CRAB-E

Competitive Robot for Autonomous Barrier Evasion

Rocker Robotics Team President
Dustin Richards | dustin.richards@mines.sdsmt.edu
Rocker Robotics Team Advisor
Rohan Loveland | rohan.loveland@sdsmt.edu

Members

Autonomy & Controls	**Mechanical**
Bennet Outland † - Sophomore, ME Dakota Edens - Senior, CENG Alexis Englund- Freshman, CENG Landon Lamoreaux - Junior, CSC Josiah Huntington - Junior, CENG Trace Houser - Sophomore, CSC Dustin Richards - Graduate, CENG/CSE* Ashley Schnetzer - Junior, CENG	Kaylee Herndon † - Sophomore, ME Daniel Block - Freshman, ME Heath Buer - Junior, ME/EE Tim Chandler - Freshman, ME Jacob Decker - Freshman, ME Steven Duong - Sophomore, ME Rachel Lauer - Senior, ME Derek Matthies - Freshman, ME Ethan Miller- Freshman, ME Sam Ryckman - Graduate, ME/CSC/CSE* Zane Wilson - Freshman, ME
Electrical	
Erick Eickhoff † - Senior, EE Chase Reinertson - Senior, EE Pratik Sinai Kunkolienker - Graduate, EE/EE*	

† Team Lead

CENG: B.S. Computer Engineering | CSC: B.S. Computer Science | CSE*: M.S. Computer Science and Engineering | EE: B.S. Electrical Engineering | EE*: M.S. Electrical Engineering | ME: B.S. Mechanical Engineering

List of member email addresses: bennet.outland@mines.sdsmt.edu, dakota.edens@mines.sdsmt.edu, alexis.englund@mines.sdsmt.edu, landon.lamoreaux@mines.sdsmt.edu, josiah.huntington@mines.sdsmt.edu, trace.houser@mines.sdsmt.edu, dustin.richards@mines.sdsmt.edu, ashley.schnetzer@mines.sdsmt.edu, kaylee.herndon@mines.sdsmt.edu, daniel.block@mines.sdsmt.edu, heath.buer@mines.sdsmt.edu, timothy.chandler@mines.sdsmt.edu, jacob.decker@mines.sdsmt.edu, steven.duong@mines.sdsmt.edu, rachel.lauer@mines.sdsmt.edu, derek.matthies@mines.sdsmt.edu, ethan.miller@mines.sdsmt.edu, samuel.ryckman@mines.sdsmt.edu, zane.wilson@mines.sdsmt.edu, erick.eickhoff@mines.sdsmt.edu, chase.reinertson@mines.sdsmt.edu, pratik.sinaikunkolienker@mines.sdsmt.edu

I, Rohan Loveland, certify that the design and engineering of this vehicle by the current student team has been significant and equivalent to what might be awarded credit in a senior design course.

Signature: ______________________ Date: ______________________

Contents

Team Introduction

Rocker Robotics is an extracurricular robotics team from the South Dakota School of Mines and Technology. The goal of the team is to teach robotics concepts to its members, bring robotics and STEM outreach to our community, and compete in a robotics competition each year. Its members have a range of robotics experience, majors, and backgrounds. Dustin Richards is the president. Bennet Outland is the autonomy and controls team lead. Kaylee Herndon is the mechanical team lead, and Erick Eickhoff is the electrical team lead.

Summary

Our robot is named CRAB-E (Competitive Robot for Autonomous Barrier Evasion). It has a four-wheel steer drive system. Across all members of the team, we spent approximately 5600 person hours on this project. We are using computer vision to identify the white lane lines and potholes. LIDAR and ultrasonic sensors are used to identify 3D barriers in our path, and a Global Positioning System (GPS) to estimate our location on the course. We create a map of obstacles that we see in the world, and then use the A* (A-star) [1] path planning algorithm to create a path that avoids all these obstacles.

The processing is split between an Nvidia Jetson Nano, Raspberry Pi, and STM32F411 microcontroller. The computers communicate with each other over an ethernet connection, while the sensors are connected to the STM32F411 via a custom printed circuit board (PCB). The E-stop is an isolated board that controls the state of a relay that can interrupt the power connection to our motor controllers. This relay can be actuated either by the E-stop button on the robot or the button on either remote using a wireless connection. The robot is capable of four-wheel drive using four NEO 550 brushless DC motors and uses two brushed motors for steering. A four-wheel steer design which is similar to the standard Ackmerman steering found in cars is used, with the added functionality of the front and rear steering systems being able to turn independently of each other. This system allows a significantly tighter turning radius than a standard Ackermann drive system. The design also included the development of a frame to support the steering mechanism, and the outer shell of the robot.

Acknowledgements

We would like to thank Dr. Rohan Loveland, our advisor, for all his help throughout this year. We would also like to thank South Dakota School of Mines and Technology and the Electrical Engineering, Computer Engineering, and Computer Science (EECS) Department for providing us funding and lab space in which to design, build, and test our robot. Additionally we thank the team's umbrella organization, South Dakota Mines CAMP, for helping us with leadership training and team administration. For assisting in design reviews, we would like to thank Dr. Rohan Loveland, Dr. Mohammadreza Mehrabian, Dr. Jason Ash, Dr. Ryan Koontz, and Mr. Lowell Kolb.

Design Assumptions and Process

Design assumptions for this robot began with its size. The robot must fit within the constraints set by the competition rules [1]. Past this, we targeted a few major design features: the robot should be car-like, have a turn radius significantly tighter than the tightest turn described in the rules, be relatively robust, and not be so tightly integrated that it's difficult to build and maintain. The team placed a large emphasis on working around materials already available in their lab. Notable parts of the robot that are in line with this include the LIDAR, most of the frame, and the wheels and tires.

Mechanical

At the beginning of the school year, our decision making was guided by decision matrices. After working through multiple group discussions based on these decision matrices, we settled on the final architecture for the robot: a rigid chassis, dual-Ackerman steering, independently driven wheels, and a body in the style of a compact pickup truck. We then started work on a wooden prototype chassis and later designed a final chassis based on aluminum T-slot extrusion, making use of the lessons learned from the prototype chassis.

Autonomy and Controls

As for sensors, computers, and software stacks, much of the decision making was based on what the members of the Autonomy and Controls subteam were already comfortable with. This led to us using an Nvidia Jetson Nano and Raspberry Pi, both running ROS [2] on Ubuntu Linux, and the Arduino framework on our STM32F411 microcontroller. Sensor choices were guided by considering what a typical autonomous car might have, leading us to use a combination of a LIDAR, stereo cameras, a GPS, and ultrasonic distance sensors.

Electrical

The primary goal for the electrical team was to integrate most of the required electronics onto printed circuit boards and perfboards, using keyed connectors and SMD/THT components available in our parts selection whenever possible. Another key design feature we focused on is the ability to hot-swap the robot's primary battery while keeping the computers running via a backup battery.

Innovations

As a team, we innovated in the areas of computer science, electrical engineering, and mechanical engineering. Some of the software innovations presented from this project include sensor processing methods and a novel waypointing method. We created a novel algorithm for detecting lane lines and converting them into an obstacle point cloud in three dimensions. The line-obstacle data is then used to generate a valid driving region for the robot. From this point, a ray is cast through the objects and the optimal target waypoint is determined after filtering. Both of these methods will be further explained in the Autonomy and Controls section of the report.

In terms of electrical engineering, we designed a printed circuit board to handle the interface between sensors and our computers. Additionally, we developed our own E-stop board and software, so we can safely stop the robot from a distance according to the competition rules. The E-stop board allows us to disconnect the motors from the battery power while always keeping power to the other electronics to avoid potential filesystem corruption if our computers were to lose power. The E-stop board is electrically isolated from the power of the rest of the robot. It uses 915 MHz radio modules that allow for two-way communication with multiple remotes to allow for multiple ways to turn the robot off if needed.

On the mechanical side of things, the steering architecture chosen is a four-wheel steer system, specifically a dual-Ackermann drive. The main benefit of this drive system is that the robot is able to execute tight turns compared to other systems [3]. Through this system, the robot is also able to execute a kind of steering informally known as “crab-steer.” This allows both the front and back wheels to be turned in a way that allows the vehicle to move in a diagonal direction while the overall orientation of the chassis is still straight. The steering is powered by two motors, front and back, that move curved linkages along a linear rail. These linkages, rather than straight ones, assist in decreasing the overall turning radius. The steering and drive systems are attached to an aluminum frame, which helps to support all mechanical and electrical components used within this design. To provide both water-resistance to electronic components and aesthetic appeal, a 3D printed shell was created to rest on top of the chassis.

Mechanical

Overview

For this project the mechanical subteam was responsible for the prototyping, designing, and manufacturing of the chassis and related components; including several subsystems such as the frame, drivetrain, and outer shell. The key idea we applied to our decisions was simplicity; we created a list of the features we wanted to include and brainstormed ways to achieve these features without adding unnecessary complication to our designs. The desire for simplicity became a driving factor in all of our decision making processes.

Frame

The backbone of the frame consists of a spine of 3” x 1 ½” T-slot with a 1 ½” x 1 ½” aluminum T-slot mounted perpendicular to the spine at each end of this center beam as can be seen in Figure (1). This creates a proper support structure for all electronic components and the payload, and adds attachment points for drivetrain components. Aluminum plates and brackets are used throughout the frame to strengthen and increase rigidity in the chassis, as these create cross braces within the frame. These components are also used to aid in mounting the drivetrain systems and shell. The T-slot material was chosen as there was an abundance readily accessible to our team. Along with this, the material provides strength to the frame without adding an excessive amount of weight to the chassis.

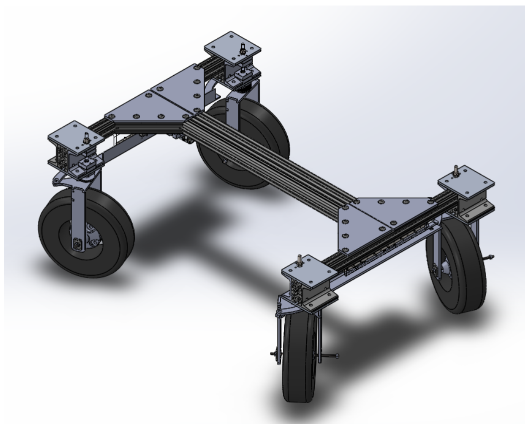


Figure 1: CAD Model of Chassis

Steering

The most unique part of our robot is the steering system. Overall, we chose a setup very similar to a dual-Ackermann drivetrain involving two pairs of steered wheels. During the process of choosing a drivetrain style, we worked through multiple options looking for the best mix of benefits and costs. The process led to the decision to use an Ackerman style drive. The weight of the robot and the ability to minimize our turning radius led to us adopting this style of steering. As shown in Figure (2), each wheel is independently powered but steered in tandem by a steering motor connected via curved linkages. Curving our drive linkages provides our system with the four-wheel drive and crab-style steering options we want, whilst drastically minimizing our turning radius. Currently, we have lowered our radius to two feet, thanks to our implementation of curved linkages. These linkages allow our wheels to pivot further around their axis than straight linkages would, all while keeping the simplicity we desire. While we considered how rough the terrain of the course will be, we decided to forgo a suspension system and instead rely on heavy duty off-road tires to reduce the overall complexity.

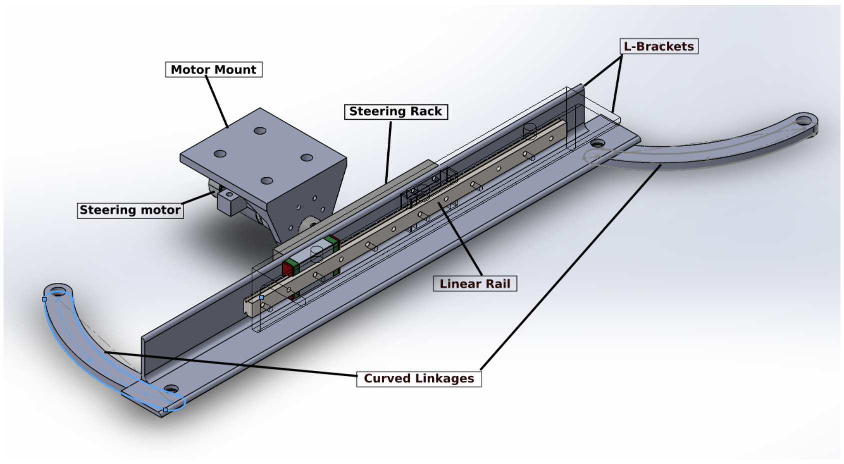
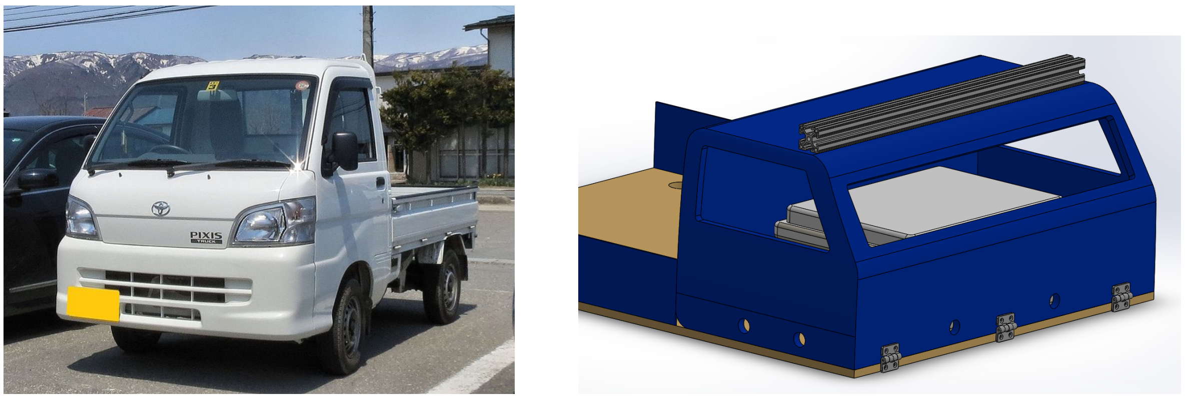


Figure 2: Steering Assembly

Outer Shell

The shell serves a variety of functions, including providing aesthetic appeal. Additional functions include providing additional mounting points for the camera and LIDAR systems, acting as a basic weatherproofing step, and providing a secure place for the payload to be stored. The design of the shell is based on Japanese Kei Trucks, which was chosen due to the relative simplicity of the shell design. The shell was manufactured in multiple pieces in order to make use of multiple large-bed 3D printers, reducing print time. Included within the shell subsystem is weatherproofing, which is done by a using commercially available waterproof container which will protect the computers, batteries, and other components.



(a) **(b)**

Figure 3: **(a)** Example of a Japanese Kei Truck that the design was based off of [4]. **(b)** "Cab" of the robot, along with other shell components.

Electrical

Computers

The intensive processing is split between a Nvidia Jetson and a Raspberry Pi. Specifically, the Nvidia Jetson Nano is being used as the main processing unit, this includes image processing and path planning calculations. The Raspberry Pi, on the other hand, will mainly deal with LIDAR data ingesting and sending commands to the STM32F411. Additional processing for less intensive systems, such as the Inertial Measurement Unit (IMU) or GPS, is done by the STM32F411.

Sensors

Primary object detection is performed by a Hokuyo UTM-30LX LIDAR sensor. This unit has a field of view of 270 degrees with a detection range of 0.1m to 30m. Line and pothole detection is performed by a pair of Intel RealSense Depth Camera D435s. These will be used to identify and locate the lines and potholes of the track. Additional location information will be obtained from an Adafruit Ultimate GPS. A BNO055 inertial measurement unit (IMU) will be used to determine the orientation of the vehicle. Ultrasonic sensors will be placed at the front of the robot to serve as emergency obstacle avoidance sensors in case the robot gets too close to an object that was missed by the LIDAR. Additionally, a suite of six ultrasonic sensors will be used as a safety measure to catch any obstacle collisions if not detected by the other sensors.

Motors

We are using 4 Neo 550 brushless DC motors as the main drive system. These motors were chosen in part because they are very powerful for their price. There is a Spark Max motor controller for each of the motors. The Spark Max motor controllers have built-in voltage regulation for the motors, as well as compatibility with CAN for easy communication. CRAB-E's steering is achieved using two brushed motors we had on hand. These motors are each controlled with a separate Talon SRX motor controller, which boasts similar specifications as the Spark Max's, but for brushed motors. Both types of motor controllers are rated for continuous currents of 60A, which is a much higher current than we expect to see.

Power Connections

Our robot uses a 4-cell, 20Ah LiPo battery to provide power to the system, which runs at approximately 14.8V at full charge. The batteries will have direct connections to our LIDAR, beacons, 5V power supply, and motor controllers. The beacon state is controlled by the STM32 to account for the current state of the robot. The motor controller connection is split with a power distribution board to get 14.8V to each of our six motor controllers, and can be interrupted by a relay for the emergency stop system, as detailed in the E-Stop subsection. We are using an off-the-shelf 5V, 50W DC-DC converter to provide 5V from our 14.8V battery. There is an additional 3-cell battery protected with a diode that will continue to provide the sensitive electronics with 5V when the main battery is disconnected. This 5V connection is routed through a bus bar to both of our computers and to our 5V logic board. The logic board is a custom PCB designed to route power to our remaining sensors and provide communication connections to the electronics not directly connected to our more powerful computers. This board contains an integrated controller area network (CAN) module to connect to our motor controllers. The logic board PCB also holds the STM32 which will connect to the other computers through its USB connection.

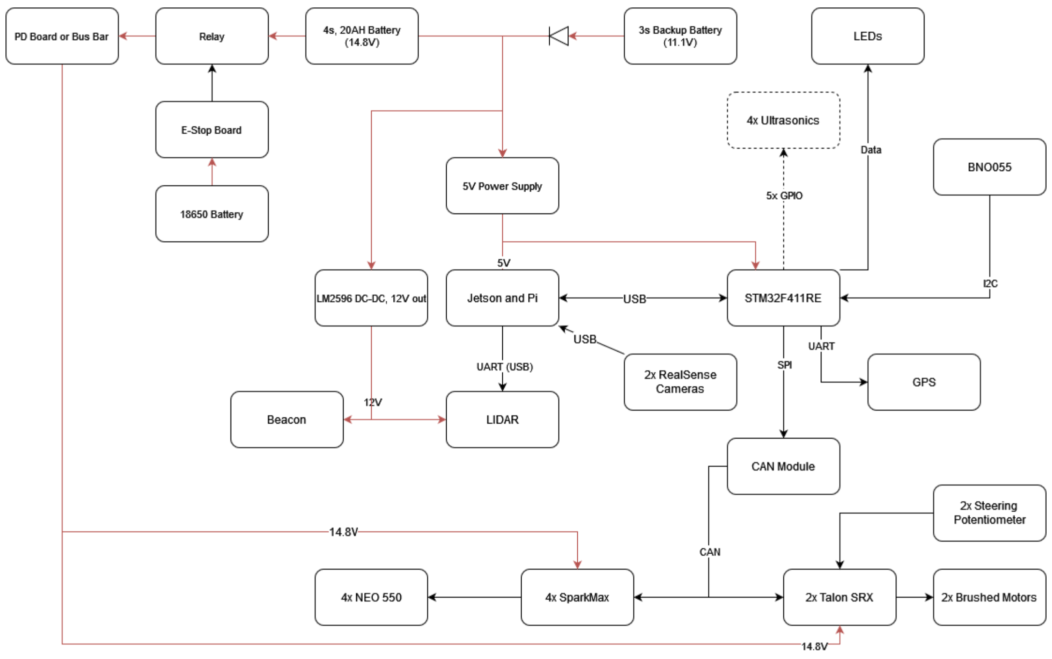


Figure 6: Electrical System Diagram

E-Stop

As required by the rules for this competition, we have designed an emergency stop system into the robot. This system is an isolated circuit from the rest of the electronics. It is based on a dual purpose custom PCB that, depending on configuration, can be used either as a transmitter or receiver. The receiver configuration will be located on the robot itself, and will be controlling the state of a relay that connects our motors and motor controllers to the battery. This battery connection is separate from the battery connection to our main computers and sensors, allowing us to kill power to all of the drive systems while keeping power to the electronics that may be more sensitive to sudden power loss. The receiver board will open the relay contacts if either the E-stop button onboard the robot is pressed, or it receives a command from the remote E-stop. The remote E-stop board uses the same PCB in the transmit configuration, which will communicate with the receiver board using RFM69HCW packet radio modules. Pressing the E-stop button on the remote sends a status update to the receiver board which will then open the relay contacts the same way that the E-stop button on the robot does.

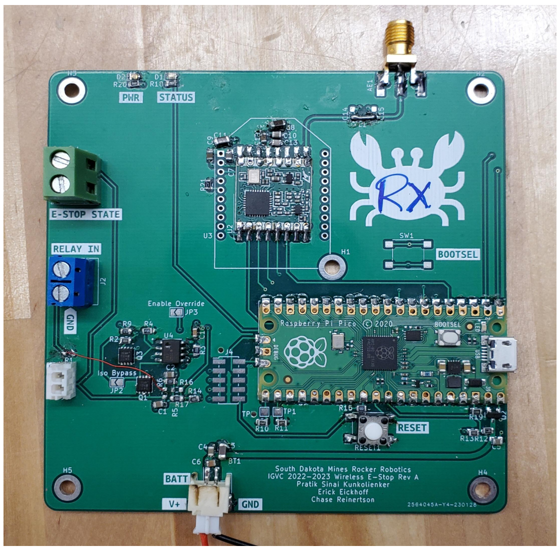


Figure 7: E-Stop Board

Autonomy and Controls

Paradigm

The approach we took towards controlling our robot autonomously is based on the concept of treating all course obstacles (lines, barrels, potholes, etc) identically. By doing so, we are able to have a robust and general framework for navigating through the course. In the case where there are not any lines, the obstacles will still be considered with the GPS coordinates acting as waypoints for the robot to follow. Through this approach, we construct our avoidance and planning paradigm to standardize the outputs from all obstacle related sensors. All of the sensor interfacing and communication will be done through ROS [2].

LIDAR

Using a 2D LIDAR unit, we are able to detect the location of the barrels throughout the course. Note that the LIDAR is positioned to not detect the ramp in the region where we are to navigate using the GPS waypoints. Through the remainder of the course, we are able to determine any reported distances to various obstacles. This is a valid assumption since everything of the height of the barrels is an obstacle in the course or something we still do not want to collide with, such as someone wandering onto the course. The LIDAR sends these obstacle locations to the map generation algorithm as described below.

Obstacle Detection & Map Generation

The robot uses two Intel Realsense D435 cameras for lane following and pothole avoidance. Each of these cameras provides an RGB stream and a depth stream. Both streams are read in and an alignment operation is performed on the depth stream to align it to the color stream. Figure 3 shows the algorithm used to convert the color image into a point cloud that can be used by the path planning algorithm. The color image is converted into a grayscale image and then adaptive thresholding is used to find where any white lines are by looking at each pixel and its neighbors to see if they are significantly different, if they are, it is probably a white line and is kept in the image. The image is then eroded to remove any noise and the top half is masked out to reduce the sample complexity since the ground is where the obstacles are. The resulting image just has the lane lines and any potholes. Combining this with the depth image allows us to create a point cloud of all the points that need to be avoided. This algorithm is repeated for the second camera. The two point clouds are then transformed to be in the global coordinate frame. These point clouds are combined with the point cloud from the LIDAR to create one point cloud of all the obstacles. All these points are mapped to an image to create a picture of everything around the robot. The path planning algorithm uses this point cloud to navigate around the course.

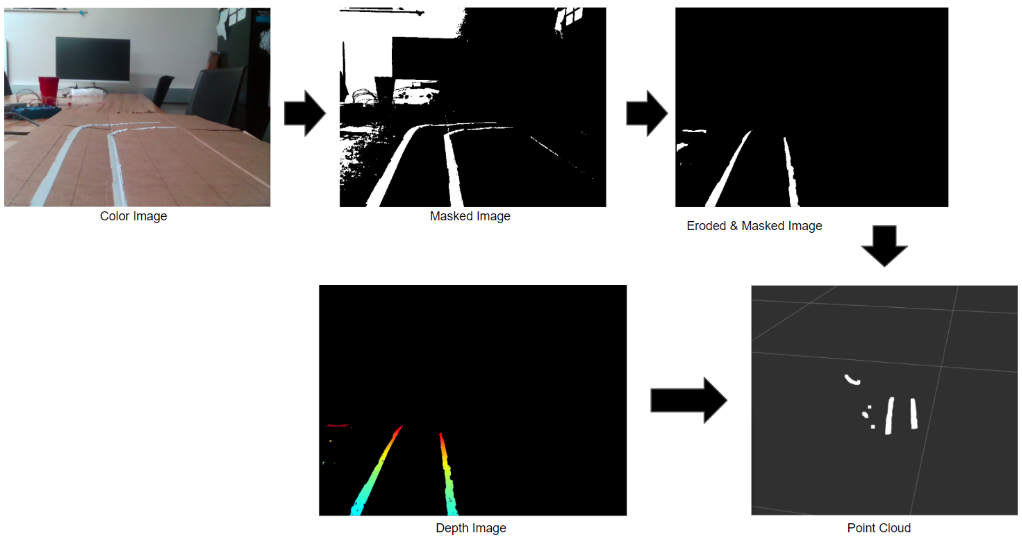


Figure 4: This is a visualization of each of the image manipulation stages discussed. The depth image here only contains the points from the eroded masked image.

State Estimation

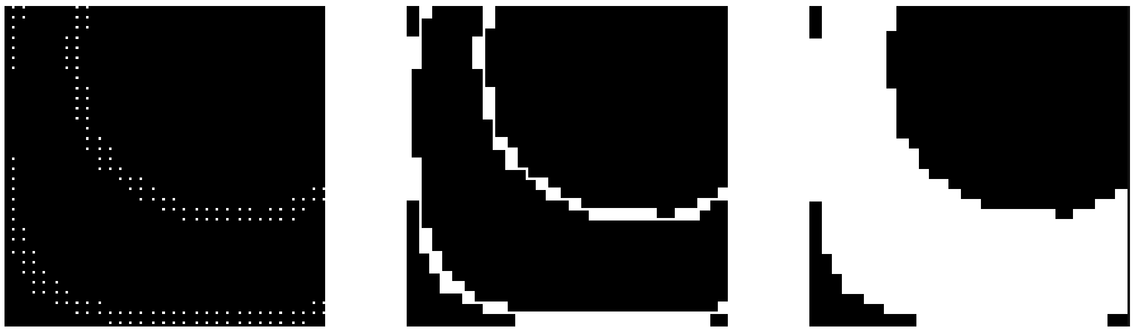
An important aspect of many of these sub-algorithms is knowing the location of the robot. In order to accomplish this we are using an Extended Kalman Filter (EKF) for state sensor fusion. An EKF is derived from the optimal, Gaussian, linear state filter, a Kalman filter, where the linear filter is used to approximate nonlinear behavior for small timesteps [5]. Given estimations of the state and the expected error of the sensors used, a robust approximation of the actual state can be determined.

For the location of the robot, we are using a combination of a GPS using the Wide Area Augmentation System (WAAS) and a 9-axis IMU. We are using a Bosch Sensortec BNO055 IMU, which nicely calculates an absolute orientation estimate. With the data from these sensors, we are able to get accurate estimates of the actual position and heading of the robot.

Path Planning

Due to the algorithm mentioned in the last section we are able to determine where the obstacles are in relation to the robot. We can then start determining what path the robot will need to take to traverse the course. In order to determine the optimal path, three different elements must be determined. First, the sub-state must be found that describes where the robot can and cannot be. Second, the target waypoint that the robot travels towards is set. Lastly, the path to move between the current robot state and the target waypoint is determined. Each of these can be considered individually.

In order to determine where the robot is allowed to drive, we can look at the local state and find all obstacles that are a fixed distance from the robot. This is done so that we may focus only on the obstacles that will have a relevant impact on the direction of the robot. From this point we take the local state that has been created and compress it to a smaller state represented by an $n \times n$ gridspace to save on computational expense. With this gridspace defined, we have a loose collection of obstacles spread throughout as seen in Figure (5a). Issues can arise with the invariability of sensor noise, resulting in obstacles not being detected or spuriously detected. To counter this, we can apply a dilation kernel to fill in any gaps where obstacles should have been detected. If there were any small areas of spuriously identified obstacles, an erosion kernel can remove the extraneous obstacles. Ultimately, we are then left with a cleaned region with the dilated obstacles being returned to their original size. The results of applying the two kernels can be seen in Figure (5b). Under the assumption that the robot is currently between the lines, we apply a flood fill algorithm to define the region between the lines where the robot is allowed to traverse. This can be seen in Figure (5c). This was accomplished using the OpenCV library [6].



(a) **(b)** **(c)**

Figure 5: Different steps in the processing of the algorithm to determine the driveable region. **(a)** Raw obstacle data from sensors. **(b)** Obstacle representation after dilation and erosion. **(c)** Drivable region determined post-floodfill.

Given that a traversable region is defined, we can start locating a waypoint for the robot to navigate towards. To accomplish this, we use a "T-beam" with a fixed width web and flange where the size of the flange is much greater than the size of the web. This is done through the following:

- Let F be the set of points from some (-l, 0) to (l, 0) varying the x component
- Collect the current orientation of the robot, θ
- Rotate flange by $\theta + \frac{\pi}{2}$
- Shift the central point in the flange to the robot through additive broadcasting
- Calculate the location of the end of the web using the given distance and the current orientation of the robot
- Shift the central point in F to the tip of the web through additive broadcasting

From this point, we want to determine a target point that is not near obstacles and would not result in needing to take sharp turns unnecessarily. To do this, we apply a Gaussian filter to the grid elements along the flange to smoothen the discontinuous steps between obstacle and non-obstacle regions. This results in a set of hills and valleys where the hills represent the favorable drivable regions since they are devoid of obstacles while adding a buffering region near the obstacles. To mitigate sharp turning, a bandpass-like filter is applied that increases the favorability of smaller turns and outside the central region is subjected to a decrementing linear trend. A linear search algorithm is then used for determining the maximum along the now filtered flange elements. Through this process, we are able to extract a better target point by taking into account the local obstacles and the turning angle.

With a target point defined, we can define the starting position to be the currently estimated state of the robot. To plan a path between the starting and target waypoint, we use A* due to its superior performance in terms of its percent optimal performance per calculation time [7]. We create the grid space so that the driveable areas have minimal weights to move between nodes while non-traversable areas have high node weights. We extract a path from the A* algorithm created for the smaller $n \times n$ gridspace. The path is now transformed from the compressed gridspace to the local region. While this does cause some discretization error the error is negligible due to the small regions considered versus the size of the compressed grid. As we step through the path, we can determine the steering angles needed.

Robot Control

Ultimately, we can break up the control of our robot into three main categories: steering, desired velocity, and stopping. In order to steer the robot, we need to ensure the wheels are oriented in the correct direction and stay in that direction over the duration. First, we need to read the target angle from the path planning algorithm. From this point, we have tuned the motors to certain easily identifiable angles and interpolate between to achieve the correct angle. To ensure that the wheels stay at the correct angle a PID loop has been tuned for the forward and rear wheels. Another problem to approach is the speed of the robot. To dynamically adjust the speed, we are correlating the speed of the robot with the obstacle density of the local region. We take a baseline of the starting point to be the least dense in terms of obstacle density and scale the target speed down as the number of obstacles increase. A maximum of 4.75 mph has been set in software to avoid exceeding the maximum speed of 5.00 mph. We also explored different methodologies for developing a remote E-stop. We are currently using a RFM69_HCW and Raspberry Pi Pico for sending and receiving E-stop commands.

Simulation

We used both physical and software simulation environments to validate our algorithms. Specifically, in the cases of the GPS, orientation sensors, and the depth cameras, physical simulation environments were created for real-world validation. In the case of path planning and state estimation, we used an in-house simulator that was developed to be lightweight and versatile. A sample output can be seen in Figure (6). Through both of these environments, we were able to test and validate the algorithms created for this project.

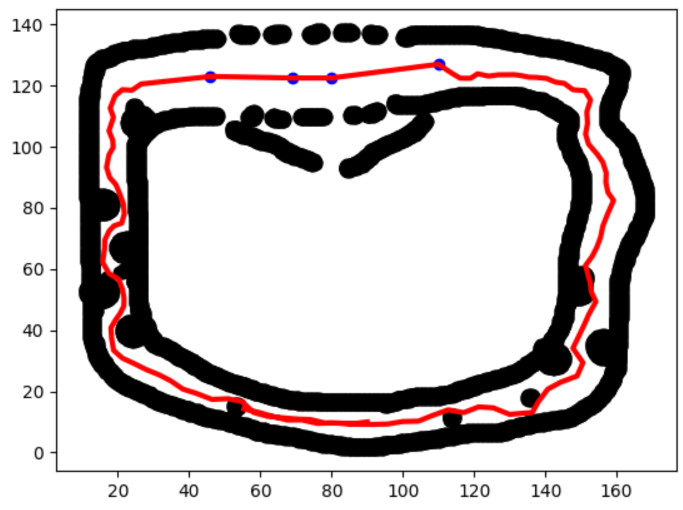


Figure 6: Example figure from the simulator used demonstrating the robot consistently traversing the course by the placement of internal waypoints. These waypoints have been connected to demonstrate the path, but the A* generated path will ultimately be used. Note that the blue dots are simulated GPS waypoints.

Potential Failure Points

There is a potential of failure points in the autonomy, electrical, and mechanical systems. In terms of autonomy, the computer vision system may identify the lines improperly. Changing lighting conditions throughout the run and shadows over the line could cause our robot to not see some of the lines. Autotuning to the lighting conditions has been implemented, but rapid changes may cause a momentary loss of obstacle data from the camera.

In the electrical system, the main two points of failure are from vibrations and water ingress. Due to the lack of a suspension and the slightly bumpy terrain, frequent vibrations are expected. Vibration of the wires over a long period of time may slowly disconnect various connectors. To combat this, electrical connections are all either soldered in place or attached by snug connectors. As for water ingress, the robot is not designed to be waterproof, but rather simply water resistant. An excessive amount of water on the robot may cause some water ingress and might cause shorts. This has been mitigated against, but cannot be guaranteed to be perfect. Should either of these cases occur, spare components and connections will be brought along to the competition. Tape may also be used to provide additional temporary sealing for problem areas. Further, as the intention was to keep each sub component as cheap, the PCBs lack electrostatic discharge protection and reverse polarity protection. To mitigate these issues, care will be taken when handling sensitive electronic components like the Jetson and key'd connectors will be used to prevent incorrect connections.

In the mechanical subsystem, there are a few points of failure that could occur. One possible failure point for the chassis is at the wheels and axles. As the entire weight of the robot rests on the two components, there is a chance that either of them could break if the robot is overloaded, or the shafts for the wheel assemblies could bend. To work around this, we used aluminum and 3D printed parts to help reduce the overall weight of the chassis and distributed weight thoughtfully throughout the chassis, allowing each wheel to receive less of the load. Repeated sharp steering angles will increase the stress through the steering mechanism that may cause failure if used for long periods of time. We resolve this issue through our control of the system by disincentivizing sharp steering angles when the path that the robot takes is planned. Another concern is vibrations in the robot loosening bolts, to prevent this a thread locking compound and lock nuts were used to keep bolts tight and in place. Design reviews were also conducted with advice from faculty to ensure a robust design in the more critical sections of the robot.

Predicted Performance

We predict to be able to perform adequately for a newcoming team. However, we do not expect optimal performance and will need to improve through additional iterations. Our robot is predicted to be able to identify obstacles reasonably well, and be able to create a path that avoids all of the identified obstacles. In simulation, the robot has been able to successfully traverse the course. Based on worst-case power consumption, we expect the robot's battery life to be at least two hours. Based on how far geared down the drive motors are, this robot should be able to climb the ramp with no issues. Lane lines will be detected five to ten feet ahead of the robot. Obstacles detected by the LIDAR will be seen at 20 to 30 feet away from the robot.

Cost Breakdown

Product Name	Unit Cost	Quantity Used
Hokuyo UTM-30LX LIDAR*	$4,675	1
Intel Realsense D435	$314	2
Nvidia Jetson Nano*	$149	1
Turnigy 20000mAh Battery*	$142	1
Rev Robotics Spark Max	$90	4
Vex Robotics Talon SRX	$90	2
Grantury Junction Box	$63.49	1
6061 Aluminum U Channel - 3ft	$42.64	1

Rev Robotics Ultra 90° Gearbox	$42	4
McMaster-Carr Metal Gear Rack	$41.61	2
Raspberry Pi 3 Model B*	$35	1
Rev Robotics NEO 550	$28	4
MGN12H 300mm Linear Rail	$28	2
1kg 3D Printed Material	$25	4
Rev Robotics 5mm Hex Shaft, 400mm length	$21.50	2
3 JSN-SR04T Ultrasonic Distance Sensor	$20	2
QCQIANG E-Stop Button	$18.29	1
McMaster-Carr Metal Gear 20° angle	$13.68	2
Rev Robotics UltraPlanetary Cartridge 4:1	$11.50	12
10” Pneumatic Tire and Wheel*	$8.49	4
Rev Robotics 5mm Hex Bearing Block	$4	4
1.5” x 3” T-Slot Extrusion - Cost is Per Inch*	$1.02	33
1.5” x 1.5” T-Slot Extrusion - Cost is Per Inch*	$0.57	96
Total Cost: $7199.34		

* Denotes items the team had on-hand and did not need to purchase

References

[1] IGVC. (2022, October). The 30th Annual Intelligent Ground Vehicle Competition (IGVC) Self-Drive.

[2] Stanford Artificial Intelligence Laboratory et al. (2018). *Robotic Operating System*. Retrieved from https://www.ros.org

[3] DY5W-sport. (2013). Wikipedia. Retrieved May 15, 2023, from https://upload.wikimedia.org/wikipedia/commons/thumb/4/4e/TOYOTA%E3%83%BBPIXIS-TRUCK_211U_0164.JPG/788px-TOYOTA%E3%83%BBPIXIS-TRUCK_211U_0164.JPG?20130509084247.

[4] McGough, J. (2022). *Introduction to Robotics* (2.3.0).

[5] Thrun, S., Burgard, W., & Fox, D. (2005). *Probabilistic Robotics*. The MIT Press.

[6] Bradski, G. (2000). The OpenCV Library. *Dr. Dobb’s Journal of Software Tools*.

[7] Fareh, R., Baziyad, M., Rahman, M., Rabie, T., & Bettayeb, M. (2020). Investigating Reduced Path Planning Strategy for Differential Wheeled Mobile Robot. *Robotica, 38*(2), 235-255. doi:10.1017/S0263574719000572