Robotic Car to Navigate Natural Landscapes

Submitted by:
Team 8
Katie Bertcher, Elisabeth Jahnke, Amir Naser, Maya Pandya, Adithya Subbiah, Jack Winkelried

Advisors:
Dr. Shai Revzen
David Kucher
Rishank Nair

April 21st, 2021

Electrical and Computer Engineering Department
University of Michigan-Ann Arbor

Table of Contents

Background

Smart harvesting is a known field of application for signal processing and computer vision (“See and Spray Machines”). Due to the perennial nature of asparagus, its sporadic growth, and the need to harvest daily, widespread automated asparagus harvesting has yet to be achieved. Automated harvesting would make asparagus much more affordable, as around 75% of production costs come from labor (Satran, 2017). Current asparagus harvesters largely use rail systems or trenches to navigate (Zitter, 2020; “GARotics,” 2012).

Motivation

Creating a vehicle capable of navigating an uneven boundary like that of an asparagus crop row would bring new technology to the current set of harvesters. Navigation without need for rails and trenches would allow greater flexibility in field planning. This car body would also have the potential to be adapted for other types of crops beyond asparagus.

Navigation is crucial to automated harvesting because the machine needs to drive close enough to harvest without running over any of the crops. Using a distance sensor and camera, the car will be able to detect the landscape, process the image to show the location of the crop, and use that information to inform its path.

To develop this boundary following vehicle, we used colored stakes in replacement of asparagus to facilitate easier, more consistent testing. Assumptions about consistent color and width are made which may only partially hold true for real asparagus, but allow us to show proof of concept for this type of row following ability.

References/Prior Work

This project explores three primary fields that each have some prior work and precedent surrounding them. These include RC car control, computer vision for distance detection, and PID control for the RC car.

1. RC Car “Hacking”

The store bought RC car needed to be “hacked” to be controllable by our system. Though there are no instructions on how to “hack” the particular model of car, RC cars are similar enough under the hood such that by following the descriptions of how others have hacked other RC models, we properly controlled the model we have (mjrovai, 2017). Taking notes from this example, we evaluated the existing motors of the car and connected our arduino control to the car using an H-bridge.

2. Object and Row Detection

Using the PiCamera and distance sensors, we can detect the position of the asparagus. Object detection has a long precedence in the field of computer vision. For instance, we implemented basic object detection in lab 2 for this class. Thresholding and contour detection are common ways to find the location of an object of known color. There are many other object detection algorithms that go beyond the scope of this project (Choudhury, 2021). After detection, there is also precedence for calculating distance from camera input. By calibrating the focal length of the camera, the distance of the camera from an object of known length can be found (Mahbub). It is also important to note that the camera follows a pinhole model, and thus extra steps must be taken to accurately measure object width as viewed by the camera (Hata and Savarese).

3. Controlling the Car

Once we determine where the car needs to go, it is imperative the car is able to follow that path. This will be done by implementing a PID control algorithm. There is precedence for controlling a car via PID control for both line following and sensed boundaries (Nova, 2020; Chong et al.). There are also well documented control design features in MatLab ("Open PID Tuner"). Based on prior designs, it is clear that our car's control system will need to be based on a state diagram and will work through the H-bridge attached to the car in order to control the motors.

High Level Design

We built a vehicle capable of autonomously navigating a row of stakes that will stop in front of obstacles. This architecture builds upon previous projects on line following robots by creating an autonomous robot able to navigate an environment where there are no predefined lines or boundaries. There are only rows of stakes and other environmental variables which the robot must be able to detect and navigate. This can be used as a precursor to alternative smart harvesting technology.

Our system works by utilizing the RC Car's 7.84V/850 mA rechargeable battery to power our motor system and Arduino Uno and a 5V battery to power our Raspberry Pi. The Raspberry Pi has a Pi camera connected to it which uses DSP algorithms learned from class to identify where the stake is. We use the findContours after using H, V, R, B thresholding on the image. Then the algorithm draws a vertical box around the whole stake, and another box that will outline the stake (even if it is at an angle). This method allows us to get values for the width of the stakes and the distance between the center of the stake to the center of the image for use in distance calculations. The distance is then calculated by using focal length to calculate the location of the stakes from their position and apparent width in the image. A linear regression approximates the line of stakes that the car can follow.

The distance information is then sent through a UART communications protocol to the Arduino Uno. This information as well as information from the distance sensors for obstacle detection is inputted into the Arduino Uno's onboard PID loop. We use a PID controller to correct the distance the car needs to be from the wall. We decided that our robot must be able to follow the

stakes at a distance of ten inches. This controller calculates the precise PWM signals to send to the servo motor and RC car brushless motor. This ultimately allows the car to drive along the stakes without driving them over and stop before hitting a boundary.

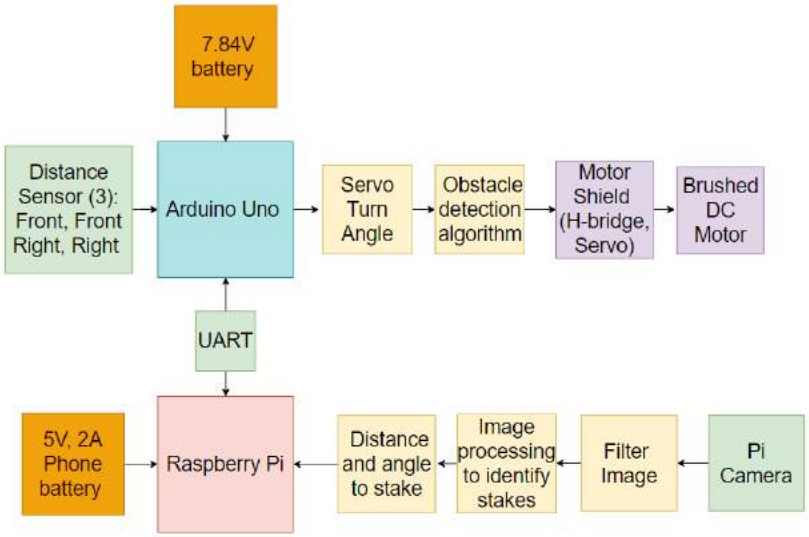

Figure 1: High Level Block Diagram of car system

The RC Car fully constructed is shown below. We utilized the original RC Car Brushed DC Motor and added an Adafruit Motor Shield and Arduino to provide speed control. We are able to control the duty cycle and car direction through the H-bridge on the Adafruit motor shield. The car steering system uses the original RC car steering linkage paired with the servo motor linkage for a ninety degree steering span. We mounted the Raspberry Pi, Camera, and battery on the back of the car, distance sensor on front, and the Arduino architecture under the hood of the car.

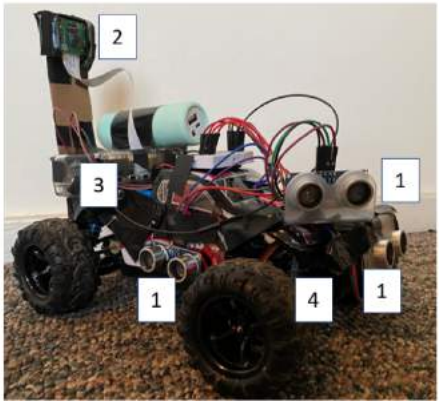

Figure 2: RC Car Fully constructed
(1) Distance Sensors (2) Pi Camera (3) Raspberry Pi and Battery (4) Arduino Uno, Servo Motor, Brushed DC Motor

Technical Challenges

There were several challenges related to the computer vision program that provided additional challenges throughout the development process. One big challenge was the sensitivity the computer vision system had towards lighting conditions. A cloudy day vs a sunny day had huge effects on the accuracy of the car being able to detect the red stakes. Even individual clouds forced the team to have to stop and re-calibrate for the new lighting conditions. This is why our final testing arena shifted from being outdoors to an ideal environment indoors in a well lit hallway. The team also ran into issues with the design of the Raspberry Pi camera. The RasPi camera uses a pinhole camera model to map 3D objects onto a planar surface. This causes inaccuracies related to the observed width of the stakes vs their actual width as instead of mapping the image to a cylindrical surface, which would have preserved the actual width data, the camera maps the world to a planar surface warping the perceived width.

The team also ran into issues regarding the UART communication bus as well as the chosen linearization of the car model. The chosen model of the car was a linearized version for simpler control design, but since the car is in actuality a non-linear model, this approximation only worked so well. Unfortunately due to this, the car can't correct horizontally as well. The UART communication bus also gave us issues as we found that the Raspberry Pi loop was running much faster than the Arduino loop, this caused a backup in the UART buffer on the Arduino as the Pi was constantly streaming. The Arduino would then process stale commands, creating huge issues in delay of the car. To mitigate this, the team chose an answer and response solution where the Arduino would indicate when it was ready for an UART packet by holding a GPIO pin high to the Pi. The Pi would then only send a single UART packet at that point in time.

System Testing

When our team reached individual milestones in the vehicle's development, testing and calibration procedures proved to be necessary. By incorporating such procedures, we could confirm that the vehicle was behaving as it should before proceeding with further developments. Each of these procedures also had to take into account the current testing course being utilized.

1. Controls calibration

In the first phase of the project, most of the team's testing efforts were placed on calibrating our control system gains. If the control algorithm were to produce unstable motion by just driving the car alone, then it would be extremely difficult to accurately respond to processed camera feed. We applied the Ziegler-Nichols method to tune the gains and ran our vehicle alongside a concrete wall to provide continuous distance sensor input. The Ziegler-Nichols process first requires users to set the derivative and integral controller gains to zero and change the proportional gain such that the car will oscillate in a controlled manner while driving along the wall. Then, using the time period of those oscillations as well as the corresponding proportional gain, the starting values for the complete set of PID gains can be calculated. From this starting point, we could then further tune our PID gains to allow for more effective control.

Ziegler–Nichols Equations for a PID Controller

$K_P = 0.45 * K_{cr}$
Where Kcr is the KP value where you get sustained Oscillations
$T_i = 0.5 * P_{cr}$
$T_d = 0.125 * P_{cr}$
Where Pcr is the period of the sustained oscillations
$K_I = \frac{KP}{T_i}$
$K_D = K_P * T_d$

Equation Set 1: Ziegler-Nichols Tuning Equations Used for Initial Values

2. Test track

When initially tuning our PID gains, we ran our vehicle alongside a long, continuous concrete wall. Then, to test the effectiveness of our obstacle detection system, we placed a cardboard obstacle at some point along the track. This setup is depicted in Figure 3.

Figure 3: Concrete Wall Test Course with Obstacle

Once our team finished calibration the PID controller, successfully implemented the obstacle detection system, and finished developing our camera-based object detection software, we transitioned to testing on our final track setup. This track lies within an indoor hallway where lighting remains constant. Like the first track, there is a wall running along the side, but now, we have placed red PVC stakes in front of the wall to represent an asparagus plant ready for harvest. To prevent these stakes from reflecting on the floor, we covered the hallway tiling with non-reflective bed sheets. Also, to mark the end of the course, we placed a cardboard obstacle that will trigger the obstacle detection system. This course is displayed in Figure 4.

Figure 4: Final Vehicle Test Course

3. Camera tests

At the beginning of this process, we calibrated the focal length of the camera in pixels by taking measurements of the apparent width of a 1.5 inch ball as it was moved toward and away from the camera. We then further tuned this focal distance using trials with the actual stakes. Initial trials were also conducted to get baseline hue, value, red, and blue values for the stakes, but due to lighting issues these values needed to be re-tuned for every new location.

After we constructed the final testing course, our team set up the pi camera facing the stakes and tuned our hue, value, red, and blue thresholding parameters to identify the stakes under the given lighting conditions. Once we determined these ideal threshold values, we modified those values in the code, fully preparing the car to detect the stakes.

Results

The RC car was ultimately able to successfully navigate a row of stakes in ideal lighting conditions. As long as all the stakes were consistently lit the same, with no interference from the sun/clouds altering the brightness of the stakes, the car was able to accurately detect all stakes within a reasonable distance. This is even true if the stakes are at an angle to the camera and are not completely vertical with respect to the camera, as shown in Figure 5. This ensures that if the car camera ever gets offset, or if the stakes are ever not vertically straight up, our system will perform the same.

With the detected stakes, we were then able to get the location of the stakes relative to the camera and were able to create a linear regression for a “wall” of stakes that the car can use as a boundary to follow. Figure 6 shows the linear regression plot, where the line is essentially the “wall” boundary that the car will follow, which was composed by using the locations of the stakes which are represented by the points on the plot. As long as there are stakes within the view of the camera, the RC car can essentially create its own path to follow while removing any outliers that may show up in case random objects in the background get detected by the camera. This differs from line following robots since line following robots essentially look down at the floor to determine whether they are following the correct boundary. Instead, the RC car will always be

able to detect obstacles in front of it while following the linear regression plot that was calculated.

We were then able to calculate and output the distance that the car is from the “wall” of stakes, which is represented in Figure 7. This is crucial to the overall pathfinding of the RC car since the car is designed to stay a set distance away from the “wall” of stakes. For our project, the RC car is set to stay 10 inches away from the stakes which gives the camera the best view of upcoming stakes in order to calculate and set the future path of the RC car.

With all of this in mind, the car is able to accurately detect stakes, create a linear regression for the path, calculate the distance the car is from the “wall” of stakes, and ultimately drive along the path of stakes.

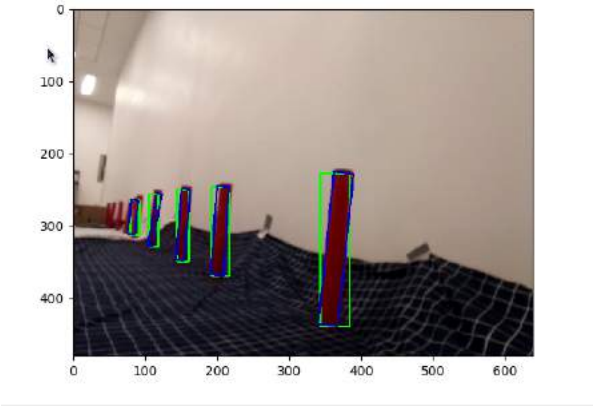

Figure 5: Camera Detection of Stakes

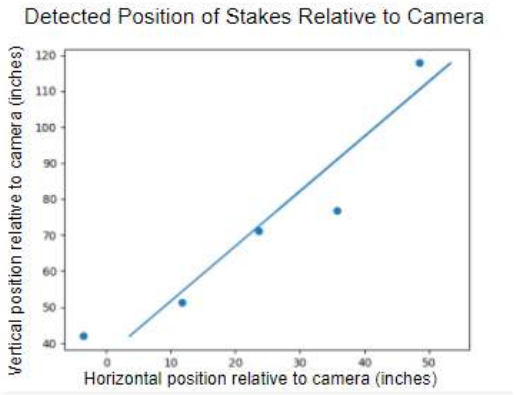

Figure 6: Linear Regression Plot of Stakes

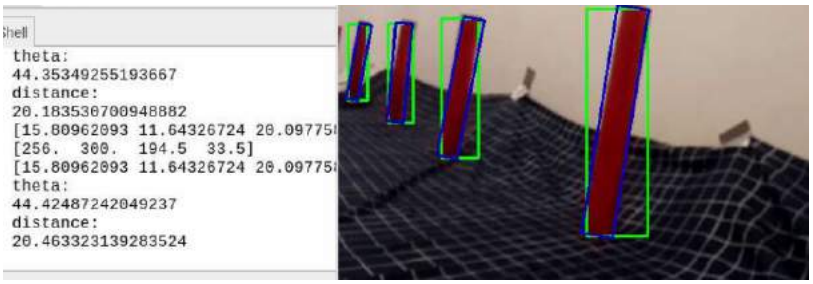

Figure 7: Distance Calculation Outputs of Stakes

Future Work

There are several steps we could take to improve our project in the future. For each subsystem: the controls, computer vision (CV) system and lastly the RC car hardware, there are multiple adjustments we would make to improve our cars performance. All of these improvements would make our car more robust. Once these adjustments are made, we would move to add a grabber to harvest asparagus.

For our current controls system, we are currently using a linear PID controller for navigation. However this is limited in our design as our car can correct if it is next to a stake but is horizontally off by a set distance. Making our PID nonlinear could combat this problem because the nonlinear control system would be able to compensate for the limited mobility of the car, and better adjust over this horizontal distance. Once this adjustment is made, we would add more features to our PID controller. This includes designing a new PID controller to account for speed in the car as well as navigate more than one row. This would make our car able to produce multiple speed settings and be used in more of a real life situation more closely resembling a field.

For our CV system, we would include more filters for stake detection and lastly modify our algorithm to detect asparagus. Presently our CV system is very sensitive to light and performs in ideal lighting conditions. In our future work, our CV team would look further into methods to make our project run in an environment with changing lighting conditions such as a field. Our CV system team would also look into making the image detection algorithm able to detect asparagus directly instead of only detecting solid color, uniform stakes.

For our RC car hardware, we would like to upgrade our car motor to handle more torque. Presently our car motor is low-torque and cannot navigate very uneven terrain. Putting in a new motor that can handle more torque would make our car able to navigate rougher terrains such as those that could be seen on a crop field.

Once these revisions are made to our car, we would switch gears to making our car a full asparagus harvester. This includes developing a robotic arm to identify and successfully grab asparagus without damaging the plant.

References

Chong, Bonnie, et al. “ECE 4760 The Autonomous Driving Car.” *Cornell ECE*, Cornell, people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2011/bcc44_acl84_my259/bcc44_acl84_my259/index.html.

GARotics, European Union’s Seventh Framework Programme, 2012, www.amlight.eu/garotics.php.

Hata, Kenji and Savarese, Silvio. “CS231A Course Notes 1: Camera Models.” *Stanford*, https://web.stanford.edu/class/cs231a/course_notes/01-camera-models.pdf

Khvoynitskaya, Sandra. “3 Types of Autonomous Vehicle Sensors in Self-Driving Cars.” *Itransition*, Itransition, 2 Nov. 2020, www.itransition.com/blog/autonomous-vehicle-sensors#:~:text=The%20majority%20of%20today%27s%20automotive,cameras%2C%20radars%2C%20and%20lidars.

Mahbub, Istiaq. “Distance Measurement.” *MD Istiaq Mahbub*, Wix, istiaqmahbub.wixsite.com/embedded/distance-measurment.

Mjrovai, “Hacking a RC Car With Arduino and Android.” *Instructables*, Autodesk, 28 Sept. 2017, www.instructables.com/Hacking-a-RC-Car-With-Arduino-and-Android/.

Nova. “Line Follower Robot (with PID Controller).” *Arduino Project Hub*, Arduino, 11 Aug. 2020, create.arduino.cc/projecthub/anova9347/line-follower-robot-with-pid-controller-cdedbd.

Satran, Joe. “Most Of Your Asparagus Comes From Abroad These Days. Here's Why.” *HuffPost*, HuffPost, 7 Dec. 2017, www.huffpost.com/entry/asparagus-farms-california_n_7029836.

See & Spray Agricultural Machines - Blue River Technology, Blue River Technology, www.bluerivertechnology.com/.

Zitter, Leah. “Inaho's Tireless Asparagus Picking Robot.” *Food and Farming Technology*, 15 Oct. 2020, www.foodandfarmingtechnology.com/news/yield-monitoring/inahos-tireless-asparagus-picking-robot.html.