# University of Detroit Mercy

# 48 GT/ E-Mobility

**Date Submitted:** May 18, 2023

**Team Captain:** Allison Sherman: shermaac@udmercy.edu

**Team Members:** Sima Alim: alimsi@udmercy.edu, Abigail Dertinger: dertinam@udmercy.edu, Nicholas Ibegbu: ibegbunc@udmercy.edu, Meghan Johnson: johnsom15@udmercy.edu, Emil Kyek: kyekem@udmercy.edu, Susan Magana: maganasm@udmercy.edu, Alan Aguirre Sullivan: aguirral@udmercy.edu, Mario Padilla Rodriguez: padillma1@udmercy.edu, William Vallespir: valleswb@udmercy.edu

**Faculty Advisor:** Dr. Michael Santora, Dr. Mohamed Nafea

Faculty Advisor Statement:
We certify that the engineering design in this vehicle undertaken by the student team, consisting of undergraduate
students, is significant and qualifies for course credits in senior design and in the undergraduate program
respectively.

Advisor Signature:

# Introduction:

The Intelligent Ground Vehicle Competition (IGVC) is a multidisciplinary, theory-based team project in which the goal is to create a class five autonomous vehicle. Held at Oakland University in Rochester, Michigan, in June, the competition draws college teams from many states and even different countries. To accomplish this task, the senior design class split into three sub-teams based on project specifications. The teams were decided as follows: Controls, Navigation/Localization, and Detection. It begins with the control team preparing the vehicle to send and receive information relating to the operation of the vehicle, from steering wheel and tire position to velocity, and using that data to operate the Dataspeed inc drive-by-wire system. The direct operation of the vehicle is dependent on information received from Navigation/Localization. The Navigation/Localization team provides the desired velocity and direction of travel instructions that the controls system uses to control the vehicle hardware accordingly. These commands are the output of the Extended Kalman Filter (EKF) and navigation algorithms which combine the inputs of the different localization sensors such as GPS, IMU, Encoders and LiDAR. The information that guides these decisions is received from the Detection team, which serves as the eyes of the vehicle. The images from the camera are fed into the obstacle, traffic sign, and lane line detection algorithms. Furthermore, LiDAR is used to get the location of the obstacles, traffic signs, and lane lines. The prediction results along with locations in respect to the vehicle from these algorithms are output to both the Control and Navigation/Localization teams through ROS nodes.

## Organization:

| **Team Member** | **Responsibility** |
|---|---|
| Allison Sherman | Traffic Sign Detection |
| Sima Alim | Vehicle Control |
| Abigail Dertinger | Navigation |
| Nicholas Ibegbu | Obstacle Detection |
| Meghan Johnson | Lane Line Detection and Camera |
| Emil Kyek | Localization |
| Susan Magana | Vehicle Control |
| Mario Padilla Rodriguez | LiDAR Detection |
| Alan Aguirre Sullivan | Navigation/Localization |
| William Vallespir | Vehicle Control |

*Table 1: Team members responsibilities*

## Design Process:

The first step taken in the design process was to understand the rules and objectives needed for the Intelligent Ground Vehicle Competition (IGVC). Once the needs of the competition were identified the team worked to create a needs/metrics matrix. It was from this matrix the team was able to group similar needs which led to the creation of the sub-teams utilized throughout the year. The sub-teams are as follows: Controls, Detection, Navigation/Localization. Each team brainstormed different designs that could be implemented on the vehicle, research was done to determine the best design and the method of implementation. The next step was identifying these designs the team worked to begin implementation on the vehicle. After being implemented the next step was to test on the vehicle, upon obtaining the results the team began to improve upon the design. Once all designs were complete the vehicle went through a final testing and validation phase in order to be able to take the vehicle to the IGVC.

# Innovations:

**Traffic Sign and Obstacle Detection:**

Requirements:

The vehicle must be able to detect and determine the type of traffic signs/ obstacle in its environment; therefore, traffic sign and obstacle detection is needed. The five different signs that the vehicle must detect are (1) stop signs, (2) no turn, (3) road closed, (4) one way right, and (5) one way left. Additionally, it must detect if there are no signs present. Furthermore, four different obstacles must be detected: (1) pothole, (2) barrel, (3) tire, and (4) mannequin. This is needed so the vehicle knows how to react and maneuver on the roads.

Problem:

For obstacle detection, regardless of the type of obstacle detected the response from the vehicle is the same. On the other hand, the vehicles response differs based on the sign detected and there is the possibility of having more than one sign in view of the vehicle. Hence, the vehicle must be able to detect multiple traffic signs in one image. However, within the dataset, not all combinations of signs are found within the same image.

Solution:

The Traffic Sign and Obstacle Detection algorithm is a sequence of parallel Support Vector Machines (SVM) trained on datasets collected by the team. This allows the vehicle to detect all signs in its's view. Since the vehicle responds the same for all obstacles, one binary SVM is used for detection. The image from the camera is sent to all seven SVMs and each returns the SVM's prediction. Each SVM is trained and tested with its own individual dataset that has the same name as that SVM, which is the object it predicts. The datasets with what images are included are found in Table 2.

| Datasets | Labels (binary label) | Type of Images | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Stop Sign | No Turn | Road Closed | One-way Right | One-way Left | No Sign | Obstacle | No obstacle |
| Stop Sign | Stop Sign (1) | X | | | | | | | |
| | Not Stop Sign (0) | | X | X | X | X | X | | |
| No Turn Sign | No Turn (1) | | X | | | | | | |
| | Not No Turn Sign (0) | X | | X | X | X | X | | |
| Road Closed Sign | Road Closed (1) | | | X | | | | | |
| | Not Road Closed (0) | X | X | | X | X | X | | |
| One-way Right Sign | One-way Right (1) | | | | X | | | | |
| | Not one-way Right (0) | X | X | X | | X | X | | |
| One-way Left Sign | One-way Left (1) | | | | | X | | | |
| | Not One-way Left (0) | X | X | X | X | | X | | |
| No Sign | No Sign (1) | | | | | | X | | |
| | Sign (0) | X | X | X | X | X | | | |
| Obstacle | Obstacle (1) | | | | | | | X | |
| | No Obstacle (0) | | | | | | | | X |

*Table 2: Types of images within each dataset*

**Lane Line Detection:**

Requirements:

The lane detection system must detect lane lines in a live video stream captured by a camera mounted on a vehicle. The system needs to output the type and location of the lane lines and their accuracy to be used for navigation/localization and control of the vehicle.

Problem:

For the detection of lane lines, the response of the vehicle depends on the type of line detected and the distance from that line. For lane keeping, a constant distance must be kept from the edge of the lane lines while following their shape down straight aways and around curves. This requires the identification and classification of lane lines, their position and their trajectory.

Solution:

To achieve the desired output, the system will first perform preprocessing on the images to filter noise and isolate white areas, then use connected component analysis (CCA) to identify and localize lane lines.

When the image is received by the system it is converted to greyscale and then binary. The image is then cropped to the region of interest (ROI), cutting out the horizon and other unnecessary details to improve efficiency and accuracy of the algorithm. A white mask is applied to the image and then eroded to remove noise and isolate areas of white pixels. The image is transformed into the bird's eye view and passed to the rest of the algorithm. A visual representation of the preprocessing system is shown in Figure 1.

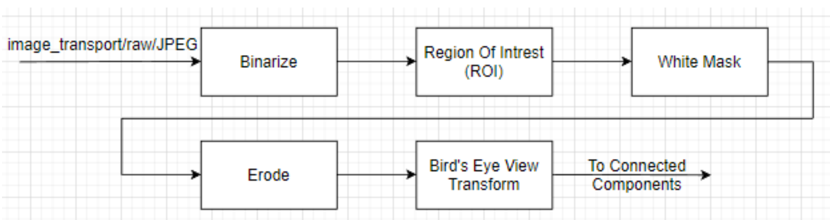


*Figure 1: Overview of Preprocessing Functions*

Lane identification and validation is done using connected component analysis (CCA). The CCA is a way to create objects out of areas of connected white pixels. Once the component is created, it is analyzed and easily identified. The components are then analyzed based on spatial characteristics to determine the potential locations of lane lines. The components are also analyzed by the region properties function to obtain the centroid, area, bounding box, eccentricity (height: width ratio), and other similar characteristics. These characteristics are then compared to the predicted lane locations. The boundaries of components with a high number of matches are extracted and put into a point cloud. The point cloud is then transformed to real-world coordinates using the image2vehicle function and the resulting point clouds are published and sent to the localization and navigation systems. An overview of this system is shown in Figure 2.

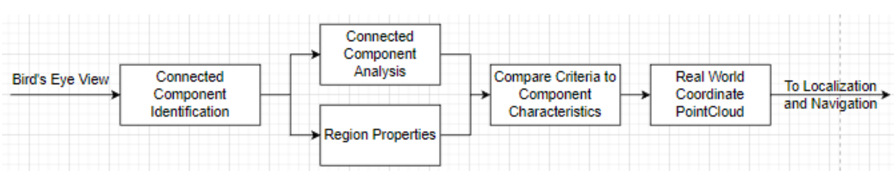


*Figure 2: Overview of Connected Component Analysis and Lane Localization*

## Mechanical Design:

The 48 GT is a Dataspeed inc. Polaris GEM e2 with a built-in drive-by-wire system. The drive-by-wire kit allows for computer control braking, steering, and shifting. The driver is able to override the system by grabbing the steering wheel or pressing on the brake pedal. The system has both CAN and USB interfaces. The steer-by-wire interface modifies the steering signal when power is applied along with when the required CAN messages are received. Additionally, the brake-by-wire interface uses a motor to physically move the pedal and add brake pressure. Finally, the shift-by-wire interfaces uses digital inputs/outputs to trigger the shift buttons in order to change gears. The steering system, brake pedal, and shift buttons all function normally regardless of the CAN messaging and the applied power to the drive-by-wire interface.

The Polaris GEM e2 has a complete factory-made body around the chassis. All computers are housed inside of the body underneath the driver and passenger seat. Furthermore, the camera is also located within the vehicle in order to help with weather proofing. Additionally, the GPS units are attached to the body of the vehicle and all wires are fed into the vehicle through the body.

## Electronic and Power Design:

The power distribution system from Dataspeed Inc has 12 channels at up to 15 amperes each at 12 volts or up to 10 ampere each for 24 volts. The power distribution system can be changed up to four units in order for a maximum of 48 channels. The system allows for computer control of fused power channels and programmable startup and shutdown sequences. There is a touchscreen which is used for both control and status. These are both available over communication interfaces including CAN, ethernet, and USB. Finally, for safety, the vehicle is equipped with 5 E-Stops. These are located on the front, back, left, right, and dashboard of the vehicle.

All the sensors (Hardware/inputs) are physically connected to an NUC, NUC stands for "Next Unit of Computing," which is a small-form-factor computer. The NUCs are used in this autonomous vehicle as onboard computers for data processing, sensor fusion, and control. This NUC is used to read and process information from all sensors by using software drivers for some individual sensors such as GPS, IMU, Encoders, LiDARs, etc and using them to connect them to ROS.

## Software Strategy and Mapping Techniques:

### Traffic Sign and Obstacle Detection:

The traffic sign and obstacle datasets are collected through a dash camera and compiled in the training repository. Additionally, the distribution of each type of image is contained within Table 3 and 4.

The Traffic Sign and Obstacle SVMs are coded in MATLAB using a linear kernel. The datasets are split into testing and training data which is found in Table 5. When training the SVM models, cross validation with holdout is used to optimize the regularization parameter (see appendix N for further explanation). Additionally, after the model is trained and cross validated, the cross-validation error is produced. This is a generalization error, and accuracy is produced by applying equation 1. Next, the trained models are used to predict the objects within the images from the test set. These images have not been previously seen by the model giving an unbiased generalization error. Both the cross validation and test accuracies are found in Table 6. This training and testing process is seen in Figure 3. Lastly, the trained models are tested on live data from the cameras (for more information about the camera data and cameras please refer to appendix L) with the use of Simulink. The SVM models within Simulink provide a prediction for the objects seen by the vehicle and outputs this into a ROS topic called cob_object_detection_msgs. This topic is sent to both the Navigation/Localization and Control teams.

$$Accuracy = 1 - error \qquad \text{equation 1}$$

| Traffic Sign data split | |
|---|---|
| **Class** | **Percentage** |
| Stop Sign | 30.2% |
| No Turn | 15.5% |
| Road Closed | 14.9% |
| One-way Right | 14.2% |
| One-way Left | 12.9% |
| No Sign | 12.3% |

*Table3: shows percentage of each sign in dataset*

| Obstacle split | |
|---|---|
| **Class** | **Percentage** |
| Obstacle | 54.6% |
| No Obstacle | 45.4% |

*Table 4: shows percentage of obstacle in dataset*

| SVM | Splitting for training and testing |
|---|---|
| No Sign | 75% |
| No Turn | 75% |
| One-way Left | 70% |
| One-way Right | 80% |
| Road Closed | 75% |
| Stop Sign | 60% |
| Obstacles | 80% |

*Table 5: Specifies the percentage of the dataset used for training*

| SVM Model | Training Accuracy | Test Accuracy |
|---|---|---|
| No Sign | 87.71% | 87.21% |
| No Turn | 81.23% | 80.89% |
| One Way Left | 86.56% | 86.52% |
| One Way Right | 85.49% | 85.45% |
| Road Closed | 88.6% | 88.09% |
| Stop Sign | 67.97% | 67.36% |
| Obstacle | 92.24% | 91.34% |

*Table 6: Shows the accuracies for cross validation and testing of the 7 different SVM models*


*Figure 3: SVM flowchart*

**Extended Kalman Filter:**

Extended Kalman Filter (EKF) is an algorithm based on the linearization of the state equation and measurement equation. It takes the data from the encoders, IMU, GPS, and LiDAR as inputs. The specifics of these data are specified in the appendix. The state measurement equation updates the state of the vehicle using the previous state at every time step.

Extended Kalman filtering is a method used to fuse multiple sensor inputs together to minimize possible errors as described in the design section above. The implementation of this method involves coordinating the multiple sensor inputs required for the vehicle's navigation. These sensor inputs would be from the GPS, the IMU, and the encoders. Before attempting to coordinate the actual sensors from the vehicle, a simulation is made to test the EKF package we are using.

## Failure modes, failure points, and resolution:

Safety is a very important consideration in a vehicle such as this, and there are many areas that need to be taken into account when designing an autonomous electric vehicle. A brief list of areas of concern would include: (1) dead battery or power failure, (2) sensor failure or damage, (3) poor weather conditions, (4) undetectable high speed objects crossing the path of the vehicle. The safety concerns listed are mitigated to some degree by: (1) charging the battery when the vehicle is not in use, (2) keeping the sensors clean and regularly inspecting for damage, (3) increasing the sensitivity of the sensors during poor weather conditions, (4) and having easy access to emergency stop buttons for those unexpected emergencies. Additionally, the obstacle and traffic sign detection algorithms are trained on images where poor weather conditions are present. Therefore, both traffic signs and algorithms are operable in more diverse weather conditions including poor conditions. During competition the vehicle is never run above 5mph, which allows for more time to react to emergency situations. This is applicable for both inside and outside of the vehicle. The vehicle is stopped remotely via the wireless E-Stop should the need arise during autonomous driving. There are also E-Stop buttons located on the front, sides, and rear of the vehicle, as well as one located inside the vehicle seen in Figure 4.


*Figure 4: E-Stop located on top of dashboard*

## Simulations Employed:

**Traffic Sign and Obstacle Detection:**

Simulations were ran within the Simulink model to test the traffic sign and obstacle detection algorithm. Images including predetermined traffic signs and/or obstacles were fed into the algorithm. The algorithm then outputted the type of sign and/or if an obstacle was present within the image. This testing was conducted with multiple different traffic signs and obstacles that would be seen in competition.

**Lane Line Detection:**

To test initial accuracy and model concepts, a simulated environment was created in the lab. The environment consisted of an 11 feet by 11 feet grid with mock white lane lines on a light grey floor in the lab. The lab also has adjustable lighting that was used to simulate different lighting conditions. Simulating an environment with low contrast between the lane lines and the road leads to a reliable and robust detection system that is still accurate in unfavorable lighting conditions.

**EKF:**

The simulation involves using a virtual robot, the TurtleBot 3, and its equipped sensors. These sensor outputs are taken by the EKF package and fused together, resulting in the output of the final transformed location of the Turtlebot 3.

The EKF package was downloaded from a GitHub repository. The essential components of the package are (I) the sensor nodes for the Turtlebot 3, (II) the navsat node which fuses the sensor data, (III) the EKF estimation nodes transform the data between the frames, and (IV) the launch file which calls all these nodes and the simulation. The main complications with the simulation are organizing the package to contain all the aforementioned parts as well as getting the data from the simulation to be fused accurately through the frames. Currently, packages compatible with the Turtlebot 3 are being researched. Previously, a turtlesim simulation was used, which used only an IMU and encoders. To simulate GPS as well the simulation needs to switch to Turtlebot 3.

Once a package is found for the Turtlebot 3, the simulation should be able to receive the Turtlebot 3 sensor data from its IMU, encoders, and GPS. Then, it transforms the data to the proper frames, and lastly fuse the data through the navsat node and output the final position data.

## Operation to Date:

Currently, all hardware and software are being integrated onto the vehicle. The camera, GPS, IMU and LiDAR are installed and operational. The camera system is sending out ROS messages to be read by obstacle, traffic sign, and lane line detection. Furthermore, the E-Stops are installed and operational.