CEDARVILLE UNIVERSITY

Delta Bee

15 May 2022

Team Members

Joshua Blackburn (**Captain**)	jblackburn@cedarville.edu
McKenzie Barlow	mbarlow@cedarville.edu
Rachael Judy	rjudy@cedarville.edu
Joshua Mundell	jmundell@cedarville.edu
Drew Murphy	amurphy141@cedarville.edu

Team Advisor

Dr. Clint Kohl kohlc@cedarville.edu

I hereby certify that the design and development of the vehicle Delta Bee, described in this report is significant and equivalent to what might be awarded credit in a senior design course. This is prepared by the student team under my guidance.

Clint Kohl

Dr. Clint Kohl Ph.D.

Conduct of Design Process, Team Identification, and Team Organization

Introduction

For the 28th Annual Intelligent Ground Vehicle Competition at Oakland University, the AutoNav Team from Cedarville University further developed their entry Alpha Bee which placed 2nd in the 2020-2021 competition. This updated robot, now named Delta Bee, will navigate unknown environments as described in the competition guidelines with no user input. Upgrades to the robot this year included implementation of a LiDAR obstacle detection system, high resolution odometry sensors on both wheels, an upgraded GPS receiver, and software enhancements for newer tools such as ROS2. The most notable innovation was the development of LiDAR-based obstacle detection to include mapping drawn on obstacles such as potholes to the LiDAR scans. The team also prioritized safety with the necessary wireless and on-robot stops, upgrading the software safety protocols.

Organization

The 2021-2022 Cedarville AutoNav Team is composed of computer and electrical undergraduate engineers. The chart below in Table 1 shows the areas of concentration for the members pertaining to Delta Bee . The work was primarily split into hardware design and software development. The hardware team members worked on CAD modeling and installing mounts for devices such as the sensors, motors, electronics, and power systems. The software team members collaborated with the hardware team on developing firmware and also developed the behavioral drive system. This included integrating ROS into the system and simulating various runs of the robot from rosbag data. The original robot being designed was for a separate auto navigation task that used many of the same technologies, and thus additional time and members were dedicated to developing the robot for the IGVC course. The chart below provides areas of concentration for the various team members.

Table 1. Team Member Areas of Concentration

Name	Major/Year	Hours of Contribution	Hardware/ Mechanical	Software
McKenzie Barlow	EE/Sr.	100	x	
Joshua Blackburn	EE/Sr.	350	x	x
Rachael Judy	CPE/Sr.	80		x
Joshua Mundell	CPE/Sr.	40		x
Drew Murphy	CPE/Sr.	300		x

Design Assumptions and Design Process

The team approached the design process using the standard waterfall design process, shown in Figure 1, with emphasis on cycling between design, implementation, and testing phases. Delta Bee and its performance in the previous competition were closely reviewed, and this evaluation was used to improve upon the design. Tasks were determined based upon team member specialization and knowledge.

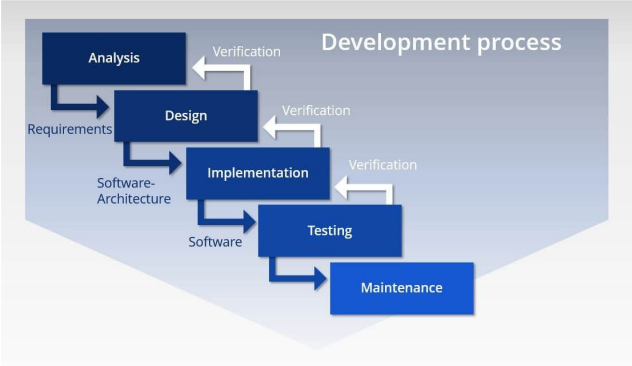

Figure 1. Waterfall Design Process

To start, the team analyzed the IGVC competition requirements and previous results. The team first met to discuss design and consulted with experts in the field. Given the success of the previous robot, it was decided to focus on a similar behavioral algorithm while implementing improved sensor odometry. This was also based on the majority of the team's senior design project which involved autonomous navigation under different constraints. The final decisions were made based upon timeline, cost, safety, and reliability. The hardware team implemented and installed the necessary components and sensors while the software team developed firmware and integrated the odometry into the drive process. Testing was also conducted through imitation of competition conditions on pavement and through running previous sensor odometry from competition through the drive control software. During the testing phase, implementations and designs were reevaluated until requirements were met.

Effective Innovations

Innovative Concepts

Behavioral Navigation

The previous robot from our university effectively used a behavioral navigation model that focused on line following, obstacle avoidance, and GPS navigation. This method allows specialization for the competition constraints instead of overgeneralization. It also allowed the team to specifically select better hardware for specific tasks instead of spending more on general sensors. The master state machine from the previous robot was overhauled to take advantage of better sensor odometry and improve upon obstacle avoidance and navigation.

Innovative Technology

Robot Operating System (ROS)

The programming framework was overhauled from the previous year to better utilize modern image processing techniques employed by OpenCV and ROS2. ROS usage is gaining popularity in industry and research sectors. It treats the sensors including the Intel RealSense

camera, the RPLIDAR module, and the GPS unit as publishers and the master state machine and subcomponents as subscribers able to collect and fuse the data as desired. From this fusion, decisions can be made about the positioning of the robot. As some of the nodes are upgraded to ROS2 for the better toolset, a ROS bridge is used to coordinate the nodes. The setup for Delta Bee is being migrated to the newer versions of the softwares. The ROS package system also advanced the modularity of design.

LiDAR Obstacle Detection and Vision

The robot this year uses the RPLIDAR-A2 for reliable mapping up to twelve meters. This LiDAR sensor provides data real time in a 2D point cloud format. The data closest and most relevant is that within three meters of the device and allows significantly better detection and distance computation of obstacles for the obstacle avoidance. This is an improvement on the VL53L0X Time of Flight sensors used in fusion with the depth camera information in previous years. The LiDAR point cloud was also modified as it was collected to insert potholes as obstacles so as to standardize the obstacle avoidance methods.

Shaft Encoders and Computer Upgrade

The Delta Bee frame was already modularized to make interchanging components fairly simple. The wheel system was upgraded to include precision optical shaft encoders (Figure 2) which give millimeter level resolution on turning. These sensors will provide feedback to ROS and allow for accurate positioning of the robot. When combined with the LiDAR data, accurate estimates can be made for how close the robot is to obstacles. The previous computer system (Nvidia Jetson) was replaced with an HP ProDesk for more computing power and real time sensor processing was upgraded to a Teensy microcontroller for precise management of the shaft encoders and light tower interface.

Figure 2. Shaft Encoders Added to System

Mechanical Design

Overview

The previous version of the robot, “Alpha Bee”, already provided a skeleton platform that met the sizing guidelines of the competition. The robot is 39” by 26” by 70” and can be viewed in Figure 3 below. Modifications were made with consideration for weight distribution and sensor placement. The two 12” front wheels provide turning, with the back caster wheel providing stability. Delta Bee was designed to be lightweight, easy to transport, and house the necessary components. For weather proofing this year, new acrylic shielding was fit around the robot to

provide light rain resistance. The front of the robot has a panel that can be swung up to insert the payload and access the electronic components.

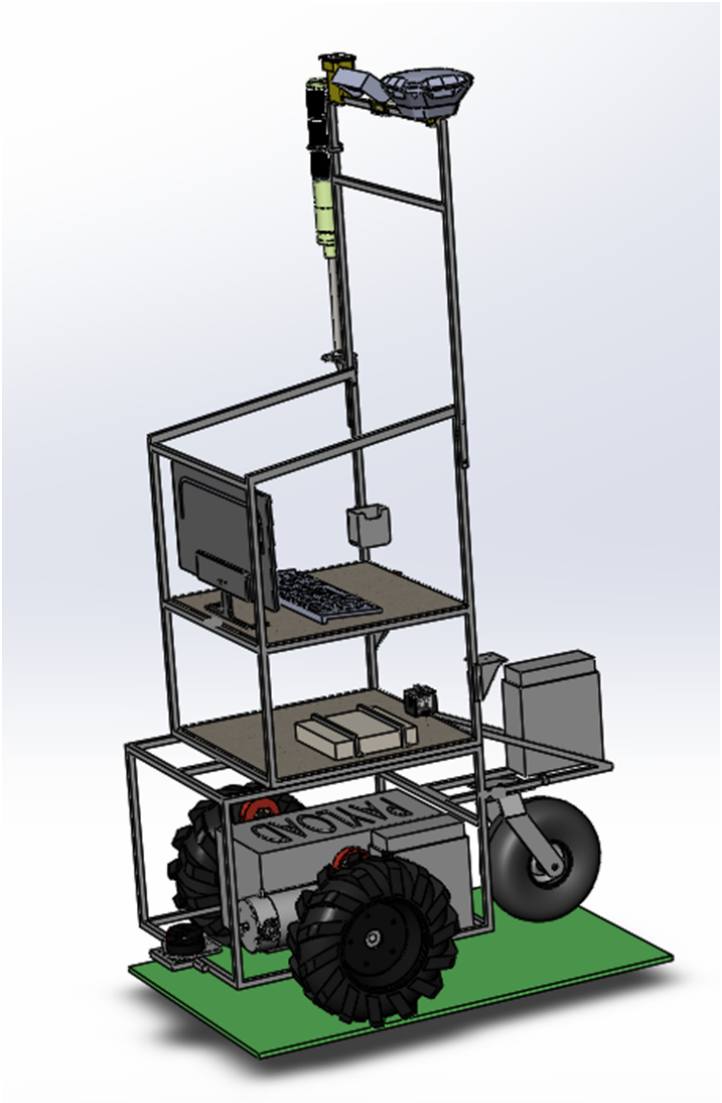

Figure 3. CAD Model of Delta Bee

Decisions on Frame Structure and Housing

The frame structure was designed to be amenable to sensor installation and development on board. In terms of materials, the frame consists of a half inch steel pipe skeleton with three tiers of plastic platforms. The materials were chosen for their rigidity and durability. On the upper platform, the monitor and other peripherals related to the user interface are mounted using 3D printed fixtures. The central platform contains the power regulator, motor controller, the computer running the system, the Teensy microcontroller, and the emergency stop hardware. This section is protected from the elements both by the other tiers and by the weather proofing setup. On the bottom tier, the motor controllers and shaft encoders are mounted, with a space

for the payload in between the devices. This places the weight at the bottom of the robot for stability and to prevent tipping. The Bodine motors generate the necessary torque to maintain the speed within the 1 to 5 mph bounds while carrying the payload.

Besides the main platforms, special mounts were designed to be placed on the front of the robot for the LiDAR and on the top for the camera and the GPS system. The camera was mounted as high on the device as possible to minimize distortion of the image for line following and pothole detection. The GPS was also mounted on the top bar to optimize its signal from the satellites. The LiDAR is mounted at the base of the robot where a special protective bar was installed to protect the LiDAR from contact. This location allows the most obstacles to appear in its line of vision. Additionally, the lights are mounted high, near the camera, for clear signification of autonomous modes and safety. These are all mounted on the steel frame using PLA and PETG 3D printed mounts designed by the team. The wires are able to be interchanged easily between components and most have been wired to be compatible with USB for simplicity of exchange.

Suspension

The robot was designed to move at a slow enough pace that an elaborate suspension system was not necessary. The pneumatic rubber wheels provide adequate suspension when crossing over bumps and navigating over the ramp.

Specific New Hardware

Shaft encoders were added to the system to monitor the distance traveled by each wheel. This modification of the drivetrain was selected to better manage precise turns. In order to install these, encoder mounts needed to be added to the existing structure, so CAD was used to design models and 3D print them for mounting. The final design of these mounts is shown in Figure 4. The design included adding two pulleys to the motors, one which rotated with the wheel and the other which was turned by a belt. The larger upper pulley is monitored by the shaft encoders.

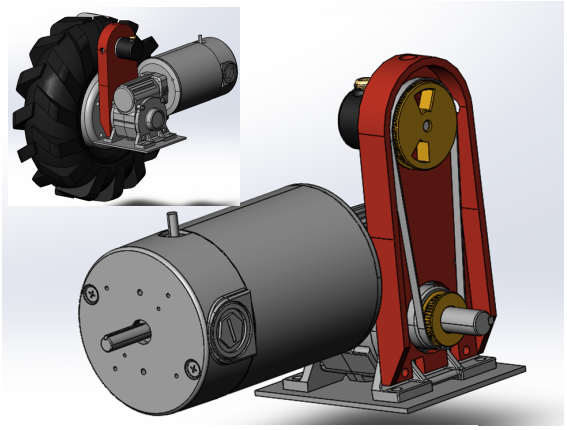

Figure 4. Encoder Mounts

Cost

The vehicle cost estimate can be seen in Table 2 below. The actual cost of the vehicle for the team was reduced from the estimate in the table due to reuse of many components already provided and the sponsorship of the university. This system is considerably cheaper than many previous competition robots and allows replacement of sensors and components for upgrades as desired to improve navigation capabilities.

Table 2. Cost Estimate of Delta Bee if Built from Scratch

Item	Unit Cost	Qty	Cost
HP ProDesk 600 G2	$600.00	1	$600.00
Elmid Reach+ GPS	$900.00	1	$900.00
Electric Gear Motors	$890.00	2	$1,780.00
RPLiDAR	$320.00	1	$320.00
Intel RealSense Camera	$300.00	1	$300.00
Light Tower	$25.00	1	$25.00
Optical Rotary Encoder	$25.00	2	$50.00
RC Radio & Receiver	$85.00	1	$85.00
Monitor	$50.00	1	$50.00
Emergency Switches	$25.00	1	$25.00
Motor Controller	$75.00	1	$75.00
batteries	$80.00	2	$160.00
wheels and Tires	$25.00	3	$75.00
Metal Frame	$250.00	1	$250.00
Teensy Microcontroller	$30.00	1	$30.00
3D Printing & Misc.	$120.00	1	$120.00
Total			**$4,845.00**

Electronic and Power Design

Overview

This year's entry utilized much of the in-place electronics system with some new additions. The system included four categories: the power and drivetrain, the computing system, the sensor array, the output units, and the safety system. The components of the emergency stop system and most of the drivetrain were well developed and kept in in-place while the sensor array, computing system, and motion and control structure were revamped. The focus was to

better compartmentalize the robot functionalities and allow interchange of components and sensors with ease. The technologies used in the project can be seen in Figure 5, and an overall connection diagram can be seen in Figure 6.

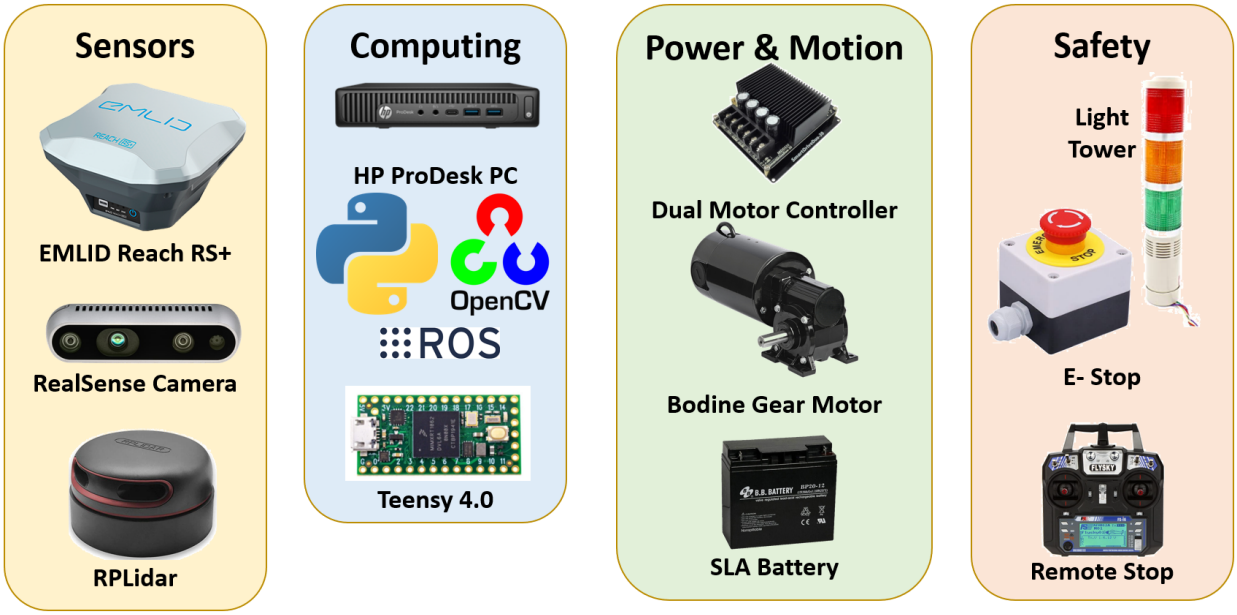

Figure 5. Technologies Used

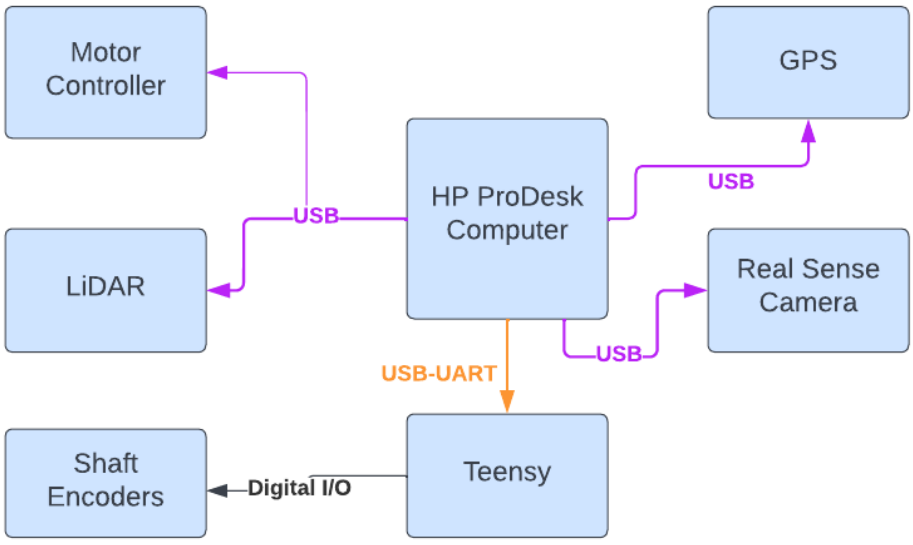

Figure 6. Connection Diagram of Components

Power System

The system was powered with two 12V batteries. Most components such as the sensor array and the peripherals were controlled off of a single 12V battery with the motor controllers requiring 24 volts to operate. The computer power system was completely redesigned to step the 24V battery to the level for the converter for the specific machine. This ensures the computer safeties and control systems never have unstable power. These motors were rated at their limits to run at a quarter horsepower at up to 8.8 amps. These limits are never approached in the competition. The motors were driven by the Cytron Technologies MDDS30 smart motor controller. A Buck converter was used for the regulation of the power to the lower power devices such as the peripherals. The input sensors are controlled through USB ports while the motors and drive system get power directly from the battery. Based on the power consumption of the various components running at a total of about 20A at the varied voltages, the minimum run time per charge while loaded can be estimated to be approximately one hour. Testing has indicated Delta Bee can run for over four hours without the battery being depleted. This is more than enough time to run the AutoNav course multiple times and is a clear underestimate as the motors draw the majority of the power and would not likely draw the specified amount since they are not run constantly at maximum output. It takes approximately six hours to recharge the overall system.

HP ProDesk Computer

The HP ProDesk Computer acts as the primary controller for the robot. It is connected to the sensor array through various USB ports and also communicates with the Teensy microcontroller to manage the shaft encoders. This computer uses an Intel Core i5 processor, has 16GB of RAM, and a 256GB SSD. It runs Ubuntu 20.04 and a mesh of ROS Noetic and Galactic. The code was primarily written in Python.

Motors and Controller

The robot uses the MDDS30 Smart Controller to manage the motors. This uses a simple serial protocol to send speed and direction commands to the Bodine 42A5BEPM-5N electric motors. These are rated at 24V and one-quarter horsepower and can easily handle the power and speed required for the course including navigating up the ramp.

Sensor Suite

Intel RealSense Depth Camera

The RealSense camera is used to capture color images for line detection and line following. The camera provides a stereo solution at up to 10m with ideal capture distance at between .3m and 3m, which fits our mechanical setup. It also provides a 69 by 42 degree field of view at 30 frames per second. Through a ROS node that subscribes to both the RealSense color image and the LiDAR laser scan, it is used for detecting and mapping potholes onto the LiDAR scan. When the camera detects a circle signifying a pothole, it maps this onto the LiDAR scan which

consists of distances and intensities at varied angles. This required development of a custom processing node for the camera that included usage of a HSV filter for shadows.

LiDAR

Delta Bee used the RPLIDAR A2 M8 (Figure 7). This two-dimensional LiDAR has a 360 degree view and turns on brushless motors, capturing the full circle of 8000 points every sweep. It has a 12 meter radius and uses optical communication to prevent electrical connection failure. This LiDAR was used for obstacle detection and avoidance and was connected to the CPU through standard USB 3.0. New nodes and processing techniques were designed for this sensor.

Figure 7. RPLIDAR

GPS

The robot also uses the EMLID Tech RS+ GPS. This is used primarily to navigate in the no-man's-land where the robot cannot line-follow and to get across the ramps. This device comes with a GEOPY library that allows precise determination of distance from a waypoint. The phase angles reported by the sensor could be used to determine orientation and moving position is used to determine heading of the robot. Obstacle avoidance methodologies are still in use during this phase.

Shaft Encoders

The shaft encoders are used to track the wheelbase speeds, maintain straight driving, and manage more precise turns for obstacle avoidance and line following. The feedback from these allowed fine tuning of turn control and also better understanding of speeds while in use.

Safety Systems

The robot included both the required on-vehicle and wireless emergency stops in addition to several other safety features. The wired emergency stop has a large red mushroom push button that immediately cuts power to the motor controller through a direct voltage cutoff system. This will bring the robot to a halt as the motors become generators and feed power back to the batteries. Also connected to the same power lines, the wireless emergency stop is triggered by flipping a switch. This switch is driven off by a servo that is connected to horizontal movement on the throttle. It can also be turned off by hand. This duplication of safety stops ensures the robot will be stopped without cutting power to the sensor array or master navigation to save restart time. These will both work regardless of whether the robot is in RC or autonomous mode. Additionally, the overall power to the robot can be cut with the clearly labeled on/off switch which protects against events such as electrical failure; this however does require a restart overhead.

The red light on top of the robot also functions as a safety light. When the light is solid, this indicates that the robot is being manually driven. While flashing, the robot is in autonomous mode.

In addition to the emergency stop mechanisms, the software is all interconnected. If one node fails, the other nodes will also stop sending messages to the motor controllers so movement will cease.

Software Strategy and Mapping Techniques

Overview

The intelligent navigation software that operates Delta Bee remains onboard the robot via the HP computer. The software is based in the Robot Operating System (ROS) and was custom developed to intake the sensor publications and navigate the course. Although more compact control devices could be chosen, the minimum space requirements allowed us to increase the area and processing power of the device. The software on the CPU provides feedback to verify all systems are operational; when the user presses the start button, the vehicle will begin autonomous operation. Instead of using the ROS navigation stack, which is more robust under SLAM conditions, the team used a behavioral pattern and a master finite state machine that functions as the primary controller for the robot. The overall architecture of the main robot master state machine appears in Figure 8 below. Delta Bee starts by searching for a line and following it until it encounters an obstacle. It then turns from lines until it has an unobstructed view and goes around the obstacle until it finds the line again. This continues until it finds a GPS waypoint where it travels to the ramp and to the next waypoint, continuing obstacle avoidance. It then returns to start. The state transitions are based upon messages published by custom ROS nodes that handle processing of the sensor array.

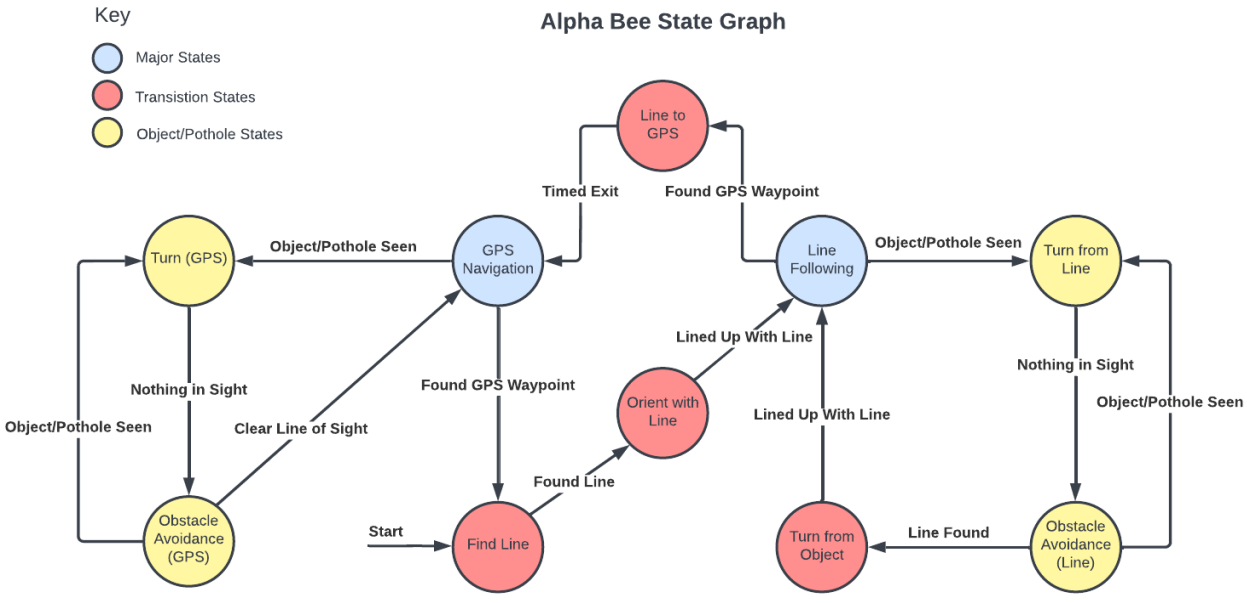

Figure 8. Delta Bee Master State Diagram

Obstacle Detection and Avoidance

Various obstacle detection and avoidance algorithms have been in development for the robot. The transition of grass terrain to parking allows a more specific configuration of the software. The team used an RGB RealSense camera to detect lane edges and manage line following. This method generally follows the line unless obstacles on the path are detected at about one meter away from the robot; in this scenario, the robot attempts to circle around the obstacle and resume line following. The detection of the lines through the Hough Transform provides the exit condition for obstacle avoidance while in line following state. During the stage where Delta Bee needs to resume line following, this is done by using the edge view of the LiDAR to follow the obstacle either back to the line or the GPS to realign with the original heading.

The obstacles themselves are detected using LiDAR. This allows Delta Bee to detect the various barrels and other obstacles in the path and then triggers a new state that takes it around the obstacle. For complex obstacles, such as switchbacks and center islands, the robot continues line following unless the obstacle will be an obstruction to the path, in which case it will circle the obstacle until it either arrives again at a line to follow in the correct direction or the robot is reoriented. Simultaneously while navigating around obstacles, Delta Bee uses the probabilistic hough transform for circles in the RGB camera to watch for potholes. One software specific modification the team made was to insert the RGB camera detected potholes onto the LiDAR scan so the obstacle treatment could be standardized. The distance mapping from camera to LiDAR required a computation of the angle and distortion of the image as it extended away from the robot.

For the state machine, there were several transitional states introduced for obstacle avoidance. These states included pivoting so the obstacle is not in front and then following the obstacle until the robot can realign itself with the lines. The various states included several sensor odometry notifications and a history of values so as to avoid strange state transitions.

Software Strategy and Path Planning

The strategy with this methodology was that it was simpler to implement and allowed course specific programming to be implemented. This way, the software team could avoid strange edge cases that might result in the robot being unable to handle the specifics of the course. In addition, with the knowledge of the constraints and previous simulations, the team was able to give previous sensor odometry to the state machine and observe how it operated in those scenarios. By designing a behavioral strategy instead of a mapping algorithm, this allows for more robust control of the robot during and after the runs. Since the team was aware of the path that would be indicated, it was not necessary to generate a map but rather just have states for the line following and the other stages of the course. Thus path planning was based on the vision and LiDAR information on obstacles and lanes with a reliance on GPS in no-man's-land. Map generation was not necessary for our algorithm; it would only be helpful if the team guided it through the entire course with GPS waypoints or a preliminary SLAM methodology to assist while running.

For path planning, the robot travels with line following at the anticipated 2 mph until it arrives at an obstacle or at a GPS waypoint. While line following, it tracks the line slope and closest

point to the robot. When it arrives at the waypoint section, the strategy, as shown in Figure 9, consists of taking the GPS readings, checking distance to the next waypoint, and correcting for heading error. Arrival at a waypoint takes precedence over the line following so it can travel between points while still continuing obstacle avoidance. The heading is also used to assist the line following stage obstacle avoidance. The state machine manages the incremental objectives as the robot travels the course.

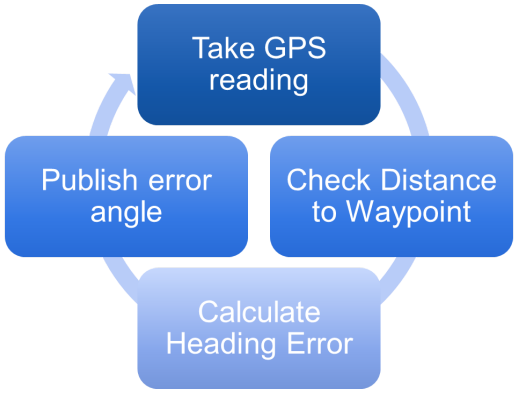

Figure 9. GPS Waypoint Strategy

Failure Modes, Failure Points, and Resolution

Delta Bee was designed to have multiple contingencies in the case of failure and various recovery operations. The team handled several possible failures in both software and hardware. One of the primary means of protection was the modularity of the system. This allows both modifying of nodes specific to a sensor and also interchanging sensors and devices in connection to the system. This encourages ease of debugging and finding failures. The team utilized this modularity in modifying sensor arrays and replacing the power supply for the new computer. Additionally, the failure modes all maintain the safety of the robot, causing voltage cutoff to the motors if anything causes major malfunctions. The emergency stop mechanisms and slow speed ensure the user can easily handle any unexpected failures.

Some of the difficulties in designing this robot included development of the sensor fusion and the shifting of focus as the primary team was originally working with a different robot for a slightly different auto navigation task. This transition was handled smoothly through porting the relevant sensors and code development to Delta Bee midway through the spring. The sensor fusion involved having the software team determine how to modify the LiDAR scan to consider the visual odometry and disregard the apparent obstruction of the robot itself. The main difficulty in the obstacle avoidance was determining the appropriate time to resume the previous behavior of either line following or pursuing a GPS waypoint. This difficulty was partially resolved by implementing GPS waypoints at the end to assist in heading for the line following. This method remains in testing.

Vehicle Failure Modes and Resolutions

Although Delta Bee was designed to be robust and modular, errors could still concur. For the software, the combination of new nodes and previous team developed ROS1 nodes presented a challenge in effectively combining the technologies. The main possible failure would be the scenario where the master state machine became out of sync with the course through bad sensor data or inadequate processing. This could be an issue if the pothole mapping location was distorted for instance and an obstacle was populated too early on the sensor odometry. The

team attempted to mitigate this type of error through using multiple measurements from various sensors before changing states and inserting transitional states. In the case of sensor failure, the nodes publish sufficient information to allow debugging and replacement of the sensor. This is minimized also by careful maintenance of the speed and sensor rates for the robot. Finally, if a node were to shut down unexpectedly, ROS would attempt to restart, and if this were to fail, the robot would be completely stopped so as to not cause a hazard.

Vehicle Failure Points

Mechanically and electrically, the vehicle has a few possible points although measures have been taken to ensure the integrity of the components. Primarily, the concept of modularity allows substitution of parts that fail, and ease of access to the robot encourages repairs in place at any time. Further waterproofing could also be done to defend against conditions worse than light rain or working on a wet and thus highly reflective course. To improve on the weatherproofing this year, the team added plexiglass/acrylic shielding around the robot. The areas of concern include the possibility of loose connections in the wiring that could trigger power outages. This potential has been mitigated with good joints and electrical tape. The risk of tipping has been mitigated by placing the weight on the same axis as the drivetrain. These changes were also made with concern for the possibility of electrical fire or crashing so as to improve safety.

Testing

The robot was tested incrementally throughout development. This included both graphing of data as the robot went through simulated and imitation outdoor courses and as it received sensor odometry from ROS bags of previous robots. Testing was primarily done in the lab and surrounding parking lots that approximated the course. During this testing process, code was constantly being redesigned and integrated.

Vehicle Safety Design

Not only did the vehicle have the emergency stops and failure resolutions discussed in the previous section, it was also developed with safety under consideration. Testing was always done through previous ROS bags and simulation before activating motor control and other devices. Additionally, the sensor array data fed into the navigation system included both real and simulated data to understand how the robot would behave under differing conditions. In the cases where the vehicle exited the course or performed an unrecognized behavior, the emergency stop could be initiated.

Performance Testing to Date

Component Testing

For the various sensors and input/output devices on the robot, all were individually tested under varying conditions. This included outdoor track testing and indoor navigation, simulated sensor array data, and imitated course conditions inside the lab environment. The path planning

and master state machine is still in development but will be tested through the same means as the subcomponents including testing on the ramp and further speed testing.

Simulations

Simulation of the course included both testing of code with invented sensor array data but also preliminary testing of the devices with the algorithm. Based on both previous and live camera feeds within the lab that were used to examine sample potholes and barrels and do line detection, the team was able to certify much of the functionality before even taking the robot to the outdoor course. The same previous course competition feeds were incredibly helpful in working with the vision system software for the robot. Usage of a rosbag from previous feed collection to test line detection can be seen in the example of Figure 10.

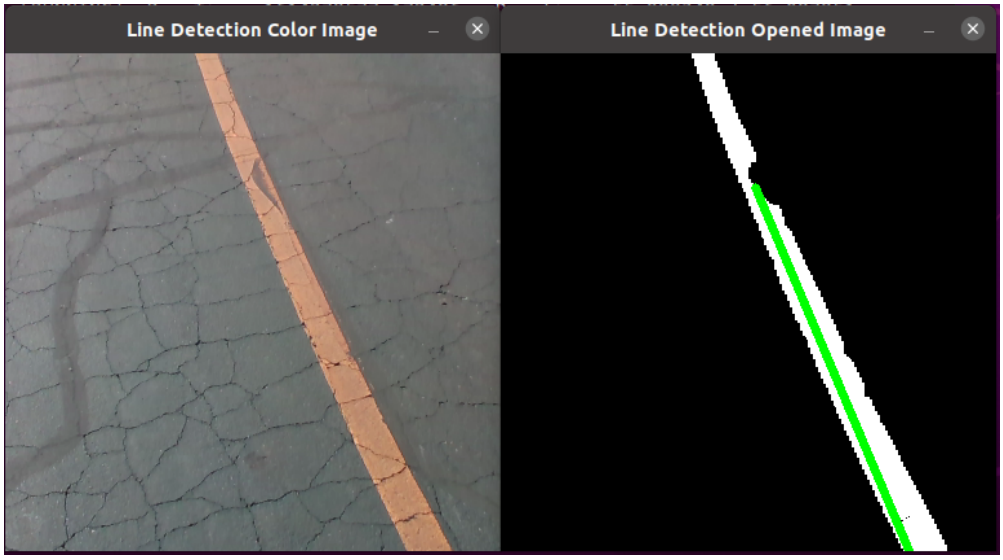

Figure 10. Example of Line Detection in Simulated Images

Initial Performance Assessments

At the time of writing, Delta Bee is in the process of navigation system development. Its power system and sensor array are in good condition for both hardware and software. Remote control operation has been successful in testing the hardware of the robot. Based on previous results, it is safe to conclude the robot can move at about two miles per hour carrying the full load and has the software in place to handle GPS navigation. The vision system simulations have also been successful and seem to handle dynamic obstacle detection well. Additionally, tests have been run with remote control of the motors while doing line and obstacle detection. Overall, Delta Bee's development toward the competition in June is progressing according to the timeline. Attention is now focused on the innovative navigation solution so it can meet the goals outlined by professors and the IGVC objectives of creating a safe, robust AutoNav robot.