2D SLAM and Path Planning for Indoor Search and Rescue Using Multiple Mobile Robots

Saqib Mehmood and Bing Qiao
[*]College of Astronautics
Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China

***Abstract*:** Robot mapping and exploration is essential to many robotic applications such as search and rescue operations in disaster scenarios, warehouse management, service robotics, patrolling and autonomous driving. With recent advances in robot navigation and sensor compactness, single robot systems can accurately model the environment and perform complex independent navigation tasks. On the other hand, multi-robot systems can speed up mapping and exploration tasks in emergencies, such as rescue missions, by using distributed sensors, thereby increasing the range of exploration tasks to the extent which is not possible with a single robot. This paper studies mapping and path planning for multiple mobile robots in an indoor environment. Combined with map merging and path planning algorithm, this paper briefly compares four SLAM algorithms (G-mapping, Hector-SLAM, Karto-SLAM and Cartographer) in indoor environment using pose error and map alignment metrics. Furthermore, real-time path planning is studied to test the mapping results. A* algorithm is used for global path planning to optimize the track, and the DWA algorithm is adopted for local path planning to avoid obstacles. Experimental results validate the SLAM, map merging, and path planning algorithms in a simulated environment. This paper's results may help researchers quickly select appropriate algorithms to build Multi-Robot SLAM systems according to their demands.

***Keywords*:** robot mapping, SLAM, Path Planning, multi-robot system.

I. INTRODUCTION

Simultaneous Localization and Mapping is an essential skill for a robot to navigate in unknown environments. The SLAM problem is the backbone for the modern autonomous driving industry and is used for many applications including warehouse robotics, search and rescue and infrastructure maintenance. Single robot SLAM has been extensively studied over the last 20 years and many solutions to the problem have been proposed. Many of these techniques formulate the SLAM problem with probabilistic filters such as the Kalman filter or information filters where the robot poses and features are stored in a large state vector which is continuously updated as the robot explores the area and new landmarks are added to the vector [1]-[2]. The landmarks can be represented as simple points such as corners, or as defined features such as lines, planes and in the case of cameras as keypoint features that are extracted from the camera images. The complexity of the SLAM problem increases with the type of environment (static and dynamic), and kind of features that are used to determine the environment [3]-[5]. For mapping in indoor environments, a common method is to represent the environment as metric maps, particularly occupied grid maps, where the obstacles and navigable spaces are represented in the grid as occupied or unoccupied cells. SLAM with occupancy grid maps has been presented as Fast SLAM, which uses a Rao-Blackwellized particle filter [6]. Other methods include representing the environment as features, e.g., lines, planes or point features [7]-[9]. Recently graph based approaches to solving the problem are gaining popularity amongst SLAM researchers [10]. In graph SLAM, the robot's poses are presented as nodes and edges and the problem is divided into front and back end. While the local SLAM and map corrections are done at the front end, the global optimization is performed as a graph optimization problem at the back end. A brief overview of recent advances in SLAM is presented in [11].

In case of multi-robot systems, the SLAM problem can be regarded as a distributed problem, where each robot performs its own SLAM and their individual maps are finally merged using some method to combine all the features [12]-[14]. The problem is challenging because constraints and errors arising from individual robots will result in a corrupted map and failed localization [15]. Secondly, the computational power required to handle such a complex process of combining maps and correcting pose errors from individual robots into a global map is very high [16]. Communication between the robots is also challenging and algorithms need to be designed such that messages can be transferred seamlessly between the robots [17]-[18].

[19] Proposed path prediction planning based on the artificial potential field to improve obstacle avoidance. [20] Combined the Q-learning algorithm with the deep learning algorithm for path planning, which enabled robots to make reasonable walking paths under complex environmental conditions. [21] Applied an improved path planning algorithm to unmanned underwater survey ships, enabling quick obstacle avoidance and return to the preset route. However, these studies did not take into account the impact of the rescue environment on the SLAM algorithm. If these algorithms are directly applied to rescue robots, it may deteriorate the accuracy of path planning and even cause incorrect path planning results. At present, there are still rare systems that can combine multi robot SLAM and path planning for indoor rescue. Therefore, it is necessary to study the impact of the rescue environment on the multi robot SLAM system and path planning algorithm and evaluate and select the SLAM algorithm suitable for the rescue environment.

In this paper, we evaluated the results of some commonly used SLAM algorithms in simulated environment, tested the map merging and path planning algorithms in an indoor search and rescue environment. These experiments revealed the demerits of some algorithms and provided a benchmark for subsequent algorithm improvement.

The rest of the paper is organized as follows. Section 2 provides a brief description of four commonly used SLAM algorithms; Section 3 briefly describes the map merging algorithm for multiple mobile robots; Section 4 describes the A* and DWA algorithm for path planning; Section 5 provides and analyzes the simulation results of SLAM comparison, multi-robot mapping and path planning; Section 6 gives the conclusion.

II. Potential SLAM Techniques

In this section, a brief description of four SLAM techniques is conducted, namely: Gmapping, Hector-SLAM, Karto-SLAM, and Cartographer.

A. Gmapping

Gmapping is a laser-based SLAM algorithm as described by [22]. Furthermore, it is the most widely used SLAM package in robots worldwide. This algorithm has been proposed by Grisetti et al. and is a Rao-Blackwellized PF SLAM approach. The PF family of algorithms usually requires a high number of particles to obtain good results, which increases its computational complexity. Also, the depletion problem associated with the PF resampling process decreases the algorithm accuracy. This happens because the importance weights of particles may become insignificant. Hence, this means that there is a small probability that correct hypothesis can be eliminated.

An adaptive resampling technique has been developed in [22], which minimizes the particle depletion problem, since this process is only performed when is needed. The authors also proposed a way to compute an accurate distribution by taking into account not only the movement of the robotic platform, but also the most recent observations. This decreases the uncertainty about the robot's pose in the prediction step of the PF. As a consequence, the number of particles required is decreased since the uncertainty is lower, due to the scan matching process. In our experiments, the number of particles used by Gmapping was 30.

B. Hector-SLAM

Hector-SLAM combines a 2D SLAM system based on robust scan matching and 3D navigation technique using an inertial sensing system [23]. The authors have focused on the estimation of the robot movement in real-time, making use of the high update rate and the low distance measurement noise from modern LIDARs. The odometric information is not used, which gives the possibility to implement this approach in aerial robots like, a Quadrotor UAV or in ground robots operating in uneven terrains. On the other hand, it might have problems when only low rate scans are available and it does not leverage when odometry estimates are fairly accurate.

The 2D pose estimation is based on optimization of the alignment of beam endpoints with the map obtained so far. The endpoints are projected in the actual map and the occupancy probabilities are estimated. Scan matching is solved using a Gaussian-Newton equation, which finds the rigid transformation that best fits the laser beams with the map. In addition, a multi-resolution map representation is used, to avoid getting stuck in local minima. Finally, the 3D state estimation for the navigation filter is based on EKF. However, this is only needed when an Inertial Measurement Unit (IMU) is present, such as in the case of aerial robots. Thus, it will not be used in this work.

C. Karto-SLAM

Karto-SLAM is a graph-based SLAM approach developed by SRI International's Karto Robotics, which has been extended for ROS by using a highly-optimized and non-iterative Cholesky matrix decomposition for sparse linear systems as its solver [24]. A graph-based SLAM algorithm represents the map by means of graphs. In this case, each node represents a pose of the robot along its trajectory and a set of sensor measurements. These are connected by arcs which represent the motion between successive poses. For each new node, the map is computed by finding the spatial configuration of the nodes which are consistent with constraints from the arcs. In the Karto-SLAM version available for ROS, the Sparse Pose Adjustment (SPA) is responsible for both scan matching and loop-closure procedures [25]. The higher the number of landmarks, the more amount of memory is required. However, graph-based SLAM algorithms are usually more efficient than other approaches when maintaining a map of a large-scale environments. In the particular case of Karto-SLAM, it is extremely efficient, since it only maintains a pose graph.

D. Cartographer

Cartographer is an active approach that provides real-time SLAM in 2D and 3D across multiple platforms and sensor configurations. It is an open-source library, developed by Google since 2016, which is also a state of art algorithm. Worth to mention, Google Cartographer does not require a particle filter algorithm for mapping. It overcomes the issue of error accumulation during long iterations by pose estimation against a recent sub-map.

In 2D SLAM, the Cartographer supports running the correlative scan matcher, which is used for finding loop closure constraints with a sub-map (at the best-estimated position) referred to as frames. In detail, scan matching occurs at a recent sub-map, therefore it only depends on the recent scans. After each sub-map is finished, there are no longer new scans that could be inserted; it automatically checks all sub-maps and scans for the loop closure. A scan matcher starts to find the scan in the sub-map if the scans and the sub-maps are close enough based on the current pose estimates [26].

The conversion process from a scan into a sub-map is given in [27]. The generated sub-maps are presented in the form of a probability grid point which contains all the endpoints of beams that are closest to that grid point. Whenever a scan is inserted into the probability grid; the hits and misses are computed. Cartographer uses the Ceres scan matching approach to increase the accuracy of the scan pose in the sub-map.

III. Map Merging Algorithm

Multirobot_map_merge algorithm provides global map for multiple robots. It can merge maps from arbitrary number of robots while expects maps from individual robots as describe by [28]. In his work the author has present a novel algorithm for merging two-dimensional maps created by different robots independently without initial knowledge of relative poses of robots. The algorithm is inspired by computer vision image stitching techniques for creating photo panoramas. Presented algorithm relies only on map data represented as occupancy grids, which allows great scalability for heterogeneous multi-robot swarms and makes algorithm easily deployable with various SLAM algorithms. The map-merging algorithm was implemented as publically available ROS package and was accepted in ROS distribution.

As discussed above map merging algorithm is inspired by image stitching algorithms. Stitching algorithms are well-understood and implementations are broadly available. General concept of multi-step stitching pipeline is described in [29]. Stitching pipeline is also well established code in Open Source Computer Vision Library (OpenCV), mostly based on [29], along with [30] and others.

Algorithm used in this work is presented in [28], in proposed algorithm author mainly solve the registration part (estimating transformation between grids). Compositing part of stitching is relatively simple for occupancy grids compared to images from camera, because we don't need to compensate exposure errors, gain and other deficiencies. Registration solves the main problem of acquiring transformation between individual frames of robots and bridges the problem of merging maps with known initial positions and unknown initial positions. Algorithm 1 and 2 offers overview of the proposed algorithm.

Algorithm1. Proposed algorithm for estimating transformation between multiple occupancy grids. Uses Algorithm 3 to estimate final transformations.

```
Input: k occupancy grids
Output: for each grid: transformation between grid and global reference
frame, or value indicating transformation could not be estimated for current
grid.
 1: Procedure ESTIMATEGRIDTRANSFORM (grids)
 2:   detect ORB features (key points) for each grid
 3:   for all (i, j) pair of grids do        »compute transform for each pair
 4:     match features
 5:     n ← number of matches
 6:     if n ≤ matches threshold then
 7:       confidence ← 0
 8:     else
 9:       try find restricted affine transformation for features with RANSAC
10:       Ψ ← number if inliers in RANSAC
11:       if Transformation found then
12:         confidence ← Ψ/(8+0.3n)
13:         P(i, j) ← restricted affine transformation
14:       else
15:         confidence ← 0
16:       end if
17:     end if
18:   end for
19:   matches ← (i, j) for matches with confidence ≥ 1.0
20:   g ← (grids, matches)
21:   h ← largest connected component in g
22:   t ← maximum spanning tree in h
23:   ESTIMATEFINALTRANSFORM (t, P(i, j) □e ϵ edges if t)   »walk t
…….and compute transformation to global reference frame See Algorithm 2
24: end procedure
```

Algorithm 2. Algorithm estimating transformations to global reference frame from pairwise transformations on spanning tree.

```
Input: t maximum spanning tree on grids, P$(i,j)$ pairwise reduced affine
……    transformation in homogeneous coordinates between grids $i,j$.
Output: $T_i \forall_i \in V$ transformations to global reference frame
1: procedure ESTIMATEFINALTRANSFORMATION$(t=(V,E), P_e \forall_e \in$
$e$
2: $e$ ← edges of t sorted by discover time in breadth-first search starting
…..from grid with global reference frame
3:  $\forall T_i : T_i \leftarrow I$            » initialize transformation with identity
4:    for all $(i,j)$ in $e$ do $T_j \leftarrow T_i P(i,j)$
5:    end for
6: end procedure
```

IV. PATH PLANNING ALGORITHMS

Path planning solves three fundamental problems:

1. The robot reaches the goal position.
2. Real-time obstacle avoidance during the moving process
3. To find the optimal path to the desired goal.

A. Global Path Planning

Based on the global path planning of the grid method, the A* algorithm is used to study the path planning. The A* algorithm follows the cost function to make the robot to directionally search for the path toward the end point. The core valuation function of the A* algorithm is

$$f(n) = g(n) + h(n), \tag{1}$$

Where n the node is abstractly understood as the next target point, $f(n)$ represents the total valuation function of the current node n, $g(n)$ represents the actual cost of the starting point to the current point, and $h(n)$ represents the estimated cost of the current node to the end point. The value of $h(n)$ determines the performance of the algorithm. In A* $h(n)$ used the Manhattan distance between the two points in space. The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is as follows:

$$D_{Man\,hattan} = |x_1 - x_2| + |y_1 - y_2| \tag{2}$$

B. Local Path Planning

The DWA algorithm is selected as the main algorithm for local path planning. The DWA algorithm requires the robot to perform numerical simulation calculations on the path of the robot within a certain speed window. Thus, it is necessary to obtain the model state expression of the robot. The two-wheeled robot based on differential drive has no velocity in the -axis direction. Since the robot is at the millisecond level in each sampling period of the program execution, the motion trajectory of the robot in the two adjacent sampling periods can be approximated as a straight line. In a period of time Δ, the robot travels a small distance at speed v, and it is at an angle θ_t to the x-axis; then, the movement increments Δx and Δy of the robot on the x-axis and the y-axis can be obtained, respectively:

$$\Delta x = x + v\Delta t\, cos(\theta_t), \qquad (3)$$

$$\Delta y = y + v\Delta t\, sin(\theta_t), \qquad (4)$$

The robot's movement trajectory is then given by

$$x_{t+1} = x_t + v\Delta t\, cos(\theta_t), \qquad (5)$$

$$y_{t+1} = y_t + v\Delta t\, sin(\theta_t), \qquad (6)$$

$$\theta_{t+1} = \theta_t + \omega\Delta t, \qquad (7)$$

Where ω is the angular velocity of the robot.
When the robot is safely avoiding obstacles in navigation, the speed (v, ω) during the whole locally planned trajectory must be within the range of speeds given by

$$V_d = \{(v,\omega) \mid \sqrt{2dis(v,\omega)\dot{v}_d} \geq v_c, \sqrt{2dis(v,\omega)\omega_d} \geq \omega_c\}, \qquad (8)$$

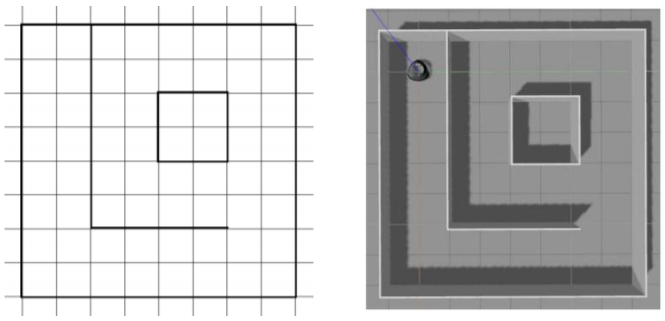
Where $dis(v,\omega)$ is the minimum distance from the current position to the point where the arc trajectory of v and ω intersects the nearest obstacle. v_c And ω_c are the current speed and angular velocity of robot while $\dot{v}_d$ is the maximum deceleration.

V. Results and discussion

In order to test the aforementioned algorithms, we carry out simulation experiments in ROS melodic running on Ubuntu 18.04 operating system using the Gazebo simulator to build the simulation environment. The virtual environment has real physical properties, and the simulation results have strong reference to the actual environment. All experiments were performed using KOBUKI robot equipped with 2D LIDAR sensor and wheel encoder.

A. SLAM Comparison

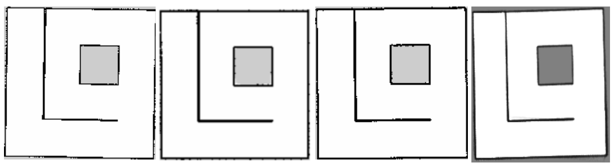
It is very important to build accurate maps for SLAM testing, as the resulting map has to be compared against the ground truth and any inaccuracies might lead to different results. For that, a map generation script was written to generate accurate Gazebo SDF descriptions of the desired map. After running the script to generate the map, the resulting map can be seen on Fig. 1b.



(a) Map in image editor (b) Generated map in gazebo

Fig. 1. Scripted map generation for gazebo.

In SLAM simulation the robot was tele-operated and the ground truth and SLAM data were collected in a rosbag file so that all SLAM algorithms get the same working data. The mapping results for the test map can be seen on fig. 2. At visual inspection, we can see that, Hector Slam and Cartographer perform better, as the noise in the walls is lower. They look straight and sharp, as opposed to Gmapping and Karto, where the walls look noisy.



(a) Gmapping (b) Hector-SLAM (c) Karto-SLAM (d) Cartographer

Fig. 2. Occupancy grid maps obtained through SLAM simulation.

To evaluate the performance of SLAM algorithms, an analysis of pose error and map alignment metric was conducted.

1. Pose Metrics

The most natural way of analyzing the poses is the squared error from the pose estimates against the ground truth. By calculating the distance from one to another and adding over time, we can get a good metric for the pose error. Fig. 3

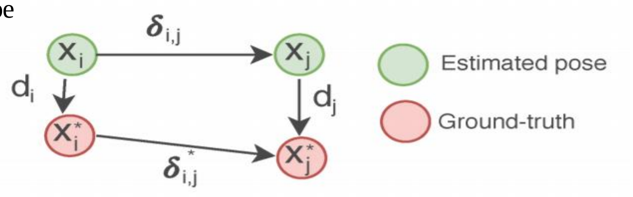


Fig. 3. Representation of metrics taken.

[31] Proposes to select the pairs (i,j) from the dataset using scan matching evaluated by a human operator. Since our dataset have the ground-truth, all possible displacements can be evaluated. Given a set of N poses, the linear displacement can then be represented by:

$$linear\ displacement = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} trans(\delta_{i,j} \ominus \delta_{i,j}^{*})^2 \quad (9)$$

$$angular\ displacement = \frac{1}{N^2}\sum_{i=1}^{N}\sum_{j=1}^{N} rot(\delta_{i,j} \ominus \delta_{i,j}^{*})^2 \quad (10)$$

The displacements $\delta_{i,j}$ and $\delta_{i,j}^{*}$ are calculated by taking the relative transformation between two poses i and j. The squared error is easier to calculate, as it relates only to the current pose. The individual pose errors are then summed across the trajectory and normalized according to the following equations:

$$linear\ sqaured\ error = \sum_{i=1}^{N}\frac{1}{N} trans(d_i)^2 \quad (11)$$

$$angular\ sqaured\ error = \sum_{i=1}^{N}\frac{1}{N} rot(d_i)^2 \quad (12)$$

The results of running the pose error metric on SLAM generated and ground-truth trajectories are given in Table 1.

Table 1
Results of map comparison using pose error metric.

	Gmapping	**Hector**	**Karto**	**Cartographer**
Linear Displacement	0.000370	**0.000244**	0.00314	0.013768
Angular Displacement	0.000036	0.000026	**0.00001**	0.000044
Linear Squared error	**0.000386**	0.001013	0.00358	0.013529
Angular Squared Error	0.000386	**0.000025**	0.00016	0.000603

In pose error metric we can see that Hector shows the best pose estimate in displacement, but Gmapping overcomes in squared error. Karto shows the good results in angular displacement but Cartographer lags behind. We can actually see why looking at the map in Fig. 2, as Cartographer's map is tilted relative to the others. This error of orientation at the start was probably what made Cartographer perform worse in the localization.

2. *Map Alignment Metric*

The second method chosen for the SLAM comparison is through map alignment. Once we have both the ground-truth and the generated maps we can then run an algorithm to align the maps properly, as suggested by [32]. First both maps are imported as a point cloud, each pixel representing a point in space as shown in Fig. 4, then ICP technique is used to align the maps. ICP or Iterative Closest Point, is a way of aligning 3D meshes. The ICP algorithm used is the one provided by [33]. It will calculate the transformation that best align the maps shown in Fig. 4a & 4b. The aligned point cloud can be seen in Fig. 4c. With the maps aligned, we then use the following equation,

$$\epsilon(map) = \frac{1}{P}\sum_{i=1}^{P} dist(p_i - p_i^{*}(p_i))^2 \quad (13)$$

To calculate the error metric, where P is the number of points in Fig. 4b, p_i represents one point in the data set and p_i^{*} is the nearest neighbor of p_i in the dataset shown on Fig. 4a. All measurements are in pixel.

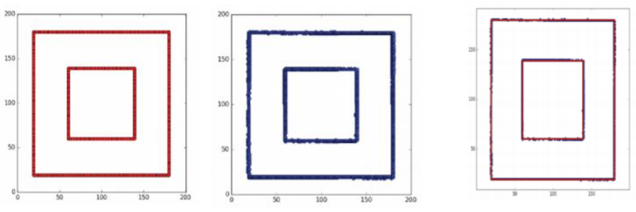

(a) Ground Truth (b) SLAM Generated (c) Aligned Maps

Fig. 4. Results of running ICP algorithm

The results of running the ICP matcher with the algorithms can be seen on Table 2. The best algorithm in all cases was Cartographer, scoring lowest. This means two things: most of the walls were placed in the correct spot and the noise is low. Gmapping is in second place, justifying the wide adoption of Gmapping in the robotic world. Karto was on third position in this test, while Hector ends up in last position.

Table 2
Pixel Squared Error

Gmapping	**Hector**	**Karto**	**Cartographer**
0.55835	0.76590	0.62870	**0.51960**

3. *Considering Rescue Environment:*

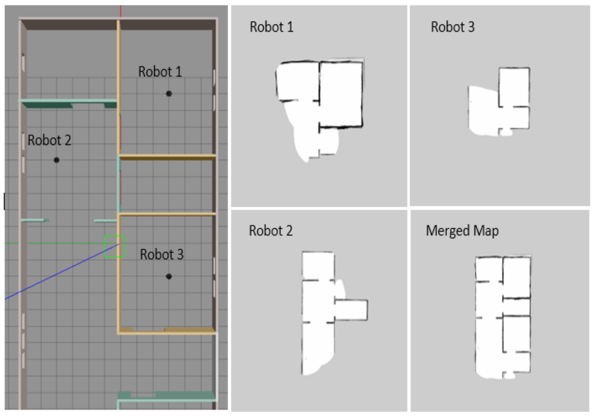
In rescue environment, there are stairs and rugged surface which make the odometer inaccurate. It means we could not choose Gmapping because it is very rely on odometer. Due to the rugged surface, IMU is also inaccurate which means Karto and Cartographer may get bad results. So, we choose Hector-SLAM for search and rescue task using multiple mobile robot. Also Hector performed very well in pose error metric and it was also close to other SLAM techniques in map alignment metric.

B. *Multiple Robot SLAM:*

To evaluate the map-merging algorithm introduced in section 3, a large house map was used in gazebo ROS. All simulated robots were KOBUKI, which formed a homogeneous exploring team. Robots were set up using KOBUKI packages available in ROS, which also configures SLAM and navigation for robots. Robots where using the Hector-SLAM package providing the SLAM algorithm and move base package part of the ROS navigation stack, providing navigation for robots. Every robot was using its own ROS namespace for topics and was using a prefix for published TF frames to allow running multiple robots under the same ROS master.

Presented merging algorithm can work with both known and unknown initial positions of robots. In our case the algorithm uses initial positions to obtain transformation between grids. Maps in this mode are not required to have any overlapping area. Simulation featured 3 robots exploring common area. Fig. 5a shows the scene used in the experiment and initial robots positions. Fig. 5b shows maps produced by Hector-SLAM and the merged map produced by a map-merging algorithm with known initial positions after mapping finished.

Map-merging algorithm presented in Section 3 can efficiently work with arbitrary number of robots. It scales well to large multi-robot systems and is designed with parallel processing in mind. The algorithm is suitable for heterogeneous multi-robot swarms and is easily deployable with various SLAM algorithms.



(a) (b)

Fig. 5. Maps produced by a multi-robot mapping in the simulator with 3 robots. The merged map (bottom right) is estimated by the map-merging node without knowledge of initial positions.

C. *Path Planning Experiment:*

After obtaining the environment map from map merging, the map was loaded under the ROS framework for path planning purposes. We use the RVIZ package under the ROS framework for path planning and navigation simulation. The constructed map is shown in Fig. 6. The cyan portion indicates the safe distance between the robot and the obstacle. The red arrow points to the target point and direction of the robot. The path planning and navigation results, are indicated by the green line. We see that the simulated path not only successfully avoids static obstacle (walls) as well as the dynamic obstacles (other robots) but also is a shortest path.

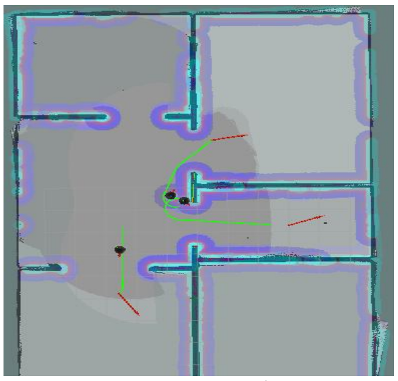


Fig. 6. Simulation results of path planning

V. CONCLUSION

In this paper, the problem of indoor search and rescue using multiple mobile robots was studied. Comparisons were done on the Gmapping, Hector-SLAM, and Cartographer algorithms for SLAM based on pose error and map alignment metric. The path planning was done by combining the A* algorithm for global path planning and the DWA algorithm for local path planning. Simulated experiments were conducted to compare and validate the results on map construction map merging and path planning. In the future, further optimization needs to be carried out in the mapping algorithms to make them more suitable to the real rescue environment.

REFERENCES

[1] Thrun, S.; Burgard, W.; Fox, D. "Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)"; *The MIT Press: Cambridge*, CA, USA, 2005.

[2] Dissanayake, M.W.M.G.; Newman, P.; Clark, S.; Durrant-Whyte, H.F.; Csorba, M. "A solution to the simultaneous localization and map building (SLAM) problem". *IEEE Trans. Robot. Autom*. 2001, 17, 229–241. doi:10.1109/70.938381..

[3] Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. SHP: "Smooth Hypocycloidal Paths with Collision-Free and Decoupled Multi-Robot Path Planning*". Int. J. Adv. Robot. Syst. 2016*, 13, 133, doi: 10.5772/63458.

[4] Huang, S.; Dissanayake, G. "Convergence and consistency analysis for extended Kalman filter based SLAM". *IEEE Trans. Robot. 2007*, 23, 1036–1049.

[5] Ravankar, A.; Ravankar, A.; Kobayashi, Y.; Emaru, T. "Symbiotic navigation in multi-robot systems with remote obstacle knowledge sharing". *Sensors 2017*, 17, 1581.

[6] Montemerlo, M.; Thrun, S.; Koller, D.; Wegbreit, B. "FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges". *In Proceedings of the IJCAI, Acapulco, Mexico,* 9–15 August 2003; pp. 1151–1156.

[7] Ravankar, A.A.; Hoshino, Y.; Ravankar, A.; Jixin, L.; Emaru, T.; Kobayashi, Y. "Algorithms and a framework for indoor robot mapping in a noisy environment using clustering in spatial and hough domains". *Int. J. Adv. Robot. Syst.* 2015, 12, 27.

[8] Wang, Y.T.; Peng, C.C.; Ravankar, A.; Ravankar, A. "A Single LiDAR-Based Feature Fusion Indoor Localization Algorithm". *Sensors 2018*, 18, 1294.

[9] Ravankar, A.; Ravankar, A.A.; Hoshino, Y.; Emaru, T.; Kobayashi, Y. "On a hopping-points svd and hough transform-based line detection algorithm for robot localization and mapping". *Int. J. Adv. Robot. Syst*. 2016, 13, 98.

[10] Thrun, S.; Montemerlo, M. "The graph SLAM algorithm with applications to large-scale mapping of urban structures". *Int. J. Robot. Res.* 2006, 25, 403–429.

[11] Cadena, C.; Carlone, L.; Carrillo, H.; Latif, Y.; Scaramuzza, D.; Neira, J.; Reid, I.; Leonard, J.J. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". *IEEE Trans. Robot.* 2016, 32, 1309–1332.

[12] Cunningham, A.; Paluri, M.; Dellaert, F. "DDF-SAM: Fully distributed SLAM using constrained factor graphs". *In Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan*, 18–22 October 2010; pp. 3025–3030.

[13] Ravankar, A.; Ravankar, A.A.; Kobayashi, Y.; Emaru, T. "On a bio-inspired hybrid pheromone signalling for efficient map exploration of multiple mobile service robots". *Artif. Life Robot.* 2016, 21, 221–231.

[14] Saeedi, S.; Trentini, M.; Seto, M.; Li, H. "Multiple-robot simultaneous localization and mapping: A review". *J. Field Robot.* 2016, 33, 3–46.

[15] Ravankar, A.; Ravankar, A.; Kobayashi, Y.; Emaru, T. "Hitchhiking robots: A collaborative approach for efficient multi-robot navigation in indoor environments". *Sensors 2017*, 17, 1878.

[16] Labbe, M.; Michaud, F. "Online global loop closure detection for large-scale multi-session graph-based SLAM". *In Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA*, 14–18 September 2014; pp. 2661–2666.

[17] Roumeliotis, S.I.; Bekey, G.A. "Distributed multi-robot localization". *In Distributed Autonomous Robotic Systems 4; Springer: Tokyo, Japan*, 2000; pp. 179–188.

[18] Leung, K.Y.; Barfoot, T.D.; Liu, H.H. "Distributed and decentralized cooperative simultaneous localization and mapping for dynamic and sparse robot networks". *In Proceedings of the 2011 IEEE International Conference on Robotics and Automation,*

Shanghai, China, 9–13 May 2011; pp. 3841–3847.

[19] H. Zhang, X. Zhang, and Q. Wang, "Path planning based on path prediction artificial potential field method for automatic following trolley," *Computer Measurement and Control*, vol. 27, no. 1, pp. 237–240, 2019.

[20] Z. Liu, S. Jiang, W. Yuan, and C. Shi, "Robot path planning based on deep Q-learning," *Measurement and Control Technology*, vol. 38, no. 7, pp. 24–28, 2019.

[21] B. Yu, X. Chu, C. Liu, H. Zhang, and Q. Mao, "Path planning method for unmanned waterway survey ships based on improved A∗ algorithm," *Geomatics and Information Science of Wuhan University*, vol. 44, no. 8, pp. 1258–1264, 2019.

[22] G. Grisetti, C. Stachniss, W. Burgard. "Improved Techniques for Grid Mapping with Rao-Blackwellized Particle Filters*", In Trans. on Robotics,* 23(1), Feb. 2007.

[23] S. Kohlbrecher, J. Meyer, O. Von Stryk, U. Klingauf. "A Flexible and Scalable SLAM System with Full 3D Motion Estimation", *In the Int. Symp. On Safety, Security and Rescue Robotics (SSRR),* Nov. 2011.

[24] R. Vincent, B. Limketkai, M. Eriksen. "Comparison of indoor robot localization techniques in the absence of GPS*", In Proc. of SPIE: Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XV of Defense, Security, and Sensing Symposium,* April 2010.

[25] K. Konolige, G. Grisetti, R. Kummerle, B. Limketkai, R. Vincent, ¨ "Efficient Sparse Pose Adjustment for 2D Mapping", *In Proc. of Int. Conf. on Intelligent Robots and Systems (IROS),* Oct. 2010.

[26] R. Yagfarov, M. Ivanou, and I. Afanasyev, "Map Comparison of Lidarbased 2D SLAM Algorithms Using Precise Ground Truth," 2018 15th *International Conference on Control, Automation, Robotics and Vision, ICARCV* 2018, pp. 1979–1983, 2018.

[27] W. Hess, D. Kohler, H. Rapp, et al., "Real-time loop closure in 2D LIDAR SLAM," *Proceedings - IEEE International Conference on Robotics and Automation,* vol. 2016-June, pp. 1271–1278, 2016.

[28] HÖRNER, Jiri. "Map-merging for multi-robot system" . 2016. *Bachelor thesis. Charles University, Faculty of Mathematics and Physics, Department of Theoretical Computer Science and Mathematical Logic.* Thesis supervisor Obdržálek, David.

[29] Matthew Brown and David G. Lowe. "Automatic panoramic image stitching using invariant features". *International Journal of Computer Vision*, 74(1):59–73, 2006.

[30] Richard Szeliski. "Image alignment and stitching: A tutorial. Technical Report" MSR-TR-2004-92, *Microsoft Research*, October 2004.

[31] KÜMMERLE, R. et al. "On measuring the accuracy of slam algorithms". *Autonomous Robots, Springer*, v. 27, n. 4, p. 387, 2009.

[32] SANTOS, J. M.; PORTUGAL, D.; ROCHA, R. P. "An evaluation of 2d slam techniques available in robot operating system". *In: IEEE. Safety, Security, and Rescue Robotics (SSRR), 2013 IEEE International Symposium on. [S.l.],* 2013. p. 1–6.

[33] FLANNIGAN, C. icp. [S.l.]: GitHub, 2019, commit 167cc4a.