BALL IS LIFE: The Autonomous Trash Can Project

Hyungsuk (Peter), Kim, Maruchi Kim, Amit Patankar, Spencer Schack

I. Abstract

This project designs and implements an autonmous moving trash can that will calculate the trajectory of a thrown trash projectile, actuate the motors of a moving base, and attempt to catch the trash projectile. The sensor(s) utilized will be an Xbox Kinect Camera. Image and depth data from the Kinect Camera will be sent to a MacOS computer to determine the trajectory of the trash projectile in 3D space. Location of the trash can will be implemented via AR (Augmented Reality) tags using ROS (Robot Operating System). The MacOS computer will then determine the appropriate motor speed and wheel direction to reach the projected destination of the trash projectile and transmit this information to an Arduino on the moving trash can via Bluetooth Low Energy.

A. Notes

From the original project charter we've decided to scale down the project by making a trash catcher instead of a basketball catcher. We realized that catching a basketball would be too difficult to implement with the resource constraints of the project. Basketballs travel much further at higher rates of speed and have the potential to severely damage equipment based upon their mass and momentum.

From the milestone update, we've decided to detach the Xbox Kinect Camera from the trash can and forego the use of a Raspberry Pi. The Raspberry Pi did not have the required processing power to calculate the trajectory of the thrown projectile in a limited amount of time.

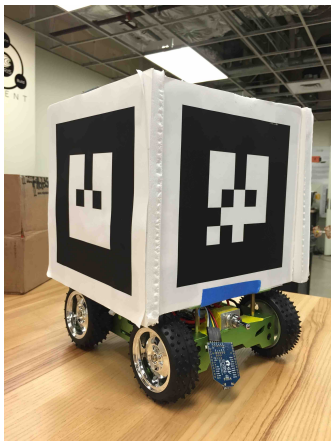


Fig. 1: Working Prototype of Autonomous Trash Can

II. Implementation

A. Software

We designed and implemented the following modules: a Find module in Python to utilize image and infrared data from the Xbox Kinect Camera to track trash projectile in 3D space using Ros and OpenCV; a Trajectory module in Python to utilize 3D space coordinates to determine trajectory of trash projectile using Numpy and Scipy. A Navigation module in Python to convert the destination coordinates to motor speeds to communicate to the Arduino on the moving trash can. All modules were developed on the MacOS platform.

Fig. 2: Xbox Kinect Camera

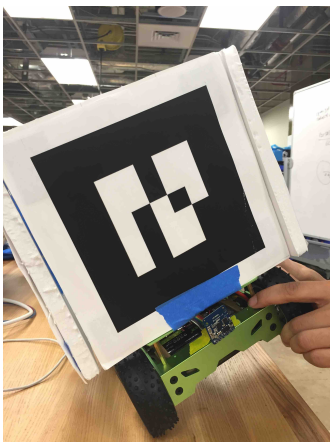


Fig. 3: Augmented Reality Tag

B. Software Notes

All modules were originally implemented to be compatible to run on a Raspberry Pi, but were modified to run on MacOS after abandonment of the Raspberry Pi for final implementation.

Fig. 4: Hercules 4WD Robotic Platform

C. Hardware

We ordered and assembled the Hercules 4WD robot kit which sports a 15A 6V-20V Motor Controller with full H-Bridges and an ATMEGA 168 board. By having independent motors on each wheel, the robot base has the capability of spinning in place. Components we added to augment the kit include a BLE module for communication, 9 DOF IMU for orientation data, and a power switch. From the MacOS computer, the robot receives x-y coordinates, and then spins and drives forward or backward accordingly to catch the trash projectile.

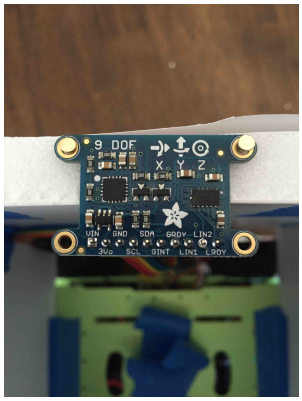


Fig. 5: Adafruit 9-DOF IMU Breakout

D. Hardware Notes

The team had originally bought bare bone components to attempt to build the base from scratch. However, some complications arose through gears fitting incorrectly, and the motor driver board being unable to support the motors purchased. As such, we decided to move to the Hercules robot platform to focus on adding BLE support and 9DOF IMU data. The additional time has also allowed time to better focus on embedded algorithms discussed later in the report.

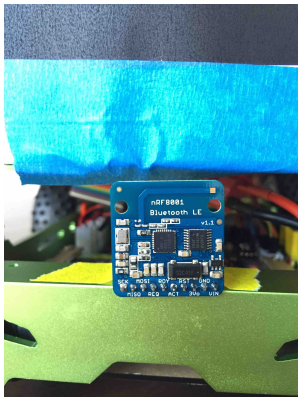


Fig. 6: Adafruit BLE nRF8001 Breakout

III. Algorithms

A. Projectile Detection

To detect and distinguish the projectile in 3D space, we used the depth data from the Kinect and filtered the data with OpenCV. First, we filtered out anything farther than 3 meters from the camera. After that, we ran the image through an OpenCV function called `findContours` which returns a list of shapes found in the image. These shapes are then scored by size, shape, and distance and the best candidate is picked from the list. The center point of the shape is then projected into 3D space, which is done by using the distance of the object from the camera and the field of view of the Kinect which is 43°vertically and 57°horizontally.

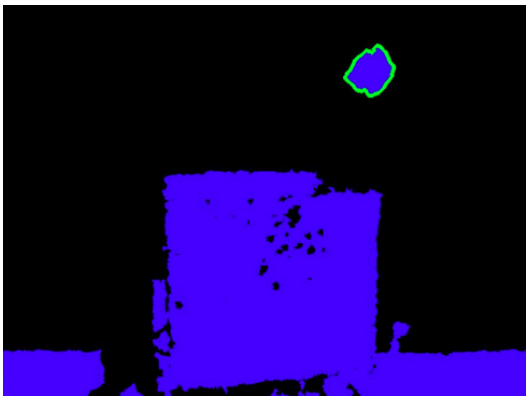


Fig. 7: Projectile Detection via OpenCV

B. Trajectory Prediction

To predict that trajectory of the projectile, we first sample 3 points tracking the trash projectile's progression through 3D space. Then using the method of least squares, we fit the sampled points to two parabolas: one modeling X-Z and the other modeling Y-Z. Using the intersection of these two parabolas we can predict the trajectory of the projectile. Before passing through the prediction algorithm, the points we sample from the Kinect are transformed from the camera's frame to the robot's frame by ROS using the

position and orientation data provided by the AR tags on the robot.

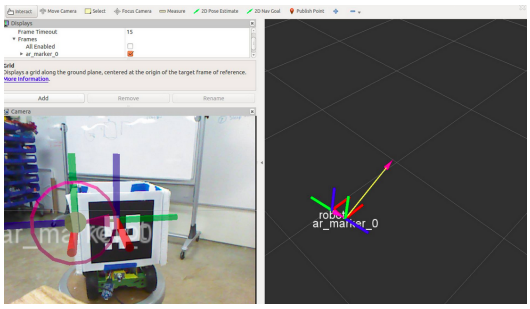


Fig. 8: 3D Visualization of Robot and Projectile in rvis

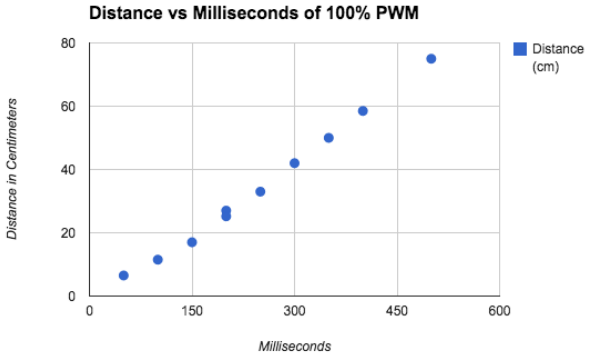


Fig. 10: Time vs. Distance of 100% PWM

C. Motor Actuation

The brushed DC motors are controlled through a NMOSFET H-Bridge Motor Controller. The MOSFETs are controlled through the microcontroller which runs at 5V logic. By providing the gates of the MOSFETs with 5V, the MOSFETs turn on for a certain percentage of the duty cycle, also known as Pulse Width Modulation (PWM). Figure 9 goes into detail of the workings of PWM. When going straight, the robot's duty cycle is at 100 percent, and while spinning in place the duty cycle is at 80 percent.

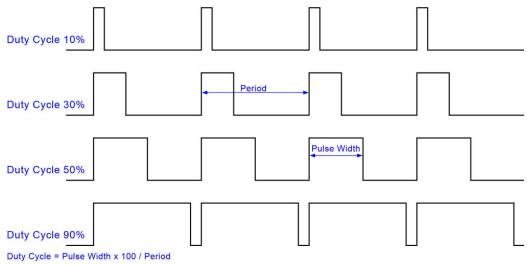
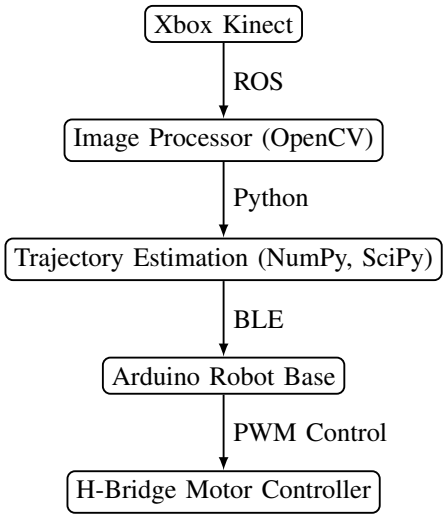


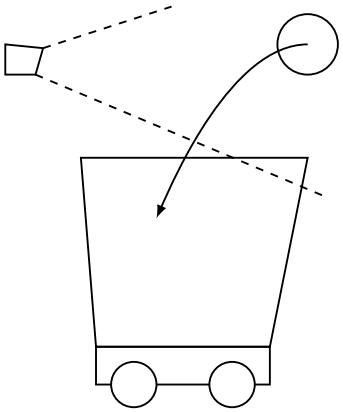
Fig. 9: Graph Detailing PWM

With PWM, we can control our robot's acceleration. However our robot needed to be aware of how far it has traveled. We ran an experiment to see how far our robot would travel under variable amounts of milliseconds under 100 % PWM. The results of our experiment show that the robot's response was quite linear, and so the way we controlled how far our robot traveled was by controlling our motors at 100 % PWM for x milliseconds.

IV. Models



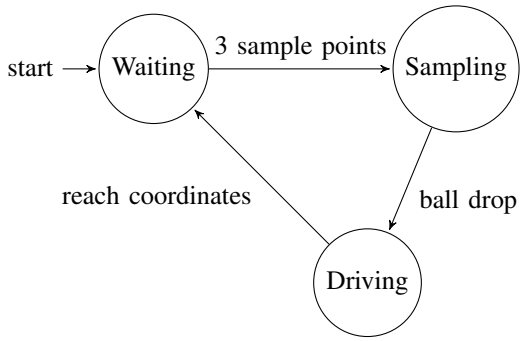
High level overview of software and hardware components and dependencies.



Layout of trash can and Kinect camera.

V. Analysis

A. Finite State Machine Model



Finite State Machine for Trajectory Module with three states for

B. Motor Speed and Torque Requirements

The required torque for the motors will be based off the speed at which we need to arrive at a goal state. This speed will depend on the time it takes to process the depth data from the Kinect and the amount of time it takes for an average piece of trash to fall to the ground.

We take some samples to see how long it takes for a piece of trash to fall to the ground:

Trial	Time (s)
1	0.5
2	0.4
3	0.3
avg.	0.4

Our goal is for the trash can to catch trash within a 2 meter radius. Assuming the worst case:

$$a = \frac{2d}{t^2} = \frac{4\text{m}}{0.16\text{s}} = 25\text{ms}^{-2}$$

We also need to estimate the total weight:

Component	Mass (kg)
Trash can	0.5
Kinect	0.5
Raspberry Pi	0.045
Wheels	0.1
Chassis	0.7
Total	**1.845**

So the required force of the wheel against the ground will need to be:

$$\begin{aligned} F &= ma \\ &= 1.845\text{kg} \times 25\text{ms}^{-2} \\ &= 46.125\text{kgms}^{-2} \end{aligned}$$

The required torque will be the acceleration times the radius of the wheels:

$$\begin{aligned} \tau &= Fr \\ &= 46.125\text{kgms}^{-2} \times 0.03\text{m} \\ &= 1.38375\text{kgm}^2\text{s}^{-2} \end{aligned}$$

The torque of the motors is:

$$\begin{aligned} 255\text{gcm} &= 0.00255\text{kgm} \\ &= 9.8\text{m/s}^2 \times 0.00255\text{kgm} \\ &= 0.02499\text{kgm}^2\text{s}^{-2} \end{aligned}$$

The the gear ratio from the motor to the wheel will need to be:

$$\frac{1.38375}{0.02499} = 55.37214886 \approx 55:1$$

C. Angle Calculation

Using the x and y coordinates provided by the computer we use an inverse tan function to calculate the angle that we are required to turn. We subtract 180 degrees from the angle measurement to give us an angle in the range from (-180,180).

The measured angle where x and y are in meters, where a deduction of 180 is present so that the robot is aware of left and right turns.:

$$\theta = \tan^{-1}(\frac{x}{y}) - 180 \tag{1}$$

To turn to the appropriate angle we created a linear time based model that actuated the motors for a specified time.

D. Feasible Catch Area

By calculating the maximum speed of the motors and incorporating the average time a trash projectile remains in flight, we are able to estimate the feasible catch area for our autonmous trash can.The worst case scenario for turning is 90 degrees which takes roughly 300 ms. Thus without the turn in the best case scenario we can accelerate to the target point and in the worst case scenario we turn 90 degrees before accelerating. We have assumed we will have 500 ms before the trash hits the ground and have measured our wheels to be 8cm in diameter.

With Turning

$$width = \frac{2\pi \times 0.04\text{m} \times 310rpm}{60\text{s}} \times 0.2\text{s} = 0.26\text{m}$$

Without Turning

$$height = \frac{2\pi \times 0.04\text{m} \times 310rpm}{60\text{s}} \times 0.5\text{s} = 0.64\text{m}$$

Figure 11 outlines the robot's "catch area", basically the feasible drop zone given the robot's physical capabilities.

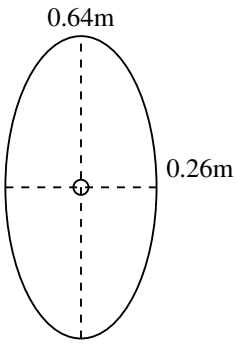


Fig. 11: Feasible Catch Zone

VI. Difficulties

A. Software

We experienced a lot of trouble setting up the Xbox Kinect Camera on a Raspberry Pi due to multiple dependency issues present on Raspbian's (Raspberry Pi's Ubuntu-based OS) outdated package manager. We circumvented this issue by deciding not to use the Raspberry Pi in our final implementation.

Another issue we faced was latency from the Kinect, our software algorithms, BLE packets, and eventually motor actuation. Due to this we had to preemptively control our robot to move below the trash and then drop it into the trash can.

B. Hardware

We experienced a trouble when we initially fixed a plastic trash can on top of our robot base. Due to the dimensions and mass of the plastic trash can, the robot would jerk and sometimes topple while attempting to rotate because of the high center of gravity.

Another complication that arose was with the 9DOF IMU. Initially we had placed it close to the robot base, but as the brushed DC motors began spinning, the team found that the magnetometer onboard the IMU was very susceptible to magnetic noise introduced by the motors. Consequently, the IMU was moved toward the lip of the trash can (seen in Figure 2). Even then, we found that the IMU orientation data was extremely nonlinear, we found much better results through just turning and stopping based on a variable amount of time.

VII. Conclusion

This project involved a lot of key concepts from class. Modeling of physical dynamics was seen through our motor speed and torque requirements analysis. Reliable realtime behavior was exemplified through the way we implemented distance and angle control. We went through modal analysis governed by FSMs through our finite state machines. Finally we implemented a simple realtime network through BLE communication between our computing platform and our robot.

Despite all of this and other careful planning, we ran into various problems that were practically impossible to foresee when applying our design to real world systems. However, through robust modeling and frequent design iterations, we were able to successfully design and implement a working prototype of our autonomous trash can.

VIII. Acknowledgments

Professor Edward A. Lee
Professor Alberto L. Sangiovanni-Vincentelli
GSI Antonio Ianopollo
CITRIS Invention Lab