Team Nightmare Self Drive Design Report

1 Group Info

University/College Name:	**Oakland University, Rochester, MI 48309, USA**
Vehicle/Team Name:	**Nightmare**
Vehicle Photo/Sketch/Symbol :	
Date Submitted:	**15 May 2019**
Team Captain's Name and E-Mail:	**Kaiqiao Tian tian2@oakland.edu**

Team Members Names and E-Mails:	**Faculty & Professional Advisors Name:**
Ahmad Abdelhafiz <aabdelhafiz@oakland.edu>;	Dr. KaC Cheok cheok@oakland.edu (Main)
Ahmad Kafrouny <akafrouny@oakland.edu>;	Dr. Wing-Yue Geoffrey Louie <louie@oakland.edu>;
Ana Farhat <anafarhat@oakland.edu>;	Dr. Micho Radovnikovich <mradovnikovich@dataspeedinc.com>;
Arjun Musham <arjunmusham@oakland.edu>;	Dr. Ghassan Abed <gabed@oakland.edu>
Bing Liu <bingliu@oakland.edu>,	Dr. Sami Oweis <soweis@oakland.edu>;
John Brooks <johnbrooks@oakland.edu>,	Dr. Kazuyuki Kobayashi (Ikko) <ikko@hosei.ac.jp>;
Kaiqiao Tian <tian2@oakland.edu>;	
Kiran Iyengar <kiyengar@oakland.edu>;	
Mckenzie King <mckenzieking@oakland.edu>;	
Narendra Kintali kintali@oakland.edu	
Nashwan Sebi <njsebi@oakland.edu>,	
Shakila Raheem <shakilaraheem7@gmail.com>,	
Saif Salih <sysalih@oakland.edu>	

Statement of Integrity:

We, the above team members, certify that the design and engineering of Nighmare self drive vehicle by the currently listed student team has been significant and equivalent to what would be awarded credit for a design project in an undergraduate-level or a graduate-level course at Oakland University.

I certify that the above Statement of Integrity is true.

[illegible] Professor Ka C Cheok

14 May '2019

2 Conduct of design process, team identification and team organization

2.1 Introduction

The Nightmare team consists of twenty students with various backgrounds, along with several advisors as well as sponsors. They were grouped into functionality teams including:
Drive-By-Wire, LiDAR & RADAR, Computer Vision, Path Planning & Control.
The grouping facilitates ease of assigning specific tasks and responsibilities.

2.2 Organization

To keep everyone on the same page, an organization chart was created using iMindMap software as shown below. The chart was very useful in providing ownership, and hence urgency, of the responsibility for each team or group, at a glance.

2.3 Design assumptions and design process

The iMindMap chart/project reports was updated on a weekly basis after each weekly meeting, to include the timing for delivery of the tasks by each group, as well as highlighting of deadlines, as shown below. This has also proven to be very effective in getting done on time or close to on time.

3 Effective innovations in your vehicle design

3.1 Innovative concept(s) from other vehicles designed into your vehicle

There are many concepts from other vehicles that the teams looked at in developing the functions of their subsystems. They include

ROS IDE. Robot Operating System (ROS) is an integrated development environment (IDE) that allows collaborative programming with libraries of tools to create complex robotics systems.

AI Programs. The field of Artificial intelligence (AI) has advanced by leaps and bounds the past several years. Computers can now speak like a real person, be trained to recognize the face of a person and identify objects.

Robotics Technologies. Embedded controllers, electronic devices, sensors and actuators have has also advanced at exponential rate with MEMS and nanotech the past several years.

Automotive & Robotics Standards. The auto and robotics standards are results of extensive rigorous tests, and hence have high reliability and quality.

3.2 Innovative technology applied to your vehicle

We applied the following specific technologies from the above to the Self-Drive SequoiaSD.
Navigation Stack, Gazebo, RViz from **ROS IDE.**
Computer vision algorithm including Convolution Neural Networks**.** Stereo camera, Kinect, Lidar, IMU, Microcontrollers, etc. from the **Robotics Technologies**.
GPS, automotive connectors, motors, etc., from **Automotive & Robotics Standard Components.**

4 Description of mechanical design

4.1 Overview

Drive-By-Wire Team

Ahmad Kafrouny, Ahmad Abdelhafiz, Ghassan Abed, Nashwan Sebi, Narendra Kintali, Kaiqiao Tian

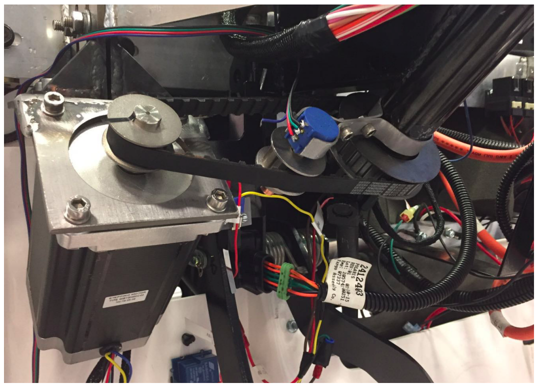
Goal. The goal for the team is to develop and implement a joystick by-wire system for controlling the steering, throttle, brake and forward-reverse shift of the vehicle. The goal is to have a high-level autonomous driving decision software control the drive-by-wire system.

Solutions. The team successfully adapted, design and implemented the following subsystems into the GEM2.

Mechanical design and components:

Steer by wire System:

The steering system in this electric car is mechanical system; the steering system has been modified by adding a stepper motor to drive the steering shaft.
The stepper motor drives the steering shaft by timing pulleys and timing belt where one of the pulleys fixed to the stepper motor shaft and the other pulley was fixed to the steering shaft.
The system is equipped with potentiometer attached to the timing belt, the potentiometer is essential to indicate the steering shaft rotation angle during driving by wire mode.

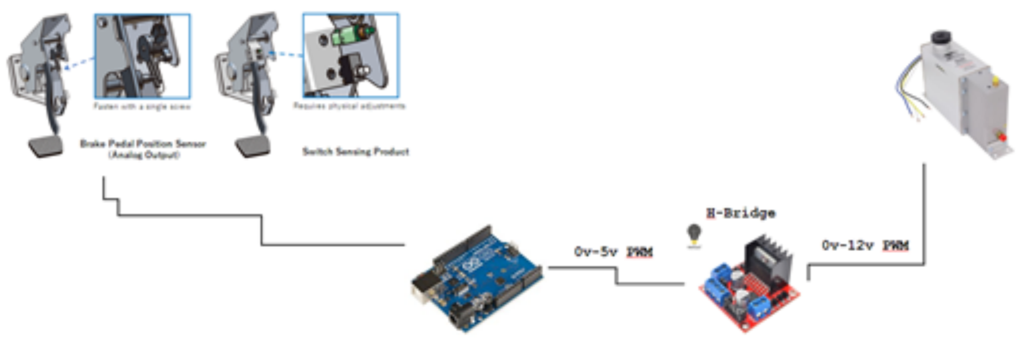


Brake by Wire System:

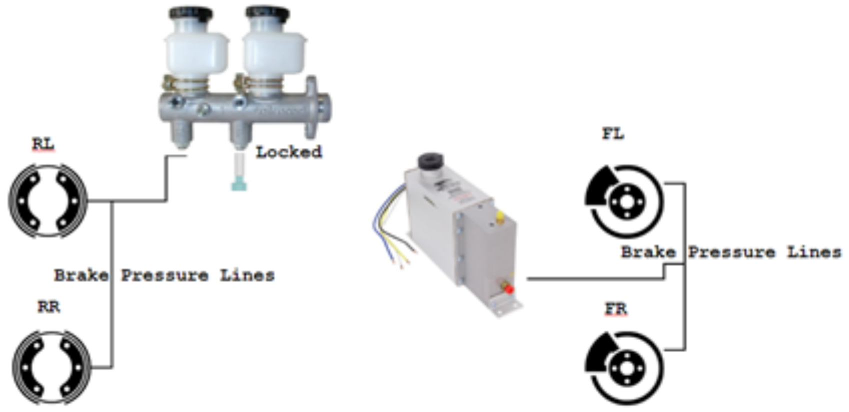
brake-by-wire technology is the ability to control brakes actuator through a Microprocessor. Anti-lock Brake System (ABS) , Electric Park Brake (EPB) , Electronic Stability Control (ESC) are also forms of brake-by-wire.

Brake-by-Wire system replaces traditional components like vacuum servo and master cylinder with electronic sensors and actuators. Drive-by-wire technology in automotive industry replaces the traditional mechanical and hydraulic control systems with electronic control systems using electromechanical actuators. In Our vehicle, we installed an Electric Over Hydraulic (EOH) actuator DX1600 from Dexter(which is commonly used for trailers) as the main actuator for the front brakes.The front brake lines are connected directly to Dexter actuator by using double banjo bolts. The traditional master cylinder has 2 10mm inlets for front and rear brakes. The EOH actuator has ¼" inlet. Therefore we had to cut new thredes for the original double banjo bolt to make it compatible with EOH actuator inlet threads.

Dexter DX1600 is capable to generate 1600psi hydraulic-pressure in the brake lines within 700ms using 12v. Since our pressure gauge shows a maximum 800psi when the brake pedal is fully pressed using the traditional master cylinder, therefore we only drive the actuator by providing 6v driven by PWM from the Arduino microcontroller that is connected to H-Bridge.


Fasten with a single screw
Requires physical adjustments
Brake Pedal Position Sensor (Analog Output)
Switch Sensing Product
0v-5v PWM
H-Bridge
0v-12v PWM

The rear brakes are still connected to the traditional mechanical master cylinder and the front brakes outlet on the master cylinder has been securely locked.



Normal Driving Mode:

The driver is still able to press the brake pedal to stop the car by creating pressure in the rear brake lines and actuate the rear brakes only.

Autonomous Driving Mode:

The Arduino is sending a PWM signal to H-Bridge which is directly connected to the control pin on the EOH actuator to control the pressure created by sending 0 to 6 volts to the actuator.

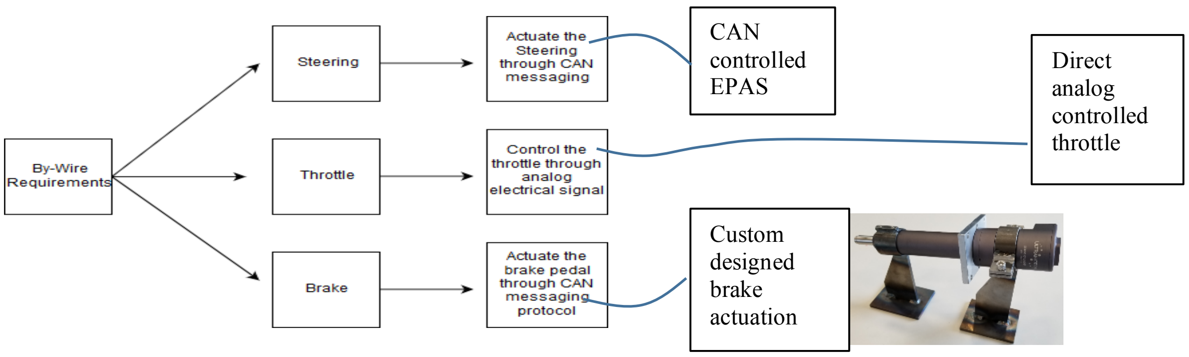
Since the EOH actuator builds-up a lot of heat in the system if it is operated for a long time when the car is in standstill mode which can damage the internal components, we installed a linear actuator to take over the EOH place.

This linear actuator is connected directly to emergency brake wire. It works as an Emergency Park Brake (EPB). After the car comes to a full stop and the speed becomes zero, the Arduino starts a 1 sec countdown. After 1 second the linear actuator pulls the hand brake lines and EOH releases the hydraulic pressure. When the system receives a drive command from ROS or the driver presses the gas pedal in normal driving mode, the linear actuator releases the brakes to let the car move forward.

Dexter DX1600 specs:

- Vented design releases pressure, moisture, and heat to protect the internal components
  - Allows actuator to be mounted in places where most actuators would overheat
- Electronic, proportional pressure valve ensures smooth, even braking
- Self-priming pump reduces time needed for brake bleeding
- Enhanced wiring detects dangerous current levels and shuts down actuator electronics in the event of a surge
- Durable aluminum housing and stainless steel fasteners resist corrosion
- Goretite seals protect the circuit board from the elements

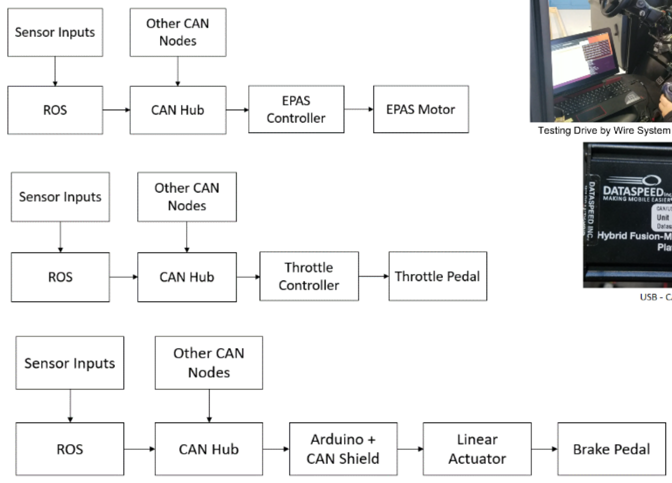
· DOT compliant
· Maximum output pressure: 1,600 psi
· Dimensions: 11" long x 6" wide x 7" tall
· Mounting hole diameter: 1/4"
· Hydraulic port size: 3/16"
· Power: 12V DC


By-Wire Requirements
Steering
Throttle
Brake
Actuate the Steering through CAN messaging
Control the throttle through analog electrical signal
Actuate the brake pedal through CAN messaging protocol
CAN controlled EPAS
Direct analog controlled throttle
Custom designed brake actuation

Drive-By-Wire Subsystems

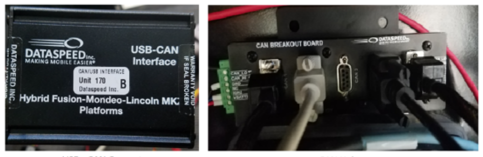
4.2 Description of drive-by-wire kit

A laptop computer with ROS was integrated to a CAN Hub (Dataspeed ADAS Kit) so it can control the steering, throttle and brake systems as shown in the diagram below. C-programs were written to actuate and test the components.

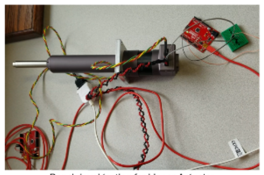


Testing Drive by Wire System using ROS and Joystick

ROS-CAN based Control Actuation



USB - CAN Converter CAN Hub


Bench level testing for Linear Actuator

Key problems overcome.

1) The EPAS in its factory setting was meant only for low torque assisted steering. It was not initially capable of turning the steering column on its own. We had to solve this issue by writing custom firmware to allow the motor to turn the column without having additional force applied to the steering wheel by the driver.

2) Calibration and tuning was required to rotate the motor smoothly. We achieve this by updating the firmware for EPAS controller, via changing the PID parameter values for EPAS controller and adding a low pass filter while sending the command to avoid override error.

3) Two designs were considered for mounting the linear brake actuator. In the end, a simpler design with set screw cylinder was implemented. It consists of steel plates being welded together and the cylinder being contained in two metal sleeves at an angle of 10°, to match up with the brake pedal.

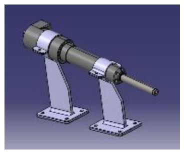
Accomplished.

The team successfully implemented and tested the subsystems for steering, throttle and brakes at the end of April 2019. The photos below show the team enjoying a ride with the remote-controlled drive-by-wire GEM2.

Drive-by-wire test runs with joystick game controller

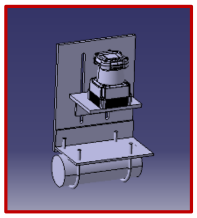
4.3 Suspension

Several mounting systems were implemented in this vehicle, as briefly described here.



- The mounts for the braking system was designed and fabricated in house using CATIA V5 and Solidworks. See photo on the right.

- The camera mounts were made using aluminum brackets to maintain a balance between stability of the mount and flexibility of orientation of the cameras. Aluminium AD20 extrusions were also used for increased mounting options.



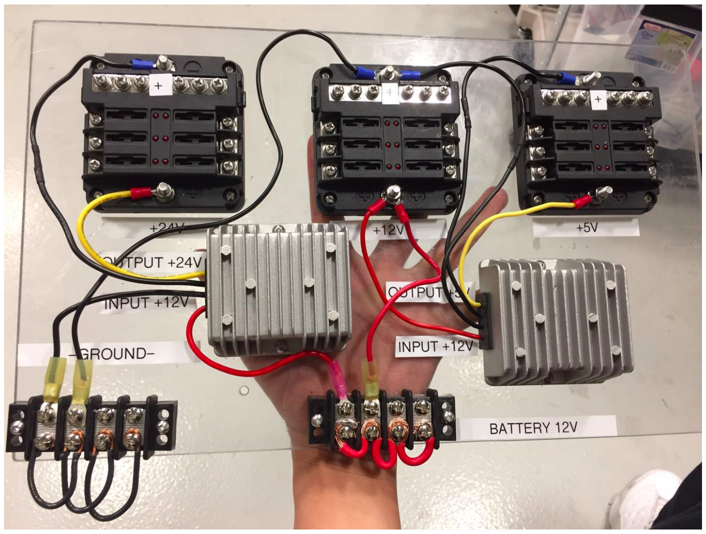
- The lidar mount was fabricated out of aluminium brackets and plexiglass. This ensures rigidity, angular and height adjustment. See photo.

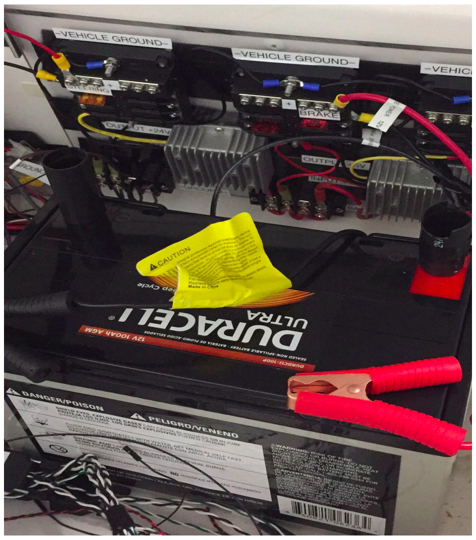
- The laptop mounts were bought and customized for our use. To reduce vibration and eventual failure of the laptops, using rubber sheets and foam sheets were considered.

- There is an additional mount customized for holding the display screen in the vehicle's cabin. This was purchased and modified to hold the screen at the required height. The mounts were designed to minimize bending and to hold the sensors without any wobbling. The materials used were mostly aluminium brackets to reduce complexity of fabrication.

4.4 Power distribution

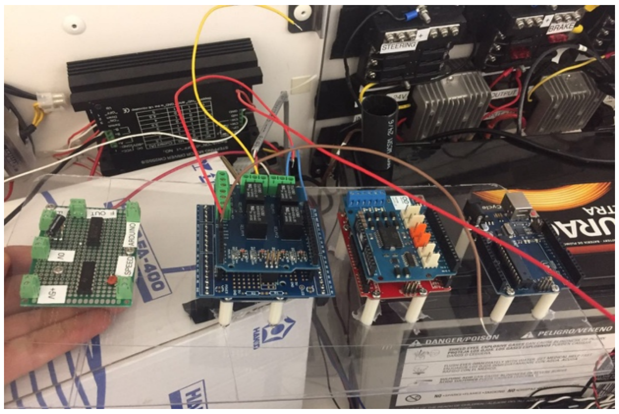
Vehicle drive-by-wire power from a separate 12V battery, and using two voltage boosters generate three level power: 5V, 12V and 24V. 5V is use for 4 Arduinos, 12V for safety light, Lidar, Radar, and 24V for steering motor.

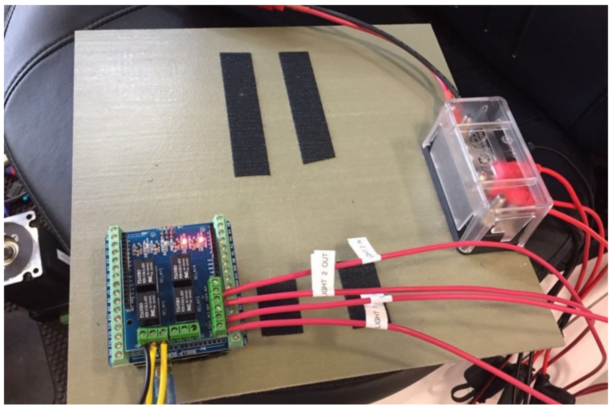




4.5 Drive-by-wire control design

We are using 4 Arduino control all the drive-by-wire system. One is for Steering motor, one is for vehicle motor speed measuring, one is for front brake control and one is for Forward/Backward control. All Arduinos running ROS topic, and control by the ROS Computer.

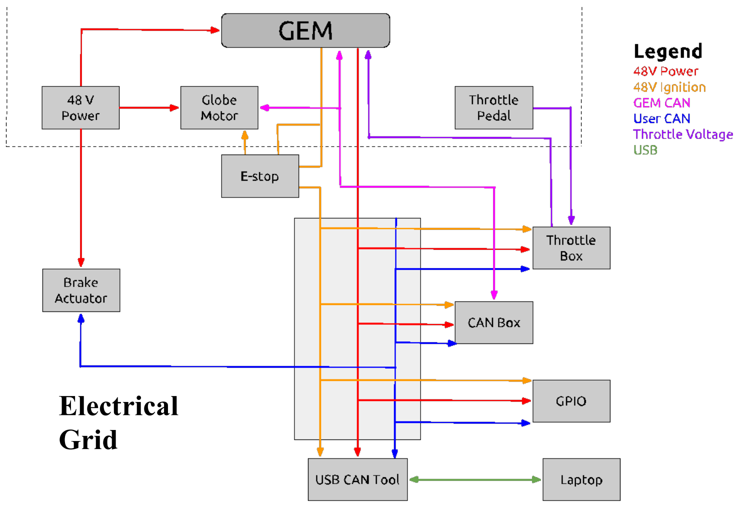


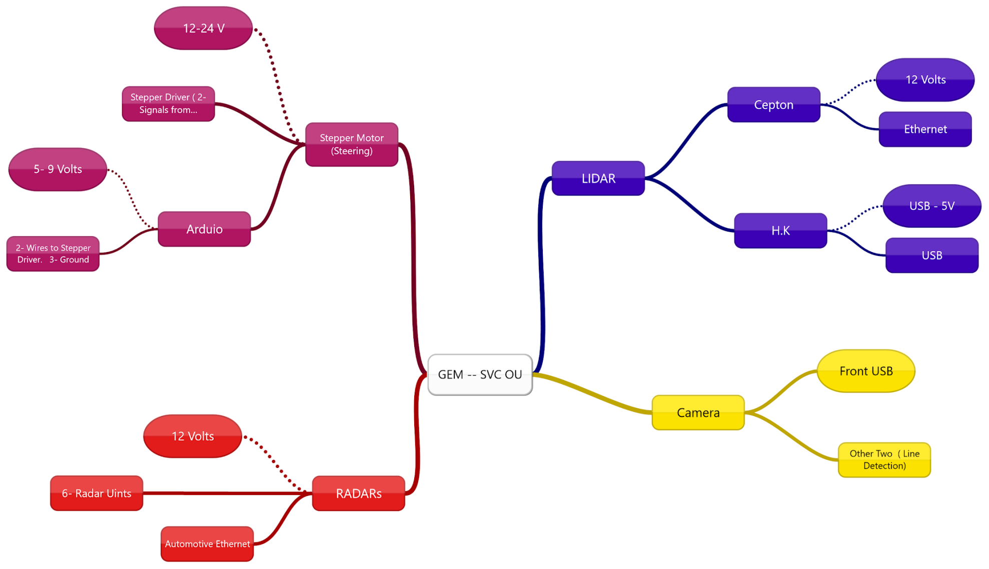


5 Description of electronic and power design

5.1 Overview

Electrical Grid. One of the essential tasks carried out by the By-Wire Team was to design the power grid for the new components to be installed on the GEM. The wiring for the drive by wire system of the vehicle is split into two main parts as shown in the diagram on the right. The components inside the dotted line are parts native to the GEM vehicle. The components outside this line are the additions made in the design process to bring the vehicle up to functional standards. The vehicle is powered from four 12 volt batteries in series (48 VDC).





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

Auxiliary Power. To distribute power to at least four laptops and monitor, we have a 12-volt 100Ah battery that connects to an 800 watt DC to AC inverter. See photo. This auxiliary power system is meant to extend the power consumption from the vehicle computer components, and thus increase overall autonomous operation time for the GEM.

5.3 Electronics suite description including CPU and sensors system integration/feedback concepts

Nightmare’s software is implemented on a Linux platforms runs Robot Operating System (ROS). The ROS community has created many learning and distributing packages for its common functions, like mapping and planning packages, drivers for sensors, such as LIDAR, Cameras, and GPS Units. The system configuration below depicts the integration of sensors, feedback and control.

5.4 Safety devices and their integration into your system

Hardware E-Stops. There is a manual red E-stop button in the driver & passenger cabin of the GEM2 vehicle at this time. The teams are currently working to install i) an external manual E-stop accessible at the rear of the vehicle, and ii) a wireless remote control E-stop. The hardware E-Stop function disconnects power from the motors and applies the brake.

Software E-Stop. To protect against a variety of failure conditions, the drive control system is programmed to automatically turn off the motors if it fails to receive commands from the computer or joystick after 200 ms. The software E-stop sets the speed of the motor to zero and applies the brake.

Flashing Signals. The teams are also currently working to incorporate visible lights to signal the state of computer control of the vehicle. E.g., flashing red lights to let everyone know that the vehicle is operating in autonomous mode. Green light means all systems are in ready mode, while red means no go, etc.

Audio Signal. Additionally, the teams will add audible sounds to alert everyone nearby when the vehicle is about to go into autonomous mode of operation. E.g., play a jingle or short burst of sound.

Smooth Acceleration. For safer driving action, both the throttle and steering commands are programmed to have a smooth limiting control factors to ensure smooth driving and prevent aggressive maneuver.

6 Description of software strategy and mapping techniques

6.1 Overview

The Navigation Stack serves to drive a mobile base from one location to another while safely avoiding obstacles. Often, the robot is tasked to move to a goal location using a pre-existing tool such as rviz in conjunction with a map. A 2D navigation stack takes in information from odometry, sensor streams, and a goal pose and outputs safe velocity commands that are sent to a mobile base. The navigation stack assumes that the robot is configured in a particular manner in order to run

6.2 LiDAR and RADAR Obstacle Detection / Ranging

LiDAR/RADAR Team consists of John Brooks, Arjun Musham and Saif Salih

The vehicle is outfitted with 6 Continental ARS-430 RADAR units. These RADARs communicate over Automotive Ethernet to an Intrepid Control Systems RAD-Jupiter media converter + switch which outputs standard Ethernet to the laptop computer. The RADAR detections are broadcast onto to ROS system.

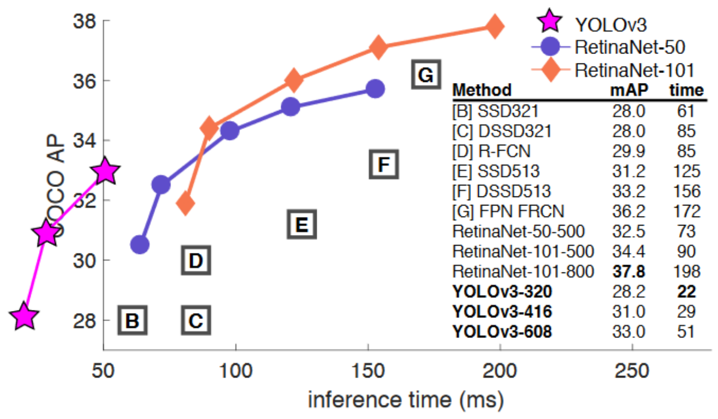
The RADAR outputs detections that are close to raw data. This makes for very noisy data points which are hard to process. Because of this the RADAR is not used for object detection, but only for object ranging. When an object is detected by the vision system the distance to that object is found by determining the angular window of the object and finding the closest RADAR data point.

The LiDAR system is still being developed. The team has been supplied LiDARs from Cepton Technologies Inc. These are solid state LiDARs with a 60° field of view that will be mounted on the roof of the vehicle. What is currently being worked on is to map driveable space around the vehicle and perform object detection with these LiDAR units.

6.3 Object Detection and Classification

Object Detection Team consists of John Brooks and Bing Liu.

For object detection we chose a convolutional neural network (CNN) design. CNNs represent the current state of the art in object detection and classification. The model used is called YOLOv3 [1], which is an acronym for You Only Look Once. This is a feed forward network which performs object detection and classification simultaneously in real time.



Method	mAP	time
[B] SSD321	28.0	61
[C] DSSD321	28.0	85
[D] R-FCN	29.9	85
[E] SSD513	31.2	125
[F] DSSD513	33.2	156
[G] FPN FRCN	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	**37.8**	198
YOLOv3-320	28.2	**22**
YOLOv3-416	31.0	29
YOLOv3-608	33.0	51

Figure 1. YOLO Accuracy vs. Inference Time

In order to use YOLO the CNN must be trained by means of labeled data. The challenge with using a deep learning approach is that a large amount of training data is required for training the feature extraction layers of the network. Ideally, this means tens of thousands of hand labelled images which is out of the scope of what our team can reasonably accomplish.

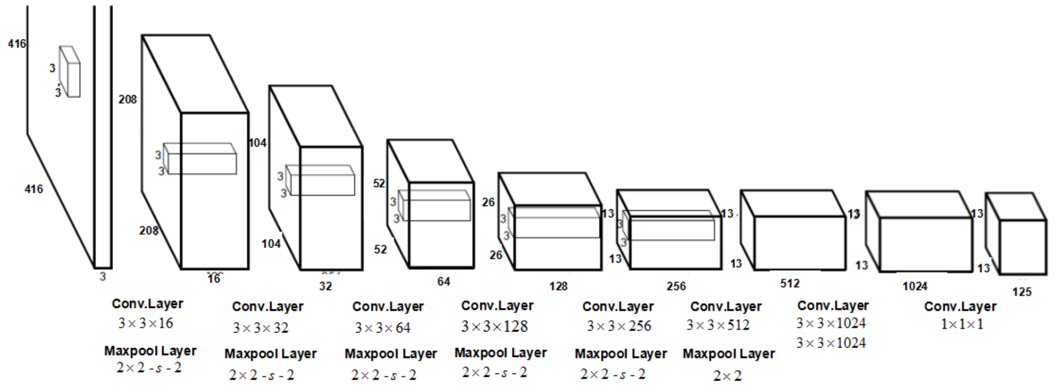


Figure 2: YOLO CNN Architecture

To work around this problem, transfer learning is used. The model consists of feature extraction convolutional layers in the beginning and fully connected layers at the end. To train the network on the IGVC signs, weights which are trained on much larger datasets (VOC and COCO) are used for the convolutional layers and the fully connected layers are randomized. When our own dataset is fed into the network we are primarily training the fully connected layers on the features of our specific objects. The result is viability on a relatively small dataset and a small amount of training.

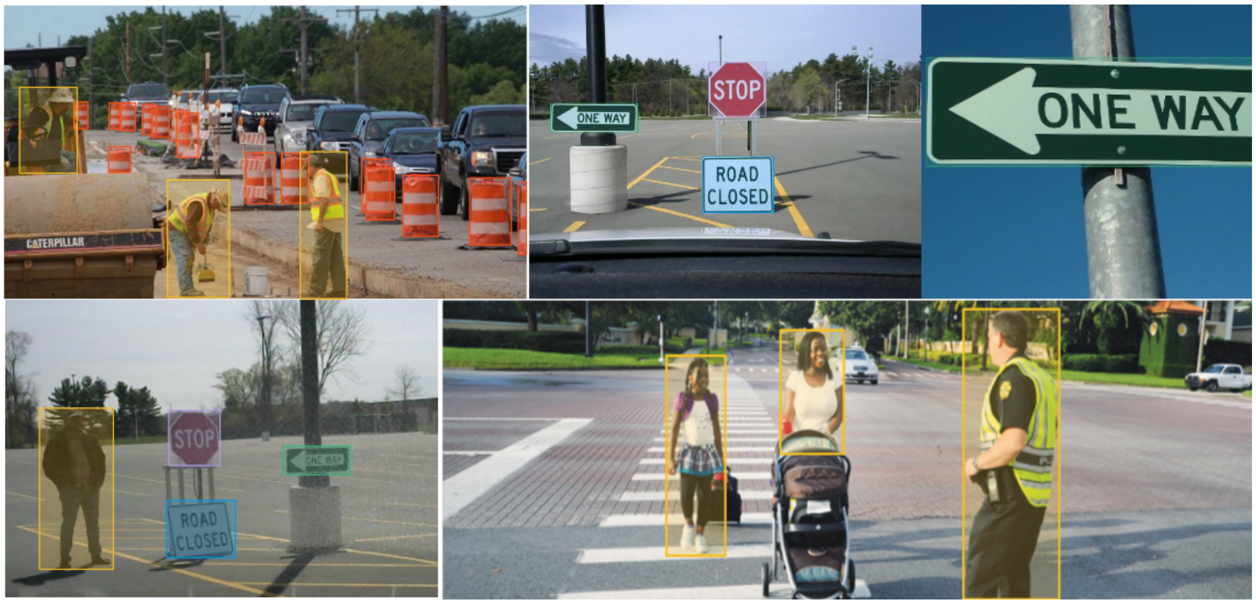


Figure 4: Labeled Images

The image dataset our team created consists of over 3000 labeled objects. These images were sourced from Google Images, the LISA traffic sign dataset, and our own collected data. As our testing continues we expect to continue growing our collected images based on any performance issues we encounter during testing.

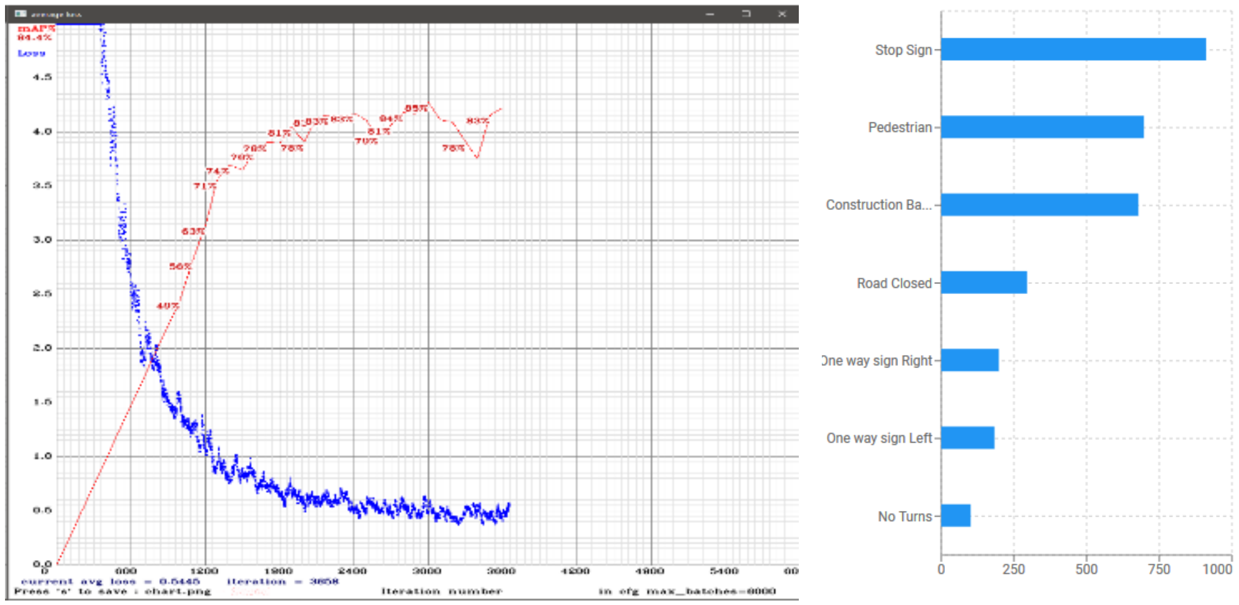


Figure 5: YOLO Training in Progress and dataset

Once the dataset was created, the network was trained on an NVIDIA GTX 1080 GPU for several hours. In order to evaluate the model's performance and ensure that it's not overfitting the data, a hold out set of 10% of our images was set aside for validation. In figure 5, the error of the training set is shown in blue and the accuracy (mAP) on the validation set in red. Once the performance on the validation set levels off or begins to fall, the network training is considered complete.

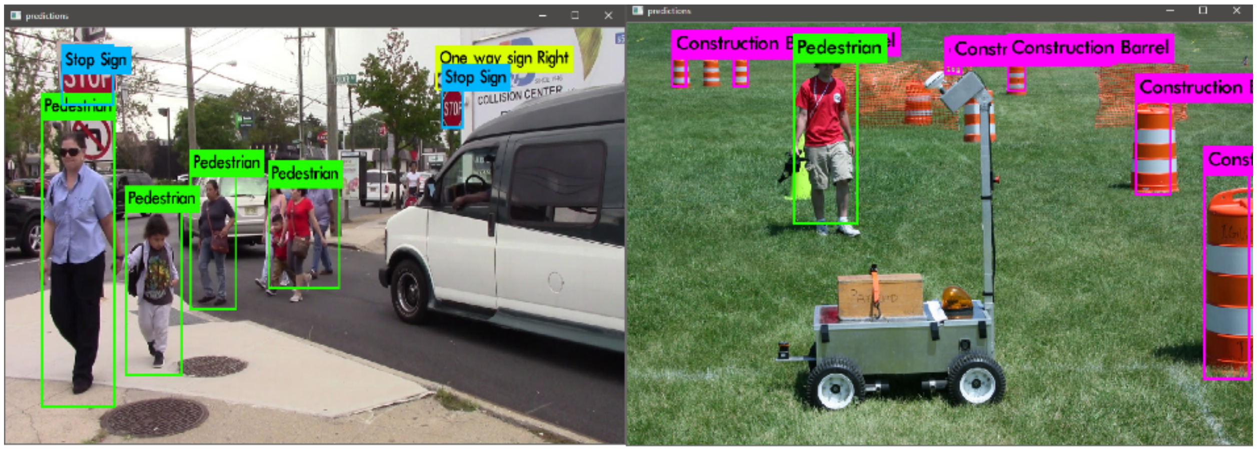


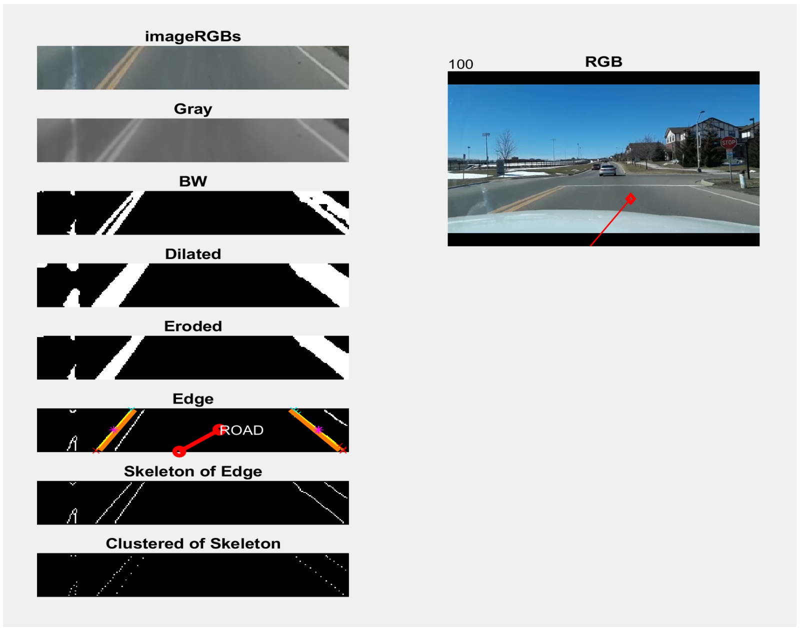
Figure 6: Results

The end result is that the trained model can detect and classify the objects of the competition quickly and with few errors. The object detections from the forward facing camera of the vehicle are converted into angular windows these windows and classifications are published to the ROS system so that they can be combined with more accurate distance measuring sensors of the vehicle such as RADAR and LiDAR sensors.

6.4 Lane Detection Algorithm

Lane Detection Team consists of Narendra Kintali and Ana Farhat.

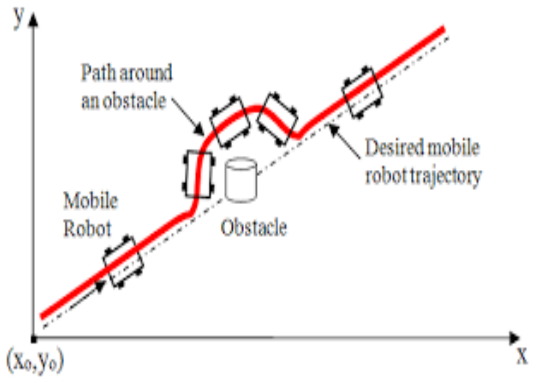
The team uses Matlab Image Processing Toolbox & Computer Vision Toolbox for detecting lanes from the camera images. Different techniques including convolution neural network, Hough transforms, were explored. At this time, we find that the classic Hough transform method provides the most viable approach in that we can manipulate and use its parameters to yield the results we look for. The figure below depicts successful detection of the lanes from a RGB video of a campus road. The team is working to keep improving reliability and robustness of the detection scheme.



6.5 Software strategy and path planning

The software strategy and path planning will require involvement of different teams including Lane Detection Team, Computer Vision Team Lidar Detection Team, GPS Team. Path Planning & Control Team, Simulation Team.

Concepts, such as depicted on the right, have been initially evaluated by simulation, which provides a basis for establishing requirements from each functionality team. This was a key to coordinating specifications and communications of the function modules.



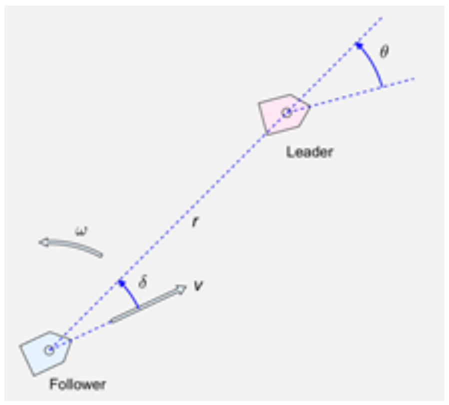
6.6 Smooth Trajectory Planner and Control Software Strategy.

The control and path planning part of the software strategy includes using different aspects of Computer Vision, Path Planning, Simulation, and Control among others. This makes use of a Smooth Trajectory Planner scheme is defined below in a clearer format, starting with definition of key variables, kinematics model, Lyapunov function, desired steering angle and desired angular speed. Further controls and dynamics are additionally introduced in order to make this a feasible system to apply real- time on a vehicle. Note that a sensor suite of camera, GPS, IMU, lidar and/or radar, can measure the key variables.

6.6.1 System Variables

The figure below illustrates the basic variables needed for the formulation of the smooth path planner (SPP):

r = distance from prime vehicle to target vehicle
θ = angle of target direction w.r.t. the joining line
δ = angle of prime direction w.r.t. the joining line
ω = angular speed of prime vehicle
v = forward speed of prime vehicle



δ represents the steering angle for the prime vehicle, and r is the separation distance between the vehicle and the target. It is assumed that the target is fixed or not moving as is generally the case for the Self-Drive competition. These variables can be determined from a combination of measurements from camera, GPS, IMU, lidar and/or radar.

Figure is top view diagram of leader-follower robotic vehicles.

6.6.2 Kinematic Model

Using kinematics relationship, it is shown that

$$\begin{bmatrix}\dot r\\ \dot\theta\end{bmatrix} = \begin{bmatrix}-v\,\cos\delta\\ \frac{v}{r}\,\sin\delta\end{bmatrix} \tag{1}$$

Since the target is stationary, it is also shown that

$$\dot\delta = \frac{v}{r}\,\sin\delta + \omega = \dot\theta + \omega \tag{2}$$

As in the illustration, equations 1,2 and Figure 1 represent a robotic vehicle as being driven to a parking space represented by the target if we are able to drive r → 0, θ→0 , δ → 0.

6.6.3 Lyapunov Stability Criterion (LSC)

For the purpose of stability analysis, consider driving $r \to 0$, $\theta \to 0$, $\delta \to 0$. To apply the LSC, the positive definite function is introduced (3) as a Lyapunov candidate

$$V = \frac{1}{2}r^2 + \frac{1}{2}\theta^2 > 0 \tag{3}$$

where r is a positive separation distance. The LSC states that if it is ensured that the time derivative of V is negative

$$\dot{V} = r\dot{r} + \theta\dot{\theta} < 0 \tag{4}$$

then, $r \to 0$ & $\theta \to 0$. That is, a speed v and ω should be obtained that produce a steering angle that yields a distance and an orientation, such that so is satisfied.

6.6.4 Desire Vehicle Orientation

A method to satisfy the LSC is to set the desired orientation as shown in (5), where k_1 is greater than or equal to 0 is a positive value to be assigned

$$\delta_{des} = \tan^{-1}(-k_1\theta) \tag{5}$$

Since $\theta \in (-\pi, \pi] \subset \Re$, (5) leads to the properties given by (6)

$$\begin{gathered} -\frac{\pi}{2} < \delta_{des} \le \frac{\pi}{2}, \\ \cos\delta_{des} \ge 0 \\ sign(\sin\delta_{des}) = sign(\delta_{des}) \end{gathered} \tag{6}$$

If $\delta \to \delta_{des}$ is driven , then (1) becomes

$$\begin{bmatrix} \dot{r} \\ \dot{\theta} \end{bmatrix} \to \begin{bmatrix} -v\ \cos\delta_{des} \\ \frac{v}{r}\ \sin\delta_{des} \end{bmatrix} \tag{7}$$

Substituting (7) into (4) and applying (6) lead to

$$\dot{V} \to \ -r\,v\ \cos\delta_{des} r\,\dot{r}\ +\ \theta\,\frac{v}{r}\ \sin\delta_{des} < 0 \tag{8}$$

According to the LSC, the Lyapunov function qualified by equations (3) & (8) implies that (1) will be stable if $\to$ $_{des}$ is driven.

6.6.5 Steering Command via Back-Stepping Control Scheme

The steering error is defined as the

$$e = \delta_{des} - \delta = \tan^{-1}(-k_1\theta) - \delta \tag{9}$$

The time derivative is given by

$$\begin{aligned} \dot{e} &= \dot{\delta}_{des} - \dot{\delta} \\ &= \frac{-k_1}{1+(k_1\theta)^2}\dot{\theta} - \dot{\theta} - \omega \\ &= -\left(1 + \frac{k_1}{1+(k_1\theta)^2}\right)\frac{v}{r}\sin\delta - \omega \end{aligned} \tag{10}$$

By inspection, a back-stepping control scheme is to drive the angular speed ω so that it follows

$$\omega_{des} = \left(1 + \frac{k_1}{1+(k_1\theta)^2}\right)\frac{v}{r}\sin\delta + k_2\frac{v}{r}e \tag{11}$$

where $k_2 > 0$ is a controller gain to be chosen. As $\omega \to \omega_{des}$, the error dynamics (10)

$$\dot{e} \;\to\; -k_2 \frac{v}{r} e \qquad (12)$$

which is exponentially stable since v, r, and k_2 are positive values. Equations 5 & 11 form the desired steering command for the Lyapunov-based smooth trajectory-planning (SPP) scheme.

6.6.6 Steering Actuation and Control

A steering mechanism is needed to produce the angular speed , which would be controlled by an actuator input The dynamics of the steering can be described by

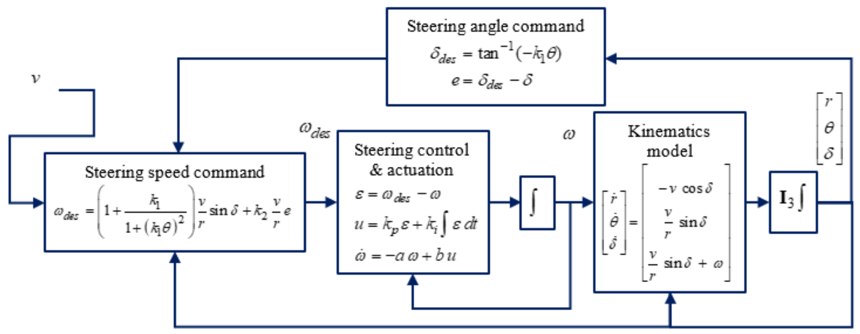
$$\dot{\omega} = -a\,\omega + b\,u \qquad (13)$$

where a & b are system parameters for the actuation. To drive ω to ω_{des}, a proportional + integral action controller can be implemented given by (14), where k_p & k_i are gains to be determined.

$$\begin{aligned} \varepsilon &= \omega_{des} - \omega \\ u &= k_p \varepsilon + k_i \textstyle\int \varepsilon\, dt \end{aligned} \qquad (14)$$

6.6.7 Overall Scheme at a Glance

Figure 2. shows the overall smooth path planning (SPP) scheme consisting of all the factors described above.



Steering angle command
$\delta_{des} = \tan^{-1}(-k_1\theta)$
$e = \delta_{des} - \delta$
v
ω_{des}
ω
Steering speed command
$\omega_{des} = \left(1 + \frac{k_1}{1+(k_1\theta)^2}\right)\frac{v}{r}\sin\delta + k_2\frac{v}{r}e$
Steering control & actuation
$\varepsilon = \omega_{des} - \omega$
$u = k_p\varepsilon + k_i\int \varepsilon\, dt$
$\dot{\omega} = -a\,\omega + b\,u$
$\int$
Kinematics model
$\begin{bmatrix}\dot{r}\\ \dot{\theta}\\ \dot{\delta}\end{bmatrix} = \begin{bmatrix}-v\cos\delta\\ \frac{v}{r}\sin\delta\\ \frac{v}{r}\sin\delta + \omega\end{bmatrix}$
$I_3\int$
$\begin{bmatrix}r\\ \theta\\ \delta\end{bmatrix}$

Figure 2. Overall smooth path planning (SPP) scheme

k_1, k_2, k_p, & k_i are design parameters whose values are usually determined with the help of simulation. k_1 determines how much reaction should be given to θ, and can range from 0 to 10. k_2 determines how fast δ should approach δ_{des} and can ranges from 1 to 5. The choice of k_p, & k_i depends on the actuators and are chosen to control the transient behavior the steering speed ω converging to.

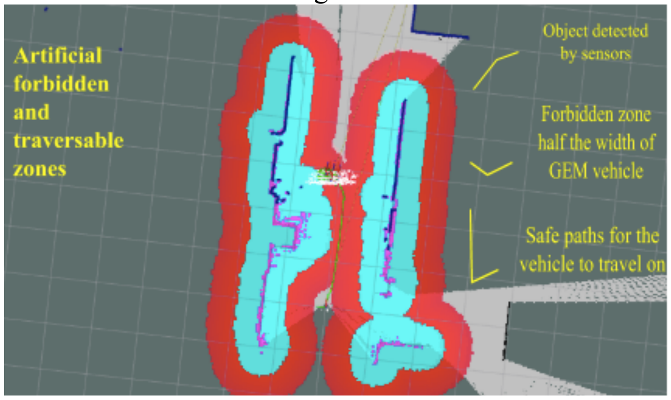
The next aspect includes the dynamics and control behaviors of the vehicle motion in the SPP analysis. This will take actuation drives, control schemes and processing delays into consideration so a more realistic expectation is obtained. This is being applied to the Nightmare Gem2 vehicle. (SPP) was derived using detail analysis involving kinematics model, Lyapunov stability criterion and back-stepping. Also, this is a relatively simple interpretation and readily adapted or switched to work with many car maneuvers. Additionally, this method shows potential for practical applications and sets the foundation and validates ideas and scenarios for eventual actual realization of the schemes.

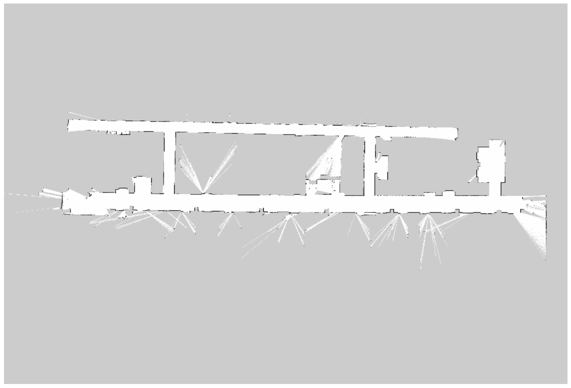
6.7 Map generation

The information from the vehicle sensor suite (lidar, stereo camera, Kinect, webcams, IMU, odometry) can be integrated using sensor fusion mapping algorithm such as Kalman filter-based SLAM (simultaneous landmark and mapping) in the ROS environment. The figure below is a map of the hallway of the third floor of the OU SECS EC building, produced by the **Lidar & Kinect team**. We are working and testing on mapping of outdoor scenes.

6.8 Goal selection and path generation

The Team uses ROS-based Navigation Stack for guiding the vehicle to accomplish its mission. The path-planning & control scheme to be employed is first simulated in the ROS-based Gazebo & RViz, as described in Section 7 on Simulation. The strategy can be explained with the help of the mapping boundaries shown in the figure below.


Artificial forbidden and traversable zones
Object detected by sensors
Forbidden zone half the width of GEM vehicle
Safe paths for the vehicle to travel on

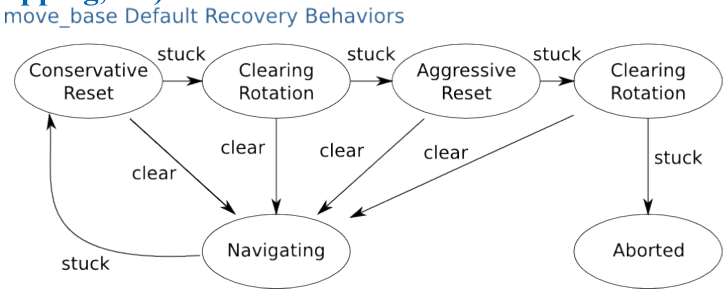


6.9 Additional creative concepts

We have also established a Parking Team to work on parallel parking, forward and reverse parking using ultrasonic only sensors, camera only sensors, and a combination of sensors. They are to use new smooth driving trajectory to achieve the objective. The **Parking Team** consists of Ana Farhat, Bharvana Narke & others.

7 Description of failure modes, failure points and resolutions

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



At this time, we have looked into techniques for handling certain vehicle failure in software and mapping. However we have not implement and tested them.

An example of the technique is the so-called Move-base Default Recover Behavior, depicted below.

Running the `move_base` node on a robot that is properly results in a robot that will attempt to achieve a goal pose with its base to within a user-specified tolerance. In the absence of dynamic obstacles, the move_base node will eventually get within this tolerance of its goal or signal failure to the user. The `move_base` node may optionally perform recovery behaviors when the robot perceives itself as stuck. By default, the move_base node will take the following actions to attempt to clear out space. First, obstacles outside of a user-specified region will be cleared from the robot's map. Next, if possible, the robot will perform an in-place rotation to clear out space.

If this too fails, the robot will more aggressively clear its map, removing all obstacles outside of the rectangular region in which it can rotate in place. This will be followed by another in-place rotation. If all this fails, the robot will consider its goal infeasible and notify the user that it has aborted.

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

Similarly, we investigated how automotive industry handle vehicle failure points, but have not implemented to test the methods. In Automotive each electronic control unit (ECU) is required to perform a list of tasks; some of these tasks are performed in a regulated matter and follows a set of standards, such as; On-board diagnostics (OBD). OBD is a self-diagnostic and reporting capability for these ECUs, it was intended to indicate any issues or problems that an ECU faces. The OBD procedure mainly divided into three sections; hardware, software, and electrical problems. Each section has their set of digital codes that are called digital diagnostic code (DTC), they provide a remedy and more information about the problem and its steps of troubleshoots. For the ECU to report any DTC, it needs to undergo a set of OBD self-tests, such as; making sure its sensors (IOs) are connected properly, also making sure it provided voltage is within accepted limits,

7.3 All failure prevention strategy

We broach the topics but did not dwell further its implementation due to lack of time.

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

We conducted testing of components and subsystems as thorough and best as we can before using and integrating them onto the vehicle. These includes by-wire subsystems, computer vision systems, lidar and Kinect systems, mapping system, path planning & control simulation, so on.

7.5 Vehicle safety design concepts

The topic of vehicle safety is always broached during meetings, especially the weekly Saturday team assembly meetings. This include safety awareness while operating the vehicle, safety while performing tests of subsystems, and others.

8 Simulations employed

8.1 Simulations in virtual environment

The team uses Gazebo and RViz in ROS environment to simulate and visualize path planning and control of a GEM vehicle in various driving scenarios, involving avoiding obstacles, lane following and going toward destination. Gazebo is a well-designed simulator that offers the ability to accurately and efficiently simulate robots in complex environments. It represents the actual world for the robot to localize itself and be able to navigate between the start and the goal points. Rviz (a 3d visualization environment) was also used to let us view what the robot is seeing, thinking and doing. It is capable to understand sensor messages like Laser scans and Point clouds and images from the camera. The ROS navigation stack uses these messages to show its current path and obstacle data.

8.2 Theoretical concepts in simulations

The concepts are still being investigated by the teams at this time.

9 Performance Testing to Date

9.1 Component testing, system and subsystem testing, etc.

At this time, the teams have completed the following tests:

- By-Wire Team: Tested steering & throttle and most parts of brakes. Working on the F-N-R shift mode.
- Lane Detection Team: Tested the Hough lane detection algorithm, working on live demo.
- Sign Detection Teams: Tested with all signs. It sometimes mixes up one way left and one way right.
- Path Planning & Control Team: Tested on simulation. Working to test with live sensors & drives.
- Mechanical & Electrical Team: Did mounting mechanicals & electrical.