Final Report

Paul Epperson

Abstract

Gaming today features many exciting opportunities for new physically-interactive technologies such as augmented reality. But these features typically consist of interacting with objects that are reachable, such as simulated popups in a room that you view through a VR headset. However, distant but physically interactive objects have been demonstrated before by projects such as Chrome Web Lab. This project, RoboTag, started work towards building autonomous vehicles that could be manipulated at any distance via a virtual portal on the Internet for a human-vs-robot game scenario.

Introduction

RoboTag is a game between a robotic agent, controlled by a human, and a team of robotic agents, working independent of human control, but collaboratively. The goal of the human controller is to navigate their robot away from the team of adversarial robots for as long as possible. The goal of the enemy robot team is to trap the human controlled robot by moving into a positional configuration around the human controlled robot.

The most visible component of this project are the mobile robots. These are physical objects, effectively RC cars - but instead of control via radio, control is provided by a stable Internet connection. This project aimed to build a simple mobile robot to provide the gaming scenario in RoboTag. It was important to use an Internet connection to implement control of the mobile robots in order to serve the long distance communication aspect of this project. While there are many ways to provide control to a mobile robot, long distance connection is typically unfavorable for latency, loss, and dropped connections. However, an eventual goal of this project is to explore this in practice.

Autonomous robotic agents are controlled via algorithms (e.g. controls). They identify their current location respective to other objects via localization. Depending on the nature of the environment, the sensor, and the task, the location provided via sensor may be assumed to be an absolute reference. My hypothesis in starting this project was that RoboTag would be sufficiently functional with a lightweight go-to-goal algorithm and naive object avoidance. Go-to-goal algorithms ingest a pair of locational coordinates and will output the actuation necessary in the next timestep to achieve the optimal reduction in distance to that coordinate pair. In this case, object avoidance was necessary in the event that the autonomous robots encountered terrain that sufficiently impeded movement in the shortest distance between the robotic agent and the goal.

The main idea of RoboTag for me was to build a foundation that would draw from several subject areas that I had touched over internships and through my education at UCSD.

From the embedded systems perspective, I would be configuring an electronic system via the array of sensors needed to provide localization, navigation, and object avoidance. The team of adversarial robotic agents would have to work together, communicating through some network. And finally, each robotic agent would have to perform autonomous functions - namely locating its position relative to its teammates and obstacles, determining an accurate goal estimate, and executing a behavior to achieve moving there.

In this project I was able to complete:

- A suite of networking utilities that allowed me to transmit and receive data between multiple microprocessors
- An implementation of a finite state machine that implemented go-to-goal with object avoidance
- Three small RC vehicles with GPS, ultrasonic, magnetometer, and accelerometer sensor input

Technical Material

Hardware

Each robotic vehicle consisted of the same components. I used a pre-laser cut chassis cutout, along with two DC electric motors, input 3V as the wheels, motors, and body. I soldered the two DC electric motors to positive and negative wires, one pair each. Each of these wires connected to separate slots in a motor controller. The motor controller was directly powered by a 6V power source in the form of four AA batteries. Four female to female jumper wires were connected from the motor controller to the Raspberry Pi Zero W's GPIO pins. Note the GPIO pin header had to be soldered on. These four wires carried forward and reverse signals for left and right motors, providing full range of motion.

The robotic vehicle had an ultrasonic sensor for object avoidance. The ultrasonic sensor needed continual voltage in and was connected to the 5V output GPIO pin of the Raspberry Pi Zero W, as well as grounded back to the Raspberry Pi Zero W's GPIO header. A small electronic circuit was put on the breadboard to reduce the output voltage of the ultrasonic signal, as supposedly the GPIO header pins cannot read input past 3V voltage. A small Python script was written to trigger radio signals and output the estimated distance in centimeters based on the estimated time of flight of the received radio echo. The GPS, accelerometer, and compass could be directly connected to GPIO. A parsing script for the GPSD daemon extracted the estimated GPS coordinates from /dev/ttyAMA0 and converted to two dimensional global surface coordinates. Three non-power, non-ground pins were used each for accelerometer and magnetometer. After configuring the I2C parameters for the Raspberry Pi Zero W's, two scripts were written to parse both the magnetometer and accelerometer output.

Networking

Each robotic vehicle had a scp'd directory of network utilities written in C. I used these scripts to establish socket connections with the other Raspberry Pi Zero W's. Each Raspberry Pi Zero W was assigned a static IP (in retrospect, I learned from others that there are better ways to do this automatically). Each Raspberry Pi Zero W broadcasted its current state information sequentially, e.g. in a ring. The sockets wrote to global buffers used by the autonomous navigation

Milestones

Complete schematic for onboard sensing, computing

Although it took a lot longer than expected, I was able to figure out how to read in data from each of the sensors (GPS, accelerometer, ultrasonic, magnetometer). In addition, I used a motor controller to control the two wheels of the particular robot chassis I ended up using.

Complete robot prototype

100+ soldered connections, 50+ jumper wires, several batteries, breadboards, resistors, and a couple of burnt out LEDs later, I was able to get three separate wheeled vehicles to move in a general direction while avoiding objects, at the same time.

Complete communication server

This milestone changed from a centralized server implementation into a less centralized network. Each robot communicated updated state information with each other over TCP sockets. I was able to simulate this on multiple Raspberry Pi's with artificial data, so I achieved this milestone.

Complete formation + rendezvous algorithm

This milestone was not completed. While I rewrote a working implementation of an existing version of the navigation algorithm for use on the Raspberry Pi, I was unable to achieve any formation at all - the GPS input data was not feasible for the limited flat areas that the smaller robot chassis could handle.

Complete application for human control of robot

This milestone is sufficiently completed. The first thing I wrote after connecting the motor controller was implement keyboard based control of the robotic vehicles. While this would ideally be more user friendly, it was effective.

Complete autonomous robotic team

There was definite progress made towards this milestone, but it is still incomplete. There are 3 robotic chassis, wired up with sensors and batteries, that can move around at the same

time and print the shared data between all the vehicles. However, their autonomy is not even remotely tested, and unless I can get better positioning data, they won't be able to perform any autonomous goals.

Conclusion

This project laid some solid groundwork for a simple adversarial robotic gaming scenario. With more accurate localization, it stands to reason that the required components of the game can be performed - chase, as well as capture. If so, this project will serve as another proof of concept for virtual portals enabling openly accessible interaction with hardware. This would enable remote control of hardware that is safe and open.