Wayne State University
Warrior Robotics

2023 IGVC Design Report

Team Captains

Luay Jawad	ljawad@wayne.edu	CS
Manavendra Desai	manavendradesai@wayne.edu	ME

"I certify that the design and engineering of the Wayne State University Robotics Team has been significant and equivalent to what might be awarded credit in a capstone design course."

Faculty Advisors

Professor, Dr. Abhilash Pandya Department of Electrical & Computer Engineering

A F Pandya

apandya@wayne.edu

Assistant Professor, Dr. Marco Brocanelli, Department of Computer Science

[illegible]

brok@wayne.edu

Assistant Professor, Dr. Azad Ghaffari, Department of Mechanical Engineering

Azad Ghaffari

aghaffari@wayne.edu

Contents

1 Team organization

Warrior Robotics is a student-run organization at Wayne State University. It is a mix of undergraduate and graduate students with diverse skill sets. It's technically oriented sub-teams fall under *Hardware*, *Perception*, *Navigation* and *Simulation*. Warrior Robotics additionally comprises sub-teams for *Outreach* and *Onboarding* of students new to robotics.

The problem statement of the 2023 Intelligent Ground Vechicle Competition (IGVC) was broken down into appropriately designed capstone projects for the student members, each project being assigned a sub-team lead. A milestone chart was prepared for each project and their progress was tracked using weekly reports, end semester presentations and demonstrations of working code on Hardware. Faculty, team captains and sub-team leads had weekly scrum meetings with the student members to identify bottlenecks and focus the development effort to meet deadlines. We utilized the industry-standard MS Teams and its channels features to organize the teams data, meetings and fluid communications.

The organization of the student members of Warrior Robotics, and their approximate cumulative hours spent, are shown in Table 1. Names of sub-team leads and team captains are in boldface. The hours listed cover time spent on robot development and capstone examination requirements.

Name	Sub-team	Hours	Name	Sub-team	Hours
Laxmi Shankar	Hardware	20	Mohamed Safawi	Hardware	20
Matthew Stephens	Hardware	80	Brian McCoy	Hardware	40
Adrian Tlatelpa	Hardware	40	**Dimitri Van Well**	Hardware	120
Slav Ivaskiv	Perception	75	Vernard Wilson	Perception	90
Andrea Cuc	Perception	90	Asmahan Hussain	Perception	90
Luay Jawad	Perception	120	Kristina Karnick	Navigation	90
Hanna Bulinda	Navigation	90	Reem Rizk	Navigation	90
Blaine Onia	Navigation	90	Dan Pop	Navigation	75
Lloyd Brombach	Navigation	90	**Abhishek Shankar**	Navigation	120
Manavendra Desai	Navigation	120	Edrees Saeed	Simulation	90
Arun Jayaraman	Simulation	40	**Maysara Elazzazi**	Simulation	90

Table 1: Organization of Team Warrior Robotics.

2 Design and fabrication of robot hardware

2.1 Mechanical system

Building upon the 2022 IGVC design, we aimed to create a solution that would benefit each team within our organization. We started by understanding the specific goals and physical requirements of each sub-team. We conducted a thorough analysis to balance factors such as cost, time, and available resources even before developing concept mock-ups. By assigning error grades to different design aspects and their impact on the overall performance of the robot, we were able to refine the design. This process ultimately led to a final concept for Karina, our entry into the 2023 IGVC (Figure 3). Once the hardware sub-team received approval for the final concept mock-up, they began material selection and manufacturing by leveraging the strengths of each team member.

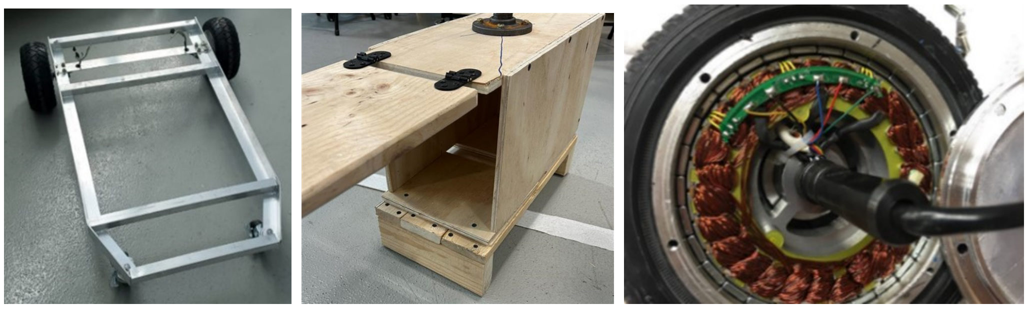

(a) (b) (c)

Figure 2: (a) The bare aluminum chassis. (b) The payload box. (c) Drivetrain motor internals and sensors.

The design for Karina was formulated with modularity and efficiency in mind. This sparked critical thinking for every component and its placement in order to achieve maximum efficiency. Building on last year's design, the focus was on improving performance and further increasing the purpose of each part, from its location and relevancy to the overall design. The chassis, the traction motors and drivetrain, and the electronics package remained unchanged (Figure 2).

Chassis: As shown in Figure 3, the floor is designated for *E-boxes*, boxes that house the electrical systems of the robot (in green). The payload box (in white) is raised to allow the boxes to be tucked underneath. Lastly, the chassis is reinforces with a quarter inch thick wooden plate.

Wheel placement: The wheel positions remained unchanged to retain the ability to ascend the ramp, and have adequate ground clearance.

Payload box: To enhance user convenience, the payload is designed to be side-filled, eliminating the need to stand in front of or reach behind the robot to lift it over the E-boxes. Additionally, the payload serves as a wire pass-through (Figure 3), facilitating the connection between the E-boxes and the sensors. The GPS is securely mounted to the payload box using a protective housing, attached through a 1/4" threaded hole beneath the GPS and the payload housing.

Sensor arm: For optimal sensor visibility, a steel bar (Figure 3) was utilized as an attachment point, with a warning light mounted on top. The sensor arm is reinforced using steel A-frames (Figure 3), accommodating a control module for the emergency stop (E-stop),

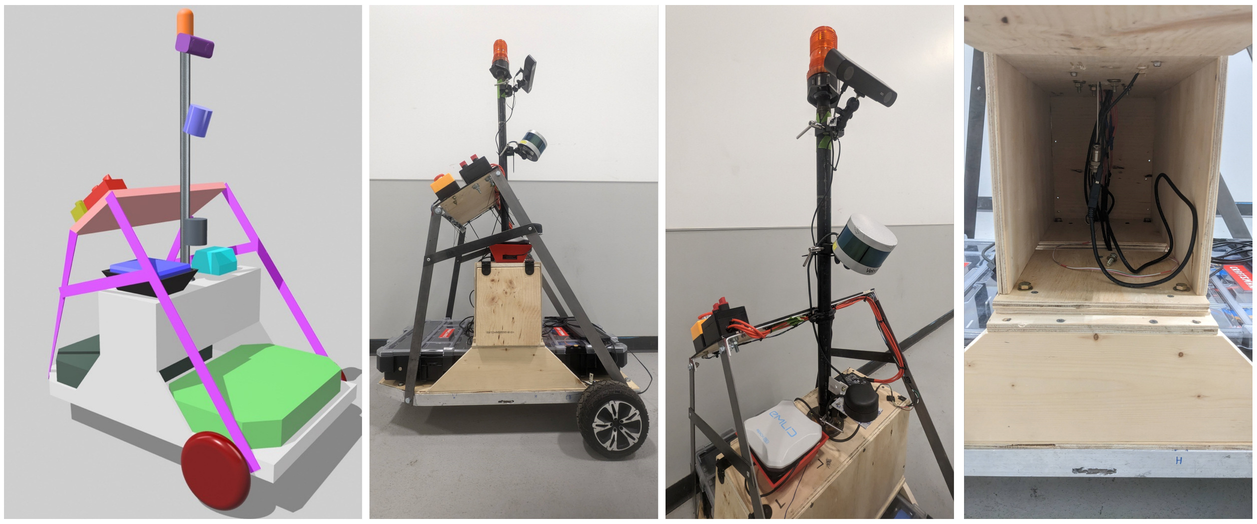


Figure 3: From left to right, (a) A rendered 3D model of Karina, (b) A-Frame bracing and control panel, (c) sensor placement and sensor arm, (d) cable runs through payload box

power switches, and a secondary monitor. This setup enables external use of the robot operating system while maintaining weatherproofing capabilities.

2.1.1 Innovation

Bracing: Steel bracing was added to the overall body of the vehicle, as shown in pink in Figure 3. It provides structure in terms of a roll cage for any unforeseen maneuvers the vehicle may face. Additionally, its geometric form is set to mimic a triangular A-frame for a shock absorbent connection to the sensor arm in order to prevent shaking of the sensors.The bracing further supports a control panel for the entire vehicle. This control panel has the option to hold a weatherproof monitor, the electrical power switches, and the vehicle e-stop button.

The Sensor Arm: The sensor arm, a steel bar, is securely attached to the vehicle chassis and the payload box, providing flexibility for sensor placement 3. Its threaded design allows for both firm attachment and easy removal. Each sensor's USB cables have disconnects to ensure a hot swap is an option. This feature proves useful for in-lab testing, repairs, or replacing specific sensor setups while keeping the robot operational. To prevent sensor displacement, friction-prevention bar clamps are utilized which allow for full removal if needed.

2.1.2 Safety and reliability considerations

For weatherproofing against the elements, we utilized IP66-rated enclosures to house our batteries, laptops, smaller sensors, and wires within our robotic system. Sensors such as the ZED2i Camera are not housed within the enclosures since they are rated IP66 and can survive adverse outdoor weather conditions. Lastly, we ensured that the wheel bearings of our brushless hub drive motors were sealed to ensure that they do not experience wear and tear due to outdoor conditions.

2.1.3 Failure modes and methods of resolution

The failure modes and methods of resolution identified are listed in Table 2. Additionally, we will carry toolkits and portable machine shops for on-field repairs.

Failure mode	Resolution
Damage to robot chassis/sensor arm/wheels	Carry spare wood, aluminum, hoverboard wheels, sensor arm
Sensor stops working	Carry a spare

Table 2: Failure modes and methods of resolution for Karina's mechanical systems.

2.2 Electrical system

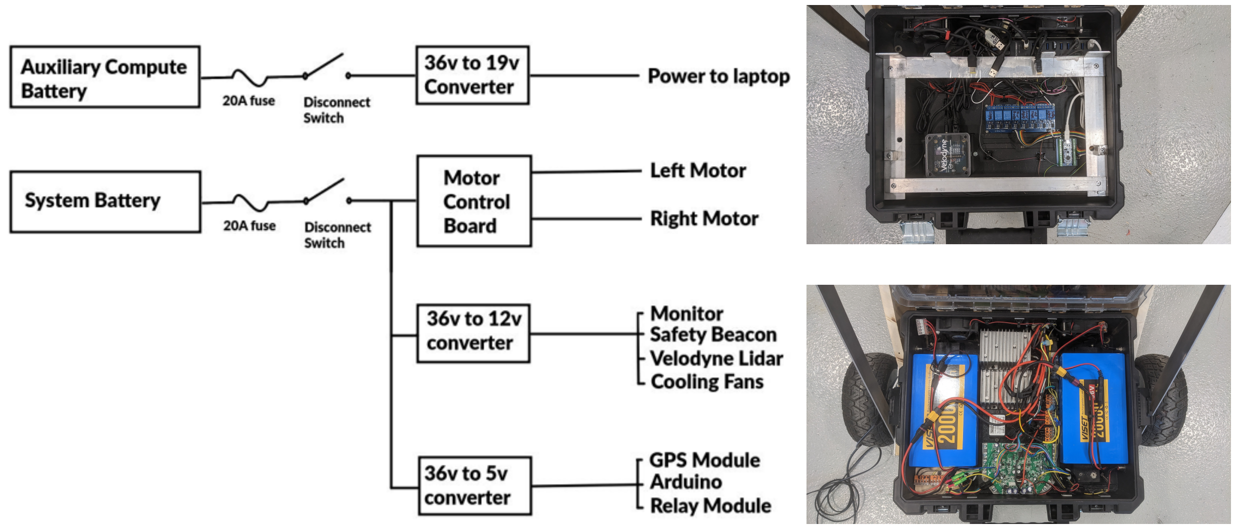


Figure 4: (a) Power diagrams for Karina are shown on the left. (b) The electrical box in the top right interfaces sensors with the laptop that runs the robot. The electrical box in the bottom right houses the batteries and microprocessors that power the hoverboard and the Velodyne 3D LiDAR on the robot.

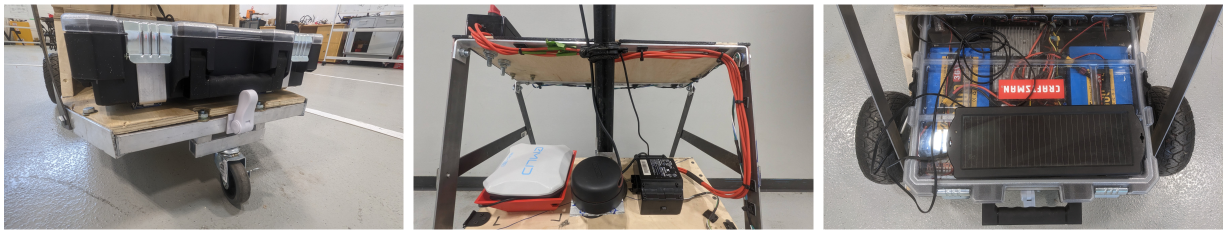
Some of our electronics components are unchanged in design from 2022, however much of the wiring and routing of cables has been redone. Karina uses two 36-volt, 20 amp-hour lithium-ion battery packs. The first battery pack provides 36 volts to the motors and its control board and 12-volt and 5-volt DC-DC converters that power the rest of the auxiliaries. The second battery is dedicated to powering the laptop via a 19-volt DC-DC converter. Both batteries can be easily changed with XT-60 quick connectors or charged in place with weather-resistant connectors mounted on the shell of the lower electronics compartment. The laptop does retain its factory internal battery, allowing for the auxiliary 36-volt battery to be changed without powering down. Each battery has its own disconnect switch.

The main battery requires 9.43 A (typical), providing around 64 minutes of runtime per charge when limiting discharge to 50% of the 20 AH capacity. The battery typically lasts for 2.65 hours, with a minimum duration of 1.8 hours, and has a max consumption of 230 watts (160 watts typical).

Karina uses a Lenovo Legion 5 gaming laptop with an eight-core, 3.2Ghz AMD CPU, an Nvidia GeForce 3050Ti GPU, and 32 GB of RAM for the main processing unit. Automatic power control to the motor driver board, Velodyne Lidar, cooling fans, and a safety beacon is provided by relays and an Arduino microcontroller interfaced to the main computer running a custom ROS node to manage it.

The motor driver board is a repurposed hoverboard control board that has been reprogrammed with firmware from an open-source ROS hoverboard project. Communication with the motor driver board, which also provides odometry feedback to the main computer, uses USB serial and an FTDI USB to TTL serial converter. The sensor suite consists of a Velodyne VLP-16 360-degree 3D LIDAR, a ZED2i stereo camera, an embedded IMU, hall effect sensors (embedded in the brushless DC motor/wheel unit), and a Reach RS+ RTK GPS module. The computer auxiliaries communicate via USB, except for the Velodyne, which uses ethernet.

2.2.1 Innovation



(a) (b) (c)

Figure 5: (a) e-box sliders and locks, (b) cable run to control panel, (c) solar panel attached next to PD e-box

Electrical and Payload Boxes: The E-boxes are positioned on self-clamping sliders, ensuring they stay in place while maintaining a streamlined design (Figure 7). This setup allows us to maintain the operability of key vehicle components. Cables are routed through passes within the payload box, which is equipped with a wire shield to protect the wires from any contact or potential damage from the payload. To support the formula of modularity, the I/O runs are detachable (besides major electrical power runs) and thus will allow for the e-boxes to be fully removable from the vehicle at any time. This promotes maintenance and upgrade capabilities that otherwise would be made difficult in an embedded modeled vehicle.

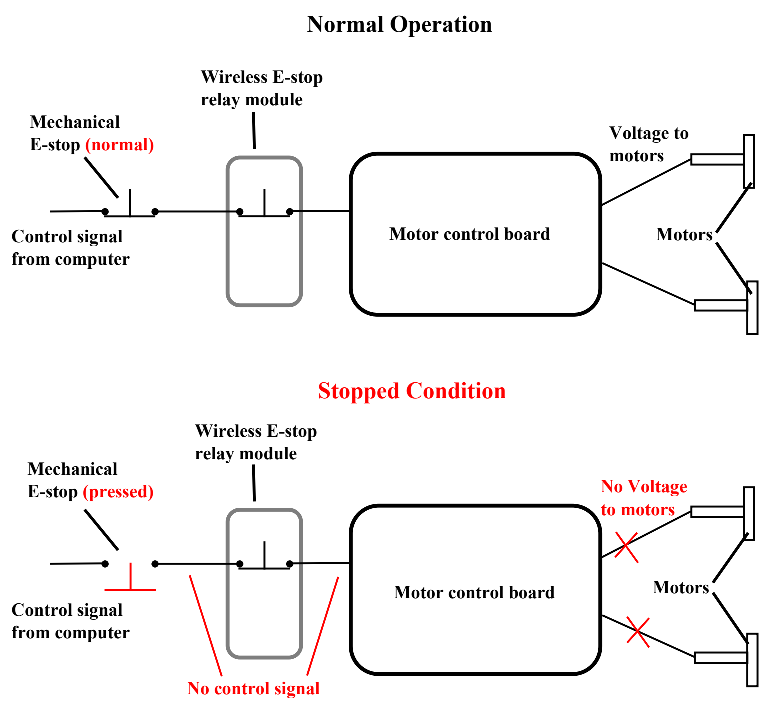

Normal Operation
Wireless E-stop relay module
Mechanical E-stop (normal)
Voltage to motors
Control signal from computer
Motor control board
Motors
Stopped Condition
Wireless E-stop relay module
Mechanical E-stop (pressed)
No Voltage to motors
Control signal from computer
Motor control board
Motors
No control signal

Figure 6: (Upper Left) Normal Operation for E-stop, (Lower Left) Stopped Condition for E-stop, (Right) Cable loom from logic E-box to rest of robot components

Solar Panel: A trickle-down solar panel was added to maintain the levels and longevity of the batteries located in the PD e-box (Figure 5c).

Temperature Sensor: A temperature sensor added to the E-box acts as a trigger for a relay to activate the cooling system if the E-box temperature rises above a threshold. This will help regulate the E-box cooling and is a safety check for the user.

2.2.2 Safety and reliability considerations

Our robot complies with IGVC Auto-Nav rules standard and has a mechanical and wireless E-stop (Figure 7). The two emergency stop systems work both independently and in tandem to ensure proper vehicle disengagement in the event of an anomaly. Both local and remote E-stops are wired in series, so either can interrupt the control signal to the motor control board. Without a control signal, the control board powers down the motors.

2.2.3 Failure modes and methods of resolution

The failure modes for the electrical system and methods of resolution identified are listed in Table 3.

Failure mode	Resolution
Hoverboard Arduino malfunctions	Carry spare
Battery puffing/explodes	Carry baking soda and PPE
Overheating due to failed temp sensor	Carry spare
Control panel malfunctions	Carry spare
E-stop/Light malfunctions	Carry spare
Code lost/corrupted	Pull from GitHub

Table 3: Failure modes and methods of resolution for Karina's electrical systems.

2.3 Materials and cost breakdown

A breakdown of the materials and costs to build Karina is given in Table 4 on page 8.

3 Architecture for robot software

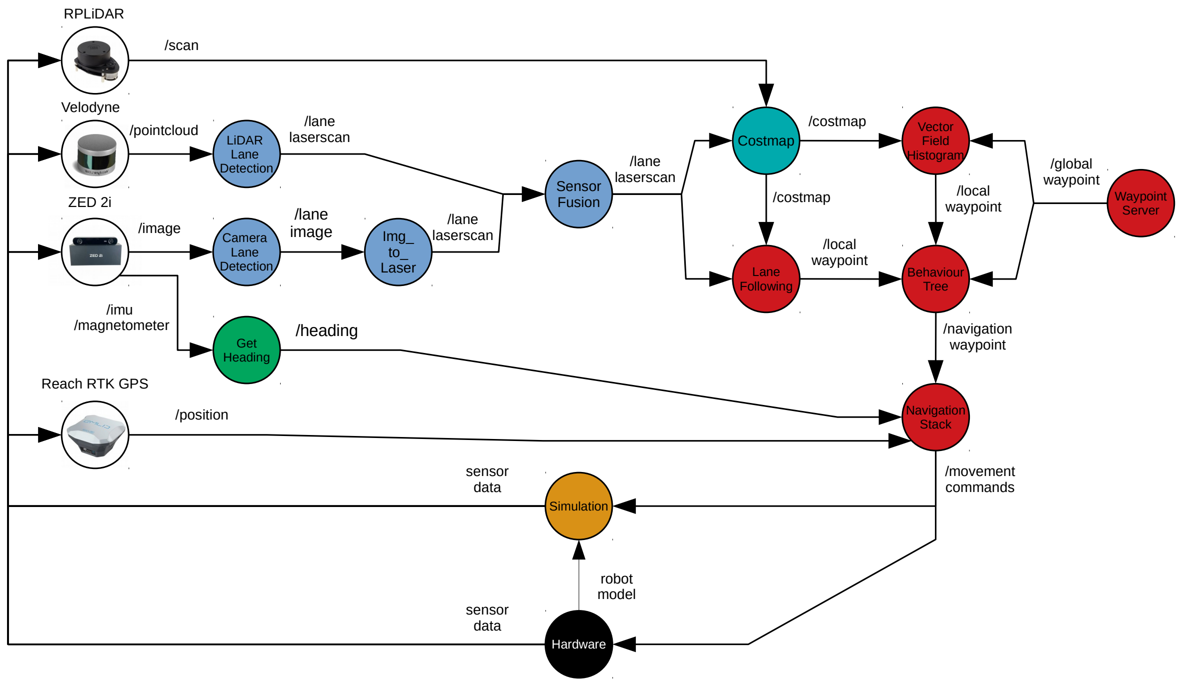


Figure 7: Pipeline for our software architecture.

Package	Item	Quantity	Unit cost (in $)	Total cost (in $)
Sensors	Velodyne Lidar	1	1350.00	1350.00
	R2 Lidar	1	230.00	230.00
	Emlid GPS	1	900.00	900.00
	Zed2i Camera	1	499.00	499.00
Motor	Drive Train/Board	1	300.00	300.00
Power	Lithium 36V Batteries	2	265.99	531.98
	5V and 12V SD-Transformer	3	29.89	89.67
	USB Hub	1	47.99	47.99
	E-stops	2	10.99	21.98
Software	Lenovo Laptop	1	1000.00	1000.00
	Secondary Monitor	1	250.00	250.00
Fabrication	Steel/Aluminum	7	35.00	245.00
	Wood	3	21.08	63.24
	Mounting Parts	3	12.90	38.70
	Electrical Housing	2	32.98	65.96
	Caster	1	12.00	12.00
Electrical	Wiring	20	30.00	600.00
	Arduino Nano	1	13.70	13.70
	Relay Board	1	12.00	12.00
Total				6271.92

Table 4: A cost breakdown for the Karina robot.

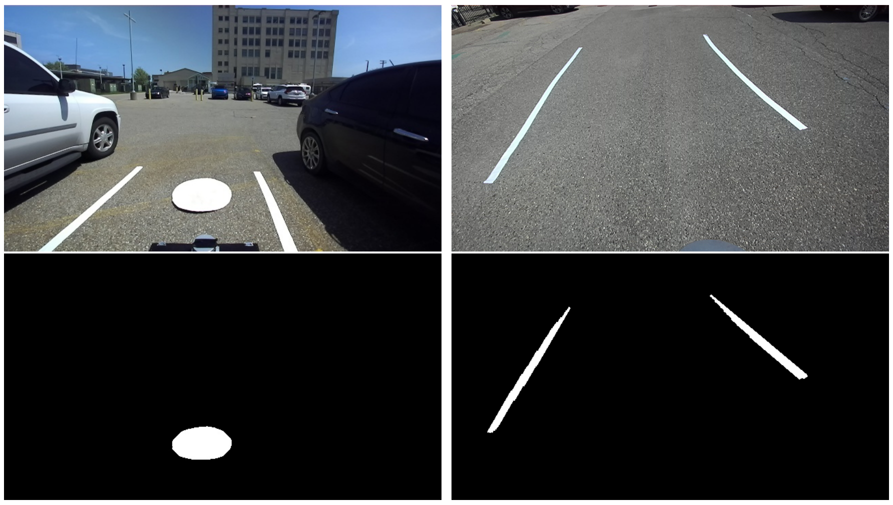
3.1 Perception

3.1.1 Lane and pothole detection

Two methods were evaluated for lane detection. The primary way is a HSV based filter that isolates white pixels with intensity greater than a threshold to detect lane lines. The results are refined by tuning the HSV parameters and by having additional filters and line fitting using Hough lines. This approach shows promising results and is computationally efficient but the performance is heavily dependent on the lighting conditions.

The other approach employs the YOLOP Convolutional Neural Network which was developed by Wu Dong et al. The model consists of one encoder followed by three decoder heads that each detect lanes, driveable area, and objects respectively. The model was trained on the BDD 100k dataset released by UC Berkeley and shows great performance, but the downside is that it requires a good GPU and takes up some memory.

Pothole detection is similar to the HSV approach except a filter is placed to extract circular or elliptical shapes from the image. Although the performance is subject to lighting conditions, several tests outdoors have shown that it can be reliable with little tuning.
The image-to-laser pipeline takes a binary image of detected lanes and converts it into an



(a) Pothole detection (b) Lane detection

Figure 8: A series of snapshots showing our lane and pothole detection results with the Zed2i camera.

accurate laserscan that can be plotted on the costmap. The pipeline has multiple steps: transformation into birds-eye, gap closure, and conversion to laserscan. The input binary image undergoes a perspective transformation to turn it into a top-down view of the lanes. This is done by using the focal length, height above the ground, and the roll, pitch, yaw angles of the mounted camera relative to the upper horizontal. After the transformation, any gaps in the lanes (due to cones getting in the way or bad detection) are closed using a Probabilistic Hough Transform. Once the gaps are closed, the image is converted into a laserscan by calculating distances and using pixel-to-meter conversions.

3.1.2 Obstacle detection

We utilized a 2D LiDAR with a 360-degree field-of-view for obstacle detection. The LiDAR was mounted parallel to the plane of the robot on the sensor arm and mounted at a height such that it could detect obstacles while not detecting the ramp that comes later in the course. The output from this LiDAR is sent to the costmap to be plotted as obstacles. To prevent the robot itself from being plotted as an obstacle, we used a box filter that encompasses the robot chassis from the `laser_filters` package to filter the raw laserscan. This is more effective than limiting the LiDARs FOV because it still allows the detection of things immediately in the robots vicinity in a 360 FOV. Since only the filtered laserscan is passed onto the costmap, the robot is not plotted as an obstacle and does not cause problems in navigation. In addition to the 2D LiDAR, the modularity of the sensor arm allows us to replace it with a more robust 3D LiDAR if required.

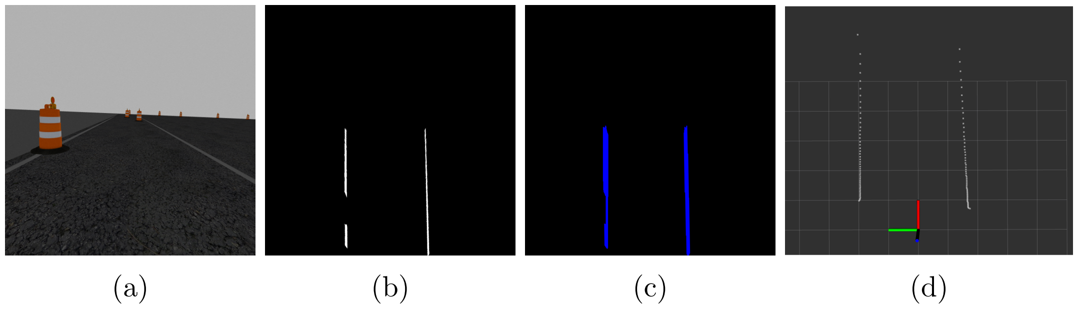


Figure 9: A series of snapshots showing our implementation of converting lanes into laserscans. (a) Raw camera image. (b) Birds-eye transformation. (c) Gap closure. (d) Laserscan conversion.

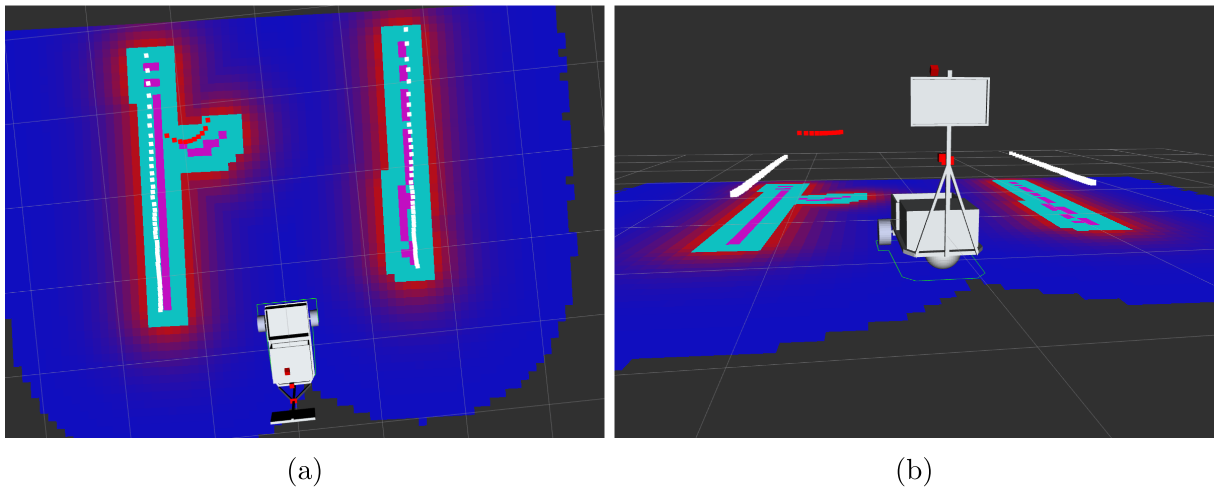


Figure 10: A pair of snapshots showing the implementation of the bi-layered cost map.

3.2 Waypoint selection

3.2.1 Cost map

The cost map is a 2D occupancy grid of the data and inflates costs based on the inflation radius. This propagates cost values outwards from each occupied cell out to a user-specified inflation radius as shown in Figure 10a. For navigation, an evolving cost map is used which updates the obstacles and the cost as the robot moves. The rolling_window parameter keeps the robot in the center of the costmap as it moves throughout the world, dropping obstacle information further than 4m from the map.

The `costmap_2d` package from the `ros-planning/navigation` software stack was used to create a rolling-window map of the immediate surroundings where the robot is at the center of that map. LiDAR and a laser scan messages generated from lane and pothole detection data are used to populate the costmap with obstacles. These are incorporated as two different layers in the costmap as shown in figure 10b. The costs are then used to calculate goals for the robot which are sent to the navigation stack in `base_link` frame.

3.2.2 Lane-following

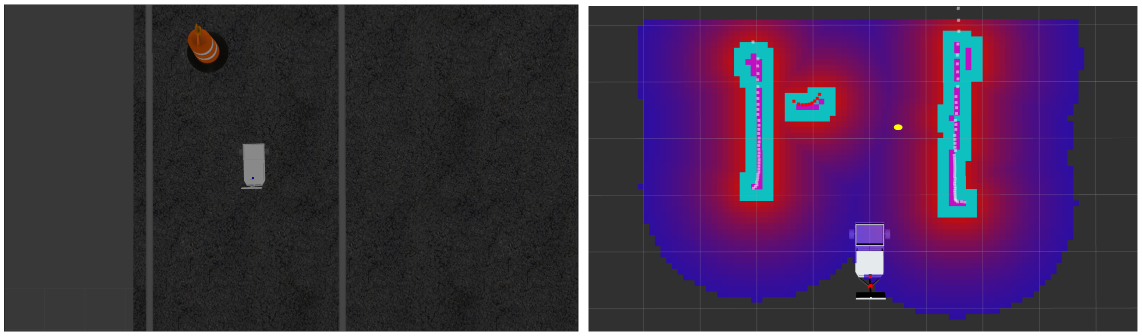
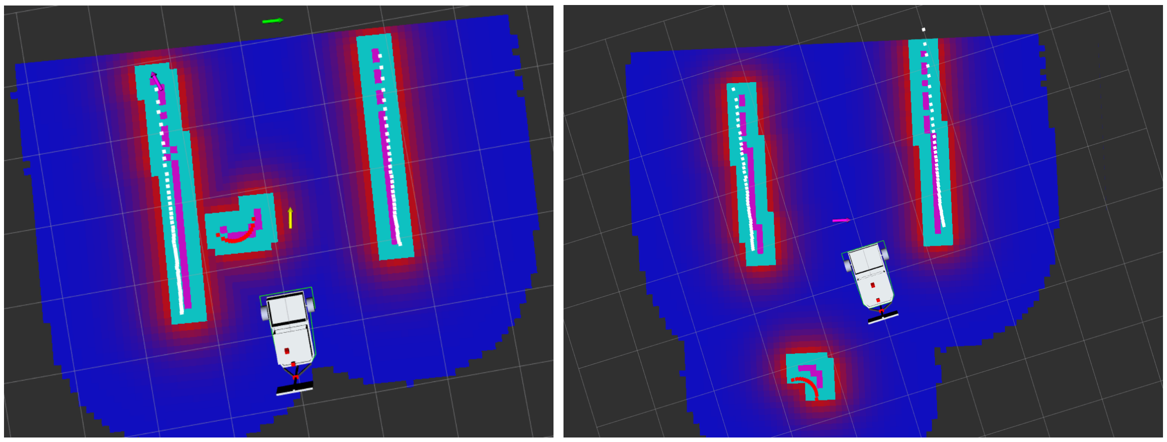


Figure 11: A pair of snapshots showing the implementation of the `lane-following` algorithm for selecting a waypoint in obstacle-free space.



(a) (b)

Figure 12: A pair of snapshots showing the implementation of the `vector-field-histogram` algorithm for selecting a waypoint in obstacle-free space.

The objective of the lane following program is to generate a local waypoint that remains within the lanes while avoiding obstacles. To obtain the local waypoint, lane scan data is utilized to identify whether the robot is following the right or left lane. The decision to avoid selecting points above 2.5 meters helps to ensure that we are using data closer to the robot for increased accuracy. To ensure the safety of the robot, a buffer of 1 meter is added to the X coordinate to keep it away from the lane. To verify that the path to the waypoint is unobstructed, the local costmap is analyzed. In case of an identified obstruction or if the cost of the path to the waypoint is 30 or greater, the lane following program will automatically adjust the waypoint's placement to a more optimal location.

3.2.3 Vector Field Histogram

Vector Field Histogram (VFH) is an algorithm used to select intermediate waypoints for the robot. The local cost map is used to select an optimized waypoint where the obstacle density is low. Figure 12a shows the waypoints generated (yellow arrow). As the robot moves towards the waypoint, it avoids obstacles while VFH continuously updates the waypoint such that it leads toward the global goal as given by the GPS points (green arrow).

The lanes are used to bound the VFH waypoint and the costmap ensures that lanes are seen as obstacles. A preliminary waypoint (shown as purple arrow) as shown in figure 12b is selected such that it is further away from the lanes but closer to the global waypoint to act as a guide for the VFH. The VFH then provides a waypoint (yellow arrow) between the global and preliminary waypoints. Additionally, the waypoints are chosen to be 1.5m away from the robot which is 1.5 times the body length.

3.3 Navigation

3.3.1 Extended Kalman Filter based localization

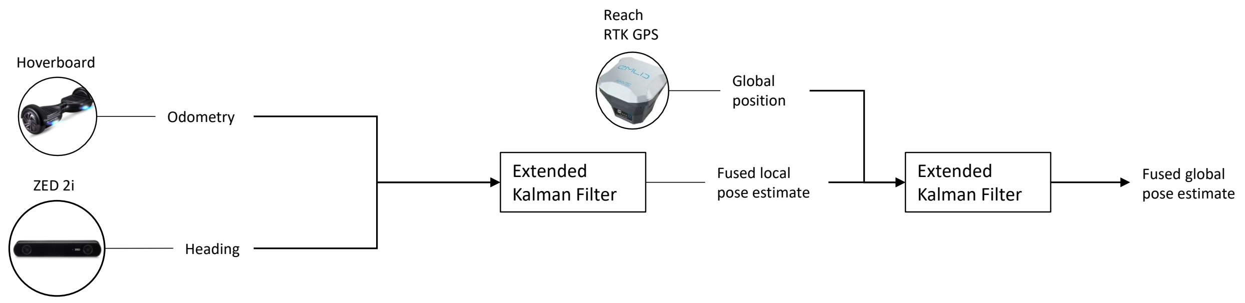


Figure 13: A flowchart describing how Karina localizes herself outdoors.

The navigation stack requires accurate pose estimation to function properly. Having a single source of position and orientation data is ill advised due to sensor noise and other random errors. Wheels are prone to slipping or sliding, magnetometers are affected by nearby magnetic fields, and the accuracy of GPS modules can be disrupted by ionic interference. A fusion of multiple data sources is the best way to ensure accurate pose estimates at all times.

Due to the sensors publishing at different rates, we use a multi-step approach utilizing the Extended Kalman Filter from the `robot-localization` ROS package. First, the XY position, yaw orientation, and yaw velocity from the hoverboard are fused with the magnetometer-corrected yaw orientation and velocity from the ZED 2i 9-DOF IMU which gives a locally accurate pose estimate of the robot, but primarily keeps the orientation steady. This output is then fused with the position information given by the GPS module to obtain a globally accurate pose estimate and keep the XY position from drifting. This allows accurate pose information to be available at all times. If the globally accurate pose is too old, the navigation stack will use the locally accurate pose from the first fusion, which will then be corrected by the second fusion when possible.

3.3.2 Innovation: Motion control

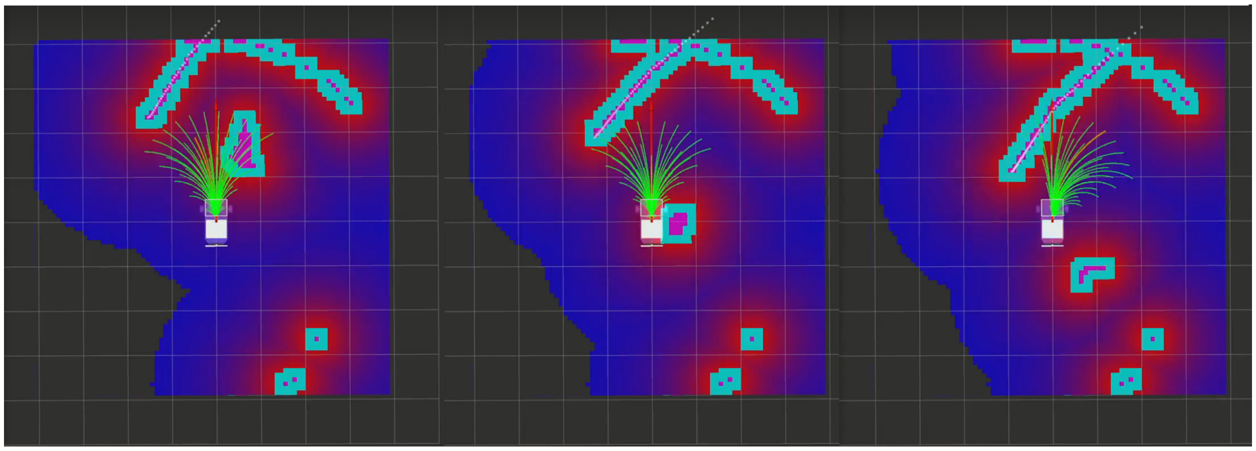


Figure 14: Snapshots of DWA planner in the presence of lanes and obstacles

The Karina Navigation Stack is designed with a bottom-up approach to achieve modularity and simplicity. The navigation system is broken into two primary tasks - trajectory and velocity control. Trajectory control is accomplished through a combination of local waypoint selection and a Dynamic Window Approach (DWA) controller. Velocity control is achieved by an agent that monitors the costmap and sensor data to adjust velocity and provide recovery behavior.

The trajectory controller, known as the Wayfinder, takes in generated waypoints from Vector Field Histogram (VFH) and Lane Follow to construct the plan, which is then sent to a controller that publishes velocity commands to the drive motors based on the plan received. The velocity controller, known as the Agent, is a single-stage process that obtains information about Karina's local environment from the costmap and sensor readings. These readings are checked to ensure no imminent collisions. If a collision is detected, the robot will reverse backwards away from the collision.

The navigation system utilizes two types of controllers - Dynamic Window Approach (DWA) and Proportional-Integral-Derivative (PID) methods. DWA (shown in 14) is useful for reactive navigation while the PID controller allows for more stable navigation, following the given path very closely. Thus, we utilize the PID controller when we are able to generate a path, and DWA when the generated path is inadmissible.If there is no admissible path generated, the Wayfinder resorts to a subsumptive approach in which the Lane Following waypoint supersedes the VFH waypoint. In the event that no waypoint is available, the robot is given a goal directly in front of it.

3.4 Simulation

To optimize testing and development while the hardware was being finished and to mitigate the impact of weather, we created a comprehensive simulation of our robot environment

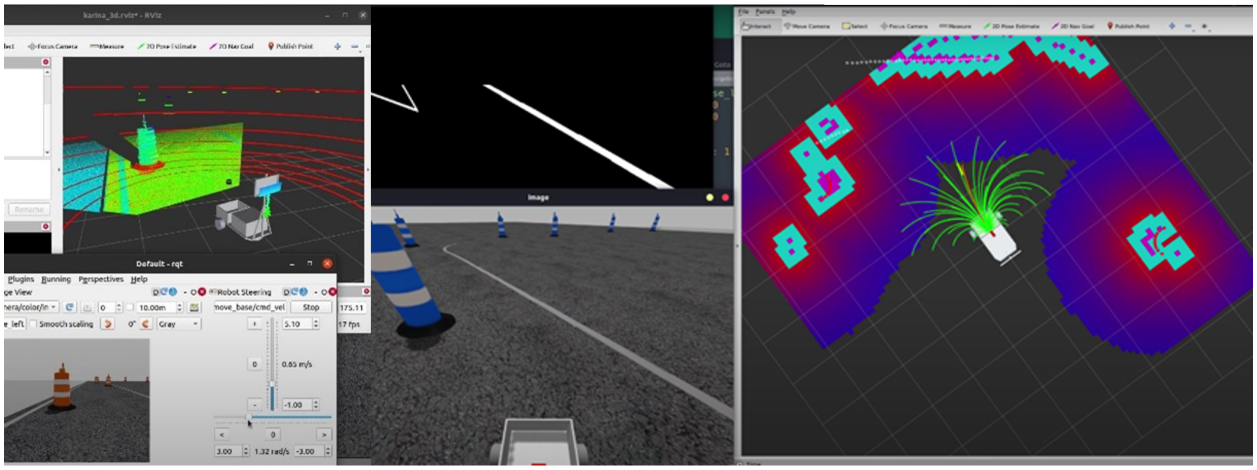


Figure 15: Snapshots of the simulation stack

and completes an entire course in simulation! Figure 15 shows our robot model with all the relevant sensors, a simulated path with lanes and obstacles and GPS way points. The simulation runs our entire code stack, allowing for rapid prototyping and testing.

The simulation goals were to integrate it into our workflow and robot testing, create diverse maps with limitless scenarios, and develop a detailed and precise robot description. Our sub-team focused on three key areas to plan the simulation: map building, robot creation, and integration/optimization. For maps, the simulation offers the ability to model diverse environments and conditions. We designed a simulated robot with variants of all the sensors used on the actual hardware, providing data streams through the same ROS topics. We used Gazebo to create realistic maps for testing. These maps were accurately scaled using satellite imagery of an IGVC course. Various lane designs were drawn on the scaled graph and converted into a digital format. These designs allowed us to optimize testing of sub-components and the entire system on a full track.

3.5 Safety and reliability considerations

Safety and reliability considerations for the algorithms and sensors in our software stack are provided in Table 5 and 6 below.

Algorithm	Reliability concern	Mitigation method
`lane-detection` `pothole-detection`	HSV thresholding is susceptible to change in lighting conditions (e.g., shadows, sunlight).	Fine tune and store HSV parameters for different lighting conditions.
`image-to-laser`	Incorrect camera configuration such as mounting angle.	Fine tune parameters and ensure the camera mount is steady.
Cost map	Bad transform data plots obstacles and lanes incorrectly.	Careful validation and testing in controlled environments.
`ekf_localization`	No new data is being provided to the `ekf` node.	Use old data to make a prediction at current timestep until new data is received.

Table 5: Safety and reliability considerations for the algorithms in the software stack.

Sensor	Reliability concern	Mitigation method
ZED 2i	Dirty lenses.	Clean carefully with microfibre cloth.
	Metal objects interfere with IMU/magnetometer.	Calibrate magnetometer/IMU to compensate resultant bias for current robot.
Velodyne 3D LiDAR	Dirty glass.	Clean carefully with microfibre cloth.
2D LiDAR	Water/dust gets into the sensor.	Weatherproof the LiDAR.
	Inaccurate readings when facing bright sunlight.	Exhaustive outdoor tests and parameter tuning.
Hoverboard	Incorrect odometry.	Fine tune the parameters in controlled conditions.
GPS	Ionic interference from the robot or ZED magnetometer.	Shield power and signal cables where necessary.

Table 6: Safety and reliability considerations for the sensors in the software stack.

3.6 Failure modes and methods of resolution

Failure modes and methods of resolution for our software stack is provided in Table 7 below.

Failure level	Failure mode	Resolution
`lane-detection`	Fails to detect lanes in quickly changing lighting conditions	Use YOLOP or 3D LiDAR lane detection.
`image-to-laser`	Lane-scan is inaccurate due to change in camera position and/or orientation	Publish `ERROR`.
`lane-following`	Selects unreliable waypoint in absence of lanes or when the robot faces lanes head-on	Publish a low confidence metric and query `vector-field-histogram`.
`vector-field-histogram`	Fails if no lane data is provided	Publish low confidence metric. Initiate recovery behavior.
`dynamic-window-approach`	Determines infeasible path	Initiate recovery behavior. Robot backs up.

Table 7: Failure modes and methods of resolution for the software stack.

Damage of sensors due to weather-proofing fails, electrical system surges, or being hit by fast moving or heavy object, will be resolved by carrying spare units for each sensor.

4 Initial tests and analysis

Karina was tested exhaustively in simulation and outdoors. Table 8, on page 17, lists the tests that proved informative in shaping the design of the robot.

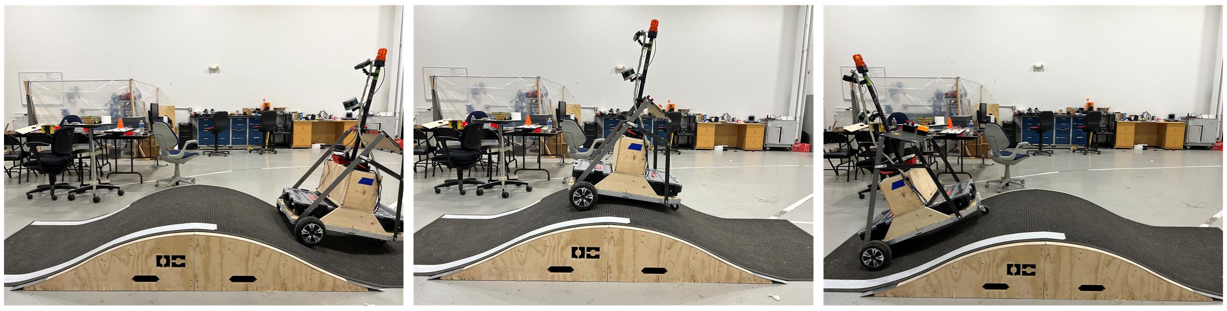


Figure 16: A series of snapshots showing Karina climb and descend a 25° ramp

Test	Result
Speed and reaction time	A maximum time of .1 seconds is required for the motor controller and hoverboard driver to respond. The total time to respond to new obstacles incorporated in the cost map is a maximum of .25 seconds.
Ramp climbing ability	As shown in Figure, the robot (without payload), can climb a ramp of 25° inclination which is beyond the 15° inclination specified
Battery life	Over the course of outdoor and indoor testing, the two 36 volt, 20 ampere-hour Li ion battery packs, consistently provided 60 minutes of runtime per charge.
`lane-detection`	Outdoor testing set the Zed2i pitch down angle at 45° to see 3 meters ahead of the robot.
`image-to-laser`	Outdoor testing and parameter tuning enabled matching of laser scans for obstacles and lanes in birds-eye view, with ground truth location of the obstacles and lanes.
Obstacle detection range	Testing in simulation and in the outdoors set the ray-tracing and obstacle detection range for the 2D LiDAR to 4 meters.
Cost map	Exhaustive testing in simulation set the cost map to be an 8m $\times$ 8m grid of 0.1m resolution and an inflation radius of 2m.
Odometry	Hoverboard encoder parameters were tested and tuned on an asphalt surface to achieve less than 3 cm inaccuracy for every 100 cm traveled.
GPS	Outdoor tests revealed electromagnetic interference from the Zed2i magnetometer. With the Zed2i power cable shielded, the GPS reports positions accurate within 10 cm.
Recovery behavior	Testing in simulation, as shown in Figure , showed that the robot is cable of stopping, backing up, and moving forward, when it inadvertently gets too close to an obstacle.

Table 8: Initial tests conducted on the Karina robot.