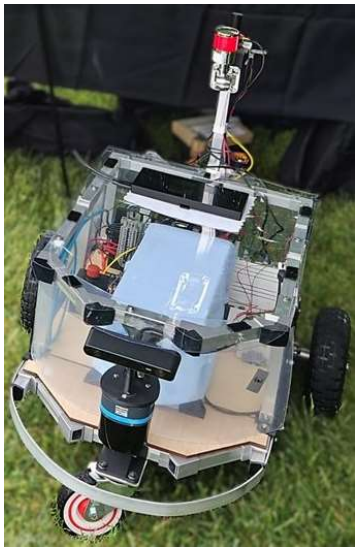
Virginia Polytechnic Institute and State University

VT CRO – AutoNav

Shogi



Team Member	Email	Phone
Cassie Freedlander (Team Captain)	cfreed@vt.edu	(571) 505-7001
McKenna Olson	mckenna27@vt.edu	(339) 368-3747
Jose Arellano	josea@vt.edu	(540) 746-2829
Nathan Fikes	nfikes@vt.edu	(571) 442-2404
Sreeauditya Motukuri	smotuku@vt.edu	(571) 428-3167
Cole Wendrowski	colew@vt.edu	(208) 954-2554
Tamir Zharmagambetov	tamirz@vt.edu	(347) 291-3365
Jacob Sigeti	jsigeti@vt.edu	(716) 395-8054
Manan Sharma	manansharma@vt.edu	(314) 920-7616

Advisors/Mentors

- Matt Nowinski, PhD, Collegiate Associate Professor at Virginia Tech
- Robin Ott, Professor of Practice and Interdisciplinary Programs Administrator at Virginia Tech

Statement of Integrity

I hereby state that the engineering design and development of Shogi was completed by the current student team and is equivalent to a senior design capstone course at Virginia Tech.

Matt Nowinski, mcnowins@vt.edu, (540) 231-9274

Table of Contents

1. Introduction

The Competitive Robotics Organization (CRO) at Virginia Tech is excited to introduce Shogi to the AutoNav Challenge of the 2026 Intelligent Ground Vehicle Competition (IGVC).

The team consists of 9 students from 4 different engineering disciplines:

- **Mechanical Engineering**: Cassie Freedlander, McKenna Olson, Jose Arellano, Nathan Fikes, Sreeauditya Motukuri
- **Computer Engineering**: Tamir Zharmagambetov, Manan Sharma
- **Computer Science**: Cole Wendrowski
- **Electrical Engineering**: Jacob Sigeti

The team is split into 3 sub-teams, each focusing on a different area of development. These sub-teams are mechanical, electrical, and software. The mechanical sub-team is responsible for designing and building the vehicle and test track equipment. The electrical sub-team is responsible for ensuring proper power distribution and wiring configurations. Finally, the software sub-team is responsible for developing and testing the algorithms used by the robot to navigate the course.

2. System and Subsystem Requirements

The team used a systems engineering approach to identify the requirements that guided the design of Shogi. Requirements were developed by reviewing IGVC competition rules, analyzing limitations from Virginia Tech's 2025 team, and identifying the functional needs of the vehicle for competition. These high-level requirements were then broken down into the subsystem requirements for the mechanical, electrical, perception, and controls systems. Each requirement was tied to a measurable metric so the team could verify whether the design met the intended objective. A traceability matrix was created to connect each requirement to its source and corresponding validation method.

Table 1: System Requirements & Validation

ID	Requirement	Source	Subsystem	Validation Method	Target Value
M1	Chassis must support full vehicle weight with an adequate factor of safety (FoS).	Good Practice	Mechanical	Finite-Element Analysis	200 lb. with FoS >= 1
M2	Vehicle must remain within competition size constraints.	IGVC Rules	Mechanical	Physical Measurement	Length: 3-7 ft Width: 2-4 ft Height: < 6 ft
M3	Vehicle must be able to operate in light rain.	IGVC Rules	Mechanical	Spray Water on Vehicle	No water leakage or damage.
M4	Vehicle is lighter than previous year's vehicle.	Team Goal	Mechanical	Weigh Vehicle	Vehicle < 120 lb.
S1	Vehicle must stop immediately when emergency stop is activated.	IGVC Rules	Safety	Functional Testing	Come to complete stop within 1 second of button press.

S2	Vehicle speed must remain below 5 mph.	IGVC Rules	Controls	Speed Testing	Speed < 5 mph
S3	Vehicle must indicate whether it is in autonomous or manual mode.	IGVC Rules	Safety	Visual Testing	Manual Mode: Solid Safety Light Autonomous Mode: Blinking Safety Light
E1	Electrical components must share a common grounding reference.	Mentor Advice, Good Practice	Electrical	Continuity Testing	All components connected to common ground.
E2	Battery system must power the robot for an entire competition run.	Good Practice	Electrical	Runtime Testing	> 5 hours runtime under average 1.5A
P1	Vehicle must identify various obstacle types with enough distance to react safely.	Good Practice	Perception	Field Testing	> 2.5m reaction distance
P2	Vehicle must have an onboard GPU to accommodate heavy computer vision.	Required	Perception	Field Testing	Up to 1024 CUDA cores.
D1	Vehicle must generate a path that avoids detected obstacles.	Good Practice	Driving Logic	Course Testing	Generated path must be no closer than 0.70 m.
D2	Vehicle must update steering directions in real time.	Good Practice	Driving Logic	Response Time Testing	Robot must react to dynamic obstacles within 0.5 seconds.
D3	Vehicle must be able to navigate using GPS waypoints as goal points.	Required	Driving Logic	Repeated Testing	Convergence from all orientations and directions under 1 m.
K1	Vehicle must complete the course within a set time.	IGVC Rules	Performance	Full-course Testing	Time < 6 minutes
K2	Vehicle must complete runs without human intervention.	IGVC Rules	Performance	Repeated Testing	Complete mock courses entirely in autonomous mode.

3. Mechanical Design

The mechanical team prioritized a design for Shogi with the goal of ensuring quick and easy construction and accessibility for maintenance, utilizing simple, available market parts, while also maintaining a lightweight vehicle with good maneuverability. Figure 1 shows a CAD model of Shogi. It meets the size requirements for the competition, M2, measuring to be roughly 43 inches long, 31 inches wide, and 31.5 inches tall.

Figure 1: Isometric View of Shogi CAD

3.1. Frame

The robot's frame, shown in Figure 2, was built from commercial-off-the-shelf (COTS), 1-inch aluminum square tubes joined with steel brackets. This was to allow a high-strength skeleton while maintaining a lower weight. A rectangular frame design was selected to improve load distribution, and an upper frame was added to increase rigidity, provide mounting points for components, and allow the vehicle to be flipped upside down for easy drivetrain maintenance.

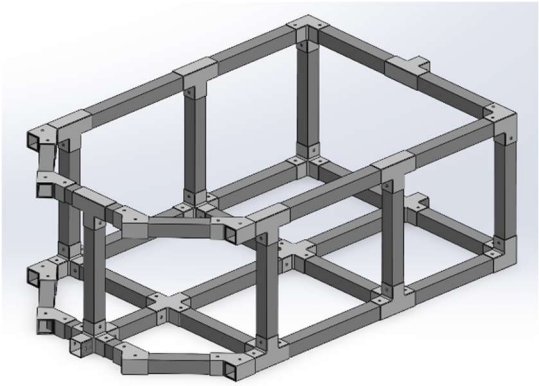


Figure 2: Shogi Frame CAD

To verify that the designed chassis could safely withstand expected loading conditions, finite element analysis (FEA) was conducted on the frame. The simulation applied a 200 lb. load to the bottom frame and a 100 lb. load to the top frame, representing both operational loading and maintenance scenarios, such as the vehicle being flipped upside down to work on the drivetrain. Results showed that the frame's highest-stress points maintained a FoS of 1.8, which successfully meets requirement M1 and confirms that the designed structure can safely support all expected loads. The FEA is shown in Figure 3 below. The final weight of the vehicle, including all electronic

components, the frame, and the powertrain, is 80.2 lb. This aligns with requirement M4, and in fact reduces weight by approximately 25% compared to last year's vehicle.

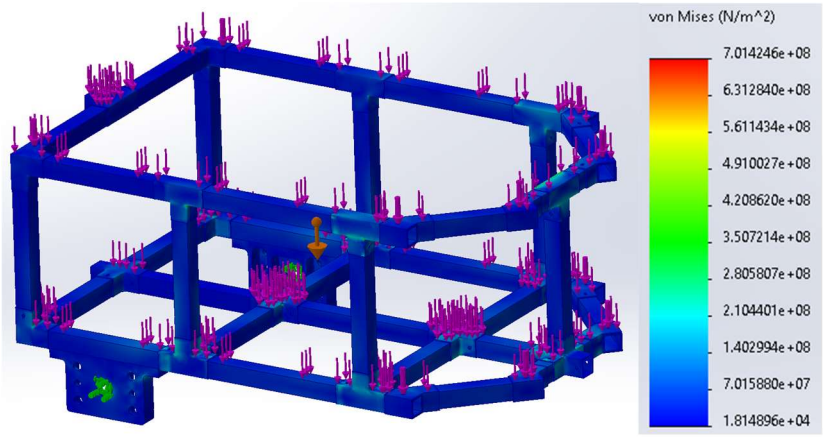


Figure 3: Shogi Frame FEA

3.2. Drivetrain

The driving system consists of a rear differential drivetrain with a caster wheel at the front. The rear drivetrain has two separate electric motors (MMP BL58-487C-24V GRA52-014), each anchored to the frame with individual mounting brackets. The motors connect to a single motor controller (Roboteq FBLG2360T), mounted to the underside of the frame with aluminum L brackets, as shown in Figure 4. The wheel and motor combination were chosen because they met requirement S2 and provided the necessary power and torque needed to move the vehicle.

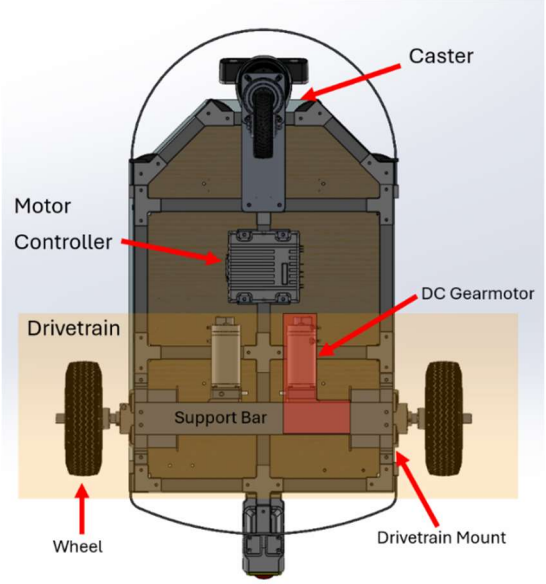


Figure 4: Shogi Bottom View – Labeled Drivetrain

3.3. Base Plate and Shell

In order to maintain a low vehicle weight, incorporate water resistance, and have a strong base plate for electronics to be mounted, the team carefully chose the materials to meet these specifications. For the base plate of the robot, where the payload and most of the electronics are located, the team opted to use lightweight plywood plates, mounted to the frame. This design, along with the frame skeleton, creates a strong base for all accessories and distributes the load across the bottom. The base plate design was cut into 3 sections to make manufacturing and

assembly easier. To meet the water resistance requirement, M3, the team decided to fully enclose the vehicle. Water is prevented from entering and damaging the vehicle with acrylic sheets placed across the top and sides of the vehicle. The sheets are sealed with 3D printed TPU strips to ensure no water leaks in without adding excessive weight to the vehicle. This was tested by spraying water on the vehicle to simulate light rain to make sure the requirement is satisfied. Figure 5 shows Shogi with the acrylic sheets added.

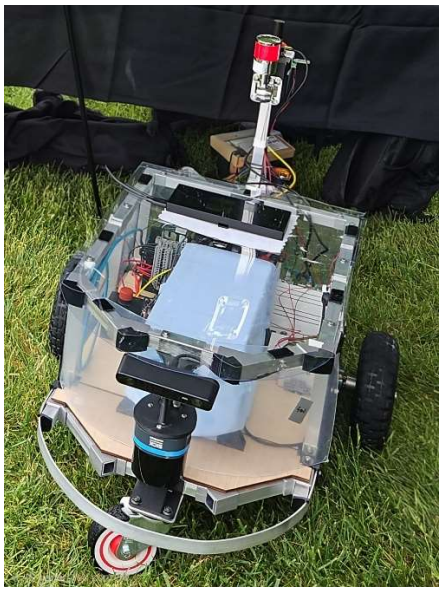


Figure 5: Shogi with Acrylic Enclosure

4. Safety

Multiple features were incorporated on Shogi to improve safety to others and to the components in the vehicle while operating on the course. For instance, two emergency stop (E-Stop) buttons can be used to stop the vehicle. The first is located on the vehicle and mechanically cuts power to the motor controller, activating safety brake switches on the motors and bringing the robot to a complete stop. The second is wireless and uses radio commands to electrically and mechanically carry out the same task. This satisfies requirement S1, as this system will allow the vehicle to come to a complete stop in 1 second or less.

Additionally, front and rear aluminum bumpers, shown in Figure 5, were installed to absorb most of the energy during any potential impacts. 3D-printed TPU corner pads were also added to prevent injury to persons or damage to nearby surroundings caused by the frame's previously sharp corners. A safety light was installed on the vehicle's mast to ensure proper visibility, and it was programmed and tested to be solid when the vehicle is in manual mode and blinking when the vehicle is in autonomous mode, satisfying requirement S3.

Shogi is also speed-limited via software in autonomous mode to ensure it does not reach an unsafe speed. It is set to a maximum speed of 1.23 mph during testing, and this can be increased to a maximum speed of 4 mph during full course testing and movement. This is to ensure the robot does not exceed 5 mph when going down an incline such as the ramp and meets requirement S2.

The team has established a safe method to charge the battery that is strictly adhered to and ensures the safety of the components on the robot and surrounding objects and people. The battery must be completely disconnected from the robot with the negative terminal first, charged from a grounded outlet under constant supervision, and reinstalled positive terminal first.

While the robot is being transported, at least two team members must lift the vehicle to prevent injury. The frame's design has many grip points which makes this easier and safer. The mast can be removed and stowed, and custom 3D-printed wheel chocks are used to ensure Shogi does not roll around during transport. These wheel chocks are also used while Shogi is parked, preventing it from rolling down inclines in case of brake failure.

5. Electrical Design

5.1. Electrical Architecture

The electrical system implemented this year has been optimized for the team's new robot design. These optimizations include removal of the 5V DC-DC buck converter, addition of a centralized ground net, and reorganization of the terminal block layout.

The robot is powered by a 24V LiFePO4 battery, which supplies the primary 20-29 V rail of high-power components including the motor controller and DC-DC converters. Regulated 12V and 19V rails support low-voltage components such as the Jetson Orin Nano, Arduino, and SICK LiDAR. The system was designed to support full autonomous testing sessions without requiring recharge between runs, while delivering stable power to sensing, compute, and drivetrain subsystems.

Phoenix Contact terminal blocks are mounted in DIN rails within the robot frame. There are two DIN rail assemblies: one assembly distributes the main 20-29V battery rail and a global ground reference, while the second distributes the regulated 12V rail, regulated 19V rail, and local ground connections for subsystem components. Connections between the two DIN rail assemblies are routed along the rear of the robot, behind the battery, to reduce wiring clutter and improve maintainability. The electrical system is shown in Figure 6.

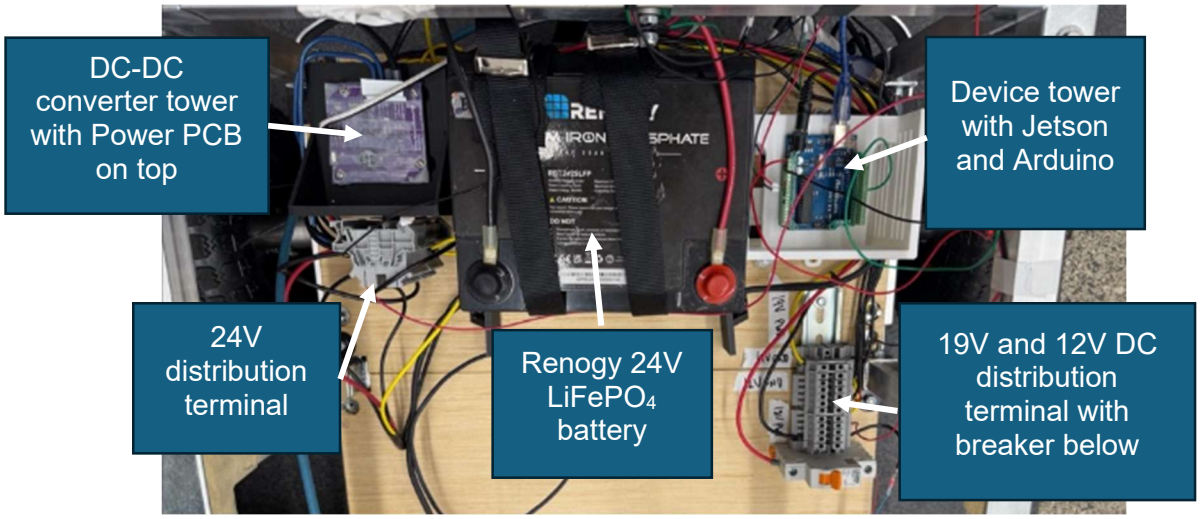


Figure 6: Labeled top view of Shogi's electrical system.

This year's electrical architecture also improves grounding strategy. Previously, system ground was tied directly to the robot's metal frame, using the chassis itself as a shared return path for electrical current. For the current design, a centralized ground distribution rail ("global ground") is used as the primary electrical reference node for all subsystems. Negative terminals associated with the 20-29V, 12V, and 19V rails all connect to this common node. A common ground node provides a more controlled reference point for all subsystems, ensuring consistent voltage levels and a low impedance return path. This grounding system meets team requirement E1 as all components share a common ground.

The regulated 12V rail was last measured at 11.92 volts across common ground. The regulated 19V rail was last measured at 19.06 volts across common ground. Both of these measurements are within a tolerable range of required output and deliver sufficient voltage to power all components.

Voltage across the primary 20-29V battery rail measures between 26.5 and 25.0 volts, with potential lowering over time as the battery's capacity drops. Battery runtime was measured through a multi-day drain process, where it was concluded that on a typical 2-3 amp draw from the battery, it is expected to last 8-10 hours without recharge. This meets requirement E2 as the battery well exceeds the expected run time for a course run, and in fact would likely be able to make it through an entire competition day without a recharge.

The motor controller on the robot is the Roboteq FBLG series which is wired up to the global ground and the battery power input through two safety brake systems. For each motor, their WVU phase wires are fed through the safety brake systems and then to the motor controller. The E-Stop both activates the brake systems electrically and cuts control to the motor controller by shorting the CTRL pin to ground. This way, when the E-Stop is activated, the electrical system can fully disconnect software control over the motors and electrically apply a magnetic brake to the motors.

5.2. Live Electrical Data Capture

In order to load real time electrical data into the robot for the team to utilize, a custom power monitoring PCB was created in KiCad 9.0, shown in Figure 7. It is inserted between the robot's Lithium Iron Phosphate ($LiFePO_4$) battery and the robot itself as a complex dynamic load. The electrical position of the PCB relative to the robot allows it to measure the electrical dynamics at the power distribution node.

The Texas Instruments INA226 shunt current monitor is used to achieve this. It serves two purposes, one of which is to measure live voltage, current, and power, and the other is to track the battery State of Charge (SoC) through coulomb counting. SCL and SDA I2C communication protocol are used to deliver data to the Jetson on the vehicle. The data can then be displayed on the onboard screen.

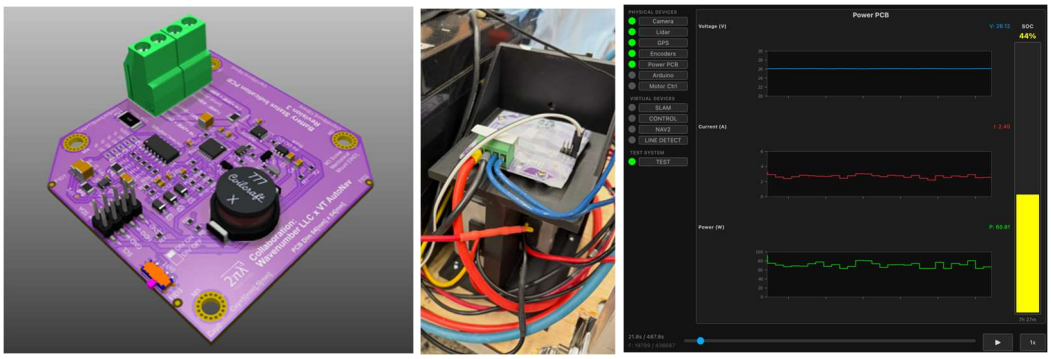


Figure 7: Power Monitoring PCB

6. Perception

6.1. Line Detection

To allow the robot to perceive course boundaries and white surface features, such as painted lines or simulated potholes, and meet perception requirements, the system uses a ZED2i stereo depth camera. The ZED2i, shown in Figure 8, provides a wide field of view, allowing the robot to observe line markings on both sides of the vehicle. Its stereo camera pair also provides depth information, which allows detected image pixels to be projected into 3D space. This also

satisfies requirement P2, since the vehicle uses the onboard NVIDIA Jetson GPU, providing up to 1024 CUDA cores to support heavy computer vision workloads such as line detection, depth processing, and costmap generation.

Figure 8: ZED2i Camera and SICK Multiscan 3D LiDAR

The line detection pipeline begins by capturing a rectified grayscale/RGB image from the ZED2i. Bright pixels are isolated using an intensity threshold, while darker pixels are removed from consideration. A local window-based filter is then applied around each candidate pixel. This filter keeps pixels that are both bright enough and visually consistent with nearby pixels, using the window's average brightness and standard deviation. This helps reject noise, shadows, glare, and isolated bright artifacts.

After the line pixels are identified in the 2D image, the corresponding depth values from the ZED2i are used to convert those pixels into 3D points in the camera frame. These points are then transformed into the robot/map frame using the Robot Operating System 2 (ROS 2) transform tree. The resulting 3D line points are published as a point cloud and passed into the robot's cost map through a custom line layer. In the cost map, detected lines are treated as obstacle regions so the path planner can avoid crossing them. Through lab testing the robot can detect lines after about 10 seconds within about a 1.5-meter distance in front of the robot. This is currently slower than desired; however, the team is confident that line detection speed can be improved by competition.

Figure 9 shows an example test setup with the robot facing a mock ramp. The white ramp edges are detected in the camera image and converted into line points, which are then used by the navigation stack as avoidable boundaries.

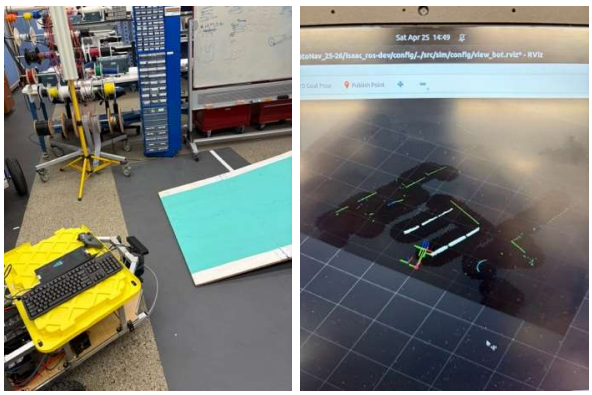


Figure 9: Ramp Line Detection

6.2. Smart Object Detection

To allow the robot to meet perception requirements by detecting obstacles while filtering out ramps, a SICK multiScan165 3D LiDAR, shown in Figure 8, is used in conjunction with a Principal Component Analysis (PCA) based detection algorithm. The LiDAR is mounted upside-down to the front of the robot to allow more azimuth aperture towards the ground and thus the expected field of obstacles. Any surface that diffusely scatters the lasers within 7.5° and -20° azimuth and 90° and -90° will be piped through the PCA-based algorithm.

For standard obstacle avoidance, obstacles are added to the cost map through LiDAR detection using the PCA-based algorithm. The 3D point cloud from the LiDAR comes in as a full 360° view of the world shown as white points in Figure 10. The PCA-based algorithm is able to extract the surface normals at each point in the 3D point cloud. These local normals can then be compared to the robot's own normal where a point above a 30% grade is added to the cost map as an obstacle. The result is the superimposition of object detecting, ground rejecting, and ramp filtering. This allows the robot to avoid obstacles, but not the ramp. This satisfies requirement P1 because the robot can differentiate between obstacles such as barrels and cones and the traversable ramp when tested in a mock course environment. Through lab testing the robot can reliably detect both static and transient obstacles within 100 milliseconds at around 2.5 meters in front of the robot.

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

Figure 10: PCA algorithm testing results. (a) Traversable 16.4% grade ramp in point cloud. (b) Ramp in normal camera vision. (c) 41% grade ramp marked as an obstacle in point cloud.

7. Driving Logic

7.1. SLAM Cost Map and NAV2 Navigation

First, the robot uses Simultaneous Localization and Mapping (SLAM) to both position itself inside a map and place avoidable entities like lines and obstacles, then it uses a system called Nav2 with the Dijkstra search algorithm to create a smooth path to follow to a goal expressed in the robot's local Euclidian space. Any points, whether from line or object detection, will be added to the SLAM cost map as dark regions. These regions are then inflated to provide a 1.1-meter buffer around the obstacles to ensure that the robot's planned path fully clears the obstacles as outlined in requirement D1. When tested, the robot is indeed able to detect and avoid obstacles in its path, satisfying this requirement. The path, shown as a green line in Figure 11, is determined in real time based on the current state of the map. The goal is shown as a red arrow.

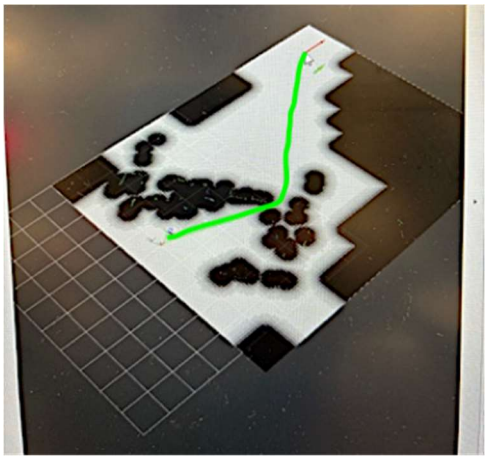


Figure 11: SLAM and Path Planning Example (green plan line traced for visibility)

Once the robot has selected a path to follow, it will launch a controller server to continuously monitor odometry with LiDAR correction, while the robot traverses the path towards its goal. As the robot drives, it may encounter an obstacle and then resample a path. If for some reason the robot gets stuck, it will launch a behavior server to attempt to get the robot unstuck. These behaviors may include but are not limited to plotting intermediate goals, following the path backwards, and employing gradient escape. This satisfies requirement D2 because it ensures the vehicle can update steering directions in real time and get out of situations where it is stuck, including complex situations such as switchbacks and dead ends.

7.2. GPS Waypoint Navigation

As part of the AutoNav course requires navigation solely using GPS waypoints, it is vital the robot can reliably navigate using GPS. To allow the robot to navigate to a GPS goal it uses two systems, the Extended Kalman Filter (EKF) and a heading correction loop over East North Up (ENU) projection. The EKF fuses sensor data from sources such as the motor encoders and Inertial Measurement Unit (IMU) to accurately define the robot's pose where heading is important for GPS waypoint navigation. The converging algorithm takes a GPS waypoint and places a candidate in the robot's local frame using ENU projection. The robot moves a small distance toward the candidate, causing a position differential in GPS space. The angle between this GPS-space displacement and the corresponding displacement in the local frame is then used to collapse the error in angle between the two frames as the robot travels. In physical GPS testing, the robot was able to get within 0.75 m of a target GPS goal from starting points in the northeast, south, and southwest. This satisfies requirement D3 because the robot can successfully use GPS waypoints as goal points when navigating a mock course.

A sequence of goals, called a mission, can be provided to the robot, and it will use ROS 2 action servers to chain goal placement together for the robot to follow. GPS goals are placed into the SLAM map via self-correcting ENU projection, while coordinate goals are placed directly into the robot's frame. With all systems in place and running, the robot can start and drive through a competition course by simply running a mission, continuously reacting to new lines and obstacle configurations as it drives.

8. Key Performance Indicators

To determine whether the vehicle can successfully complete the AutoNav course, the team selected key performance indicators based on its primary autonomous functions: line following, obstacle avoidance, and GPS waypoint navigation. These metrics were chosen because they determine whether the vehicle can successfully complete the course without

penalties. Line following performance was measured by testing the vehicle on a mock course and observing whether it remained within lane boundaries. The team targeted consistent lane tracking with minimal deviation and a 100% successful completion rate, since leaving the course would result in penalties. Recent testing showed the vehicle successfully followed lines in approximately 40-50% of trials, but it does not always detect lane boundaries quickly enough to react when new lane boundaries appear at close range. As mentioned in section 6.1, this feature is being adjusted to ensure line detection occurs more quickly to improve line following rate.

Obstacle avoidance was evaluated by placing cones throughout the course and using visual observation and recorded test footage to determine whether the vehicle successfully detected and maneuvered around obstacles without collision. Obstacle avoidance was tested using a variety of course layouts with different obstacle types, including cones, barrels, and barricades. Current testing showed the LiDAR system can detect obstacles from up to 10 meters away and has consistently avoided obstacles during testing. The robot also accounts for its own footprint when planning paths to ensure it can safely maneuver around obstacles without collision. GPS navigation was evaluated by comparing the robot's final position to the intended waypoint location. The team set a target accuracy of reaching within 2 meters of each waypoint to meet competition requirements, and recent testing showed the robot consistently performed within 1 foot of the target location from multiple starting positions and orientations. Overall, testing demonstrated that the vehicle can reliably complete its core autonomous functions and is capable of navigating the competition course, setting the team on track to satisfy requirements K1 and K2 by the competition.

9. Analysis of Complete Vehicle

9.1. Failures and Lessons Learned

As Shogi was being built and the software systems were being tested, some failures occurred that led the team to learn some valuable lessons that will improve the project for future VT CRO teams. For example, the transition from CAD to manufacturing unveiled a few issues with tolerancing. Originally, the team cut the frame's tubing and drilled mounting holes manually because there was no access to more precise fabrication equipment. When it was time to install the flooring, the team found that mounting locations were not true to the CAD and the flooring could not be installed. To address this, the team switched the mounting holes to slots, which allowed the installation to progress as intended. Additionally, while testing the vehicle, the team found that the smooth plastic caster wheel would slip on the ramp's edge instead of rolling over it, preventing the robot from climbing the ramp. To solve this traction issue, the team 3D printed a custom TPU tire with grooves specifically designed to climb the ramp. After testing three iterations of the tire, Shogi is now able to climb onto the ramp at any speed and orientation.

When designing the electrical power monitoring PCB, revision 2 of the PCB was thoroughly lab tested to survive a range of DC voltages up to 30V, steady state currents up to 5A, steady state loadings like sine waves, and noise. Despite this, when installing it onto the robot and turning the robot on, the shunt resistor exploded like a fuse. A postmortem analysis on the robot and PCB found no components damaged except for the shunt resistor, which was the designed failure mode. Afterwards a solid metal pulse resisting shunt resistor was installed and transient analysis of current measured showed that upon turning on the robot, the combined current draw is comparable to that of a wall outlet for about 2 milliseconds, demonstrating that this failure had been rectified.

9.2. Software Organizational Overview

While Shogi was being built, the team used last year's robot as a software testbed. This allowed this team to develop and test perception and navigation algorithms. To build, test, and

implement new software features and fixes, team members create a new git branch titled “feature/desired_feature” or “fix/desired_fix”. The team’s software is entirely version-controlled via git. Code is written on team members’ personal laptops and then pulled onto the robot’s Jetson for real-world testing. The team also tests in simulation, which is explained in further detail in section 9.3. If bugs are identified, they are shared with the team and posted on the team Jira board to ensure prompt cross-team cohesion and triage bug urgency. Additionally, the team collects real-world measurements during testing using an automated testing data collection system on MATLAB for further analysis.

9.3. Simulation Overview

To allow the team to test software prior to physical testing, a simulation is used. First, a high-resolution Blender exported STL reconstruction of a specific set, importantly containing 5 ramps at different angles, is created. Then an agent carrying a LiDAR that exactly matches the specifications of the real LiDAR mounted on the robot is created. PCA grade detection, as a filtering algorithm, was first tested rigorously in simulation, constantly tweaking parameters until it was correct, and then could be deployed on the robot once thousands of simulations passed. To pass, the agent must not collide with obstacles regardless of shape and traverse the correct ramp based on a threshold. Figure 12 shows the simulation environment used to test the PCA-based object detection and avoidance algorithm.

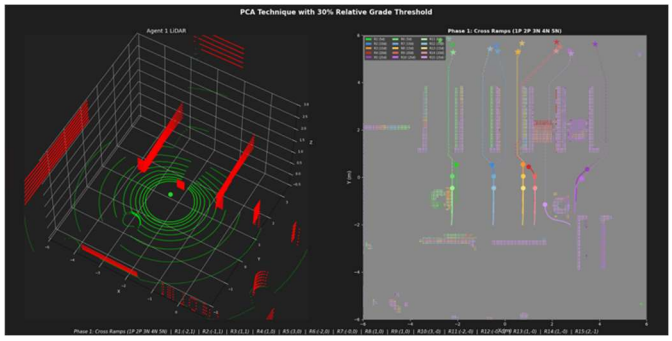


Figure 12: PCA-based LiDAR Simulation

9.4. Physical Testing Overview

While developing the obstacle avoidance system, the team wanted to compare the programmed hyperparameters with real-world performance. To do this, the Nav2 robot and inflation radii were varied and the resultant obstacle avoidance distances were measured. As Figure 13 shows, the robot was not respecting the robot radii and was both mapping and navigating within the robot’s programmed perimeter. This had previously been compensated for with unrealistically large inflation radii. Therefore, this hands-on testing revealed a hidden failure case and improved the robustness of the team’s perception-navigation stack once the problem was found and fixed.

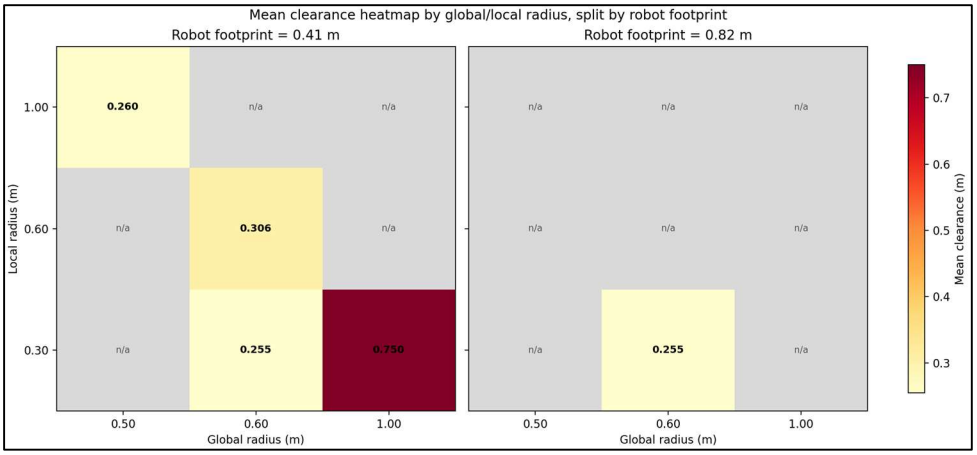


Figure 13: Inflation and robot radii vs. clearance distance.

10. Cybersecurity Analysis

The vehicle has several cybersecurity risks because it uses ROS 2 networking, Docker containerized deployment, and many external sensor/control interfaces. These are acceptable during development, but they would need to be hardened before the vehicle is widely deployed.

10.1. Unsecured ROS 2 / DDS Network Communication

The robot uses a DDS-UDP network, so the ROS 2 stack on the Jetson can communicate with tools like RViz on a laptop. If this network is not secured, another device on the same network could potentially discover ROS topics, read sensor data, spoof messages, or publish unsafe commands to topics such as /cmd_vel. To harden this, the system should use ROS 2 security features such as SROS2, DDS authentication, encryption, and access-control policies. The robot should also operate on an isolated network with firewall rules and only allow approved machines to join.

10.2. Over-Permissive Docker Container Access

The vehicle software runs inside a Docker container with access to the GPU, USB devices, serial ports, mounted project files, and hardware such as the ZED camera, LiDAR, GNSS receiver, encoders, Roboteq motor controller, and E-Stop interface. If this container were compromised, an attacker could potentially affect vehicle behavior or access critical hardware. Before production, the container should follow least-privilege design: only required devices should be mounted, privileged/root access should be avoided, and development tools should be removed.

10.3. Unauthenticated Sensor and Control Interfaces

Many vehicle interfaces are trusted by default, including USB/UART connections for GNSS, Roboteq motor control, encoder feedback, joystick control, and E-stop communication. A spoofed or compromised device could send false GPS data, fake encoder readings, or unsafe control inputs. To reduce this risk, the system should verify device identity using stable hardware IDs, vendor/product IDs, and signed firmware where possible. Sensor data should also be sanity-checked against other sensors, and all motor commands should pass through a safety layer that enforces speed limits, command timeouts, and E-Stop priority.

10.4. RF Interference with the Wireless Keyboard

Installed on the robot is a small screen that forwards the Nvidia Jetson Orin Nano display to a format that makes it easier for the team to control the robot's systems. It allows normal computer inputs like mouse and keyboard, and the wireless keyboard is connected via an RF

USB receiver. To ensure that this is safe, three layers of protection are utilized. First, the graphical user interface (GUI) on the screen operates in a WM Open box kiosk mode, which is industry standard for Linux based OS, where it rejects injected keystrokes and super user commands. Next, users can easily lock the screen using the shortcut Ctrl + Shift + L when not in use, preventing unauthorized access to the Jetson. Finally, teletypewriter (TTY) switching in the GUI is disabled such that only recovery mode is Secure Shell (SSH), which is a secure method the team uses already.