Multi-robot Maze Surveying and Action

Tai Wan Kim, Joseph Hwang, Jeff Yang, Kamakshi Moparthi, and Andrea Robang

***Abstract*— We attempt to build a simple multi-robot system capable of exploring an unknown maze-like environment and carrying out a complex set of actions. The project consists of different modules such as multi-robot mapping, collision avoidance, object detection, and path finding. While we were only partially successful in each of these modules, we report our current results and will continue to work to see if some of the challenges we face now can be addressed.**

I. Introduction

Multi-robot systems integrate several robots to accomplish sophisticated tasks. One notable use case is smart factories such as the Amazon Robotics Fulfillment Center (ARFC). Amazon warehouses are filled with shelves associated with a specific QR code. Amazon KIVA robots keep navigating through the warehouse. Each KIVA robot is assigned a set of selves which it needs to carry to the fulfillment associate responsible for packing the order. With the help of a QR code at the bottom of the shelves, the robot can identify the shelf it needs to carry to the associate. Aside from smart factories, there may be other use cases such as exploration/rescue robots for hazardous environments.

The project aims to build a simple multi-robot system with the above use cases in mind. Our multi-robot system is designed to use multiple robots to i) explore and map an unknown environment; ii) detect objects of interest; iii) and plan a multi-robot path for efficient coverage of all locations of interest. The project designs and integrates several components: *multi-robot mapping, collision avoidance, path finding, and object detection*.

While we implemented each of these modules, the results weren't as effective and accurate as we expected. We will continue to work over the weekend to see if we can achieve better results.

II. Related Works

Our project integrates several components including multi-robot exploration, collision avoidance, efficient path finding, and object detection. In particular, *map merging/sharing* and *collision avoidance* are two critical aspects of multi-robot exploration. [1] proposes a fast and accurate map merging algorithm based on Hough transformation. [2] proposes efficient communication of maps leveraging pose-graph representation over the standard occupancy grid representation. [3] introduces the step-forward dynamic moving approach for collision avoidance, incorporating omni-directional vision systems, automatic control, and dynamic programming.

A notable prior work similar to our approach is [4], where the objective is to deploy multiple robots to explore and map an unknown environment and locate fire sources in an efficient manner. [4]'s method consists of three main components, multi-robot decision making, map merging, and fire source detection. To avoid multiple robots surveying the same region, [4] proposes an A*-based frontier selection; a robot is assigned to a frontier based on the cost needed to reach the frontier. To avoid possible collisions, [4] also implements a collision avoidance algorithm based on the set rules of engagement. For map merging, a single global map is shared during exploration. When each robot acquires new information about the environment, it requests for and updates the global map. Lastly, a temperature and chemical sensor is used to locate fire sources.

While our work has been inspired by [4] in its overall scheme, the specific method used for each component is made easier. Instead of using a frontier-based exploration, we resort to using PID-based wall-following or random walking robots to cover the given environment. We also adopt less sophisticated rules of engagement for collision avoidance such as one robot stepping aside and making way for another robot in a narrow corridor. For object detection, we assume that our object of interest is in specific color and relies on RGB camera. However, we also consider novel aspects such as effective path finding after the objects of interest have been found.

III. Problem Statement

Our problem consists of two main stages, the exploration stage and the action stage. We assume that both phases take place in a maze-like environment with straight, as opposed to curved, walls (e.g., a map from PA3). During the exploration stage, the goal is, given each robot's initial pose and dimensions (width and height) of the environment, i) publish a single occupancy grid representation of the map; and ii) the list of poses of every detected object. Two detection robots are used for exploration. Next, during the action phase, the goal is to traverse all coordinates of the detected objects using two action robots without overlapping (same object location being visited twice by different robots) or collision.

Modules	Inputs	Outputs
Multi-robot Mapping	Starting pose of each robot. The dimensions of the map.	A single occupancy grid representation of the explored map.
Object Detection	--	List of poses of each object.
Path Finding	List of coordinates to be traversed.	Ordered list of coordinates to be traversed.

Table 1: Problem statement in terms of input and output

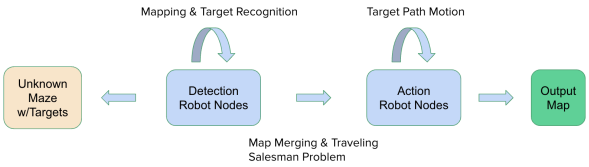

Figure 2: Implementation Overview

IV. Proposed Approach

A. Multi-robot Mapping

To map out the world, we use two (potentially several) detection robots to run a wall-following/random walk algorithm to move around different area of the grid. Each robot has its own perception and will run the algorithm separately on their capacity, i.e., each detection robot runs a separate set of ray tracing algorithm. For any of the detection robot, after it map out over 90% of the map area (i.e., less than 10% of the grid is -1) it will stop the mapping operation (either return to starting point/desired location then become an action robot, or just move to the exit/out of the grid). Then the separate map will be merged by taking an average. To be more precise, if any location has -1, that point will be disregarded instead of being averaged to the result. Otherwise, we simply take the average value from different detection robots for each location. According to probability theory, each grid point has 90% chance of being covered by one robot. If we have two robots, that grid point has 1% (1-90%*90%) probability of not being covered by both robots, which means 99% of the times it will be covered by at least one of the robots. We can also adjust this value (lower) and increase the number of robots for this probability analysis. Afterwards, the merged map will be sent/published to the action robot.

B. Collision Avoidance

Since this project involved multiple robot agents operating at the same time in the same map, we needed to take into consideration the possibility of collisions. This logic of collision handling was not a simple problem as there were many variable factors. We approached the problem from an avoidance angle, handling the collision possibilities prior to an actual collision. This required the following components:
i) detection of possible collision; ii) actions to avoid possible collision; c) integration to prior state for purpose of path or goal following.

First, we considered several potential methods of detection of moving targets and possible collisions:

- A constantly updated dictionary that would contain the locations of all robots.
    - If the robot detects it is in the same area (cell) as another robot, then take avoidance action.
- Moving Target recognition using laser scan data.
    - Check the difference in perceived distance closing against robot linear velocity.
    - If the difference is much greater than a 1:1 ratio, then there is a moving frame ahead.
- Object recognition using RGB-D camera to detect action and detection robots.
    - If a robot is detected, then perform appropriate avoidance action.

 # Pseudocode for collision avoidance

while robot is moving*:*

if perceived change in distance ahead over time is 1.5x greater than linear velocity*:*

Sidestep

Move forward until initial distance to moving target detected past.

Sidestep back to initial corridor spacing in axis.

Figure 3: Pseudocode and illustration of side-stepping logic

We decided to attempt the Moving Target recognition first as the main collision detection strategy for both action and detection robots. The avoidance actions were designed to be control-based using state changes and conceptually similar to what we humans would likely do during hallway encounters, side-step and walk forward while hugging the walls until the event has passed (See Figure 2).

The assumptions are that the walls are straight like in a business use-case of warehouse / grocery store aisles. And the directions of encounters are relatively head-on due to the binary direction influenced by simple mazes with straight not curved walls. In the case of curved walls, the avoidance action could transition to PID wall-following in the side-stepping/wall-hugging phase. We also attempted a simpler solution to collision avoidance for a simplified version of our project, with 2 robots acting as both detection and action agents, using PID control with different gain and goal difference. In this case, the robots moving left and right of the maze walls, will encounter each other typically once depending on the complexity of the maze.

C. Path Finding

The pathfinding implementation is used for individual robots in the maze setting. The inputs for this problem are the robot's starting location, the coordinates of the goal targets it has to traverse, and the occupancy grid created by the detection robots. The main consideration for creating a path for the robot to follow is that we want to find as optimal of a solution as possible. The implementation relies on a modified form of A* which first goes through the list of goal targets to find the closest one to the current location of the robot by calculating Euclidean distance. We then find the path to that target by iterating over the neighboring cells of the current cell/location of the robot with 8 possible directions (4 cardinal directions and 4 diagonals). With those possible actions, we look at if they are traversable based on the confines of the occupancy grid and the cost to getting to each possible cell, using previous cost to get to the previous cell and the heuristic which is calculated here as Euclidean distance from the possible cell to the target. As we traverse possible cells, we are always moving first to the cells calculated to have the

smallest cost, which helps optimize the search. Iterating over these steps should provide a path with the cells traversed to get to the goal cell. After finding that path, we remove that goal target from the list of points to traverse to and continue this process until all targets have been traversed.

D. Object Detection

To locate objects of certain color, we subscribe to and leverage information from both the RGB camera and depth camera (point cloud). First, we read the RGB image and if present, locate the colored object using the OpenCV package. The pixel coordinates of the located object are stored. Next, we read the point cloud 3D coordinates corresponding to the saved pixel coordinates. After applying transformation from the camera's reference frame to global reference frame, the coordinates are appended to a list of poses of identified objects, which is published as a PoseArray type message. For simplicity, we disregard the z coordinate. To account for possible errors in camera readings and prevent the same object being added multiple times to the list, we append an object only if the detected object is not within 1m distance of any other objects already in the list.

E. Integration

The plan for integration is to combine the individual modules created into a structural passing of information:

- Map and maze instantiation to open availability to robot topics (`/robot_1/odom etc…`).
- Detection node running PID control - left and right of the maze entrance/starting point.
- Map merging component publishing master map to map topic.
- Object detection component publishing a list of poses of detected objects.
- Action robot logic in another python file calling planning logic from `path_planner.py` with the Traveling Salesman solving functions.
- Action robot also managing the movement according to the input Pose array from the TSP solving functions.
- Collision logic interrupting the path following state to the avoidance side-stepping state until past encounter and then return to the path following.

We plan to have 3 python files to run i) for the maze & detection robot control; ii) for object detection logic and interaction with camera; and iii) for the action robot.

V. Experimental Results

Unfortunately, we encountered more problems than we expected and couldn't build a working system. Below, we evaluate individual modules and discuss the challenges we encountered.

A. Collision Avoidance

Strategy	Times Attempted	Successful Avoidance	Success Rate
Side Stepping	5	3	0.6

Table 2: Success Rate of Side-Stepping Logic

Figure 4: Collision avoidance in a narrow corridor

We encountered difficulties with integration of the side-stepping logic into a map workspace with 4 robots as the moving target detection portion using Laser scan was very fickle and inconsistent unless the encounter was head-on potential collision in a corridor with ample width. This approach also excluded the sudden encounters that occurred around corners. We set up tests for sidestepping and the results were not very consistent due to the clause in the code requiring both "moving object detection" and within minimum range before moving. Taking away the minimum range caused the robot to maneuver as early as the laserscan logic detected motion, which potentially was farther than sensible (Table 2).

So, another option was using the grid cells to project the location of each robot and change state to opposite wall following PID within those encounter cells. This also proved to be a challenge because of the complexity of the integration with other components and was not completed at the time of this paper. The challenges to this problem are not only the collision handling and moving target recognition but also the adjustment to the original goal or path following for the action and detection robots. If an action robot is moving on a path to its targets and encounters a possible collision, we need to interrupt the behavior, pass the encounter, then resume the path. And finally, future work to extend the collision logic to the collision handling for extremely constrained corridors with no space for side-stepping was introduced in the presentation. The backtracking for a lower priority action robot. That action robot follows its path until the encounter is no longer detected (Figure 4).

Challenges and Limitations The use of laserscan for moving target detection was not very successful and object recognition was not at the state for use in moving target detection logic. It was challenging getting the laserscan moving target logic to properly recognize consistently and not operate with frequent false-positives. The sidestep maneuver was also not great for corner encounters. We tried the grid PID approach but was unsuccessful in the time writing this paper. The backing out collision logic for extremely narrow corridors was also not completed at the time of writing this paper. The collision avoidance problem especially in a constrained environment like a maze proved to be quite difficult as it had to account for many things like distances from robots to obstacles both moving and non-moving as well as different encounter patterns such as head-on, side, corner. For future work, the team would complete the logic for grid cell based detection with PID avoidance first to see if it resolved the moving object detection inconsistency of the laserscan sidestepping method.

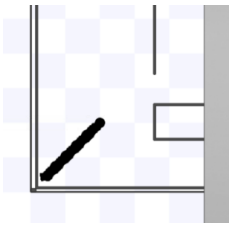

Figure 5; Illustration of Path Finding. the robot moving along target destinations along the same line but with varying coordinates (1, 1), (0.5, 0.5), (0.8, 0.8)

Then the team would analyze the performance of different encounter types. If unsuccessful, then the team would try another PID based method instead of forward direction, rather the distance between moving objects once detected, then move around centroid between the two robots. Also, collision encounters with greater than 2 robots present would be another scenario to handle. And finally, the backtracking out of narrow corridors using the path would be another future work to complete (Figure 5).

B. Path Finding

To test the path finding node, several tests were run using a 2D simulation maze environment and a set of target destinations. Some functionalities tested include the choosing of the closest node and the resulting path created. The movement/navigation along that path was also included in the code but was not the primary function of this task. Overall, the algorithm was able to find optimal paths between different nodes though it did take significantly longer for some of the coordinates of the target destinations, especially those further from the current position of the robot. The movement implemented also showed evidence of overshooting, especially as the robot progressed along its journey throughout multiple target destinations.

C. Object Detection

To evaluate our implementation for object detection, we simulated a maze-like environment with colored objects and a husarion rosbot on gazebo (Figure 6). When the robot is not moving, the estimated location of the object is fairly accurate. (Table 3)

However, performance degraded when we tried detecting objects as the robot moved via the teleop command. While our hopes were that the camera readings would be reliable enough to return consistent location of the same object, it didn't work as expected. The location estimates were inconsistent especially when the robot is rotating. For instance, in one failed attempt, the robot detected more than two objects while there were two objects spawned on the map (Table 4).

D. Integration

The team struggled with the integration of the different components into a complete work which would connect the detection phase to the action phase. This part proved to be a huge challenge as each individual component was developed with separate ideas and philosophies such as threading vs. controlling 2 robots in one node for the detection / action robots. There was a lot of complicated code in each version of the state machines of the individual components and some parts of each component had issues which prevented passing of information to the next component, such as target object list unable to move a target list to the planning phase and the action robots not having the logic to continue from the detection phase seamlessly. The future work would be to complete the integration using a consistent framework between the different components.

Trials	X Estimate (m)	Y Estimate (m)
1	1.74366143	-0.21009448
2	1.74363446	-0.19699347
3	1.77416959	-0.27793801
Mean Error	0.0891782	0.0286797

Table 3: Estimated location of an object at ((1.843, -0.205))

Actual Object Locations	(1.843, 0.021) (-2.166, 0.534)
Estimated Object Locations	(1.74469183, -0.03813495) (2.30827641, -0.99215752) (1.47820245, -1.55997539) (0.49757871, -0.91214765) (-0.74300534, -0.51198231) (-2.0786158, 0.57423039) …

Table 4: Estimate gets inaccurate when the robot is in motion.

E. Multi-robot Mapping

In this experiment, we conducted a mapping task using two robots to map the area individually and then merge the two. Robot 1 mapped the right side of the area, while Robot 2 mapped the left side. To present the map clearly, we displayed the two maps and published them on the same Rviz (Figure 7). Next, to show our merging function, we simply merge the map by robot2 into the map of robot 1 in Figure 8. The map on the lower level is the map merged as in the figure.

VI. ETHICS

There are several ethical concerns if this technology was to be used in real-life applications. Take for instance a scenario where a robot has to rescue people from fire. While robots can reduce risk taken by human rescuers, there is a dilemma of whether a robot, that may potentially have errors and be biased, can be trusted with tasks of great consequence. The robot should be tested thoroughly beforehand to prevent any unfortunate circumstances.

As robots can provide time-efficient solutions to most use cases, it is in companies' benefit to turn towards full automation. Robots replacing human workforce could result in mass unemployment. The company should act responsibly to mitigate any unwarranted social impacts and cooperate closely with government and legislature.

Furthermore, in a hybrid-setup where human monitoring is required, a small malfunction at the robot's end could be life threatening to humans working in the same environment. Such risk could be mitigated to an extent by strengthening safety

measures e.g., providing humans with safety vests and helmets. The robots could also be designed and programmed to act in a manner that mitigates risk (e.g., immediate shut down when an error is caught).

VII. Conclusion

Our project aimed to build a simple multi-robot system capable of exploring an unknown maze-like environment and carrying out a complex set of actions. The project consists of different modules such as multi-robot mapping, collision avoidance, object detection, and path finding. While we implemented each of these modules, we faced challenges we did not expect and weren't able to put together different parts as an entire system. We will continue to work on the project over the weekend and hopefully, we will be able to address some of the challenges discussed above and work to integrate different parts.

Acknowledgment

We want to thank Professor Alberto Quattrini Li for his support and guidance.

References

[1] S. Carpin, "Fast and accurate map merging for multi-robot systems," Autonomous Robots, vol. 25, no. 3, pp. 305–316, 2008. doi:10.1007/s10514-008-9097-4

[2] M. Pfingsthorn and A. Birk, "Efficiently communicating map updates with the pose graph," 2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2008. doi:10.1109/iros.2008.4651182

[3] C. Cai, C. Yang, Q. Zhu and Y. Liang, "Collision Avoidance in Multi-Robot Systems," 2007 International Conference on Mechatronics and Automation, Harbin, China, 2007, pp. 2795-2800, doi: 10.1109/ICMA.2007.4304002.

[4] A. Marjovi, J. G. Nunes, L. Marques, and A. de Almeida, "Multi-robot exploration and fire searching," 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2009. doi:10.1109/iros.2009.5354598

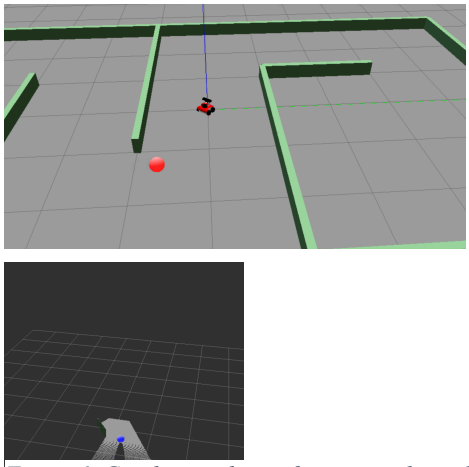

Figure 6: Gazebo simulation for testing object detection and rviz simulation for point cloud readings.

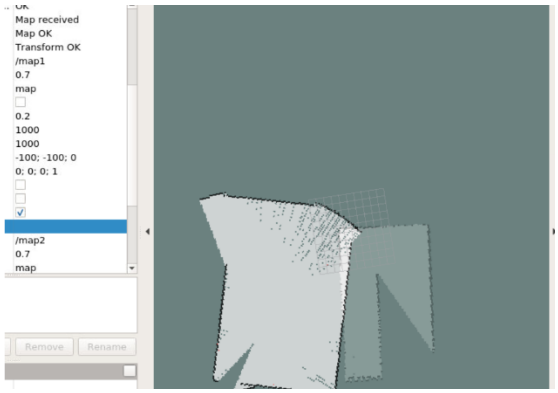

Figure 7: Mappings from two robots.

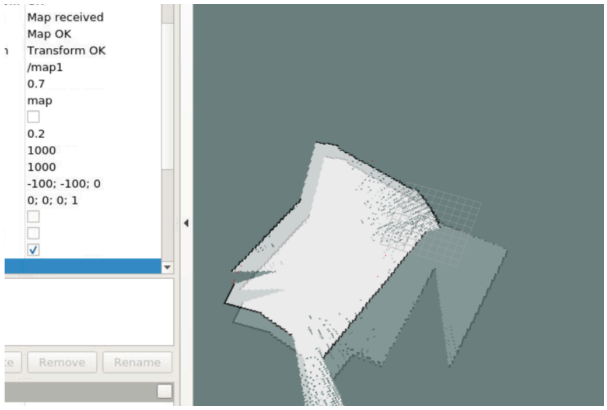

Figure 6: Merged Map