2024 Intelligent Ground Vehicle Competition

Wayne State University
Warrior Robotics

Vehicle: **Shanti^3**

Submission Date: May 15, 2024

Team Captains

Hanna Bulinda	President | Software	hg3805@wayne.edu
Nathan Chantiny	Software Lead	hk2133@wayne.edu

Team Members

Lucas Fraizer	Hardware Lead	ho7683@wayne.edu
Jaideep Siva Senthilprabhakar	Software	ho5863@wayne.edu
Adrian Tlatelpa	Software / Hardware	hh5426@wayne.edu
Lloyd Brombach	Software	as7923@wayne.edu
Andrea Cuc	Software	gw4930@wayne.edu
Ryan Ross	Hardware	hq9172@wayne.edu
Michael Jessie	Hardware	aj5442@wayne.edu

"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 Advisor

Abhilash Pandya, Ph.D.

Professor, Department of Electrical & Computer Engineering

Contents

1. Conduct of Design Process and Team Identification and Organization

1.1 Introduction

Warrior Robotics is a student-run organization that integrates the talents and efforts from Freshman undergraduates to graduate students. The team embraces its diversity, with members bringing a wide array of skills and perspectives from various fields of study. This diversity not only enhances the team's innovative capabilities but also mirrors the nature of the modern robotics industry. Warrior Robotics aims to foster a hands-on learning environment where students can apply theoretical knowledge to real-world engineering challenges, thereby preparing them for professional success in the competitive field of robotics. We also have extensive training modules setup to enhance skillsets needed for the program.

Building on last years' experience, this year, our vehicle design integrates new hardware and software solutions for improved stability and maneuverability. Mechanically, we introduced a 3D-printed weatherproof shell, a custom 3D-printed camera mount for enhanced camera stability and protection, and an upgraded three-camera setup providing a greater than 180-degree field of view. We also realized the need for accuracy especially during various lighting conditions. Hence, our software innovations focused on using parameter optimization techniques for the perception module. We utilized optimization techniques for camera positioning, and dynamic HSV thresholding to adapt to varying lighting conditions. In addition, we use real-time pose estimation using April Tags for precise camera orientation adjustments. These enhancements collectively improve our robot's resilience and navigational accuracy in diverse environments. The details of each of these is provided in the report. Each section of the report (Hardware, Electronics and Software) has a failure modes and mitigation strategy individually.

1.2 Team Organization

The team currently comprises of nine members from various backgrounds in their respective majors. Prior to becoming an official member, everyone is provided with a series of training modules to prepare them for the use of the various software and hardware tools commonly used in the lab. Upon successful completion of their modules, they are assigned an official role on the team. To optimize our efforts, we have structured the team into two sub-teams: Software and Hardware. This division represents a minor shift from the organizational structure of previous years. With a smaller group, this new arrangement has proven to be the most effective. Each sub-team takes charge of responsibilities that are related to their area of expertise.

Software Team: Tasked with the development and testing of all software components, this sub-team handles everything from writing code for autonomous navigation to simulating environments for testing. Members work in sprints to manage tasks efficiently, allowing for rapid development and iteration of software features.

Hardware Team: Responsible for the design, assembly, and maintenance of the vehicle's mechanical and electrical components. This sub-team collaborates closely with the Software Team to ensure that all hardware modifications enhance the functionality and reliability of the software applications.

1.3 Design Assumptions and Process

Adopting the agile methodology, Warrior Robotics prioritizes flexibility and responsiveness to change, which is crucial in the fast-evolving field of robotics. Our agile approach is characterized by weekly sprints where team members discuss their progress, challenges, and next steps. This method promotes continuous improvement and adaptation, with the following key practices:

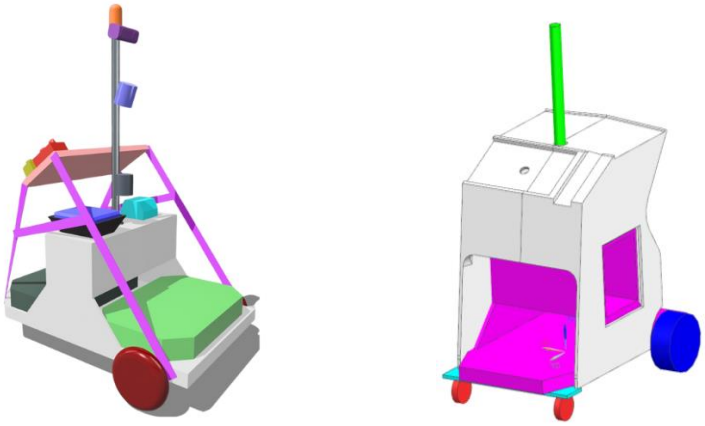
Sprint Planning: At the start of each sprint, tasks are clearly defined and assigned to team members by their respective team lead, ensuring that everyone is aware of their responsibilities for the coming week. A sprint typically lasts up to two weeks.

Weekly SCRUM: These meetings occur weekly, and allow team members to report on their progress, discuss any challenges they face, and outline their plans for the upcoming week. This constant communication ensures that issues are quickly addressed and that the team remains aligned with project goals.

Sprint Reviews: At the end of each sprint, the team reviews completed work and discusses what can be improved in the next iteration. This reflection ensures that the team is always learning and growing from past experiences.

2. Architecture of Mechanical System

2.1 **Description of Components**



(a) *(b)*

Figure 1. (a) A rendered 3D model of former robot Karina-2023 (Left), (b) A rendered 3D model of Shanti (Right)

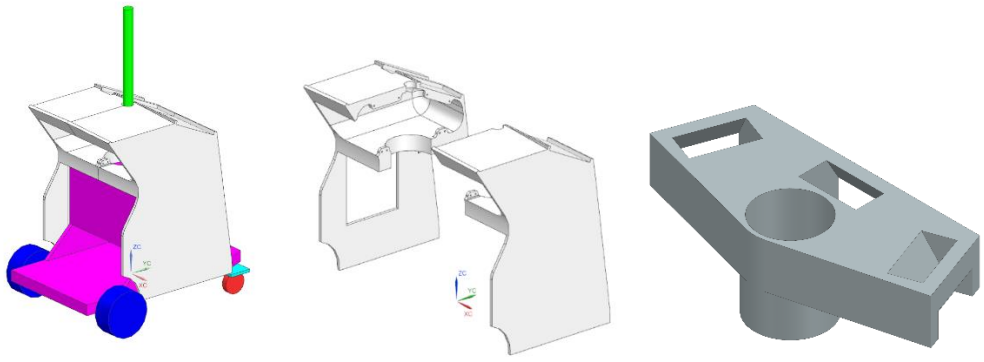
***Caster Wheels & Wheel Placement*:** To enhance the stability and maneuverability of our robot, changes were made to the placement of the caster wheels. Initially, a single caster wheel was positioned at the center back of the robot **(Figure 1a)**. However, based on performance assessments and stability requirements, we decided to install an additional caster wheel, positioning them similarly to the hoverboard wheels: one on the left and the other on the right (**Figure 1b**). This adjustment not only improved the lateral stability of the vehicle but also optimized its turning ability, particularly on uneven terrain. The dual-caster setup ensures that the robot maintains better contact with the ground, reducing the risk of tipping and improving its response to steering inputs. These modifications are critical as they align the caster wheels with the vehicle's center of gravity, promoting a more balanced distribution of weight and improved dynamic stability.

Chassis: This year, we decided to keep the main design elements unchanged, modifications were implemented that allowed us to add an additional caster wheel on the base of the chassis. These adjustments were made without altering the overall configuration of the E-boxes and the payload box.

Payload Box: We have decided to maintain the design of the payload box, as it has proven effective for our setup. The payload box is designed to be side-filled, this enhances the user convenience by removing the need to stand in front or reach behind the robot to access it over the E-boxes. It also serves as a wire pass-through, which simplifies the connection between the E-boxes and the sensors.

Sensor Arm: This year, we decided to remove the A-Frame support (**Figure 1a**), which proved to be somewhat flimsy and less effective than anticipated. Instead, the sensor arm, shown in green in Figure has been reinforced with the new 3D-printed shell, which now also houses the control module for the emergency stop (E-stop), power switches, and a secondary monitor. This updated arrangement not only allows for the external operation of the robot's operating system but also enhances the overall weatherproofing capabilities.

2.2 Decision on frame structure, housing, and design



(a) (b) (c)

Figure 2. From left to right, (a) A rendered 3D model of the front of Shanti, (b) A 3D exploded view of the shell, (c) A rendered 3D model of the camera mount.

2.2.1 **Innovation:** *3D-Printed Shell*

A shell was designed in Siemens NX (Computer-aided design software) with the purpose of providing a sturdy, water-resistant housing for the sensor equipment, touch pad, vehicle e-stop button, electrical power switches, and two electrical boxes. The shell consists of two pieces as seen in **Figure 2b**. These pieces connect in the middle and are designed to be easily removed if repairs are needed. The pieces were 3D printed with Acrylonitrile Butadiene Styrene (ABS). ABS was chosen due to its sturdiness, heat resistance, and relatively low cost. The shell sits on top of the wooden payload box and is fastened to the sides of the chassis with hook and loop fasteners (for support). The sensor arm runs through the center of the shell and provides additional support **(Figure 2a)**.

2.2.2 **Innovation:** *Two Additional Cameras*

The decision to phase out the Zed2i came from the interference issues caused by its internal IMU, along with reduced lane visibility. Initially, we considered retaining the Zed if adjustments to the configuration of our other sensors could resolve these issues. However, these modifications failed to show any improvements. Specifically, the lanes close to the robot were still obscured, creating a visible gap between *base_link* and the regions where the lanes should have been detectable. By adding two cameras on the left and right sides, our field of view exceeds 180-degrees by extending visibility on the sides. This setup eliminated the visibility gaps, ensuring that the robot could navigate through the lanes without inadvertently crossing over them.

2.2.3 **Innovation:** *3D-Printed Camera Mount*

The camera mount has been designed to accommodate three cameras **(Figure 2c)**, each positioned at a 70-degree angle relative to one another. This configuration allows the cameras to face downward at a 55-degree angle, ensuring an optimal view of the lanes. The mount includes tailored slots that securely hold each camera in place, preventing any movement during operation. The design of the mount provides enough clearance to comfortably fit over the sensor arm. For stability and security, it is securely fastened using a stainless-steel hose clamp. Additionally, cable routing has been integrated into the design, allowing for an organized passage of wires into the sensor arm, and for protecting the cables from any environmental exposure or mechanical wear.

2.3 Safety and Reliability

To protect our system, we have continued to use two IP66-rated enclosures to protect our batteries, laptops, smaller sensors, and wiring. The wiring within the payload box is further protected by the 3D-printed shell that was made of ABS. We've moved away from using the ZED2i Camera; instead, we have three cameras that are now housed within a 3D-printed mount made of PLA (polylactic acid), which offers additional protection. Additionally, we have sealed the wheel bearings of our brushless hub drive motors to prevent wear and tear from outdoor conditions, ensuring continued durability and performance.

2.4 Failure Modes and Mitigation Methods

While the type of failure cannot be anticipated, we want to ensure that we are prepared in the event it does occur within our Mechanical system.

Failure Mode	Potential Cause	Mitigation Method
Damage to *chassis*, *sensor arm*, or *wheels*	If the robot were to fall or if it were to bump into something.	Carry additional materials for repair such as wood, aluminum, spare wheels of each type, and lubricant for casters.

Table 1. Mechanical Failure Modes and Mitigation Methods.

3. Architecture of Electronic and Power Design Systems

While many of our electronic components retain their original designs from 2023, the wiring and routing of cables have been slightly revised to better suit our current needs. Shanti is powered by two 36-volt, 20 amp-hour lithium-ion battery packs. The first battery pack supplies 36 volts to the motors and its control board, as well as powers the 12-volt and 5-volt DC-DC converters for auxiliary components. The second battery is specifically allocated to power the new laptop—a Lenovo Legion Pro 5i Gen 8, with a 16 core 4.8Ghz Intel Core i7 CPU, an NVIDIA® GeForce RTX™ 4050 GPU, and 32 GB of RAM—through a 19-volt DC-DC converter. Both batteries are equipped with XT-60 quick connectors for easy swapping and can be charged in place with weather-resistant connectors mounted on the shell of the lower electronics compartment. The laptop retains its internal battery, allowing for seamless battery swaps without needing to shut down.

The main battery typically provides approximately 64 minutes of runtime per charge, assuming a 50% discharge of the 20 AH capacity, with a maximum consumption of 230 watts (160 watts typical). This setup ensures sustained operational capacity for extended periods.

In a significant update to our sensor components, we have replaced the Zed2i stereo camera, which was causing interference and inadequate lane coverage, with three Logitech C290S webcams. These cameras offer improved performance and reduced electromagnetic interference. Additionally, the UM7-LT IMU has been integrated to enhance sensing accuracy.

The motor driver board—a repurposed hoverboard control board—has been reprogrammed with open-source ROS hoverboard firmware. It communicates with the main computer via USB serial and an FTDI USB to TTL serial converter, providing essential odometry feedback. The sensor suite now consists of the three Logitech webcams, UM7-LT IMU, hall effect sensors, and a Reach RS+ RTK GPS module. This year's configuration excludes Ethernet-dependent devices, streamlining our communication setup to primarily USB connections.

3.1 Description of Components

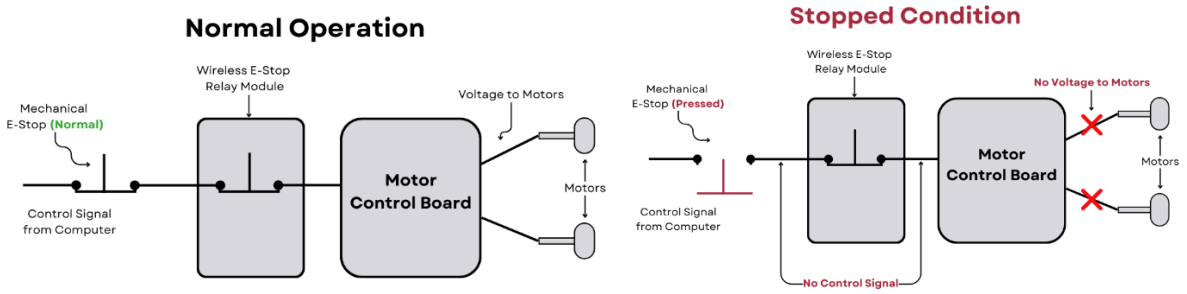


Figure 3. Normal Operation of E-Stop (Left) and Stopped Condition of E-Stop (Right).

Electrical Boxes (E-box) and Payload Box: The E-boxes are mounted on self-clamping sliders to ensure they remain securely in place while maintaining a streamlined design. In our 2024 design, we implemented changes by adding a new wire shield, accommodating the modifications made by removing and adding additional wires through the payload box. To uphold modularity, the I/O runs are detachable, except for major electrical power runs. This allows the E-boxes to be fully removable from the vehicle at any time, enhancing maintenance and upgrade flexibility which would be difficult with a more integrated design.

Temperature Sensor: We've continued to use a temperature sensor to activate the cooling system if the E-box temperatures go above the threshold. It helps with regulating the cooling and serves as a safety check.

3.2 Safety and Reliability

Based on the rules that the IGVC has set, we are using both a mechanical and wireless E-stop. These emergency stop systems are designed to operate both independently and together, providing safeguards to deactivate the vehicle quickly in case of any issues. The local and remote E-stops are wired in series, and the activation of either system will disrupt the control signal to the motor control board, leading to the shutdown of the motors **(Figure 3)**.

3.3 Failure Modes and Mitigation Methods

Failure Mode	Potential Cause	Mitigation
Battery Failure	Overcharging, Short Circuits or wiring faults, or Physical damage	Carry an additional battery of each type needed, for fire emergencies – carry PPE and baking soda.
Overheating of Electrical Equipment	Insufficient cooling or ventilation, or a Malfunction with temperature sensor.	Ensure cooling system is still functional and carry spare temperature sensor.
E-Stop Malfunction	Electrical or wiring faults, Mechanical wear, Firmware issues.	Carry a spare E-Stop button

Table 2. Electric and Power Failure Modes and Mitigation Methods.

4. Software and Systems Integration

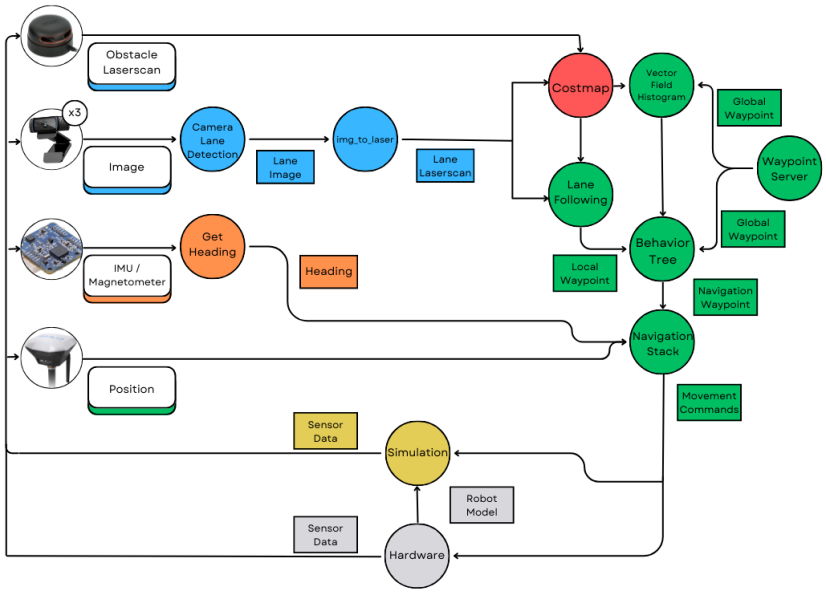


Figure 4. Our Software Architecture and how it interacts with our sensors.

The modifications made to our software architecture were designed with modularity, efficiency, and robustness in mind. The strategy incorporates a multi-layered approach combining perception, navigation, and control systems to handle complex environments and tasks set by the IGVC. Our software stack is built on ROS (Robot Operating System), facilitating seamless integration of various sensors and actuators.

4.1 Perception

The Perception module is critical for Shanti's interaction with the environment. This module utilizes a combination of LIDAR, three-camera setup, and machine learning algorithms to accurately detect lanes, potholes, and obstacles **(Figure 4)**.

4.1.1 Mapping Technique Overview

4.1.2 Cost map

This mapping technique that we've used involves creating a 2D occupancy grid where each cell in the grid represents the spatial cost of navigating that area. Costs are inflated based on proximity to the detected obstacles, effectively creating a buffer zone that helps the robot avoid collisions. The cost map updates dynamically as the robot moves, using sensor data from LIDAR and cameras to adjust the costs associated with various cells based on the presence of obstacles. The lane and obstacle/pothole detection data are added to the cost map as two separate layers. This mapping is crucial for navigation decisions, ensuring the robot avoids obstacles and follows safe paths.

4.1.3 Lane Detection and Conversion

This process involves detecting lanes using camera images, which are then processed to create a binary representation of detected lanes. This binary image is transformed into a bird's-eye view using perspective transformation techniques. The transformed image is used to generate a laser scan, which is then plotted on the cost map. This technique ensures that the robot can navigate effectively by following lane markings, critical in structured environments like roads or tracks.

4.1.4 Obstacle Detection

Our robot is equipped with a 2D LiDAR capable of a 360-degree field of view. It is mounted parallel to the plane of the robot on the sensor arm and positioned at a height to avoid detecting a pre-planned ramp, the LiDAR identifies obstacles within its optimized field of view. The data collected from the LiDAR is integrated into the cost map, marking detected obstacles accordingly. To prevent the robot itself from being mistakenly marked as an obstacle on this map, a box filter from the laser filters package is employed. This filter effectively isolates the robot chassis within the raw laser scan, allowing the robot to detect immediate surroundings without accidental self-detection. Only the filtered laser scan data is passed onto the cost map, ensuring clear navigation paths.

4.1.5 Lane and Pothole Detection

We employ an advanced HSV-based filtering technique alongside YOLOP, a deep learning model, to ensure reliable lane detection under various lighting conditions. The system switches between these methods based on the environmental context, ensuring optimal performance. The primary method we use is the HSV-based filter that isolates white pixels above a certain intensity, detecting lane lines. This method is refined by adjusting HSV settings and incorporating additional filters and Hough lines for line fitting. While computationally efficient, its performance heavily depends on lighting conditions. The alternative method we utilize is the YOLOP Convolutional Neural Network, trained on the BDD100k dataset, which excels in detecting lanes, drivable areas, and objects but requires a robust GPU and considerable memory. Pothole detection mirrors the HSV technique, using a filter to identify circular or elliptical shapes, proving reliable in outdoor testing with minimal tuning. The image-to-laser pipeline converts detected lanes from a binary image to a precise laserscan for mapping, involving steps like perspective transformation for a bird's-eye view, gap closure, and pixel-to-meter conversions for distance measurements.

4.1.5.1 *Optimization Algorithm Development*

In our pursuit to refine the parameters for both camera positioning and HSV settings, we used the SciPy optimization library, particularly utilizing the Nelder-Mead simplex method. This decision was driven by the need to triangulate the most effective values for our multi-camera setup, enhancing the robot's environmental perception capabilities. To optimize our parameters, we employed a steepest descent algorithm. Data collected on lane distances at various angles was used in this process (**Figure 5b**). The objective function was defined as the difference between the measured data and the computed data. By minimizing this error function, we were able to produce optimized parameter values. The optimization process involved the following steps:

Data Collection: We gathered data on lane distances at different angles, ensuring a comprehensive dataset for accurate optimization.

Objective Function Definition: The objective function was formulated as the difference between the measured lane distance data (angle vs. distance, polar coordinates) and the computed lane distances based on initial parameter estimates.

Error Minimization: Using a steepest descent algorithm (Nelder-Mead simplex method), we iteratively adjusted the parameters to minimize the objective function, effectively reducing the discrepancy between the measured and computed data.

By systematically minimizing the error function, we enhanced the accuracy of our system, ensuring reliable performance in diverse environments.

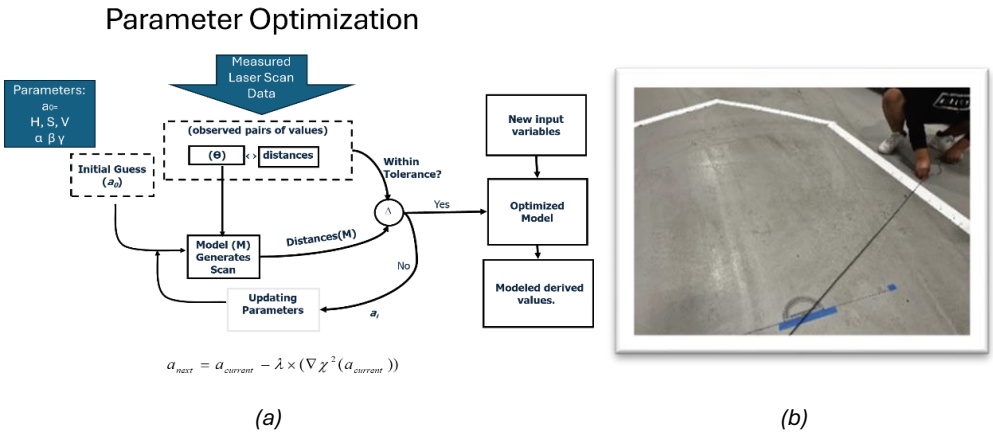


Figure 5. (a) Optimization Method Used. (b) Cross checking data by manually taking measurements of lanes at various angles.

Challenges Encountered

Despite the benefits, the Nelder-Mead method presented specific challenges, primarily its tendency to converge to local minima. This characteristic poses a significant risk in optimization as it heavily relies on the initial starting point provided. Our approach involved setting the initial guess based on our best estimates, which may not always lead to globally optimal solutions. This limitation highlights an inherent risk: the final optimized values might only represent the best solutions relative to our initial assumptions rather than true optimal values.

4.1.5.2 **Innovation:** *Optimization for Camera Positioning*

The transition from a single-camera setup to a three-camera configuration was designed to improve Shanti's field of view. This involved adjusting the orientation of each camera to capture optimal lane visibility:

Left Camera: Positioned to capture the positive y-direction.
Middle Camera: Aligned for the positive x-direction.
Right Camera: Geared to capture in the negative y-direction.

These changes created a need for parameterization of the camera settings. We integrated parameters into the launch file for easier modification and adaptation:

Rotation Parameter: Adjusts the direction lanes are drawn for each camera in the visualization tool (RViz).
Home Position Parameter: Defines the starting point of lane drawing in RViz for each camera.

The optimization was designed for the camera setup that focuses on adjusting:

Yaw, Pitch, and Roll: Orientation parameters of the cameras.
Pixel to Meter Conversion: Ensures accurate spatial translations in image processing.

4.1.5.3 **Innovation:** *Optimization for Image Processing*

The purpose of optimizing the image processing code was to adapt to different environmental conditions and improve lane detection accuracy.

Parameter Optimization: The script specifically tunes parameters that directly affect image processing, especially those difficult to adjust manually or through direct observation.

Weather Adaptation: The script dynamically adjusts the image processing parameters to suit current weather conditions, ensuring that the robot's vision system remains effective under varying light and weather scenarios.

Goal of Optimization: The primary goal is to calibrate the robot effectively before every run, ensuring the highest accuracy in lane detection and navigation.

Focus on Efficiency: By automating the calibration process through optimization scripts, the team aims to reduce the time and effort required for manual calibrations, thus speeding up the preparation process for each run.

4.1.5.4 ***Innovation*** **-** *AprilTag for Camera Pose Estimation*

AprilTag detection was implemented to provide real-time pose estimation of the camera relative to known marker positions. This system utilized a library specifically designed for quick detection and precise localization of AprilTags—square, black-and-white markers that can be easily recognized. By identifying these tags in the camera's field of view and computing their positions and orientations, the system accurately determined the camera's 3D pose relative to these markers. This information is crucial for tasks requiring precise navigation and interaction with objects in the environment, as it allows the robot to understand its spatial relationship to these objects.

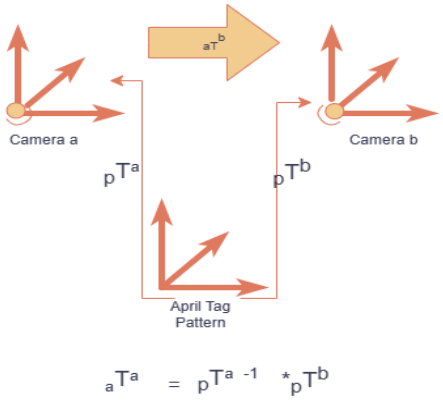


Figure 6. Computing transforms from one camera to another using the AprilTags.

In our project, we innovatively utilized a single AprilTag placed strategically between two cameras to calculate the spatial transforms necessary for multi-camera coordination. This approach allowed us to derive precise relative orientations and positions between the two cameras, which is crucial for tasks requiring synchronized lane detection from a single perspective.

By placing the AprilTag centrally, each camera captures images of the same tag from different viewpoints. The detection and decoding processes of the AprilTag from these separate angles enable us to extract the individual camera poses relative to the tag. With the intrinsic parameters of both cameras pre-calibrated, we apply the pose estimation data to compute the rotation and translation vectors for each camera with respect to the AprilTag.

The critical step involves using these vectors to calculate the relative transform between the cameras. By understanding the pose of each camera relative to a common fixed point (the AprilTag), we can algebraically derive the transformation needed to convert coordinates from the frame of one camera to the other. This transform is vital for combining data from both cameras to achieve a unified and accurate representation of the environment.

The simplicity and accuracy of using a single AprilTag for this purpose not only streamlined our setup but also reduced the computational overhead typically associated with more complex multi-tag configurations.

The spatial transforms between multiple cameras were calculated using the pose information derived from the AprilTag detections (**Figure 6**). By establishing a rigid transformation matrix that included rotation and translation vectors between cameras, the system could unify the visual data from different perspectives. This capability was particularly beneficial for complex image processing tasks like lane detection, where multiple camera angles

provided a more comprehensive view of the robot's surroundings, thus enhancing detection accuracy and reliability even in visually cluttered or dynamically changing environments.

4.1.6 Camera Calibration using ROS usb_cam Package

In the camera calibration process, we address two primary types of distortion: radial (pin-cushion and barrel) and tangential. Radial distortion causes straight lines to appear curved, and this effect intensifies as we move away from the center of the image. Tangential distortion occurs when the lens is not aligned perfectly parallel to the imaging plane, causing some areas of the image to appear closer than they are.

To correct these distortions, we need to determine five key distortion coefficients that characterize the lens imperfections. These coefficients are critical as they help us mathematically correct the distortion in the images captured by the camera. Along with distortion coefficients, we require intrinsic and extrinsic parameters of the camera.

Intrinsic parameters include the focal length (f_x, f_y) and the optical centers (c_x, c_y) of the camera. These parameters are crucial as they form the camera matrix, a 3x3 matrix that transforms 3D camera coordinates to 2D image coordinates (**Figure 7**). This matrix is unique to each camera and, once calculated, can be reused for any image taken with the same camera.

Extrinsic parameters consist of rotation and translation vectors that help translate a 3D point's coordinates into the camera's coordinate system, essential for understanding the camera's orientation and position in space.

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

2D Image Coordinates | Intrinsic properties (Optical Centre, scaling) | Extrinsic properties (Camera Rotation and translation) | 3D World Coordinates

Figure 7. Obtaining 2D image coordinates using this method.

For stereo vision systems, correcting these distortions is paramount before any advanced processing. To compute these parameters, we capture multiple images of a known pattern, such as a chessboard, from various angles and distances. The chessboard provides a predefined pattern where the corner points are easily detectable and have known positions in space. By correlating these known points with their positions in the image, we can compute the distortion coefficients and camera matrix. To ensure accuracy, it is advisable to use at least 10 different images of the test pattern during the calibration process.

4.2 Navigation and Localization

The Navigation module processes data from the Perception module to plot a safe and efficient course through detected lanes and around obstacles.

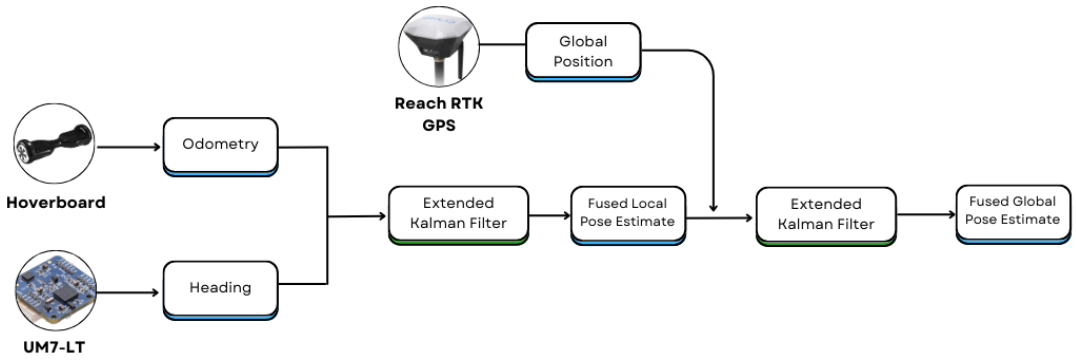


Figure 8. Demonstrating how odometry, GPS, and IMU data are fused to give a global pose estimation.

4.2.1 Mapping Techniques

Extended Kalman Filter Based Localization

This technique uses an Extended Kalman Filter (EKF) to fuse data from various sensors to produce accurate estimates of the robot's position and orientation (**Figure 8**). The EKF combines data from odometry, IMU, and GPS to correct for the individual inaccuracies and biases of each sensor, providing a reliable and precise global pose estimate. This method is fundamental for navigation tasks, ensuring the robot understands its location within the map and can make informed decisions about movement and path planning.

4.2.2 Calibration and Heading Determination from UM7-LT Module

Calibration of the UM7-LT module involved setting up the sensor to accurately measure the robot's heading using its integrated inertial measurement unit (IMU). This process included compensating for magnetic distortions and aligning the sensor's output with the robot's movement axes. The calibrated UM7-LT provided real-time data on the robot's orientation relative to Earth's magnetic north, which is crucial for navigation and orientation tasks. Accurate heading information ensures that the robot can maintain a consistent trajectory, perform precise turns, and correct any drift in its path, leading to more reliable and accurate navigation in diverse environments.

4.2.3 Waypoint Selection

Integrates data from the perception layer to dynamically adjust the robot's path. It uses a cost map (**Ref.** 4.1.1), that is generated from sensor inputs to navigate around obstacles and between lanes while adhering to competition rules.

4.2.4 Path Planning and Following

Employs Lane Following (LF), Vector Field Histogram (VFH) and Dynamic Window Approach (DWA) for real-time trajectory planning. These algorithms allow Shanti to adjust its path on-the-fly based on real-time obstacle data and lane information. Both LF and VHF are used to find intermediate, or local, waypoints for the robot to move towards, but each path planning algorithm uses this approach differently.

4.2.4.1 Vector Field Histogram

The Vector Field Histogram (VFH) algorithm is used to identify optimal waypoints for the robot, utilizing a local cost map to determine areas with low obstacle density. As the robot progresses towards these waypoints, VFH dynamically updates them to ensure navigation toward the overall destination as defined by GPS coordinates. The cost map categorizes lanes as obstacles to refine waypoint accuracy. VFH selects waypoints that are strategically placed away from lanes yet near enough to guide the robot efficiently toward its broader navigational goals.

4.2.4.2 Lane Following

The lane following program is designed to generate a local waypoint that stays within lane boundaries while navigating around obstacles. By analyzing lane scan data, it determines whether the robot is in the right or left lane. A 1-meter buffer is added to the X coordinate to maintain a safe distance from the lane edges, enhancing safety and accuracy. If an obstruction is detected or the path cost exceeds 30, the program automatically recalibrates the waypoint to a more suitable location, ensuring the path remains clear and navigable.

4.3 Graphical User Interface (GUI)

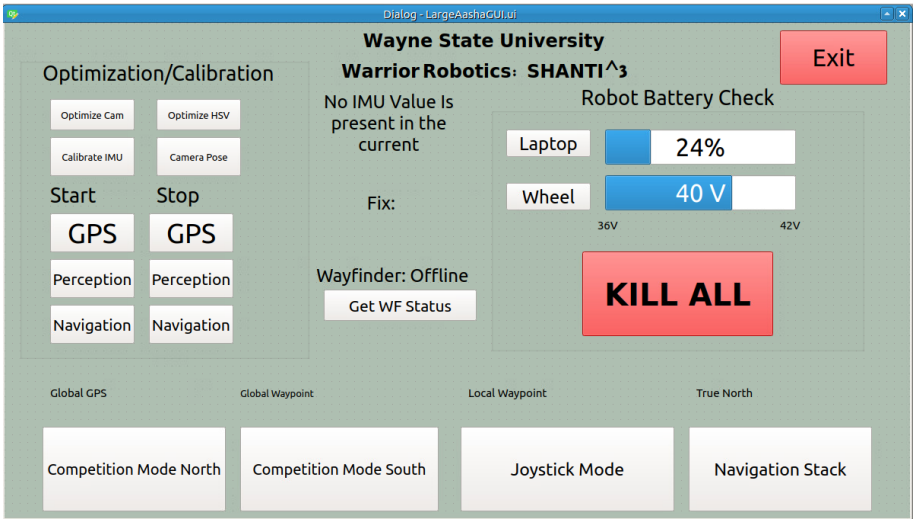


Figure 9. Graphical User Interface (GUI) system.

The GUI we designed for our robotic system is a comprehensive tool that enhances ease of use of various critical functionalities and system integration. It features an optimization and calibration section dedicated to enhancing the performance of the robot's three-camera setup. This section allows for the execution of launch files that optimize camera orientation and threshold values for line detection across varying lighting conditions. It also facilitates the calibration of the IMU sensor and can recompute the pose of each camera should the system fall out of calibration. Beyond these optimization tasks, the GUI provides real-time monitoring of both the PC and robot battery levels, displays GPS location data along with the relative fix status, and offers a clean, efficient method for terminating all running nodes. Additionally, the interface includes a one-button launch feature for initiating competition mode, streamlining the process for both north and south runs. This GUI not only simplifies complex operations but also enhances operational efficiency and system integration (**Figure 9**).

4.4 Control System

4.4.1 **Motion Control:** The Shanti Navigation Stack uses a bottom-up approach for modularity and simplicity, splitting navigation into trajectory and velocity control tasks. Trajectory control merges local waypoint selection with a Dynamic Window Approach (DWA), while velocity control, managed by a controller called the Agent, adjusts speed based on sensor and cost map data to prevent collisions. If a collision is imminent, the robot reverses.

The system employs DWA for reactive situations and a PID controller for stable, path-adherent navigation. When no viable path is available, the system defaults to simpler navigation directives.

4.4.2 **Safety and Emergency Protocols:** Includes redundant systems for emergency stops and fault detection, ensuring Shanti can safely halt operations if unexpected conditions are detected.

4.5 Integration and Testing

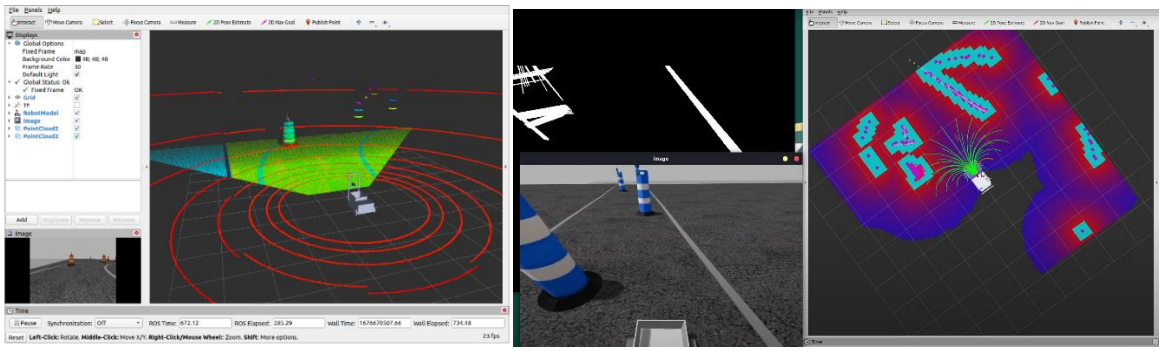


Figure 10. Simulation Environment utilizing: Gazebo (Mid) and RViz (Left and Right).

4.5.1 **Simulation:** To optimize testing and development during hardware finalization and to mitigate weather impacts, we developed a comprehensive simulation of our environment (**Figure 10**). This simulation includes our complete robot model, sensors, simulated paths with obstacles, and GPS waypoints, allowing for rapid prototyping and extensive testing. By integrating this simulation into our workflow, we ensured thorough testing of all software components in a controlled environment that replicates IGVC conditions. This approach prevents potential hardware damage and allows for iterative improvements.

4.5.2 **Feedback Loops:** Continuous data logging and monitoring systems provide feedback for iterative improvement, enabling quick adaptations to software strategies based on real-world performance.

4.6 Safety and Reliability

4.6.1 **Error Handling and Recovery:** Robust error handling mechanisms are integrated throughout the software stack. These mechanisms address potential failure modes identified during testing, such as sensor failure or data corruption, ensuring Shanti remains operational and safe. (See **Tables 3 and 4**)

4.6.2 **Redundancy:** Critical components, especially in the perception and navigation modules, have redundant systems to maintain functionality if one system fails.

Algorithm	Reliability Concern	Mitigation Method
lane-detection and pothole-detection	HSV thresholding is susceptible to change in lighting conditions.	Run parameter optimization algorithm for different lighting conditions.
image-to-laser	Incorrect camera configuration such as mounting angle.	Utilize the camera positioning optimization and ensure the 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 3. Algorithm - Reliability Concerns and Mitigation Methods.

Sensor	Reliability Concern	Mitigation Method
Logitech Cameras	Camera Failure, Dusty Lens	Use the spare camera, Wipe down lens with appropriate cloth
2D LiDAR	Lidar Failure	Use spare 2D LiDAR
IMU	IMU Failure	Use spare IMU
GPS	GPS Failure	Use spare GPS
Hoverboard	Hoverboard Failure	Use spare wheels and board to troubleshoot

Table 4.Sensor - Reliability Concerns and Mitigation Methods.

4.7 Failure Modes and Mitigation Methods

The failure modes can be found in various areas of code, but by identifying the potential level of where the failure modes may occur, we will be able to come up with possible solutions (**Table 5**).

Failure Level	Failure Mode	Mitigation Method
lane-detection	Side lanes were not being seen	Added three-camera configuration
lane-detection	Fails to detect lanes in quickly changing lighting conditions	Run optimization for HSV or use YOLOP.
Image-to-laser	Lane-scan is inaccurate due to change in camera position and/or orientation	Run optimization for camera positioning or publish ERROR.
Image-to-laser	Orientation of cameras not accurate	Used AprilTags to get exact transformations
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 a low confidence metric. Initiate recovery behavior.
dynamic-window-approach	Determines infeasible path	Initiate recovery behavior. Robot backs up.

Table 5. Software Failure Modes and Mitigation Methods.

5. Analysis

5.1 Initial Testing Results

Speed and Reaction Time

- A maximum time of 0.1 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 0.25 seconds.

Ramp Climbing Ability

- The robot (without payload) can climb a ramp of 25° inclination, which exceeds the specified 15° inclination.

Battery Life

- During 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 Zed 2i pitch down angle at 45° to see 3 meters ahead of the robot.

Image-to-Laser Matching

- Outdoor testing and parameter tuning enabled matching of laser scans for obstacles and lanes in bird's-eye view, with ground truth location of the obstacles and lanes.

Obstacle Detection Range

- Testing in simulation and 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 x 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 3cm inaccuracy for every 100 cm traveled.

GPS Accuracy

- Outdoor tests revealed electromagnetic interference from the Zed 2i magnetometer. With the Zed 2i power cable shielded, the GPS reported positions accurate within 10cm.

Recovery Behavior

- Testing in simulation showed that the robot is capable of stopping, backing up, and moving forward when it inadvertently gets too close to an obstacle.

5.2 Lessons Learned

This year, we learned the importance of rigorous testing and iterative design. Early simulations and field tests revealed critical issues that, when addressed, enhanced the robot's stability and navigational accuracy. One key point that we learned about was the importance of optimization techniques in solving accuracy issues with the robot. It was nice to see the use of esoteric mathematical techniques to solve for physical constraints and errors. It truly was where theory meets practice.

5.3 Top hardware failures and potential solutions

The most significant hardware failures included the misalignment of caster wheels and the instability of the sensor arm. To address the caster wheel issue, we repositioned them to improve lateral stability and added an additional caster wheel to the setup. For the sensor arm, we replaced the flimsy A-frame support with a reinforced 3D-printed shell, which also improved weatherproofing and housed essential control modules. Carrying spare materials, such as wood, aluminum, and extra wheels, has proven critical for on-the-spot repairs.

5.4 Software-In-Loop Virtual Environment

The Software-in-the-Loop (SIL) virtual environment allows us to test our algorithms in a simulated setting before deploying them on the actual robot. This environment includes a detailed model of our robot and its sensors, as well as a virtual representation of the competition terrain. Using this setup, we can conduct thorough testing and optimization, significantly reducing the risk of failures during physical testing and competitions.

5.5 **Physical Testing to Date (Predictions versus Actual)**

Our physical testing has revealed discrepancies between predicted and actual performance, particularly in navigation and obstacle detection. While simulations predicted high accuracy in lane detection, real-world tests showed variability due to lighting conditions and sensor alignment. Adjustments made based on these findings, such as optimizing the camera positioning and improving the robustness of the perception algorithms, have brought the actual performance closer to our predictions, but continuous refinement is necessary. We still have much more testing to do.