University

North Dakota State University

Vehicle

Autonomous Bison

Date Submitted

5/16/18

Team Captain

Andrew Jones
andrew.jones.4@ndsu.edu

Team Members

Rodi Murad
rodi.murad@ndsu.edu

Avijeet Tomer
avijeet.tomer@ndsu.edu

Ruisi Ge
ruisi.ge@ndsu.edu

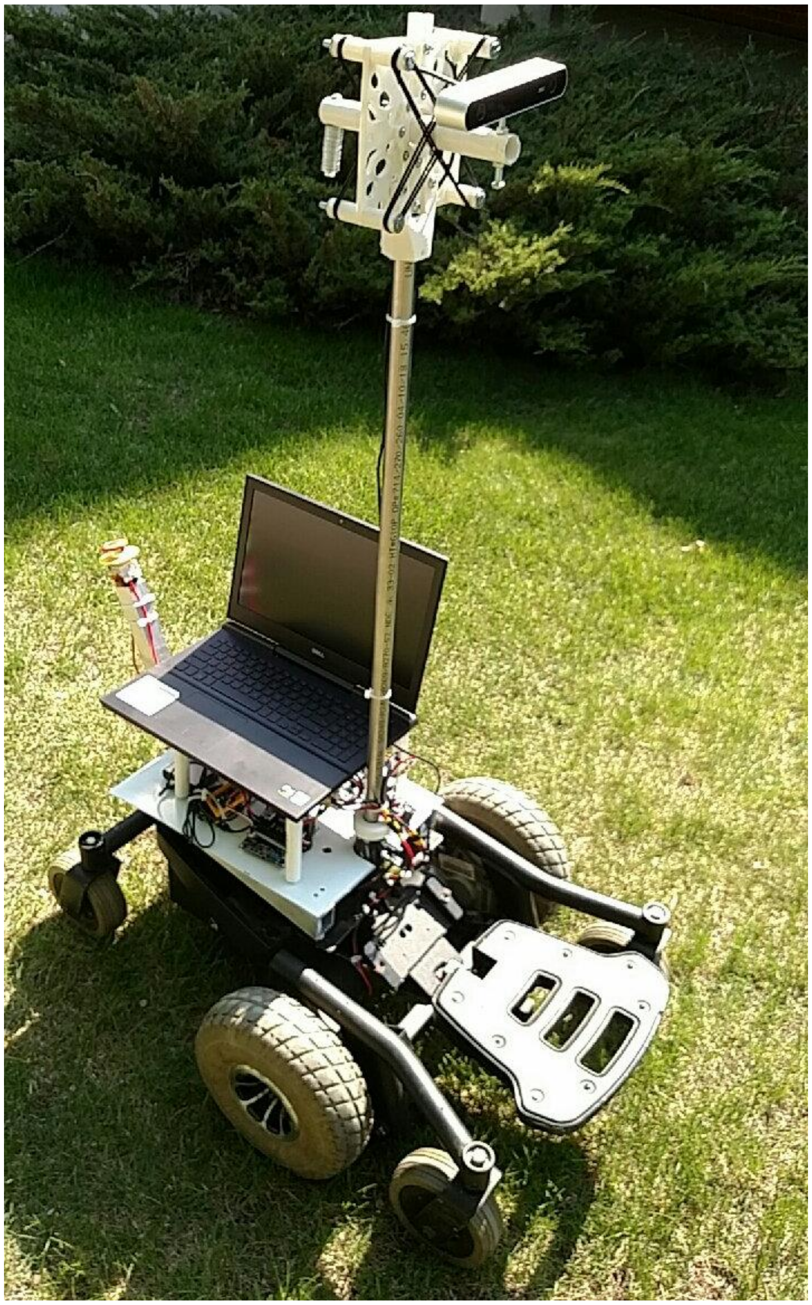
Daniel Anderson
daniel.anderson.3@ndsu.edu

Technical Advisor

Ben Kading

Faculty Advisor

Dr. Jeremy Straub



2. Statement of Integrity

NDSU NORTH DAKOTA STATE UNIVERSITY

5/10/2018

To Whom It May Concern:

I've been asked to provide a statement regarding the originality of the NDS Autonomous Bison's vehicle design. The team has undertaken significant work, from both a design and development perspective to create this vehicle. The design, while based on and recycling a chassis from a wheel chair is unique and designed by the team members. The recycled chassis presented an opportunity for the team as well as a number of challenges that they had to adapt to and was an ecologically friendly way of robot creation (particularly given the large number of these frames that may outlast the associate electronic components). The team integrated this chassis and a variety of hardware and software components and developed new original software as part of their solution. This work, in my opinion, meets or exceeds the requirements of most senior design courses at most colleges and universities. Please don't hesitate to contact me if I can provide any additional information or clarification regarding the foregoing.

Sincerely,

[illegible]

Jeremy Straub, Ph.D., M.B.A.
Assistant Professor

2. Design Process and Team Organization

2.1 Introduction

The Autonomous Bison team participating in this year's Intelligent Ground Vehicle Competition is comprised of students from North Dakota State University (NDSU). This marks the first year of participation in the IGVC for all team members and advisors. The design process alone has been a great learning experience for the team and we are excited to participate in this year's competition.

2.2 Team Organization

Andrew Jones	Team Lead
Rodi Murad	Software
Avijeet Tomer	Software
Ruisi Ge	Electrical Design
Daniel Anderson	Mechanical Design

2.3 Design Assumptions and Design Process

Overall, the design process was initially divided into two subgroups: the software, and the electrical/hardware design. The software side involved using simulations and developing a strategy that could be fine-tuned once specific vehicle characteristics were known. The development process itself has aspects of the Agile methodology, primarily its focus on developing code to satisfy use-cases.

On the physical vehicle side, the emphasis was placed on the stability of the robot during maneuvering and its ability to traverse 15-degree incline ramps. In order to accomplish this, we needed to ensure that the robot had a low center of gravity, satisfactory wheelbase dimensions, and high maneuverability. To this end, the decision to utilize an electronic wheelchair as the base platform became the preferred approach.

3. Innovations

Of all the components and modifications implemented into the vehicle, the ZED camera's capabilities (depth perception, positional tracking, and RGBA color values) allow us to have a single point for generating data that can be used as the primary obstacle detection and odometry. While we aren't the first team to utilize the ZED, we aim to use its various forms of available sensor data to make an autonomous vehicle with less reliance on LIDAR data.

Our software also aims to minimize the usage of the Robot Operating System (ROS) due to the performance drop when using larger data types such as point clouds. Thus, our software design is geared towards designing specific modules that can be used to complete competition tasks and objectives.

In addition, a 3D printed wheel hub that is used to adapt a larger wheel to enable higher speeds for the vehicle. The biggest design challenge was designing the keyway and the main shaft to be able to handle the torque of normal operation. Tolerances were within 1/1000th of an inch, which is challenging for 3D printing applications. Another feature to mitigate stress on the object is a crush zone on the inner area of the hub to increase friction between the shaft and the hub. This 3D printed design was used instead of a metal version due to cost savings and ease of manufacture (requires access to a broaching machine). In addition, the strength of 3D printed components with a significant infill and structural design proved to be capable of handling the application's demands.

4. Mechanical Design

4.1 Overview

The base vehicle used for our competition entry is a donated Jazzy Select 6 power chair. This electric wheelchair model met the vehicle dimension, speed, and maneuverability requirements for the competition. The following sections describe the frame structure, suspension, and weather proofing design details.

4.2 Frame structure and Housing

The cinder block payload will be placed on the wheelchair footrest and secured by a bungee cord. This placement location helps offset the weight of the batteries and achieve a lower center of gravity. Furthermore, rubber shock absorbers mounted on 3D printed parts are utilized to help reduce shock load.

The two 12-volt lead acid batteries are housed in a battery enclosure towards the rear of the vehicle. This enclosure was part of the original wheelchair design and was only slightly modified to accommodate adding component mounting layers above this area. The components mounted on the layer above it (middle layer) include the h-bridges, microcontrollers, E-stop relays and wire routing. The laptop is above this middle layer, thus comprising the top layer of the three-layer subsection for electronics and power distribution.

The Zed camera is situated on a stainless-steel tube mast, roughly 4.5 feet from the ground. This height offset is intended to provide a field of view above the common traffic barrel obstacle height. To reduce camera vibration, the zed is placed on a 3D printed anti-vibration mount, based on common commercial models.

4.3 Suspension

The drive train consists of two front caster wheels, two mid drive wheels, and two rear caster wheels. The frame has Independent left and right rockers for the front and middle wheels. The rear wheels are laterally connected. The front rockers are particularly advantageous for transitioning to and from ramp inclines, as it helps the mid drive wheels maintain a level of traction. No spring devices are used due to relatively low speeds and the existing design of the wheelchair. Furthermore, the lack of springs reduces complexity and modes of failure.

4.4 Weatherproofing

A custom removable enclosure cover was made for the laptop, in order to help guard it from adverse weather conditions (especially rain). The zed camera has a plastic guard on top to help protect it from rain and partially reduce glare from the sun. The battery enclosure has holes in the bottom so that it can drain any water that enters due to rain or puddles. Despite these weather proofing measures, the steel mast may be vulnerable to lightning, thus rendering the vehicle unsafe for use in thunderstorms.

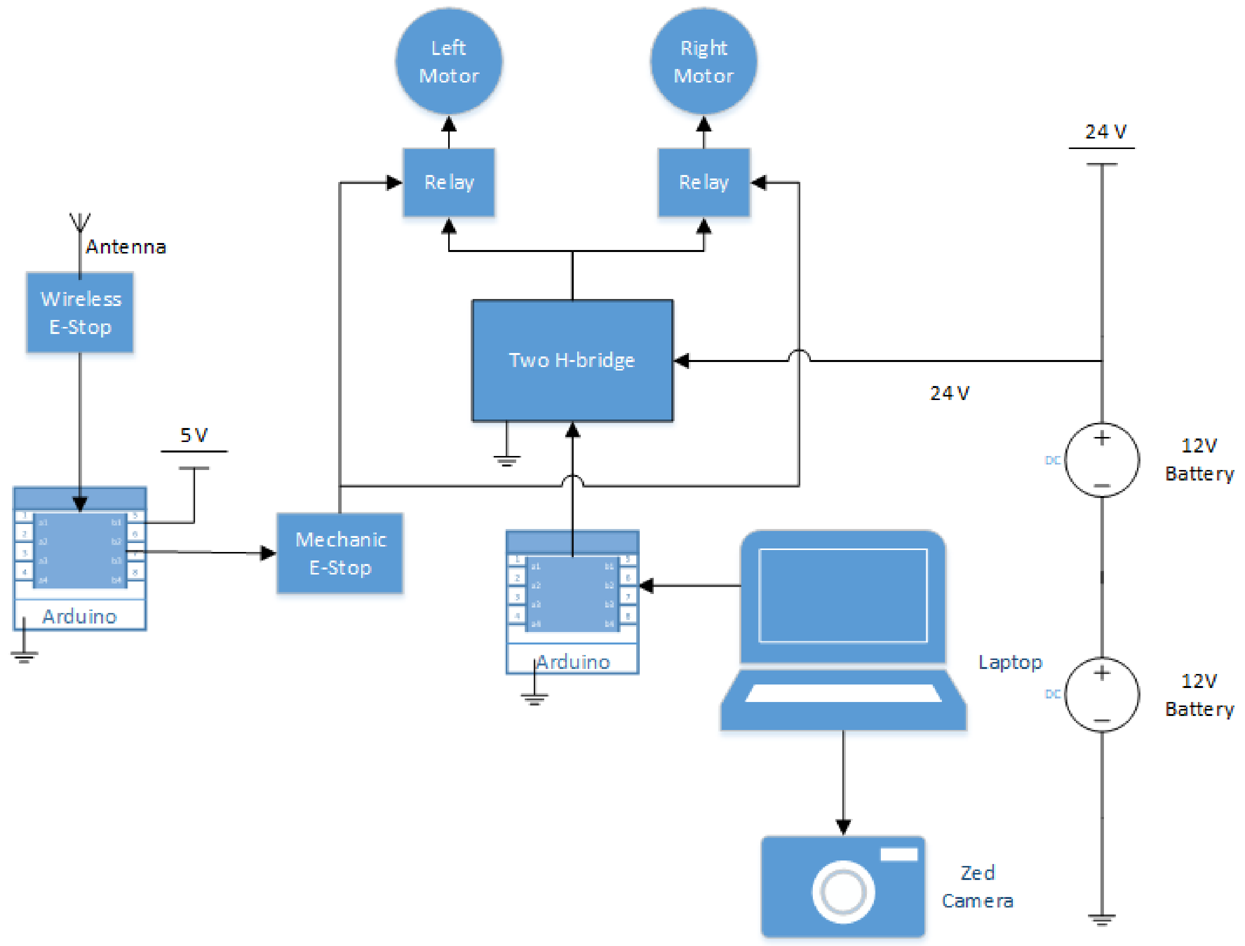
5. Electrical System Design

5.1 Overview

The electrical system design of the vehicle can be categorized into two parts: the power distribution system, and the sensor system. The power distribution system includes the batteries, motors, relays, and power circuit design. The sensor system includes the control laptop, ZED stereo camera, ultrasonic sensors, LIDAR, GPS, and Arduino microcontrollers. The following sections describe the power distribution system and sensor integration in more detail.

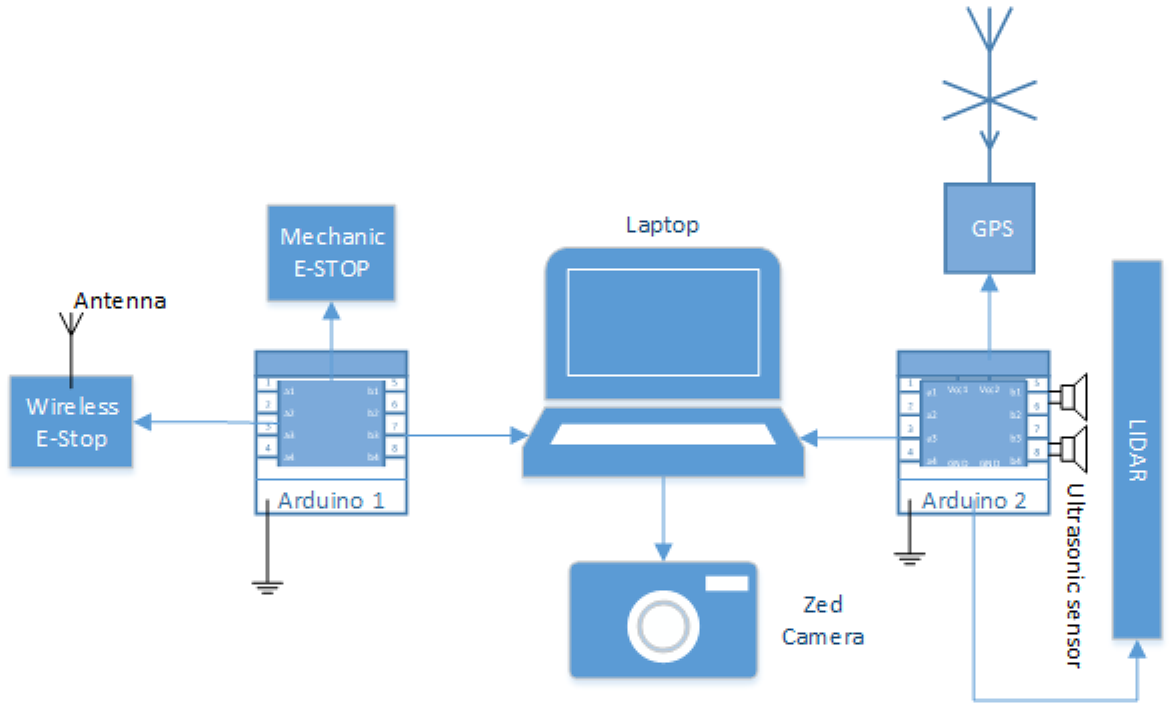
5.2 Power distribution system

The two 12v lead acid batteries are wired in series to accommodate the 24v requirement of the motors. The batteries have a capacity of 20AH, and the vehicle draws between 3A to 5A depending on speed, acceleration and other factors. This data brings the maximum run time of the vehicle to 4-6 hours. Instead of using voltage regulators to step down the 24v from the batteries, the laptop battery provides power to the vehicle’s sensors and microcontrollers via USB. With the laptop’s screen turned to low and peripheral devices plugged in, its max run time is approximately 4 hours.



5.3 Electronics Suite

- **Laptop**: A major constraint for the control laptop is that it needed to have a CUDA capable GPU to be compatible with the ZED SDK. For this, we chose to use a Dell Inspiron 15 7567 with a Nvidia GeForce GTX 1050Ti GPU. In addition, Ubuntu 16.04 is used as the vehicle control operating system for certain library compatibilities.
- **ZED Camera**: A stereo camera that gives us access to RGB images, depth images, 3D point clouds and visual odometry. It has compatibility with outdoor environments and an effective depth sensing range of 20 meters.
- **H-Bridges**: The type of motor controller used for our vehicle is the CyTron MD30C, which accommodates the 24volts that the electronic wheelchair motors use. In addition, these H-Bridges can handle well above the typical current draw of the vehicle.
- **LIDAR**: A Garmin LIDAR-Lite v3 is used for range correction for the ZED and also as a means of close-range obstacle detection.
- **Ultrasonic Sensors**: Two HC - SR04s are used to detect lateral obstacles, which is valuable for avoiding immediate collisions when turning.
- **GPS Module**: Ublox NEO-M8N



5.4 Safety Devices

- **Mechanical E-Stop**: A latching pushbutton is connected to the relays to insure the safety of the system. Once it is pressed, it will open the relays and cut off power to the vehicle's motors. An Arduino monitors whether it has been triggered but does not affect its operation.
- **Wireless E-Stop**: An APC220-43 RF module is used in conjunction with an Arduino to provide a remote turn-off mechanism for the motors. As with the Mechanical E-Stop, the Wireless E-Stop opens the relays in order to cut the power. The Arduino does not receive instructions from the Laptop, but it does notify it in the event of the E-Stop being triggered.

6. Software Strategy and Mapping Techniques

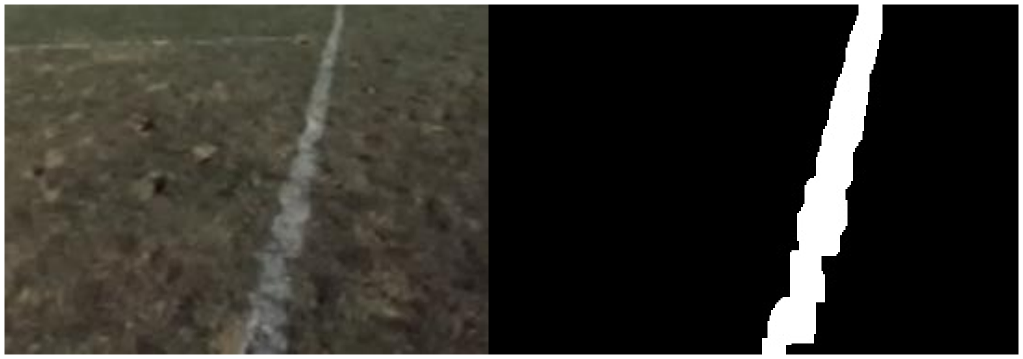
6.1 Overview

A software design decision for our project was to mitigate or entirely not use Robot Operating System (ROS). While it offers a very good environment for rapid prototyping, it has some notable performance issues when dealing with larger data types such as point clouds and images. Thus, our strategy consists of using python 3 (which is supported by a ZED SDK wrapper) in conjunction with OpenCV for image analysis. The following sections describe our approach to obstacle detection, path planning, map generation, and goal selection.

6.2 Obstacle Detection and Avoidance

Obstacles throughout the course will consist of striped construction barrels and various colored drums (e.g., white, grey, black, blue, red, etc.), light posts, street signs, trees and shrubs, and simulated potholes with a solid circular shape.

The ZED camera is utilized as the primary obstacle detection sensor. The laptop uses the ZED SDK to generate a point cloud in each frame such that each pixel has (x,y,z) spatial position values and a RGBA color value. The raw images from the ZED camera map one to one with the point cloud, allowing analysis and filtering in the image to be mapped to the corresponding point cloud position values. This is especially useful for detecting the white boundary lines and simulated potholes, as the images can be color filtered for high luminosity, denoised, and subsequently mapped to the point cloud. This methodology yields a positional boundary and pothole obstacle grid.



The depth images from the ZED camera are used to detect the physical obstacles such as the striped construction barrels. The depth received can be compared to the base case (only ground) and differing values of a certain tolerance are flagged as obstacles. These can then be mapped to the current frame's point cloud to place the objects in the obstacle grid.

We've also attempted to use Haar cascades for purely image-based object detection of construction barrels and other obstacles that can be intercepted throughout the course, however, as of right now the results are not entirely satisfactory and continuation on increasing the effectiveness of detection will be continued until the desired results are met or time no longer permits continuation.

In addition, two ultrasonic sensors and a point based lidar are used as a fail-safe obstacle detection method which update the obstacle grid based on their measured readings. These are especially useful for very close-range detection, where the ZED's minimum detection range may contain blind spots.

6.3 Path Planning

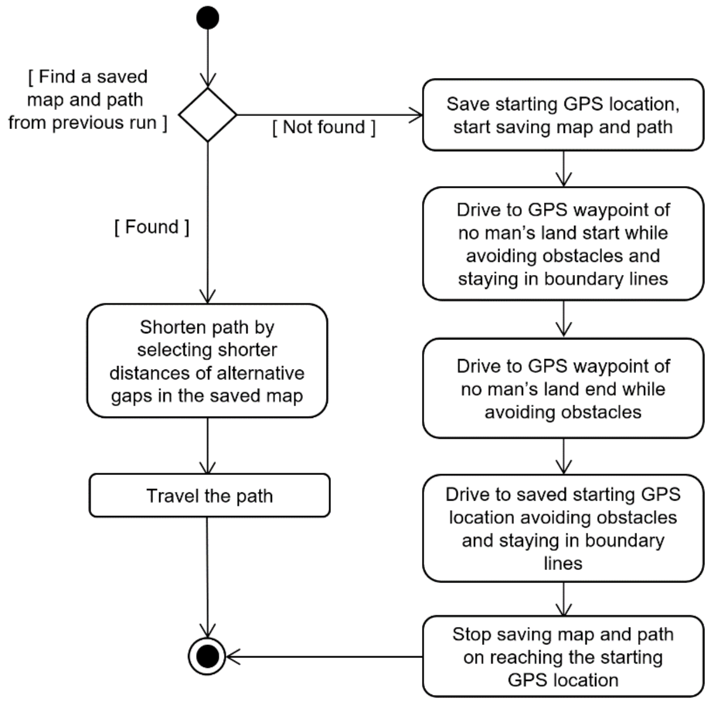
Any generated path must take into account the size of the robot along with its maneuverability. Our vehicle has a zero-turn radius, although a small tolerance is added in case of terrain variance or control irregularities. These variables set which paths are traversable on a given obstacle map. In addition, the ZED's depth images have a specified maximum range of 20 meters, giving our effective path planning on an initial run a limited scope.

6.4 Map Generation

The detected obstacles and boundaries are first placed in a local obstacle grid. A global map is then generated by transforming the local coordinate scheme to a global reference frame. The primary positioning methodology is the use of the ZED SDK's visual odometry. This is coupled with the GPS module's location reading to add data from the local obstacle map to the global map. This method is similar to ROS's navigation stack.

6.5 Goal selection and Path Generation

- **Goal selection:** The robot sets its first goal as the GPS coordinates of the first waypoint before no-man's land, followed by the second GPS waypoint inside no-man's land. Upon finding the destination in no-man's land, the new waypoint is to the exit of no-man's land and then to the final goal waypoint.
- **Pathing for First run:** The robot attempts to find 36 inches wide gaps between obstacles to drive towards, if no gap is found up to 20 meters away, the farthest one before a detected obstacle is selected. On reaching the farthest obstacle in that case, the robot turns towards the second farthest obstacle it previously detected to find the next sufficiently wide gap to drive towards. The camera returns its position in each frame, allowing capturing of the path traveled. In addition, the heading for this step is ultimately toward the given GPS target, to avoid backtracking due to potential obstacle traps.
- **Pathing for Subsequent runs:** With more of the map layout known a-priori, the vehicle can utilize pathing algorithms such as A*. The obstacle map can still be updated during these subsequent runs and also potentially correct any global transform mistakes due to GPS inaccuracies.


[Find a saved map and path from previous run]
[Not found]
Save starting GPS location, start saving map and path
Drive to GPS waypoint of no man's land start while avoiding obstacles and staying in boundary lines
[Found]
Shorten path by selecting shorter distances of alternative gaps in the saved map
Drive to GPS waypoint of no man's land end while avoiding obstacles
Travel the path
Drive to saved starting GPS location avoiding obstacles and staying in boundary lines
Stop saving map and path on reaching the starting GPS location

7. Vehicle Failure Modes, Failure Points and Resolutions

7.1 Software Failure Modes and Resolutions

- **Color filtering failure**: The color values returned by the camera for each pixel are primarily used to detect boundary lines and simulated potholes. The camera SDK allows enabling auto white balance to account for dynamic sunlight conditions while detecting white lines and potholes. Each frame is also scanned for white lines and potholes by OpenCV functions as a color detection failure backup to ensure the same set of pixels is considered sufficiently white for lines and potholes by both the camera's pixel color values and OpenCV functions.

- **Mapping failure**: In case of sufficiently high physical obstacles which are in the center of the track, when the robot selects one of the two paths around it, the map generated contains a set of pixels behind the obstacle that are inaccurately represented as an obstacle due to the camera's field of view. We ignore it because it is not useful for the robot's path towards its goal of a given GPS waypoint. The blue squares represent the robot, black squares are the obstacle, green lines show the limit of the viewport and blue lines show the boundary of the line of sight obstructed by the obstacle. Since the camera cannot see objects behind the obstacle (especially potholes), the area in red is marked as an obstacle too in the generated 2d map. As the robot selects the gap on the left and changes its position from Figure 1 to 2, the map is updated such that the red area reduces. As the robot passes the obstacle in Figure 3, some of that red area is maintained as an obstacle in the generated map because of the limit of camera's front facing view.

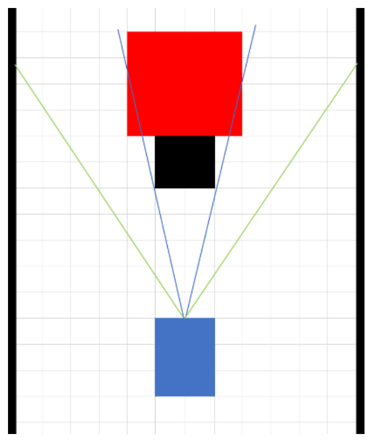
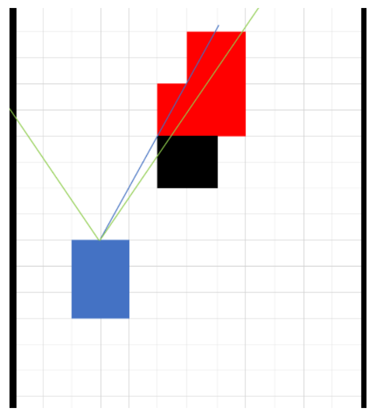
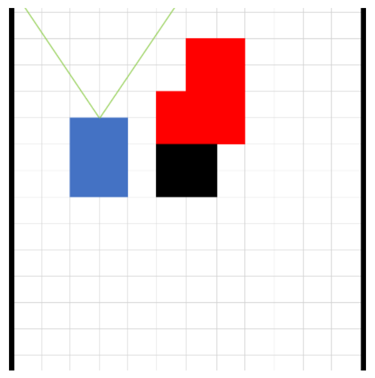
		

Figure 1	Figure 2	Figure 3

7.2 Vehicle Failure Points and Resolutions

- **Short Circuit**: A circuit breaker is in place to cut power to the system in the event of an electrical short. Resolving this issue would involve finding and correcting the short circuit and reenabling the circuit breaker.
- **Wheel Puncture**: In the event that a wheel is punctured, the punctured wheel is removed and replaced with a spare. This failure point is not detected automatically and would only be discovered upon observation of the problem by an observer.
- **Structural fatigue of 3D printed parts**: While the 3D printed parts on the vehicle are made to have a certain level of durability, they may wear over time or break from an impact. Replacement 3D printed components are easy to print, and a cache of spare components are available to swap out any broken components.

8. Simulations employed

The ZED SDK can record and save data in SVO format, which contains not only the video but also 3D point clouds for each frame and the camera's position relative to the starting point for movement tracking. We recorded SVO files in an outdoor environment, which was set up similar to the IGVC course, and ran our program against the recordings to virtually simulate our robot's behavior. The simulation using SVO files was identical to actually using the camera for obstacle detection, map generation and path planning. The spatial location values in the 3D point cloud in every frame of the SVO file allowed us to code for detecting physical obstacles, and the color value along with OpenCV functions to detect the boundary lines. The point cloud also allowed us to generate a map using the expected distance from the camera to the ground in any given frame and compare it to the actual distance. The location information in each frame of the SVO file allowed us to create the path taken by the robot on the generated map.

9. Performance Testing to Date

- **Vehicle Speed Testing:** Vehicle is able to quickly adjust current speed in the 1-5mph range.
- **Vehicle Maneuverability:** Vehicle is able to perform a zero-point turn.
- **Vehicle Ramp Climbing:** Vehicle is able to traverse inclines exceeding 15-degree slope.
- **Boundary Line Detection:** White lines are detected by the software. Adjustments are needed to accommodate differing lighting conditions.
- **Obstacle Detection:** The software is able to detect obstacles such as striped traffic barrels.
- **Wheel Hubs:** Initial testing of the 3d printed wheel hubs shows the components have enough strength to support the vehicle along with additional weight. Long term strength and performance TBD.
- **E-Stop:** The two E-Stop mechanisms have been tested and are functioning properly.

10. Cost Breakdown

Item	Cost
Laptop	$800
Zed Camera	$450
CyTron MD30C H-Bridges	$70
Garmin Lidar-Lite v3	$115
Ublox NEO-M8N GPS	$40
Relays	$20
Batteries	$100
HC-SRO4 Ultrasonic Sensors	$20
Wheels	$90
Wheelchair Frame & Motors	Donated
Total	**$1,705**