Luca Amblard, Dan Barychev, Hana Frluckaj
18-500: ECE Capstone
21 Mar 2020

Statement of Work

New Challenges

To prevent the rapid spread of COVID-19, ECE Capstone has been converted into a remote course. This is a drastic change to the structure of our project, which was dependent on several integrated hardware modules. We can no longer work together on building our previous vision, a motorized device that moves to anticipate an object thrown given data from motion sensors of the user's throwing hand. Therefore we will be moving to a simulated version of the project.

Redesign Plan

We've decided to retain the arm component with the same IMU I2C circuit that each member can work with in parallel, while all three of us pivot to build a simulation for the ball flying through the air and the robot catching it. We will keep the hand-wrist IMU configuration largely the same. Since the robot will be simulated, we won't use a Jetson Nano, simplifying how the IMUs communicate to our data processing system. Instead, the Particle Photon will communicate data to a Processing simulation showing the ball travelling and the robot moving to its predicted landing location. All computation (i.e. raw data conversion, kalman filters, parabolic landing estimation, PID control system) will be executed on a laptop. We chose a Processing visualization software package due to its ease of use in terms of responding to real time data and compatibility with a number of input formats (e.g. Arduino serial out).

Simulation and Process

The simulation will show five things: the user and the robot starting positions, the target location, the user's pre throw and the robot moving during the pre throw, the ball traveling to the actual landing location (horizontal/vertical speeds, altitude, and the air time will be displayed), the robot moving to the predicted landing location (speeds for the ball and the robot will calculated).

The simulation will prompt the user for the actual landing location and the time of travel of the ball. The thrower will actually throw the ball in real-life to determine this data. The system will record both the pre throw and actual throw. At the actual throw, we'll measure the ball's actual landing location and its time of travel and this data will be manually entered into the simulator. With all of the data from the throw present, the Processing simulation will render a 2D animation of the robot's reactions to each of the pre throws and then the actual throw. This 2D animation will come from applying matrix operations to the incoming data within a program, which will then help us render an actual image to be displayed. This will also include an estimate as to whether the robot was able to truly catch the ball based on the real landing position.

All throws will occur on a grid-like surface where X/Y positions are clearly defined. This grid will allow us to determine the actual landing location of the ball. In order to accurately measure distance, we will throw a hacky sack to prevent rolling/bouncing. To determine the actual time of flight, the ball will be filmed traveling through the air using a mobile phone slow motion camera.

There was going to be an IMU on the robot and a Kalman filter run on the Jetson Nano for the robot to keep track of the distance left to travel to reach the target location. Now, the simulation will assume that “perfect" IMU readings are supplied to the robot from some "magic IMU”. Additionally, we will not be using the omni-wheel chassis but we will still indicate the rotation speed of each motor. Dan has the omni-wheel chassis so he will experiment with it to determine the ratio of rotation speeds of the wheels needed to make the robot move in a particular direction. Another issue that arises is that we will not have the actual rotation speeds of the motors from the encoders. Our solution to this problem is to add a random error in the range of the motor error to the target speed and use this as the actual speed for the PID control system.

Verification

The simulation will determine the distance between the predicted landing location and the actual landing location and also the angle between them to determine the accuracy of the prediction. The distance of the true landing location from the estimated landing location will be calculated by comparing the X/Y positions of the two locations. Similarly, the angle between them can be easily determined by drawing two imaginary lines back to the user’s position (0,0).

Updated Architecture Diagram