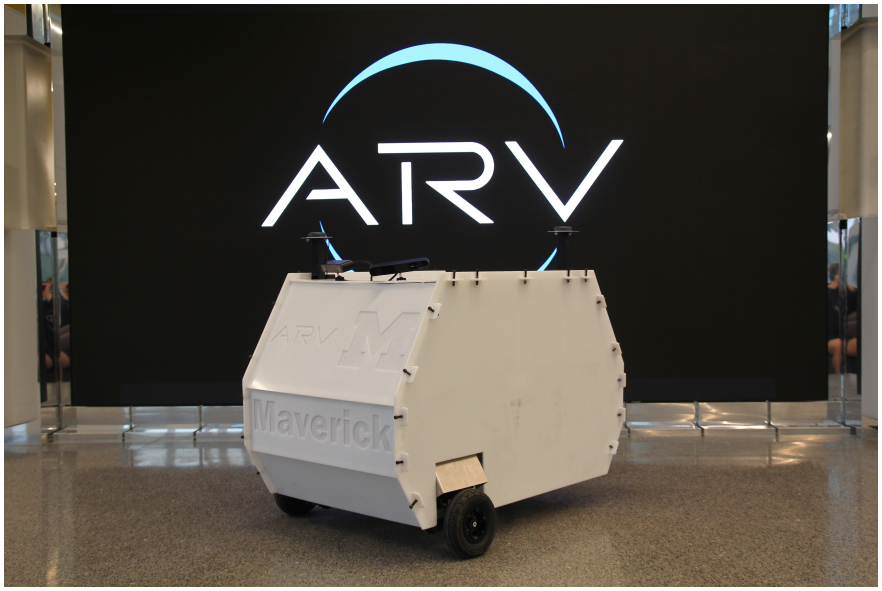
# Autonomous Robotic Vehicle Team
# University of Michigan




## Maverick

**Submitted:** May 15th, 2026

**Challenges:** AutoNav, Self-Drive
**Team Captain:** Matthew Gawthrop (mgawthro@umich.edu), (248) 918-7794
**Key Team Member:** Brinda Kapani (bkapani@umich.edu), (404) 408-1676
**Faculty Advisors:** Xiaoxiao Du (xiaodu@umich.edu), (573) 823-1086
Damen Provost (provostd@umich.edu)

**Statement of Integrity:** The design and engineering of the vehicle by ARV has been significant and equivalent to what might be awarded credit in a senior design course.

Xiaoxiao Du

| **Team Roster** (*Lead, **Assistant Lead) | | | | | |
|---|---|---|---|---|---|
| **Executive** | Matt Gawthrop | mgawthro@umich.edu | **Navigation & Sensors** | Ryan Liao* | ryanliao@umich.edu |
| | Eric Bi | ericbi@umich.edu | | Ethan Hardy* | hardyem@umich.edu |
| | Brinda Kapani | bkapani@umich.edu | | Caitlyn Trievel | ctrievel@umich.edu |
| **Platform** | Yuri Acauan Carnino* | ycarnino@umich.edu | | Erika Chen | erikachn@umich.edu |
| | Hazel Tao** | hazeltao@umich.edu | | Anna Novak | annanova@umich.edu |
| | Derrick Graves | derri@umich.edu | | Julian Whittaker | juliwhit@icloud.com |
| | Ryker Donkersloot | rykerd@umich.edu | | | |
| | Chris Riecker | criecker@umich.edu | **Business** | Nithin Reddy* | nkreddy@umich.edu |
| **Embedded Systems** | Katherine Shih* | katshih@umich.edu | | Skylar Scott | scskylar@umich.edu |
| | Sneha Das** | snehadas@umich.edu | **Computer Vision** | Maya Echtinaw* | mayaecht@umich.edu |
| | Chihyun An | jamesann@umich.edu | | Pranav Mallela** | pmallela@umich.edu |
| | Eric Barbieri | ericbarb@umich.edu | | Swati Sahu | sahusw@umich.edu |
| | Ruey Day | rueyday@umich.edu | | Aditya Suri | adsuri@umich.edu |
| | Dev Veeramani | devv@umich.edu | | Thuong Vo | vothuong@umich.edu |
| | Isaac Song | isaacso@umich.edu | | Evan Hsueh | ehsueh@umich.edu |
| | Ash Wu | awand@umich.edu | | Max Maley | maxmaley@umich.edu |
| | Bany Huang | bany@umich.edu | | Arnav Devalapally | darnav@umich.edu |
| | Brojananda Paul | bropaul@umich.edu | | John Omega | johnom@umich.edu |
| | Yi Keen Lim | yklim@umich.edu | | Alex Dumitrascu | agdumi@umich.edu |
| | Hannah Hysell | hannerd@umich.edu | | | |

# 1 System and Subsystem Requirements

Maverick's requirements were developed through a top-down process. IGVC rules and lessons from prior competitions were translated into measurable engineering targets that guided subsystem design. For each requirement, we state its motivation, measurement method, and target value.

## 1.1 Core Vehicle Requirements (Mechanical, Safety, Electrical)

| Requirement & Motivation | Measurement Method | Target Metric |
|---|---|---|
| *Mechanical*<br>**Drivetrain torque on inclines:** The drivetrain must produce enough torque to climb the steepest course incline without stalling. *Motivation:* IGVC permits ramps up to 15% grade, and the vehicle must hold $\geq$1 mph average over the course. | Drive up a 15° ramp; log pitch via the onboard IMU and velocity via motor encoders. | 1–5 mph continuous speed on the ramp, no stalling. |
| **Vehicle dimensions:** The fully assembled vehicle must fit within the IGVC dimensional envelope. *Motivation:* Section I.2 requires 3–7 ft length, 2–4 ft width, <6 ft height (excluding E-stop antenna). | Measure footprint and total height of the assembled robot with a tape measure. | 3–7 ft L, 2–4 ft W, <6 ft H. |
| **Payload retention:** The 20 lb payload must remain securely mounted across the full speed range. *Motivation:* Section I.3 requires a 16"×8"×8", 20 lb payload; if it falls off, the run is terminated. | Drive across the full 1–5 mph speed range with the payload installed; visually monitor for displacement. | <0.5" displacement. |
| **Weatherproof enclosure:** Electronics must be protected from water ingress during operation. *Motivation:* Section II.3 allows runs in light rain or drizzle. | Physical fit check of the assembled HDPE cover; visual inspection of water shedding. | Full electronics coverage with 7/8" oversize tolerance. |
| *Safety*<br>**Wireless E-stop range and latency:** The wireless E-stop must bring the vehicle to a quick, complete stop from at least 100 ft range. *Motivation:* Section I.2 requires effectiveness at a minimum of 100 ft. | Trigger the wireless E-stop at measured intervals up to 100 ft; log the time delay from trigger to full stop. | Complete stop within <500 ms from 100 ft. |
| **Mechanical E-stop:** A hardware-based push-to-stop button must bring the vehicle to a quick, complete stop. *Motivation:* Section I.2 requires a center-rear, non-software E-stop, red, $\geq$1" diameter, 2–4 ft above ground. | Press the mechanical E-stop while driving at 5 mph; measure stopping time and distance. | Complete stop within <300 ms and <2 ft. |
| **Safety light state indication:** The vehicle's status LED must clearly communicate its operational state. *Motivation:* Section I.2 mandates a solid light when powered and flashing in autonomous mode; we extend this with color states for clearer human-facing indication. | Visually observe the LED through transitions between teleop, autonomous, estop triggered, and recovery states. | Correct color/flash pattern for every state. |
| *Electrical*<br>**Onboard power runtime:** Onboard batteries must power all sensors, compute, and motor controllers for a full competition day at the voltage the ODrives require. *Motivation:* Section I.2 mandates onboard power generation. | Run the robot continuously while logging runtime and bus voltage via the Power Distribution PCB's power monitor. | $\geq$8 hr runtime without main power rail dropping below 26 V. |
| **Speed envelope and smooth acceleration:** Velocity must stay within the IGVC band, and acceleration must be gentle enough to prevent tip-over. *Motivation:* Section I.2 enforces a 1 mph minimum average and a hardware-governed 5 mph maximum. | Log encoder velocity across a full test run; compute mean, peak, and time-derivative (acceleration). | 1–5 mph velocity envelope; peak acceleration <1.0 $m/s^2$. |

## 1.2 AutoNav Requirements (Perception, Driving Logic, KPIs)

| Requirement & Motivation | Measurement Method | Target Metric |
|---|---|---|
| *Perception*<br>**General obstacle detection:** The vehicle must detect and localize unmapped physical obstacles protruding ≥10 cm above the ground plane within its field of view, in near real-time. *Motivation:* Section II.2 places randomly colored barrels and natural obstacles along the course; no course memorization is allowed. | Place unmapped obstacles within the 5 m × 5 m field of view; verify correct rendering on the occupancy grid and measure the offset between detected obstacle centroid and true centroid. | ≥95% detection rate; ≤5 cm centroid localization error; <50 ms end-to-end latency per camera frame. |
| **Lane line and pothole detection:** The vehicle must reliably detect 3" white lane lines and 2 ft potholes within its field of view. *Motivation:* Section II.2 ends the run if the vehicle crosses a lane or drives over a pothole. | Place lane lines and potholes within the 5 m × 5 m field of view; compare detected boundaries on the occupancy grid to ground truth and measure lateral offset (lane lines) and centroid offset (potholes). | ≥95% detection rate; ≤5 cm localization error; <50 ms end-to-end latency per camera frame. |
| *Driving Logic*<br>**Waypoint navigation:** The vehicle must autonomously reach each GNSS waypoint while avoiding obstacles and staying in-bounds. *Motivation:* Section II.2 provides waypoint pairs that must be reached within a 2 m radius. | End-to-end autonomous runs on a physical test course with mock waypoints, lanes, and obstacles. | ≥95% success rate to waypoint. |
| **Failure recovery:** The vehicle must autonomously detect and escape from dead-ends, traps, and blocked paths. *Motivation:* Section II.4 disqualifies the run if the vehicle blocks traffic for over one minute. | Intentionally block the planner's path during physical tests; time how long until a new path is generated and motion resumes. | New path in <45 s; ≥95% recovery success. |
| *KPIs*<br>**Average speed and variability:** The vehicle must hold a consistent speed inside the IGVC band. *Motivation:* Section II.2 enforces a 1 mph minimum (checked in the first 44 ft) and a 5 mph maximum. | Continuously log encoder velocity over a full run; compute mean and standard deviation (excluding recovery segments). | Mean 1.5 mph with ±0.5 mph std. dev. |
| **Global GNSS waypoint accuracy:** The localization stack must place the robot accurately enough in the global frame to consistently hit waypoints. *Motivation:* Section II.2 defines a 2 m radius as a successful waypoint hit; tighter accuracy gives margin against GNSS jitter. | Place a physical marker at a known surveyed GNSS coordinate, drive the robot to it autonomously, and measure the residual distance with a tape measure. | ±1 m global accuracy. |

## 1.3 Self-Drive Requirements (Perception, Driving Logic, KPIs)

| Requirement & Motivation | Measurement Method | Target Metric |
|---|---|---|
| *Perception*<br>**Lane color discrimination:** The perception system must reliably distinguish yellow dashed lane lines from solid white lane lines. *Motivation:* Section III's rules-of-the-road logic depends on lane color (e.g., solid white = no-cross); misclassification directly causes line-crossing failures. | Compute the ratio of intersecting pixels between the yellow and white lane masks within the final occupancy grid. | < 5% overlapping pixels. |
| **ML object identification:** The vision system must detect pedestrians, tires, and potholes with high confidence. *Motivation:* Section III.6 functions tests (FI–FII, FV) require detection of these objects in static, obstructed, and dynamic scenarios. | Extract native YOLOv8 confidence scores during functions-test rehearsals. | >75% confidence threshold. |
| *Driving Logic*<br>**Solid lane boundary compliance:** The vehicle must never cross a solid white line. *Motivation:* Section III.7 ends the run if two front wheels cross a solid white line; line excursions also cost 25-point penalties per function. | Run 3 left and 3 right turns; count wheel crossings of solid white lines per function via video review. | 100% compliance (zero crossings). |

| Requirement & Motivation | Measurement Method | Target Metric |
|---|---|---|
| **Intersection turn execution:** The vehicle must complete left and right turns at intersections efficiently and into the correct lane. *Motivation:* Tests FIII.2–FIII.3 require stopping at the stop bar, turning, and merging into the correct lane; turn time drives the Full Course completion budget. | Compare autonomous turn completion time against an optimal 3 mph teleoperated baseline on the same intersection. | Autonomous turn ≥75% as efficient as optimal teleop (i.e., ≤1.33× teleop time). |
| *KPIs*<br>**Turn efficiency:** The vehicle must clear each intersection turn quickly enough to leave time for the rest of the Full Course. *Motivation:* Section III.7's Full Course chains 21 functions at 100 points each; per-turn time directly affects completion. | Stopwatch from one stop-line to the next during intersection-turn functions; cross-reference encoder logs. | <20 seconds per turn. |
| **Velocity envelope consistency:** The vehicle must hold the function-test target speed band reliably. *Motivation:* Function tests (FIII–FV) specify 3–5 mph target speeds; deviations risk qualification failure (too slow) or stop-line overshoot. | Log encoder velocity across function-test runs; compute percentage of run time in-band. | Velocity capped at 5 mph; majority of run time in 3–5 mph. |

## 1.4 System Architecture Evolution

After establishing our key requirements, we reviewed our past competition performances and identified several major shortcomings in our previous designs that had the biggest negative impact on performance. These requirements and findings guided this year's design and led to the AutoNav and Self-Drive architectures shown below.

### 1.4.1 AutoNav Architecture

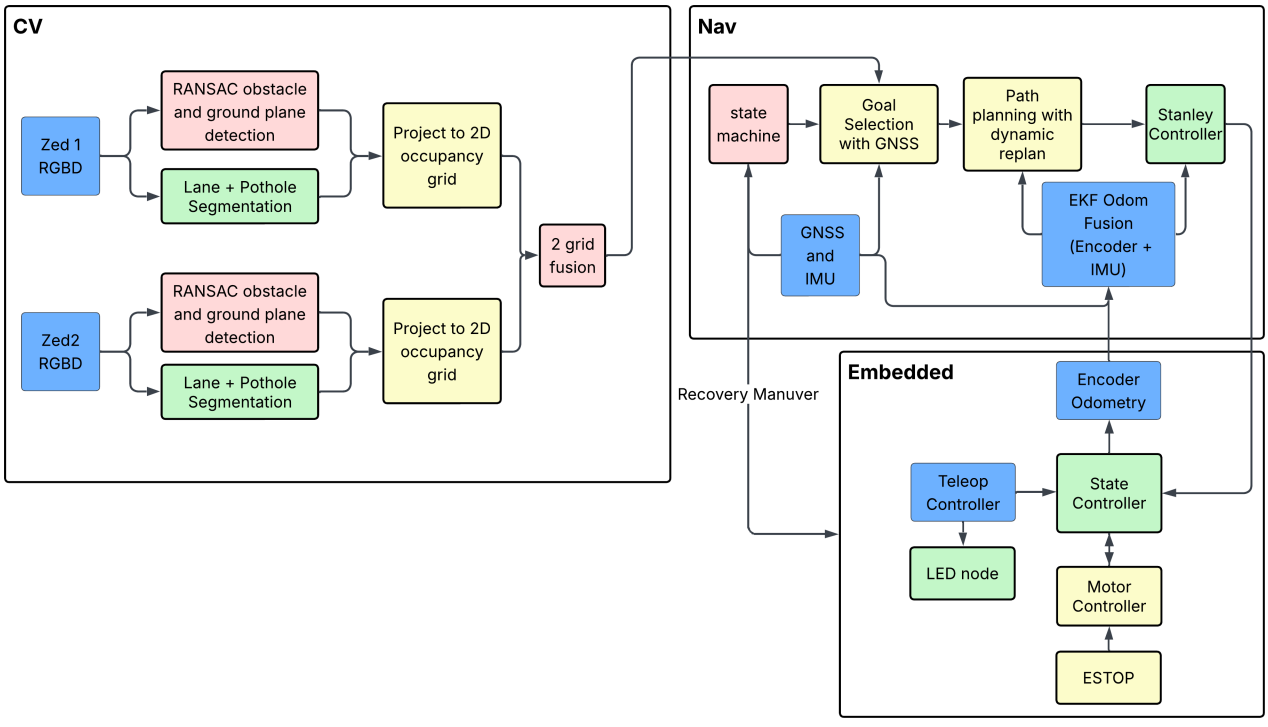



Figure 1: AutoNav system architecture across computer vision, navigation, and embedded systems.

Our AutoNav architecture is a custom-built stack designed to process camera data, localize the robot, and drive the motors across three main subsystems:

- **Computer Vision:** Two RGBD ZED 2i cameras process raw RGB and depth data, detecting obstacles and packaging them into a 5m by 5m 2D occupancy grid.
- **Navigation:** This subsystem localizes the robot using an Extended Kalman Filter (EKF) to fuse VectorNav VN-300 IMU data with wheel odometry. GNSS waypoints are used for goal selection, while a state machine manages navigation behavior. An A* planner routes through the occupancy grid, and a Stanley Controller outputs velocity and steering commands.
- **Embedded Hardware:** This stack manages physical execution and safety systems, including motor command execution, the LED state indicator, remote and physical E-stops, and system power distribution.

### 1.4.2 Self-Drive Architecture

This year we decided to focus our self-drive efforts towards the qualification and functional tests so that we could refine our techniques and master the most important aspect of the challenge. Our Self-Drive stack is specifically designed to make the self-drive challenge look like a normal autonav run, allowing us to reuse the entire downstream navigation and embedded pipeline. When lane lines are sparse we draw them in to provide autonav-like boundaries, and our waypoints are transformed into the global navigation frame so they appear as GNSS coordinates. Doing this ensures that path planning, EKF state estimation, and the Stanley Controller remain invariant across competition modes, significantly reducing technical debt.

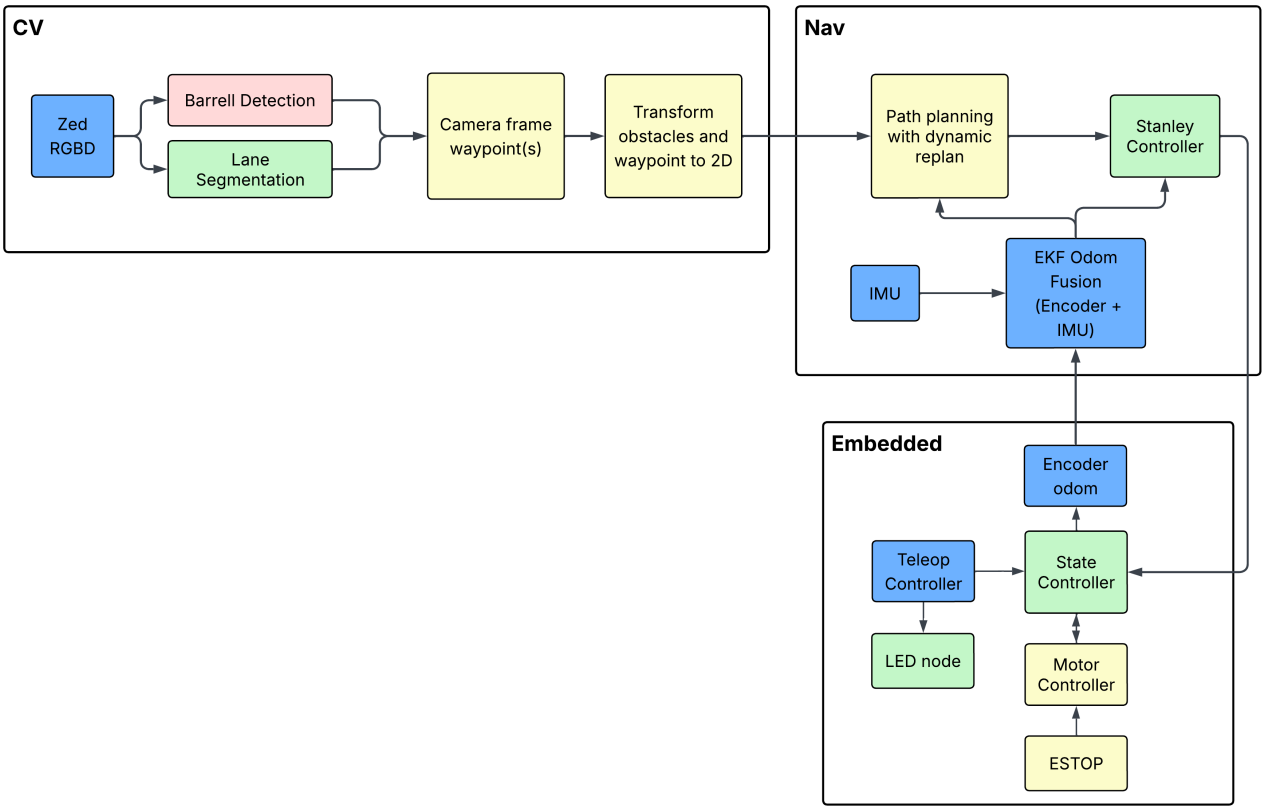



Figure 2: Self-Drive system architecture across computer vision, navigation, and embedded systems.

# 2 Mechanical Design

## 2.1 Design Overview

Maverick's chassis is a low-slung, octagonal aluminum frame chosen to maximize the internal electronics footprint while staying within the IGVC dimensional rules. The vehicle is differentially driven by two front-mounted brushless motors with a free-rolling rear caster, and is enclosed by a removable High-Density Polyethylene (HDPE) cover that provides weatherproofing.

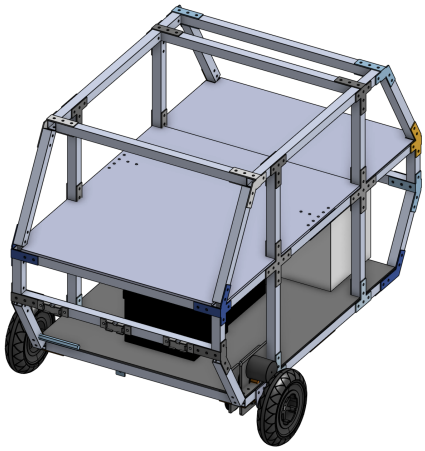


Figure 3: Full CAD model of Maverick.

## 2.2 Significant Mechanical Components

This section outlines the key mechanical components of the robot and explains the design decisions behind each subsystem. These components were selected and configured to balance stability, performance, accessibility, and sensor reliability.

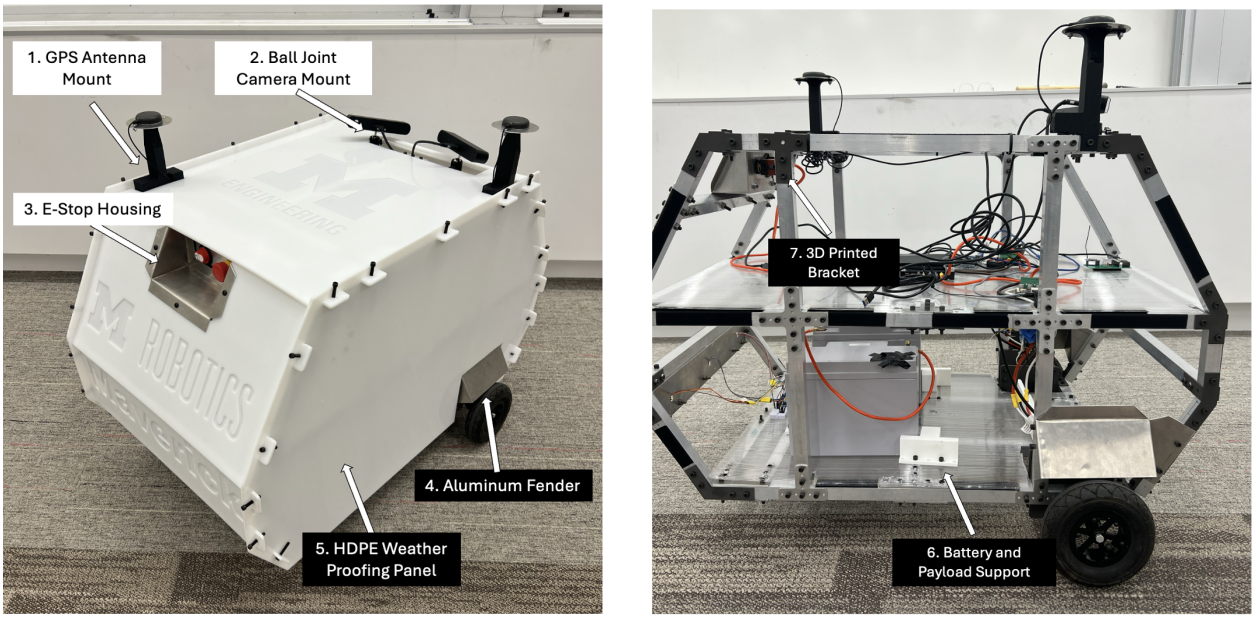



Figure 4: Major Physical Design Features of Maverick

### 2.2.1 Gearbox and Drivetrain

The gearbox and drivetrain operate with an overall gear ratio of 9:170, achieved through two stages: a first stage ratio of 1:5 and a second stage ratio of 9:32. This relatively high gear reduction was intentionally chosen to prioritize torque output over speed, directly addressing our *drivetrain torque on inclines* requirement. The high reduction provides enough torque headroom to climb the steepest course incline comfortably while keeping the top speed safely under the 5 mph cap. The reduction additionally provides smoother accelerations and more controlled movement, enhancing overall stability.

The drivetrain consists of two motor-driven front wheels and a caster wheel at the rear. This configuration simplifies control while maintaining maneuverability. The gearbox itself is constructed using two primary plates that constrain the internal components, ensuring consistent alignment. The entire assembly can be seen in figure 5.

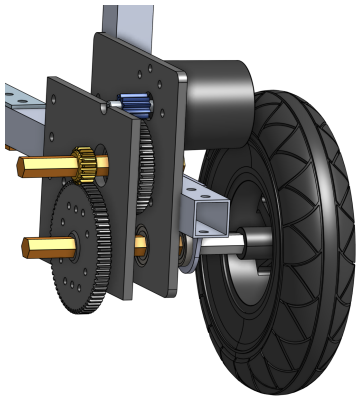


Figure 5: The gearbox assembly attached to the frame.

### 2.2.2 Camera Mount

The robot utilizes two cameras mounted on the front left and right sides of the frame. Each camera is mounted using a ball joint mechanism (Figure 4.2), which allows for fine angular adjustments during setup and calibration. Throughout our full scale testing we have found that these mounts hold their position consistently for 5+ hours. This allows for our computer vision algorithms to run smoothly without re-calibration.

### 2.2.3 Antenna Mount

The antenna mount is composed of two main components: the antenna riser and the mounting plate. The riser is fully 3D printed from PLA and elevates the GNSS antennas approximately 6 inches above the robot's frame to help avoid electromagnetic interference (Figure 4.1). The mounts reduced our average GNSS error from 0.82m to 0.7m.

### 2.2.4 Frame Shape and Brackets

The current frame is a significant reduction in height compared to previous years and comfortably meets the IGVC dimensional limits (3–7 ft long, 2–4 ft wide, <6 ft tall), satisfying our *vehicle dimension* requirement with margin on every axis. We designed the robot to be octagonal in shape to increase the amount of space available for electronics on the middle level of the vehicle. To accomplish this, we designed and 3D printed brackets to support various connections of the vehicle. The vehicle utilizes an octagonal frame design, which was specifically chosen to maximize the internal surface area for electronics mounting on the middle chassis level. To hold the robot

together we chose to go with 2 different bracket materials to save on weight without sacrificing any structural integrity:

- 3D Printed Components: In areas of lower mechanical stress, high-strength 3D printed brackets were used to secure complex connections (Figure 4.7).
- Aluminum Components: On load-bearing joints, some pre-ordered and some custom made aluminum brackets were used. These provide the necessary rigidity to support the heavy electronics and payload through all maneuvers.

### 2.2.5 Payload and Battery

As with last year's design, the payload and battery are stored in the bottom level of the vehicle. This low center of gravity keeps the vehicle stable throughout the course, especially on ramp sections where the robot has the potential to tip over — directly supporting our *payload retention* requirement of <0.5" displacement across the full 1–5 mph speed range. To keep the batteries and payload in place, we designed and 3D printed supports that were screwed into the bottom sheet of plastic (Figure 4.6).

## 2.3 Weatherproofing

To satisfy the *weatherproof enclosure* requirement, a shed-roof style cover was developed using 1/4" HDPE and aluminum sheet. The design features 1/2" overhangs and a 7/8" oversize tolerance to get a consistent seal over the electronics.

### 2.3.1 Joinery and E-stop Protection

The enclosure utilizes a mortise-and-tenon system with 1/8" depth dados, secured by 1/4" socket cap screws. This joinery provides high mechanical strength and the tight panel-to-panel tolerances necessary to prevent water entry.

Because the manual E-stop requires an external button, we added a dedicated E-stop housing shield (Figure 4.3). This works to capture and redirect any water that may get stuck in the E-stop area.

### 2.3.2 Fenders

To protect the drivetrain, custom fenders were bolted directly to the frame (Fig. 4.4). These ensure the gearboxes and motors remain shielded from runoff and debris both during operation and while the main HDPE cover is removed for maintenance or transport.

## 2.4 Requirement Validation

The table below compares each mechanical requirement against the most recently measured actual value. Together, these results indicate that Maverick's mechanical design meets or exceeds every mechanical requirement derived from the IGVC rules.

| Requirement | Target | Measured Actual | Status |
|---|---|---|---|
| Drivetrain torque on inclines | 1–5 mph on 15° ramp, no stalling | 2.71 mph sustained on 15° ramp, no stalling observed | Success |
| Vehicle dimensions | 3–7 ft L, 2–4 ft W, <6 ft H | 3.3 ft L × 2 ft W × 3.05 ft H | Success |
| Payload retention | <0.5" displacement across 1–5 mph | 0.43" max displacement observed across full speed range | Success |
| Weatherproof enclosure | Full electronics coverage with 7/8" oversize tolerance | Full coverage verified; 2/2 water-shedding trials passed | Success |

Table 4: Mechanical requirement validation summary.

**Discussion.** The drivetrain validation confirms that the 9:170 gear reduction provides sufficient torque margin to climb the steepest IGVC-permitted incline without stalling, which was the primary driver of the gearbox design. The dimensional measurements show the octagonal frame sits comfortably inside the IGVC envelope on all three axes, leaving room for the antenna mast to extend above the chassis without violating the 6 ft height limit. Payload retention testing showed the 3D-printed supports hold the cinder-block payload firmly even under hard acceleration and ramp transitions, with no significant displacement across the 1–5 mph range. Finally, the weatherproofing cover sheds water effectively in light-rain conditions, with the fenders and e-stop shield closing the two gaps that the main HDPE cover does not cover by itself.

# 3 Safety

## 3.1 Transport, Parking, and Charging Safety

Maverick includes several safety measures for transport, parked operation, and battery charging. The robot cannot be driven unless the E-Stop is in the loop; the remote E-Stop defaults to the triggered state and must be intentionally disengaged before operation. To reduce electrical risk, all exposed wire ends are covered with heat-shrink tubing, terminal connections are screwed in securely, and wires are mechanically secured to prevent loose connections or accidental contact. During operation, all battery leads are taped to prevent accidental shorts. For charging, we use a 20A charger that screws directly into the battery terminals, reducing the risk of disconnection during charging. This keeps the charging current well below the battery's 50A maximum charging current rating. We chose LiFePO4 batteries because they are safer and more stable than many other lithium battery chemistries.

## 3.2 Operational Safety: Mechanical E-Stop

Main battery power routes through a 50 A circuit breaker on the embedded shelf, protecting downstream components from short circuits. This power also flows through an easily accessible, rear-mounted E-stop. When pressed, this hardware-based switch physically breaks the main circuit to the motor controllers and stops our robot.

## 3.3 Wireless E-Stop

The wireless E-Stop system uses a custom PCB with an integrated ESP32 and LoRa module for long-range wireless communication. This subsystem addresses our *Wireless E-stop range and latency* requirement: the remote must reliably stop the vehicle from at least 100 ft away, with low enough end-to-end latency to keep the vehicle within the expected safety margin. LoRa was chosen over Bluetooth or Wi-Fi because it is more reliable over long distances in outdoor environments. Rather than using a standard breakout board, we integrated the LoRa front-end onto PCB to improve signal integrity through more efficient routing and ground returns. The receiver is powered through the laptop's USB port, keeping it electrically isolated from the 26V motor bus. The remote E-Stop stops the vehicle through software: the receiver sends a signal to the laptop, and the embedded stack responds by commanding zero velocity to the motor controllers.

## 3.4 Safety Light State Indication

To satisfy the *safety light state indication* requirement, an Arduino-driven RGB LED strip on the chassis communicates the vehicle's operational mode. The strip changes patterns based on ROS messages published by the navigation node. While the competition rules only require basic solid and flashing states, we implemented a custom color mapping to provide more visual feedback of

the robot's status for debugging:

| State | LED Pattern |
|---|---|
| Teleop | Solid blue |
| Autonomous | Flashing blue |
| No-man's land (autonomous) | Flashing green |
| Recovery mode | Flashing yellow |
| E-stopped | Solid red |

Table 5: Safety light color mapping.

### 3.5 Requirement Validation

| Requirement | Target | Measured Actual | Status |
|---|---|---|---|
| Wireless E-stop range & latency | Complete stop <500 ms from 100 ft | 300 ms stop time at 300 ft over 10/10 trials | Success |
| Mechanical E-stop | Complete stop <300 ms and <2 ft at 5 mph | 4 s / 7 ft stopping distance at 5 mph | Failed |
| Safety light state indication | Correct color/flash pattern for every state | All 5 state transitions verified visually | Success |

Table 6: Safety requirement validation summary.

**Discussion.** The wireless E-Stop meets the 100 ft range requirement with a large margin and stops the vehicle well under the 500 ms latency target. The mechanical E-Stop successfully cuts power to the motor controllers, but the stopping distance did not meet our target because removing motor power also removes active braking, causing the vehicle to coast before coming to a full stop. This result shows that the mechanical E-Stop path is reliable as a hardware power cutoff, but future iterations should add a hardware braking path or braking resistor so the robot can stop more quickly after power is interrupted. The safety light's state-machine implementation was validated through every operational transition without errors, allowing observers to identify the robot's mode at a glance.

# 4 Electrical/Electronic Design

## 4.1 Significant Power-Consuming Components

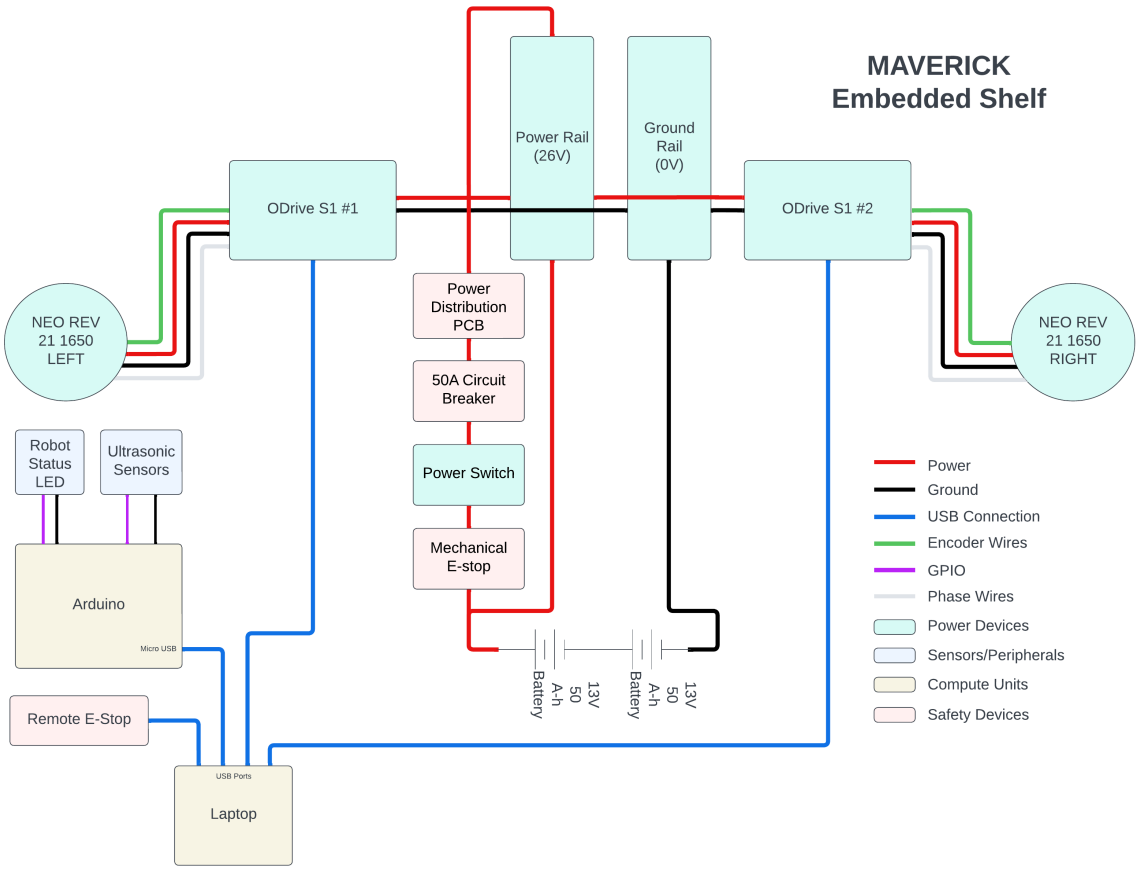



Figure 6: Electronic component and power diagram.

Maverick uses two ODrive S1 motor controllers and an Arduino Mega for low-level control. It is powered by two 13V batteries connected in series, and power is distributed to the robot's components through the Power Distribution PCB. A laptop is connected to the two ODrives, the Arduino, and the wireless E-Stop receiver. The 26V power rail drives the ODrives and runs through the mechanical E-Stop, so pressing it physically cuts power to the motor controllers. The wireless E-Stop uses a separate software path through the laptop, as described in the Safety section. Each ODrive is connected to a NEO brushless motor to control Maverick's left and right wheels. The Arduino controls an RGB LED strip that indicates the robot's state based on ROS messages, and it also reads three ultrasonic sensors used for recovery behavior to avoid backing into obstacles.

## 4.2 Power Distribution and Monitoring

Our Power Distribution PCB safely distributes power to the two ODrives and the STM32L412KBT6 microcontroller onboard. It also logs system voltage and current draw and includes an LED display for real-time power monitoring. This board, together with the external 50A fuse and main E-Stop, helps us meet our *onboard power runtime* requirement of $\geq$8 hr at 26V without dropping below the ODrives' minimum bus voltage. The external 50A fuse, power switch, and mechanical E-Stop

are connected directly to the main power path, while the remote E-Stop and Arduino are powered separately through the computer.

The power distribution system is sized based on logged power consumption data, with two 50Ah 12.6V batteries connected in series providing the 26V rail required by the ODrives. The PCB's voltage and current logging are used to validate the runtime requirement and detect abnormal power conditions during operation. Each battery also has an internal Bluetooth monitoring system that provides additional voltage, current, discharge, and temperature data, giving us a redundant way to monitor the power system and cross-check the measurements logged by the PCB. With our 20A charger, the two 50Ah batteries can be recharged in approximately 2.5–3 hr.

### 4.3 Speed Envelope and Smooth Acceleration

To satisfy the *speed envelope and smooth acceleration* requirement, Maverick utilizes a dual-layer velocity cap. The ODrives are firmware-locked to the 5 mph IGVC maximum, while the Stanley Controller limits commands within the 1–5 mph range. Mechanical gear reduction further attenuates jerk, keeping peak acceleration safely below the vehicle's tip-over threshold.

For smoothness, we tuned per-motor PI gains by analyzing real-time velocity traces in the ODrive GUI. By iteratively minimizing overshoot and steady-state error, we achieved a clean step response without oscillation or ringing. This ensures the velocity profile tracks controller commands precisely across all operating speeds without acceleration spikes.

### 4.4 Rear Obstacle Detection

To support safe recovery behavior, Maverick uses three HC-SR04 ultrasonic sensors mounted across the rear of the robot. The sensors are spaced apart to provide rear obstacle coverage and detect objects when the robot is backing up.

The Arduino reads the ultrasonic sensors and streams the distance measurements serially to the laptop. These readings allow the software stack to customize recovery behavior and stop the robot when an obstacle is detected within a defined distance threshold.

### 4.5 Requirement Validation

| Requirement | Target | Measured Actual | Status |
|---|---|---|---|
| Onboard power runtime | $\geq$8 hr, rail $\geq$26 V | 20+ hrs continuous power at >26 V | Success |
| Speed envelope & smooth acceleration | 1–5 mph; peak accel <1.0 $m/s^2$ | Mean 1.21 mph, peak 1.96 mph; peak accel 0.36 $m/s^2$ | Success |

Table 7: Electrical/Electronic requirement validation summary.

**Discussion.** The runtime test was performed end-to-end with all sensors, compute, and motor controllers active, and the 26 V rail never dropped below the ODrives' minimum bus voltage during the run. The velocity logs confirm the vehicle stays within the IGVC 1–5 mph envelope at all times, with the firmware velocity cap enforcing the 5 mph upper bound and the Stanley Controller keeping commanded velocities within the allowed range during normal operation. Peak acceleration stayed comfortably under the threshold due to PI tuning and the high gear reduction.

# 5 Perception

## 5.1 System Overview and Shared Components

The backbone of our perception stack across both challenges utilizes RGB-D cameras as a pseudo-LiDAR system. A RANdom SAmple Consensus (RANSAC) algorithm fits a best-fit ground plane equation directly to the camera's depth image, and any pixel whose depth lies above the plane by more than a tunable threshold is flagged as a protruding physical obstacle. The flagged pixels are then projected into 3D using their depth values and camera intrinsics, and binned into a bird's-eye-view occupancy grid. This allows us to meet our general obstacle detection requirement without relying on custom machine learning models or brittle color filters for every new obstacle variation. Because the algorithm recomputes the ground plane equation every frame, the entire stack is independent of the camera's angle of depression, requiring no software re-tuning if the cameras are physically re-aimed.

## 5.2 Internal Representation of the Environment

The driving logic relies on an internal representation of the environment structured as a binary 2D occupancy grid. This grid represents a fixed 5 m $\times$ 5 m physical area directly in front of the vehicle with a resolution of 5 cm per cell, populated with 0s (free space) and 1s (obstacles or boundaries). To maintain a simplified representation, this grid is continuously populated by fusing the protrusion data with specific feature masks generated by our color-filtering and machine-vision pipelines. We keep each occupancy grid independent and do not stitch them together in consecutive iterations.

## 5.3 AutoNav Perception Specifics

To achieve a wider field of view capable of viewing obstacle layouts from lane to lane, Maverick utilizes a two-camera setup. This ensures the vehicle perceives the entire width of the course, directly contributing to our *general obstacle detection* requirement. Each camera runs on an independent ROS node, feeding a 2D occupancy grid into a central fusion node that combines them using a fixed, pre-tuned transformation.

Predictable course features such as lane lines and potholes are isolated using a custom HSV filtering library, supporting our *lane line and pothole detection* requirement. A unique edge case is the ramp obstacle, which RANSAC correctly identifies as a protrusion; to allow the robot to drive over it, we utilize HSV filtering to identify its specific hue, actively masking it out of the final occupancy grid. To prevent this mask from accidentally removing barrels in no-man's-land, the ramp-masking logic is disabled in that section. Because the ramp is masked at the perception layer, the driving logic requires no special-case handling to climb it.

## 5.4 Self-Drive Perception Capabilities

For Self-Drive, the perception stack utilizes our custom HSV library to create a binary mask for yellow and white lane lines, addressing our *lane color discrimination* requirement. Advanced obstacles are handled by custom YOLOv8 models—trained using team-gathered, annotated datasets addressing our *ML object identification* requirement.

### 5.5 Requirement Validation

| Requirement | Target | Measured Actual | Status |
|---|---|---|---|
| General obstacle detection (AutoNav) | ≥95% rate, ≤5 cm error, <100 ms latency | 99% rate, <5 cm error, 50-60 ms latency | Success |
| Lane line & pothole detection (AutoNav) | ≥95% rate, ≤5 cm error, <50 ms latency | 97% rate, <5 cm error, 6-11 ms latency | Success |
| Lane color discrimination (Self-Drive) | < 5% pixel overlap between white & yellow masks in final occupancy grid | Max overlap over 5 trials driving around an intersection: 0% | Success |
| ML object identification (Self-Drive) | >75% confidence on pedestrians, tires, potholes | Pedestrian: 86.27%; Tire: 87.03%; Pothole: 82.43% | Success |

Table 8: Perception requirement validation summary.

**Discussion.** The 2-camera + RANSAC approach generalizes across lighting conditions without ML retraining, which is what gives us our high detection rate on general obstacles. Low-gamma camera settings improved HSV reliability for lane and pothole detection. The lane color discrimination overlap is well below the 5% threshold needed to enforce rules-of-the-road logic without false-positive line-cross penalties, and the YOLO models exceed the 75% confidence threshold across all required Self-Drive object classes.

## 6 Driving Logic

### 6.1 AutoNav Driving Logic

**State Machine.** All navigation nodes subscribe to a shared latched `state` topic managed by a central state machine with four states: Normal, No Man's Land, Recovery, and Finished. States are priority-ordered from lowest to highest; subsystems request transitions via ROS services, and the state machine resolves conflicts and publishes only on change, ensuring every node always operates in a consistent mode.

**Localization.** We use a local EKF that fuses encoder linear velocity and IMU yaw rate to produce an odometry-to-robot transform. Since this drifts over time, we correct it using the VN-300 INS. We continuously compute the offset between the INS global position and the EKF's local odometry and apply it as a correction. GNSS waypoints are converted from WGS84 to ENU, then transformed into the local frame using this offset.

We rely solely on the VN-300 for global positioning for two reasons. First, it derives heading from dual GNSS receivers at a known offset rather than a magnetometer. This is critical given heavy magnetic disturbances in our test environment and is also well-suited to IGVC's open-air conditions. This alone achieves <1m waypoint accuracy without additional sensor aid. Second, its onboard multi-sensor fusion delivers high accuracy out of the box, simplifying our architecture and minimizing tuning effort.

**Goal Selection via Ray-Casting.** Periodically, the goal-selection node ray-casts across the robot's forward arc. Each ray is scored by its free length weighted by its alignment toward the current GNSS waypoint, so the selector always biases forward progress toward the target. A smoothing window reduces score noise across adjacent rays. If the best ray falls below a minimum progress threshold, no goal is returned and the state machine transitions to RECOVERY.

**Obstacle and Lane Avoidance via Inflation.** Before path planning consumes the occupancy grid, a preprocessing step inflates every occupied cell using a hard exclusion radius followed by an

exponential soft-cost falloff, and adds an occupied border along grid edges to discourage planning near boundaries. Because physical obstacles, lane lines, and potholes are all written into the same binary occupancy grid, inflation applies uniformly to every obstacle type without special-casing each one.

**A* Path Planning with String-Pulling.** Given a local goal, an A* planner routes from the robot's current position to that goal on the occupancy grid. If the goal cell is unreachable, A* routes to the closest reachable cell instead, ensuring the planner always makes some forward progress. The raw A* path is post-processed with a string-pulling algorithm that finds the furthest subsequent waypoint reachable in a straight line of sight and discards everything in between, removing staircasing artifacts from grid-based search and producing a smoother path for the controller to follow.

**Path Tracking via Stanley Controller.** The planned path is executed by a Stanley controller, which simultaneously corrects heading error and cross-track error. This produces smoother and more stable path following than Pure Pursuit, particularly on a robot with the inertia of Maverick. The controller also looks ahead along the upcoming path, estimates curvature from the accumulated heading change, and proactively caps speed before entering a curve rather than reacting after the fact. When the state machine is Finished, Path Tracking halts the robot.

**Recovery Behavior.** When the state machine enters Recovery, a dedicated node takes control. We researched established stuck-recovery techniques for differential-drive robots (backing up with obstacle-guarded reversal and in-place rotation to break out of local minima) and derived our own implementation from these principles. The node uses median-filtered rear ultrasonic readings to guard against reversing into unseen obstacles, and selects between a straight backup or a rotational sweep followed by backup depending on how much rear clearance is available. Once the maneuver completes, the state machine returns to Normal and goal selection resumes. We selected our configuration thresholds and timings based on our robot's physical dimensions and turning radius, and although we were not able to validate recovery in real-world conditions before the submission deadline, we are confident the implementation is sound.

Together, these components form a pipeline designed to meet our *waypoint navigation* and *failure recovery* requirements: GNSS-biased ray-casting keeps the robot progressing toward each waypoint, uniform inflation handles every obstacle type consistently, A* with string-pulling produces clean paths, and the Stanley Controller executes them smoothly. Recovery cleans up any path planning failures by ensuring the robot can escape the situations where the planner cannot find a forward path.

## 6.2 Self-Drive Driving Logic

Custom state machines analyze visual features such as yellow lane line positions and slopes to generate waypoints in the camera frame, which are then transformed into global coordinates. Simultaneously, naturally detected solid lane lines are mapped into the occupancy grid as rigid obstacles to strictly enforce our *solid lane boundary compliance* requirement.

In areas where these natural lines become sparse, such as open intersections, we draw artificial lane lines into the occupancy grid to guide the robot to the generated waypoint. As introduced in Section 1.4.2, this combination of global-frame waypoints and artificial constraints creates an abstraction layer that perfectly mirrors a standard AutoNav environment. This lets us to use the same path planning and tracking algorithms as AutoNav, allowing us to satisfy difficult requirements like *intersection turn execution* without much additional developmental and testing burden.

### 6.3 Requirement Validation

| Requirement | Target | Measured Actual | Status |
|---|---|---|---|
| Waypoint navigation (AutoNav) | ≥95% success rate | 100% success over 6 randomized 20 m long courses | Success |
| Failure recovery (AutoNav) | New path <45 s; ≥95% recovery success | N/A | Untested |
| Solid lane boundary compliance (Self-Drive) | 100% (zero crossings) | 1 crossing over 3 left and 3 right turn trials | Fail |
| Intersection turn execution (Self-Drive) | ≥75% of optimal teleop efficiency | Autonomous 16.14 s vs. teleop 14 s (88.61% efficiency) | Success |

Table 9: Driving Logic requirement validation summary.

**Discussion.** The waypoint-navigation success rate confirms the GNSS-aware goal selector and inflation layer work together effectively across the variety of obstacle layouts we tested. Recovery was implemented and tested in isolation but not validated end-to-end in real-world conditions before submission; we are confident in the implementation based on the design rationale described in Section 6.1, and full validation is the immediate next testing priority. For Self-Drive, 1 solid-line crossing across 6 turn trials shows the occupancy-grid-as-constraint approach is close to but not yet at zero-crossing reliability, and intersection-turn times are well within the per-turn budget needed for the Full Course.

## 7 Key Performance Indicators

### 7.1 AutoNav KPIs

The AutoNav KPIs are average speed/variability (target: 1.5 mph ± 0.5 mph) and global GNSS waypoint accuracy (target: ±1 m). Average speed and variability are measured by continuously logging encoder velocity over a full run and computing the mean and standard deviation. The GNSS accuracy measurement method and results are detailed in the next section. These KPI values were selected because they represent the minimum performance needed to complete the AutoNav course within the time budget while still meeting waypoint requirements. The ±1 m global accuracy target is supported on the hardware side by the 3D-printed antenna riser described in the Mechanical Design section, which elevates the GNSS antennas above onboard EMI sources.

#### 7.1.1 GNSS Accuracy Validation

| Trial | Reference Coordinates (lat, lon) | Measured Coordinates (lat, lon) | Error (m) |
|---|---|---|---|
| 1 | 42.294221, -83.709965 | 42.294218, -83.709954 | 0.9643 |
| 2 | 42.294235, -83.709812 | 42.294227, -83.709815 | 0.9231 |
| 3 | 42.294076, -83.710000 | 42.294072, -83.710009 | 0.8636 |
| 4 | 42.294125, -83.710077 | 42.294123, -83.710083 | 0.5413 |
| 5 | 42.294175, -83.709885 | 42.294164, -83.709884 | 1.2260 |

We used the Google Maps satellite view to collect (lat, long) coordinates of physical reference points near our workshop. This reference is not ideal, as it relies on a human precisely selecting a point on Google's software-aligned satellite map, but it was the most feasible method given our available resources. We then moved the robot to these reference points and noted down the (lat, long)

reported by our system. Finally, we calculated the distance between the reference and measured coordinates.

Our GNSS device also outputs position error measurements, which we recorded for 5 minutes. With 20593 samples, our position error was mean=0.7205m and median=0.7221m.

With these two datasets, we determined that we successfully reached our target of 1m of error from our GNSS measurements. We only had one reference location where our GNSS error had a higher value than our target value, but we feel this is a non-issue since it was a single measurement with a non-perfect reference.

### 7.2 Self-Drive KPIs

The Self-Drive KPIs are turn efficiency (target: <20 seconds per turn) and velocity envelope consistency (target: 5 mph hard ceiling, 3–5 mph cruise during open-road segments). Turn efficiency is measured via stopwatch and log analysis at intersections; velocity envelope consistency is measured via encoder logs. The <20 s per turn target gives us a comfortable margin to complete the functions we plan to attempt during a Full Course run within a reasonable total time budget. The 3–5 mph cruise band was selected to match the function-test target speeds specified in Appendix B (FIII–FV) while staying under the hardware 5 mph ceiling.

### 7.3 Requirement Validation

| Requirement | Target | Measured Actual | Status |
|---|---|---|---|
| Average speed & variability (AutoNav) | 1.5 mph ± 0.5 mph | Mean 1.21 mph, std. dev. ∼ 0.3 mph | Success |
| Global GNSS waypoint accuracy (AutoNav) | ±1 m | Mean error 0.90 m (5 trials); sensor-reported median 0.72 m (20,593 samples) | Success |
| Turn efficiency (Self-Drive) | <20 s per turn | 16.14 s mean across 3 intersection tests | Success |
| Velocity envelope consistency (Self-Drive) | Max 5 mph; majority of run in 3–5 mph | Average 3.2 mph throughout tests; zero excursions above 5 mph | Success |

Table 10: KPI validation summary.

**Discussion.** Achieving the target KPIs is a strong indicator that the robot is competitive on both target courses. The GNSS accuracy is well within tolerance for the 2 m waypoint hit radius defined in Section II.2. Average AutoNav speed sits in the middle of the IGVC band, leaving enough margin to complete the course inside the 6-minute window. Self-Drive turn times are well under the per-turn budget needed to chain together the 21 functions of the Full Course, and the velocity envelope holds tight thanks to the firmware cap and the Stanley Controller's commanded velocities.

## 8 Analysis of Complete Vehicle

### 8.1 Lessons Learned During Construction and System Integration

Several lessons learned shaped this year's design. First, we learned to validate key components early. Sensor-level uncertainty compounds through the full stack, so checking depth camera accuracy and IMU bias early helped us avoid mistaking sensor issues for bugs in driving logic.

Second, we learned to integrate on the physical robot as soon as possible. Subsystems that

worked in isolation or simulation often failed on the real vehicle due to timing, coordinate frame, and real sensor behavior that was not captured in simulation. Early end-to-end testing exposed these problems while there was still time to fix them.

Third, we learned to keep cross-team interfaces small. The occupancy grid serves as the main contract between perception and navigation, allowing the CV and Nav teams to iterate independently without breaking each other's systems.

### 8.2 Failed Components and Mitigations

During testing, the bracket supporting our caster wheel failed and fractured underneath the robot. Initially, the bracket was 3D printed because it allowed for rapid prototyping and iteration, and our stress test indicated it would withstand operation in the parking lot environment used for IGVC. However, our test did not account for the vibrations and impacts experienced while transporting the robot over rough terrain to the test site as well as degradation over time. To address this issue, we manufactured the brackets out of aluminum for increased strength. Since implementing these changes, we have not observed any additional damage.

### 8.3 Software Testing, Bug Tracking, and Version Control

All source code lives in a monorepo hosted on GitHub. Development follows a feature branch workflow: every change is authored on a named branch, opened as a pull request against main, and merged only after review. Pull requests link to the GitHub issue they resolve, creating a traceable link between a reported defect or requirement and the commit that addresses it. A GitHub Actions CI pipeline ensures all changes can build, pass the full unit test suite, and pass style and type checks before merging.

We apply three levels of testing before code is deemed ready to merge. Core components like coordinate transform and occupancy grid computation use unit tests to ensure numerical stability for our localization and obstacle detection requirements as well as to prevent regressions. Navigation algorithms must demonstrate stable end to end behavior in simulation. Hardware drivers are required to be tested on the competition laptop and demonstrate stable message publication.

### 8.4 Simulation-Based Testing

Our simulation goal is to test as much of the stack as possible and allow every member to contribute without needing the physical robot. We previously used Gazebo, but found it difficult to create test courses and too computationally demanding for most team members' machines, as well as not being supported for ARM computers, which a significant portion of our members use.

To address this, we built a lightweight custom simulation stack that directly simulates our sensor inputs. A sensor simulator package publishes synthetic encoder velocity, IMU, and GNSS topics with configurable Gaussian noise and Ornstein-Uhlenbeck drift to model slow varying errors like GNSS multipath. An occupancy grid simulator package reads a JSON map of static obstacles and lane lines and publishes them relative to the robot's ground truth position each frame, with ray cast occlusion to simulate obstacles blocking the camera's field of view. This keeps the simulation semantically identical to the real pipeline while being lightweight enough to run in a virtual machine (a common member setup), and allows the full navigation stack to be validated against our waypoint navigation and failure recovery requirements before and concurrently with any physical testing.

### 8.5 Physical Testing and Simulation Comparison

One discrepancy between simulation and physical testing was path tracking. We initially used Pure Pursuit, which performed well in simulation and on previous robots, but caused sustained

oscillations on the physical robot that prevented reliable path following. The root cause was that our simulation assumed ideal differential-drive dynamics, while the inertia of the real drivetrain introduced a delay between the commanded and actual direction changes, which Pure Pursuit could not adequately compensate for. We switched to a Stanley Controller, which corrects both heading and cross-track error simultaneously and is more robust to drivetrain lag. After the change, the robot achieved smooth and stable path following during physical testing.

Another difference we encountered between simulation and physical testing was the generation of the occupancy grid. Initially, the simulation assumed a perfect bird's eye view of obstacles, meaning that the area behind obstacles could be seen. However, the real camera is mounted at or below the height of obstacles, which means that obstacles could occlude the areas behind them and limit visibility. To address this discrepancy, we added occlusion modeling using vectorized NumPy operations to mask regions behind obstacles from the robot's perspective. After implementing this change, the simulated occupancy grid closely matched the real-world occupancy grid upon visual inspection.

# 9 Cyber Security Analysis

Modern robotic and human-driven vehicles are network-connected, contain large amounts of software from multiple vendors, and are an inviting target for cyber-attacks. Below we list three cyber vulnerabilities present in Maverick and discuss what steps should be taken to harden or remove them before series production and wide deployment.

## 9.1 Open Network Ports on University Wi-Fi

By default, all of Maverick's network ports are open to MWireless, the University of Michigan's campus wireless network, which is accessible to any student or faculty. This poses a significant attack surface for an operational vehicle. *Mitigation:* Before series production, the vehicle should run on an isolated network with strict firewall rules. Only authorized devices should be able to communicate with the vehicle, and ROS endpoints should be authenticated and encrypted.

## 9.2 Unencrypted Wireless E-Stop Link

The wireless E-Stop currently uses an unencrypted LoRa link. An attacker within RF range could potentially spoof, replay, or jam the E-Stop signal. *Mitigation:* The radio link should use authenticated encryption with a rolling code or nonce-based scheme to prevent replay attacks, and the receiver should fail safe (stop the vehicle) if the link is lost.

## 9.3 Unprotected Microcontroller Firmware

The Arduino, the ESP32 on the wireless E-Stop, and the STM32 on the Power Distribution PCB expose unauthenticated programming interfaces, so brief physical access could allow malicious firmware to be flashed onto them. *Mitigation:* Microcontrollers should enable readout and flash-write protection, lock debug interfaces, and use signed firmware so the bootloader rejects unauthorized images.