).

[12] “Puck Lidar Sensor, High-Value Surround Lidar,” *Velodyne Lidar*. https://velodynelidar.com/products/puck/ (accessed Jan. 27, 2022).

[13] “Jackal UGV - Small Weatherproof Robot - Clearpath,” *Clearpath Robotics*. https://clearpathrobotics.com/jackal-small-unmanned-ground-vehicle/ (accessed Jan. 27, 2022).

[14] S. J. Julier and J. K. Uhlmann, “Building a million beacon map,” in *Sensor Fusion and Decentralized Control in Robotic Systems IV*, Oct. 2001, vol. 4571, pp. 10–21. doi: 10.1117/12.444158.

[15] J. Yang, “A Survey of SLAM Research based on LiDAR Sensors,” vol. 1, no. 1, p. 8, 2019.

[16] H. Taheri and Z. C. Xia, “SLAM; definition and evolution,” *Eng. Appl. Artif. Intell.*, vol. 97, no. Complete, 2021, doi: 10.1016/j.engappai.2020.104032.

[17] T. Taketomi, H. Uchiyama, and S. Ikeda, “Visual SLAM algorithms: a survey from 2010 to 2016,” *IPSJ Trans. Comput. Vis. Appl.*, vol. 9, no. 1, p. 16, Jun. 2017, doi: 10.1186/s41074-017-0027-2.

[18] R. Mur-Artal and J. D. Tardós, “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras,” *IEEE Trans. Robot.*, vol. 33, no. 5, pp. 1255–1262, Oct. 2017, doi: 10.1109/TRO.2017.2705103.

[19] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, “An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics,” *Intell. Ind. Syst.*, vol. 1, no. 4, pp. 289–311, 2015, doi: 10.1007/s40903-015-0032-7.

[20] T. Pire, T. Fischer, G. Castro, P. De Cristóforis, J. Civera, and J. Jacobo Berlles, “S-PTAM: Stereo Parallel Tracking and Mapping,” *Robot. Auton. Syst.*, vol. 93, no. Complete, pp. 27–42, 2017, doi: 10.1016/j.robot.2017.03.019.

[21] M. Labbé and F. Michaud, “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation: LABBÉ AND MICHAUD,” *J. Field Robot.*, vol. 36, no. 2, pp. 416–446, Mar. 2019, doi: 10.1002/rob.21831.

[22] A. Nuchter, M. Bleier, J. Schauer, and P. Janotta, “IMPROVING GOOGLE’S CARTOGRAPHER 3D MAPPING BY CONTINUOUS-TIME SLAM,” *ISPRS - Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.*, vol. XLII-2/W3, pp. 543–549, Feb. 2017, doi: 10.5194/isprs-archives-XLII-2-W3-543-2017.

[23] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, “SegMatch: Segment based loop-closure for 3D point clouds,” *2017 IEEE Int. Conf. Robot. Autom. ICRA*, pp. 5266–5272, May 2017, doi: 10.1109/ICRA.2017.7989618.

[24] A. K. Guruji, H. Agarwal, and D. K. Parsediya, “Time-efficient A* Algorithm for Robot Path Planning,” *Procedia Technol.*, vol. 23, pp. 144–149, 2016, doi: 10.1016/j.protcy.2016.03.010.

[25] C. Saranya, K. K. Rao, M. Unnikrishnan, Dr. V. Brinda, V. R. Lalithambika, and M. V. Dhekane, “Real Time Evaluation of Grid Based Path Planning Algorithms: A comparative study,” *IFAC Proc. Vol.*, vol. 47, no. 1, pp. 766–772, Jan. 2014, doi: 10.3182/20140313-3-IN-3024.00050.

[26] S. Choueiry, M. Owayjan, H. Diab, and R. Achkar, “Mobile Robot Path Planning Using Genetic Algorithm in a Static Environment,” in *2019 Fourth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA)*, Jul. 2019, pp. 1–6. doi: 10.1109/ACTEA.2019.8851100.

[27] S. Lavalle and J. Kuffner, “Rapidly-Exploring Random Trees: Progress and Prospects,” *Algorithmic Comput. Robot. New Dir.*, Jan. 2000.

[28] D. Fox, W. Burgard, and S. Thrun, “The dynamic window approach to collision avoidance,” *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997, doi: 10.1109/100.580977.

[29] Z. Yongzhe, B. Ma, and C. K. Wai, “A Practical Study of Time-Elastic-Band Planning Method for Driverless Vehicle for Auto-parking,” in *2018 International Conference on Intelligent Autonomous Systems (ICoIAS)*, Mar. 2018, pp. 196–200. doi: 10.1109/ICoIAS.2018.8494025.

[30] B. Cybulski, A. Wegierska, and G. Granosik, “Accuracy comparison of navigation local planners on ROS-based mobile robot,” in *2019 12th International Workshop on Robot Motion and Control (RoMoCo)*, Jul. 2019, pp. 104–111. doi: 10.1109/RoMoCo.2019.8787346.

[31] D. Fox, J. Ko, K. Konolige, B. Limketkai, D. Schulz, and B. Stewart, “Distributed Multirobot Exploration and Mapping,” *Proc. IEEE*, vol. 94, no. 7, pp. 1325–1339, Jul. 2006, doi: 10.1109/JPROC.2006.876927.

[32] K. Konolige *et al.*, “Centibots: Very Large Scale Distributed Robotic Teams,” in *Experimental Robotics IX*, Berlin, Heidelberg, 2006, pp. 131–140. doi: 10.1007/11552246_13.

[33] M. Pfingsthorn, B. Slamet, and A. Visser, “A Scalable Hybrid Multi-robot SLAM Method for Highly Detailed Maps,” in *RoboCup 2007: Robot Soccer World Cup XI*, Berlin, Heidelberg, 2008, pp. 457–464. doi: 10.1007/978-3-540-68847-1_48.

[34] S. Thrun *et al.*, “A system for volumetric robotic mapping of abandoned mines,” in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, Sep. 2003, vol. 3, pp. 4270–4275 vol.3. doi: 10.1109/ROBOT.2003.1242260.

[35] C. Marshal Revanth, D. Saravanakumar, R. Jegadeeshwaran, and G. Sakthivel, “Simultaneous Localization and Mapping of Mobile Robot using GMapping Algorithm,” in *2020 IEEE International Symposium on Smart Electronic Systems (iSES) (Formerly iNiS)*, Dec. 2020, pp. 56–60. doi: 10.1109/iSES50453.2020.00024.

[36] S. Saat, W. Rashid, M. Tumari, and M. Saealal, “HECTORSLAM 2D MAPPING FOR SIMULTANEOUS LOCALIZATION AND MAPPING (SLAM),” *J. Phys. Conf. Ser.*, vol. 1529, p. 042032, Apr. 2020, doi: 10.1088/1742-6596/1529/4/042032.

[37] M. A. Haseeb, J. Guan, D. Ristić-Durrant, and A. Gräser, “DisNet: A novel method for distance estimation from monocular camera,” p. 6.

[38] A. O’ Riordan, T. Newe, D. Toal, and G. Dooly, *Stereo Vision Sensing: Review of existing systems*. 2018. doi: 10.1109/ICSensT.2018.8603605.

[39] C. Jing, J. Potgieter, F. Noble, and R. Wang, “A comparison and analysis of RGB-D cameras’ depth performance for robotics application,” in *2017 24th International Conference on Mechatronics and Machine Vision in Practice (M2VIP)*, Nov. 2017, pp. 1–6. doi: 10.1109/M2VIP.2017.8211432.

[40] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, “ORB: An efficient alternative to SIFT or SURF,” in *2011 International Conference on Computer Vision*, Nov. 2011, pp. 2564–2571. doi: 10.1109/ICCV.2011.6126544.

[41] D. Galvez-López and J. D. Tardos, “Bags of Binary Words for Fast Place Recognition in Image Sequences,” *IEEE Trans. Robot.*, vol. 28, no. 5, pp. 1188–1197, Oct. 2012, doi: 10.1109/TRO.2012.2197158.

[42] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, Sep. 2013, doi: 10.1177/0278364913491297.

[43] “The EuRoC micro aerial vehicle datasets - Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, Roland Siegwart, 2016.” https://journals-sagepub-com.myaccess.library.utoronto.ca/doi/10.1177/0278364915620033 (accessed Jan. 24, 2022).

[44] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2012, pp. 573–580. doi: 10.1109/IROS.2012.6385773.

[45] J. Engel, J. Stückler, and D. Cremers, “Large-scale direct SLAM with stereo cameras,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2015, pp. 1935–1942. doi: 10.1109/IROS.2015.7353631.

[46] T. Whelan, R. F. Salas-Moreno, B. Glocker, A. J. Davison, and S. Leutenegger, “ElasticFusion: Real-time dense SLAM and light source estimation,” *Int. J. Robot. Res.*, vol. 35, no. 14, pp. 1697–1716, 2016, doi: 10.1177/0278364916669237.

[47] T. Whelan, M. Kaess, H. Johannsson, M. Fallon, J. J. Leonard, and J. McDonald, “Real-time large-scale dense RGB-D SLAM with volumetric fusion,” *Int. J. Robot. Res.*, vol. 34, no. 4–5, pp. 598–626, 2015, doi: 10.1177/0278364914551008.

[48] C. Kerl, J. Sturm, and D. Cremers, “Dense visual SLAM for RGB-D cameras,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Nov. 2013, pp. 2100–2106. doi: 10.1109/IROS.2013.6696650.

[49] F. Endres, J. Hess, J. Sturm, D. Cremers, and W. Burgard, “3-D Mapping With an RGB-D Camera,” *IEEE Trans. Robot.*, vol. 30, no. 1, pp. 177–187, Feb. 2014, doi: 10.1109/TRO.2013.2279412.

[50] T. Morris and T. Morris, “Stereo images, laser data and wheel odometry: for testing and evaluation of stereo visual odometry and visual SLAM algorithms,” *Research Data*

Australia. https://researchdata.edu.au/stereo-images-laser-slam-algorithms/453751 (accessed Jan. 27, 2022).

[51] J. Shi and Tomasi, "Good features to track," in *1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 1994, pp. 593–600. doi: 10.1109/CVPR.1994.323794.

[52] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, Nov. 2004, doi: http://dx.doi.org/10.1023/B:VISI.0000029664.99615.94.

[53] G. R. Bradski and A. Kaehler, *Learning OpenCV: computer vision with the OpenCV library*, 1. ed., [Nachdr.]. Beijing: O'Reilly, 2011.

[54] K. Konolige, "Small Vision Systems: Hardware and Implementation," in *Robotics Research*, Y. Shirai and S. Hirose, Eds. London: Springer London, 1998, pp. 203–212. doi: 10.1007/978-1-4471-1580-9_19.

[55] M. Labbé and F. Michaud, "Appearance-Based Loop Closure Detection for Online Large-Scale and Long-Term Operation," *IEEE Trans. Robot.*, vol. 29, no. 3, pp. 734–745, Jun. 2013, doi: 10.1109/TRO.2013.2242375.

[56] M. Fallon, H. Johannsson, M. Kaess, and J. J. Leonard, "The MIT Stata Center dataset," *Int. J. Robot. Res.*, vol. 32, no. 14, pp. 1695–1699, Dec. 2013, doi: 10.1177/0278364913509035.

[57] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 2, pp. 239–256, Feb. 1992, doi: 10.1109/34.121791.

[58] R. Vincent, B. Limketkai, and M. Eriksen, "Comparison of indoor robot localization techniques in the absence of GPS," Orlando, Florida, Apr. 2010, p. 76641Z. doi: 10.1117/12.849593.

[59] "Ceres Solver — A Large Scale Non-linear Optimization Library." http://ceres-solver.org/ (accessed Jan. 24, 2022).

[60] R. Kümmerle *et al.*, "On measuring the accuracy of SLAM algorithms," *Auton. Robots*, vol. 27, no. 4, pp. 387–407, 2009, doi: 10.1007/s10514-009-9155-6.

[61] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2D LIDAR SLAM," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1271–1278. doi: 10.1109/ICRA.2016.7487258.

[62] B. Douillard *et al.*, “On the segmentation of 3D LIDAR point clouds,” in *2011 IEEE International Conference on Robotics and Automation*, May 2011, pp. 2798–2805. doi: 10.1109/ICRA.2011.5979818.

[63] “Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography: Communications of the ACM: Vol 24, No 6.” https://dl-acm-org.myaccess.library.utoronto.ca/doi/abs/10.1145/358669.358692 (accessed Jan. 24, 2022).

[64] A. Geiger, P. Lenz, and R. Urtasun, “Are we ready for autonomous driving? The KITTI vision benchmark suite,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, Jun. 2012, pp. 3354–3361. doi: 10.1109/CVPR.2012.6248074.

[65] B. Gao, H. Lang, and J. Ren, “Stereo Visual SLAM for Autonomous Vehicles: A Review,” in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, Oct. 2020, pp. 1316–1322. doi: 10.1109/SMC42975.2020.9283161.

[66] I. Noreen, A. Khan, H. Ryu, N. L. Doh, and Z. Habib, “Optimal path planning in cluttered environment using RRT*-AB,” *Intell. Serv. Robot.*, vol. 11, no. 1, pp. 41–52, Jan. 2018, doi: 10.1007/s11370-017-0236-7.

[67] N. Ragot, R. Khemmar, A. Pokala, R. Rossi, and J.-Y. Ertaud, “Benchmark of Visual SLAM Algorithms: ORB-SLAM2 vs RTAB-Map,” Colchester, United Kingdom, Jul. 2019. doi: 10.1109/EST.2019.8806213.

[68] B. Hernández and E. Giraldo, “A Review of Path Planning and Control for Autonomous Robots,” in *2018 IEEE 2nd Colombian Conference on Robotics and Automation (CCRA)*, Nov. 2018, pp. 1–6. doi: 10.1109/CCRA.2018.8588152.

[69] M. Korkmaz and A. Durdu, “Comparison of optimal path planning algorithms,” in *2018 14th International Conference on Advanced Trends in Radioelecrtronics, Telecommunications and Computer Engineering (TCSET)*, Feb. 2018, pp. 255–258. doi: 10.1109/TCSET.2018.8336197.

[70] B. P. Gerkey and K. Konolige, “Planning and control in unstructured terrain,” 2008.

[71] G. Nagib and W. Gharieb, *Path planning for a mobile robot using genetic algorithms*. 2004, p. 189. doi: 10.1109/ICEEC.2004.1374415.

[72] P. E. Hart, N. J. Nilsson, and B. Raphael, “A Formal Basis for the Heuristic Determination of Minimum Cost Paths,” *IEEE Trans. Syst. Sci. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968, doi: 10.1109/TSSC.1968.300136.

[73] Y. Hu and S. X. Yang, “A knowledge based genetic algorithm for path planning of a mobile robot,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004*, Apr. 2004, vol. 5, pp. 4350-4355 Vol.5. doi: 10.1109/ROBOT.2004.1302402.

[74] I. Noreen, A. Khan, K. Asghar, and Z. Habib, “A Path-Planning Performance Comparison of RRT*-AB with MEA* in a 2-Dimensional Environment,” *Symmetry*, vol. 11, p. 945, Jul. 2019, doi: 10.3390/sym11070945.

[75] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, and T. Bertram, “Trajectory modification considering dynamic constraints of autonomous robots,” in *ROBOTIK 2012; 7th German Conference on Robotics*, May 2012, pp. 1–6.

[76] C. Rösmann, W. Feiten, T. Wösch, F. Hoffmann, and T. Bertram, “Efficient trajectory optimization using a sparse model,” in *2013 European Conference on Mobile Robots*, Sep. 2013, pp. 138–143. doi: 10.1109/ECMR.2013.6698833.

[77] “eband_local_planner - ROS Wiki.” http://wiki.ros.org/eband_local_planner (accessed Jan. 28, 2022).

[78] J. J. Johnson, L. Li, F. Liu, A. H. Qureshi, and M. C. Yip, “Dynamically Constrained Motion Planning Networks for Non-Holonomic Robots,” *ArXiv200805112 Cs Eess*, Aug. 2020, Accessed: Jan. 24, 2022. [Online]. Available: http://arxiv.org/abs/2008.05112

[79] “A Realistic RoboCup Rescue Simulation Based on Gazebo,” *springerprofessional.de*. https://www.springerprofessional.de/en/a-realistic-robocup-rescue-simulation-based-on-gazebo/7383196 (accessed Jan. 24, 2022).

[80] Z. Zivkovic, O. Booij, B. Krose, E. A. Topp, and H. Christensen, “From Sensors to Human Spatial Concepts: An Annotated Data Set,” *IEEE Trans. Robot.*, vol. 24, pp. 501–505, May 2008, doi: 10.1109/TRO.2008.918046.

[81] “Jackal,” *GitHub*. https://github.com/jackal (accessed Apr. 13, 2022).

[82] “urdf - ROS Wiki.” http://wiki.ros.org/urdf (accessed Jan. 30, 2022).

[83] D. Oram, “Rectification for any epipolar geometry,” 2001. doi: 10.5244/C.15.67.

[84] A. Davies and P. Fennessy, “Chapter 6 - Digital image processing,” in *Digital Imaging for Photographers (Fourth Edition)*, A. Davies and P. Fennessy, Eds. Oxford: Focal Press, 2001, pp. 93–141. doi: 10.1016/B978-0-240-51590-8.50007-X.

[85] “move_base - ROS Wiki.” http://wiki.ros.org/move_base (accessed Jan. 24, 2022).

[86] “explore_lite - ROS Wiki.” http://wiki.ros.org/explore_lite (accessed Apr. 13, 2022).

[87] “NumPy.” https://numpy.org/ (accessed Apr. 13, 2022).

[88] “map_server - ROS Wiki.” http://wiki.ros.org/map_server (accessed Jan. 24, 2022).