EECE 5500 Mobile Robotics Class Project

Team Bebop

Avish K. Patel
Electrical Engineering
Northeastern University
Boston, USA
patel.avish@northeastern.edu

Durkesh K. Jevanandem
Electrical Engineering
Northeastern University
Boston, USA
jevanandem.d@northeastern.edu

Elikplim Gah
Mechanical Engineering
Northeastern University
Boston, USA
gah.e@northeastern.edu

Tarun Pati
Mechanical Engineering
Northeastern University
Boston, USA
pati.ta@northeastern.edu

***Abstract*—This report presents an implementation of reconnaissance with a turtlebot3-based autonomous system in an unknown environment. Building such reconnaissance systems is often non-trivial because of multiple objectives for the exploration paradigm like mapping the unknown environment, finding the victims, and the safety of both the robot system and the victims. The objectives of mapping and obstacle avoidance are achieved in this work by implementing a frontier-based exploration in conjunction with simultaneous localization and mapping (SLAM) based on the Lidar sensor measurements. The turtlebot3 system used in this work is a robot operating system (ROS) based Raspberry Pi 3 system. A Raspberry Pi camera is used to detect the victims in the environment. The objective of finding the victims is achieved by an information-based exploration and OpenCV detection algorithm. Finally, the efficacy of the proposed approach is tested in a simulated environment with April tags standing in for the potential victims.**

***Index Terms*—Autonomous system, SLAM, ROS, Disaster management and mitigation**

I. Introduction

Disaster management and mitigation is one of the most important applications of autonomous systems. Autonomous robot systems have successfully been employed in disaster scenarios like search and rescue, disaster environment mapping, and supply delivery to hard-to-access places. But due to resource constraints and safety concerns, they have yet to be widely adopted. The focus of this paper is to present a novel exploration and detection of potential victims in a disaster setting like fire rescue and Earthquake disaster mitigation. The application of such autonomous systems will facilitate the firefighters in efficiently handling the disaster management, at the same time improving the safety of the firefighters and the victims. Such disaster environments often involve low visibility, high temperature, and other hazardous settings and require rapid response. Hence, the use of autonomous robotic systems instead of manually operated robotic systems is recommended.

Several robotic systems have been proposed in the literature for exploring a disaster environment. Most of these works usually involve creating a global map of the disaster area using sensor measurements, while simultaneously transmitting important information like the position and condition of the victims and information about the structural integrity of the buildings etc., [1]. A CMU-based Highly Intelligent Mobile Platform (CHIMP), which performs difficult tasks like grasping a doorknob, was implemented in paper [2]. A system similar to CHIMP with four legs for stability was analyzed in disaster settings in the paper [3]. A disaster response robot with four independent caterpillars and four arms was presented in paper [4] named OCTOPUS. The robot has symmetrical sides to lessen blind spots in an emergency situation. This paper focuses on the exposition of the Turtlebot3 autonomous system as a possible system for exploration and detection in a disaster environment. The advantages of Turtlebot3 compared to other similar systems are its small size, ease of implementation, and deployment.

In this research, a Turtlebot3 mobile robot has been used to construct a fully autonomous system to conduct reconnaissance in a simulated disaster setting. For performing onboard computations for SLAM and detection, we use a Raspberry Pi 3 running the Robot Operating System (ROS) framework. The ROS is an open-source operating system for robots that is comparable to Windows or Linux in that it includes a library and algorithms for controlling robots, as well as drivers for various devices that can be added for additional functionality. SLAM approach is used to guide the turtlebot and create a map based on Sonar, LiDAR, and other sensor measurements. The ROS's SLAM approach [6], [7] is broken into two sections. Surveying and producing a map is the first step, then sending it as a topic and saving it to a map server. The second step involves identifying the location of the survey robot by localizing it for ROS using an Adaptive Monte Carlo Localization (AMCL) algorithm, which determines the robot's location on the map. [5], [8]–[11] The robot may then be directed to travel automatically and avoid obstacles on the map using ROS Navigator. Numerous SLAM libraries, including SLAM Gmapping, Hector SLAM, SLAM karto, etc., are available in ROS. [5] Hector SLAM [8]–[10] and LiDAR sensors in UGV robots were utilized in a number of studies for accurate positioning and the creation of 2D maps.

Contributions. In this work, we propose the use of two exploration strategies to achieve two different objectives of mapping the disaster environment and detecting the potential victims in that environment. We are using a Lidar sensor for mapping and a Raspberry Pi camera for detection. The idea of balancing the objectives of safety, mapping the disaster environment,

and detecting the victims is still an open problem. In this work, we explore a few different strategies and list their merits and demerits. We have employed a sense of optimality in our composite exploration problem of trying to minimize the time it takes to explore the environment and simultaneously detect the victims while trying to maximize the information. To evaluate the efficacy of our problem, we have created a simulated environment to run our Turtlebot3 system, we have also used Apriltags as stand-ins for any potential victims in the environment.

II. Problem statement

Problem 1: Given an unknown disaster environment, a list of potential victims that may be present in the environment, a Turtlebot3 autonomous system with some prebuild packages device a strategy to map out the disaster environment while simultaneously searching for the victims from the list that is available apriori.

Note that to facilitate the ease of solving this problem, we have made some assumptions in this work that are listed below.

Assumption 1: We are making an assumption that we have the list of potential victims present in the environment apriori, we are also assuming the list matches perfectly with the actual victims present in the environment i.e. there are no unknown victims (not on the list) present in the environment.
Note that such an assumption is reasonable because in most disaster scenarios of the modern world, due to the extensive surveillance systems we have in place, we have a list of potential victims present in the environment prior to deploying the autonomous robot systems for disaster management.

And as stated earlier, in this work we are employing Apriltags as stand-ins for the potential victims in the environment.

Assumption 2: To make the detection problem computationally simple, we are reducing the search space for detecting the victims by assuming that the Apriltags are only present on the walls and obstacles. The Apriltags are also at an appropriate height relative to the Raspberry Pi camera so that they are within the field of view (FOV) of the camera within the detection range.
Note that such assumptions are justifiable because in a real disaster scenario it is simple to extend our system to search over the entire map instead of a reduce-map for the detection algorithm. We can also employ a camera with a larger FOV. This way we can detect victims in any position in the environment and at any relative height within the detection range.

Assumption 3: Lastly, we are assuming that the detection range of the camera used for detection is smaller or equal to the sensing range of the LIDAR sensor.
This is a reasonable assumption to make, as LIDAR sensors usually have a very large detection range compared to standard cameras.

III. Specifications and Packages

A. Hardware

1) Turtlebot3: Turtlebot3 is a standard platform robot based on ROS. It contains OpenCR microcontroller and a Raspberry Pi for directional drive and onboard computation respectively. For this project, we have chosen Turtlebot3 because of the extensive support and the availability of open-source ROS packages. There are two versions of turtlebot3 available on the market, Turtlebot3 burger and Turtlebot3 waffle pi. We have chosen Turtlebot3 burger over Turtlebot3 waffle pi as it is comparatively less expensive due to the proprietary Intel RealSense with software development kits for recognition.

2) Raspberry Pi 3: Raspberry Pi is a small size, low-cost computer that can be deployed on many autonomous and non-autonomous systems as an onboard computer. For the purpose of this project, we can deploy Ubuntu on the Raspberry Pi and the Turtlebot3 can be controlled by SSH directly into the system with a remote PC. We can also easily add the additional components required for the task, like LIDAR sensors and a camera to the Raspberry Pi board on the Turtlebot3.

3) Lidar sensor: Turtlebot3 uses a 360° LIDAR sensor to map out the environment using SLAM and Navigation packages.

4) OpenCR: To run the differential drive, Turtlebot3 uses a 32-bit ARM Cortex-M7 processor-based OpenCR microcontroller.

5) DYNAMIXEL: Two DYNAMIXEL actuators are used to drive the wheels.

6) Lithium Polymer Battery: The entire Turtlebot system is powered using either a wired power supply or a Lithium Polymer (Li-Po) barrtey when operating remotely.

7) Raspberry Pi camera: An additional camera module was added to the Raspberry Pi onboard the Turtlebot3 for the purpose of Apriltag detection.

8) Apriltags: As stated earlier, we are using Apriltags as a stand-in for any potential victims in the simulated disaster environment. They are used because they provide a very robust visual fiducial system for mimicking a potential victim in the disaster setting. Most cameras can be used to detect Apriltags either using OpenCV or machine learning-based algorithms.

B. Software

1) ROS: ROS is an open source "operating system" for robots. It provides packages and tools for controlling a robot's actions. Being open source, we are able to reuse available packages and resources to aid task completion. The following ROS packages are used in our project.

a) Turtlebot3 packages: This is a package provided by ROBOTIS. It contains packages relating to the robot bringup, SLAM and motion planning.

b) turtblebot3_slam: This package is another provided by them

c) cv_camera: This is a ROS implementation of the OpenCV camera driver. It allows video device capture using the cv::VideoCapture object of OpenCV. This allows us to receive a video feed from the Raspberry Pi camera connected to the system.

d) Apriltag_ros: This is a ROS wrapper of the Apriltag 3 detection algorithm. Since we use apriltags as stand-ins for victims in the environment, we employ this algorithm to enable

detection of the tags within the environment. It applies the algorithm to the camera feed and publishes the detected tags and tag poses as topics. It uses the camera feed by accessing them using raspicam_node with ROS.

e) Explore_lite: The explore_lite package allows for the implementation of greedy frontier exploration behaviour. It relies on information provided by a SLAM package to determine frontiers for exploration. The determined goal is then sent to move_base which provides navigation inputs to achieve the goal.

IV. Main Results

A. Navigation

Navigation involves moving the turtlebot to a goal in the environment. In this work, we use an open-source ROS package called move_base to achieve navigation. We choose to utilize move_base because it is built on the foundation of a navigation algorithm. Given a target in the world, the move_base package offers an implementation of an action that will try to reach it using the mobile base. Internally, the move_base package builds a navigation plan by linking a local planner and a global planner. Which involves maintaining two costmaps one for the local planner and one for the global planner. Additionally, move_base contains other navigation aspects which adhere to an established API's specified in ROS standard interface. This makes it easier for the user to construct their own algorithms for each component of robotic navigation like Local and global planning, Recovery, and Costmaps using this standard interface.

B. Mapping

In order to navigate the turtlebot in the disaster environment, first we need to gain a sense of how the environment is or how it changed due to the disaster scenario, so a map of the disaster environment needs to be constructed. Mapping involves building a map of the environment using the sensor measurements in conjunction with the information about the location of the turtlebot on that map. For this purpose, we use a ROS package for gmapping which facilitates laser-based Simultaneous Localization and Mapping (SLAM) as a slam_gmapping ROS node using the measurements from the LIDAR sensor onboard. This creates a 2-D occupancy grid from the LIDAR sensor data and the pose data. Gmapping is a widely robotic system, which creates the grid-maps based on Particle filters. The gmapping system subscribes to LaserScan msgs/sensor msgs, and publishes tf transformation and OccupancyGrid/nav msgs.

C. Detection

By accessing the camera feed we can deploy various detection strategies to detect all the apriltags/victims present in the environment. These strategies can be classified into two different categories.

1) Discrete detection: Since the FOV of the camera used for detection is limited (less than 360° in all directions), what we can do at the start is employ a local search by rotating the turtlebot by 360°, then stop the search and use some exploration strategy to move to a new location and repeat the local search at the new location. We can repeat this over and over until we cover the entire search space.

2) Continuous detection: An alternative strategy we can employ is to detect the tags from the continuous video stream while navigating the turtlebot for mapping. The advantage of this is that this does not conflict with the exploration strategies deployed for mapping the environment, unlike discrete detection strategies. The clear disadvantage being we may not cover the whole search space with the camera by the time the mapping exploration ends (due to the limited FOV of the camera)

So, a combination of both strategies is optimal. This is further discussed in exploration section IV-D.

D. Exploration

Exploration differs from navigation and SLAM in the sense that the disaster environment needs to be explored to create the maps or to detect the Apriltags. Two exploration paradigms are used in this project, the objective of the first exploration algorithm is to map the environment, and the objective of the second exploration is to detect the April tags. But in some challenging circumstances, these algorithms encounter a bottleneck and get stuck in a loop.

1) Exploration for mapping: For mobile robots, a variety of exploration algorithms are available, like the Bug0 algorithm, Bug1 algorithm, wall following algorithms, etc. Rapidly-exploring Random Tree (RRT) was one such algorithm we came across. But one problem with using RRTs is that the rectangular region around the turtlebot needs to be added manually to the RVIZ window, which works contrary to the idea of full-autonomy. Hence, RRT strategies were not chosen, even though they are designed such that the estimated distance between a randomly picked point and the tree rapidly reduces. Finally, we have decided to employ explore_lite ROS package which contains an algorithm for frontier-based exploration, which is one of the most common exploration strategies used. The regions of the map between the unexplored space and the open space are called frontiers and moving the turtlebot to a new frontier, the map of the disaster environment can be continuously built until there are no new frontiers to explore. The advantage of the frontrier exploration is that it can be deployed for mapping cluttered, small, or large spaces. Using occupancy grid frontiers, we can explore the disaster environment with ROS by employing the frontier_exploration package with explore_server node. By utilizing the occupancy map constructed by the slam_gmapping package, this node publishes the goal location to the move_base package.

2) Exploration for detection: Similar to how frontier-based exploration is used in conjunction with LIDAR sensor measurements to cover the entire disaster environment for

mapping, we need an additional exploration strategy for detection which uses the camera sensor measurements (images or videos) to cover the entire search space for detection. In this paper as a consequence of Assumption 2 the search space is just the interface between the occupied and the unoccupied space instead of the entire map. Hence, by exploring the entire search space using the camera sensor measurement and a detection algorithm, we can detect for all the apriltags (victims) present in our environment. But because of Assumption 1, we don't need to explore the entire search space but can stop the exploration as soon as we find all the victims (Apriltags) present in our victims' list that is available apriori. Based on the information of the environment (obtained from mapping exploration and SLAM), information about already explored search space (for apriltags) in the known environment and the unexplored search space, the information about the victims found and yet to be found from the victims' list, we can build an information-based exploration strategy for detection. In this exploration, we employ the detection algorithm on the continuous video feed, as well as rotate the turtlebot at every new position to not miss the tags outside the camera's FOV while moving.

Note that we can use many different strategies to deploy these two exploration strategies.

1) *Strategy 1*: We can first deploy the Frontier based exploration algorithm for mapping out the unknown disaster environment, and after we complete the mapping (i.e. no new frontiers to explore), next with the now known map of the environment we can deploy the information-based exploration over the search space until we find all the apriltags in the list. Then we can stop the exploration. But this is very inefficient because it employs the two exploration algorithms one after the other, which wastes valuable time for disaster management.
2) *Strategy 2*: We can simultaneously deploy both exploration strategies. In this case, at each step of the composite exploration strategy, the algorithm produces two goal positions for the turtlebot to move to. One goal point is from the exploration strategy for mapping, and the other is from the exploration strategy for the detection of apriltags. Next, we select the nearest goal point to our current position to move to next (this is usually the goal point obtained from the exploration algorithm for tag detection, as the sensing range and FOV of the camera is usually smaller than the LIDAR sensor). We can stop the composite exploration once we map the entire environment. And because we are using the nearest goal point, by the time we stop the exploration we would have mapped the entire unknown environment as well as searched over the entire search space.
3) *Strategy 3*: Like strategy 1, strategy 2 is also inefficient because we are using a lot more movement than needed to map the environment (because we are choosing the nearest goal point), on top of that we are also exploring the entire search space for the tags (we don't need to explore the entire search space but only till we find all the tags). We can modify Strategy 1 to build a new strategy that is more efficient than strategy 1 and potentially more efficient than strategy 2. In this strategy, we first deploy the exploration strategy for mapping and while doing the mapping we will also use the continuous video stream for detecting the tags, and after we complete mapping the entire environment and missed some of the apriltags, we can deploy the second exploration strategy to find the remaining apriltags. The best efficiency this strategy can achieve is when we detect all the tags in the list from the continuous detection (from video stream) while mapping, so we do not have to deploy the second exploration. The worst performance of this strategy is equal to the performance of strategy 1.

E. Pose Estimation

Since apriltags are used as stand-ins for the potential victims in a disaster environment, estimating their pose in the global coordinates becomes very important. When the turtlebot autonomous system detects the apriltags (victims), it only outputs the apriltag pose in the camera frame and not the world frame. This information is published on the tag_detections topic, but is inaccurate for localizing the tags within the environment. To correct this, we run a node that publishes the correct tag pose with respect to the map, and this is written to a text file.

V. Simulated Example

To evaluate the performance of our proposed approach, we deployed the Turtlebot3 autonomous system in a simulated disaster environment as seen in Figure 1

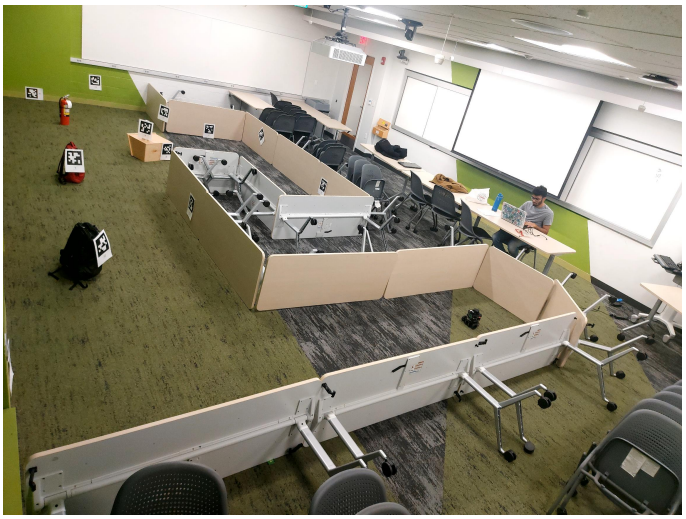

Fig. 1. Simulated disaster environment with Apriltags as victims

We have used 16 apriltags from $36h11$ family as stand-ins for the victims in this simulated environment. The 2-D occupancy map generated using our proposed approach can be seen in Figure 2. As can be seen from the simulated environment in Figure 1 and from the occupancy map in

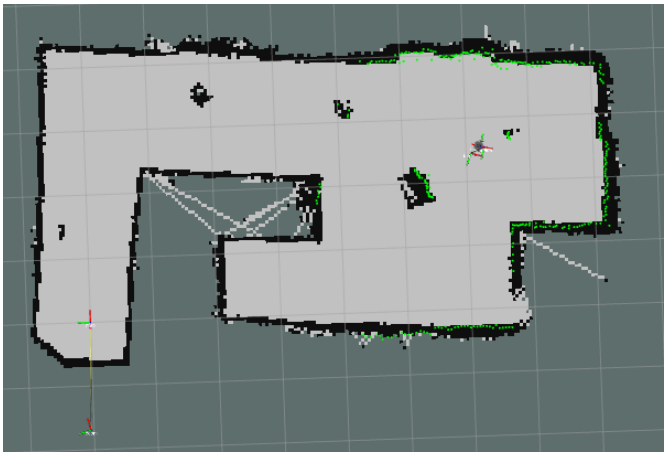

Fig. 2. Occupancy Map of the Simulated Disaster Environment

Figure 2, our proposed approach successfully generated a 2D occupancy map of the disaster environment. We saw significant variation in results with regard to detecting the apriltags in the environment. We run 3 separate experiments and had different results.

Run	Time	Number of detections	Max speed
Run 1	3:12	7	0.22
Run 2	3:49	15	0.18
Run 3	5:21	10	0.1

We found that reducing the max speed of the robot improved our ability to detect the tags, but our biggest issue remained an inability to transmit the image information synchronously. Code and launch files can be found at https://github.com/agbeT/EECE-Mobile-Robotics-Class-Project.

In one of our runs, we detected 15 apriltags and their transformations as evident from Figure 3 below

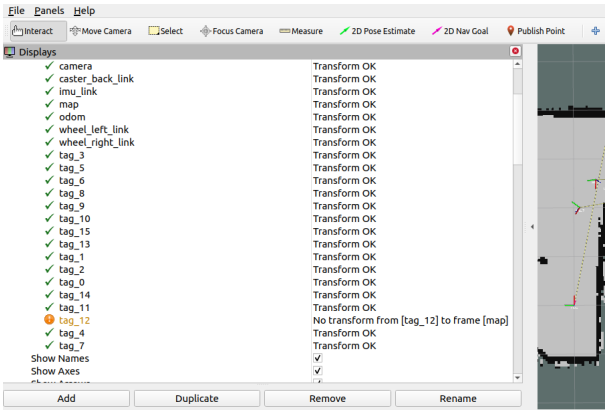

Fig. 3. Apriltag Detection

VI. Challenges Faced

1) Our biggest challenge was the time needed to complete the project. No team member had prior experience working with ROS. This made debugging issues we encountered a challenge as there was no experience to draw on and the forums and resources available typically assumed some level of familiarity. This combined with the delay in getting all robot components and schedules of team members meant we had to complete the project in what felt like very little time.
2) We encountered significant issues with sending information from the turtlebot to the remote PC. This we found to be because of the connection which we achieved using our phone hotspot. To resolve this, we limited our camera output to 2 fps but still had issues.

VII. Conclusion and Future Work

A reliable, fully autonomous turtlebot3 system has been implemented to conduct reconnaissance in a disaster environment simulation. As seen from the results, an occupancy map was successfully generated. The experiments carried out with different strategies, modifying the exploration frontier values, speed of the robot, and their actual impact on a particular environment provided insights into the power and capabilities of the algorithm developed in this work. This project has a lot of room for growth, including the addition of a random sample technique that can efficiently discover all the victims (apriltags), the addition of a bayesian search method, avoiding bottlenecks, and extensively exploring the surroundings.

References

[1] Yang, Kyon-Mo, Jong-Boo Han, and Kap-Ho Seo. "A multi-robot control system based on ROS for exploring disaster environment." 2019 7th International Conference on Control, Mechatronics and Automation (ICCMA). IEEE, 2019.

[2] Haynes, G. Clark, et al. "Developing a robust disaster response robot: CHIMP and the robotics challenge." Journal of Field Robotics 34.2 (2017): 281-304.

[3] Klamt, Tobias, et al. "Supervised autonomous locomotion and manipulation for disaster response with a centaur-like robot." 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018.

[4] Kamezaki, Mitsuhiro, et al. "Design of four-arm four-crawler disaster response robot OCTOPUS." 2016 IEEE international conference on robotics and automation (ICRA). IEEE, 2016.

[5] Yoonseok, P. Y. O., et al. "ROS robot programming." Seoul, Republic of Korea: ROBOTIS Co., Ltd (2017).

[6] Balasuriya, B. L. E. A., et al. "Outdoor robot navigation using Gmapping based SLAM algorithm." 2016 Moratuwa Engineering Research Conference (MERCon). IEEE, 2016.

[7] An, Zhen, et al. "Development of mobile robot SLAM based on ROS." International Journal of Mechanical Engineering and Robotics Research 5.1 (2016): 47-51.

[8] Kohlbrecher, Stefan, et al. "A flexible and scalable SLAM system with full 3D motion estimation." 2011 IEEE international symposium on safety, security, and rescue robotics. IEEE, 2011.

[9] Arras, Kai O., et al. "Efficient people tracking in laser range data using a multi-hypothesis leg-tracker with adaptive occlusion probabilities." 2008 IEEE International Conference on Robotics and Automation. IEEE, 2008.

[10] Ibragimov, Ilmir Z., and Ilya M. Afanasyev. "Comparison of ROS-based visual SLAM methods in homogeneous indoor environment." 2017 14th Workshop on Positioning, Navigation and Communications (WPNC). IEEE, 2017.

[11] Zaman, Safdar, Wolfgang Slany, and Gerald Steinbauer. "ROS-based mapping, localization and autonomous navigation using a Pioneer 3-DX robot and their relevant issues." 2011 Saudi International Electronics, Communications and Photonics Conference (SIECPC). IEEE, 2011.

[12] Christie, Benjamin A., Osama Ennasr, and Garry P. Glaspell. "Autonomous navigation and mapping in a simulated environment." (2021).