Boise State University - Design Report

15 May, 2022

Bender Team contact information:

Camron Collinsworth camroncollinswor@u.boisestate.edu,
Chuck Burnell c.burnell@gmail.com,
Curtis Jeppesen curtisjeppesen@u.boisestate.edu,
David Auner davidauner@boisestate.edu,
Enrique Moya enriquemoya@u.boisestate.edu,
Gigi Brandes giadabrandes@u.boisestate.edu,
Griff Allen griffallen@boisestate.edu,
Keegan Nelson peternelson861@u.boisestate.edu,
Letizia Moro letiziamoro@u.boisestate.edu,
Matthew Oberg matthewoberg@u.boisestate.edu,
Miles Herget milesherget@u.boisestate.edu,
Nardos Ashenafi nardosashenafi@u.boisestate.edu,
Nathan Sundquist nathansundquist@u.boisestate.edu,
Oliver MacDonald olivermacdonald@u.boisestate.edu,
Parker Parrish parkerparrish@u.boisestate.edu,
Trevor Pratt trevorpratt980@u.boisestate.edu,
Wankun Sirichotiyakul WankunSirichotiyakul@u.boisestate.edu,
William McKinney william@u.boisestate.edu

I hereby certify that the design and development of the vehicle Bender, 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.

Dr. Steve Swanson
College of Innovation and Design
Email: steveswanson@boisestate.edu

1 Introduction

The Boise State University IGVC Team will be competing for the third year with an improved vehicle: Bender. Design improvements for the 2022 competition began in the summer of 2021, with the aim to improve the state estimation, and the robustness of obstacle and lane detection. The entire physical design of Bender was also re-imagined. This includes a full reconstruction of the frame and chassis of the robot.

1.1 Team Organization

This year's team consisted of 14 members. The team leader was responsible for managing the team's procurement of materials in tandem with the faculty mentor and ensuring that design goals were met. The mechanical team was responsible for wheel assembly, chassis design, and the electronics housing and mounting. The electrical team handled all electrical component integration. The software team was responsible for sensor integration and creating navigation protocols.

Team Member	Major	ME	EE	CS	Hours
Nathan Sundquist	Physics (BS)	X			380
Gigi Brandes	Mechanical Engineering (BS)	X			
William McKinney	Mechanical Engineering (BS)	X			80
Miles Herget	Interdisciplinary Studies (BS)	X		X	120
Oliver MacDonald	Engineering (BS)		X		
Camron Collinsworth	Computer Science (BS)			X	80
Matthew Oberg	Computer Science (BS)			X	100
Nardos Ashenafi	Electrical/Mechanical Engineering (PhD)		X		50

1.2 Design Assumptions and Design process

Our design process consisted of a five step process:

1. Analyze problem: Define the problem that needs to be solved
2. Prioritize: Determine which problems are most critical and which problems must be done first
3. Research solutions: Research different approaches, generate ideas, and select the most promising
4. Implement solution: Implement the solution in code

5. Test solution: Test the solution, first in simulations and then in the real world. Collect data and repeat

2 Effective Innovations

2.1 Electrical Innovations

For this latest version of Bender, all of the electronics had to be redesigned and transferred to fit in the new chassis. Aside from the obvious changes made to reroute the electrical components, there was an increase in quality of connections and wires to improve the robustness of the design and the power system was split into a 40 volt and a 20 volt system. The split in the power system allowed for a dedicated 40 volt system to run our new Odrive motor controllers and a dedicated 20 volt system for the rest of our components. This avoided the possibility of damaging the more fragile electronics while giving the vehicle a longer run time and increasing the power of the hub motors. Considerations also had to be made to accommodate the new lighting and safety changes that were introduced in the latest design.

2.2 Software Innovations

One of the largest software innovations that was implemented in the newest version of Bender is the changing of navigation systems. In the previous design, navigation was done by a more custom design, but now, the majority of navigation is handled by highly-customizable libraries that work natively within ROS. Since these libraries are quite well tested, this has pushed a greater emphasis on gathering precise and reliable data to feed to the software. This has cascading effects on the innovation processes that the team has worked on. For example, image processing has had a great deal of thought put into it for this year, same with the algorithm to generate our costmaps.

Aside from specifically navigation algorithms, certain sensors have had their design implementations changed from the ground up. We now have the capability to gather position information solely from inertial measurements instead of relying only on wheel odometry, which allows for more accurate measurements to be taken under varying conditions. Since the inertial data is gathered by the same device that handles GPS data, the nodes that handle GPS have been redesigned as well. This allows for greater accuracy when navigating between GPS coordinates.

2.3 Mechanical Innovations

This year, Bender can accommodate much more adjustment for fine-tuning suspension/ ride height. This adjustment comes from using custom sliding mounts on each corner to mount the suspension springs. Additionally, this year instead of pulleys being used to turn the hub motors on Bender, the team chose chains and sprockets. This reduced the amount of slippage that came from using a belt and pulley turning system.

3 Mechanical Design

3.1 Overview

The decisions made by the mechanical design team centered around meeting three main goals. First, the wheel assemblies needed to achieve point rotation for each wheel as opposed to sweeping turns in order to simplify the steering algorithms. Second, the comprehensive design was built with modular repair and upgrades in mind. Finally, the modular design had to allow for readily accessible electronics throughout the stages of prototyping.

3.2 Decision on Frame Structure, Housing, Structure Design

Bender's mechanical components are broken down into two basic sections: frame and wheel assemblies.The frame was created to be lightweight while still providing essential structural integrity, while the wheel assemblies are durable and modular. The frame is constructed from 6061 alloy, two inch by four inch extruded aluminum rectangular tube section with a 0.125 inch thick side wall. A .125 inch steel sheet is bolted to the bottom of the frame on which the electronics are mounted. Above this section is a second steel sheet which covers the components in the body and is fixed with a ratchet strap for securing loads for transport. Tungsten Inert Gas (TIG) welding holds the base frame components together. Custom brackets and braces hold the vertical camera rack to the body. The vertical structure was designed with 80/20 aluminum extrusion to maximize potential mounting space for the sensory systems. Sliders attached to the vertical assembly can be adjusted for optimal camera angles. Mounted at the top of the body is the LIDAR sensor in the front and in the rear the e-stop and LED signal.

3.3 Suspension

Bender's suspension is accomplished by using several sets of Redcat RC shock absorbers. The decision to use these commercial off the shelf parts was to ensure reliability (cutting down on troubleshooting bugs from creating our own suspension devices), low cost, and time savings. Bender has four of these shock absorbers on each corner, and each shock absorber is tightened to its maximum, providing the highest possible stiffness. Each shock absorber is mounted with custom hardware, including an adjustable sliding mount (key innovation) and hex nuts that were turned to a smaller radius (acting as ball joint mount that allows the springs to rotate and adjust as they compress).

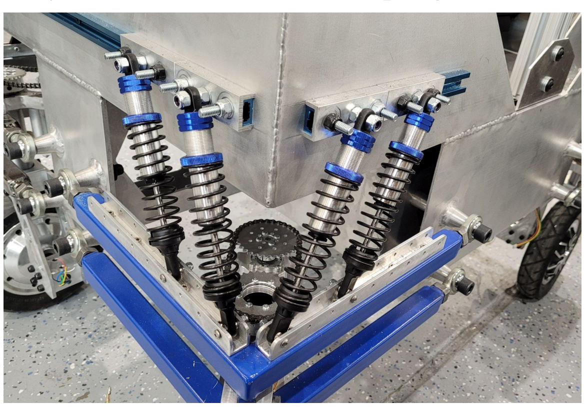


Figure 1: Bender’s suspension system

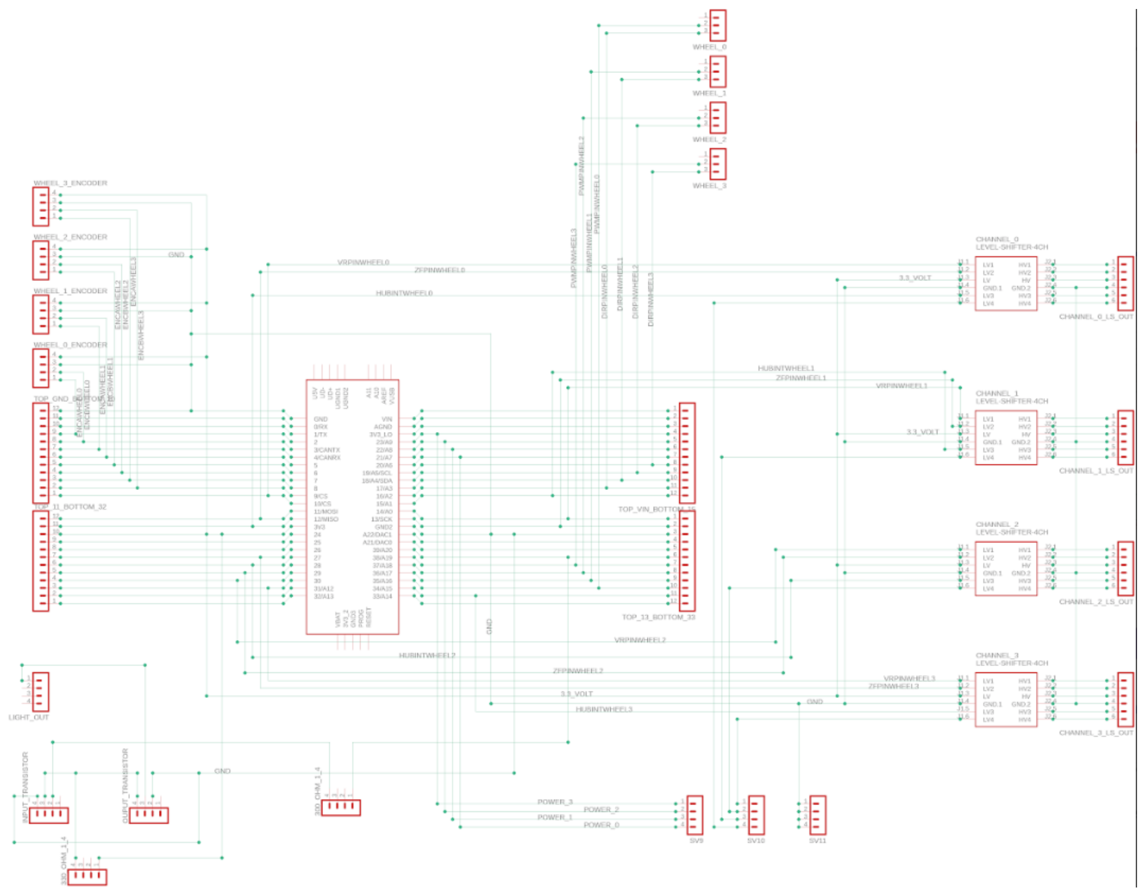
3.4 Weather Proofing

The mechanical team was in charge of creating weatherproof housings for Bender’s electronic components while also allowing ventilation to prevent overheating. The team decided on drilling holes in the side of the chassis to allow airflow, while covering and sealing the top and bottom of the electronics housing. During wet conditions plastic coverings can be mounted to prevent water damage to electronic components such as the camera, Raspberry Pi, and laptop. In this case, overheating will be prevented by utilizing the metal chassis and mounts as a heat sink.

4 Electronic and Power Design

4.1 Overview

The design of the electrical system was meant to run three independent systems which communicate between each other. First, a laptop rides in the chassis of the vehicle and deals with the heavy computational functions which control the vehicle. Secondly, a Raspberry Pi 4 is used to run the motor controllers and certain other lower level software. Finally, hub wheel motor controllers are on their own system which led to an onboard 20 volt and 40 volt power system. Each electrical task provides unique design challenges as the Raspberry Pi and the motors have vastly different functions. Another factor would be the electronics that are associated with the onboard laptop. In our design, a laptop is placed on Bender. This laptop is responsible for running the majority of the software and sensor functions on Bender. The laptop and its connections are mostly separate from the electrical and power distribution design because they are essentially a stand-alone system that is necessary for operation.



(PCB Schematic)

4.2 Power distribution system

The power distribution system is comprised of two independent systems powered by standard drill batteries. 20 volt batteries are used to power the main components which control the vehicle while 40 volt batteries are used to power the Odrive motor controllers which control the hub motors.. From our testing, we were able to achieve approximately one hour of runtime under average use conditions. The benefit of using simple drill batteries is that they are somewhat cheap, easily available, and are robust and built to handle adverse environments. Batteries can be swapped on the fly to quickly recharge the running time of the rover. When charging is necessary, each battery can be recharged in under an hour using the proper charging stations.

4.3 Electronics suite description

The brains of Bender are split between a Raspberry Pi 4 located on the main chassis of the rover connected to a Teensy 4.1, as well as an onboard laptop running Ubuntu Linux and ROS. The computer does all of the navigation computations, and handles the Lidar, GPS, and camera. The Pi takes in the results of the computations done on the laptop, and sends commands to the Teensy which directly communicates with the onboard electronics such as the motor controllers. The laptop can access the Pi via ssh, either by wireless network or directly by using an ethernet connection. The Raspberry Pi and other sensitive electronics are run by the 20 volt batteries while electronics such as lidar and camera are powered by the laptop.

We use a lidar unit called RPLidar A2 to detect obstacles; it provides an all in one lidar solution with premade code libraries that are easily integrated into ROS. Camera data is handled by a logitech USB camera and the OpenCV library. All of the data from lidar and the camera are fed into a laptop that is mounted on the rover and connected to the Pi via SSH.

Each wheel has two motors attached to it, one being a hub motor integrated into the wheel to control directional velocity and position while the other is a planetary motor which controls the vehicles angular velocity and position. The hub motors are run on the 40 volt power system and are controlled by ODrive V3.6 motor controllers while the planetary motors are run on the 20 volt system and are controlled by Cytron DC motor controllers. These motor controllers are a great choice because they are open source and fairly available and affordable. The rover features all wheel drive capabilities, great maneuverability and positioning, as well as individual wheel control.

4.4 Safety Devices and Integration

Bender is equipped with many safety features. Some of which are very clearly shown in physical design. Firstly, there are bright blinking lights on either side of the rover to signify that the vehicle is active. These lights are on whenever the rover has power. There are also lights within the robots internal electronics that will be turned on for various technical reasons. The next safety feature that should be very apparent is the bright red emergency stop button that is mounted on Bender. This button will immediately cease all operations on the rover by directly cutting power.

Less obvious safety features are also included in the design of bender. The power that the wheel motors can use is limited so that Bender does not exceed a safe operating speed. Also, important central processes within the rover are configured in a way that if they are no longer healthy for whatever reason, other processes within the rover will automatically shut down. This prevents motor units from producing unexpected behaviors if their coordinating processes produce an error.

5 Software Strategy and Mapping Techniques

5.1 Overview

To manage the navigation software, we are using the Robot Operating System (ROS). This is a set of tools and libraries that allow for the construction of hobbyist and professional grade robots. In practice, ROS consists of many independent nodes that can easily communicate with each other. The nodes follow a publisher and subscriber type relationship with a master node that every node is ultimately subscribed to. Adding new nodes is very simple, as is allowing the node to publish to the list of active node topics or allowing a node to subscribe to an already published topic. There are many useful libraries that aid in the creation of code, some of which are used in Bender. All of the source code for Bender is arranged within ROS nodes.

We employ ROS' navigation stack to integrate our sensor data and execute navigation protocols. With proper messages received from camera and lidar nodes, the *costmap_2d* library constructs a cost map from its peripheral view. The costmap is represented by grids consisting of numerical values ranging from 0 to 100. The higher the cost, the more likely there is an obstacle or a lane line in that location. We process the costmap information in order to find a viable path that helps the robot traverse the course. Finally, we use a ROS package called *move_base* in conjunction with the costmap to generate viable trajectories. The overall navigation protocol is summarized in Figure 2 and it is discussed in detail in the following sections.

5.2 Obstacle Detection and Avoidance

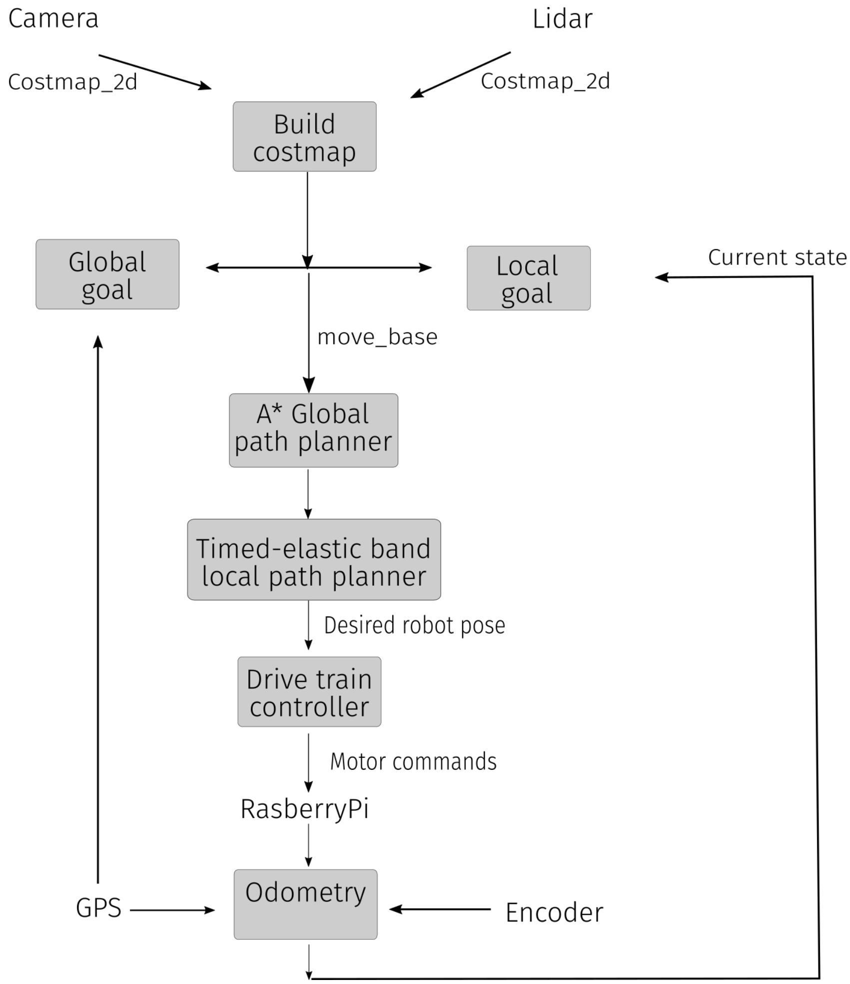
Obstacles are detected using lidar and/or cameras and are converted to a costmap to be processed by other libraries. In our process, white lines are detected using an onboard camera. A mask is applied to the image to block any portion of Bender that is in view, then a gaussian blur is applied to smooth edges in the image. Next the image is converted to high-level synthesis in Opencv, then a gamma correction is applied. After that, a threshold is performed on the image based on the overall luminance of the image and it is dilated to fill in the gaps. Next, a canny edge detection algorithm is performed and the contours of the edges are detected. The contours are filtered and finally transformed onto the ground plane of the image.

After the image is processed and the costmap is generated, the costmap is published to a ROS topic called *obstacle array message,* to be processed in the *costmap_2d* package. The lidar data was directly converted to the costmap, but the camera required further processing. Using OpenCV functions, we developed an image processing pipeline to extrapolate the white lines from the camera feed and represent the information as a lidar scan. The first step was to mask the feed with solid black rectangles to remove the parts of the Bender in view of the camera. Next, Gaussian Blur was used to smooth edges along with converting the whole image to the HSL (Hue, Saturation, Luminance) model to focus on Lightness for the white lines. Gamma Correction was then used with a threshold on luminance to create more distinct levels of luminance that are relatively based with little dependence on outdoor lighting. Dilation was used to fill in small gaps between areas of white. Canny Edge Detection was used to detect the edges of the lines, then contours were found so the algorithm could filter them for size. Finally, we used Homography to convert the image to the laser scan format used by the lidar. This new format was easily combined with our costmap

5.3 Software Strategy and Path Planning

Our plan to move Bender through the challenge is to use GPS waypoints. The course layout is known, so we will place a GPS waypoint at each corner of the rectangle-like path. Bender will then navigate to each waypoint in succession. GPS will inform the global goal that works with the local goal to find the most efficient route. Both of these goals are updated with the current costmap to ensure obstacle avoidance while advancing towards the global goal. We detail more about the global goal, local goal, and costmap below.

Aside from catering to requirements set by the competition, another strategy employed by our software was to adhere to the design pattern of modular, scaleable, and documented code that was maintained by knowledgeable team members. This design pattern allowed for multiple teams to work independently while still providing quality code for the rover.



(System integration and navigation layout)

5.4 Map Generation

Bender uses *costmap_2d* to construct a local and global map of its surroundings. As Bender navigates a path, lidar and camera data are used to create a constantly-updating local map of its surroundings that can be used to create and reach short term destination goals. The local map is used to generate a larger global map that reflects the total path that Bender has traversed. This is made by stitching together individual local map segments.

Going more in depth about map generation, the map generated by Bender is an amalgam of the various costmaps used to navigate towards each local goal. Each costmap is generated by making a so-called laser-image from both lidar and camera data. In this laser-image, a line is drawn at regular angular intervals from Bender to sense and detect each obstacle that is near the robot in each direction. This allows for an accurate plane to be constructed that assigns values to various points in the immediate vicinity of the rover.

5.5 Goal Selection and Path Generation

Bender has both local and global goals that the rover tries to reach as it navigates. A global goal is given by GPS coordinates and they represent Bender's ultimate waypoints, while local goals are tangible paths that the robot can take to move closer to a global goal.

Local goal

A 100 by 100 local costmap grid is received from the *costmap_2d* stack. Let C_{ij} represent the cost of grid located on the i^{th} row and j^{th} column. To identify each lane, we build a cost histogram along the edges of the local map. The histogram simply averages the cost grids along the columns and rows as follows.

$$\overline{C}_i = \frac{1}{100} C_{ij},$$
$$\overline{C}_j = \frac{1}{100} C_{ij},$$

where $\overline{C}_i$ represents the average cost of row i and $\overline{C}_j$ represents the average cost of column j. The base of each lane can be found by the following operation:

$$lane_1 = argmax_k \{\overline{C}_k = \overline{C}_i \cup \overline{C}_j : k = 1, \ldots, 200\},$$
$$lane_2 = argmax_k \{\overline{C}_k = \overline{C}_i \cup \overline{C}_j : k = 1, \ldots, 200 \mid lane_1\}.$$

With the base of the first and second lanes known, we can easily identify the two lanes over the entire local costmap. This helps us place a local goal in between the two lanes.
To find the exact location of the local goal between the two lanes, we draw a circle around the robot at its current location. Then we extract an arc out of that circle such that the arc lies in between the two lanes. *The exact location of the local goal corresponds to the minimum over the value function along the arc*. A value function of each grid v_{ij} is simply the average over the cost of neighboring grids.

$$v_{ij} = \frac{1}{4}(v_{i-1,j} + v_{i,j-1} + v_{i+1,j} + v_{i,j+1})$$

This operation is repeated until each v_{ij} value converges. This allows us to evaluate the cost incurred by stepping on a grid in the long run. For instance, grid $i = 3, j = 3$ may have low cost on the local costmap but this grid may be very close to an obstacle. Unlike the costmap, the value function v_{33} reflects the possibilities of having an obstacle nearby.

Global goal

These points are given as GPS waypoints. Our algorithm simply switches from the local goal to the global goal as we approach our waypoints.

Once the local or global goal is known, we pass this destination as an action to the global path planner. This path planner is implemented in the *move_base* ROS library. In particular, we use the A* global path planner to find trajectories clear of any obstacles and lanes. The global path planner is assisted by the Timed-Elastic-Bands local path planner, which takes the trajectory from the global path planner and optimizes it to the kinematics of the robot, the desired minimum turning angle and other desirable objectives.

5.6 Additional Creative Concepts

The majority of the newly added creative concepts for this iteration of Bender's software revolved around the new navigation system and improvements to the camera and sensory package. Altering the calculation of our costmaps and the pipeline of sub-processes that allowed that to happen was a unique challenge that our team had to take on. Working with a pre-built navigation library was both a blessing and a curse because it took a lot of responsibility away from us, but simultaneously gave us more work to do in the form of conforming our data to something that the Timed Elastic Bands library could utilize.

6 Failure modes, Failure Points and Resolutions

6.1 Vehicle failure modes (software, mapping, etc) and resolutions

Software failures during operation are a possibility, but no crashes have been encountered during testing that didn't have an obvious root cause, such as a wire being disconnected or a battery being depleted. Given the independent nature of our code design, individual nodes within ROS failing may not necessarily endanger other nodes provided that they do not publish crucial data. Furthermore, it is possible for dead nodes to revive themselves in the event of an error and continue operation as normal. As previously mentioned, this is highly unlikely given our strict quality control procedure and intensive testing and code review.

6.2 Vehicle failure points (electronic, electrical, mechanical, structural, etc) and resolutions

The basic structural design of Bender is quite robust. Therefore, we find it unlikely that a mechanical failure would happen during regular operation of the robot. From our testing, we have found that the most likely components to fail would be the motor controllers. It is an uncommon occurrence that a motor will encounter an error and need to be reset in order to continue operation. It is even more unlikely, but still possible, that a motor controller could fail outright. In this case, it is probable that the controller would need to be replaced.

Another failure point that was encountered during quality assurance testing of the rover was the failure of the emergency stop button. Both the housing and the button itself had encountered structural failure during testing and both had to be replaced. One could argue that this would be a safety issue. However, the system is failsafe and Bender was immobilized because of the error. In the updated mounting of the emergency stop button, both the housing and mounting system have been reinforced.

6.3 All failure prevention strategy

All failure prevention is done by ensuring the highest degree of robustness at each stage of development. Employing the philosophy that the most thorough way to perform a task is to perform each of its subtasks as best as possible. In each stage of design, the reliability and robustness of the implementation was considered. Doing this included taking into account the expected real world use cases of every part of Bender's design. To better achieve this task, there was clear communication and documentation of every part that was delivered to the final robot. Regular meetings were scheduled to discuss changes to the design of the robot and a careful review process was implemented to reduce the amount of defects that were introduced to the final design of Bender. The last step in failure prevention was to provide testing that goes above and beyond adequacy. No failure prevention process is perfect because there are far too many variables to take into account, but a well designed engineering process like the one implemented with Bender's design can heavily reduce the amount of defects in the final product.

6.4 Testing (mechanical, electronic, simulations, in lab, real world, etc.)

Testing of Bender was done in stages both in the real world and in simulation. Schematics for the mechanical and electrical designs of the robot were created and testing was done as the designs were created. Since the team was creating a new version of Bender that is a successor to Bender 1.0 and Bender 2.0, we were able to test certain newer components and software on the previous designs of Bender prior to committing them to the newer iteration.

Simulation testing was done mostly to provide proof of concept records of the navigation and sensory modules, as well as test various tweaks and changes that were theorized during the development of Bender. Testing within a simulated environment proved to be a highly valuable resource, but was not necessarily indicative of how the robot would perform in the real world.

Real world testing took place rather slowly and in an iterative manner. After each of the individual systems were tested and proved to be operational, Bender was tested with some basic scenarios in our testing environment near our design shop on campus. Firstly, a basic course was set up using some simple straight lines laid out using white tape. After the rover was able to successfully navigate the basic course layout, more obstacles were added. Basic turns were simple to create using white tape and barrels were added to the course to simulate obstacles. Finally, a type of chicane was set up using multiple barrels, this proved to be a good test of Bender's capabilities. We had intended to test using a form of elevated bridge, but were unable to source a structure to simulate this type of obstacle.

6.5 Vehicle safety design concepts

Safety should always be a top priority whenever you are performing any form of science, robot design is no exception. Bender has built-in safety systems to ensure the safety of everyone nearby. Firstly, there is a stop button located on Bender that is brightly colored, easily accessible, and will immediately stop all functionality of the robot by cutting power to the system. Secondly, there is a remote stop feature that will perform a similar action of stopping the robot by cutting power. Using the remote stop function, a critical system is killed that provides a heartbeat to all motor control systems on Bender. Without this heartbeat signal, no movement operations can be performed. The remote stopping mechanism was designed in the fashion to best preserve the delicate electronics on Bender, while still safely and reliably stopping the robot as quickly as possible.

7 Simulations employed

7.1 Simulations in virtual environment

Bender is simulated in a virtual environment using ROS Gazebo. In the simulation, Bender can navigate through different pre-built courses to test various aspects of navigation and sensory performance. During simulation, there are a few factors that behave quite differently from real life testing. Notably, certain pieces of data such as odometry are perfect within the simulation. Obviously, data in the real world will never be perfect, which is where we run into a gap between testing and actual implementation. Even though the simulation is a valuable tool that can help with many aspects of development, it does not necessarily speak for exactly how Bender will perform in real life.

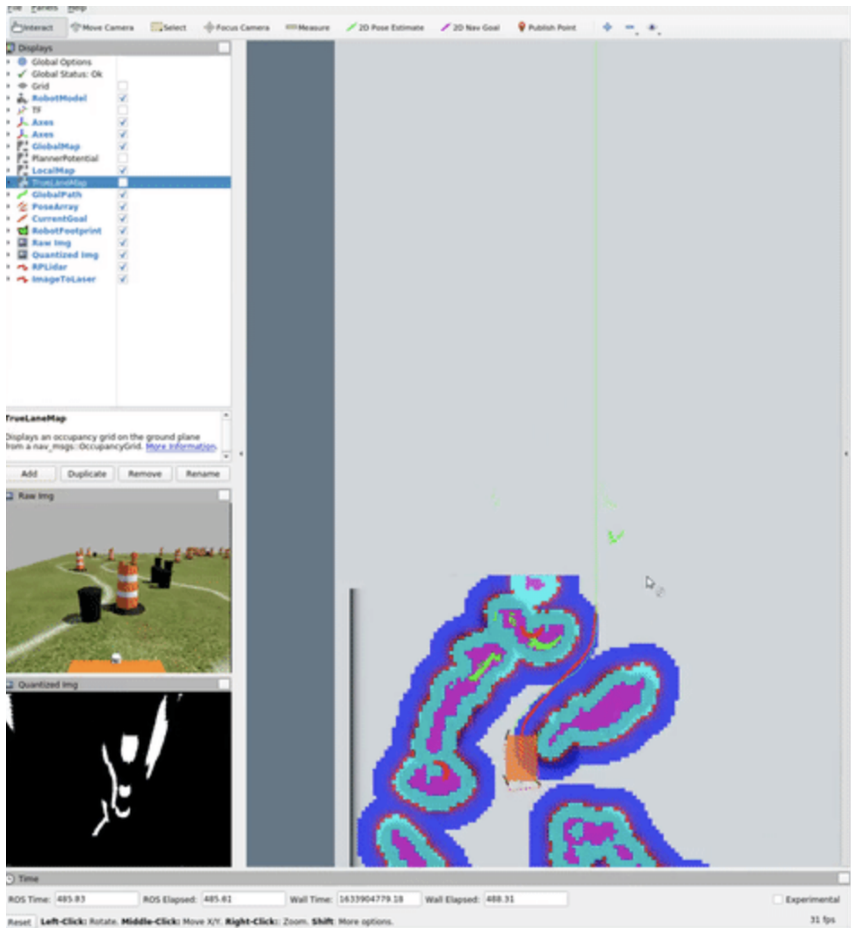


Figure 2: Simulation of Bender in Gazebo, viewed in Rviz

7.2 Theoretical concepts in simulations

Some concepts within the simulation are handled quite differently from real life testing. Firstly, there is a lack of intense shadows that would otherwise obstruct the image that we are receiving from the camera. In real life, shadows can have a drastic effect on the image that the robot receives. With certain processing techniques, this can be somewhat mitigated, but drastic changes in brightness across a landscape can still affect the decision making process of the software.

Secondly, odometry and gps data is simplified within the simulation. On the actual robot, odometry is handled by combining data from an inertial measurement unit (pixhawk px4) and also from taking data from the wheels of the rover. Each wheel has encoders that are equipped with holofex sensors. By measuring the number of signals from the holofex sensors, we can estimate the amount of distance traveled by each wheel. Accounting for slippage, this can provide us with fairly accurate odometry data.

GPS data is provided by the pixhawk as well. In the simulation, all GPS and odometry data is perfect, as well as lidar readings. This helps immensely with certain functions within Bender. Aside from general navigation being far easier, the creation of maps that Bender generates as it traverses a course is more accurate.

8 Performance Testing to Date

8.1 Component testing, system and subsystem testing, etc.

In our testing, we have found that a majority of the newly implemented systems are performing well within our expectations. The newly implemented navigation system has achieved update rates of approximately one every second. This is taking into account processing the costmaps and planning an appropriate path. Another important improvement on this design of Bender would be the improved frame and chassis. With the new design, Bender is much better at adapting to changing terrain and transfers less vibration from the ground to the electronics of the vehicle. This prevents connections from coming loose and provides a better image and lidar quality. In our tests, the newer version of bender hardware and software were able to perform better than the previous iteration in all practical aspects.

9 Initial Performance Assessments

9.1 How is your vehicle performing to date

As of the writing of this document, Bender is performing satisfactorily. The robot meets all of our criteria for a functioning autonomous vehicle and we are confident that it will be able to perform well in the upcoming competition based on preliminary testing. The remainder of the time before the competition will be spent finalizing the design of Bender, conducting further testing, implementing bug fixes, and implementing a shipping strategy to transport the robot across the United States.