University of Detroit Mercy (UDM)

Team “Operation 313” Proudly Presents: “Annie”

Date of Submission: 4/27/2025

Team Captain:
Evan Varga – evanvarga@gmail.com

Team Members:
Oriekaose Agholor – agholor2004@gmail.com
Andres Diaz Navas – andresdiaz2100@gmail.com
Brandon Smith – BSmithEduAcc@gmail.com
Dalton Kanerva – dalton.kanerva@yahoo.com
Edrees Ahmed – edreesahmed27@gmail.com
John Gazdecki – jgaz3003@gmail.com
Linu Hanna – linuhanna@yahoo.com
Simon Yeldo – simonyeldo9@gmail.com
Sumaia Alghaiti – sumaiaalghaiti@gmail.com

Faculty Advisors: Michael Santora, Youssef Bazzi

Statement of Integrity: We have neither given nor received unauthorized aid in completing this work, nor have we presented someone else’s work as our own.

Conduct Of Design Process, Team Identification and Team Organization

Introduction:

Operation 313 proudly presents “Annie”, short for “Anomaly.” Annie was developed as a fully autonomous vehicle capable of following standard road laws at low-speed (3-5mph). It was designed to the specifications of the IGVC, with the intention to compete May 30 – June 2, 2025.

Organization:

The team was divided into sub-groups tasked with individual aspects of development. These sub-groups are Hardware, Navigation, Simulation, System Integration, and Vision and Sensing.

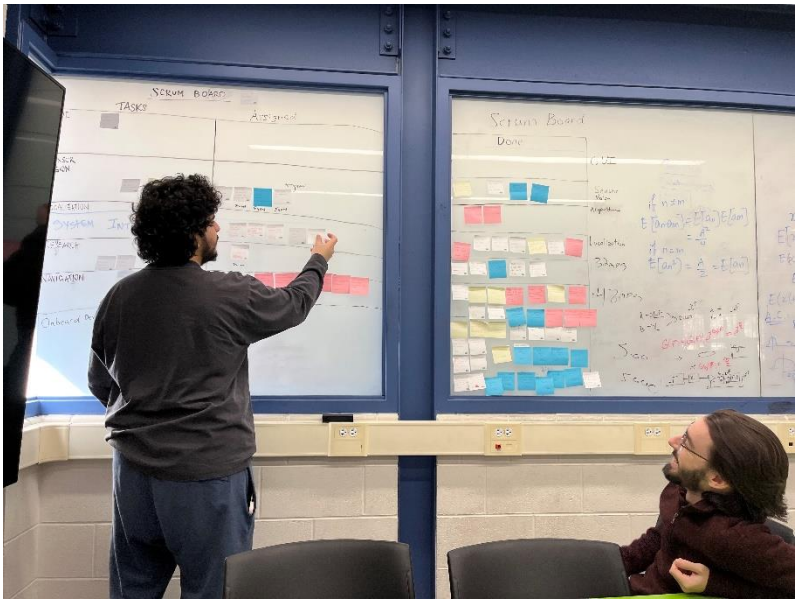
Member Name	Academic Department	Assigned Sub-Groups	Hours Contributed
Evan Varga	Electrical Engineering	Vision and Sensing – Lead System Integration	250
Andres Diaz Navas	Electrical Engineering	Navigation – Lead	241
Brandon Smith	Robotics and Mechatronic Systems Engineering	Navigation System Integration Vision and Sensing	252
Dalton Kanerva	Robotics and Mechatronic Systems Engineering	System Integration – Lead Hardware	372
Edrees Ahmed	Electrical Engineering	Hardware Navigation Simulation	361
John Gazdecki	Electrical Engineering	Hardware – Lead System Integration	354
Linu Hanna	Electrical Engineering	Navigation Vision and Sensing	226
Oriekaose Agholor	Robotics and Mechatronic Systems Engineering	Simulation – Lead System Integration Navigation	380
Simon Yeldo	Electrical Engineering	Navigation	299
Sumaia Alghaiti	Electrical Engineering	Vision and Sensing Navigation	320

Labor Cost at $20/h: $61,100

Design Assumptions and Design Process:

The design process started from scratch, and changes to IGVC requirements prompted the development of an entirely new robot platform, Annie. First, the IGVC requirements and functional tests were examined to determine general size and construction of the robot. Next, we looked at what information Annie must gather to behave in the desired way. With this foundation set, the hardware components and software algorithms were selected for gathering and processing environmental information, allowing for autonomous robot operation. From there, an iterative process was employed for prototyping and validation until final design choices were reached.

Task assignments and updates were given at weekly meetings, and a scrum board was used to track progress and individual tasks.



System Architecture of the Vehicle

Mechanical, Power, and Electronic Components:

Mechanical Component	Quantity	Total Cost
BotWheel Explorer Kit	2	$598
Battery Enclosure	1	$17.99
LeMotech Electrical Box, IP67, 11.8”x7.7”x5.2”	1	$39.99
Outdoor Electrical Junction Box, 18.1”x12.6”x6.3”	2	$139.98
LeMotech Electrical Box, IP67, 4.7”x3.5”x2.7”	1	$14.99
Makerele Cable Gland Assortment Kit	1	$26.49
Makerele NPT ¾" Cable Glands	1	$9.05
Rubber Mat for Waterproofing Measures	1	$11.65
3D Printing PLA Filament	2	$36.00

NUC and Router Mounting	2	$37.98
6061 Aluminum Bar 1/4" Thick x 12" Wide, 2 Feet Long	2	$116.20
Flat-Surface Machine Bracket 6061 Aluminum, 60mm x 20mm	16	$92.16
Medium-Strength Steel Serrated Flange Locknut Class 8 (100)	2	$13.32
Cart-King Caster Swivel with 5" Diameter Red	2	$56.40
6061 Aluminum Sheet 0.160" Thick, 6" x 48"	1	$59.61
Class 10.9 Steel Hex Head, M6x 1 Thread, 40 mm Long (5)	8	$75.36
Class 10.9 Steel Hex Head, M6x 1 Thread, 25 mm Long (5)	8	$65.92
Threaded T-Slot Nut, M6 x 1 mm Thread, for 8 mm Wide Slot	16	$84.64
6x1/2” Flat Head Screws	5	$15.79
6x5/8” Flat Head Screws	1	$3.69
6x3/4” Flat Head Screws	1	$3.69
6x3/4” Pan Head Screws	1	$3.69
4” Cable Ties	1	$6.44

Power Component	Quantity	Total Cost
12.8V 30Ah – LiFePO4 Battery	6	$479.94
36/43.8V 10A LiFePO4 Charger	1	$42.99
DC-DC Buck Converter 48V-24V	1	$35.14
DC-DC Buck Converter 48V-12V	1	$29.99
Fuse Block Panel with LED warning indicator	2	$35.18
1A Automotive Fuses (25 pc.)	1	$5.79
2A Automotive Fuses (20 pc.)	1	$4.99
3A Automotive Fuses (20 pc.)	1	$4.99
4A Automotive Fuses (20 pc.)	1	$4.99
Shrink Tubing for Wire Connections	1	$11.97
Copper wire Lugs	1	$15.88
150A Circuit Breaker with Manual Reset	1	$24.99
20A Circuit Breaker with Manual Reset	2	$31.90
Set of Bus Bars	1	$23.99

Electrical Component	Quantity	Total Cost
NUC	1	$779.00
TP-Link AXE 5400 Archer AXE75	1	$158.99
Proscilla GT1290	1	$1395.26
Proscilla GT1290 I/O cable	1	$71.28
3.5mm/F2.0 Edmund Optics Lens	1	$606.00
GPS-RTK-SMA ZED F9P Breakout	2	$519.90
GNSS Multi-Band Helical Antenna	2	$239.90
Ellipse Series Inertial Sensor (IMU)	1	$2000.00

OS1 Ouster LiDAR	1	$18000.00
LED Strobe Light for Emergency Warning	1	$24.99
Wireless e-Stop 2.4GHz Transmitter	1	$330.00
Wireless e-Stop 2.4 GHz Receiver	1	$187.00
90-Degree USB-C Adapters	3	$17.81
90-Degree HDMI Adapters	1	$9.50
USB-A 1 to 4 Extender	1	$10.27
USB-C Cables	1	$7.41
90-Degree Coaxial Cable Adapters	1	$8.03

Total Architecture Cost: $26,645.60

Total Development Cost (Labor + Hardware): $87,745.60

Safety Devices

Safety was at the forefront of our minds throughout the development of Annie. Annie is equipped with a mounted e-stop located in an accessible location at the back center of the vehicle. In addition, there is a blinking light that will indicate when the robot is in autonomous mode. Furthermore, a wireless e-stop is equipped that allows any user to shut down Annie remotely in the event of an emergency.

In addition to shutdown features, all electrical components are fed through fuses, thus protecting against over-current in the event of a malfunction. Lastly, there are circuit-breakers for 12V components, 24V components, and the batteries themselves, allowing for the disconnection of hardware before it is handled by a person.

Significant Software Modules

The software modules are split into individual ROS2 nodes, allowing for specific tests to be run at any time. For example, during a static pedestrian detection test, it is unnecessary to run the lane detection and vehicle navigation code, as the robot does not need to move for this test.

Modularizing software also allows for easy transitions between various states of operation. This provides value when Annie is in full autonomous mode, where it must transition between lane-keeping, lane-merging, intersection behavior, etc.

Documentation of Integration:

The platform and layout of the vehicle provide simple integration of hardware and software systems. Batteries are mounted over the wheels for balance and have short feeders going into electrical enclosures. The 12V enclosure houses the NUC and router, which are next to each other for packaging efficiency; furthermore, GPS units are close to the NUC for data and power plug in. All device cables are as discrete as possible while maintaining ease of access. The software and algorithms for controlling Annie exist on the onboard NUC. This makes Annie's systems clean, simple, and primed for success.

Effective Innovations

There are several innovations within the design of Annie. The power system and vehicle frame were designed to account for modifications and future additions. The 12V and 24V buck converters have spare capacity and multiple slots available in their respective fuse boxes. The vehicle frame is made from slotted aluminum that allows for easy changes to the location of mounted devices and mounting of additional components. There are provisions in place for including a RADAR on the vehicle, which will be a key addition, since the automotive industry uses RADAR in current production. Moreover, past vehicles from UDM have had electrical systems that are hard to access and difficult to modify. Annie utilizes weather-proof electrical boxes with strategic internal mounting schemes for easy access and modification. The slotted aluminum framing and open vehicle underside allow for efficient and discrete wiring. The vehicle framing and wiring have the space and capacity to support larger and higher voltage batteries in the future.

Description of Mechanical Design

There are several key components for Annie including motors, omni-wheels, IMU, 2D camera, LiDAR, NUC, router, Arduino Uno, emergency stops, lights, and antennas that were acquired as ready-made units. The full assembly was modeled in SOLIDWORKS to check fit and design mounting solutions before beginning physical assembly.

The mechanical design started with determining space and dimensional requirements. It then continued with CAD layout iterations to improve clearances, access, and organization. The material selected for the chassis tower was 3030 aluminum T-slot extrusions, which offers excellent strength and mounting flexibility. The lower frame is composed of 6061 aluminum stock for its lightweight durability. The brackets and plates were custom-designed in SOLIDWORKS and plasma-cut or CNC machined in-house, which streamlined assembly and enhanced customization options.

Instead of traditional suspension, the vehicle isolates sensitive parts like the IMU, which was mounted with a custom 3D-printed bracket dampened by rubber O-rings between the device and the robot base. Aluminum motor mounts were carefully designed to handle load without introducing stress points, which improved drive motor reliability.

Sensitive electronics were installed on ABS plastic mounting plates inside enclosures for weather protection, clean wiring, mounting location flexibility, and easy servicing. The externally mounted sensors and devices with acceptable IP ratings used a mix of 3D-printed PLA and aluminum mounts, which kept the overall design lightweight and stable.

For weatherproofing, Annie utilizes sealed electrical boxes with water-resistant inserts and outdoor-rated connectors. Elevating critical components on the vehicle, coupled with 3D-printed covers, protects against water damage without resorting to flexible coverings. Therefore, the vehicle is strong, modular, outdoor-ready, and prepared for the IGVC competition and future adaptations to the vehicle.

Description of Electronic and Power Design

In designing any mobile vehicle, it is crucial to have efficient and safe power systems to run sensors and electronics. The first step to design Annie's power systems was analyzing the voltage and power requirements of motor controllers, sensors, and other electronics needed for vehicle operations. Batteries were then selected based on the voltage range of Annie's motor controllers and total power demand. The design includes weatherproof enclosures meeting the minimum need of IP42 ratings to protect sensitive electronics and power equipment from light rain. Thus, all of Annie's electrical systems are successfully powered from the safety of enclosures.

The design utilizes two sets of three Lithium Iron Phosphate (LiFePO4) batteries (12.8V nominal) in series configurations. The vehicle run time is approximately three hours with the ability to quickly change battery sets for an additional three hours. A single 43.8V charger was purchased for the battery sets and has been tested for functionality after initially charging the batteries individually. The maximum charging time for the battery sets is three hours. There are 12V and 24V systems on the vehicle, so there are buck converters to step the main battery voltage down to fuse boxes that feed the devices on Annie. There are also DC circuit breakers to cut power to the buck converters and one for the main batteries. Additionally, there are positive and negative bus bars for connecting the vehicle systems to the main batteries with easy swapping.

The vehicle has an NUC with an Intel CPU for onboard processing, a TP-Link router for Ethernet and WIFI capability, plus a Raspberry Pi connected to a CAN HAT for motor control communications. A LiDAR and Proscilla 2D camera are used for detecting lane lines, road signs, and obstacles. Embedded wheel encoders, an Inertial Measurement Unit (IMU), and GPS are used for vehicle odometry and subsequent software control. A mechanical e-stop is set up to stop the vehicle when pressed. Additionally, there is a wireless e-stop receiver that will mechanically cut power to the propulsion when activated by the transmitter. Lastly, a light is connected to Arduino controls via relays. This light blinks when the vehicle is operating autonomously.

Description of Software System

The 2D camera used (Proscilla GT1290) was set to capture images at 12 frames per second. The raw image data was published on a ROS2 topic, since it would have to be processed in multiple ways.

Blob detection is needed to identify pedestrians wearing orange construction vests. To meet this requirement, the RGB image is converted to HSV, and passed through a thresholding filter that sets all non-orange pixels to black. The image is then converted back into RGB, matching the formatting of standard ROS2 messages. The image containing only orange pixels is then published as the output of the 'Pedestrian Detection' algorithm.

For lane detection, the raw camera data is cropped down to below the horizon line. This serves to eliminate unnecessary information, speeding up processing. First, the blob detection algorithm is adjusted to extract yellow pixels, since some lane lines will be yellow. The remaining yellow pixels are converted to white, which will be added back into the image later. From there, the original cropped image is converted to grayscale. Then, custom contrast-enhancement transforms are applied to accentuate the white lane lines. The binarized yellow is then added back to the image. Next, the image is convolved with a Sobel kernel to extract transitions from dark to light values. The output of this is put through another custom threshold such that only strong transitions remain, leaving white pixels only on the transitions from road to lane line. Next, the

Hough Transform is applied, which looks for white pixels along a linear line. The two lane lines are extracted through outputting the pixel coordinates at the top and bottom of each lane line. This is published for any navigation algorithm to use for calculating vehicle behavior.

Object detection is done using an Ouster OS1 LiDAR. Data processing extracts object position and sends it through a ROS2 topic informing navigation algorithms of potential hazardous objects.

Data processing of the LiDAR can be split into three sections: filtering, ground segmentation, and clustering. Filtering reduces information bandwidth reducing delays in object detection. Initially, the point cloud generated from the LiDAR is passed through a voxel, down sampling the points. Then, four different point clouds are created with each having their own filter specifications and dimensions best suitable for particular modes of operation. One point cloud is focused on the area in front of Annie to detect tires, barrels, and pedestrians. The second and third examine the left and right sides of Annie. The fourth views the front-right quadrant to detect the presence of stop signs. After each point cloud is established, the ground plane is identified using a [0 0 1] vector, and any point normal to that vector is segmented and removed. Lastly, clustering is done through similarities in points' Euclidean distance, to characterize any points together and establish which points belong to what object. Once each cluster is established, the point in each cluster nearest to Annie is then sent to any algorithm that needs to know the object's position.

The lane lines of the course are defined through the combination of multiple image processing filters/masks, in which the Hough transform derives lines of appropriate angle. With lane lines derived, the process for lane following and centering involves tracing these lane lines to an intersection point on the image. The intersection's pixel location on the X-axis of the image is compared to the value of half the horizontal resolution of the image. This difference in pixel value is proportional to the Cross Trek Error, and is input to a PID controller, whose output is the commanded angular velocity to the vehicle. This method works especially well for curves, as the intersection of the lane lines veers to the side of the screen that the curve is bending.

For localization, the Extended Kalman Filter (EKF) is the chosen state estimation algorithm. Balancing the capacity of handling non-linearities with computational efficiency, it remains a relatively lightweight algorithm compared to other non-linear filters. The EKF also presents great flexibility, allowing the integration of sensors with different update frequencies and noise levels. It has been widely used in robotics, aerospace, automotive and navigation, thus providing users with an active and supportive community.

The algorithm works in three stages: initialization, prediction and correction. In the initialization stage, the EKF starts with an initial state estimate (position, velocity, and acceleration), or an initial guess and its uncertainty. It then goes into the prediction stage by producing a prior estimate of the next state and a prior covariance matrix using a state-space model. Based on the prior state estimate, a measurement prediction is then produced. The first step in the correction stage is the calculation of the Kalman Gain, which weighs how much the model estimate should be corrected. The Kalman Gain is calculated using the measurement noise covariance matrix, which quantifies the level of "confidence" in sensor data. The state estimate and the covariance matrix are then updated, producing the posterior state estimate and the posterior covariance matrix.

The vehicle is equipped with wheel encoders, an IMU and a GPS module. The wheel encoders use a localization method known as Odometry, which is the estimation of position from internal motion sensors. Odometry relies on dead reckoning, where position is calculated by quantifying wheel motion (ticks) over time. The IMU is composed of a gyroscope, accelerometer and magnetometer, and it provides an estimate of velocity and orientation. The GPS module is instrumental in determining the vehicle's global position.

When integrated into the EKF, these sensors participate in the correction of model predictions. The introduction of redundancy through multiple sensors ensures a more robust localization against sources of inaccuracy, such as sensor malfunctions and measurement errors.

In the waypoint following algorithm, the EKF output is used in an iterative commanding of angular and linear velocities according to errors in heading and position. This consistently changes angular velocity until a threshold is met for the desired position. This algorithm is used in merging, parallel parking, and left turn functions.

Cyber Security Analysis using RMF

Cybersecurity is an important consideration in the automotive industry, especially as the development of more autonomous cars continues. There are several ways to undermine the vehicle's integrity, such as CAN bus attacks, exploitation of vision system vulnerabilities and ROS2 network dependencies.

The CAN bus is a critical subsystem for vehicle control and should be categorized as a high-impact level under Risk Management Framework (RMF) guidelines. Disrupting the CAN bus can cause loss of steering and speed control. For example, the lowest CAN ID injection is a well-known cybersecurity threat that has caused many issues for the automotive industry. Some of the most recent incidents were the CAN bus attacks against Toyota's newer RAV4 models. There are various methods to mitigate the effect of possible attacks, such as implementing a whitelist of accepted CAN IDs, to ensure only approved ID lines reach the ECU.

The vision systems of vehicles, including those for traffic sign detection, present easily exploitable vulnerabilities. Vision systems are considered high impact because any false positives could result in unsafe behavior, such as sudden stops. Issues of this type are common in Tesla vehicles because of camera dependency. Integrating object detection via LiDAR with static obstacle detection could improve robustness. At present, the vehicle detects static signs before carrying out image processing; however, improvements can be made for better text recognition. This would ensure that text imperfections will not be as problematic.

Finally, the vehicle is highly dependent on ROS2 nodes. The network that these nodes are on provides a channel for the injection of malicious commands. If the ROS2 network is compromised, DDS security features such as topic encryption and high-security firewalls would prevent further damage. Also, network monitoring that records node connections and traffic activity can raise an alert if anything suspicious is detected.

Analysis of Complete Vehicle

Throughout this extensive project, many lessons and experiences were learned specific to each member and their tasks. However, as a group, the lessons learned together can be divided into two categories: hard skills and soft skills. The hard skills gained came from learning about vehicle autonomy and the different devices and systems that make it work. We also got experience coding algorithms, working with electrical components, and solving mechanical problems. On the soft skills side, teamwork was the most important thing we developed. This project relied heavily on working together, dividing up the workload, and helping each other through the challenges. Without teamwork, Annie wouldn't have made it this far.

One of the biggest hardware failures we faced had to do with Annie's wheels. During testing, we found that the added weight of the frame, hardware, and payload decreased the top speed to under 5mph. To fix this, we added a second wheel on each side and installed two more ODrive S1 speed controllers; however, this introduced CAN bus communication issues between the controllers. This caused the wheels to operate improperly. Currently, Annie can operate with one set of wheels but will not meet the speed requirements of the competition. We're still working on solving this problem, but there's a chance it will still be an issue when competing.

Even though Annie's drive system is limited, other parts of the vehicle worked well. The IMU and GPS were tested separately and performed as expected. We used them to build our localization system, which combines GPS, IMU, and encoder data through an EKF. This system worked well in simulation, giving us accurate position estimates for waypoint navigation.

For safety, Annie is equipped with a mechanical e-stop mounted on the back and a wireless e-stop that lets us shut her down remotely if needed. All electrical components are protected with fuses and circuit breakers, and the sensitive electronics are housed in weatherproof enclosures. The frame is made from T-slot aluminum, which is strong and easy to modify. We also mounted the IMU on rubber dampers to protect it from vibrations.

Going into the competition, we know that Annie's drive system is our biggest risk. If the CAN communication between the motor controllers can't be fixed, we will have to run her on one set of wheels, which means she won't be able to reach the required speed. We have spare cables and controllers ready, and we'll continue troubleshooting during the competition to see if we can get the second set of wheels working.

As for predicted performance, here's what we've seen so far:

- Speed: We expected around 1.5 m/s with both sets of wheels, but we're only getting about 0.6 m/s with one set.
- Battery life: Annie can run for 2.5 to 3 hours on the current LiFePO4 battery setup.
- LiDAR Distance: The distance seen in front of Annie ranges from 1-25 meters away. The left and right sides go to a maximum distance of 10 meters, and any stop signs within 10 meters are detectable through the LiDAR.
- Localization accuracy: In simulation, waypoint accuracy is about ±0.2 meters, but in real-world GPS-only tests, it's closer to ±0.5 meters. We haven't fully tested EKF localization on the actual robot yet because of the drive issues.
- Obstacle detection: The LiDAR can detect obstacles up to 20 meters away, and the camera picks up lane lines and signs within 10 meters.
- Handling complex obstacles like switchbacks, center islands, and dead ends was successful in simulation, but we haven't been able to test these scenarios in the real world yet.

For software testing, we used GitHub for version control and tracked bugs using GitHub issues. We tested individual algorithms and ran integration tests in simulation. Our main testing environment was Gazebo, where we tested lane following, obstacle avoidance, and intersection behaviors.

Physical testing has been limited because of the drive system problems. We've tested the IMU, GPS, LiDAR, and camera separately and confirmed they work. But the full system, combining sensors, localization, and navigation algorithms has only been tested in simulation so far. In the end, Annie's software and sensors are ready, but the hardware issues with the drive system are holding her back.

Unique Software, Sensors, and Controls for Self-Drive

Obstacle Navigation

The vehicle utilizes Lidar obstacle clustering to determine the position of obstacles with respect to the vehicle's coordinate frame. The functionality of stopping at an obstacle lies in creating a threshold in the X direction. Once this threshold is passed, the vehicle will be commanded to stop.

Navigation around an obstacle relies on a separate function called lane state detection. For the IGVC course our team developed two means of deriving these states: dashed line detection and yellow line detection. Dashed line detection finds the lane lines, and counts segmentation in either lines, if more segmentations are found in the left lane line, the vehicle knows the left lane line is dashed, therefore it's in the right lane. The second method is yellow line detection, which applies a mask to the image and filters yellow pixels. The image is converted to a binary map where the occupancy of the left and right side of the screen is compared. The higher value of occupancy count indicates there is more yellow on that side of the screen, meaning that the centerline is located on that side of the screen. This tells Annie she is in the lane opposite the side of the detected yellow centerline.

Once the vehicle has determined the lane state, it will generate a point a safe distance before the obstacle in the opposing lane, and use waypoint following to travel to that point, in which lane following will take over and recenter the vehicle in the other lane.

Intersection Management

We verify the presence of an intersection by detecting horizontal lines in the camera image. To accomplish this, we use the Hough Transform specifically tuned to detect lines with angles close to horizontal. By cropping the lower portion of the image and applying the Hough Transform with an appropriate angle filter, the vehicle can robustly identify stop lines that typically appear at intersections. When one or more strong horizontal lines are detected, the system transitions into intersection management mode.

There are three possibilities for intersection management: right, left, or straight.

Right Turn

When turning right at an intersection, the vehicle is lane following by maintaining a set distance between the center pixel of the screen and the right lane line. This cross-trek error is plugged into a PID controller which controls the angular velocity toward and away from the right lane line.

Left Turn

The lack of lane line presence is a major issue when turning left at an intersection. Our team navigated this by utilizing the fact that the IGVC course mimics dimensional road standards. An intersection of roads, both with lane lines 10’ in width, allows us to calculate the radius of curvature our vehicle will need to follow to end up in the correct lane after turning. When the vehicle encounters a left turn intersection, it calculates the path along this curve, interpolating points along the curve, and finally employing the waypoint following algorithm until either the curve is complete, or the lane lines are visible again. When the lane lines become visible, the vehicle will continue with standard lane following procedure.

Straight

When progressing straight through an intersection, the vehicle initially drives forward for a preset duration of time. This approach allows the robot to safely clear the intersection without requiring immediate lane detection, accounting for gaps where lane markings may be missing. During this period, the robot maintains a constant linear velocity with zero angular velocity to ensure it moves straight ahead. After the set time elapses, the system begins searching for lane lines again. Once two distinct lane lines are detected, standard lane-following behavior is resumed, using cross-track error and PID control to maintain the correct position between the lanes.

Parking

Pull in

To do this an algebraic and cartesian solution was implemented. The center point of the parking spot is obtained by estimating the distance from the robot to each obstacle using LiDAR. Once that is obtained, the robot stops. It then rotates by about pi/2 radians using its odometry and localization. Finally, it moves forward into the parking spot until it is in line with the two barrels.

Parallel

To parallel park, the vehicle starts adjacent to the topmost barrel of the parking space. Because the vehicle only has one forward facing camera, the parallel parking procedure operates exclusively off lidar and EKF values. After pulling up adjacent to the front barrel, LIDAR clustering identifies the front and rear barrels position in reference to the vehicle. With these barrel locations, a binary occupancy map is generated, with the shape of a parallel parking space. It is parameterized according to the coordinate values of the barrels. The dimension of the parallel parking space is taken to be the common standard of 8’6” x 23’, however these values are easily changeable if the dimensions of the IGVC-provided space differ. Because the binary occupancy map cannot have negative array values, all coordinates are displaced by 500 and distances are magnified by 10 for a higher resolution. The location of the parallel parking space on the binary occupancy map is created where it would be with respect to the vehicle, whose location is always (500,500) due to the previously mentioned displacement. With the vehicle’s starting position specified, an A* path is generated using the vehicle’s position and orientation as the starting location, and the center of the parallel parking space as the final position. Using the waypoint following algorithm, the vehicle then follows the points on the path until parallel parking is completed.

Functionality integration

The core functionalities of the vehicle are tied together using a state machine generated through MATLAB state-flow. The operating modes, such as obstacle navigation, lane following, and intersection handling operate according to circumstantial flags and a global intersection timer. The intersection timer is necessary for the vehicle to decide how to behave at each intersection. The vehicle's default state is lane following. Algorithms that identify obstacles, potholes, and road signs cause transitions to other states.

To enter these states, multiple sensor inputs are utilized. LiDAR data is processed to trigger the sign flag when a highly reflective object is detected. Annie then uses the camera's stop sign reading algorithm to verify the stop sign's authenticity. If obstacles are detected on the vehicle's path, the lane state detection algorithm runs, so that the lane change algorithm knows which direction to merge.

When encountering the final intersection, the value of the intersection counter enables the pull-in parking algorithm, completing the course.

Stop Sign Read and Overlay

When the LiDAR flags something that could be a stop sign, the sign validation algorithm starts. It reads the latest camera image and applies filters to extract red pixels coordinates through RGB thresholds. The resulting masked image is passed through median filters, and morphological processes to close any small gaps in the mask. This causes the text to appear much cleaner. .finds the strongest red region, crops around it, and inverts the colors to make the text easier for OCR to read. The column and row sum vectors are calculated to identify the indices of where the stop sign exists in the image. It uses these values to dynamically crop any camera image around the stop sign. With less information present, the Optical Character Recognition (OCR) algorithm provides more accurate results. If the read text does not say "STOP", the HSV filtering described in the blob detection algorithm is used to filter for red pixels. The same processing used on the RGB image is applied to the HSV image. This redundancy gives greater confidence and robustness of stop sign validation. The resulting text is overlayed onto the original camera image and published as the final output of this algorithm.

Simulation

To speed up and reduce the cost of developing the vehicle, in addition to testing developed algorithms, a clearpath husky and TurtleBot robot was used. Sensors such as LiDAR and 2D camera were simulated, with their respective algorithms validated using ROS2 nodes. MATLAB and ROS2 on a Linux Ubuntu operating system was used to develop the algorithmic control of the vehicle. The clearpath husky was simulated first, as that was the machine we planned to use; however, due to malfunctions, the team decided to build our own vehicle with the BotWheel Explorer as a starting point. Due to odometry issues in simulating the husky, the turlebot3 robot was employed. It's better odometry allowed for continued development of other algorithms in simulation.

The algorithms were implemented in such a way that the simulation nodes of the virtual sensors could be replaced with the actual nodes of the physical sensors. This improves editability and compatibility between simulated and real-world algorithms.

Initial Performance Assessment

Annie was built from the original BotWheel robot and most of our algorithms were developed from scratch. The following is the initial performance of the robot:

• The chassis and hardware setup can withstand paved roads without coming apart

• The hardware enclosures and casings are IP42 rating, capable of withstanding light rain or splashes of water

• The robot can move at a current max speed of 2.2 mph

• The sensors are fully functional with readings within an acceptable margin of error

• The robot lasts 3 hours on one battery pack and has another 3-hour set in reserve.

• The following algorithms work and have been tested in the real world

- Orange blob detection for pedestrians
- Stop sign detection
- Lane following

The rest work in simulation but have yet to be tested in the real world.

• The Localization waypoint accuracy is about ±0.2 meters for simulation; in real world GPS-only tests, it is about ±0.5 meters.

• The LiDAR range is about 20 meters for object detection in front of Annie. Barrels on the left and right side are detectable at 10 meters, and stop signs are detectable at 10 meters in front of Annie.