TurtleBot Tag

Ben Hart
David Booker-Earley
Johnathan Chadwick
Yuen Bak Ching (Max)

ME-4451 Group Project
Lab Section A
Fall 2018

Abstract:

The proposal for this project was to program two different Turtlebots to play a one-sided game of tag using one large bot and one small bot. The small bot will wander around an unknown area using its Lidar sensor to detect and avoid all nearby objects, driving away from anything that gets too close. The large bot will use the Xbox Kinect camera to detect and track the wandering small bot through color recognition, and then chase the small bot, tag it, pause to spin, and repeat. Milestones for this project were the following: obstacle avoidance with SLAM, color recognition and tracking, bump-sensor tagging, and testing / debugging the MATLAB code. Results and implications for alternative implementation are discussed; the project's successful outcome indicated that this project can be used for a basis in understanding basic but key aspects of the following: autonomous exploration over unknown areas (i.g., a rover for Mars, military or first-responder search and rescue missions, etc.); laser guided munitions; safe and efficient self-driving vehicles to reduce traffic and accidents; and introducing people of all backgrounds to Robotics.

Keywords: autonomous robot, Lidar sensor, MATLAB, object recognition, object tracking, obstacle avoidance, robotics, ROS, TurtleBot, Xbox Kinect

Introduction

Robots have many practical and impactful uses in day-to-day life. Any and all research into their functions and capabilities can help improve the overall quality of life for anyone; there currently exists many future implications in various subjects, including medical, environmental and leisure. The scope of this project revolves around the usage of ranging and visual detection devices such as a Lidar sensor (Light Detection and Ranging) and a camera to help with tasks such as obstacle avoidance and color tracking, respectively. These functions already serve helpful uses for machines such as Roombas and can be applied to many other devices to help with safety issues. Also, the color tracking portions overall have many possible applications for further work. While one of the primary reasons for the completion of this project was the amusement in possibly seeing a robot chase another one around rooms autonomously (let's be honest, we needed some amusement), the other main motivating reason was to understand key aspects of autonomous robotic swarms while developing new skills over the project's life. Research in autonomous robots is especially growing as the interest in solving real-world problems increases across many nations. Meaningful applications include education, transportation, assisting first-responder units, increasing efficiency in product handling inside warehouses, and assisting medical professionals.

Specific Aims - Proposal

The end goal of this project is to have at least two separate robots interacting with each other autonomously: one avoiding all nearby obstacles and the other searching for and chasing the first one. An obstacle avoidance SLAM algorithm will be applied to a small TurtleBot (Bot-S), while a chasing-algorithm will be deployed to the big TurtleBot (Bot-L). Bot-S will avoid obstacles using its attached Lidar sensor to detect object distances and relative locations for all points around the robot that will cross the Lidar sensor's plane of rotation. Its goal is to move around an unknown play area (MRDC 3336, or its connected hallway) without running into anything, while avoiding the chasing-robot, Bot-L. Using the Kinect camera, Bot-L will be programmed to search the surrounding environment to detect, chase (at various speeds), and then bump (i.e., tag) Bot-S based on specific colors attached to it. After tagging the other robot, which will be determined by either the bumper sensor readings or proximity between the robots, Bot-L will stop and spin to celebrate, and then await the next detection of Bot-S. This game of tag will continue based on the discretion of the observers, but ultimately, the goal is to have these robots play this game of tag for several rounds of tagging. The number of each robot type can be varied to make the game more interesting.

System Overview

The project requires at least two TurtleBots, one small (Bot-S) and one large (Bot-L), which are shown in **Figure 1**. Each TurtleBot is connected to one computer (laptop or desktop) and controlled through MATLAB, a router and ROS (accessed through a lab-provided laptop placed atop Bot-L). Additionally, rather than explicitly calling ROS commands, sensor data acquisition and control algorithms were implemented using the *Robotics System Toolbox Support Package for TurtleBot-BasedRobots*[1] from MATLAB.

The algorithm for Bot-S uses Lidar data to locate all objects around the robot over 360 degrees to efficiently navigate throughout the unknown area and obstacles shown in **Figure 2**. This SLAM-based obstacle avoidance code utilizes robot commands and an internally mapped point cloud of the surrounding environment to determine where Bot-S should go next. Nearby obstacles (e.g., objects to be avoided) are detected when they crossed the Lidar sensor's plane of rotation as done by the foot and legs in **Figure 3**, and a point-cloud is generated using the data from the Lidar sensor. Each point is stored as a distance and angle relative to the sensor's centroid and pose; the points are indexed in the code between 0 degrees and 359 degrees; in MATLAB, the 360 degree Lidar range corresponded to many groups of 360 rows. More specifically, the data is stored and interpreted as points at "0, 1, 2, …, 358, 359, 0, 1, 2, …" degrees repeatedly and respectively; it was assumed that the 0 degree and 360 degree mark were treated the same way by MATLAB. To prioritize what Bot-S should avoid, these points are indicated in the code as closely grouped points (or clusters) on the 2D-live generated point cloud map; a threshold was set around Bot-S such that anything within this specified range (based on the generated map data) would activate the avoidance code segment that essentially stated: move in the direction of the least dense and least intense point-cloud cluster.

The algorithm for Bot-L uses bumper sensors and the Xbox Kinect located near its base. Additionally, a yellow GoPro camera and sounds were later added: the GoPro showed the point of view of Bot-L shown by the left side of **Figure 3**, car engine sounds were added for when Bot-L detected and chased Bot-S, and "kobuki_msgs" sounds from MATLAB played when bumper sensors were hit with sufficient force. Colormask-files were made and then edited using MATLAB's Image Processing Toolbox; one file was made for orange, blue, and red, respectively. Bot-L's code uses these files as a basis for color-recognition done by the Xbox Kinect camera in order to search the surrounding environment and track Bot-S. A command sequence is then implemented to have the large bot chase at set speeds until Bot-S is either bumped (i.e., tag) or lost (out of detection range). After a successful tag, Bot-L will then stop, spin to celebrate, and await the next detection of Bot-S. Tagging was counted if, and only during the chase, the bumper sensor was hit or if Bot-S was lost after Bot-L entered the avoidance threshold of Bot-S (i.e., if Bot-S was underneath the Xbox Kinect sensor with the sensor positioned on the lower portion of Bot-L).

Experimental Results

The milestones for this project were the following: obstacle avoidance with SLAM, color recognition and tracking, bump-sensor tagging, and testing / debugging the MATLAB code; which coincide with the timeline of major milestones submitted with the project proposal.

If an object was positioned outside of the threshold for Bot-S but made a high-quality cluster with the projected coordinate points, Bot-S would still detect and avoid it. Additionally, it avoided everything stationary that it encountered, and at times would compensate for the approaching Bot-L by detecting the moving nearby clusters (Bot-L had high-quality clusters). Thus, Bot-S successfully navigated while being chased throughout unknown environments (i.e., the Robotics lab room MRDC 3336, its chairs, students, backpacks and connecting hallway).

Difficulties:

Despite the following difficulties, the proposed game of TurtleBot Tag was successfully completed.

Bumper sensors were not as sensitive as previously expected and thus required an input force larger than the average contact force that resulted from bumping or tagging Bot-S. One solution was to update the code such that Bot-L would continue to chase Bot-S until the tri-colored tape-sleeve was completely out of Bot-L's viewing range (underneath the Xbox Kinect sensor); which, as a result, also created a sufficient distance-threshold around Bot-L for contact with Bot-S to exist despite the pose of Bot-S.

For one day of training, network connectivity yielded issues on each device used (laptops and desktops). One solution was to connect one laptop to the router via ethernet cable.

There was a limited amount of lab time available until the final weeks, yet the project required a considerable amount of hands on testing. The solution was to setup a virtual environment using Gazebo to simulate and test a large TurtleBot's color recognition and chasing MATLAB scripts. This enabled the testing of Bot-L outside of available lab hours and indeed ensured a timely completion of project timeline deadlines.

Conclusions

This project required the development of new skills in key aspects of autonomy such as obstacle avoidance with SLAM along with object recognition and tracking; thus, a better understanding has been achieved for controlling various TurtleBots using their respective sensor data with ROS and MATLAB. This project also showed some of the unexpected difficulties that arise when programming autonomous robots, such as network and computer connectivity issues. These problems can be just as difficult as the programming itself.

Extensions of this project include: Autonomous exploration in unknown terrain (i.g., a rover for Mars, military or first-responder search and rescue missions, etc.); laser guided munitions; safe and efficient self-driving vehicles to reduce traffic and accidents; and introducing people of all backgrounds to Robotics.

Although the project was successfully completed, with more time to research the sensor specs and analyze their respective limitations thoroughly, it is possible that this project could've been improved to have better self-controllability for each autonomous robot. Additionally, having at least 2 large TurtleBots and 3 small TurtleBots would have been a great alternative. Replacing the wireless computer connections with hardline (ethernet) connections to reduce input delay would drastically improve the turtlebots' performance. The desktop computers used were also older models and often caused the robots to respond slowly when Matlab was performing multiple processing tasks. Using newer computers and other adjustments could also enable obstacle avoidance on Bot-L, along with a more sophisticated path planning algorithm on both Turtlebots.

Appendix:

Figures

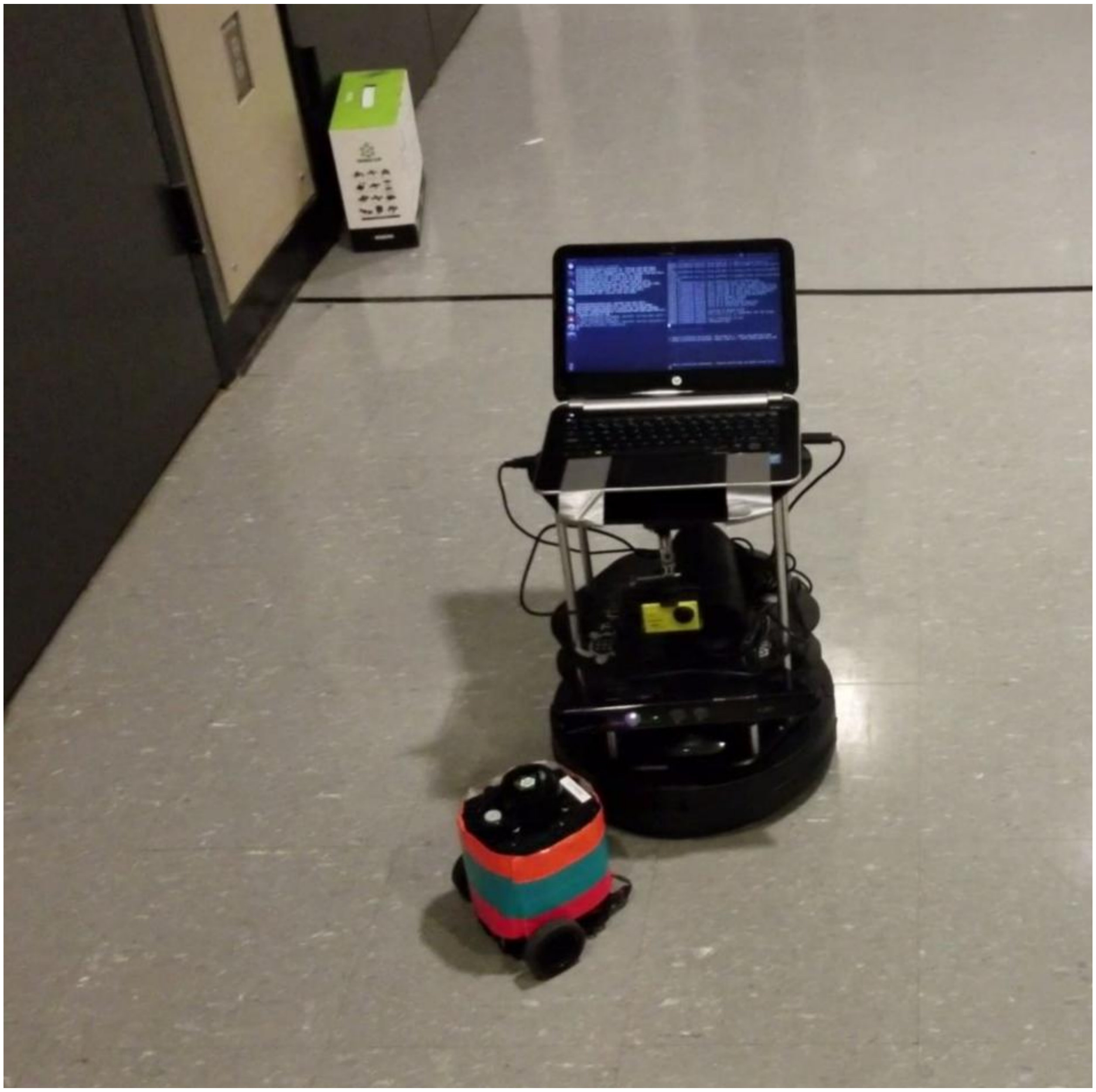

Figure 1: Small (front left) and Large (back right) TurtleBots

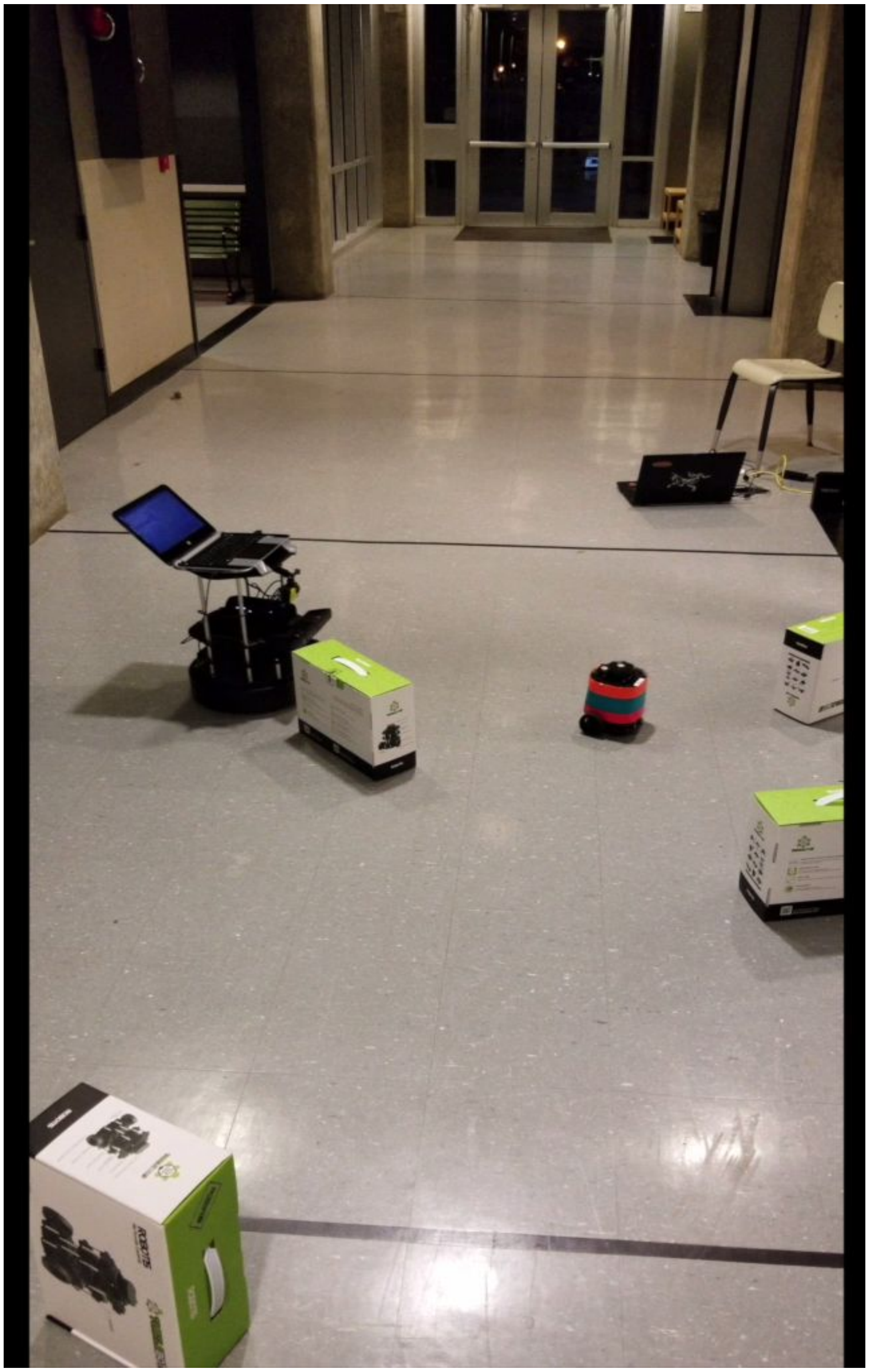

Figure 2.a: TurtleBots in MRDC hall with obstacles (boxes)

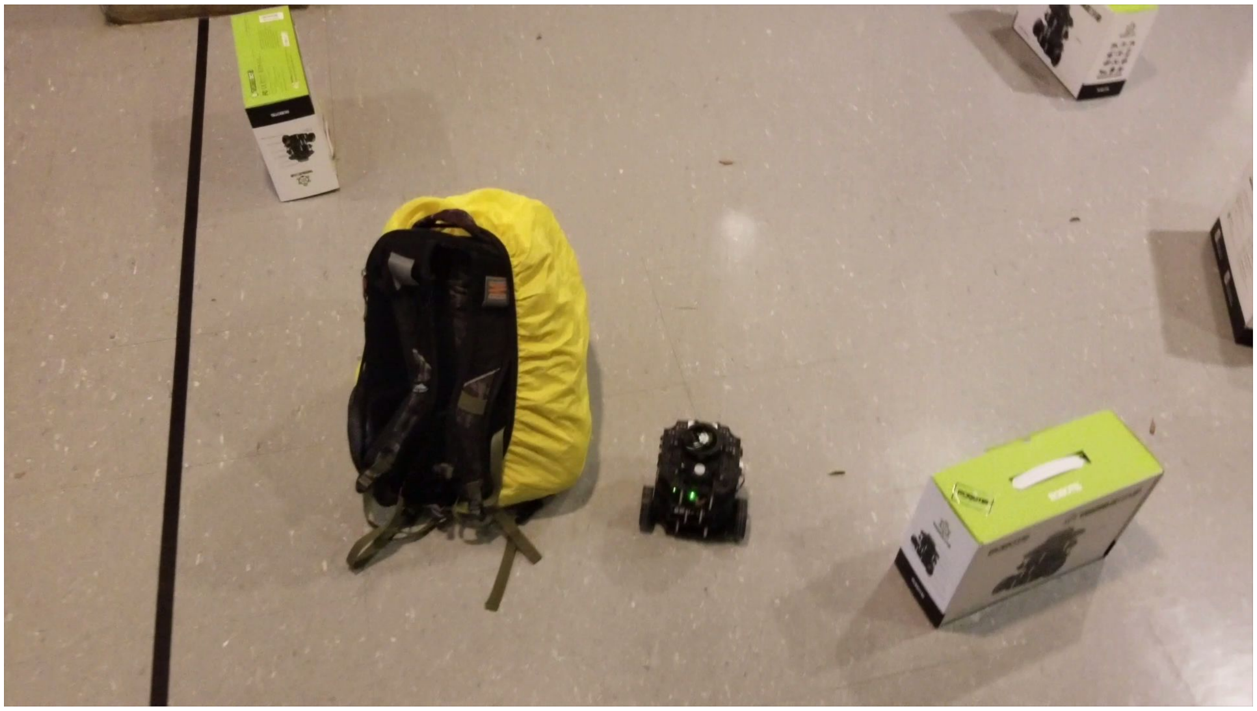

Figure 2.b: Bot-S navigating around obstacles in MRDC hall

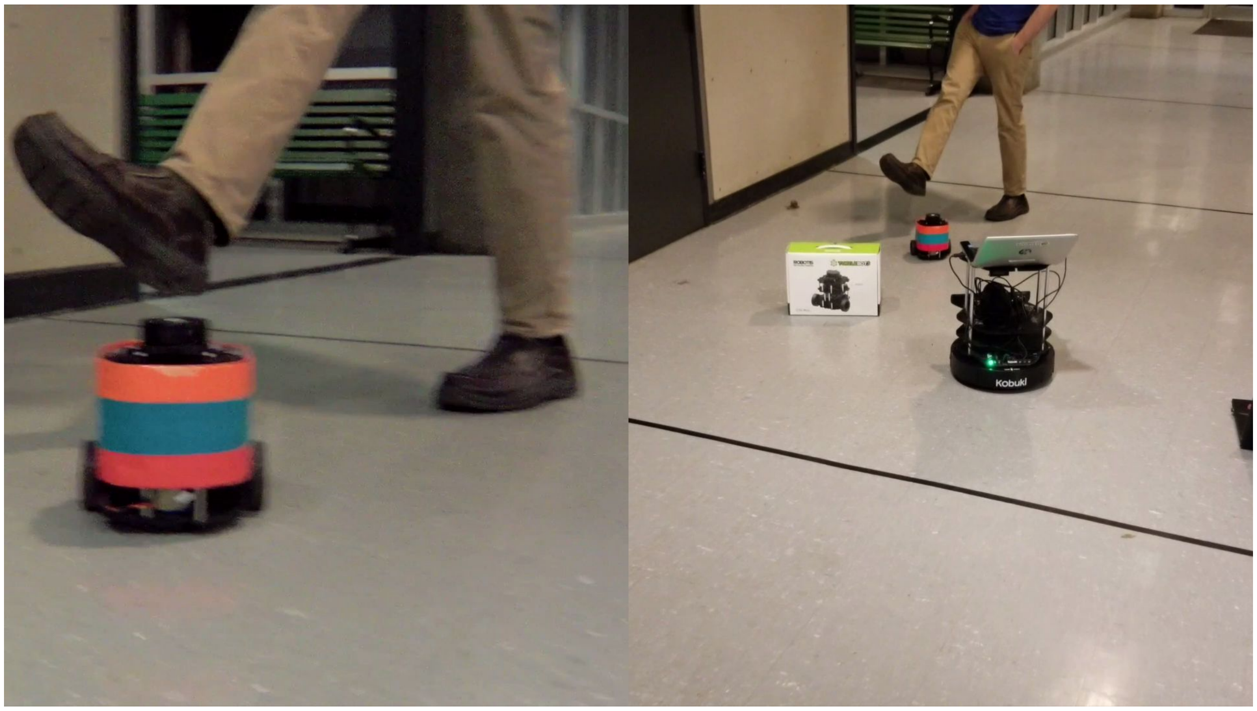

Figure 3: Picture-frames from GoPro footage (left) and cell phone (right)

References

[1] MathWorks. *Robotics System Toolbox Support Package for TurtleBot-Based Robots*. (https://www.mathworks.com/hardware-support/turtlebot.html). Retrieved Nov. 2018.