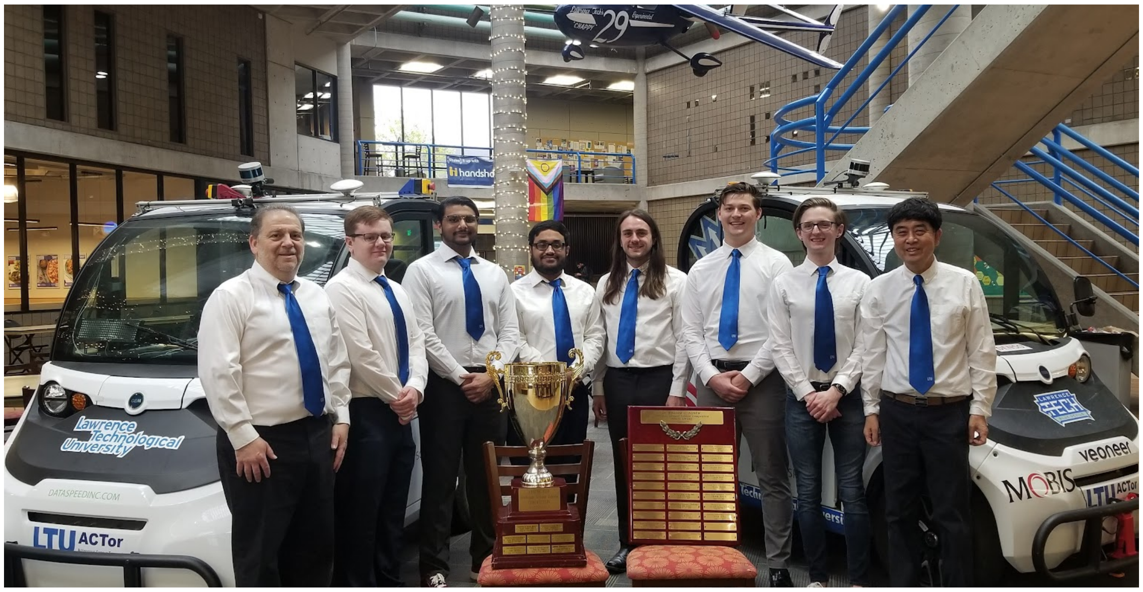
IGVC 2023 Self-Drive Design Report

Lawrence Technological University

Team: ACTor



Date: May 15, 2023

Team Captain	Justin Dombecki	jdombecki@ltu.edu
Members	Devson Butani	dbutani@ltu.edu
	Adilur Choudhury	achoudhur@ltu.edu
	Ryan Kaddis	rkaddis@ltu.edu
	Austin Ramsey	aramsey@ltu.edu
Faculty Advisors	Nicholas Paul	npaul@ltu.edu
	Giuseppe “Joe” DeRose	gderose@ltu.edu
	ChanJin “CJ” Chung	cchung@ltu.edu

Faculty Advisor Statement

I, CJ Chung, Nicholas Paul, and Joe DeRose of the Department of Math and Computer Science at Lawrence Technological University, certify that the design and development on the ACTor research platform by the individuals on the design team is significant and is either for-credit or equivalent to what might be awarded credit in a senior design course.

ChanJin Chung [illegible] [illegible]

1. Conduct of design process, team identification and team organization

Introduction

Our Autonomous Campus Transport (ACTor) vehicle has been a contestant in IGVC, for years. ACTor is an autonomous vehicle research platform built on a Polaris GEM e2 base. The vehicle is equipped with a drive-by-wire system from Dataspeed, which consists of throttle-by-wire, brake-by-wire, steer-by-wire, and shift-by-wire. Computers include a primary laptop computer with GPU and Raspberry PI 3s. Our perception sensors are Velodyne 16-Beam 3D LIDAR, Hokuyo URG 2D LIDAR, Piksi multi real-time kinematics GNSS, and front-facing Mako PoE camera. The vehicle is able perform IGVC self-drive tasks, like lane following, obstacle avoidance, waypoint navigation, and object detection. This report describes the ACTor autonomous system and the methodologies used for the system development and integration.

Major hardware and software changes from 2022 IGVC system [1] include: complete rewiring, new GPU-integrated laptop, updates to Web UI, GPS software upgrade, YOLO-based[10] trained models for IGVC tasks (i.e. stop sign detection, tire detection, pedestrian detection).

Team Organization

The team meets in-person at least once a week on Tuesday to set goals, discuss technologies to develop systems for the vehicle using the incremental agile development concept. These meetings also provide a way to collaborate in creating new ideas and divide up tasks for the upcoming week. Each member was delegated to specific tasks based on prior experiences and interest. The team is composed of computer science students (see Table 1). Each student puts around 5 to 10 hours a week coding, testing, documenting, maintaining, or in meetings to accomplish the tasks.

Name	Degree Program	Primary Responsibilities
Justin Dombecki	M.S. Computer Science	Lua engine (Core), Web interface, GPS & Waypoint control, Camera sensing and image republishing, Lane detection, lane centering & keeping, Drive By Wire System
Devson Butani	M. S. Computer Science	Mechanical & electrical systems, Re-wiring, eStop, Yolo-based stop sign & pedestrian detection
Adilur Choudhury	B.S. Computer Science	Yolo-based tire detection; Pedestrian detection, Pothole detection & avoidance; LED display system
Ryan Kaddis	B.S. Computer Science	Turns, Tire & pothole detection and avoidance
Austin Ramsey	M. S. Computer Science	LiDAR based obstacle detection and avoidance; Parking

Table 1: Team members and roles

Design Assumptions and design process

(1) Highest priority is to meet user requirements through early, continuous and incremental delivery of working software. (2) We are able to adapt to changes in requirements. (3) Deliver working software frequently. (4) In-person interaction is an efficient and effective method for sharing information within a team. (5) Progress is measured by working software. (6) Simplicity, the art of maximizing the amount of work not done, is essential.

2. EFFECTIVE INNOVATIONS IN ACTOR DESIGN

Innovative technology applied to ACTor

The ACTor project is a research environment for students interested in autonomous software development. The project was designed to be modular, incremental, and dynamic meaning that software modules are easy to switch out, add, or remove allowing rapid prototyping and development. Testing results show that the ACTor vehicle is capable of performing IGVC Self-Drive challenges. The design spawned several research projects involving software engineering, machine vision, and deep learning. Innovations we achieved include:

- Lua script language to specify vehicle behaviors in high level
- Human and object detection using Yolo v8
- Research into lane following alternative using deep learning

- Extension of Gazelle Sim, a lightweight 2D simulator

This year, we consolidated all our individual components that were previously distributed in different electrical enclosures around the vehicle into a single custom dashboard, see Figure 1. These components have shown their reliability and durability through several iterations of development, and are now integrated into the vehicle as permanent parts rather than temporary test units.

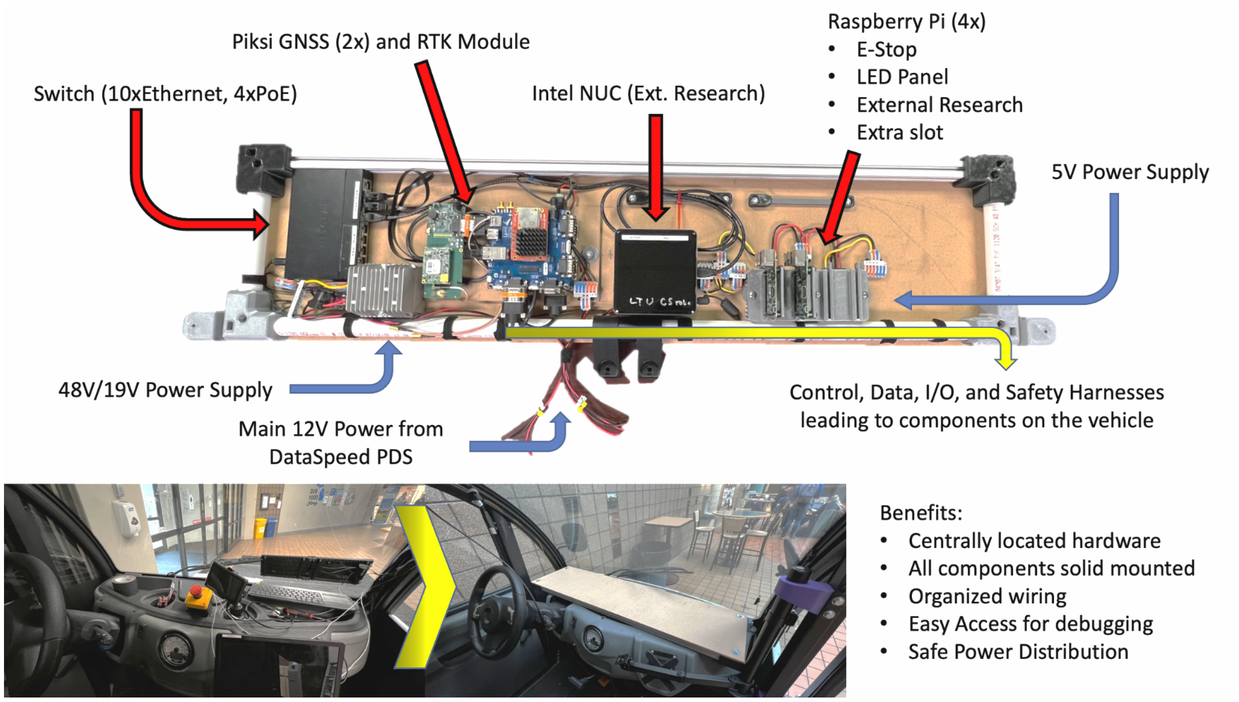

Piksi GNSS (2x) and RTK Module
Switch (10xEthernet, 4xPoE)
Intel NUC (Ext. Research)
Raspberry Pi (4x)
• E-Stop
• LED Panel
• External Research
• Extra slot
5V Power Supply
48V/19V Power Supply
Main 12V Power from DataSpeed PDS
Control, Data, I/O, and Safety Harnesses leading to components on the vehicle
Benefits:
• Centrally located hardware
• All components solid mounted
• Organized wiring
• Easy Access for debugging
• Safe Power Distribution

Figure 1: Major electronics moved under a custom Dashboard

For ease of use, the main laptop (detachable) is on a swivel arm attached to the frame with 3D printed mounts, see Figure 2. It can be operated by the passenger or the driver safely and comfortably. Since the vehicle has no air conditioning, the arm also allows it to swing outside for debugging or demonstration.

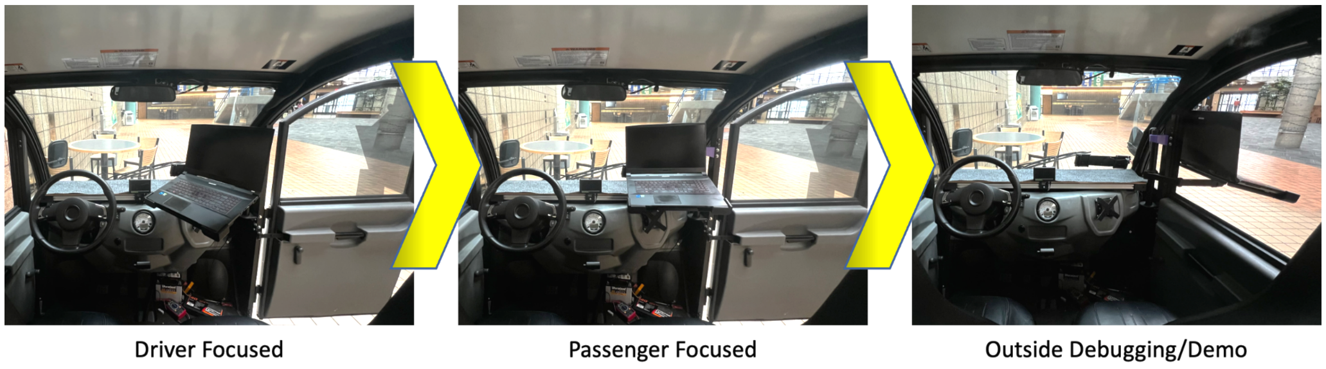


Figure 2: Laptop swivel arm

3. Description of Mechanical Design

Overview

ACTor is a modified Polaris Gem e2 jointly sponsored by MOBIS, DENSO, Veoneer, Realtime Technologies, Dataspeed, and

SoarTech. It has a drive by wire system, developed and installed by Dataspeed Inc., for autonomous driving. The vehicle has outdoor-rated sensors. Other electrical components are inside the vehicle and mounted securely. Since the vehicle can handle common paved surfaces and up to 30% grade, the suspension is kept stock to the factory design, using MacPherson struts on the front and independent trailing arms for the rear Figure 3 shows the ACTor 1 vehicle with its hardware components.

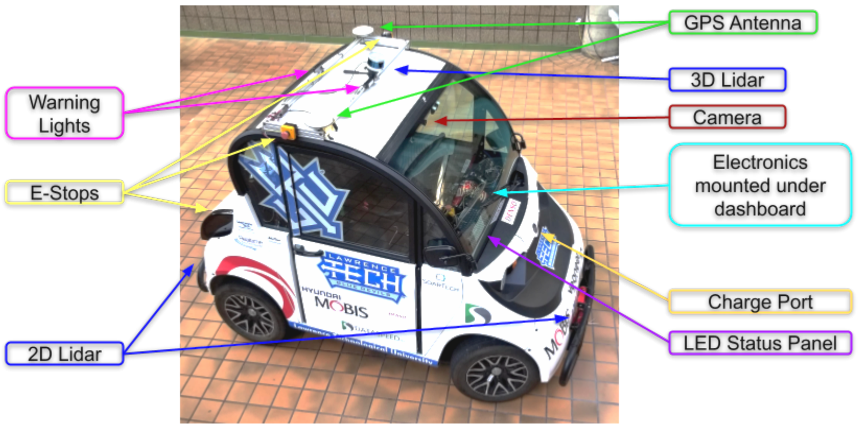


Figure 3: ACTor components on the outside

Item	**Price**
Polaris GEM e2 vehicle with various options such as doors and trunk	$15,000.00
New Polaris GEM ADAS Systems (Drive-By-Wire systems by Dataspeed) including installation fee	$35,000.00
Intel NUC Mini PC kit NUC7i5BNH Core i5	$543.00
Velodyne VLP-16 “PUCK” 3D LiDAR, 16 beams	$7,999.00
Hokuyo UTM-30LX 2D LiDAR	$6,500
Hokuyo URG-04LX-UG01 LiDAR	$975
Swift GPS, Piksi Multi GNSS	$1,644.56
Additional rover module and antenna to get GPS heading info	$896.00
Mako PoE Camera	$1,031.00
MSI Gaming Laptop, Intel 8-Core i7-11800H, 16GB RAM, 512GB SSD, GeForce RTX 3050 Ti 4GB	$1,258.99
Miscellaneous items including lenses & filters, e-stop switches, wireless e-stop, LED strobe lights, cabin camera, RPIs, inverters, switches, router, mounting rack, and LED panel, etc.	$3,000.00
Total	$73,847.55

Table 2: Estimated cost of ACTor vehicle

Description of drive-by-wire kit

The Polaris Gem e2 has a top speed of 20 mph and a range of 20 miles. However, while under autonomous mode we enforce lower speed limits using the DatasSpeed ADAS Development Vehicle Kit via ROS. This drive-by-wire (DBW) kit allows the vehicle to be driven using native or electrically controlled interfaces with ROS. The Universal Lateral/Longitudinal Controller (ULC) modulates the native accelerator/brake pedals, steering wheel and gear selection systems to achieve the desired linear and angular velocity targets for the vehicle. We use the default parameters of the ULC, which have a velocity dependent linear acceleration limit (0.9 - 1.2 m/s2) and a constant deceleration target of 1.5 m/s2. Figure 4 shows the linear velocity response with the default acceleration limits for the 5.2 mi/hr speed limit test. The Dataspeed ULC uses a geometery_msgs/twist ROS message with extra parameters to adjust the vehicle behavior, such as acceleration limits. This interface made it easy to connect the DBW system to the vehicle’s ROS environment.

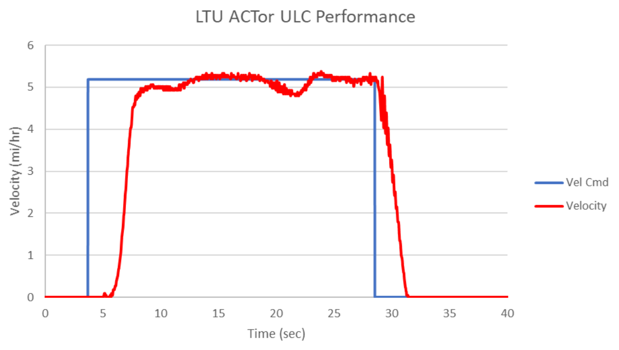


Figure 4: Dataspeed ULC linear velocity control example

Weather proofing

The stock Polaris Gem e2 has weather-sealed doors and body. All of the external mounted sensors (roof, front/rear bumper, etc.) are outdoor-rated. These sensors are connected to the in-vehicle components through a common weather-sealed conduit between the window and body, which is behind the passenger seat.

4. DESCRIPTION OF ELECTRICAL AND POWER DESIGN

Overview

The ACTor vehicle is powered by the stock Polaris Gem e2 batteries. They power three systems: native vehicle circuits, a 1kW DC-AC inverter and the DataSpeed Power Distribution System (PDS). Figure 5 shows how the computers and sensors are powered from these three systems.

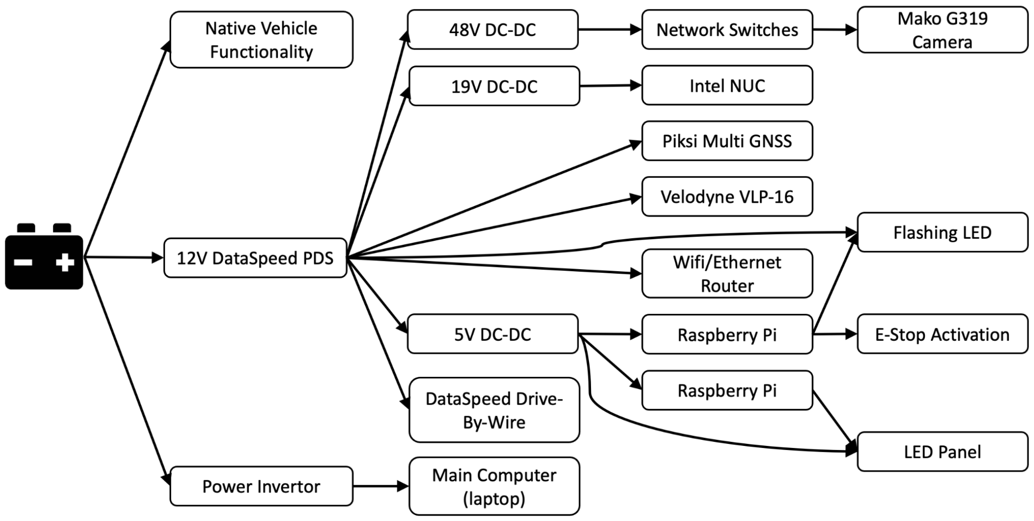


Figure 5: Vehicle, sensor, and components power distribution

Power distribution system (cacacity, max. run time, recharge rate, additional innovative concepts)

The factory-installed batteries take 6-8 hours to charge fully and provide 20 miles of range (while using all autonomous electronics). The DataSpeed PDS protects components from overload and offers control of all the circuits via a touchscreen interface or the CAN bus. Some of the power efficiency losses come from the 1kW inverter (92% eff.) and DC to DC converters (92-96% eff.) adding upto 800W that supply a range of voltages to individual components.

Electronic suite description including CPU and sensors system integration/feedback concepts

The vehicle's components are controlled by a laptop (Intel 8-Core i7-11800H, 16GB RAM, 512GB SSD, GeForce RTX 3050 Ti 4GB VRAM) that runs Ubuntu. This laptop also runs our route scripting tools and performs object detection and sensor fusion using the camera and LiDAR. These sensors enable high-accuracy real-time detection and 3D positioning of pedestrians and obstacles. The laptop's discrete GPU allows us to use deep-learning models such as our alternative lane-following system and Yolo v8 based detection algorithms. We also use Raspberry Pi 3s for hardware e-stop, remote e-stop, safety and status lights, and an LED panel to display status information outside the vehicle.

Our vision system consists of an Allied Vision Mako G-319C [2] PoE camera mounted beside the rear-view mirror. It captures 2064 × 1544 images using a 6mm 1stVision LE-MV3-0618-1 lens [3] at 1.8 full stops and a 50 degrees field of view. This allows us to capture the lanes and detect road signs up to 30 feet away. We also use a circular polarizing filter to reduce reflections from the road, other cars and inside our car. A Velodyne VLP-16 "Puck" LIDAR [4] donated by Veoneer provides 360 x 15 degrees of field of view with a radius of 100 meters. It outputs 300,000 points per second across its 16 channels. We also use Hokuyo URG-04LX-UG01 [5], a small two-dimensional lidar with 4 meters of range, mounted on the front and the back. It helps us detect immediate obstacles of significant size like tires, cones and curbs. These 3 lidars also assist us during parking operations.

To navigate, the vehicle uses two Piksi Multi Modules [6], which are RTK GNSS receivers that can access multiple bands and constellations. These modules provide position and heading data with centimeter-level accuracy and high update rates. This is ideal for a moving vehicle that needs to cover large distances in a short time.

We use Ethernet connections for most of the components to ensure reliability and ease of debugging in a noisy electrical environment.

This networked architecture also allows remote access for testing and monitoring purposes. The only components that do not use Ethernet are the 2D lidars, the cabin webcam, and the USB CAN interface to the DBW system. The vehicle will automatically activate an emergency stop if it detects any failure in the DBW connection. Figure 6 shows the wiring schema of the ACTor vehicle.

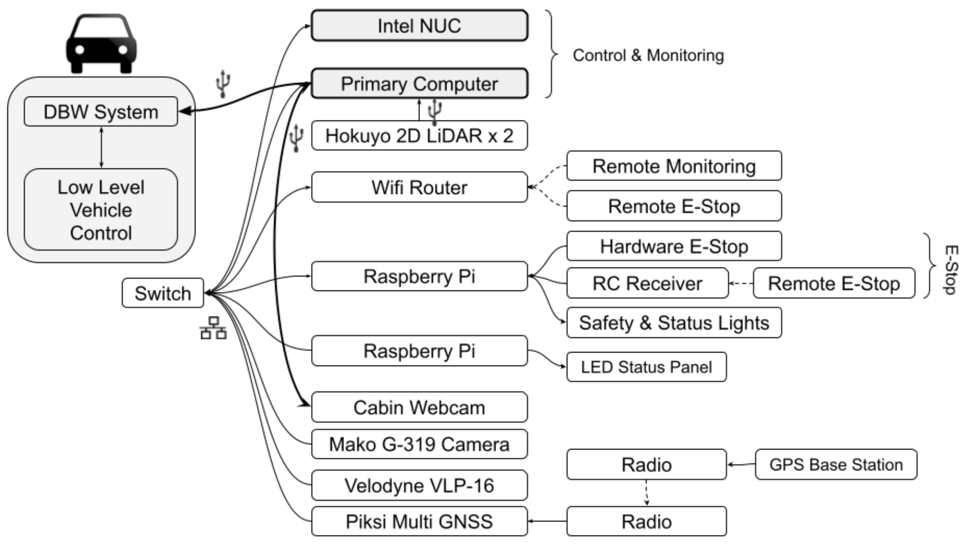


Figure 6: Data connections (Solid lines - physical cable connections, Dashed lines - wireless communication)

Safety devices and their integration into your system

The emergency stop system consists of two parts. The first part is a loop circuit that connects all the E-stop buttons to a Raspberry Pi. If any button is pressed, the circuit will break, and the Pi will send an E-stop message to the main computer via Ethernet. The main computer will then safely halt the vehicle. The second part is a heartbeat mechanism that requires constant communication between the main computer and the Pi to enable any interaction with the DBW. Additionally, the Pi controls warning lights on top of the vehicle that flash when the vehicle is in autonomous mode. This dual safety system ensures that all critical components are functioning properly before allowing autonomous driving.

5. Description of software strategy and mapping techniques

Overview

As the primary function of the ACTor vehicle is for enabling research, the ACTor software architecture requires the ability to onboarding new students quickly and allow them to integrate their ideas simply. Using the distributed and modular software design principles of the Robot Operating System (ROS) [7], our software design should enable quick development cycles by allowing its inputs and outputs to be interchangeable. With these requirements in mind, our software can be quickly tested and allows for smooth implementation of new hardware and software.

The system architecture is distributed into packages: sensors, input processing, route input, route system and web API. Figure 7 shows a high level overview of the contents of these packages.

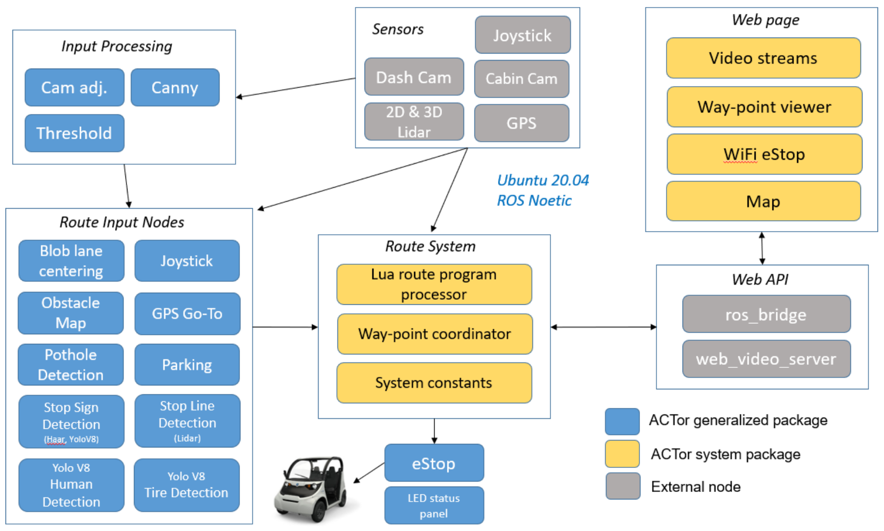


Figure 7: Basic data flow through core packages

Obstacle detection and avoidance

Obstacle Detection and Resolution

The obstacle avoidance package currently uses input from the VLP-16. Using built in functionality, ground and obstacle PointClouds are generated from the VLP-16's input.
The current implementation of the obstacle avoidance algorithm checks regions defined by parameters as shown in Figure 8. These regions are published to the route system, which determines the action to be taken. If an obstacle is within an emergency region, the vehicle will halt. If it is far ahead in the road, it may execute an avoidance maneuver or halt depending on the scenario. The obstacle detection node allows for detection of objects in arbitrarily defined spaces. This allows different events to occur based on which detection region the object is in.

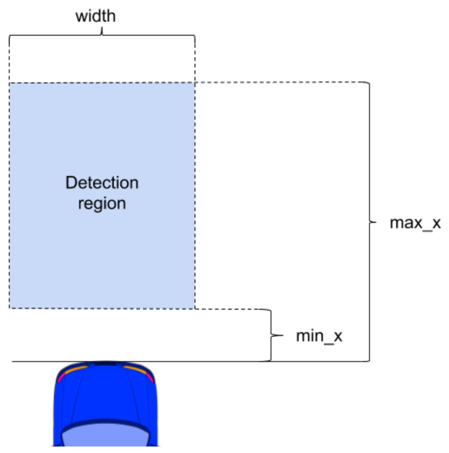


Figure 8: How to define detection region

Pedestrian Detection

To locate humans, we employ a very effective deep neural network model Yolo v8 that has been pre-trained on the large COCO dataset and can recognize a "person". With the help of the discrete GPU on the main computer, it can swiftly and precisely detect any person in the image, and we combine this information with our LiDAR data to obtain a labeled and highly accurate 3D positioning of pedestrians. This allows us to determine their location and enable basic interactions, such as stopping, avoidance or lane changing. Figure 9 shows an example of the Yolo v8 person detection on a rainy day.

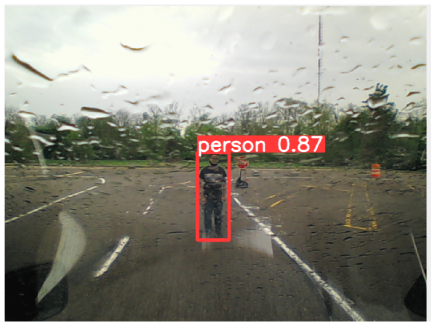


Figure 9: An example of detecting people

Sign Detection

Our previous algorithm used color filtering and HAAR classifiers to detect signs based on shape and color. We trained the HAAR classifiers with thousands of images of stop signs. The algorithm was fast and worked in different environments. However, it also detected anything else that had a similar shape and color to a stop sign (eg: red car at a distance). There were many false positives and false negatives including fake stop signs that did not say "stop" explicitly.

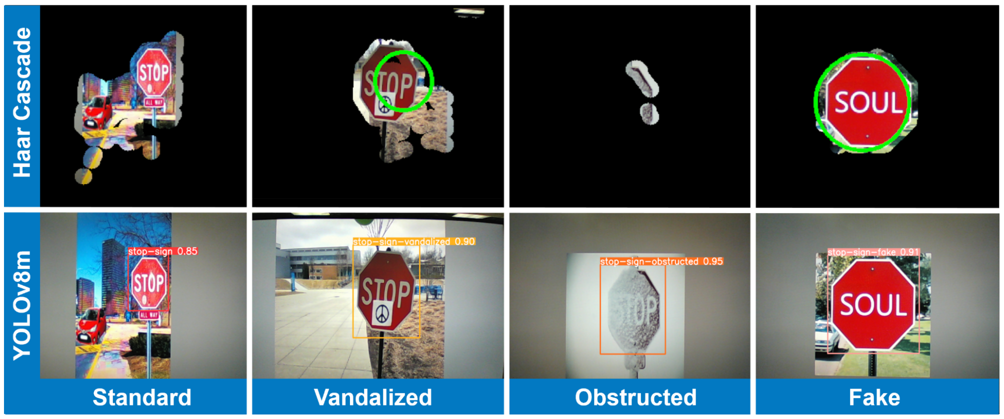


Figure 10: HAAR Cascade v/s Yolo v8 Detecting stop signs in various conditions

We developed a new deep neural network to detect stop signs using Yolo v8 and a custom dataset with more than 3600 images. Our new model can detect stop signs in various conditions such as clear, obstructed, vandalized or fake. It can also distinguish fake stop signs from real ones that are in poor conditions as shown in the figure above. Using our laptop GPU and clear images from the Mako camera, we can achieve fast detection speeds (120ms interval, roughly 1 feet of travel at 5mph) while reducing false positives and identifying fake signs. This new Yolo v8 architecture also allows us to add more road signs if necessary in the future. Figure 10 shows a comparison between our previous algorithm and the new Yolo v8 solution given certain use cases.

Pothole and Tire Detection & Lane Change

Pothole detection is accomplished through color filtering to isolate the pothole in the frame. The pothole – and everything else within the color range – is depicted as a white mask, and the rest of the image is shown as a black background. If the mask covers a sufficient portion of the frame region, then the vehicle will execute the lane following algorithm.

Tire detection is accomplished using YOLO v8 object detection. The YOLO v8 tire model is responsible for finding tires using the AVT Mako camera. When the tire is detected by the camera with a high confidence level (see Figure 11) and is within a close range from the vehicle, the lane changing program will be triggered. Lane changing uses dead reckoning to turn for a certain amount of time and reactivates the lane following algorithm afterwards.

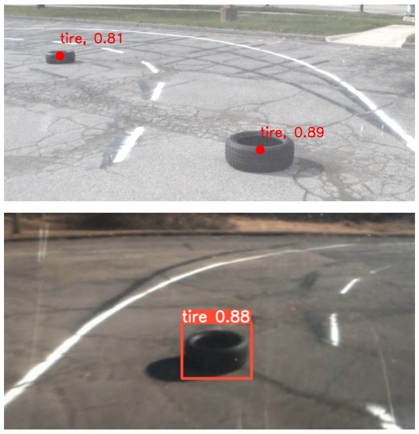


Figure 11. Tire Detection Examples

Lane Following

Lane following consists of two functions: lane detection and centering within a lane. Both of these are handled by a single node called 'blob'. Figure 14 shows the steps of the lane detection algorithm [8].

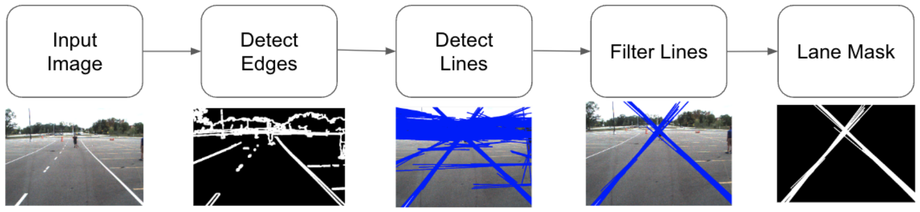


Figure 14: Lane detection filtering process

Using OpenCV functions (i.e. Canny edge detection, blurring. dilation), the second image is generated, which highlights the edges of the initial input image. Next, applying a Hough transform determines lines from the edges.

Those lines are filtered by their angle, only 45 degrees of vertical are accepted. The remaining lines are extended in length to produce the final 'blob' lane mask. Lane centering uses the 'blob' lane mask image to generate springs that will push and pull the vehicle to the center of the lane. In Figure 15, a fan of probes is sent from the front center of the vehicle (*i.e.* red dot) to find the Hough lines, creating springs at the intersections. The length of each spring determines its force and impact on centering the vehicle in the lane.

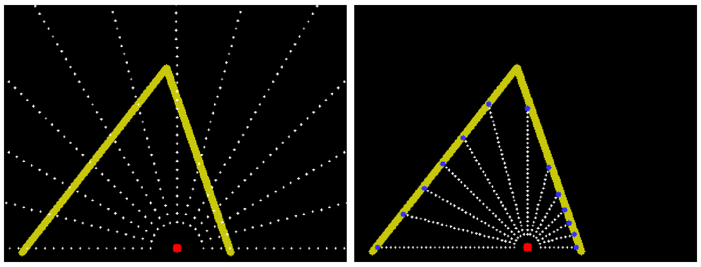


Figure 15: Rays are generated from the vehicle. Springs are created, where the ray intersects a Hough line.

Denoting the front center of the vehicle as C_V and the center of the lane as C_L, the goal of the lane centering algorithm is to minimize the distance between C_V and C_L. The generated springs together attempt to achieve an equilibrium by pushing C_V toward C_L at shorter springs lengths, and pulling C_V toward C_L at longer lengths. The horizontal component of these combined forces is the steering input to the vehicle. Figure 16 shows these terms in context to the vehicle within a lane, while Figure 17 shows the representation within the 'blob' node.

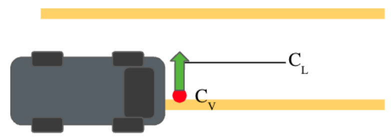


Figure 16: Lane centering reduces the distance between the front center of the vehicle C_V and the center of the lane C_L.

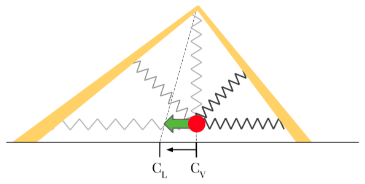


Figure 17: Springs with the shorter lengths push C_V toward C_L, while springs with longer lengths pull.

Software strategy and path planning

Sensors

The sensors package contains publisher nodes and configuration files for each sensor (e.g. GPS, LiDAR, camera). Each sensor node translates raw data into ROS messages and publishes them on ROS topics, which creates a higher level abstraction for the vehicle routing system. Integrating new sensor hardware or software is simple, given that the new nodes publish to the existing ROS topics used by the router. Specific to the LiDAR, there are actually two nodes that manage this sensor type, as a pair of front and rear Hokuyo URG two-dimensional LiDARs are used for tasks, where the Velodyne unit (mounted on the roof) is not able to make the necessary detections.

GPS

Using a two antenna Piksi Multi GNSS configuration, ACTor is capable of high precision position and heading accuracy using SBAS. With an optional base station, a RTK fix can be achieved for even better accuracy. Our ROS nodes use positional coordinates (latitude & longitude in the ECEF format), inertial measurement units (IMU) and heading info (NED - north east down). This GPS data is used to calculate the distance from a static waypoint. For example, in the merging task, a waypoint is set at the point at which the lanes diverge, indicating to ACTor when to begin turning into the correct lane. This year required the upgrade of this Piksi software to the latest supporting ROS version 1, which moved from a Python to a C++ node. As such, many of the topics and message types required reimplementation. For additional debugging, the positional and heading information has been added to the Web UI.

Input Processing

The input processing package performs data transforms on specific topics before forwarding to the route input package. Used specifically for the camera, the package applies algorithms (filters) to the image data to control white balance, gaussian blur, and other options from OpenCV, which provides more normalized inputs into the routing package.

Route Input

The route input package acts like a switchboard operator for the vehicle route system. Used mainly for image processing, this middleware between sensors and routing may be extended to process a sensor fusion for any specific task. The routing package is able to subscribe and unsubscribe from each task-specific node, thereby only running those nodes required for a task. As many nodes are resource intensive, this pattern reduces computational overhead on the system. This route input package is applied in the following use cases: lane centering ("blob"), obstacle maps (avoidance & emergency monitoring), objection detection (stop sign, one way sign, stop line, pothole), waypoint follow and parking maneuvers. These nodes can even be combined together to accomplish the necessary tasks.

Route System & Lua Scripting

ACTor is a frequent competitor at IGVC Spec2/Self-Drive. For many years, our team has used the Lua scripting language to create a routing abstraction, which has negated much of our code compilation overhead [9]. During competition, this allows the team to focus on detection and routing tasks, and not waiting on the code to compile. The Lua scripting abstraction is within the 'Router' node, shown in Figure 12, which outputs a Twist message to the ACTor Drive-by-Wire system within ROS. To determine the correct Twist message, the router will receive the Sensor data streams and -- either use Lua scripting override instructions -- or forward instructions provided by our navigation nodes. As our Lua functions are capable of reading/publishing ROS std_msgs, we are able to build up methods of abstraction, send ROS topics directly, or set preconditions for other instructions to execute. Our system is centered around the router, its primary goal is to feed the route script as much data as possible.

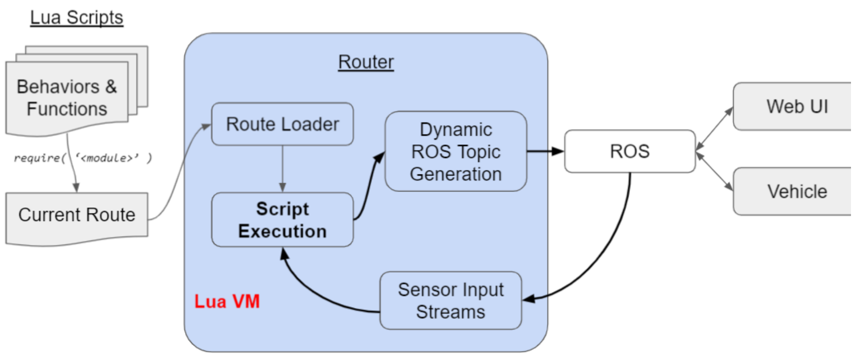
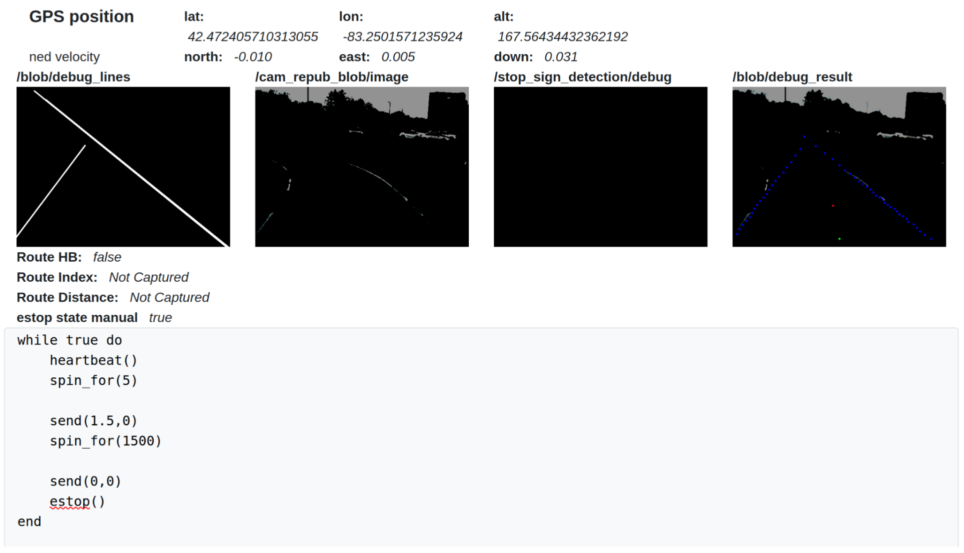


Figure 12: Implementation of Router Server

Web API

Lua scripting can be developed within a Web Application, which is hosted on the main computer. This Web UI is configurable using React and Node JavaScript frameworks along with Bootstrap CSS. These frameworks have allowed the team to customize the page, at-will, which provides debugging insights -- such as a live feed of the lane centering 'blob' node and GPS coordinates. The ACTor vehicle receives Lua route instructions from an editable text field on the Web App which uses predefined functions in the 'Router' node. Figure 13 shows an example of the web interface.



GPS position lat: 42.472405710313055 lon: -83.2501571235924 alt: 167.56434432362192

ned velocity north: -0.010 east: 0.005 down: 0.031

/blob/debug_lines /cam_repub_blob/image /stop_sign_detection/debug /blob/debug_result

Route HB: *false*
Route Index: *Not Captured*
Route Distance: *Not Captured*
estop state manual *true*

```
while true do
    heartbeat()
    spin_for(5)

    send(1.5,0)
    spin_for(1500)

    send(0,0)
    estop()
end
```

Figure 13: Web interface example

Map generation

A map is generated when a particular function requires the use of a GPS waypoint, allowing the user to see the current vehicle location relative to waypoints. However, most of the path planning is done within the Web UI particular to the function at task.

Goal selection and path generation

The Web UI is used to create a path plan given the required task. The follow_lanes() function allows the team to use the 'blob' node, ensuring the vehicle stays on course. The dist_from_waypoint() function provides understanding about the vehicle's position on the course. Functions like look_for_stop_sign() and distance_from_object() allow the team to set up real-time control flow, given the current driving conditions. Combining these functions, and writing others during the competition, allow the team to maintain flexibility around course conditions and the particularities of each task.

Additional creative concepts

Parking

IGVC requires three parking tasks: Pull In, Pull Out, and Parallel Park. The Pull In task demands that the ACTor drive straight down a lane and then efficiently turn into a parking spot from the furthest lane. This is accomplished using distance commands based on GPS heading info. The Pull Out task demands that the ACTor pull out of the aforementioned parking spot, turn onto the given outside avenue, and continue until close to a barrel. The movement is achieved using distance commands, but barrel detection is achieved using a 2D LIDAR. The Parallel Parking task requires that the ACTor successfully parallel park without crossing certain boundaries, which represent real-life barriers and objects. This is accomplished using distance commands as well.

LED Panel System

ROS River is a LED Panel System that shows the user information about the ROS system on a LED display (See Figure 18). It runs on a Raspberry PI 3B that is connected to the main computer's ROS stack via ethernet. The display can be scaled to any size using the WS2812B LEDs. ROS River subscribes to some ROS topics and displays the data accordingly. For example, it can display the data from the topic "display/text" directly or it can also run in "auto" mode that displays information from the topic "rosout" based on the ROS core's status. The ROS core is responsible for communicating the vehicle's state and objective to people outside the vehicle. The LED system helps people to understand the vehicle's current situation & potential issues. Without the display, only the people inside the vehicle would have this info.

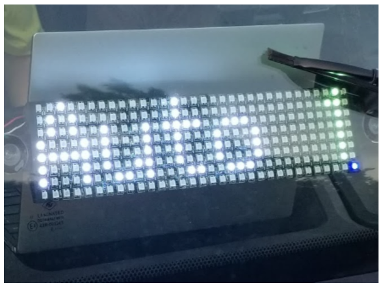


Figure 18: LED Panel system "River"

6. Description of failure modes, failure points and resolutions

Vehicle failure modes (software, mapping, etc) and resolutions

Table 3 summarizes each failure mode with risk level and description how to resolve.

Failure Mode	**Type**	**Risk**	**Resolution**
Camera is unable to determine lane lines	Software	Medium	Verify camera calibration before runs
A ROS node crashes	Software	Medium	Depending on which node crashes one of two things will happen: 1. Non-critical: The node is automatically relaunched by the system 2. Critical: The primary computer issues a stop command within 200ms
Invalid actions or routes are received.	Software	Low	Navigation immediately enters a paused state and halts route execution.
Estopped	Software	Low	Enable DBW system

Table 3: Vehicle failure modes

Vehicle failure points (electronic, electrical, mechanical, structural, etc) and resolutions

Table 4 summarizes each failure point with risk level and description how to resolve.

Failure Point	**Type**	**Risk**	**Resolution**
Loss of Power	Hardware	Very Low	Autonomous mode is automatically deactivated and the safety driver takes control. Primary computer (powered by battery) reports error.
Switch or Network Malfunction	Hardware	Low	Primary computer is unable to contact the external safety monitor and issues an E-Stop command within 200ms.
Inverter Malfunction	Hardware	Low	Primary computer is unable to contact the external safety monitor due to loss of power and issues an E-Stop command within 200ms.
Raspberry Pi (Safety Monitor) malfunction	Hardware	Low	Primary computer detects irregularity in external safety monitor and issues E-Stop command within 200ms.
E-Stop malfunction	Hardware	Very Low	E-stop requires an active signal, if interrupted, the vehicle executes stop command within 200ms.
Camera Malfunction	Hardware	Low	Computer displays disconnection error. If needed, the driver may E-Stop the vehicle.
Lidar Malfunction	Hardware	Low	Computer displays disconnection error. If needed, the driver may E-Stop the vehicle.
GPS Malfunction	Hardware	Low	Computer displays disconnection error. If needed, the driver may E-Stop the vehicle.

Table 4: Vehicle failure points

All failure prevention strategy

A Raspberry Pi acts as an external "emergency monitor" (EM) that oversees the safety measures of ACTor's hardware and software. The EM is connected to the ROS network and acts as a mediator between the ACTor software and the vehicle. The ACTor software cannot directly control the vehicle, but has to send a motion command to the EM, which checks if the command is within the limits of max speed, max turn radius, etc. and then passes it to the vehicle through the "host" node (see Figure 19). The EM also monitors the lidar node and the E-Stop subsystem for any emergency signals and stops the vehicle immediately if any are detected. To prevent failures of the EM and the E-Stop, additional precautions were taken. The host node will issue a blocking stop command within 200ms if the EM loses power, disconnects from the network, or sends invalid data. The E-Stop subsystem requires an active external hardware signal, so any malfunction will also trigger a stop command within 200ms. A safety light is attached to the vehicle and flashes when the vehicle is in autonomous mode.

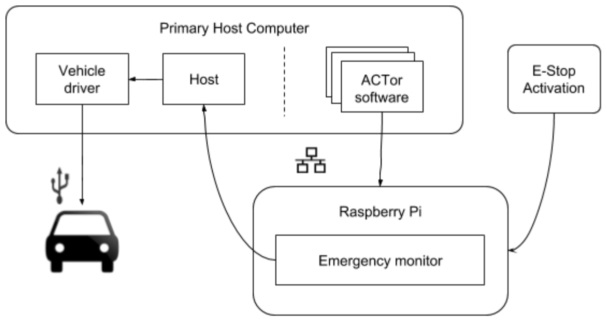


Figure 19: Emergency Monitor for Safety

With minimal latency, the vehicle stops after an eStop event is triggered. Figure 20 shows how the vehicle speed changes during an eStop event. In this example, the target deceleration was set to 1.5 m/s2 and the vehicle was moving at 5 mi/hr before the event. However, because of some delays in the feedback system, the actual deceleration was around 0.75 m/s2 on average.

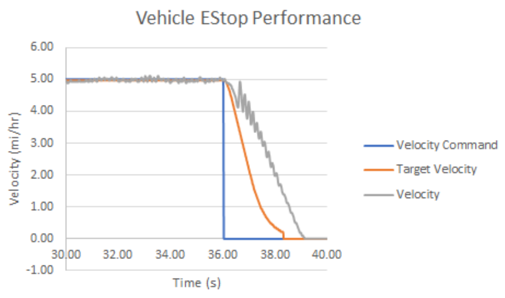


Figure 20 - An eStop event triggered at 36 seconds, complete stop at 39 seconds

Testing (mechanical, electrical, simulations, in lab, real world, etc.)

As the EM is an integral part of the ACTor system, this mechanism must be in place for the system to function. As such, the emergency monitoring system is tested in the real world each time the team tests a function or integrates a new node.

Vehicle safety design concepts

The EM is the main safety feature for software and hardware failure scenarios. In addition, 2D LiDARs have been mounted on the front and rear bumper. Implemented for use in parallel parking, these can be used for further vehicle safety, by invoking the estop if a minimum distance threshold is exceeded in the front or rear of the vehicle.

7. SIMULATIONS EMPLOYED

Simulations in virtual environment

The team completed multiple simulation investigations using GazelleSim, a LTU developed ROS simulation package that supports kinematic models of akermann steer or differential steer robots, a pinhole camera model, an ideal lidar model and GPS tracking model. The ground plane is supplied to the simulation environment as an image and circular and rectangular obstructions may be added to simulate approximated lidar directions. Obstructions were added to the simulation environment to represent barrels, pedestrians and stop signs. Multiple software algorithms were tested by simulating the vehicle behavior on the IGVC and LTU campus course models. This was done to minimize the physical testing required to validate the system performance.

8. PERFORMANCE TESTING TO DATE

Component testing, system and subsystem testing, etc.

The software architecture is modular (see Software Systems section), which allows testing each function (node) separately or in combination with others. The nodes undergo rigorous integration testing during development and on the field. The vehicle is tested on specific situations for integration tests. At the time of publication, most of the nodes have been tested thoroughly and several integration tests have been completed. The tests of most IGVC functions were performed on a test course created at LTU campus shown in Figure 21. The vehicle's mechanical, electrical, and physical hardware have no major performance issues so far.

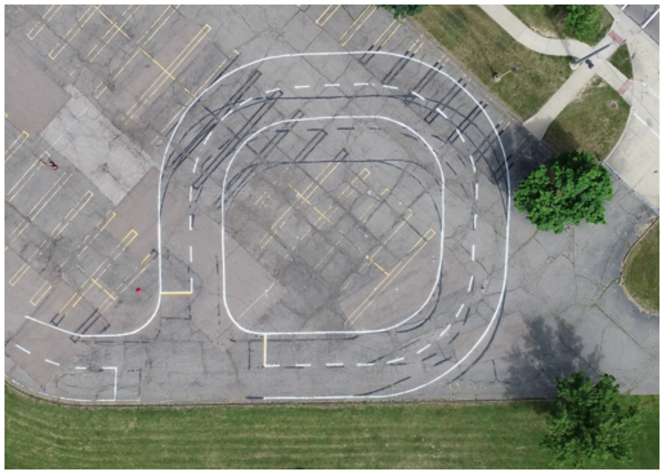


Figure 21: Test course at LTU

9. INITIAL PERFORMANCE ASSESSMENTS

How is ACTor performing to date?

The team is able to test many of the IGVC functions on our test course at LTU. Table 5 shows the status of each qualification, machine vision, traffic sign, intersection, parking, VRU, curved road and other tests.

Qualification Tests	*Parking Tests*
Complete - Q.1: E-Stop Manual	**Complete** - FIV.1: Parking. Pull Out
Complete - Q.2: E-Stop Wireless	**Complete** - FIV.2: Parking. Pull In
Complete - Q.3: Lane Keeping (Go Straight)	**Complete** - FIV.3: Parking. Parallel
Complete - Q.4: Left Turn	
Complete - Q.2: Q.5: Right Turn	*VRU Tests*
	Testing - FV.1: Unobstructed STATIC pedestrian detection
Machine Vision Tests	**Testing** - FV.2: Obstructed DYNAMIC pedestrian detection
Complete - FI.1: White Line Detection	**Testing** - FV.3: STATIC pedestrian detection. Lane changing
Complete - FI.2: Static Pedestrian Detection (Vision)	**Testing** - FV.4: Obstacle detection. Lane change
Complete - FI.3: Tire Detection	
	Curve Road Tests
Traffic Sign Tests	**Complete** - FVI.1: Lane Keeping
Testing - FII.1: Stop Sign Detection	**Testing** - FVI.2: Lane Changing
	Other Tests
Intersection Tests	**Testing** - FVII.1: Pothole Detection
Complete - FIII.1: Lane Keeping	**Complete** - FVII.2: Merging
Complete - FIII.2: Left Turn	
Complete - FIII.3: Right Turn	*Simple Main*
	Planning - Simple Main

Table 5: Status of each IGVC function on design report submission date. (Planning, Developing, Testing, Complete)

10. UNIT TESTING RESULTS

The mandatory unit tests have been tested at the LTU test course. The following details the results of those tests.

Unit Test 1: Emergency Stop

Referring to Figure 22 -- around 69.32 seconds, the E-Stop is triggered above the driver door of ACTor. At 71.50 seconds, the vehicle came to a complete stop. That is 2.18 seconds from E-Stop initialization to a complete stop.

Unit Test 2: Emergency Stop remote

Referring to Figure 23 -- around 13.00 seconds, the remote E-Stop is triggered. At 15.76 seconds, the vehicle came to a complete stop. That is 2.76 seconds from remote E-Stop initialization to a complete stop

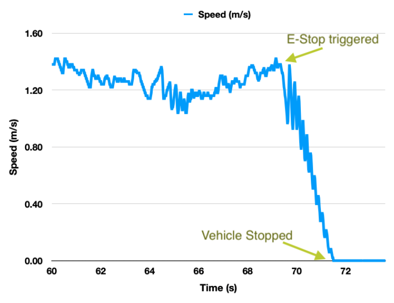


Figure 22: Emergency Stop unit test

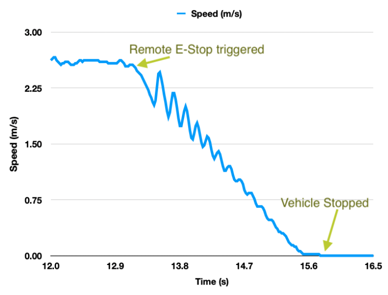


Figure 23: Emergency Stop remote unit test

Unit Test 3: Speed limit test

3 mph (1.34 mps): Referring to Figure 24 -- setting the speed limit to 3 mph (1.34 mps), the vehicle starts accelerating at 18.54 seconds and reaches the target speed at 22.42 seconds. That is 3.88 seconds to accelerate from 0 to 1.34 mps.

5.2 mph (2.28 mps): Referring to Figure 25 -- setting the speed limit to 5.2 mph (2.28 mps), the vehicle starts accelerating at 24.27 seconds and reaches the target speed at 32.84 seconds. That is 8.57 seconds to accelerate from 0 to 2.28 mps.

6.0 mph (2.68 mps): Referring to Figure 26 -- setting the speed limit to 5.0 mph (2.68 mps), the vehicle starts accelerating at 12.80 seconds and reaches the target speed at 17.28 seconds. That is 4.48 seconds to accelerate from 0 to 2.68 mps.

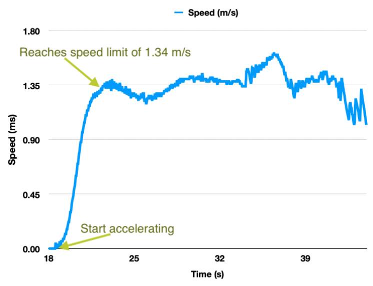


Figure 24: 3 mph speed limit unit test

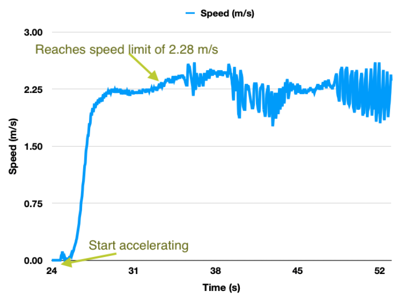


Figure 25: 5.2 mph speed limit unit test

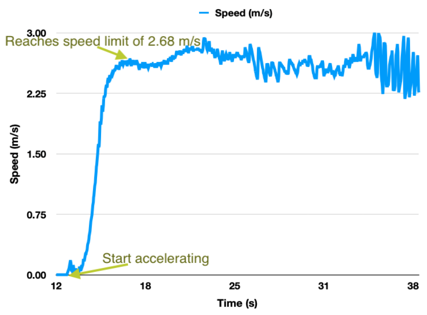


Figure 26: 6 mph speed limit unit test

Unit Test 4: Right Lane boundary is crossed

Referring to Figure 27 & 28 -- around 61.78 seconds, the remote E-Stop is triggered. At 64.02 seconds, the vehicle came to a complete stop. That is 2.24 seconds from remote E-Stop initialization to a complete stop during a right turn.

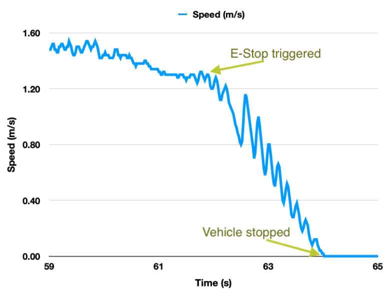


Figure 27: right boundary unit test (speed vs time)

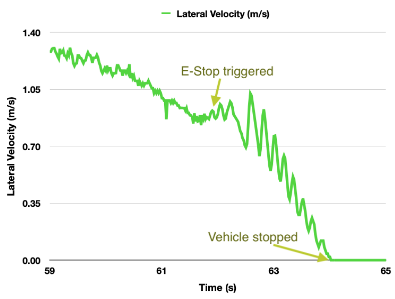


Figure 28: right boundary unit test (lateral speed vs time)

Unit Test 5: Left Lane boundary is crossed

Referring to Figure 29 & 30 -- around 91.00 seconds, the remote E-Stop is triggered. At 93.16 seconds, the vehicle came to a complete stop. That is 2.16 seconds from remote E-Stop initialization to a complete stop during a left turn.

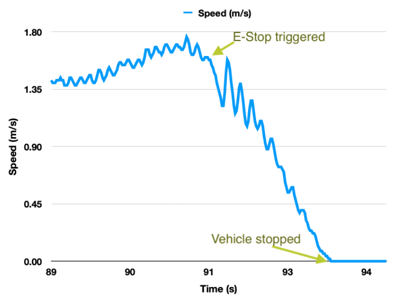


Figure 29: left boundary unit test (speed vs time)

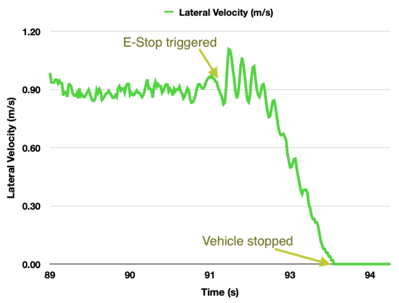


Figure 30: left boundary unit test (lateral speed vs time)

Unit Test 6: Object detection

Referring to Figure 31 & 32 -- around 26.24 seconds, the vehicle recognizes the object is within the set safe distance of 4.5 meters. The vehicle begins to brake. At 28.44 seconds, the vehicle came to a complete stop. That is 2.20 seconds from recognizing an object in front of the vehicle to a complete stop at a safe distance of 2.9 meters.

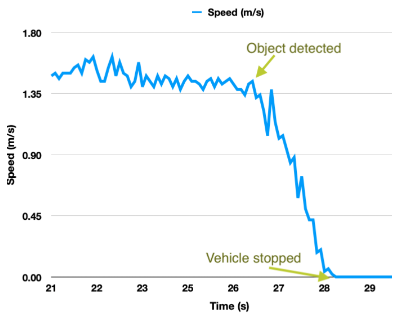


Figure 31: object detection stop unit test (speed vs time)

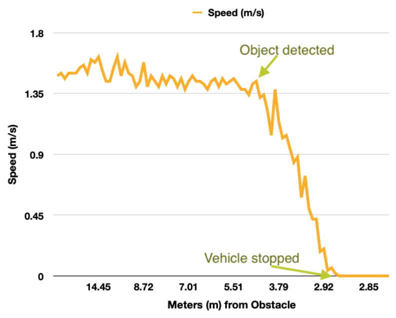


Figure 32: object detection stop unit test (speed vs distance)

Unit Test 7: Backing up operation

1 mph (0.44 mps): Referring to Figure 33 -- setting the reverse speed limit to 1 mph (0.44 mps), the vehicle starts accelerating at 5.28 seconds and reaches the target speed at 10.30 seconds. That is 5.02 seconds to accelerate from 0 to 0.44 mps, in reverse. The vehicle continues to accelerate to find equilibrium for an average velocity at 0.44 mps.

2 mph (0.89 mps): Referring to Figure 34 -- setting the speed limit to 2 mph (0.89 mps), the vehicle starts accelerating at 84.94 seconds and reaches the target speed at 88.48 seconds. That is 3.54 seconds to accelerate from 0 to 0.89 mps, in reverse. The vehicle continues to accelerate to find equilibrium for an average velocity at 0.89 mps.

3 mph (1.34 mps): Referring to Figure 35 -- setting the speed limit to 3 mph (1.34 mps), the vehicle starts accelerating at 25.32 seconds and reaches the target speed at 29.24 seconds. That is 3.92 seconds to accelerate from 0 to 1.34 mps, in reverse. The vehicle continues to accelerate to find equilibrium for an average velocity at 1.34 mps.

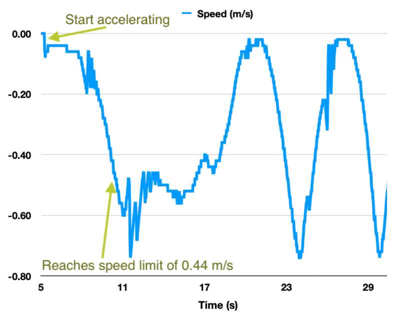


Figure 33: 1 mph reverse speed limit unit test

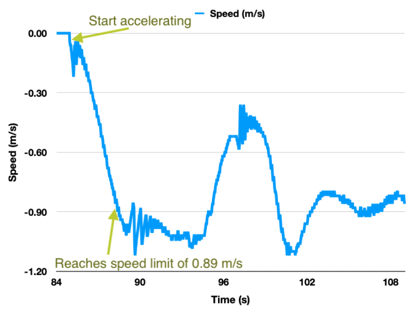


Figure 34: 2 mph reverse speed limit unit test

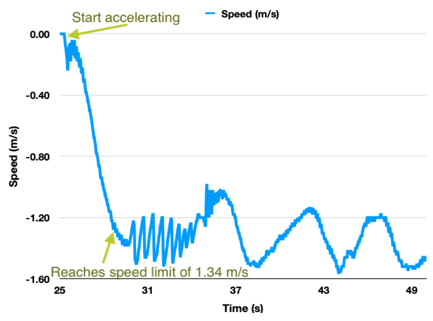


Figure 35: 3 mph reverse speed limit unit test

REFERENCES

[1] 2022 Self-Drive Design Report, http://www.igvc.org/design/2022/27.pdf, accessed 05-15-23
[2] Mako g-319, accessed 05-15-31, https://www.edmundoptics.com/p/allied-vision-mako-g-319-1-18-inch-color-cmos-camera/33094
[3] 1st vision 1" 2 to 3 megapixel oem lens series, https://www.1stvision.com/lens/spec/1stVision/LE-MV3-0618-1, accessed 5-13-21
[4] Velodyne puck, https://velodynelidar.com/products/puck/, accessed 05-15-2023
[5] URG-04LX-UG01, https://hokuyo-usa.com/products/lidar-obstacle-detection/urg-04lx-ug01, accessed 05-15-23
[6] Swift navigation piksi multi gnss, https://www.swiftnav.com/piksi-multi, accessed 05-15-2023
[7] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source robot operating system," in ICRA workshop on open source software, vol.3, no. 3.2. Kobe, 2009, p. 5.
[8] Paul, N., Pleune, M., Chung, C., Faulkner, C., Warrick, B., Bleicher, S., A Practical, Modular, and Adaptable Autonomous Vehicle Research Platform, IEEE International Conference on Electro Information Technology 2018
[9] Mitchell Pleune, Nicholas Paul, Charles Faulkner, C. J. Chung, Specifying Route Behaviors of Self-Driving Vehicles in ROS Using Lua Scripting Language with Web Interface, 2020 IEEE International Conference on Electro/Information Technology
[10] Redmon, Joseph et al. "YOLOv3: An Incremental Improvement". arXiv. (2018)