ROSbot Search and Rescue: Locating Objects of Interest in an Unknown Environment

John Chrosniak, Justin Guo, Zongyu Li, Fiji Marcelin

Abstract— **Autonomous mobile robots have become increasingly important in today's digital world. Robots are now becoming commonplace solutions for a variety of tasks, ranging from vacuuming a house to policing a city. Search and rescue missions have also become a popular task for robots, as autonomous mobile robots can navigate in unknown areas without human supervisors to help locate the object or person of interest. This can be achieved much more efficiently with swarms of robots than humans, and also mitigates the risk of any potential risk to human life in dangerous recovery settings. In this project, we demonstrate methods for autonomous mobile robots to navigate a previously unknown environment, located any potential objects of interest, and generate a visual map for a human to retrieve the object of interest.**

I. INTRODUCTION

An autonomous mobile robot (AMR) is any robot that can understand and move through its environment without being overseen directly by an operator. AMR is especially helpful when people cannot simply access some hazardous locations. When these situations arise, the use of search and rescue robots is a useful and safer alternative to the utilization of search and rescue animals. This is due to the robot's ability to be controlled by a human operator and provide real-time video and sensory data to its operator. The main metrics of performance for search and rescue robots are survivability, mobility, sensing, communicability and operability. Functions of these robots include performing tasks such as delivering medical or food supplies, removing obstructions, mapping hazardous terrain, searching for hurt or lost civilians, and extinguishing fires.

In applications where a search and rescue robot must map the terrain or search for a specific object under autonomous control, it is important for the robot to be able to localize itself in its environment. Occupancy grid mapping is a method used to help the AMR familiarize itself with its environment. In order to create the occupancy grid, the AMR must receive data from its sensors and convert it it into a grid of squares, each containing a weight that indicates the probability of that square being occupied by a solid object. Current implementations of AMR technology use light detection and ranging (LIDAR) systems to create a point cloud of data, with the AMR in the center. The data obtained from this point cloud is then used to populate the values of the occupancy grid. However, this process must be continuous, as the robot must map the environment as it is moving to find its objective.

This is where the use of a simultaneous localization and mapping (SLAM) algorithm comes into play. A SLAM algorithm allows the robot to build a map of its environment while computing its position in that environment. However, one drawback of SLAM algorithms is that the AMR's observations and map landmarks are both unknown. The AMR is creating a map as it moves, but if it encounters an object that is nearly identical to one that was previously observed, the robot may not be able to properly orient itself. This could lead to problems with localization, causing the robot to select an incorrect path due to the landmark being associated with a desirable path.

While SLAM is a useful method by which to localize the robot within its environment, this implementation of a search and rescue robot only implements an occupancy grid mapping algorithm in order to preserve system resources.

II. MOTIVATION

Autonomous robots can be used for multiple purposes in search and rescue operations, such as building the environment map, searching for people and lost objects, and carrying equipment to those in need [1]. Our motivation for this project stems from identifying important objects in search and rescue missions, such as an airplane's black recorder box.

III. METHODS

Our automated search and rescue operation consists of three major components. These can be divided up into object detection, mapping, and motion planning.

A. Object Detection

We must be able to estimate the surrounding state of the robot as we navigate through an unknown environment and look for objects of interest. The LiDAR sensor on board the ROSbot was used to detect surrounding objects to avoid collisions. Specifically, distance measurements from the LiDAR were used to construct a potential field for the robot to traverse, with obstructed areas receiving a high negative cost.

To detect objects of interest, we scanned the reference object using our robot's onboard LiDAR system by placing the robot around the object in multiple views. These reference point clouds were saved in the robot's filesystem and loaded in at runtime to compare with the robot's surroundings. The scan matching algorithm was performed by first filtering out any LiDAR points greater than 2 meters away, as objects are harder to identify with LiDAR sensors at further distances. A euclidean-based clustering algorithm provided by PCL [2] was then used to propose regions of interest that could potentially be objects of interest. The iterative closest point

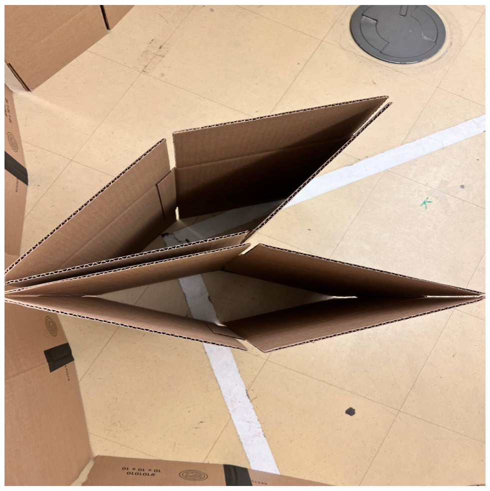


Fig. 1: The chosen object of interest. This was constructed because the object has a unique shape that is distinguishable from other objects in the 2D LiDAR plane.

(ICP) algorithm was then used to compare the prior-recorded scans of the object of interest with each cluster. If the two scans had significant overlap, then it is likely that the detected cluster is an object of interest.

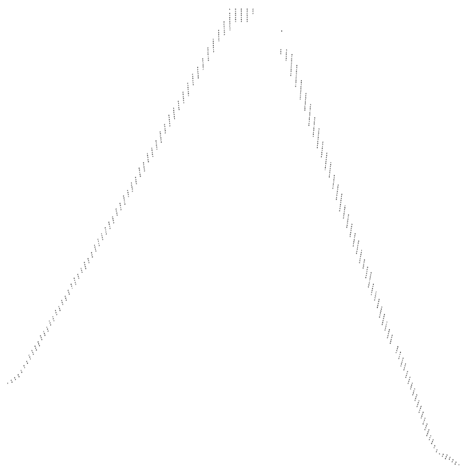
1) Reference Object's Point Cloud: 0ur reference object was two boxes of flattened cardboard that we stood upright in a "V" like shape, seen in Figure 1. We captured three point cloud distributions of the object by placing the robot towards each side of the object while running and saving the close LiDAR points of the robot. These three point cloud distributions allow us to compare to the robot's LiDAR readings for similarity in centralized clusters. Visual representations of these scans are shown in Figure 2.

2) Iterative Closest Point: The ICP algorithm was first introduced by Chen and Medioni [3], and Besl and McKay [4]. In the ICP algorithm, one point cloud, the reference, or target, is kept fixed, while the other one, the source, is transformed to best match the reference. The transformation is achieved by iteratively combining translation and rotation needed to minimize an error metric. The algorithm takes in the source point clouds, and the initial estimation of the transformation to align the target to the source. It outputs the refined transformation. Specifically, in our case, we use the error metric to estimate the object's location in the occupancy grid. The ICP algorithm consists of the following steps [5].

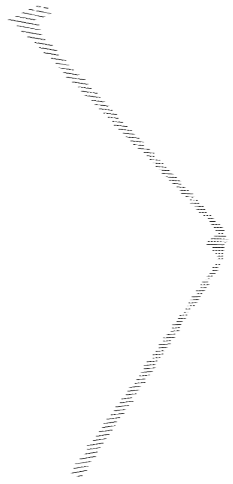
- For each point in the source point cloud, match the closest point in the reference point cloud.
- Estimate the combination of rotation and translation with a point-to-point distance metric minimization technique by using root mean square.
- Transform the source points using the obtained transformation.
- Iterate.

B. Mapping

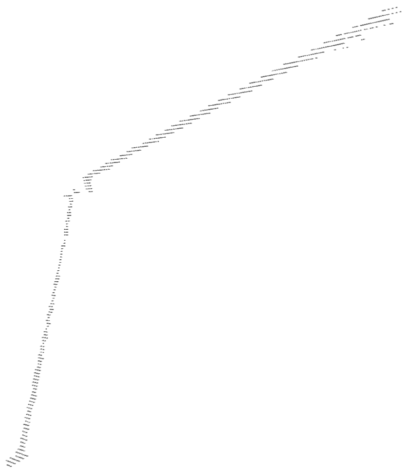
An occupancy grid was used to track obstacles within the global space of the robot operation. To perform this, we ini-



(a) LiDAR scan from the front view



(b) LiDAR scan from the left view



(c) LiDAR scan from the right view

Fig. 2: LiDAR scans of the chosen object of interest from different possible angles. Each of these scans was used as the reference object for the ICP algorithm.

tialize a grid space where each cell has an equal probability of being occupied or not occupied. After affixing the LiDAR readings to the global operation space, we collected a list of occupied cells, which came from the direct points returned by the LiDAR, as well as a list of unoccupied cells, which were computed using Bresenham's line drawing algorithm from the robot's current position to the LiDAR points' positions. The occupancy grid's cell values were then updated using log-odds probability, representing obstacle states as a sum of repeated logarithmic probability measurements.

C. Motion Planning

A potential field was used to make the robot be attractive to the goal and repellent to the obstacles. The potential field was generated using

$$U(q) = U_{att}(q) + U_{rep}(q) \tag{1}$$

where U_{att} represents the attractive potential, and U_{rep} represents the repulsive potential.

1) Attractive Force: Representing position of the goal using $q_g = (x_g, y_g)$ and the position of our robot as $q_r = (x_r, y_r)$, our attractive potential function becomes:

$$U_{att}(q) = \frac{1}{2}\xi(||q_r - q_g||)^2 \tag{2}$$

We can then write the attractive force of the goal point as:

$$F_{att}(q) = -\nabla U_{att}(q) = -\xi(q_r - q_g) \tag{3}$$

In practice, we collect the current position of the robot using the odometer's wheel encoder readings and travel to a fixed goal position. WE then tune the value of ξ until the vector of the attractive force becomes large enough to direct the robot to the goal point alongside the repulsive force.

2) Repulsive Force: The repulsive force is the representation of an obstacle pushing away the robot as if it had gravitational dynamics. Using the same representation of position $q_g = (x_g, y_g)$ as the goal and $q = (x, y)$ for any arbitrary position, we can calculate the repulsive potential function, assuming that each object has an effective repulsive range p_0 and has a function of the distance away from the object $p(q)$:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\eta(\frac{1}{p(q)} - \frac{1}{p_0})^2 & p(q) \le p_0 \\ 0 & p(q) > p_0 \end{cases} \tag{4}$$

We can then write the repulsive force of the obstacle points as:

$$F_{rep}(q) = \eta(\frac{1}{p(q)} - \frac{1}{p_0})\frac{1}{p^2(q)}\nabla p(q) \tag{5}$$

Where $\nabla p(q)$ is the unit vector pointing in the direction of the closest obstacle:

$$\nabla p(q) = \frac{q - b}{||q - b||} \tag{6}$$

In practice, we treated each individual LIDAR reading that reported a distance as an immediate obstacle. By taking the sum of the individual forces of each LIDAR point, we created the net repulsive force vector.

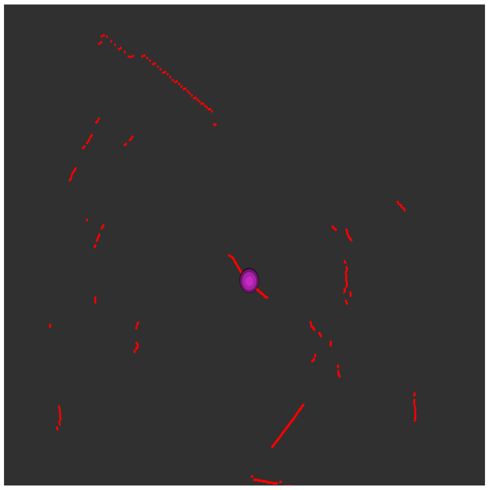


Fig. 3: Detected location (pink dot) of the object of interest overlayed with a LiDAR scan from the robot. The detected point is set to the centroid of the LiDAR cluster that has been classified as an object of interest.

3) Combining two forces: We add together the repulsive force and the attractive force to get our resulting force vector:

$$F_{total} = F_{rep} + F_{att} \tag{7}$$

We then set the linear velocity of the robot as the magnitude of the resulting force vector and the angular velocity of the robot as the angle of the resulting force vector.

IV. RESULTS

A. Object Detection

Our ICP algorithm was able to detect the object when the observed LiDAR scan matches the pre-stored object scan. We test the localization part of our method by placing the object in front of the robot. Figure 3 shows the localization of the object shown as the pink dot. Measurements were noisy at times due to the cluttered environment the robot was operating in. However, false negatives did not occur frequently, which we argue would be the desired behavior in a search and rescue mission.

B. Mapping

We created a test scenario for the occupancy grid. The robot was driven using the joystick while the robot was updating the occupancy grid. The test environment is shown in Figure 4. After navigating the test environment, the occupancy grid we obtained is shown in Figure 5. The blue sections represent the occupied area on the global frame. The black sections represent free space. The grey sections represent uncertain areas. Based on the surrounding area of occupied cells, we can see that our robot is in an enclosed area that matches the overall shape of the test environment.

C. Motion Planning

The motion planning algorithm allowed the robot to navigate the unknown environment without colliding with any obstacles. There were some instances when the robot stopped

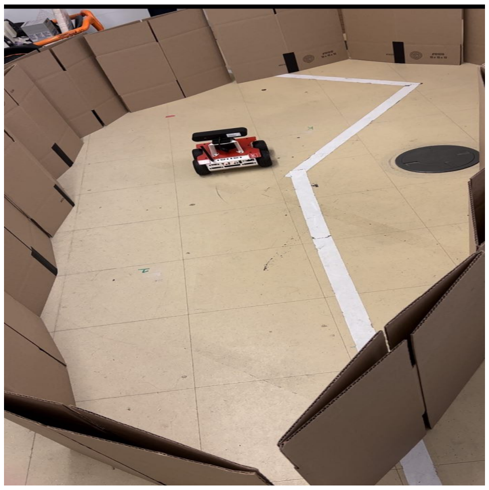


Fig. 4: Environment used for testing the occupancy grid mapping. The goal of this test was to ensure the measured grid had minimal noise and resembled the shape of the boxes.

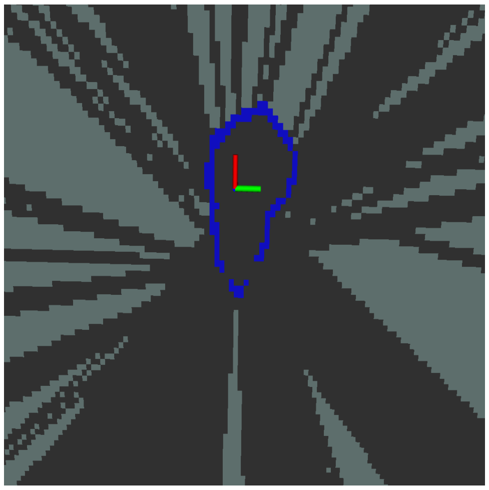


Fig. 5: The occupancy grid resulting from the test in Figure 4. The occupancy grid correctly resembles the shape of the aligned boxes and correctly classifies the interior as unoccupied space.

prior to reaching its goal due to getting stuck in local minima. Human intervention was then needed to remove the obstacle and allow the robot to continue navigating towards its target destination. A more robust motion planning algorithm would likely be needed for a realistic search and rescue mission, such as using the occupancy grid along with a graph-based planner to generate a smoother trajectory around obstacles.

D. Integration

To test the integration of the system components we instructed the robot to travel three meters forwards through the lab workspace while avoiding obstacles, locating objects of interest, and mapping its surroundings. The resulting occupancy grid and detection locations are shown in Figure 6.

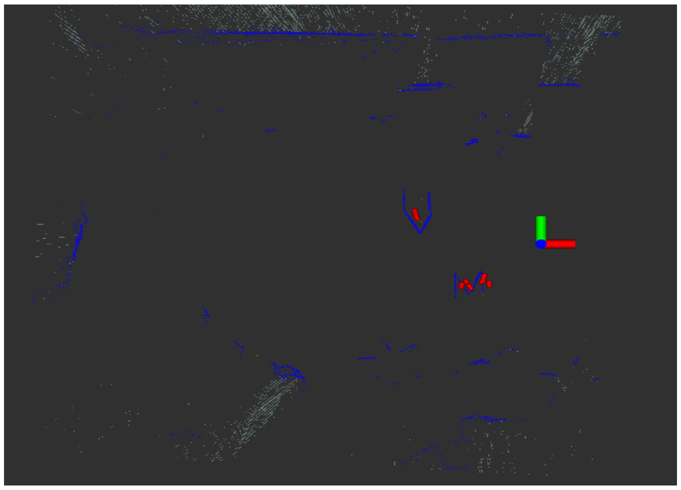


Fig. 6: The resulting occupancy grid after the robot navigated through the lab workspace. Blue cells indicate occupied spaces, black indicates unoccupied spaces, gray indicates unknown spaces, and red indicates detections of the object of interest while navigating.

V. CONCLUSION

The robot was able to successfully navigate through the room and correctly located the two objects of interest placed in the environment. The resulting occupancy grid also provided a fairly accurate representation of the testing environment. The grid contains information from wall to wall of the testing environment and shows other objects that were present in the room, such as chair legs and desks. Thus, the robot was able to perform all desired tasks.

A. Further Improvements

As mentioned previously, the LiDAR-based object detection algorithm struggled with noise due to the small information available in a 2D LiDAR scan. If this robot were to be deployed for a realistic search and rescue scenario, a 3D LiDAR scan would provide much more information to match features. A deep-learning based approach would also provide better results than the ICP algorithm as this would allow for the detection of objects that might not have a uniform shape, like people.

The biggest limitation of the study is the lack of area traversed. Further testing with completely navigating a larger environment would be a useful test to prove this system works at scale. To accomplish this, we would likely need an improved localization method to ensure our grid mapping remains accurate as the robot travels. A SLAM-based approach would be a likely candidate given that we are already generating an occupancy grid. We would also need a more sophisticated high-level controller that rewards the robot for navigating unexplored areas within the occupancy grid.

REFERENCES

[1] N. Ruangpayoongsak, H. Roth, and J. Chudoba, "Mobile robots for search and rescue," in *IEEE International Safety, Security and Rescue Rototics, Workshop, 2005.*, 2005, pp. 212–217.

[2] R. B. Rusu and S. Cousins, “3d is here: Point cloud library (pcl),” in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 1–4.

[3] Y. Chen and G. Medioni, “Object modelling by registration of multiple range images,” *Image and vision computing*, vol. 10, no. 3, pp. 145–155, 1992.

[4] P. J. Besl and N. D. McKay, “Method for registration of 3-d shapes,” in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. Spie, 1992, pp. 586–606.

[5] S. Rusinkiewicz and M. Levoy, “Efficient variants of the icp algorithm,” in *Proceedings third international conference on 3-D digital imaging and modeling*. IEEE, 2001, pp. 145–152.