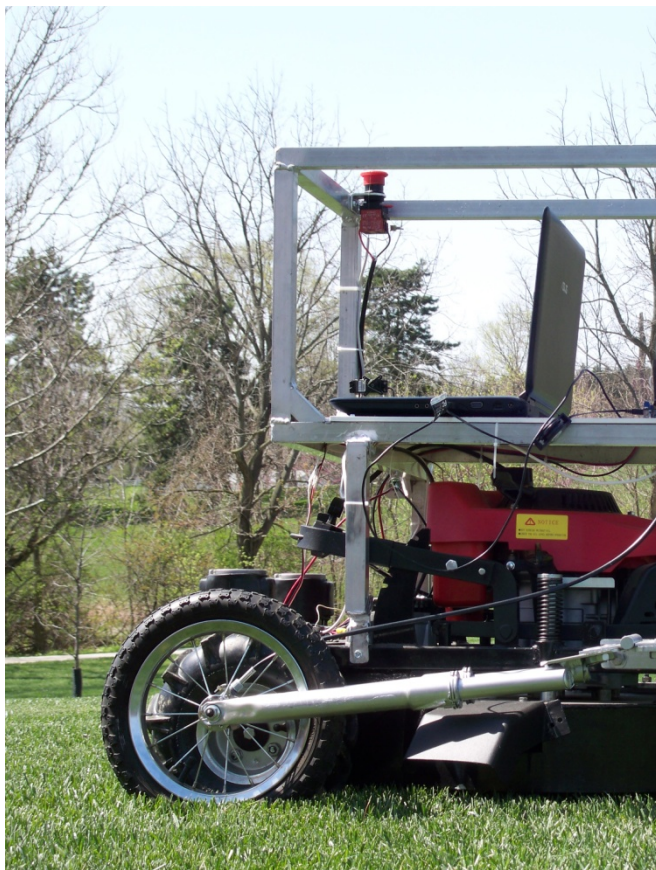


Technical Report

2011

Cedarville University Autonomous Lawnmower – The SnD Robot



Cedarville University

5/12/2011

Table of Contents

INTRODUCTION	2
TEAM ORGANIZATION	3
ROBOT OVERVIEW	4
HARDWARE	5
MOWER.....	5
INTERFACE BOARD	6
COMPASS.....	6
CAMERA	7
MOTOR CONTROLLER	7
TRIMMER	8
WHEEL ENCODER/ODOMETER.....	8
EDGE SENSOR	9
FRONT BUMPER	9
IR DISTANCE SENSORS	9
SOFTWARE	10
COMPUTER NAVIGATION SYSTEM.....	10
NAVIGATION PROGRAM	11
USER INTERFACE	11
FIELD CLASS.....	12
IOBOARD CLASS	13
VISION CLASS	13
SAFETY FEATURES.....	17
CONCLUSIONS	18
SYSTEM SPECIFICATIONS.....	18
COST.....	19

Introduction

This report will detail the design and implementation of our entry for the 8th Annual ION Robotic Lawn Mower Competition. This year's team is comprised of underclassmen engineering majors, and our design is based on last year's senior design effort. The previous system was reliable because it kept the mechanical systems robust but simple. The main objectives for our team were to create a graphical user interface for the mower and to improve the vision and navigation systems. Improving these components has allowed for much better results in mowing tight corners and better navigating throughout the field.

The components used for our mower allow it to be commercially competitive but technologically innovative. The main mower chassis was purchased as a radio controlled mower and then modified for autonomous operation. In order for the mower to be a commercially viable autonomous mower, the cost must be considered. The final price of our mower slightly extended beyond \$2000 which is competitive with currently available automatic mowers. We achieved this cost by utilizing low cost sensors such as a digital compass, optical odometer, infrared distance sensors, sensitive touch sensors and a web camera. A laptop computer acts as the main control center and interfaces with sensors and motor controllers through USB connections.

The *SnD* Robot is capable of trimming around the fence and flowerbed, line following, compass following, detecting and avoiding of obstacles, and being aware of its location on the field and what areas remain to be mowed.

Team Organization

This year's project was an extracurricular competition team for three student team members. Each member was responsible for a key portion of the project. While in charge of a specific portion of the design, each team member aided in the design and implementation of other portions of the project. Because this year's design utilized last year's foundation, previous members are recognized for their contribution to the overall system success.

Current Team Members

Nicker Kerner, *B.S. Mathematics B.S. Computer Science*
Student Head and Navigational Systems
Sarah Norris, *B.S. CpE*
Image Processing and Computer Vision
Danielle Scarpone, *B.S. EE*
Image Processing and Computer Vision

Previous Team Members

Lt. Brandon Brown, *B.S. CpE USAF*
Navigation
Miles McGee, *B.S. CpE*
Sensors and Hardware
Krista Ray, *B.S. CpE*
Computer Vision
Nick Kerner, *Undergraduate Mathematics*
Dead Reckoning/Data Processing

Faculty

Dr. Clinton Kohl
Team Advisor
Mr. Dave Denlinger
Fabrication

Robot Overview

In order to develop the best navigational algorithm, previous methods were researched such as using an accurate differential GPS system, but this method is too expensive to be practical for our application. Dead reckoning has also been used but poses problems because of the cumulative error that occurs with each calculation. The method we decided to implement navigates by using the web camera and other low cost sensors. This design allows the mower to both effectively and accurately navigate the field while avoiding both dynamic and static obstacles.

We implemented this design by using sensors such as a web camera, wheel encoder, digital compass, edge sensor, and front bumper. Using these sensors, we determined an estimate of the mower's location on the field and used this calculation to determine its next move. The mower reads the values of each sensor and then references a predetermined path to follow. After completing the path, the mower returns to any grass that has not been cut that is still within the boundary lines.

Hardware

The *SnD* system uses hardware such as a webcam, bumper, compass, I/O board, motor controller, and a variety of sensors in addition to the conventional lawnmower and trimmer attachment. Figure 1 below depicts the hardware scheme.

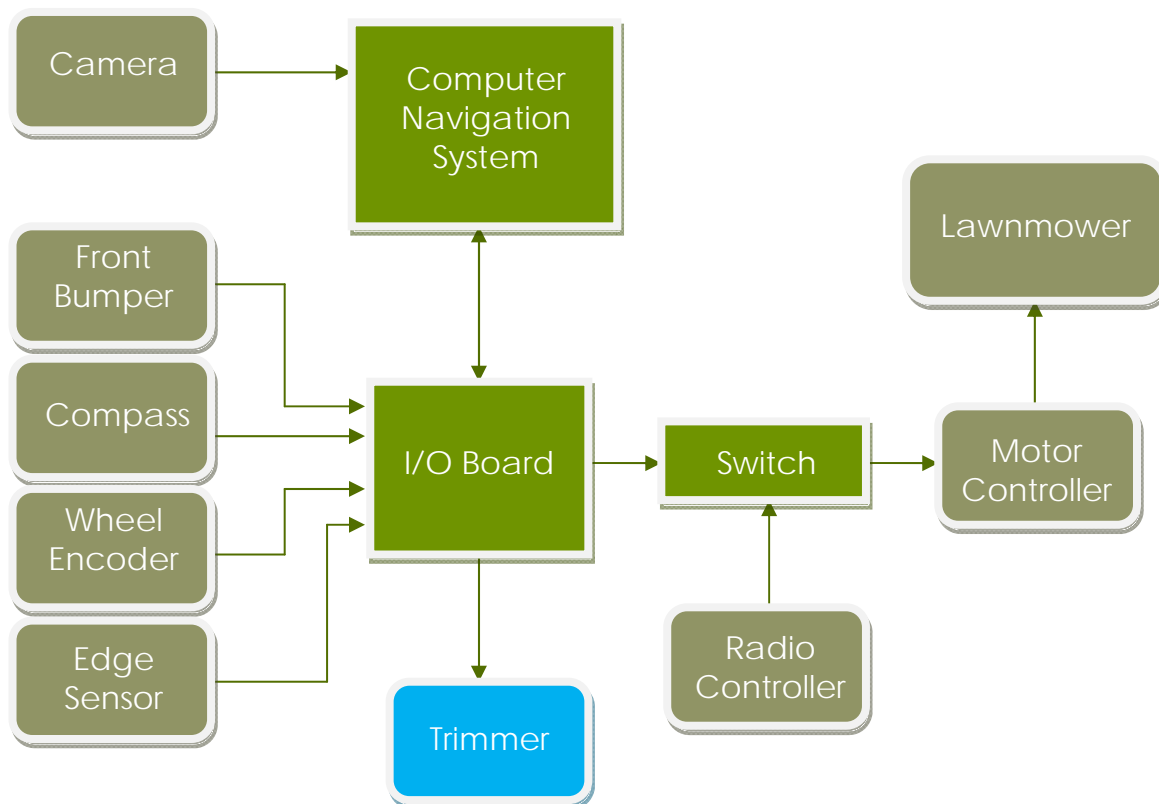


Figure 1. Hardware Block Diagram

Mower

Because our focus was on the navigational system, we started with a prefabricated radio controlled mower. Our options were limited because very few companies manufacture mowers that would be needed for this design. Dr. Kohl suggested Evatech but we found a similar product, the G18 by Hanxing, which offered a more cost effective alternative. In order to support the hardware needed, we designed a frame that mounted to the frame of the mower. Mr. Dave Denlinger manufactured the frame and other mounting components.

Interface Board

An expansion USB input/output board was needed in order to accommodate the variety of sensors with multiple interfaces such as I²C, servo peripherals, digital converters, LEDs, and switches. The LEDs and switches are used in a control and feedback circuit.

We chose the Digilent I/O Explorer board because of its simplistic design, flexibility, and low cost. The board is very practical for most engineering projects because it has a wide variety of connections that allow for many different types of sensors and controllers but costs under \$100.

The Digilent I/O Explorer board also included a C++ API. A solid API is extremely helpful in minimizing coding time and maximizing program reliability. Some of the benefits of using these functions included predefined interfaces between servo and I²C communications allowing us to focus on programming the *SnD*.

Compass

We chose the Devantech CMPS03 after attempting to use the CMPS01 Rev 7 electronic compass module already owned by the university because it failed to communicate with the I/O Explorer board properly. The CMPS03 module was donated by Digilent, who purchased it to perform I²C tests with the I/O Explorer board. The CMPS03 communicates using the I²C interface using either a byte format (0-255) or a word format (0-3599). We use the word format for our application since it gives a reading accurate to 1/10 of a degree. The CMPS03 connects to the I/O Explorer Board through its two wire interface and then communicates the readings to the navigation code on the computer.

Camera

The biggest challenge for choosing a camera was finding a fairly low resolution camera (640x480), with a USB interface and the ability to operate in bright outdoor conditions. These specifications limited us to webcams that tend to automatically boost brightness in order to work well in low-light indoor conditions. Initially, we tried a webcam that Dr. Kohl had from a previous project but discovered that the only image received in bright light was mostly white. We tested two more webcams and chose the one that had better marketing specifications. We tried applying filters through the camera's software in order to create a clearer outdoor image. Unfortunately, this image did not work with the Vision class because the image appeared to be digitally filtered instead of the camera's aperture changing size. As a result, we decided to purchase another camera, the Logitech Webcam Pro 9000. This webcam delivered the correct image resolution, producing a very clear image even in bright sunlight, and including a USB interface.

Motor Controller

We replaced the motor controller that came with the lawn mower because of its limited control configuration, which negatively affected the maneuverability of the mower by not allowing for counter steering in turns, requiring a fairly large turning radius and making manual radio controls unintuitive. We decided to use the Sabertooth 25A 6V-24V Regenerative Motor Driver from Dimension Engineering because it easily accommodated two motors running on a 12V system. The motor controller also has an independent speed and direction operating modes.

The inputs of the motor controller are on a switch and can be manually changed between computer and user radio control. The switch also allows for the *SnD* to be moved by remote control for transportation to and from the field.

Trimmer

In order to mow the grass in the high point areas closest to the fence and flowerbed, we purchased a Porter Cable battery-operated grinder to act as a trimmer. We adapted the grinder by using a 3-blade accessory, and then Dr. Dave Denlinger mounted it to the side of the *SnD*.

Ideally, the trimmer cuts right next to the obstacles, even hitting them gently, to ensure full cut of these areas. The three plastic blades are more resistant to breaking and cut the grass more efficiently than a normal weed trimmer. The trimmer is activated by the main program through the I/O Explorer board and a small relay circuit.

Wheel Encoder/Odometer

The wheel encoder is used to measure distance traveled by the mower. The encoder allows for the use of dead reckoning in navigation when paired with the compass. The encoder is composed of three main parts: the gearing mechanism, cable, and sensor. The gearing mechanism and cable were originally intended for use with a mechanical bicycle odometer. As the wheel turns, the gear spins, causing the cable to turn. The cable is then connected to a custom-made sensor. The sensor contains an infrared sensor which reads black “tick” marks on an internal gear, which is connected to the cable, causing a counter to increment on a PIC 16F676 microcontroller. The PIC keeps a running total of these “ticks” as an 8-bit number and is connected to the I/O Explorer Board via a digital port. When the count reaches 255, it wraps back around to 0. The I/O Board class is responsible for continually reading this value and accounting for the wrap around.

Edge Sensor

The edge sensor is a lever on the side of the robot with a plastic half-circle on one end and a spring on the other end that forces the half-circle outwards by default. The spring end is connected to a potentiometer that allows the I/O Board to read an analog signal to determine the amount of force being placed on the sensor. In this way, the amount of force can be used to adjust the wheels to follow along objects, such as fences and flowerbeds.

Front Bumper

To detect the presence of an obstacle in front of the mower, we used a bumper type sensor. The mechanism consists of the physical bumper and the sensor it actuates. The bumper, when pressed, activates the sensor, which consists of a spring loaded plastic slider and an infrared interrupt sensor. Normally, the plastic slider is in front of the infrared sensor, resulting in a reading of '0', but when the bumper is pressed, the slider moves out of view of the sensor resulting in a reading of '1'. The bumper is extensively used when fence following to ensure a proper turn is accomplished, causing the mower to back up and turn once the sensor is triggered.

IR Distance Sensors

Several infrared distances are used for obstacle detection. An array of four sensors is located on the front of the robot to aid in identification of the dynamic obstacle. These sensors are connected to a PIC microcontroller which communicates serially with the computer via USB. An additional distance sensor is located on the left side of the robot to aid in the following of the flowerbed. The shape of the flowerbed leaves the possibility that our side sensor will not be able to read it when navigating its edges. This additional distance sensor is used to detect the presence of the flowerbed when the side sensor is unable to read it.

Software

The software design for the *SnD* Robot is controlled from the robot's navigation system, which contains the Vision Class, which is for environment sensing capabilities; the field class, which stores what has and has not been mowed; and the communication with the I/O board. Figure 2 depicts the software hierarchy and components of the design.

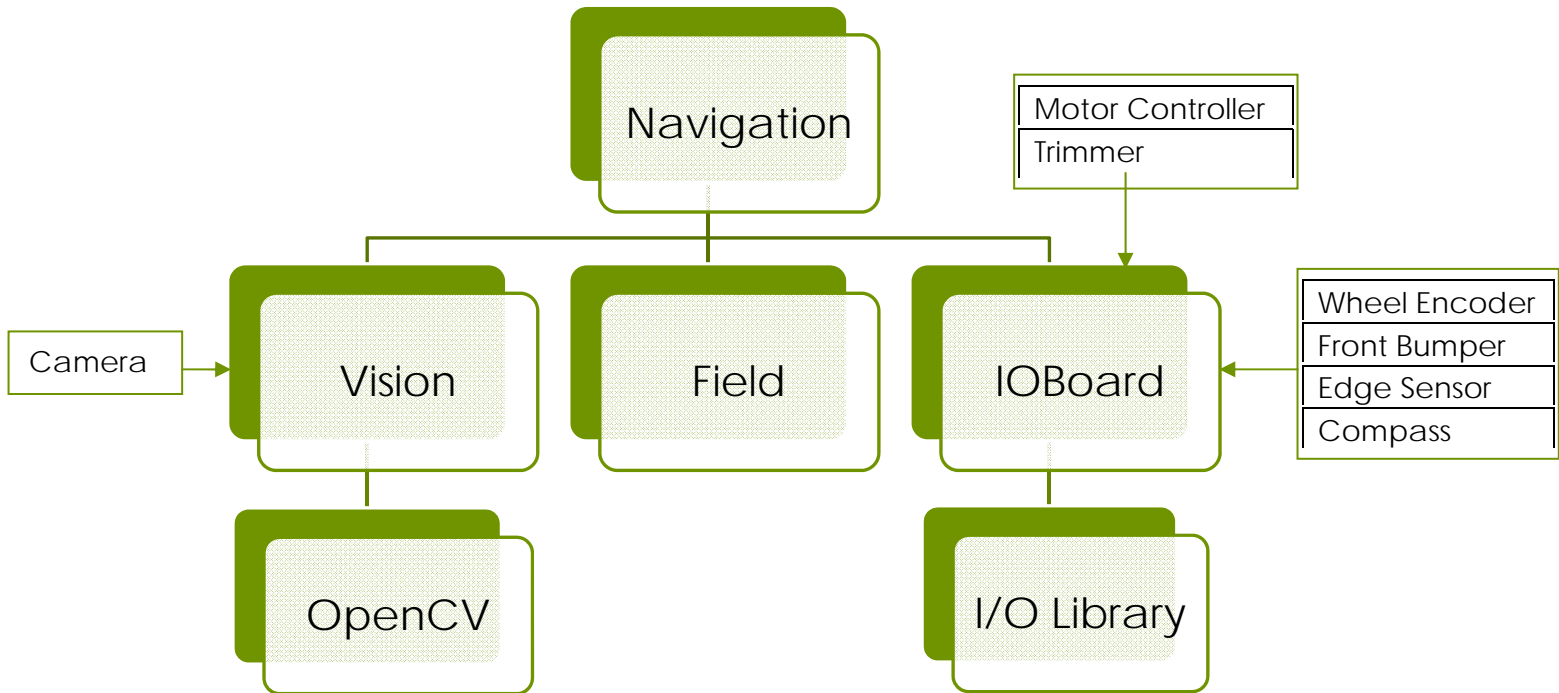


Figure 2. Software Block Diagram

Computer Navigation System

The Computer Navigation System is the central machine for the robot. It is composed of an Asus laptop computer running a custom navigation program on Windows XP Professional. In order to navigate autonomously, several sensors are connected to the computer, all of which are utilized in the navigation program. The web camera and I/O expansion board (the Digilent I/O Explorer) are connected directly through USB.

Navigation Program

The navigation program tracks the robot's position through dead-reckoning odometry (compass and odometer) and makes corrections based on real time data received through the Vision class and touch sensors. Then, based on a pre-determined path algorithm, the program decides the next move for the robot and sends the appropriate commands to the motor controller. Once the path has been completed, the robot will zigzag across the remaining field using the Vision class to stay within the boundaries. The program is written in C++ using Visual Studio 2008 on a Windows XP environment. The navigation program is built using a class hierarchy to improve readability and modularity. There are three major classes that are used – the vision class, the field class, and the IOBoard class.

User Interface

The graphical user interface (GUI) contains the navigation algorithm and uses the sensor subclasses to get digital/analog readings in order to determine location and attempt to follow the pre-determined path. The optimal path was created using the scoring procedure of the competition and is accomplished by a dead-reckoning, state machine paradigm. In each state, the robot prioritizes one of its sensors to navigate. Once an exit condition is satisfied by either the odometer/compass or front/side bumper the robot advances to the next state guided by a different sensor. For example, while navigating the outside border, the Vision class guides the robot, but when the front/side bumper is pressed, the robot ignores vision and uses its side sensor to navigate around the object.

The intended path begins opposite the fence area and follows the white boundary line using the Vision class until either the front or side sensor is hit. The *SnD* will mow the fence area first, since this is the most difficult, and location estimation is most accurate at the beginning. The

robot then uses both its side sensor and its front sensor to “feel” its way around the fence. Once the robot is turned parallel with the far segment of the fence, as verified by the compass, it assumes that the grass closest to the fence is mowed. The robot will finish mowing the fence area by moving in a zigzag pattern, using both the front bumper and the vision to activate the 180 degree turns. When the *SnD* exits the fence area, it will again search for a line to follow and then start zigzagging (mowing the large square portion of the field) until the front/side bumper is activated, indicating that the flowerbed has been hit. Similarly to the fence following algorithm, the robot circumvents the flowerbed with the side sensor, using the compass to determine when the entire perimeter has been covered.

The algorithm then uses the Field class (explained below) to get the location of the area with the most uncut grass. After heading to this area, the mower begins to move about randomly for the remainder of the time, using the vision class to stay within bounds.

Field Class

The purpose of the Field class is to keep track of where the robot currently is located and also where it has been. Using odometry and compass updates, the Field class is able to estimate the robot’s location, which is stored in an array representing the field. The class is able to check if the mower is within bounds and to account for obstacles such as the flowerbed. The inherent error in dead-reckoning is corrected somewhat when the robot reaches certain locations, determined when the front and side sensors are activated. The Field class also gives the location of the largest area of uncut grass, so the robot can determine where to cut when the main algorithm is finished.

IOBoard Class

The IOBoard class was constructed to abstract the native API of the IOBoard and customize the functions for our project. For example, servo control was emphasized to effectively steer the robot in response to inputs. Instead of inputting specific pulse widths for different servos, one function can now control both wheels, taking a more intuitive -100 to 100 integer value for the speed of each wheel. A positive number sends a command to turn the wheel forwards, and a negative number sends a command to turn the wheel backwards.

Vision Class

The OpenCV library is used for the main implementation of the Vision class. The Vision class uses image processing techniques to transform the video that is being streamed from the camera into a usable set of information for the computer's navigation system. The Vision class uses the transformed video to determine the location of the white boundary line and the mower's proximity to the line. The goal is to detect the white boundary line and send location data about the line to the Navigation program. The Navigation program will then decide how to move the mower based upon the location of the boundary line.

The OpenCV library is an Intel project that has been released as open source. We decided to use OpenCV because a team member had prior experience with the library, we could program in C++ allowing for easy integration, and the library focuses on real-time image processing. We decided to use the second version of the library because the newer version worked better with C++ and previous versions of code worked with only minor modifications. We briefly considered using MATLAB's Image Processing Toolbox or writing our own code but quickly realized that MATLAB did not fit our needs as well as OpenCV and due to our lack of familiarity with MATLAB and desire to keep all the code in one language. We did not feel we

had the knowledge or time required to write our own library, as that would require in-depth knowledge of image processing and programming. OpenCV seemed to be the best option for our needs.

The main goal of the Vision class is to detect the location of the boundary line relative to the mower. The vision class takes a frame from the web camera that is mounted on the lawnmower and processes the frame to determine if a white line is visible and, if it is visible, its location within the frame. The top and bottom points of the line are returned to the Navigation class as a percentage of the width of the frame. In the case of a horizontal line the percentage of the height of the frame is returned. This allows the Navigation class to control the robot to line follow or simply stay inside of the white line.

A new instance of the Vision class will initialize the video stream. The video stream can either come from a camera or a video file which was very useful during development and testing. When the navigation class calls the method `getLines()` on the Vision object the current frame from the video stream is grabbed and made into an image. The frame is converted to the hue-saturation-value (HSV) color space (Figure 3) and then the saturation image (Figure 4) is pulled out and a threshold is applied. Before the threshold is applied the saturation image must be inverted because saturation white is represented by black and vice versa. Because we are looking only for the white it was easiest to invert the saturation image and perform the remaining processing on that image.

A binary threshold sets all the darker pixels to black and the lighter ones to white. This highlights the line while removing everything else from the image. Applying the morphological operations erode and dilate to the binary version of the video frame helps reduce noise (Figure 5). The

Hough Transform is the desired function used to find the lines in the image, implemented in C++ with the function `cvHoughLines()`. The Hough Transform requires a sequence (grouping) of points, so contours are found using `cvFindContours()`. A contour is a grouping of points that describe a line or curve found in the image. The contours returned by this function describe the exterior of the white line (Figure 6). The presence of contours indicates the likelihood of the presence of a line and `cvHoughLines()` is then able to be called, using the found contours as input, to apply the Hough Transform (Figure 7).

The first fifty lines returned from the Hough Transform are divided up, based on their slope, into either horizontal or vertical lines. The average of each group is taken and used to determine the location of the line. The line location is returned as a percentage. For a vertical line it represents the horizontal distance from the top and bottom left corner to the line that intersects the top and bottom of the frame. The horizontal line information is the vertical distance of the line down from the top left corner of the frame. This information is passed to the Navigation class to decide how to move the mower.

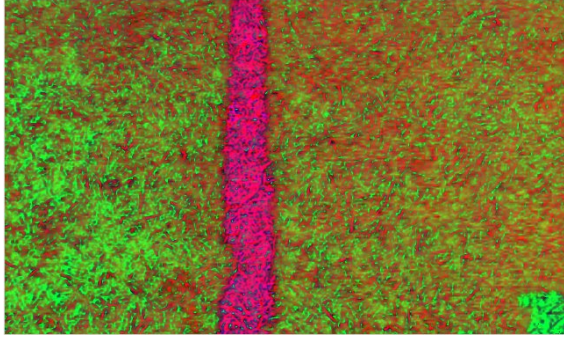


Figure 3: HSV Image

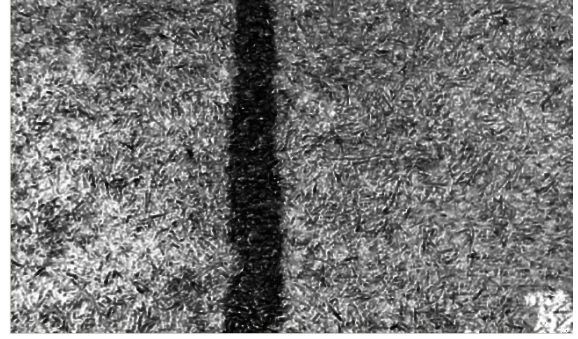


Figure 4: Saturation Image

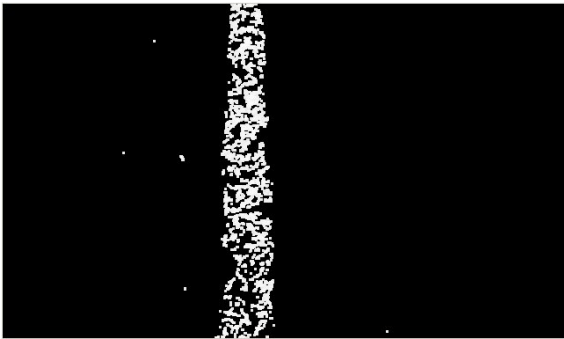


Figure 5: Threshold Image

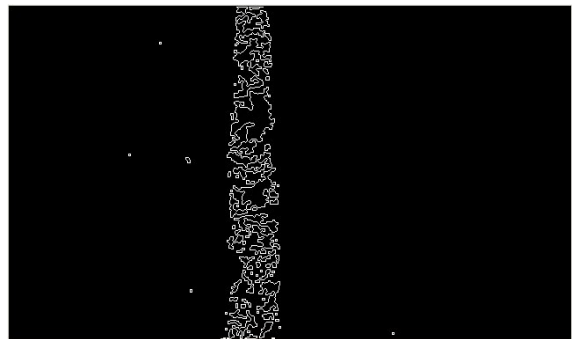


Figure 6: Contours from Threshold Image

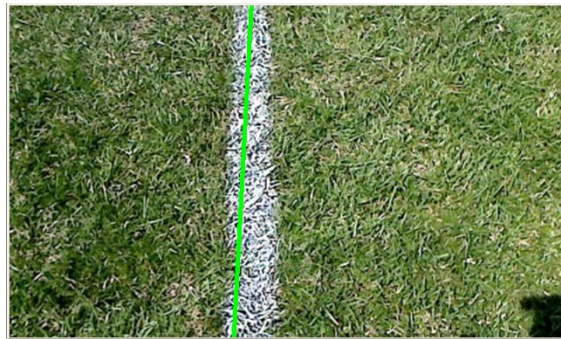


Figure 7: Original Image with Line from Hough Transform

Safety Features

We understand that with any lawnmower, especially one that is not under direct human control, safety is an important priority. Our design incorporates two ways of disabling the robot, both by halting the movement of the robot and by disabling the engine if needed.

Wireless Emergency Stop

An emergency kill switch is located on the radio controller. The radio control for the robot cannot perform any functions while the robot is in autonomous mode except for the emergency stop function. The kill switch on the controller actuates a servo which depresses the manual stop button described below.

Manual Emergency Stop Button

On the top rear portion of the robot is a red emergency stop button 40mm in diameter. This button when pressed cuts all power to the motors, while also shorting out the spark plug on the engine causing it to stop running. This button is equipped with a locking mechanism which ensures that the button remains pressed, to reengage the power, the button must be twisted, which restores power to the motors, and allows for the engine to be restarted.

Conclusions

Through testing and many design changes throughout the course of this project, we believe that we have developed a competitive system that has the ability to perform well. Final system specifications and cost can be found below. We would like to emphasize our total cost for the project, which is much less than teams in the past. We feel that for this to be considered a successful project, it should not only perform well, but also be plausible for actual development and implementation.

System Specifications

Table 1. Design Specifications

Displacement of Engine (cc)	163
Engine Start Mode	Recoil & hand-operated
Bore x Stroke (mm)	68 ×45
Fuel Tank Capacity (l)	2
Lubrication System	Splash
Valve	OHV
Dimensions(L × W × H) (mm)	1016x723.9x876.3
Weight (kg)	86.7
Mowing height (mm)	30-80
Mowing width (mm)	457(18")
Mowing Speed (m/s)	0-1.1
Electric Motor Power (W)	2×200
Radio Control Distance (m)	>100
Alternator Power (kW)	0.9
Wheel size (mm)	76.2×44.5(front) 254×90(rear)
Battery 12v (Ah)	14
Gasoline Usage	Approx ½ gal/acre

Cost

Table 2. Cost of Components

Hanxing Lawnmower G18/Radio Controller & Shipping	\$1,084.00
Dimension Engineering Saber tooth 2 X 25Amp Motor Controller	\$131.86
ASUS P50IJ-X1 Laptop (Dual core T4300 2.1GHz, 4GB RAM,4USB)	\$449.99
Porter Cable Battery Operated Grinder	\$33.73
Weed Trimmer 3 Blade Adapter	\$12.96
Logitech Webcam 9000	\$99.99
Omron A22-01 NC Switch	\$7.21
Digilent I/O Explorer Board	\$100.00
Mounting for Grinder and Trimmer Attachments	\$200 .00
Daventech Digital Compass	\$57.00
IR Sensors	\$55.00
TOTAL	\$2,231.74