Steve

Team Captain:
Jack Little | little018@gannon.edu

Faculty Advisors:
Steven Rowland | rowland005@gannon.edu

"I certify that the design and engineering of Mega-Mind by Gannon University has been significant and equivalent to what might be awarded credit in a senior design course."

Nick Devine | devine003@gannon.edu
Faculty Advisor
Date Submitted: 5/19/2023

1. Introduction

1.1 About us

The 30th Annual Intelligent ground vehicle competition is a return to form for the Gannon IGV Team. After not participating for a year, increased interest in the project brings us to where we are today.

Withing Gannon, IGV is a unique opportunity for engineers to compete in their specialty. It also prides students with their first opportunity to work with engineers outside their Major before Senior Design.

The team has put a lot of effort into Steve since deciding to enter in this year's competition. We learned a lot during development and look forward to seeing Steve and other team's vehicles compete.

1.2 Organization

The Gannon IGV team is composed of undergrads from multiple engineering programs. Below, each student is listed along with what they worked on and time committed.

Table 1: Team Member Organization

Name	Major	Year	Mechanical	Electrical	Software	Hours
Austin Lippert	ME	Sophomore	x			10
Phu Ly	CS	Freshman			x	25
Danial Rutkowski	ME	Sophomore	x		x	10
Aashir Tuladhar	CS	Sophomore			x	35
Andrew Snowdy	EE	Junior		x		20
Jack Little	CE	1st year Grad		x	x	50

1.3 Design Process

A common theme you will see throughout this report is we focused heavily on modularity in this project. This also applies to the method of development we chose. Splitting into groups based on major, each team was given tasks that did not prevent the others from making progress. Utilizing a Rumba with a programming port, each group, including software, was able to start development in early January. Every week each group would get together separately, then all at once every two weeks. This provided us with just enough communication that every meeting was productive, with gaps being filled in using Microsoft Teams.

2. Mechanical Design

2.1 Frame Structure/Housing/Design

Steve is built upon a Jazzy wheelchair base. We found this to be an excellent starting point, as it provided us with a strong, rigid frame, powerful motors, and wheel implementation meant for maneuvering around tight corners. We made several modifications to the vehicle. In the back, we welded on two shelves meant to hold our compute and power electronics box. Each box is designed to be easily removable from the vehicle for repairs, quickly disconnecting from the rest of the vehicle. In the front, we welded together a 1.5-inch square steel tube arch for attaching peripherals such as a lidar and three cameras. At the bottom base of the wheelchair, where the adjustable seat used to attach, we have built a wooden enclosure for the lead acid batteries that is strong enough to hold the weight of the payload.

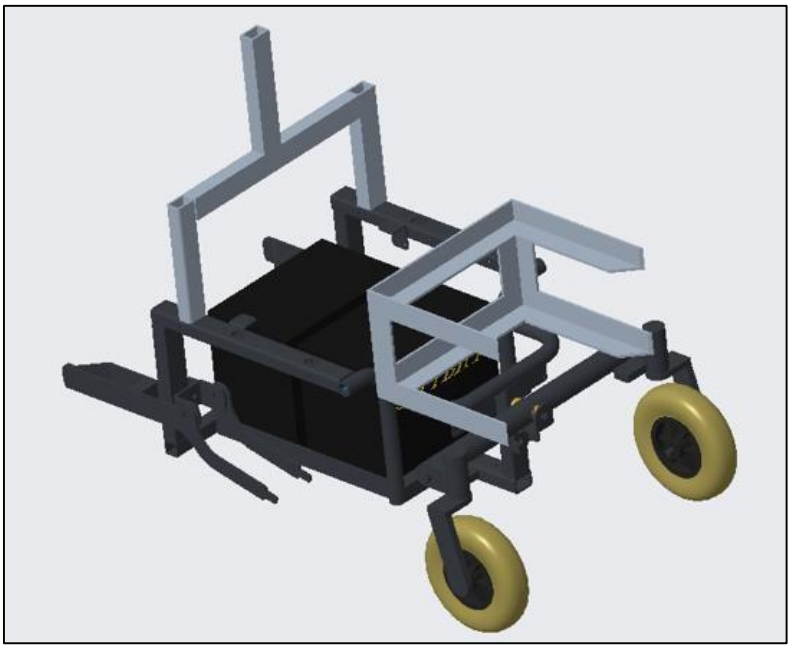


Figure 4. Mega-Mind 3D Frame Model

2.2 Suspension

Our vehicle still has its original suspension, but we found that on a smooth surface such as pavement, the suspension is not of huge importance, especially with our softer wheels.

2.3 Weatherproofing

Our vehicle is what we would call drizzle-proof. While the batteries, electronics, and even cameras are relatively well protected in their enclosures, the nature of how the lidar works prevents us from

covering it, making it a point of concern in anything more than a light drizzle. Determined to compete in rain or shine, we have a replacement for the lidar if needed.

3. Electrical Design

3.1 Power Distribution Systems

Steve has two separate and isolated power sources.

The electronics in the compute box are powered by two High current USBC power banks. Our Anker 537 power bank is capable of supplying up to 60W through its USB C port and powers our Nvidia Jetson NX, along with all the peripherals connected to it. We have a second weaker Getihu power bank that supplies power to the Micro Router we use for Wi-Fi communication and provides power to the motor in the lidar. In our testing, we have never been able to drain either power bank completely, so we estimate the run time to be greater than 6 hours.

There are two 12V 35Ah/20HR Lead acid batteries stored in the bottom of the vehicle. This provides power to our Sabretooth motor controller and, through that, the motor. This power circuit also contains both our physical and remote e-stop. Designing two separate power sources for the robot was intentional, as it allows us to leave our computer running and configured as we utilize the e-stop in testing. We have also never drained the Lead-acid batteries, so the estimated runtime is greater than 6 hours.

3.2 Computer

The heart of Steve is an NVIDIA Jetson Xavier NX. With a 6 Core arm processor, 8GB of Ram, A Nvidia Volta GPU, and WIFI support. It provides us with the competent processing power we need, along with a reasonable power requirement for a portable vehicle. Another reason we picked it was its capability to run full desktop Ubuntu Linux. This was great for development, as it allowed us full IDE support on the vehicle, and students could easily set up their own comparable Linux environment on their laptops for development.

3.3 Lidar

For our Lidar, we are using a Slamtec RPLIDAR A2. A 360 Degree 2D laser scanner with a range up to 12 meters with an angular resolution of 0.9°. The primary use of our lidar is cone avoidance. While cone avoidance can be done with a camera, we thought it would be best to separate responsibilities, leaving the lidar to detect cones. The lidar communicates with the Jetson through serial over USB and is interpreted by a provided C++ library.

3.4 Camera

We are using 3 Arducam 5mp wide-angle USB cameras mounted at different angles on the vehicle for line detection. Looking into line detection prior, we concluded the method we would use did not

need super high-resolution video but a nice wide view. With each camera having a 90° lens, the three provide us a 270° view of the ground in front of us.

3.5 GPS

Our GPS is a ZED-F9P on a Spark Fun GPS-RTK2 breakout board. After calibration, the unit is capable of a 10mm accuracy, so we use the device for both position and orientation. Communicating through USB Serial, we utilized the provided Python Library.

3.6 Status LEDs

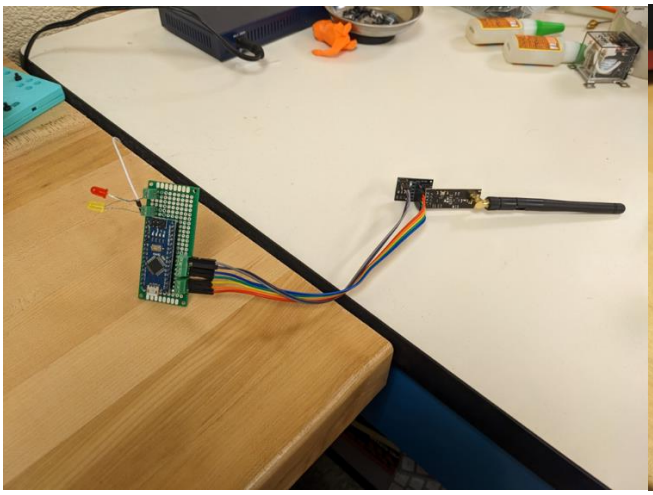
Along the chassis of the vehicle are several strips of WS2812B LEDs. These LEDs all wire back to an Arduino Pro Mini Microcontroller that controls color and brightness. The Arduino is connected to the Jetson through USB Serial. The Jetson sends different commands for the various states of the vehicle during operation.

3.7 Safety Devices and E-Stop

Steve is equipped with both a mechanical e-stop on the vehicle and a remote e-stop triggered by a stand-alone device.

The Mechanical e -stop is connected to a relay that stands between the lead-acid batteries and the Sabretooth motor controller.

The second e-stop is composed of two separate components. A Receiver and a transmitter, both utilizing Arduino Nanos, communicate through NRF adaptor boards. The receiver's outputs are tied to two pins on the Sabretooth that need to be High in order for the controller to function. When providing power to the Sabretooth and the receiver, the receiver will not turn on the motor controller, till it has received a go-ahead from the e-stop transmitter. If the receiver receives a stop command from the transmitter, it will permanently set the output pins low. To reset the receiver, you need to power cycle using the mechanical e-stop on the vehicle.



4. Software Design

4.1 Overview

The software for Steve is composed of several separate C++ and Python programs that communicate together using Linux syscipc. This was done so that each program could be designed to do one thing perfectly. This also allowed us to create a more modular design where we could use the programing language we felt best fit the task.

4.2 Obstacle Detection

There are two main methods of Obstacle Detection for Steve.

4.2.1 Lidar

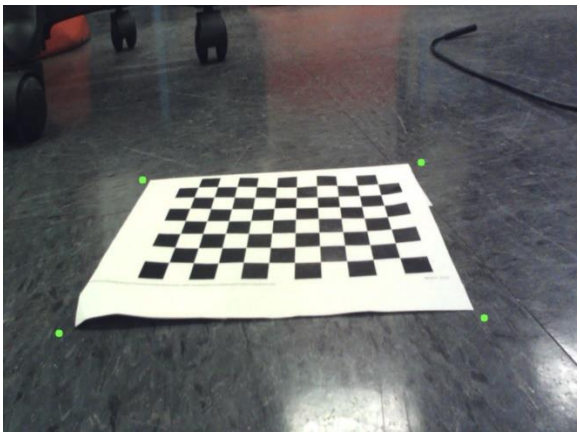
The lidar mounted on the top of the vehicle provides us with distance data of objects 360° around the vehicle. This is our main method of detecting obstacles, such as cones on the track. To simplify the data from the lidar, the program we use for the data collection takes in an argument called sectors. The number passed will divide the 360° view into said sectors. The program will then divide the distance data into associated sectors and return the average distance per sector. This makes the mapping of data onto the Sim significantly easier. We usually operate at 40 sectors of resolution.

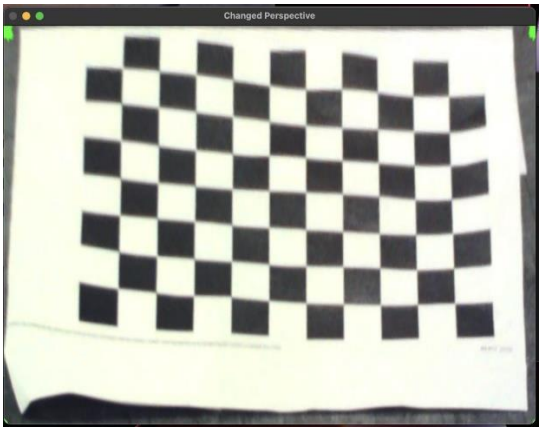
4.2.2 Camera

The three cameras providing a 270° field of view of the ground, allow us to detect lines on the track using the OpenCV python Library.

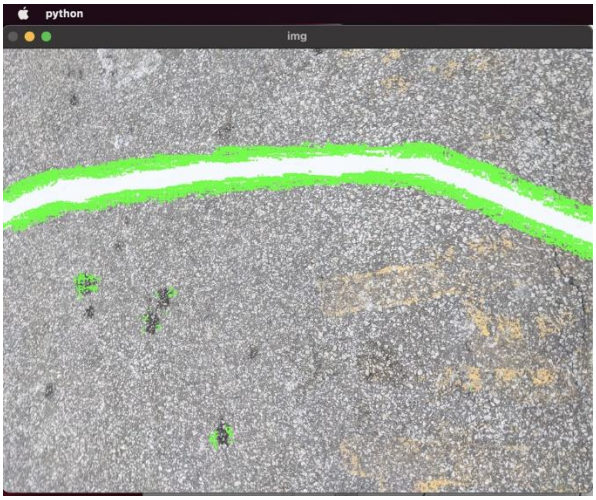
First, you need to fix the distortion for all of the camera lenses. Naturally, there is a barrel or fish-eyed distortion for every frame we capture, leading to inaccurate line detection. Using the chessboard calibration technique, we extract important data such as the camera matrix, region of interest, and the corrected camera matrix. Using these matrices and data points leads to an un-distorted picture making our line detection much more accurate.

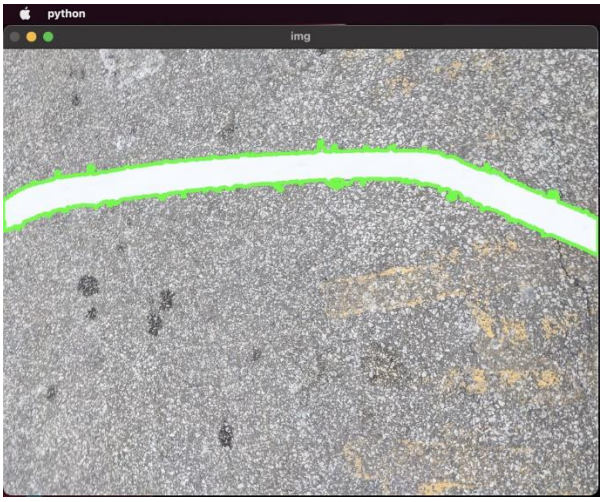
After fixing distortion, we apply an image perspective transform to get a top-down or a bird's eye view from the camera. For this, we locate reference points by plotting four markers to make a square using one of our captured frames. These 4 points provide us with the perspective change needed to give us a top-down view. This top-down view allows us to map later results more accurately.





To detect the white lines, we used contour detection. For each frame we grab, we applied various pre-processing techniques or filtering, such as grayscale, gaussian blur, and thresholding. We also apply contour filtering based on the contour area, which prevents us from detecting anomalies as lines.





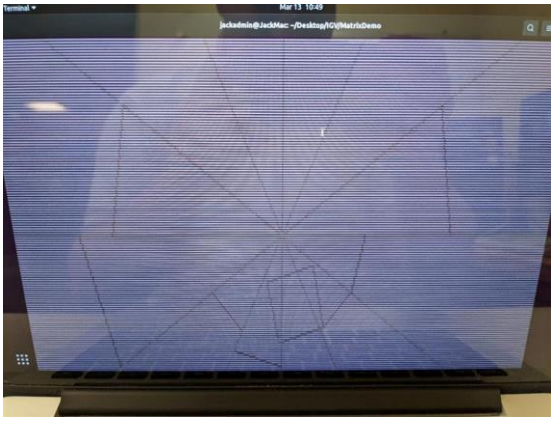
4.3 Path Planning and Avoidance

Our path planning is divided between our Sim and AutoNav programs.

5.3.1 Sim

Sim contains an 801 by 801 2D array that represents a 2d map of the area around the vehicle, with the vehicle being located at 400, 400. Each value in the array represents a centimeter in the real world. Similar to the lidar program, the Sim takes a sectors argument. This allows us to limit the number of paths we simulate, along with the number of directions the vehicle can go. This was useful for testing with smaller datasets and then scaling as needed.

At the start of each cycle, the most recent data in shared memory provided by the cameras and lidar is collected, then obstacles are mapped onto the area around the vehicle. The Simulator then draws in a line at the center of the map that represents the width of the vehicle. Then following the center line and angle of a sector, we move the car one centimeter at a time away from the center till we have detected a collision with an object. This is done once per sector, and at the end, we have a list of the maximum distances possible if we were to follow the center of each sector. This data is then stored in shared memory for the AutoNav program to review before picking the best path for the vehicle.



5.3.2 AutoNav

Our AutoNav program Pulls the sector data from shared memory and implements the GPS library. After identifying the direction it is facing and where it needs to go, reflecting on the distance achievable per defined sector, it pics the best option. AutoNav is the top-level program, combining and making decisions on everything being captured and processed.

5. Failure Identification and Resolution

5.1 Electronical Failure and Resolution

The electronic design of Steve is very modular in a purposeful way so that it is easy to repair. As mentioned in the mechanical design, we have two separate boxes called the compute and electrical. Each box is designed to detach from the rest of the vehicle easily. Making repairs or replacement of individual parts easy. We have replacement parts for everything, and with all communication to peripherals being serial over USB, swapping the Jetson if necessary, would not be a competition-ruining event.

5.2 Mechanical Failure and Resolution

Some components on Steve are 3D printed, and since we plan on bringing a 3D printer to the competition, provides lots of flexibility. That being said, if the vehicle were to be seriously damaged, say, the frame bent or a wheel broke, we would be in a bad situation. Utilizing our e-stops, we have prevented any damage to the vehicle, and that is our plan moving forward.

5.3 Mechanical Failure and Resolution

Gannon returns to the competition this year after a 1-year gap. With most of the leadership graduating after the 2021 competition, our team is mostly freshmen and sophomores. Because of this, software development was started from scratch, and several times prevented the testing and development of other parts of the project. A considerable effort has been made to make sure that we don't start from scratch next year.

6. Simulations

6.1 Sim Program

Although we did not utilize simulation in the design of Steve, the AutoNav program utilizes a simulator that simulates the vehicle moving through a mapped-out version of its surroundings at several predetermined angles.

7. Performance Testing and Assessment to Date

7.1 Progress so far

The first component we finished was the lidar implementation and software, so most of our testing has focused around that feature. We have been able to get the vehicle to maneuver through a tunnel of objects with variable obstacles and curves 25ft long with confidence only using the lidar.

Because of the modularity of our software development, we are confident that once the camera data is ready to be implemented into the Sim, our progress in obstacle avoidance should stay the same.

Line detection is working, but we haven't started plugging the data into the sim yet.

For weather reasons, the GPS is written but not implemented or tested. Once we can create a large testing track outside, this feature will be flushed out.

7.2 Closing Remarks

Steve is designed with modularity in mind, allowing us to debug and diagnose issues more easily as we developed. We look forward to the competition in early June, along with seeing all the other autonomous vehicles developed by the other teams.