Team 8 CS81 Final Project Report: Robot Dog

Jordan Kirkbride, Eric Lu, Julian (Xiao Yi) Wu, and Wendell (Wending) Wu

***Abstract* — This report serves as a description and summary of Team 8's CS81 Final Project: Robot Dog. We address the feasibility of creating and we provide an implementation of a robotic dog that is capable of searching a foreign environment for a target object ("ball") and returning the object to a specified location using laser sensor and camera data. As a consequence of our implementation, the robot is also capable of mapping the explored environment as it proceeds in its search. We demonstrate through use of an OccupancyGrid paired with a laser sensor, a wavefront-based frontier exploration algorithm, computer vision object detection and tracking using OpenCV and a camera, and a PD controller for object retrieval movement patterns that our robot dog implementation is capable of autonomously completing the specified task. Experimental results show mixed success, especially in challenging environments and with unforeseen edge cases.**

I. Introduction

Nowadays, in a world increasingly based around AI and offloading human labor to machines, autonomy in robots is more valuable than ever. All it takes is one look at Kiwi delivery robots or factory construction robots, for example, to see the benefits of unsupervised robotic tasks. If we were to similarly replace high-risk or high-cost tasks like ocean-based survivor rescue or last mile deliveries with robots, we could potentially solve many issues at once. For these types of tasks, we propose a robot that can self-guide, locate an object, and retrieve it successfully, as we believe such a robot can be immensely useful as an autonomous fetching mechanism for tasks that currently require significant human supervision.

The objective of our project is to create a simpler prototype of this robot as a proof-of-concept. Therefore, we aimed to program a robot that can fetch a red ball. We want to work on a robot that can work in foreign environments, avoids obstacles, successfully retrieves the target object, and brings the object back to the user.

The inputs the robot will receive are data from a laser sensor, images of the camera output, and the location of the user. The robot will explore through a frontier-based algorithm in conjunction with its own dynamic map formed through processed laser sensor data. The outputs the robot will provide are a simple mapping of the environment as it explored, and ultimately bringing the object back to the user successfully as a physical "output" of the task completed.

There are a few assumptions we make about the situation:

1. The environment is on a level plane.
2. There exists a viable path for the robot to retrieve the object.
3. There exists a region where the robot will have a clear view of the target object.
4. The robot has the necessary hardware to contain or hold the retrieved object.

Some metrics we will measure are:

1. Time to find target object
2. Rate of success for finding the target object

II. Relevant Literature

Our chosen problem of a robotic "dog" playing fetch has been the subject of much research with various approaches attempted. We will highlight some successful implementations of robotic fetch specifically from the US Army TARDEC-funded CANINE (Cooperative Autonomous Navigation in a Networked Environment) program, which is directly related to our problem of robotic fetch. The first notable example is the R-MASTIF, made by the SRI-UPenn team who used a modified Segway RMP 200 robot with multiple cameras and lidars [1]. They implemented a unique computer vision-based approach for detecting previously unseen objects out to 15 meters on flat terrain, as well as odometry for GPS-denied localization. For retrieval, they designed a scooping mechanism to pick up basketball-sized objects.

Another notable project from the program was LABRADOR, which was a robot based on iRobot's PackBot unmanned ground vehicle (UGV), also equipped with an explosives ordinance disposal (EOD) manipulator arm and a custom gripper [2]. They developed a vision-based object learning and recognition system that could recognize objects presented to the robot in real-time. They also implemented a waypoint navigation system based on GPS, IMU, and odometry data. This enabled the robot to search a specified area using a variety of coverage strategies, including outward spiral, random bounce, random waypoint, and perimeter following behaviors. Unfortunately, the full system was not integrated in time for the CANINE competition, but it is still worthy of note for inspiration in our project.

A final CANINE project to mention was that by Neya Systems LLC, who developed a robot capable of learning and recognizing target objects, conducting area searches among distractor objects and obstacles, and relocating the target object [3]. Neya collaborated with the Robotics Institute at Carnegie Mellon University to develop vision-based solutions for probabilistic target learning and recognition. They also used a Mission Planning and Management System (MPMS) to orchestrate complex search and retrieval tasks using a set of modular autonomous services for robot mobility, perception, and grasping.

Many similar experiments have also been conducted, although not directly in relation with a "dog" like robot. One example is by Nguyen et. al. [4] where they implemented a robot that aimed to help those with motor disabilities to get objects from a flat surface (as would be the normal surface inside of one's house). They achieved this by creating a novelty sensory and kinematic configuration that was capable of movement across flat surfaces and of object grabbing. The object detection was achieved via a laser sensor. The robot itself along with the logic they produced for it led to a success rate of 86% in retrieving a specified object and returning it to a designated location in their experiments.

These examples are among many that aim to improve technology that combines robotics with computer vision to perform complex tasks. As shown, this topic is not limited to the world of "dog fetching tasks" within the context of the CANINE competition. The overall general problem of computer vision based object detection and retrieval can be used to help those with motor impairments, along with numerous other beneficial use cases. Our approach hopes to merge many of the implementations seen in [2]-[4], with the closest match being found in example [1], although we hope to provide the user with another output in the form of the occupancy grid generated by the exploration performed.

III. Approach and Implementation

All code was created with Python and the ROS library. The simulation was run within ROS Gazebo with a Turtlebot3 Waffle Pi model. We successfully allowed our robot to map its environment, detect and make its way to an object, and then plan a path to its original location using:

Frontier-based exploration: In order to map its environment and find the target object, the robot must have some way of exploring its environment, whether it be through a BFS, DFS, or some other exploration algorithm that will allow the robot to systematically explore. For the robot we have chosen, we will achieve this through processing of the laser scanner data to create a partial map of the robot's sensed environment, and to utilize the map to choose further locations to explore. This builds off of code previously written for Programming Assignment 4, where an originally unknown OccupancyGrid is slowly filled in with empty or occupied cells based on the robot's odom position and the gathered laser sensor data, making sure that only the portion visible to the robot's camera in front of it is mapped as visited areas in our OccupancyGrid to ensure the robot will not accidentally miss facing the target ball if it is behind the robot.

From the partial occupancy grid, we then utilize a wavefront-based algorithm inspired by the previously learned BFS exploration algorithm to first choose a new target location to explore and to plan a path to the chosen location for the robot to take, all the while updating its OccupancyGrid at the specified rate while the robot is traveling to its next chosen target. This wavefront-based algorithm, as introduced in its pseudocode fashion in [5], is composed of an outer and inner BFS, will recursively search for the frontier of the occupancy grid and returns a list of groupings of points, where each grouping is one contiguous section of frontier. We then further processed the output of this algorithm in order to calculate the centroid of each clustering of points in order to pick which cluster to explore next. After picking such a cell, we then feed this target destination to code that will plan a path to the target, loosely inspired by the path planning Programming Assignment 3. Upon reaching the destination, we then perform these steps all over again. This loop then continues indefinitely until the target ball is within the field of vision, at which point the next phase of the program begins.

Identifying a goal object: The robot, upon being close enough to the target object, must be able to sense and identify the object in question in order for its primary objective to be achieved. For the robot we have chosen, this will be achieved with a combination of camera imagery and image processing to identify the target object. The algorithm mainly works by using OpenCV's library and detecting contours in the images provided from the camera subscription. Upon detecting a contour and checking the color of the detected object, the robot will rotate until the object is primarily in the center of the images. Then it will move towards the target and continue to adjust its rotation every frame. There are a few parameters we had to tune, such as the threshold of the redness of the ball, how close the robot should get to the ball before the PID controller takes over, and the turning rate of the robot.

Collection and delivery of the ball: Once the ball has been detected and the robot is within a specified distance of the ball, the PD controller engages and aims to keep the ball .5 meters in front of the robot at all times while it makes it way back to the starting location via the frontier based mapping that occurs. These were our original intentions that were not fully brought to fruition and are discussed further in depth in the section on the challenges we faced. In the simulations, the PD controller is disabled as to allow for us to test all of the other features correctly. Because of this, once the robot reaches the ball, it is assumed that the ball is in the possession of the robot without any further intervention (i.e. it's as if it's a dog and it has the ball in its mouth, meaning there is no need to track its position any longer). Therefore, successfully reaching the ball in our case means the ball was successfully collected and successfully delivering the ball back to the original location is achieved by simply navigating back to the robot's start position. Challenges were faced in navigating back to the original position once the ball was reached, as will be discussed in the challenges section.

IV. Experiments and Discussion

We ran several experiments in simulation to test our code and the effectiveness of our robot dog and gathered data for the two metrics we were evaluating against.

In these experiments, we placed the robot in five consistent starting locations: either one of the four corners of the grid environment we chose for the robot, or the center of said grid, with the robot's orientation matching that of the world at the start. The red ball was then placed in a random position in the world to simulate a random ball toss. In total, we utilized two different environments—one with obstacles in between, like a maze, and one empty square world, totalling 10 different experiments. For each trial, we timed to see how long it would take the robot to find the ball and its success rate (success in finding the ball). After 10 trials, our results are presented in Table I.

TABLE I. RESULTS

Trial/World	Success (Y/N)	Time (s)
1.1	Y	29
1.2	Y	38
1.3	Y	32
1.4	Y	28
1.5	Y	45
2.1	N	n/a
2.2	N	n/a
2.3	Y	113
2.4	Y	100
2.5	N	n/a

Each trial with prefix 1 denotes the square world, and 2 denotes the maze world. As shown above, the maze world proved significantly more challenging for the robot, with a success rate of only 40% compared to the 100% success rate in the empty square world, and a significantly longer find time due to the longer exploration time required before being able to find and reach the ball. We averaged 34.4s to reach the target ball in the square world, with most of the time being put towards reaching the ball after finding it, as it was easy for the camera to detect the ball with no obstacles, as long as it came in the field of vision.

The maze world proved significantly more challenging though, as we would occasionally run into errors with the camera thinking the wall was somehow the same color as our target ball, and the robot would run into the wall as a result, leading to a 40% success rate. Additionally, it would take significantly longer to find the ball due to the obstacles in the way, making for longer exploration times. However, due to our contour-based detection algorithm, sometimes the ball would be partially in frame but hidden halfway behind an obstacle and trigger the detection phase, meaning the robot would get stuck on the corner of an obstacle while tracking the ball due to its linear movement once the ball is detected.

Some pictures of the tests in action are provided as follows:

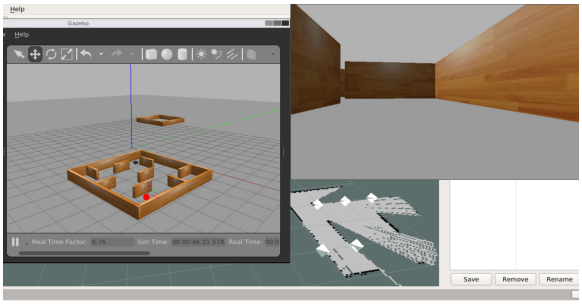

Figure 1. Example run in a maze environment (left) with all outputs shown: camera view (top right) and mapped world (bottom right) shown as an OccupancyGrid with frontier centroids shown as white squares.

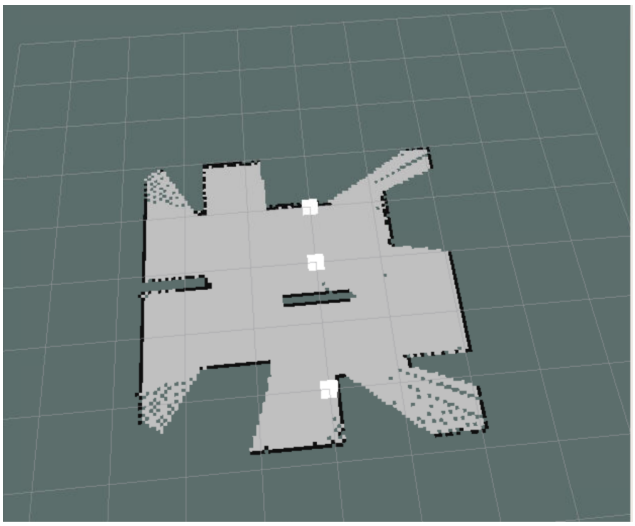

Figure 2. Testing our wavefront frontier-detection algorithm in rviz, showing the centroids of each clustering of detected frontier points as white squares overlaid on the shown OccupancyGrid.

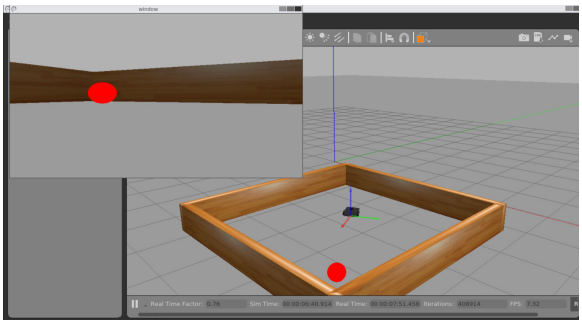

Figure 3. Robot in the process of navigating to the red ball in the open square environment.

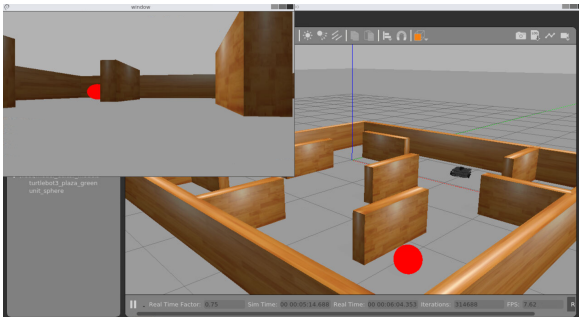

Figure 4. Red ball is partially occluded by an obstacle, yet the robot navigates in a straight line toward the detected ball.

V. Challenges and Limitations

We aimed to fully incorporate the PD controller that we created to allow for the robot to keep the object in front of itself as it moved its way back to the starting location. We were presented with one main challenge while integrating: the limitations of our simulation. We originally were going to allow the robot to "retrieve the ball" by adding static arms in front of the robot so it could have some way to control the object, but we were unable to do this in simulation. With this limitation presented to us we decided to focus on the exploration of our map and the detection of the object via computer vision. Although the full integration was not completed, the PD controller in theory should work as intended, with its goal of keeping the ball in front of the robot at a distance of .5 meters being correctly achieved. When tested separately, the PD controller correctly and efficiently allowed the robot to navigate to a specific location. This makes us confident that with the right modifications to the robot's model in our simulation, the PD controller would correctly keep the ball within possession of the robot. A more advanced simulation and even real world testing could help us confirm this.

OpenCV and simulating the ball tracking was also a challenge. It took a long time to get the Gazebo simulation up and running, and we could not get the images from the camera subscription due to incorrect topic subscriptions. After a lot of debugging, we finally realized that we subscribed to the wrong topic. In addition, the red ball recognition algorithm worked well, but we had some trouble with various lighting and shadow conditions. In certain cases when a dark shadow appears over the red ball, the algorithm could not locate the contours of the ball and missed the ball location.

One of the last large obstacles we encountered was our inability to successfully break from the ball tracking phase of the program and into the return-to-user stage. It was unclear to us given the lack of simulation "arms" at what stage we should break from the camera-based ball tracking stage, partially due to lack of information about the real-world position of the ball—meaning, we could not use the distance to the ball as a metric for a transition, which would have been the obvious choice, and implementing this without the assumption of a stereo-based camera for our robot would have been beyond the scope of our project.

VI. Conclusion

In this paper, we presented our approach towards creating a robot that could fetch a red ball. We used a frontier-based algorithm for exploration, OccupancyGrid mapping for constructing the explored environment, camera-based algorithms through OpenCV for ball detection, and a PID controller for maintaining control of the ball and bringing the ball back to the user. We discovered through our simulation experiments that even with our proposed methods seemingly working well in theory, a full working implementation even in simulation proved challenging with many edge cases involving camera detection messing up, which was one of the more difficult parts to implement to begin with.

We learned that creating such a robot took tremendous effort and it worked quite smoothly in simulation. However, we realize that implementing the code on an actual robot will provide another whole set of challenges due to robot errors, sensor noise, and environmental inconsistencies. The task of creating a robot that works well in any and all foreign environments is incredibly challenging, and we were proud to have accomplished a somewhat working prototype in just simulation environments.

VII. Ethics

In any field of robotics, it is important to think about the ethical nature of our creation and the impact it can have on the people and environment around us. There are many circumstances where our project could inspire greater developments to society. The main features of our project are object detection, retrieval, and delivery, which provides an extremely wide range of possible use cases that could help humanity for the better (and unfortunately possibly for the worse). One example is that our project could help enable cheaper and more efficient ways of cleaning trash from the environment through the use of a trash detection robot that picks up trash and returns it to a trash can. Another possible benevolent case use could be using robot detection to find injured people (for instance, individuals lost in the wilderness, or injured in a battlefield) and bring them back to a safe location to receive treatment. The possibilities are almost endless for positive use cases. However, we must also consider the circumstances where our project's findings could be abused. For instance, our project could enable the development of robots used for organized theft, or weaponized robots. In fact, the CANINE competition that was discussed previously in the relevant work had many entries sponsored by the U.S Army for the very purpose of bomb detection and retrieval (which is what they tell us, but it could easily be used for malicious purposes by the Army for offensive capabilities). In order to encourage the ethical use of our findings, we suggest greater regulation and/or the development of hard-coded robot limitations. Companies should be honest about the technology they are creating and the purposes for which they are creating it for.

VIII. Allocation of Effort

The frontier mapping and navigation was a joint effort between Eric and Wendell. Wendell combined elements of the OccupancyGrid mapping functionality originally found in Programming Assignment 4 with components from the path planning utilized in Programming Assignment 3 in order for the robot's mobile base and core functionality in movement to work properly. Eric then implemented the pseudocode drawn up for frontier exploration in order to allow the robot to autonomously choose locations to explore in a systematic function, integrating his standalone functions with the existing basic movement already implemented, thus accomplishing our goal of exploration. Julian worked on setting up the Gazebo simulation environment, using OpenCV's imaging library, and the red ball recognition algorithm. Jordan was responsible for the implementation of the PD controller to adjust the robot's position in relation to the goal object's position. As a team, we had an open discussion regarding the ethics surrounding our project and the potential implications that could arise from such technology, which is reflected in the Ethics section. For our experiments, we discussed the best way to test the success of our project as a team, and then ran the experiments together as a team. The rest of the report was allocated according to each teammate's specific role in code implementation.

IX. References

[1] Aveek Das, Dinesh Thakur, James Keller, Sujit Kuthirummal, Zsolt Kira, Mihail Pivtoraiko, "R-MASTIF: robotic mobile autonomous system for threat interrogation and object fetch," Proc. SPIE 8662, Intelligent Robots and Computer Vision XXX: Algorithms and Techniques, 86620O (4 February 2013); doi.org/10.1117/12.2010720

[2] Brian Yamauchi, Mark Moseley, Jonathan Brookshire, "LABRADOR: a learning autonomous behavior-based robot for adaptive detection and object retrieval," Proc. SPIE 8662, Intelligent Robots and Computer Vision XXX: Algorithms and Techniques, 86620P (4 February 2013); doi.org/10.1117/12.2011834

[3] Brian A. Stancil, Jeffrey Hyams, Jordan Shelley, Kartik Babu, Hernán Badino, Aayush Bansal, Daniel Huber, Parag Batavia, "CANINE: a robotic mine dog," Proc. SPIE 8662, Intelligent Robots and Computer Vision XXX: Algorithms and Techniques, 86620L (4 February 2013); doi.org/10.1117/12.2010302

[4] Nguyen, Hai & Anderson, Cressel & Trevor, Alexander & Jain, Advait & Xu, Zhe & Kemp, Charles. (2010). El-e: An assistive robot that fetches objects from flat surfaces.

[5] Autonomous Robotics Labs, "Exploration and path planning," *cw.fel.cvut.cz*. [Online]. Available: https://cw.fel.cvut.cz/b192/_media/courses/aro/tutorials/04_exploration.pdf. [Accessed June 4, 2023].