Technologies for Safe & Efficient Transportation

THE NATIONAL USDOT UNIVERSITY TRANSPORTATION CENTER FOR SAFETY

Carnegie Mellon University | UNIVERSITY of PENNSYLVANIA

SmartShuttle: Model Based Design and Evaluation of Automated On-Demand Shuttles for Solving the First-Mile and Last-Mile Problem in a Smart City

FINAL RESEARCH REPORT

Contract No. DTRT-13-GUTC-26

Introduction

The final project report for the SmartShuttle sub-project of the Ohio State University is presented in this report. This has been a two year project where the unified, scalable and replicable automated driving architecture introduced by the Automated Driving Lab of the Ohio State University has been further developed, replicated in different vehicles and scaled between different vehicle sizes. A limited scale demonstration was also conducted during the first year of the project. The architecture used was further developed in the second project year including parameter space based low level controller design, perception methods and data collection. Perception sensor and other relevant vehicle data were collected in the second project year. Our approach changed to using soft AVs in a hardware-in-the-loop simulation environment for proof-of-concept testing. Our second year work also had a change of localization from GPS and lidar based SLAM to GPS and map matching using a previously constructed lidar map in a geo-fenced area. An example lidar map was also created. Perception sensor and other collected data and an example lidar map are shared as datasets as further outcomes of the project.

Brief Problem Description

A major component of mobility in a smart city is the use of fully electric driverless vehicles that are used for solving the first-mile and last-mile problem, for reducing traffic congestion in downtown areas and for improving safety and helping in the overall reduction of mobility related undesired emissions. Currently available Smart Shuttle solutions have serious interoperability problems due to the low volumes of production and due to the fact that they are developed and manufactured by small startup companies in contrast to OEMs with their series production capability and large R&D departments. Current Smart Shuttle sensing and automation architectures are, therefore, also not easily scalable and replicable. Success of Smart Shuttles in Smart Cities requires an interoperable, scalable and replicable approach which is what this project addresses through model based design techniques.

Project Approach, Methodology and Results

Our model based design approach uses a unified software, hardware, control and decision making architecture for low speed smart shuttles that is scalable and replicable. Robust parameter space based design is used for easily scalable low level control systems development. Our model based design approach uses model-in-the-loop and hardware-in-the-loop simulations before road testing. This method was demonstrated in proof-of-concept testing/deployment in non-public areas including parking lots.

A unified scalable and replicable architecture and the hardware-in-the-loop simulator for automated driving were prepared in this project. Extensive model-in-the-loop and hardware-in-the-loop simulations were used for testing the automated driving system in the lab setting first. Testing included communication with other vehicles and

instrumented traffic lights using a DSRC on-board unit (OBU) modem and a DSRC road-side unit (RSU) modem that were added to our connected and autonomous driving hardware-in-the-loop simulator. Four different platforms in two different vehicle size categories were used in experimental work for replicability and scalability. These vehicles were used in generating the experimental results some of which are given in this report. Please refer to the enclosed papers for more detailed results.

Our work on unifying the structure with scalability and replicability is to create a standard base for hardware structure along with a library to be used by developers for faster and easier automation of vehicles. The hardware structure includes different types of sensors to achieve enough coverage, resolution and also robustness to external disturbances. Data from these sensors is processed by a high processing power computer to create meaningful information, which is used by a low-level controller, i.e. a dSpace MicroAutobox in our vehicles, to drive the vehicle autonomously by interfacing with actuators and sending necessary commands. The unified architecture is shown in Figure 1.

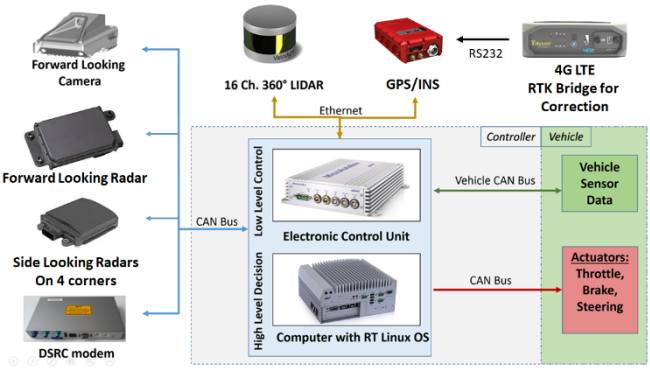


Figure 1. Unified architecture.

Replicability and Scalability

Using this unified architecture, two different sized vehicles were automated. These are a sedan, representative of passenger vehicles and a small neighborhood electric vehicle, representative of small on-demand electric shuttles. Perception sensors such as Lidar, Camera, Radars were implemented as well as a GPS Sensor with RTK correction for localization. The dSpace MicroAutobox unit is used for low level controls and an in-vehicle Linux PC with a GPU is used for sensor data computation. Moreover, DSRC (dedicated short-range communications) radios are added to have the capability of communicating with other vehicles, pedestrians, bicyclists and infrastructure. Pictures of the vehicles and the implemented hardware as well as the evaluation of the scalability approach is presented in our papers and is shown graphically in Figure 2 here. On the top are the two vehicles we automated first. These are a 2015 Ford Fusion and a Dash EV neighborhood electric vehicle. Both vehicles were made drive-by-wire first and the sensor, hardware and computing architecture of the Ford Fusion sedan was scaled down to the Dash EV vehicle. The low level longitudinal and lateral controllers developed for the Ford Fusion using the parameter space design approach were also scaled down and used in the Dash EV after some re-tuning due to changes in vehicle parameters. The architecture/results were, then, replicated as we migrated from the

2015 Ford Fusion to the 2017 Ford Fusion and from the white Dash EV to the red Dash EV both in the bottom of Figure 2. The drive-by-wire system for the Ford Fusion vehicles uses a commercially available system that uses CAN bus commands to control the actuators. This drive-by-wire was moved from the 2015 Ford Fusion to the 2017 Ford Fusion with a software upgrade that enabled shift-by-wire on top of the previously available throttle, brake and steer-by-wire. The drive-by-wire systems of both Ford Fusion vehicles were based on the original OEM actuators such that when the drive-by-wire system was turned on, we had the original vehicle which we can legally drive on public highways. The drive-by-wire system of the Dash EV vehicle was prepared in-house and is illustrated in Figure 3 for our first Dash EV vehicle. The throttle in this electric vehicle uses a potentiometer which was changed according to the autonomous driving controller using an additional circuit that changes the throttle potentiometer reading. A linear actuator was used to pull/push the brake pedal for brake-by-wire and a smartmotor and an extra half steering linkage was added to provide driver independent steering as needed for steer-by-wire operation. An extra electronic circuit with a manual/autonomous conversion switch was added for switching between normal vehicle operation and drive-by-wire operation. As the two Dash EV vehicle dimensions and characteristics were very similar, the drive-by-wire system was easily transferred from the first vehicle (white one on top in Figure 2) to the new one (red one in bottom of Figure 2). A remote e-stop kill switch with an RF link is also being added to the new Dash EV vehicle. Our new Dash EV vehicle has a license plate like or Ford Fusion vehicle and can be driven on public roads in the manual mode.

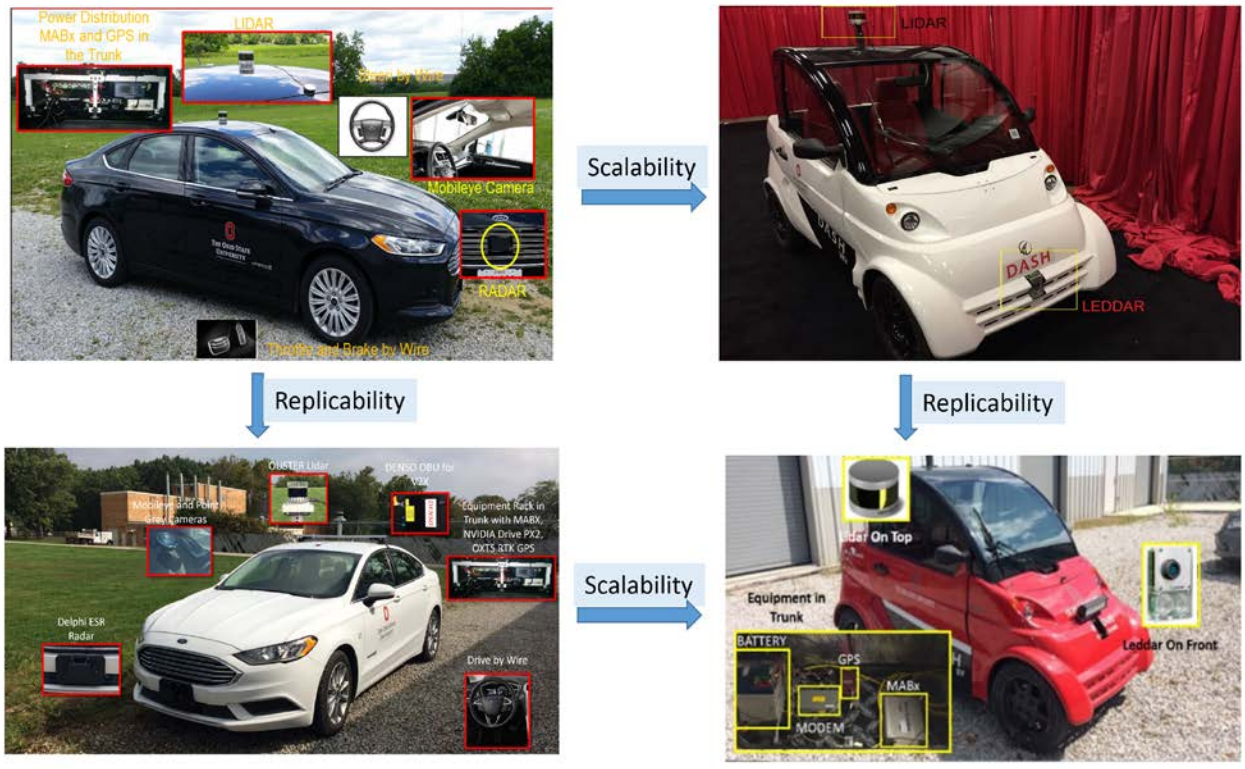


Figure 2. Scalability and replicability.

Figure 3. Dash EV drive-by-wire system.

Along with the unified architecture, a unified Simulink library was also created. This library shown in Figure 4 consists of different types of blocks, including low-level control blocks for steering, throttle, brake, shift; sensor blocks for receiving data from the sensors in order to have environment perception and localization; and finally control and decision-making blocks for low and high level control of the autonomous vehicle. We are currently extending this library for use with NVIDIA Drive PX 2 GPUs. It is slightly modified for CarSim soft sensors and then used in the hardware-in-the-loop (HIL) simulations presented later in the report.

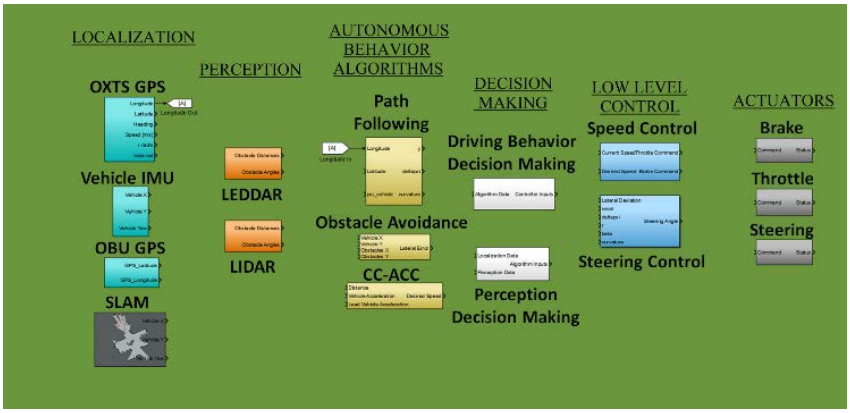


Figure 4. Simulink vehicle automation library.

Controls - Low Level Longitudinal Control

Parametric modeling is the first step necessary in designing controllers. For the longitudinal speed control of the Ford Fusion sedan, we performed system identification on experimental data of different throttle step input and obtained a simple first order model. A typical example is presented here for a 15% throttle step input. The experimental data is recorded from the vehicle by giving a step input to the throttle and then recording the vehicle velocity [1]. A first order transfer function was then fitted to the vehicle velocity data. The transfer function obtained through this method for 15% throttle input is given in Equation 1.

$$G(s) = \frac{0.1515}{s + 0.07496} \tag{1}$$

The curve fit comparison of experimental, Simulink model and constructed Carsim model are shown in Figure 5 for 15% step throttle input. The procedure is repeated for different throttle openings to obtain a family of plants which are then used in a parameter space design to prepare a scheduled PID longitudinal speed controller.

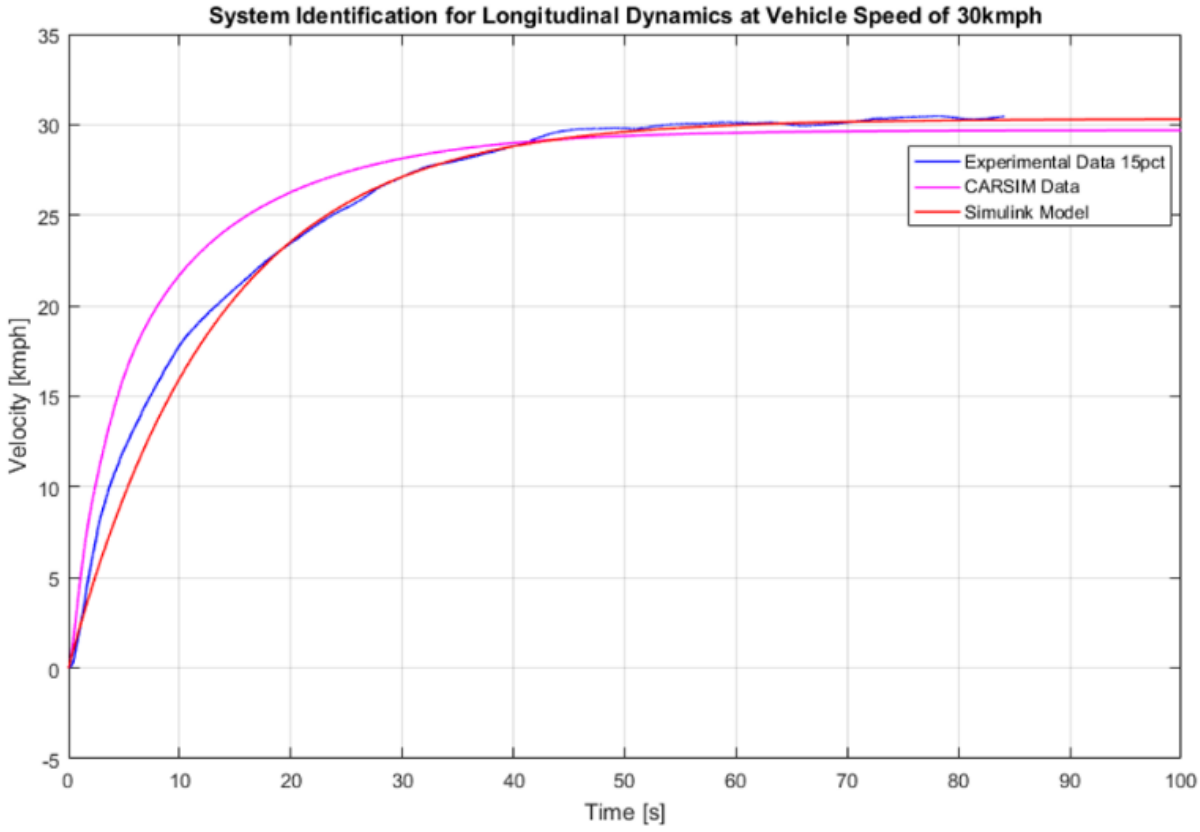


Figure 5. Velocity graph for experimental result, Carsim model and Simulink model for 15% throttle.

It should be noted that the longitudinal dynamics of the vehicle is highly non-linear and changes with throttle position. While a simple curve fit at one throttle opening is shown and used for initial step, a more comprehensive multi-model determination based on the experimental data we collected is in progress and will be used in our future work.

Controls - Low Level Lateral Control

Lateral controller design also requires the development of a model of the lateral dynamics of the vehicle. Both Simulink and Carsim models were developed for both the sedan and neighborhood electric vehicles. The model building process involved the use of testing machines (for the sedan) as shown in Figure 6 for our 2015 Ford Fusion sedan and identification of parameters using standard tests (for the sedan and the neighborhood electric vehicle). The same models and low level controllers designed based on them as the architecture and controllers were replicated and transferred from our 2015 to 2017 sedan. A similar replication of model and low level controllers was used without problems as we migrated from one Dash EV vehicle to the next one.

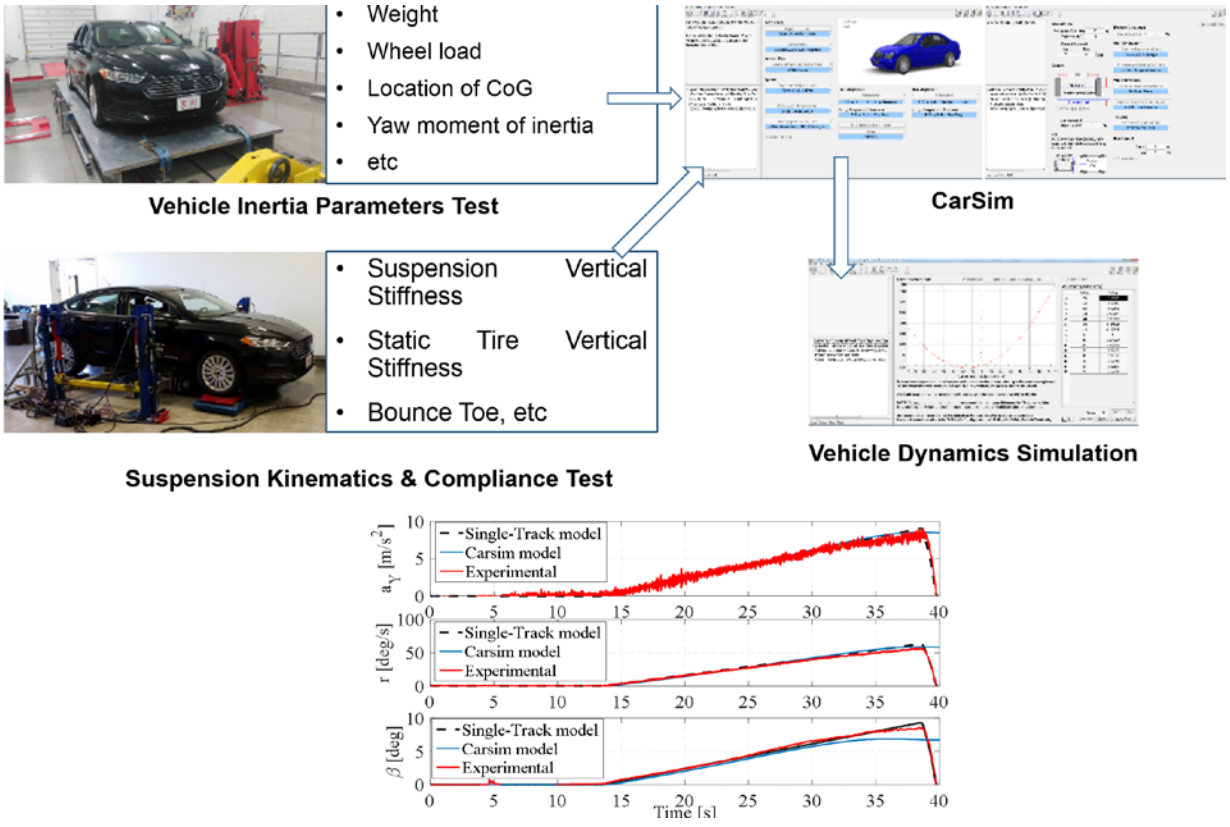


Figure 6. Measurement of vehicle model parameters of 2015 Ford Fusion.

Measured, identified and estimated parameters for the sedan and small electric vehicles are listed in Table 1.

Table 1. Vehicle lateral model parameters.

	Ford Fusion	Dash
m	1977.6 kg	350 kg
J	3728 kgm^2	350 kgm^2
lf	1.3008 m	1.06 m
lr	1.54527 m	0.96 m
R	0.3225 m	0.24 m
Cf	1.9e5 N/rad	1.8917e4 N/rad
Cr	5e5 N/rad	1.8917e4 N/rad
k_p	0.15	0.9272
k_d	0.1	0.0801

Satisfactory lateral control requires high accuracy of path tracking, and robustness to system parameter variations like vehicle load, speed and tire cornering stiffness. Figure 7 shows variation regions of these parameters for our two vehicle platforms, the low-speed shuttle Dash and the Ford Fusion sedan. The design procedure for lateral control we used satisfies D-stability and robust performance and is easily replicable and scalable to other vehicle platforms.

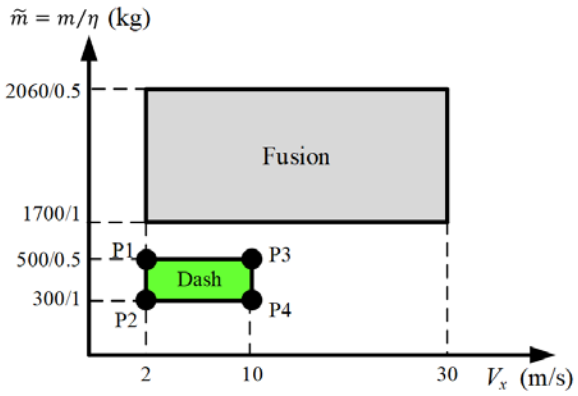


Figure 7. Uncertainty region of vehicle mass m, longitudinal speed V_x and parameter η for Dash and Fusion experiment platforms.

The system roots are confined in the D-stability region (Figure 8) in the complex plane to satisfy requirements like settling time, damping ratio and bandwidth. The mixed sensitivity criterion is also raised to ensure robust performance in frequency domain. The boundaries of the D-stability region, along with the points satisfying the mixed sensitivity critical criterion, are mapped to the parameter space of control parameters, k_p and k_d, for the robust proportional-derivative (PD) controller. Figure 9 shows an example of the control parameter space at one operating condition. The PD control parameters are chosen from the selectable region after overlapping parameter space for all operating conditions. A model regulator is added in combination with the robust PD controller to further reduce tracking error (Figure 10). The model regulator works by rejecting the road curvature as a disturbance for improving path tracking error in the presence of unstructured and parametric error in the vehicle model. Sometimes, we use

a classical controller plus a feedforward controller since we usually have preview of the road ahead. This approach also improves path tracking performance considerably.

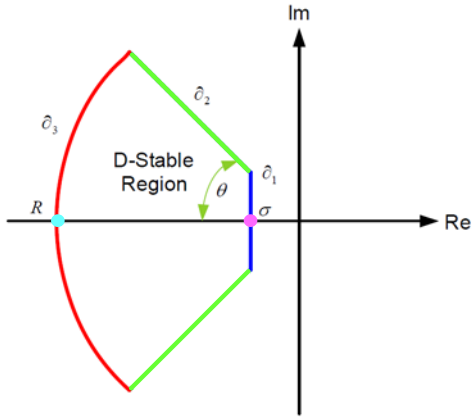


Figure 8. Illustration of D-stability region in the complex plane.

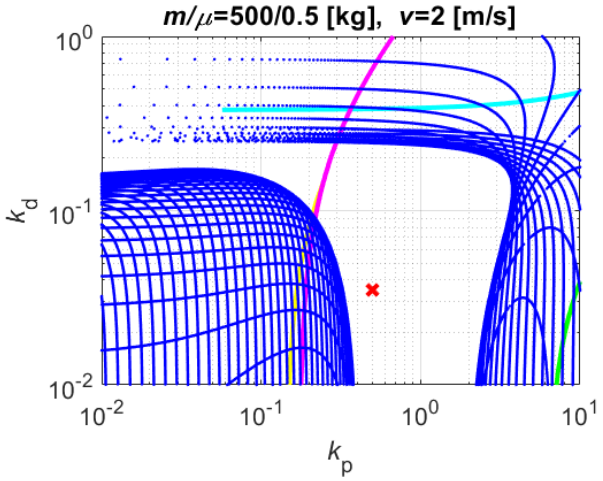


Figure 9. Parameter space of k_p and k_d at one uncertainty vertex

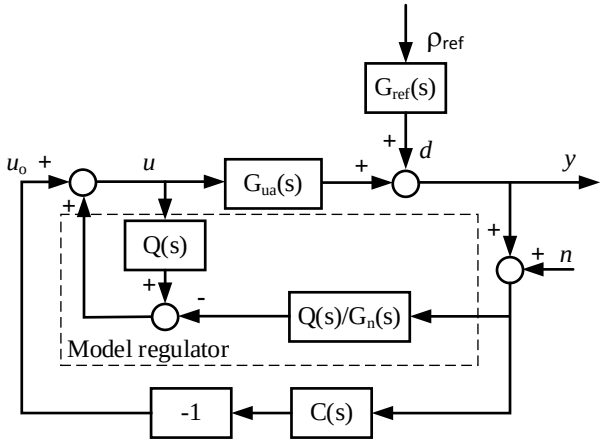


Figure 10. System diagram with the PD controller and model regulator.

Localization - eHorizon

eHorizon is an electronic horizon equipment with a detailed map inside the device that can extract road information such as GPS, speed limits, heading direction, locations of the intersections, road curvatures, traffic light and STOP sign positions. In short, it gives us a 1-2 km preview of the map ahead. Due to the information it can provide, eHorizon is a powerful tool for Path Planning and Eco-Route Planning applications, as well as Fuel Economy studies for Automated and Connected Vehicles. eHorizon uses ADASIS v2 (ADAS Interface Specification) protocol and sends map related information over a CAN network. Map data is transmitted, deconstructed by eHorizon, transmitted in a series of messages, and then reconstructed by the device that eHorizon is connected to, such as the MicroAutoBox (MABX).

The ADASIS protocol is founded on the idea of a “horizon”. The horizon is a defined distance ahead of the vehicle in which relevant map features are provided. The horizon data mainly comprises of paths (possible routes that the vehicle could take) and profiles (data about those routes such as traffic signals and slope), as seen in Figure 11. eHorizon calculates the Most Probable Path (MPP) that the vehicle is expected to follow and sends the information and profiles on this main path to the user.

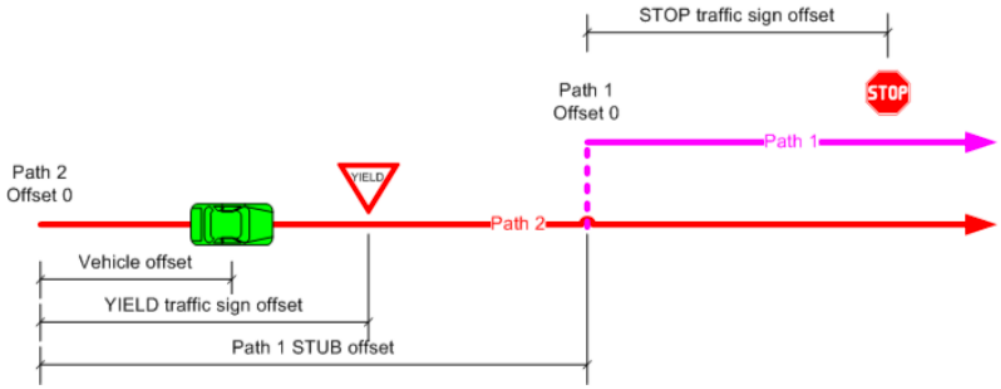


Figure 11: eHorizon Paths and Profiles

The eHorizon unit comes with its own software for visualization of the route the vehicle is following. An example of how the eHorizon software looks during travel can be seen in Figure 12. The orange arrow in Figure 12 indicates the heading direction and position of the vehicle. The solid blue line on the map illustrates the Most Probable Path the vehicle is expected to follow. The green circles in the left side of Figure represent the traffic lights and the red squares are used to illustrate the STOP sign locations.

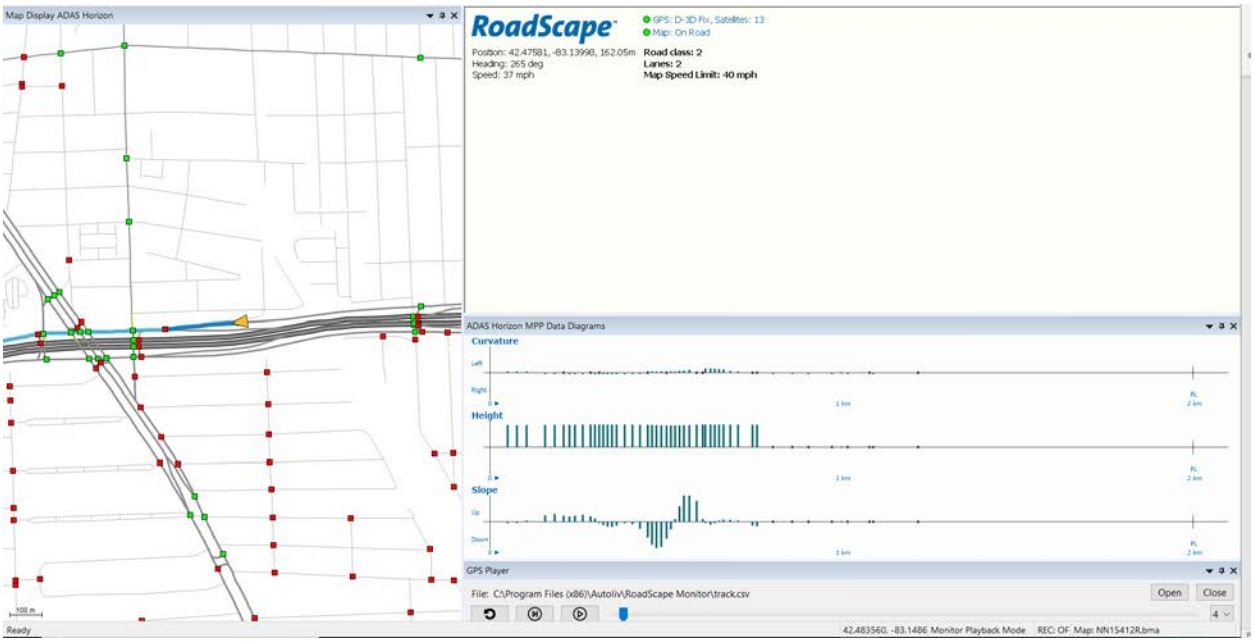


Figure 12: eHorizon window during in-vehicle experiment

As an example, some experiments were run with the eHorizon Autoliv RoadScape unit of the Automated Driving Lab (ADL) which was placed in the Ford Fusion sedan. For a drive starting from and ending at our lab building at CAR-West, some results are summarized in Figure 14. The 1st subplot in Figure 13 shows the GPS points acquired with the eHorizon unit during the actual testing. Another information that could be extracted from eHorizon was the heading angle, and the results were plotted in the 2nd subplot. eHorizon was able to provide what type of road the vehicle was on, and these results were plotted in the 3rd subplot. Finally, the speed limit of the route the vehicle travelled on was gathered from eHorizon and was plotted in the 4th subplot.

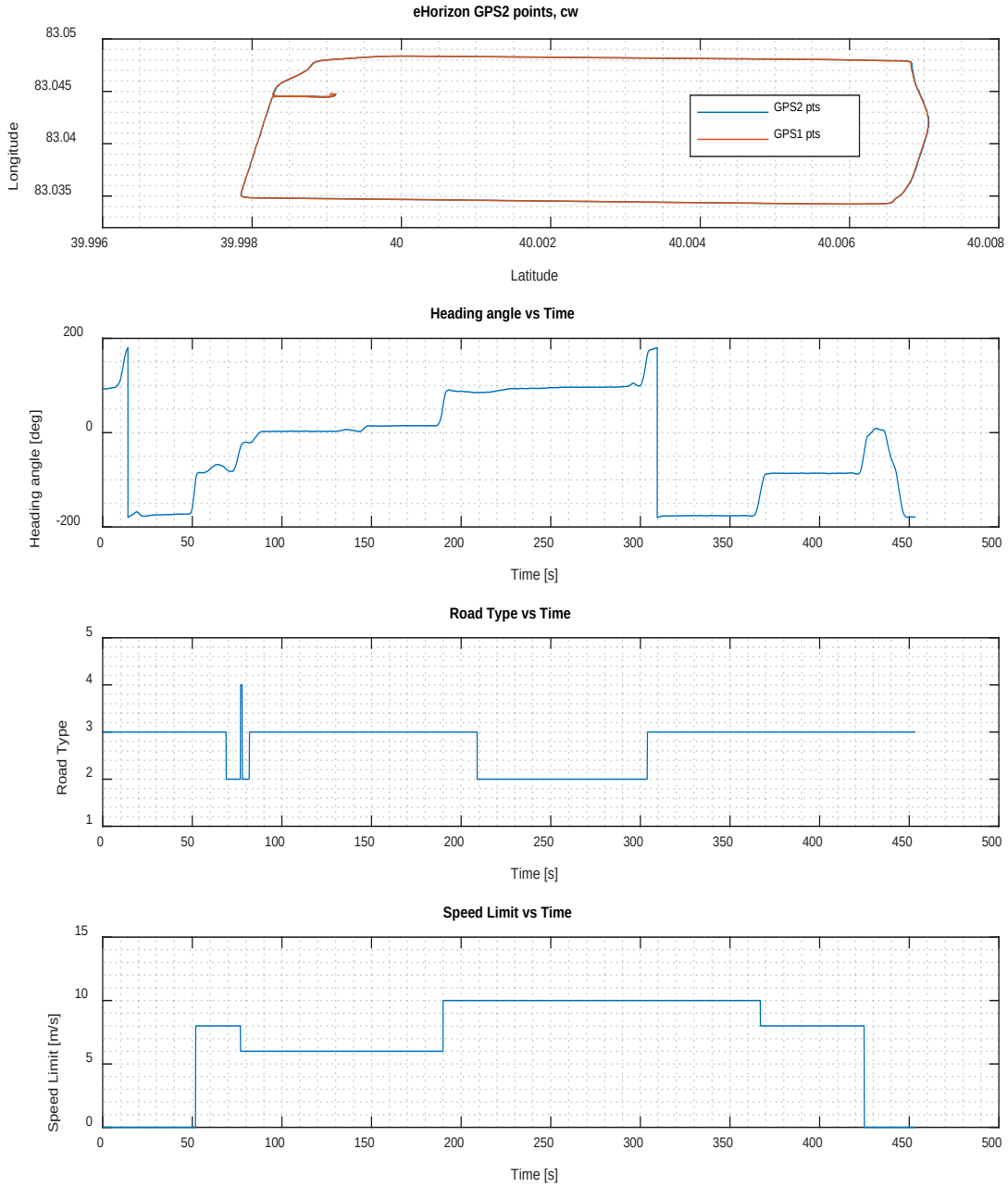


Figure 13: eHorizon results

The eHorizon unit is also able to provide upcoming traffic light and STOP sign information. To get the upcoming traffic sign information, an in-vehicle experiment was conducted with the eHorizon unit in Columbus and the test route can be seen in Figure 14. In Figure 14, red squares represent the position of the STOP signs and the yellow circle represents the position of the traffic light. Since we believe that the map is just like another sensor, the e-horizon or map preview sensor is also an important part of our unified autonomous driving architecture.

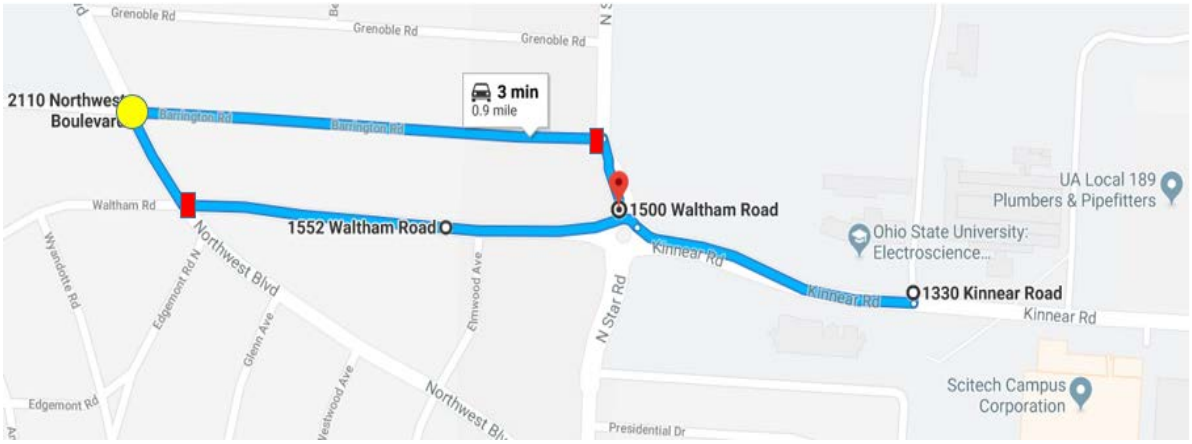


Figure 14: Test Route for eHorizon test

On the test route, shown in Figure 14traffic sign information was gathered and can be seen below in Figure 15. Whenever the upcoming traffic sign was a STOP sign, an internal variable in the eHorizon unit provided value 2. Similarly, whenever the upcoming traffic sign was a traffic light, the same internal variable in the eHorizon unit provided the value 1. Even though there were a total of 3 traffic signs for this route, the eHorizon unit changed between 1 and 2 more than 3 times. The reason for that is because the eHorizon unit predicts the Most Likely Path the vehicle is going to travel and depending on that path, shows the upcoming traffic sign info. If the driver does not follow the Most Probable Path predicted by eHorizon, then eHorizon collects the correct traffic sign information for the route the vehicle is actually following from the map.

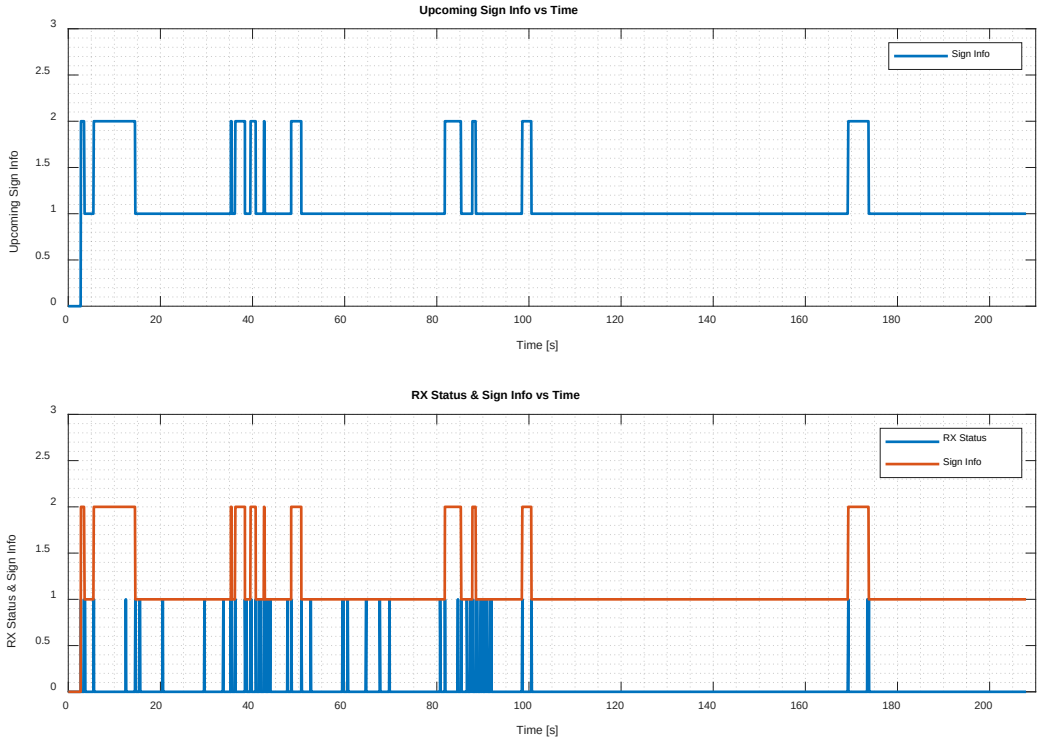


Figure 15: eHorizon providing upcoming traffic sign info

Map Building/Matching Lidar Maps

In the research and application of autonomous vehicle, to know about the current state and position of the ego vehicle is very essential. It is a demanding task to localize the ego vehicle and estimate the state of the ego vehicle as this information is needed by other functions in autonomous driving. To estimate the vehicle state, odometry and IMU data are used in different autonomous driving research and application. Also, in the past few decades, the development high accuracy dual antenna GPS encourages good performance of vehicle localization especially when real time kinematic corrections are used. In the meanwhile, the navigation suite is able to provide elaborate information about vehicle state alongside with global localization. However, high accuracy GPS cannot work well when the signal receiving gets weak due to the blockage from the environment, such as bridges and skyscrapers in an urban area. Thus, localization methods based on camera and Lidar sensors are developed.

In our study, to ensure the precision of vehicle localization and vehicle state estimation, the GPS localization method with OXTS Navigation suite and map matching based localization are implemented.

RTK-GPS

For the GPS based localization, the RTK-GPS combination are used. The OXTS GPS navigation suite provides location information that is up to 20 cm accuracy with IMU data and has a built in Kalman filter. The RTK bridge from Intuicom® is a real time kinetic localization equipment that receives location information accessing the ODOT VRS System provided by Ohio Department of Transportation [2]. Combining the two devices under strong signal receiving, the accuracy of global localization is as high as 2 cm. Typical accuracy values were around 5 cm while this dropped to 20 cm in downtown areas where some GPS satellite signals were lost.

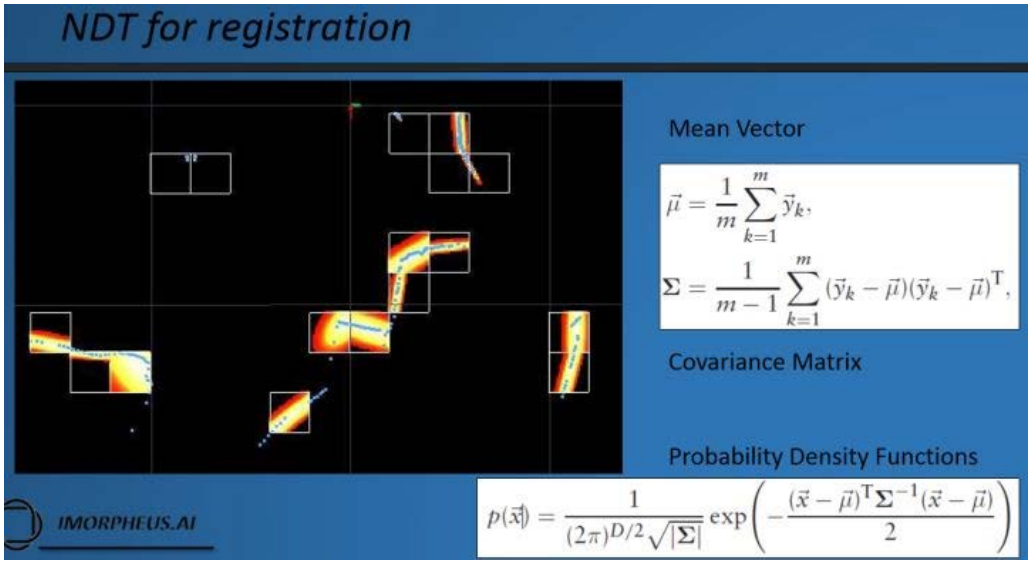
Map Building

Map matching based localization technique is based on algorithms of point cloud transformation and matching. The requirements of applying map matching in localization include well built map and good matching algorithm. In this project, a 3D point cloud map was built for map matching based localization using 3D lidar data.

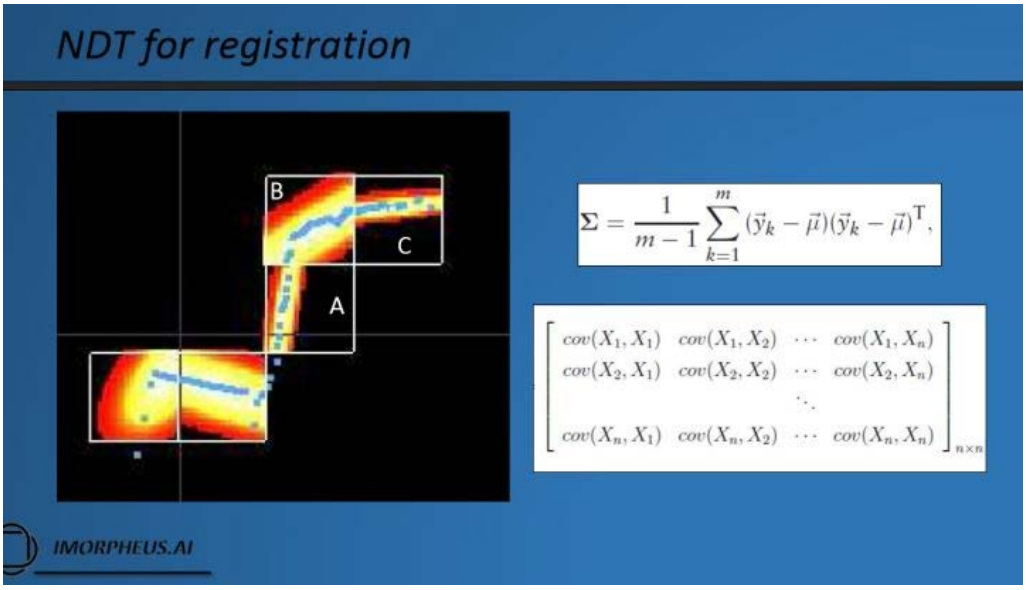
The 3D map is built from 3D point cloud data captured by a Lidar sensor. And the mapping technique is based on normal distribution transform [3] for scan matching and ndt_mapping [4], that is, align different frames of point cloud data by transforming them into same reference frame and overlap the parts where they have similar point cloud.

For the normal distribution transform of a single frame point cloud data, a piecewise-continuous probability density function is generated as in Figure 16. Mean and covariance matrices are calculated from the point cloud within the grid. The higher the probability of point cloud lying in a grid, the larger is the value of its probability density.

Thus, using the probability density function gives a good description of Lidar point cloud with respect to the density and probability of showing up.



(a)



(b)

Figure 16. NDT for 2D laser scan data. Blue dots are original point cloud, probability distribution is shown as red and yellow area. (a)The black area above is the laser scan to be transformed. Grid the laser scan and generate a density function of each grid. (b) Difference in probability of different grids. (IMORPHEUS.AI)

For 3D point cloud data, the covariance matrix is composed from the eigen value and eigen vectors of the 3D point cloud, but the idea is the same with 2D point cloud.

Map Matching based Localization

Given a 3D point cloud map, the map matching based localization is used here as a compensation for GPS localization. Recently, various algorithms of scan matching are developed which are able to find the transform between two point clouds. Iterative Close Point (ICP) algorithms [5] and Normal Distribution Transform (NDT) are two commonly used algorithms with good performance. The approach we use in this study is the NDT method along with odometry extrapolation. [6-7].

For the NDT algorithm, the inputs are pre-built 3D point cloud map and the current point cloud scanned by Lidar Sensor. By matching those two point cloud data, a transformation will be generated as $T = (t, R)$ where t is the translation of current point cloud scan in the map and R is the rotation. From the transformation information, the goal of localization in the reference frame of map is achieved. Figure 17 shows a graphical illustration of the NDT algorithm. With odometry extrapolation, a better guess of initial transformation for matching is acquired so that the precision of matching between current scan and pre-built map is increased.

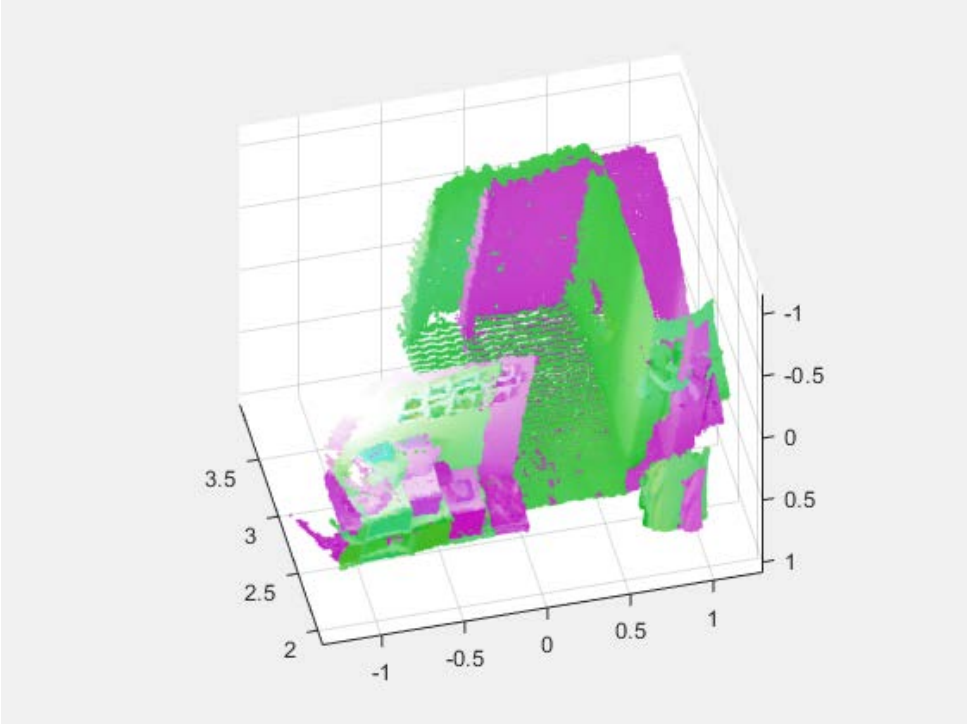


Figure 17. Sketch of NDT scan matching. The green point cloud and pink point cloud are aligning in the same reference frame after transformation.

Example Lidar Maps

Using the methodology discussed in previous part, we have built the map for several places in Columbus, Ohio. Examples are shown in Figures 18-20. Figure 18 shows the Scioto Mile route in downtown Columbus where an AV shuttle by a commercial AV tech

company is operating seven days a week. Figure 19 shows the main route of the planned Ohio State University pilot AV shuttle deployment. The route in Figure 20 is the route we use in our initial tests of localization and autonomous driving. We have used these maps for map matching based localization and compared the results with high accuracy GPS recording of the same experiment and determined that accurate localization was achieved in real time operation.

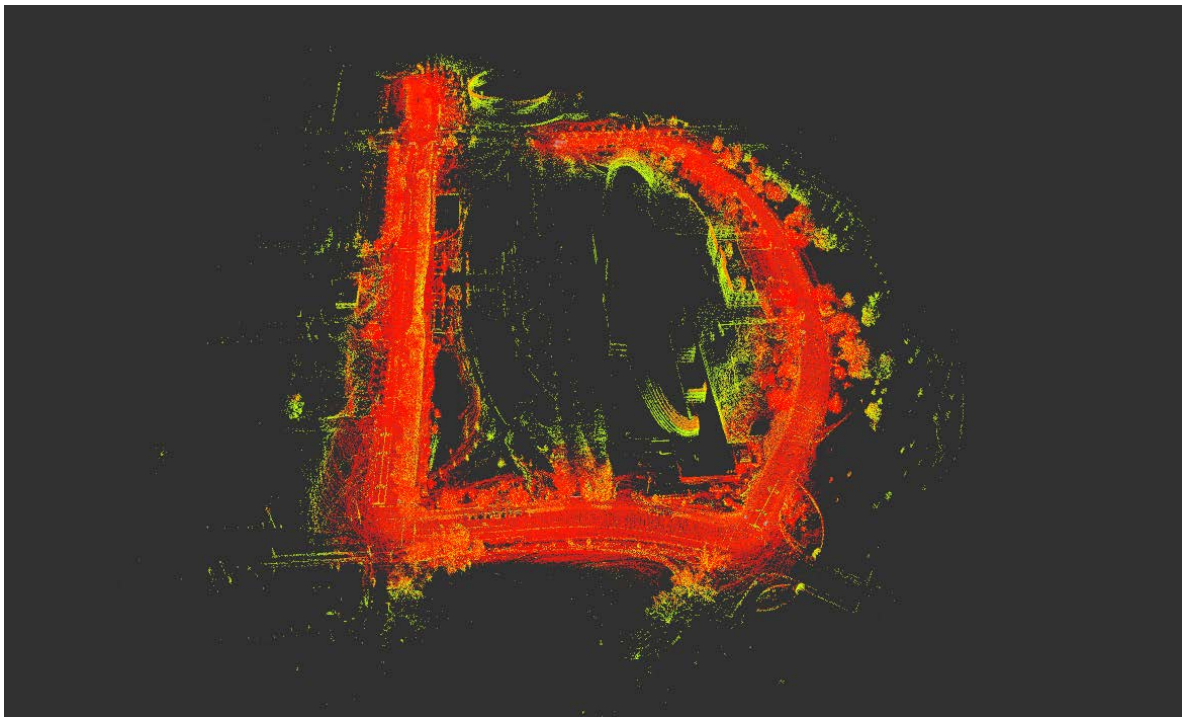


Figure 18. 3D Point cloud Map of Scioto Mile around COSI downtown Columbus.

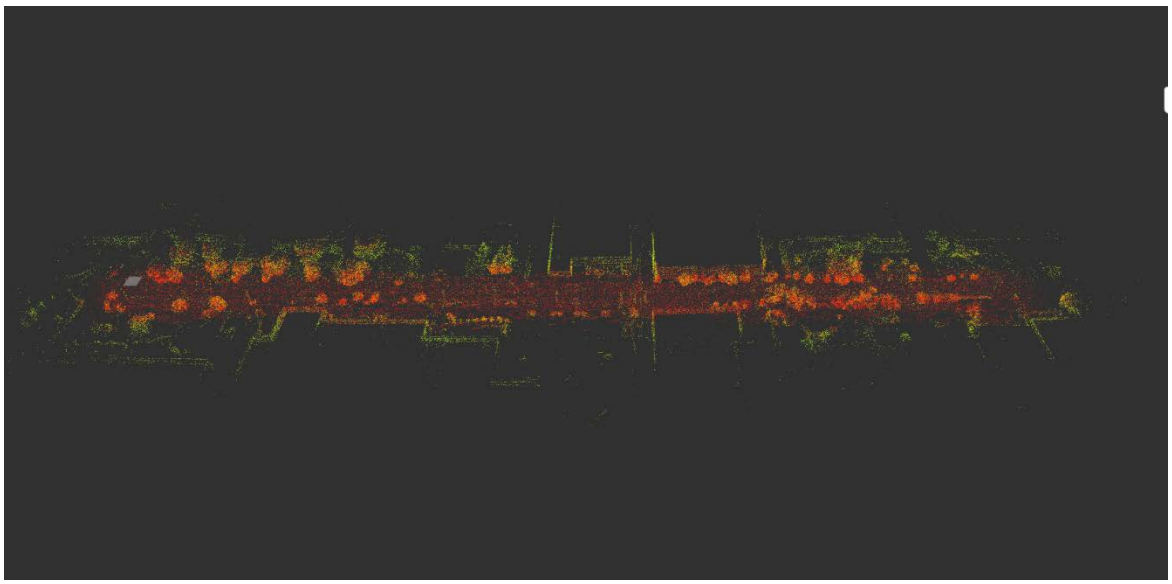


Figure 19. 3D Point cloud Map of OSU AV shuttle pilot route from CAR to CAR West.

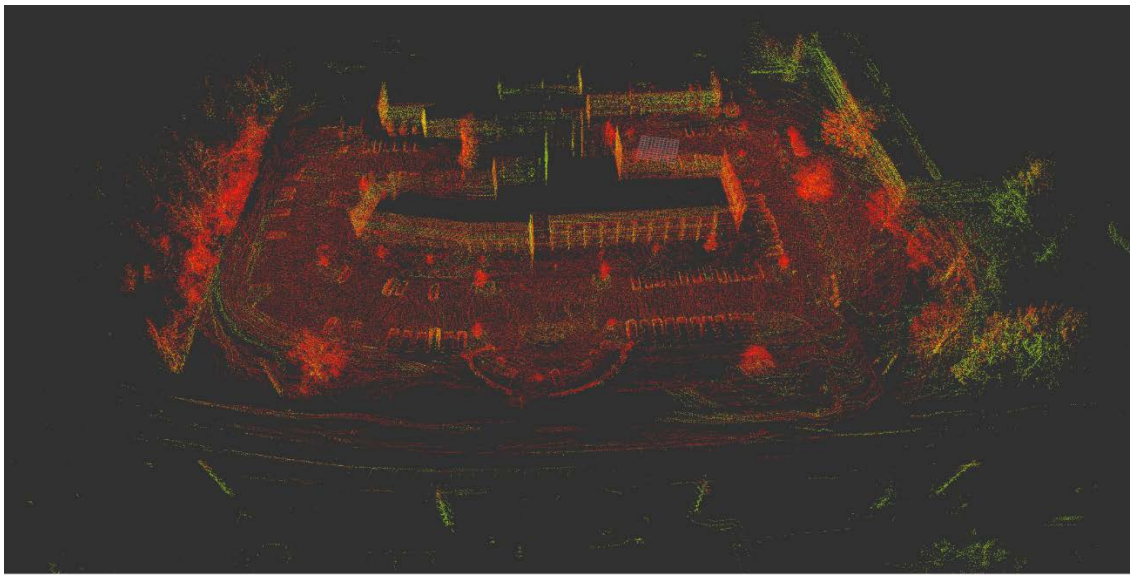


Figure 20. 3D Point cloud Map.of parking lot route at CAR West.

In this part of the report, different techniques for vehicle localization were presented and applied to real time the autonomous vehicle operation. We emphasized map matching based localization in comparison to RTK GPS localization. NDT based map building and map matching were implemented for building 3D point cloud map and solving the problem of vehicle localization and state estimation.

Path Tracking

The low level automated driving tasks are lateral and longitudinal control. The path determination and path tracking error computation are described briefly in this section. The path tracking model consists of two parts, which are offline generation of the path and online calculation of the error according to the generated path. These parts are explained in following subsections.

Offline Path Generation

The path following algorithm employs a pre-determined path to be provided to the autonomous vehicle to follow. This map is generated from GPS waypoints where these points can be pulled from an online map or can be collected through recording during a priori manual driving. These data points are then divided into smaller groups named segments with equal number of data points for ease of formulation. These segments are both used for curve fitting and velocity profiling through the route. After dividing the road into segments, a process of fitting a third order (or higher order) polynomial is performed as

$$\begin{aligned} X_i(\lambda) &= a_{xi}\lambda^3 + b_{xi}\lambda^2 + c_{xi}\lambda + d_{xi} \\ Y_i(\lambda) &= a_{yi}\lambda^3 + b_{yi}\lambda^2 + c_{yi}\lambda + d_{yi} \end{aligned} \tag{2}$$

where *i* represents the segment number and the terms *a*, *b*, *c*, *d* are polynomial fit coefficients for the corresponding segment. Fitting the data points provides effective replication of the curvature that the road carries and also eliminates the noise in the GPS data points. To provide a smooth transition from one segment to another by satisfying continuity of the polynomials and their first derivatives in X and Y, we use

$$\begin{aligned} X_i(1) &= X_{i+1}(0) \\ Y_i(1) &= Y_{i+1}(0) \end{aligned} \tag{3}$$

The X and the Y points derived from the GPS latitude and longitude data using a degree to meter conversion, are fit using a single parameter λ, where λ is the variable for the fit which varies across each segment between *0* to *1*, resulting in

$$\frac{dX_i(1)}{d\lambda} = \frac{dX_{i+1}(0)}{d\lambda} \tag{4}$$

$$\frac{dY_i(1)}{d\lambda} = \frac{dY_{i+1}(0)}{d\lambda} \tag{5}$$

Error Calculation

After the generation of path coefficients, an error is calculated for the lateral controller to use as input. Heading and position of the vehicle is provided by means of localization, in this case either SLAM, map matching or GPS. Using these, the location of the car with respect to the path, in other words, the deviation from the path is calculated. This approach reduces both oscillations and steady state lateral deviation compared to calculation with respect to position only. In order to find an equivalent distance parameter to add to the first component distance error, a preview distance l_s is defined. Then, the error becomes,

$$y = h + l_s \tan(\Delta\psi) \tag{6}$$

where $\Delta\psi$ is the net angular difference of heading of the vehicle from the heading tangent to the desired path and *y* is the total error of the vehicle computed at preview distance l_s as is illustrated in Figure 21. Finally, this error is fed to a robust controller which controls the actuation of steering of the vehicle.

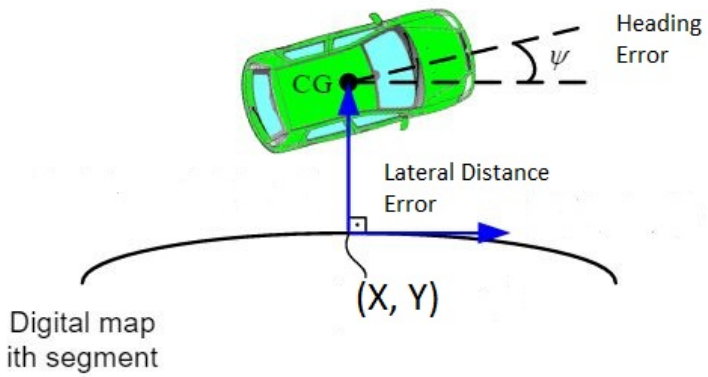


Figure 21. Illustration of error calculation.

Car Following

Cooperative Adaptive Cruise Control Design and Implementation

For the longitudinal control of the designed system, a speed controller is used when there is no traffic. In the case of a vehicle(s) in front, a Cooperative Adaptive Cruise Control (CACC) algorithm is designed. V2V communication is used if the vehicle in front has a modem and is capable of sending basic safety message set information and its acceleration. If this is not the case, the CACC algorithm automatically defaults to Adaptive Cruise Control (ACC) for car following. The developed CACC controls the inter-vehicle time gap between the target vehicle and ego vehicle using a feedforward PD controller. In this design, the feedforward information is the acceleration of the target vehicle which is communicated through a Dedicated Short-Range Communication (DSRC) modem. In this report, exemplary simulation and experimental results with the designed CACC system are presented. The presented results indicate that CACC improves the car following performance of the ego vehicle as compared to Adaptive Cruise Control alone.

The control structure of the designed CACC system is shown in the block diagram in Figure 22. String stability in both ACC and CACC modes are checked in advance. The designed control system is similar to the one designed and shown to be string -stable in [8]. Since the vehicle does not have built-in ACC, the low-level controller is designed as a gain-scheduled PI controller. As an upper controller, a PD controller with a feed-forward controller is used. To sustain the string stability a constant time headway spacing policy is employed [9]. The input of the feedforward controller is the acceleration of the target vehicle which is transmitted through DRSC radio communication.

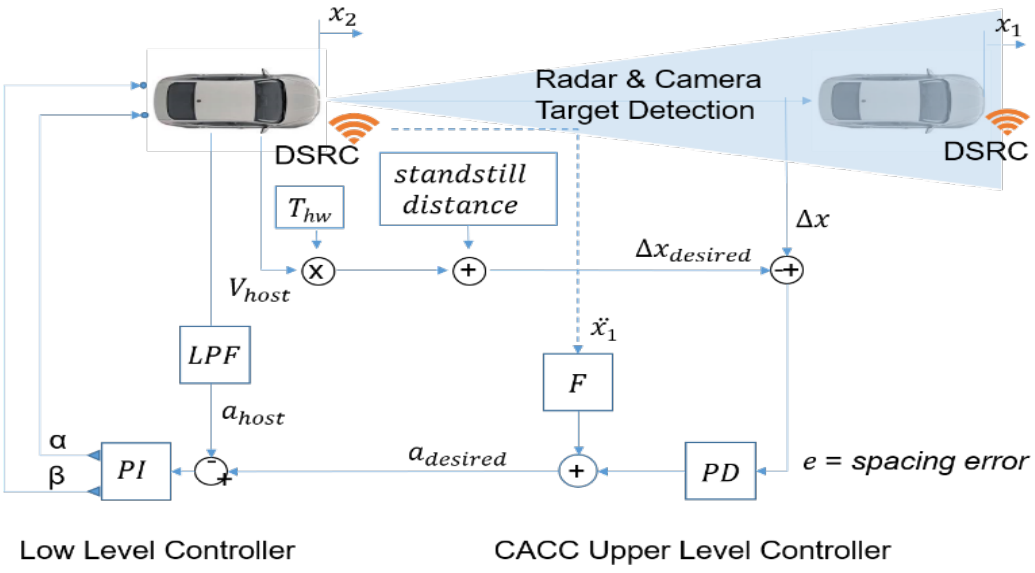


Figure 22. Cooperative Adaptive Cruise Controller (CACC) block diagram.

Development of the initial CACC model is done in the CarSim - MATLAB co-simulation environment [10]. CarSim is a vehicle simulation environment with the capability of simulating the dynamics of the vehicle. It can also simulate the target vehicle as a kinematic object. In the simulation, the target vehicle is driven by an Intelligent Driver model. By changing the desired speed and/or acceleration limits for the target vehicle, one can create different driving scenarios using the Intelligent Driver Model (IDM) [11]. Similarly, in the experiments, the target vehicle speed profile is chosen to be the same as the simulation target vehicle speed profile. The target vehicle speed profile is generated in real time with IDM driver similar to the simulation environment. In the result reported here, the target vehicle accelerates to 20 km/h and 25 km/h consecutively then it stops. In the CACC scenario, the simulated acceleration values for the target vehicle are broadcasted through DSRC OBU and are received by another OBU for the ego vehicle. One can see the experimental results for the ACC and CACC for 0.6 second time gap overlaid onto the simulation results on Figure 23.

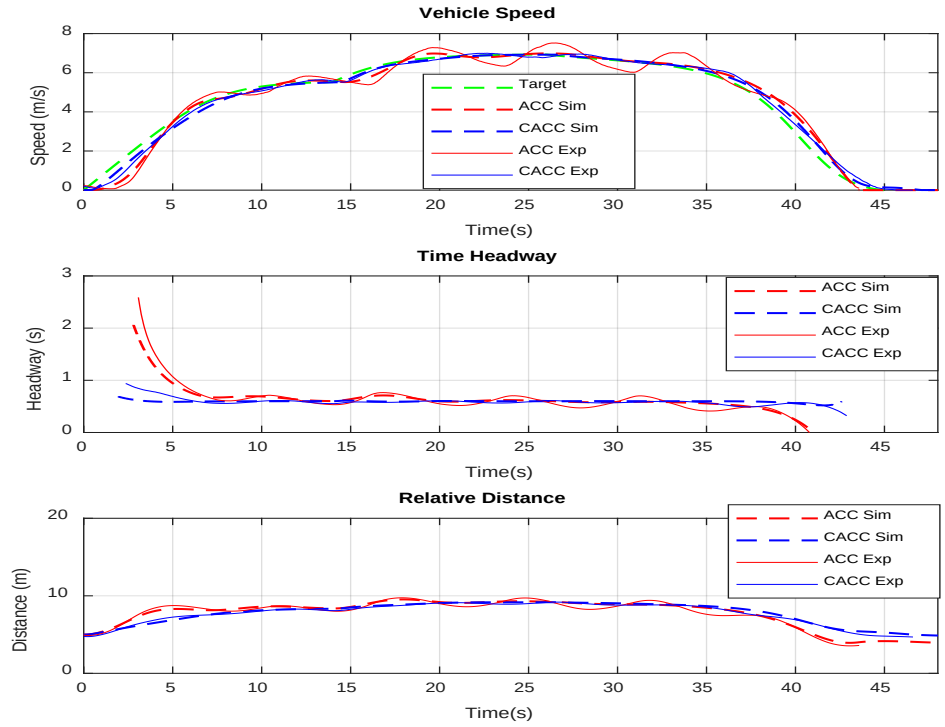


Figure 23. Comparison of ACC and CACC experimental results with simulation results for 0.6 second desired time gap.

The simulation results match with the experimental results. The small mismatches between the experiment and CarSim simulation are caused due to the fact that the CarSim vehicle model is not an exact model of the experimental vehicle. In response to the speed changes in the target vehicle speed profile, the CACC controlled vehicle starts accelerating and decelerating faster as compared to ACC. Thus, the CACC controller can follow the target vehicle more accurately. CACC time gap following performance is much better than the performance of ACC. More details on the designed system can be read in [12].

Pedestrian Collision Avoidance

Path Modification with Elastic Band

Pedestrian collision avoidance is introduced in this subsection but the method presented is applicable to collision avoidance with other obstacles also. After the position of the pedestrian is obtained, an alternative path to be followed is created if a collision is imminent. For our case, this path is created by modifying the existing points on the path according to the position of the pedestrian. Our path modification algorithm is based on the elastic band theory, where socially acceptable distance is also considered in modifying the deformed path. In elastic band theory, the initial path is deformed by internal and external forces acting on the band. Internal forces are spring forces which hold the band or the path together while external forces keep the band or path away from the pedestrian like artificial potential field forces. Figure 24 shows an

initial path displayed as an elastic band which is deformed by both internal and external forces in the presence of a pedestrian.

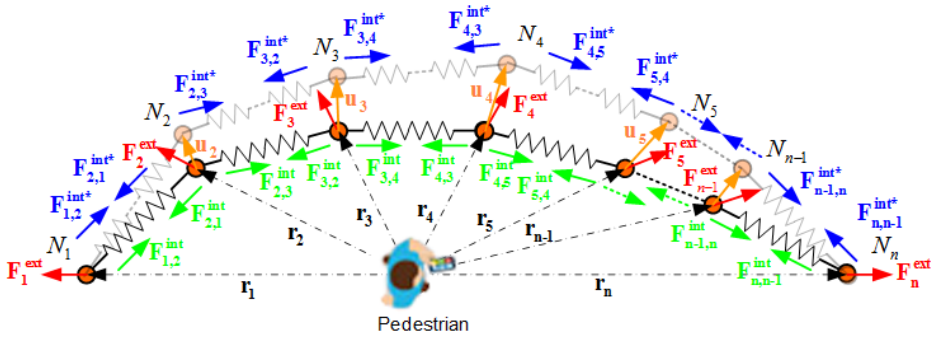


Figure 24. An initial path deformed by internal and external forces in the presence of a pedestrian (or other obstacle) on the path

The elastic band is a sequence of displaceable nodes denoted by N_i in Figure 24 that initially correspond to the original local path of the autonomous shuttle in the vicinity of the detected pedestrian. The initial positions of the nodes N_i with respect to the pedestrian are shown by position vectors r_i. Internal forces are formulated by adding springs with stiffness k_s and spring force $\mathbf{F}_{\mathbf{i,j}}^{\mathbf{int}}$ acting on node i due to the adjacent nodes N_j with $j=i+1$ or $j=i-1$ for $i=1,2,\ldots,n$. The function of internal forces is to hold the nodes or the local path together as a displaceable part of the route of the autonomous shuttle as shown in Figure 24. External forces $\mathbf{F}_{\mathbf{i}}^{\mathbf{ext}}$ acting on node N_i with $i=1,2,\ldots,n$ are defined once a pedestrian is detected to deform the band and hence the local path away from the pedestrian like artificial potential field forces. The external forces keep deforming the local path around the pedestrian who may be moving while the internal forces keep the nodes together in the form of a collision free path to be followed. u_i for $i=1,2,\ldots,n$ are the deformations of the nodes under the action of external and internal forces when a pedestrian is detected. The internal forces $\mathbf{F}_{\mathbf{i,j}}^{\mathbf{int}}$ become $\mathbf{F}_{\mathbf{i,j}}^{\mathbf{int*}}$ after the deformation of the local path. After a pedestrian is detected, external forces are applied and the nodes of the deformed path become the new positions r_i+u_i for $i=1,2,\ldots,n$ as determined by the balance of internal and external forces acting on the nodes.

The static balance of internal forces acting on node N_i in Figure 24 before a pedestrian is detected are

$$\mathbf{F}_{\mathbf{i,i-1}}^{\mathbf{int}}+\mathbf{F}_{\mathbf{i,i+1}}^{\mathbf{int}}=k_s\left(\mathbf{r}_{\mathbf{i-1}}-\mathbf{r}_i\right)+k_s\left(\mathbf{r}_{\mathbf{i+1}}-\mathbf{r}_{\mathbf{i}}\right)=0 \tag{7}$$

After the pedestrian is detected and external forces are applied, the static balance of internal and external forces acting on node N_i in Figure 24 become

$$\mathbf{F}_{\mathbf{i,i-1}}^{\mathbf{int*}} + \mathbf{F}_{\mathbf{i,i+1}}^{\mathbf{int*}} + \mathbf{F}_{\mathbf{i}}^{\mathbf{ext}} = k_s\left(\mathbf{r}_{\mathbf{i-1}} + \mathbf{u}_{\mathbf{i-1}} - \left(\mathbf{r}_i + \mathbf{u}_{\mathbf{i}}\right)\right) + k_s\left(\mathbf{r}_{\mathbf{i+1}} + \mathbf{u}_{\mathbf{i+1}} - \left(\mathbf{r}_{\mathbf{i}} + \mathbf{u}_{\mathbf{i}}\right)\right) + \mathbf{F}_{\mathbf{i}}^{\mathbf{ext}} = 0 \tag{8}$$

which using the identity in Equation 7 becomes

$$\mathbf{F}_{\mathbf{i}}^{\mathbf{ext}} = -\left[k_s\left(\mathbf{u}_{\mathbf{i-1}} - \mathbf{u}_{\mathbf{i}}\right) + k_s\left(\mathbf{u}_{\mathbf{i+1}} - \mathbf{u}_{\mathbf{i}}\right)\right] = -k_s\left(\mathbf{u}_{\mathbf{i-1}} - 2\mathbf{u}_{\mathbf{i}} + \mathbf{u}_{\mathbf{i+1}}\right) \tag{9}$$

The external force $\mathbf{F}_{\mathbf{i}}^{\mathbf{ext}}$ acting on node N_i is calculated as a repulsive force using

$$\mathbf{F}_{\mathbf{i}}^{\mathbf{ext}} = \begin{cases} \left(\mathbf{F}_{\mathbf{i}}^{\mathbf{ext}}\right)_{\max}, \left|\mathbf{r}_{\mathbf{i}}\right| < d \\ -k_e\left(\left|\mathbf{r}_{\mathbf{i}}\right| - r_{\max}\right)\dfrac{\mathbf{r}_{\mathbf{i}}}{\left|\mathbf{r}_{\mathbf{i}}\right|}, \; d \le \left|\mathbf{r}_{\mathbf{i}}\right| \le r_{\max} \\ 0, \quad \left|\mathbf{r}_{\mathbf{i}}\right| > r_{\max} \end{cases} \tag{10}$$

where |…| denotes the magnitude of the argument. $\left(\mathbf{F}_{\mathbf{i}}^{\mathbf{ext}}\right)_{\max}$ in Equation (10) is used to saturate the repulsive force within $\left|\mathbf{r}_{\mathbf{i}}\right| < d$ so that it does not go to infinity as $\left|\mathbf{r}_{\mathbf{i}}\right| \to 0$. k_e is the stiffness associated with the repulsive force and r_{max} is the range of the repulsive force. Once a pedestrian is detected and localized with respect to the path of the vehicle using V2P communication, Equations 9 and 10 are solved to obtain the new coordinates r_i+u_i for i=1,2,…,n of the locally deformed path. In the case of a moving pedestrian, the computations are repeated at each time step to continue to locally deform the obstacle avoidance path to be followed.

The distance d in Equation 10 is also used to model the physical dimension of the autonomous vehicle $d_{vehicle}$. Noting that $\left|\mathbf{r}_{\mathbf{i}}\right| < d$ is a circular region around the pedestrian to be avoided, d is adjusted such the pedestrian stays within that region during any short duration relative displacement and the V2P communication delay of the car within the detection sampling instants. In the case of a moving pedestrian, the circular region $\left|\mathbf{r}_{\mathbf{i}}\right| < d$ keeps moving with the pedestrian, requiring the local path modification calculations based on solving Equation 9 and 10 to take place within the steering control sampling time.

In socially acceptable collision avoidance, the circular region $\left|\mathbf{r}_{\mathbf{i}}\right| < d$ is increased to also accommodate a pedestrian socially acceptable distance of about 1.5 m and d is calculated using

$$d = d_{vehicle} + d_{pedestrians} + d_{social} \tag{11}$$

where $d_{vehicle}$, $d_{pedestrian}$ and d_{social} account for our the autonomous vehicle dimensions, the pedestrian possible motion between two V2P detection sampling instances and the social acceptance distance for pedestrians. A maximum possible pedestrian speed of 1.5 m/sec is used in computing $d_{pedestrian}$. While pedestrian (obstacle) detection was based on V2P communication here, the method is applicable for the case of detection using other sensors like lidar, camera and radar.

Following the Modified Path

A preview distance of 15 m to the V2P detected pedestrian is used in the HiL simulations and experiment here to modify the path based on the elastic band method of the previous sub-section. After modification of the points on the path with the elastic band method, the modified path should be followed instead of the actual path. Our past work involved fitting several segments of cubic polynomials to the deformed elastic band nodes. This method was implemented in real time for a stationary pedestrian in our previous work. However, this approach limited our real time computation speed for moving pedestrians where the curve fits had to be repeated at each time instant. For this reason, a simpler method for following the modified path with a point to point approach is formulated and used here. This method runs much faster since it uses a much faster computation to determine lateral deviation at each node. This method is illustrated in Figure 25.

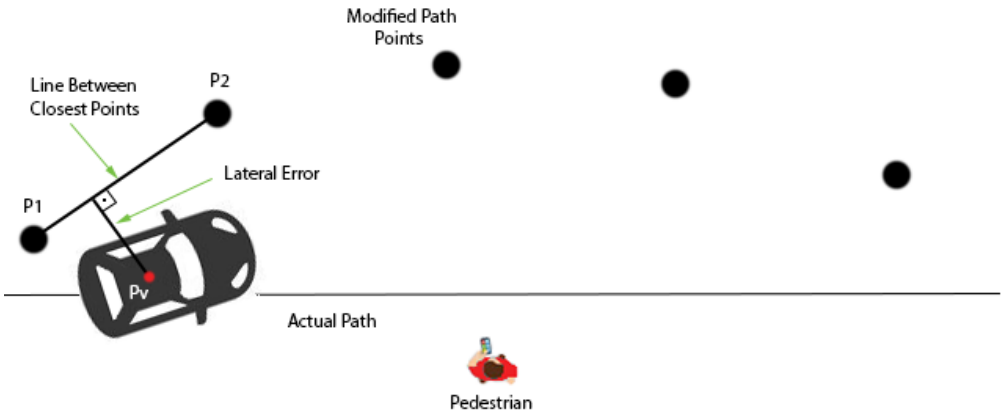


Figure 25. Lateral error calculation for pedestrian avoidance

In our new approach, the orthogonal distance to the line between two deformed nodes, which are the closest to the vehicle are calculated as shown in Figure 25. This distance is considered to be the lateral error for the modified path and used as the source of the lateral error for the steering controller, when there is a need to avoid the pedestrian. Calculation for a full elastic band method modified path consisting of 500 nodes and one pedestrian (obstacle) takes approximately 20 ms for one step while this time was approximately 200 ms for our previous method that used cubic curve fitting as explained. The lateral error $\boldsymbol{e_y}$ for this method after the V2P based detection of a pedestrian on the path within the 15 m preview distance is

$$e_y = \frac{\det(\begin{bmatrix} a \\ b \end{bmatrix})}{\|a\|} \tag{12}$$

$$a = P_2 - P_1 \tag{13}$$

$$b = \mathrm{P}_v - P_1 \tag{14}$$

where *a* and *b* are row vectors calculated from Euclidean coordinates of closest points P_1, P_2 and vehicle's coordinate P_v. $\|a\|$ is the length of the vector a. This calculation is done at every step that needs obstacle avoidance behavior when there is a V2P detected pedestrian (obstacle) who is nearby. A flowchart that presents the steps of the overall algorithm and decision making is shown in Figure 26.

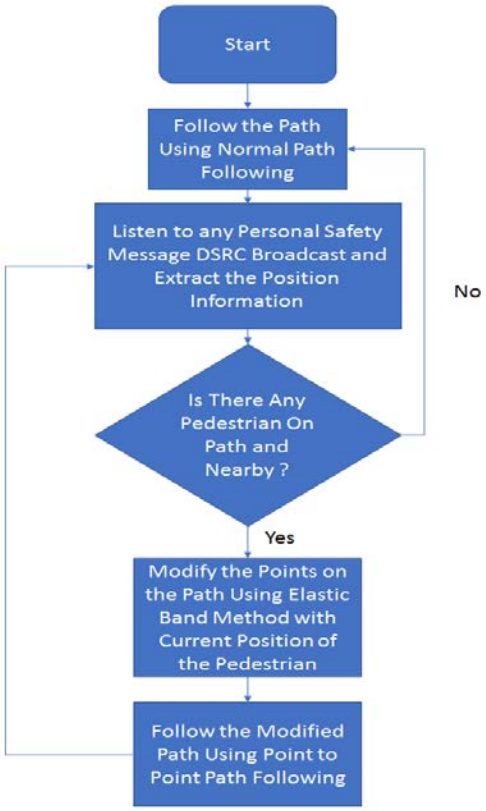


Figure 26. Flowchart of the algorithm

Processing/Perception

Nvidia Drive PX2

The Nvidia Drive PX2 AutoChauffeur platform features two discrete GPUs and various functions for the development of autonomous vehicles. The Drive PX2 is intended to be used for online processing of trained neural networks to enable Autopilot and self-driving functionalities. The platform features already trained networks like DriveNet, LaneNet, and OpenRoadNet. Some functions that can be easily implemented with the help of these networks include basic object detection and tracking, free-space detection, and lane detection. Such algorithms were run against the data we collected in the Easton Town Center shopping area. The results give promise to the development of autonomous vehicles using the Drive PX2 as the main controller in the vehicle. The Drive PX2 neural networks are based on the use of camera sensors and are suitable for autonomous shuttles in dry weather conditions.

The lane-detection algorithm provides an indicator on the lane the vehicle is currently on along with markings on the adjacent lanes when they are present. The algorithm uses Nvidia's pre-trained neural network LaneNet as shown in Figure 27.

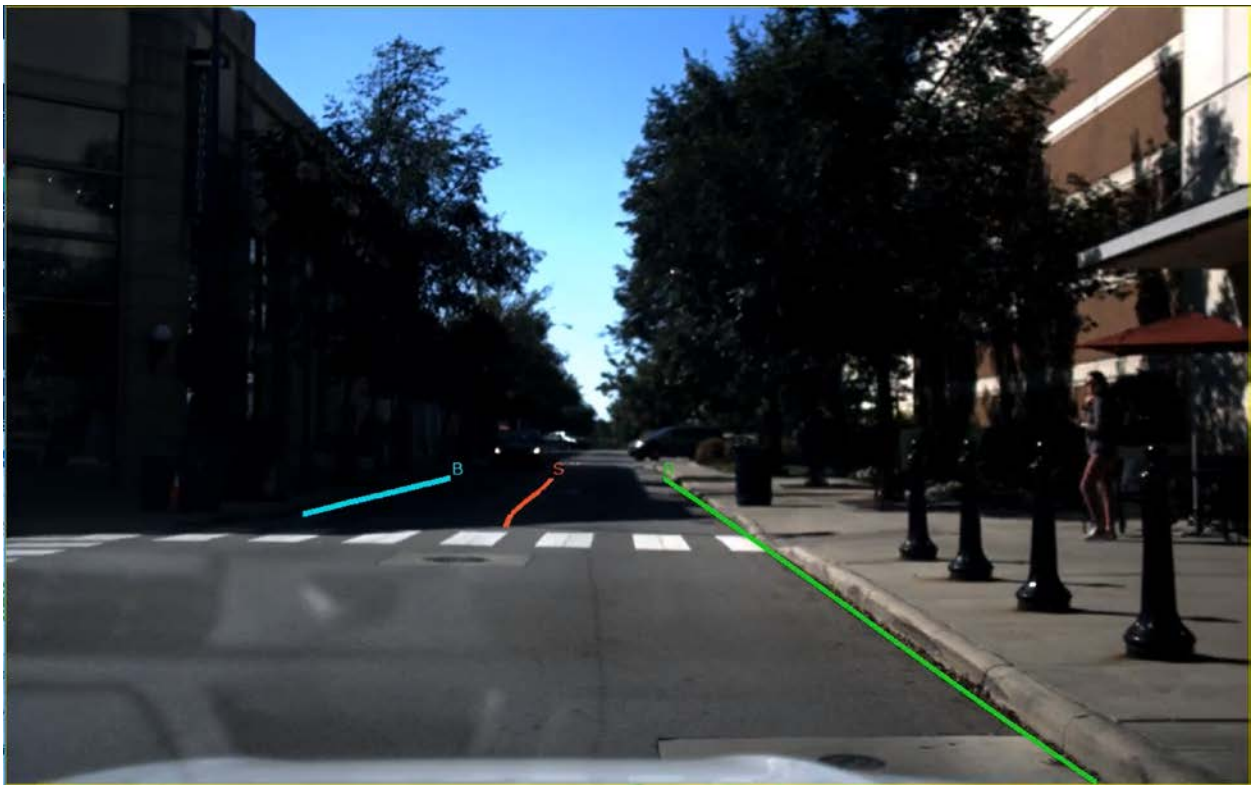


Figure 27. Lane detection algorithm.

The free-space detection algorithm provides drive-able collision-free space and labels each pixel on the boundary with colors indicating whether the pixel is associated with a vehicle, pedestrian, curb or other. The algorithm uses Nvidia's already pre-trained neural network OpenRoadNet as shown in Figure 28.

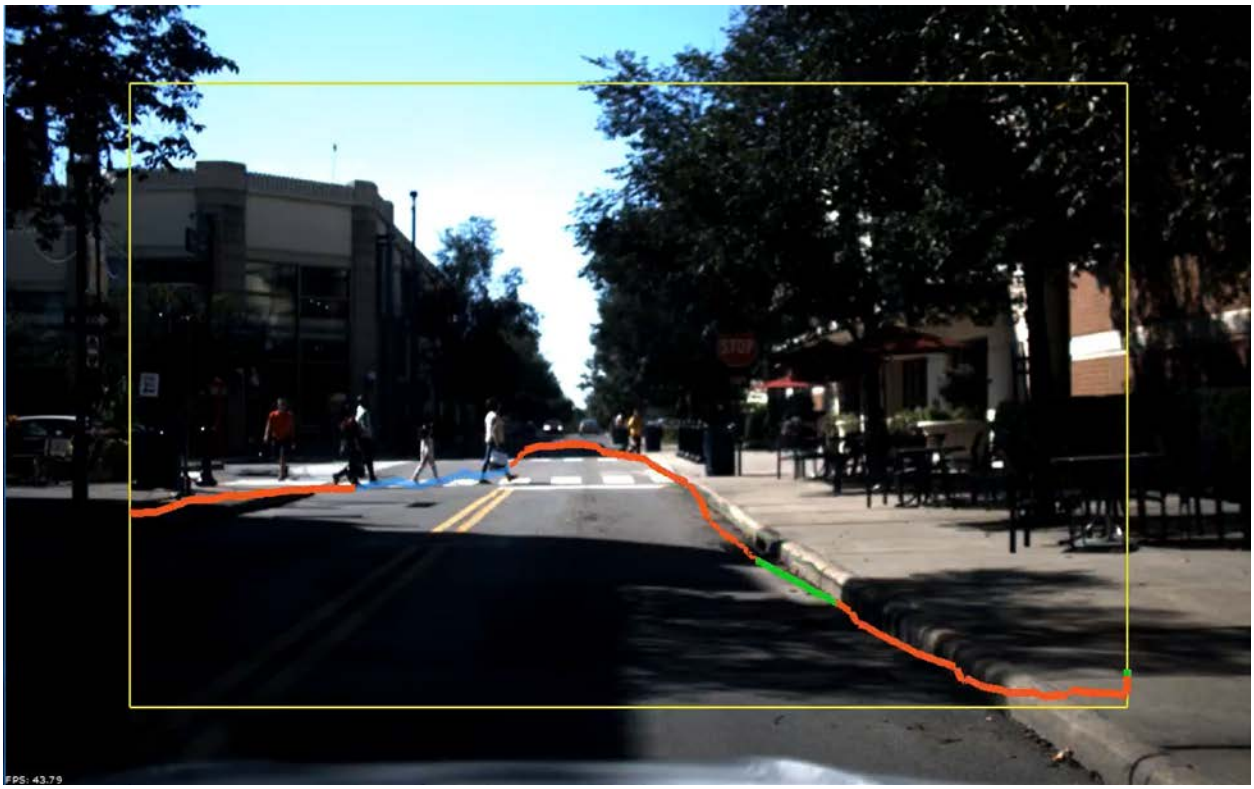


Figure 28. Free space algorithm.

Furthermore, the Nvidia PX2 architecture provides support for other external sensors which can be utilized in conjunction with some of the more sophisticated functions in the library to detect road parameters and actuate the vehicle via the car controller box.

Detection and Classification with YOLO

For the perception, one of the popular deep neural network methods YOLO [13] has been implemented in our experimental vehicle to detect the relevant objects on the road. In our case, relevant information can be considered as vehicles, pedestrians, cyclists, traffic lights stop signs and so on. For this perception task, a camera mounted at the windshield of our experimental vehicle is connected to the in-vehicle computer which is equipped with an Nvidia GPU. Example detection results for the deployed algorithm can be seen in Figure 29.

Figure 29. Example perception results with YOLO real-time object detection algorithm at the Ohio State University main campus.

Semantic Segmentation

We investigate several neural network architectures that are tailored to the semantic segmentation task for outdoor scenarios.

FC-DenseNet

FC-DenseNet [14] is adopted from DenseNet architecture [15], which is built from dense blocks and pooling layers. Each dense block is iteratively concatenated with previous feature maps. Jégou et al. [14] suggest to extend DenseNet to a fully connected network similar to fully connected ResNet [16]. Regular fully connected networks use convolution, upsampling operations, and skip connections. In order to make DenseNet fully connected, the authors use a dense block rather than a convolution operation. Figure 30 illustrates an overview of FC-DenseNet.

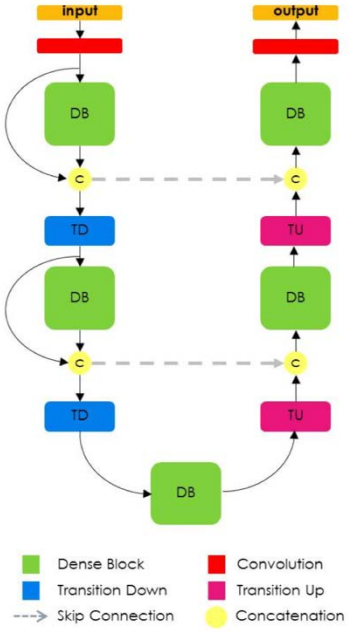


Figure 30. FC-DenseNet architecture. Transition Up (TU) is the upsampling operation, which is a 3x3 transposed convolution. Transition Down (TD) is consist of batch normalization, ReLU, 1x1 convolution, dropout with p = 0.2, and 2x2 max pooling [14].

BiSeNet

The second model is BiSeNet [17], which is designed to perform real-time semantic segmentation. The two main components of BiSeNet are the Spatial path and the Context path. Spatial path tries to preserve the input resolution by encoding rich spatial information using large feature maps. The Spatial path is comprised of three convolutional layers, each followed

by batch normalization [18] and ReLU [19]. The Context path is designed to provide large receptive fields. The Context path is a lightweight backbone architecture such as Xception [20] that downsamples feature maps in order to obtain high level context information.

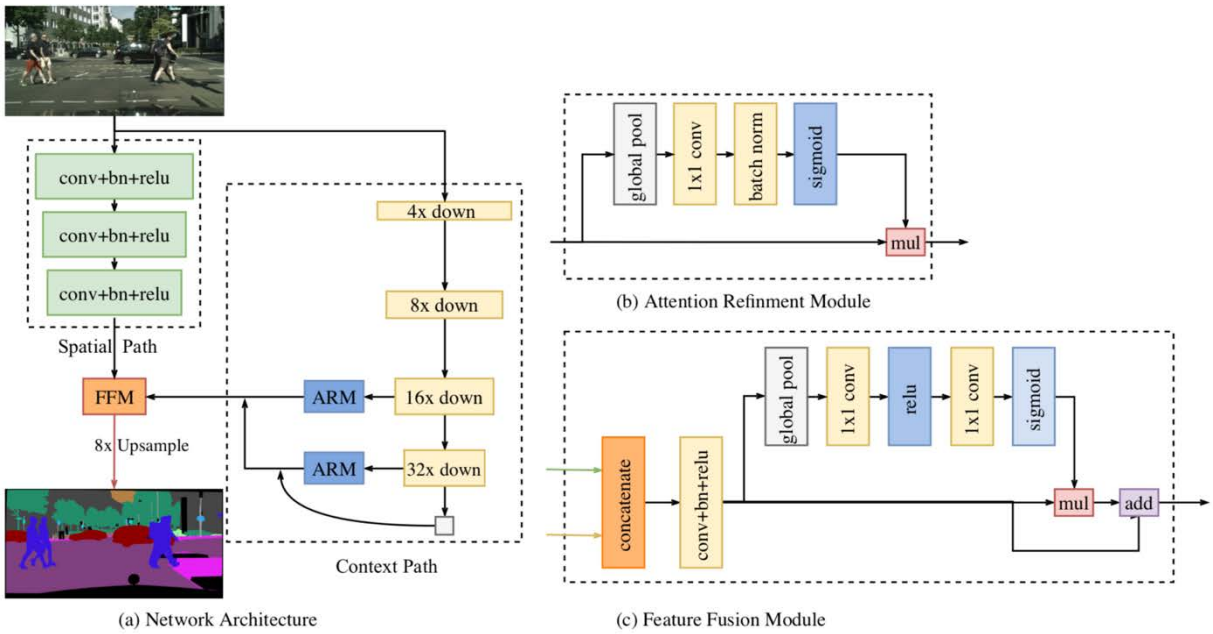


Figure 31. BiSeNet architecture. a) The main architecture with Spatial path and Conext path. b) Attention Refinement Module (ARM). c) Feature Fusion Module that is responsible for combining feature maps of different information level [17].

The Context path also includes an Attention Refinement Module (ARM), which is a global pooling average followed by 1x1 convolution, batch normalization and a sigmoid layer. The final output of this module is the multiplication of the input of this module by the output of the sigmoid layer. ARM is designed to capture global context information and computes a vector that guides the feature learning process. Since the Context path and the Spatial path contain different information about the scene, one cannot simply add feature maps resulting from these components. The Spatial path provides features that are rich in detail, while the Context path generates features that are high level. Combing these features is done via a module called Feature Fusion Module (FFM). FFM concatenates features, applies batch normalization, then uses a global average pooling. Figure 31 illustrates the BiSeNet architecture and its modules.

Dense-ASPP

A major problem in autonomous driving is the significant change in objects scales. Dense-ASPP [21] tackles this problem by using Atrous Spatial Pyramid Pooling (ASPP). ASPP [22] offers large receptive fields while preserving the spatial resolution. ASPP

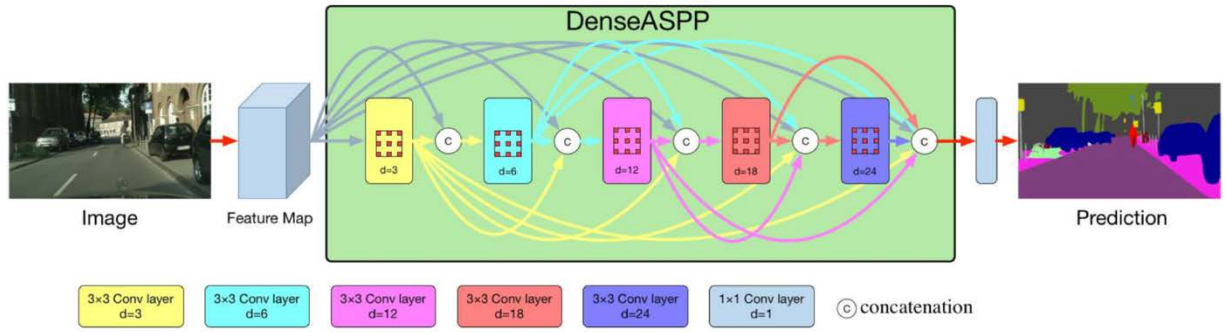


Figure 32. DenseASPP architecture [21].

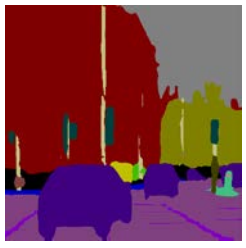
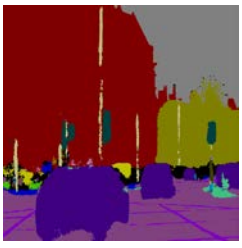
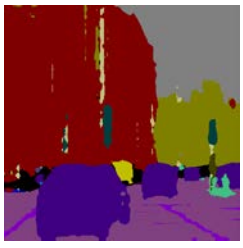
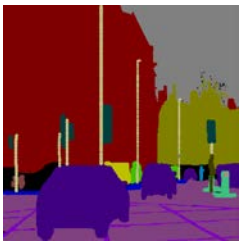
concatenates multiple atrous-convolved features with different dilation rates to generate multi-scale features. In order to make features dense enough for autonomous driving scenarios, ASPPs are densely connected. Dense-ASPP is basically a set of densely connected atrous convolutional layers (Figure 32). For a more detailed description of this network please refer to [21].

Semantic Segmentation Experiments

We investigate the performance of the three models: BiSeNet, DenseASPP, and FC-Dense on the CamVid dataset. CamVid is an outdoor scene dataset which is recorded at 15 fps from the driver's point of view. It contains 367 training, 101 validation, and 233 test images totaling up to 701 images. The dimension of images is 960x720, and the dataset includes 32 semantic categories: Animal, Pedestrian, Child, Rolling cart/luggage/pram, Bicyclist, Motorcycle/scooter, Car (sedan/wagon), SUV / pickup truck, Truck / bus, Train, Misc., Road, Shoulder, Lane markings drivable, Non-Drivable, Sky, Tunnel, Archway, Building, Wall, Tree, Vegetation misc., Fence, Sidewalk, Parking block, Column/pole, Traffic cone, Bridge, Sign / symbol, Misc. text, Traffic light, Other.

Some categories comprise less than 0.1% of the dataset. Therefore, we do not report them in the following tables.

Table 2. Visual comparison between BiSeNet, FC-DenseNet, DenseASPP.

Image	BiSeNet	DenseASPP	FC-DenseNet	Ground Truth
				

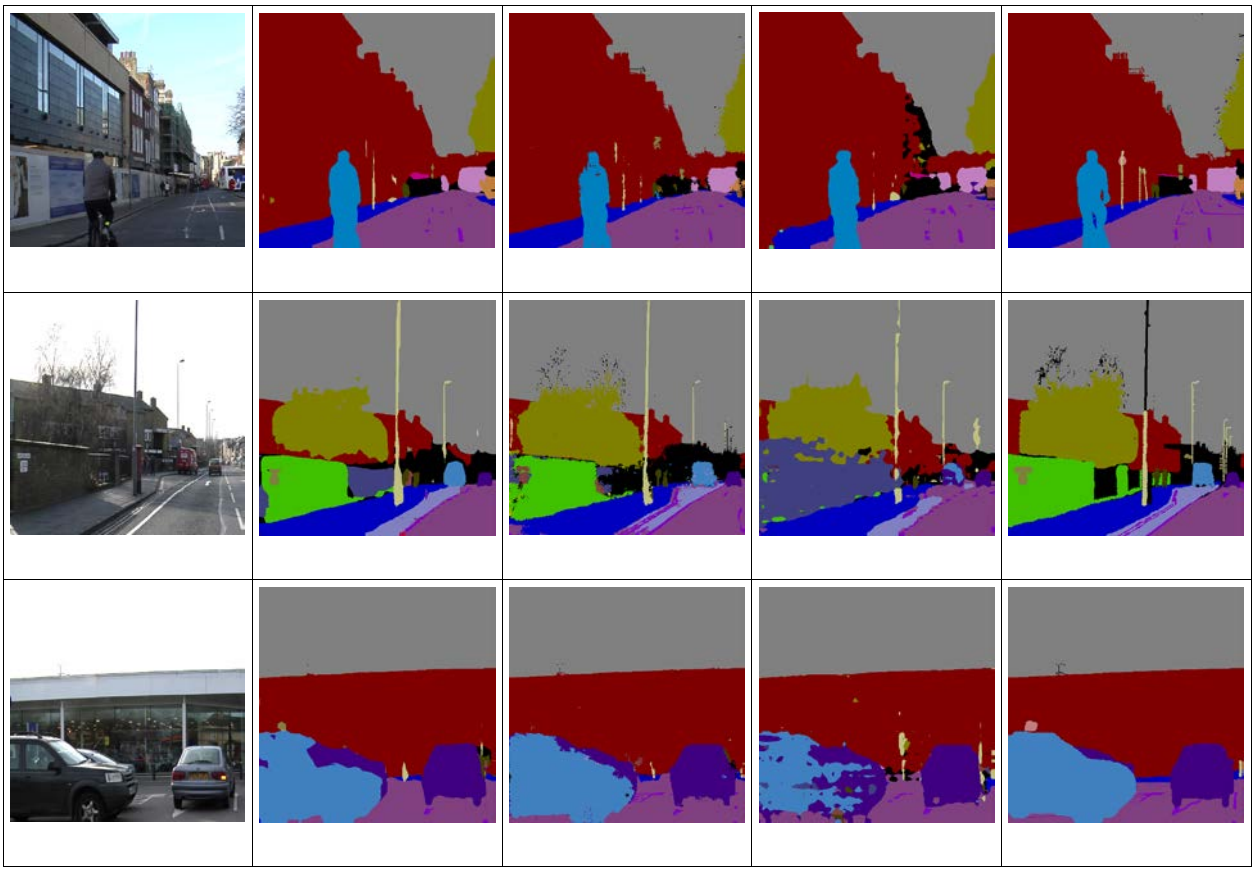


Table 3 Accuracy, precision, recall, F1-score, and mean IoU for BiSeNet, DenseASPP, and FC-DenseNet. Values are given in percentage (%).

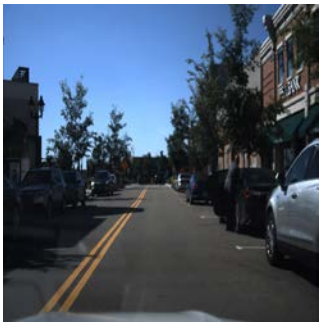
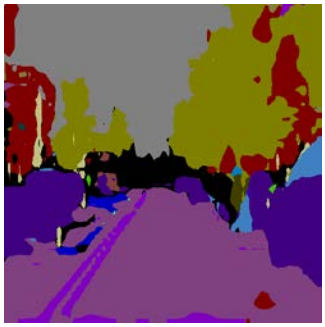
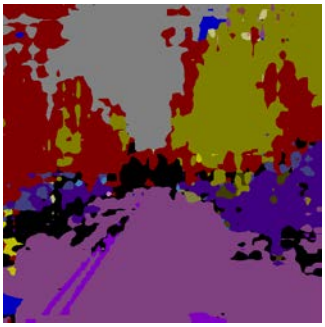
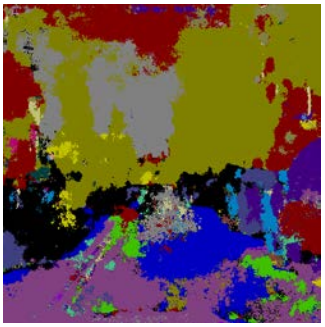
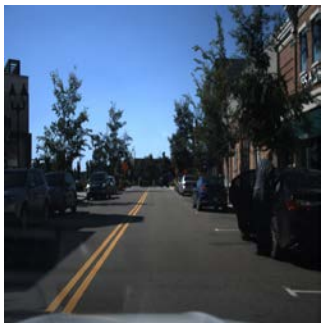
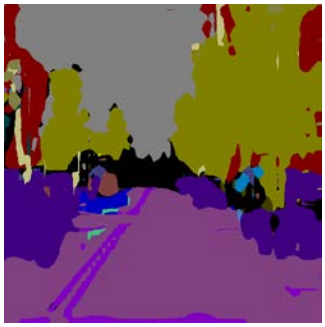
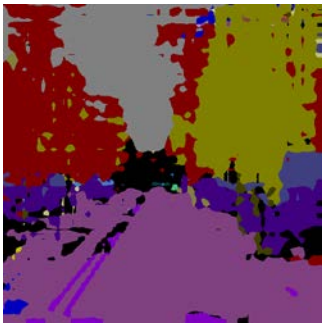
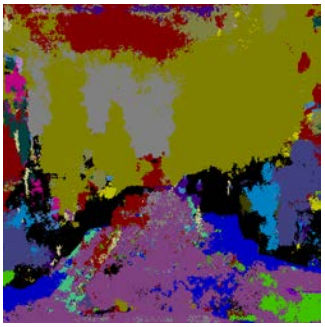
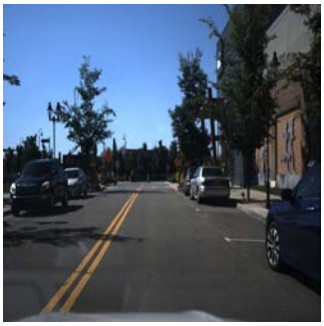
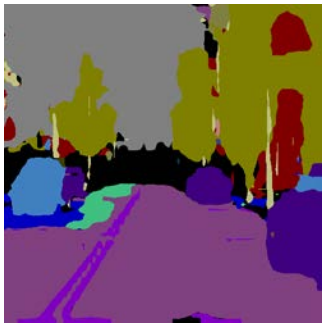
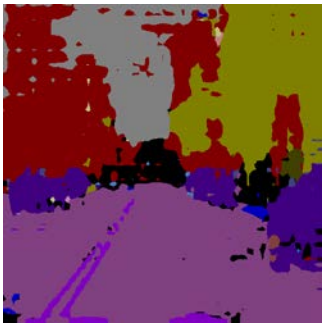
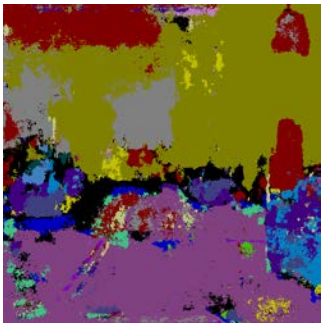
	Accuracy	Precision	Recall	F1-score	Mean IoU
BiSeNet	84.67	86.07	84.67	84.21	49.72
DenseASPP	77.03	78.38	77.03	75.35	39.12
FC-DenseNet	85.96	88.11	85.96	86.0	49.57

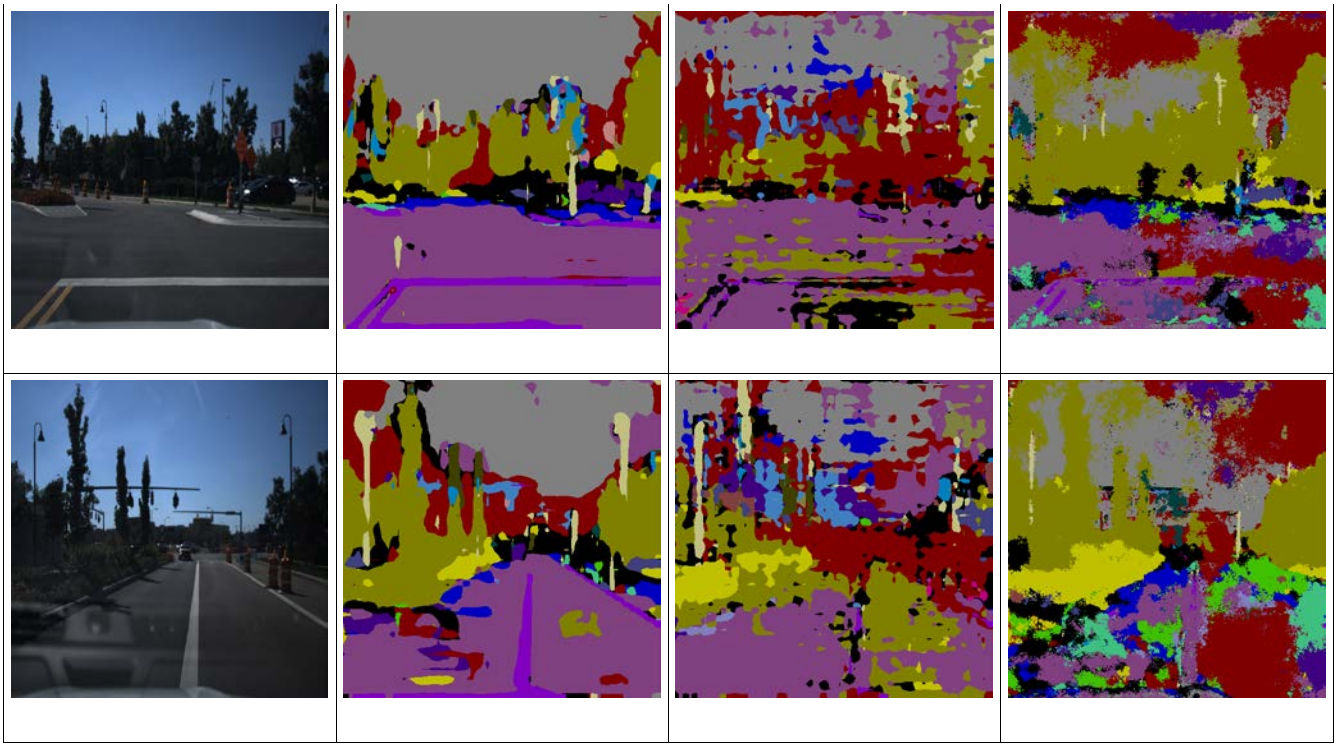
Table 4 Class-wise accuracy for BiSeNet, DenseASPP, and FC-DenseNet. Values are in percentage (%).

	Bicyclist	Building	Car	Fence	Pedestrian	Sidewalk	Sign	Sky	Tree	Road	Pole
BiSeNet	88.30	85.88	82.50	84.02	67.83	82.43	75.02	95.14	84.46	91.75	51.99
DenseASPP	83.64	77.89	82.56	85.34	48.91	61.70	63.77	87.00	80.65	82.80	38.38
FC-DenseNet	88.18	86.50	79.57	81.85	58.90	84.41	61.87	96.12	86.42	91.58	38.90

	Cart	Child	Lane Markings	Motorcyclist	Parking	Traffic Light	Wall
BiSeNet	73.38	96.11	39.20	97.61	87.93	79.36	74.48
DenseASPP	70.59	93.79	32.83	97.64	90.17	71.73	61.22
FC-DenseNet	70.47	95.44	50.84	97.62	89.99	74.99	73.12

	Road shoulder	SUV	Bus	Other Moving Obj.	Vegetation	Void
BiSeNet	92.36	79.21	94.26	87.64	85.39	42.30
DenseASPP	92.67	67.39	92.23	84.07	80.73	41.52
FC-DenseNet	95.08	78.43	92.78	85.40	87.15	47.00

Input	BiSeNet	DenseASPP	FC-DenseNet
			
			
			



Decision Making

At the core of autonomous vehicle control system, the supervisory controller plays a very important role in determining the appropriate state and subsequent course of action when faced with different situations. Decision making has been a research topic that has drawn broad attention and great progress is being made with the various methods that are deployed.

To take care of the uncertainties in the environment, many research efforts in decision making have utilized the probabilistic method for errors that occur in the perception part of autonomous vehicle. Markov Decision Process is one of the most popular methods. A lot of work has been done in decision making based on Markov Decision Process or POMDP to deal with the uncertainties existing in perception and situational awareness. Also, since artificial intelligence is also commonly used and well developed, decision making based on machine learning has also been studied and implemented widely in autonomous driving.

Due to the heavy computation and uncertainties in the probabilistic method and machine learning method, the platform for supporting decision making has to have a powerful computer like the NVIDIA GPUs used in our hardware architecture for good performance. The rule-based decision-making framework like the simple example in Figure 33 is used in this report because of lower computation complexity and ease of implementation on most of the vehicle platforms we use. Work on replicable and scalable probabilistic or machine learning based decision making is work in progress.

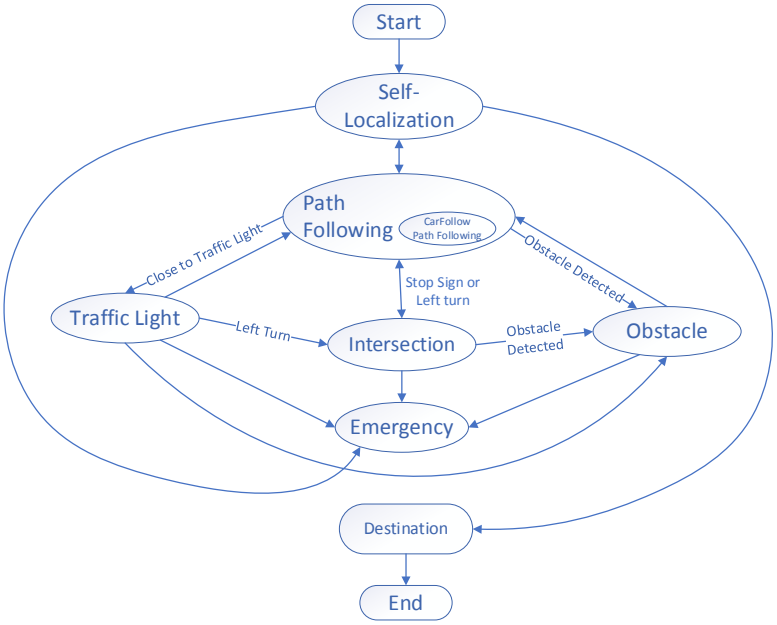


Figure 33. Diagram of Framework for Decision Making in the Unified Autonomous Vehicle Architecture with some of switching conditions shown.

Evaluation - Data Collection and Sharing

We used the Ford Fusion sedan to collect multiple sensor raw data for use in autonomous driving research. The data collection has resulted in eight datasets which we share in a box site and provide interested researchers with a publicly accessible link. We will continue these data collection and sharing exercises in more of the planned autonomous shuttle routes in Columbus, Ohio. The data collection covers areas including downtown COSI routes (Figures 34-36), Easton town center (Figure 37), road connecting CAR with CAR West (Figure 38), the parking lot around CAR West (Figure 39) center and some highway routes around Columbus (no figure display). The data is classified based on the area and sensors used for data collection. Further development will include data from other areas and data labelling for various ways of utilization.

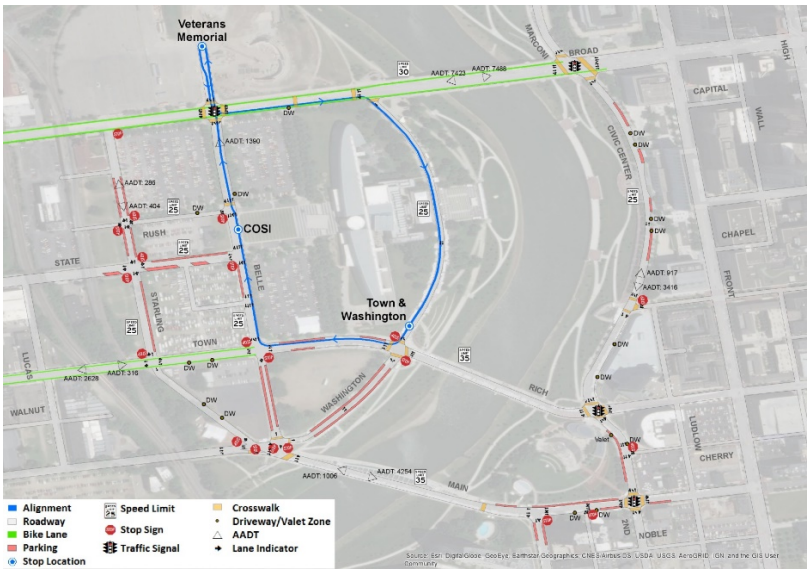


Figure 34. COSI Loop 1.

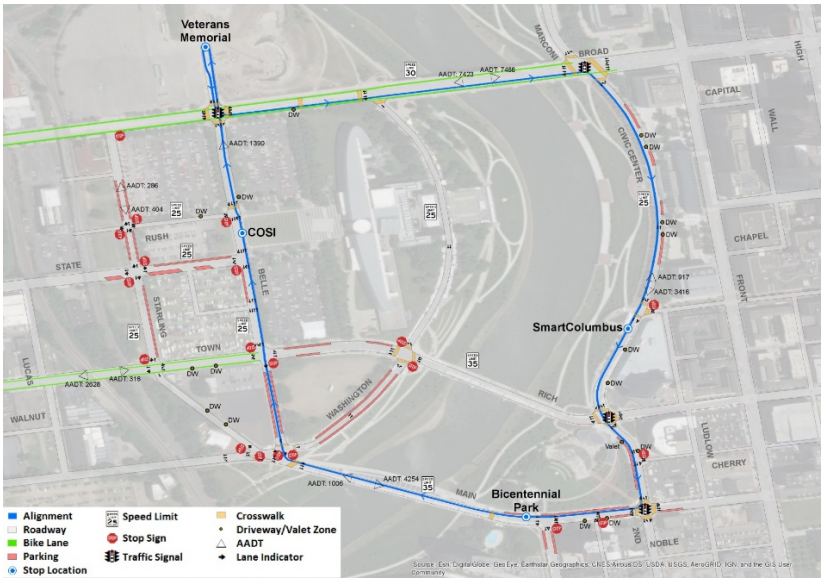


Figure 35. COSI Loop 2.

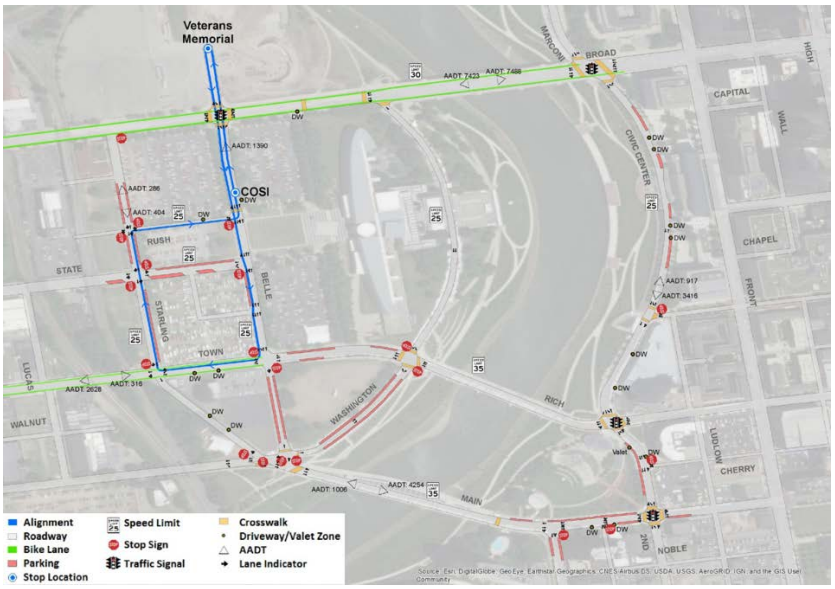


Figure 36. COSI Loop 3.

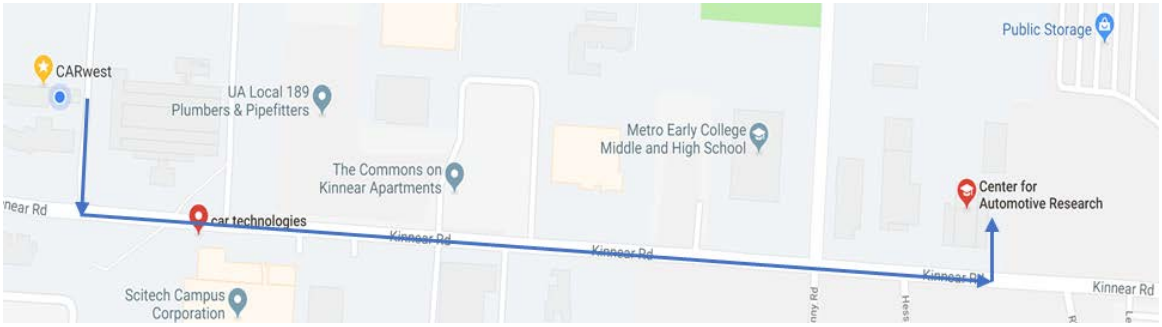


Figure 37. CAR to CAR West route.

Figure 38. Easton Town Center with position of road signs.

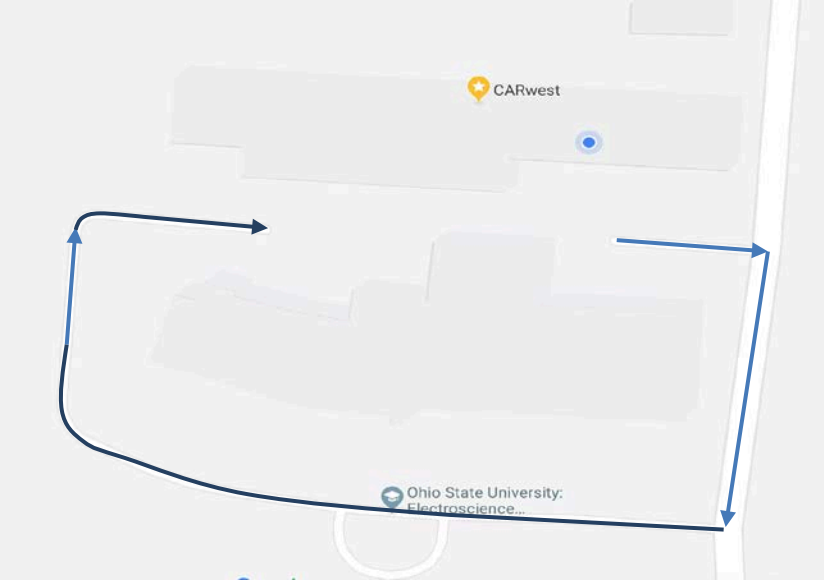


Figure 39. Parking lot at CAR West.

The sensor suite used includes Velodyne VLP-16 16 channel 3D Lidar, Ouster OS1-64 64 channel 3D Lidar, Pointgrey Camera with 30 fps and OXTS xNav 550 GPS/IMU data. From those sensors, different types of data are present in the dataset, which are, pointcloud data, images, GPS location data and IMU data. All the data are raw data stored in the format of a ROS bag file in order to make the data accessible to a wider audience. The rosbag play command can be used to playback the data. For the convenience of storing and using, we split the recording to consecutive files of 20 seconds each and used a sequential ordering starting from 0, meaning the first 20 seconds of data. The data is available for download at: https://osu.box.com/s/wl3ax8iywcciz0swzikdfbo7mzamrhey.

Evaluation - HiL Simulator

To extensively evaluate the performance of the developed low level controllers, along with both high level supervisory control, decision making and sensor placement, a hardware-in-the-loop simulator is employed. The HIL simulator is a platform where we can run several test scenarios in a realistic environment in real time. It also allows us to test our actual experimental hardware in the system. In the simulations, a high fidelity CarSim vehicle model is used to simulate the lateral and longitudinal motions of the vehicle alongside the soft perception sensors. While this model is running in the dSpace Scalexio HIL platform, a MicroAutobox controller, an in-vehicle PC with Nvidia GPU and DSRC modems emulate the actual control and communication scheme in this simulation environment. Control strategy and decision making were implemented inside the MicroAutobox while high fidelity vehicle dynamics model runs in real time on SCALEXIO. Traffic is added using other vehicles in Carsim and co-simulation with SUMO and PTV Vissim. The HIL simulator setup can be seen in Figure 40.

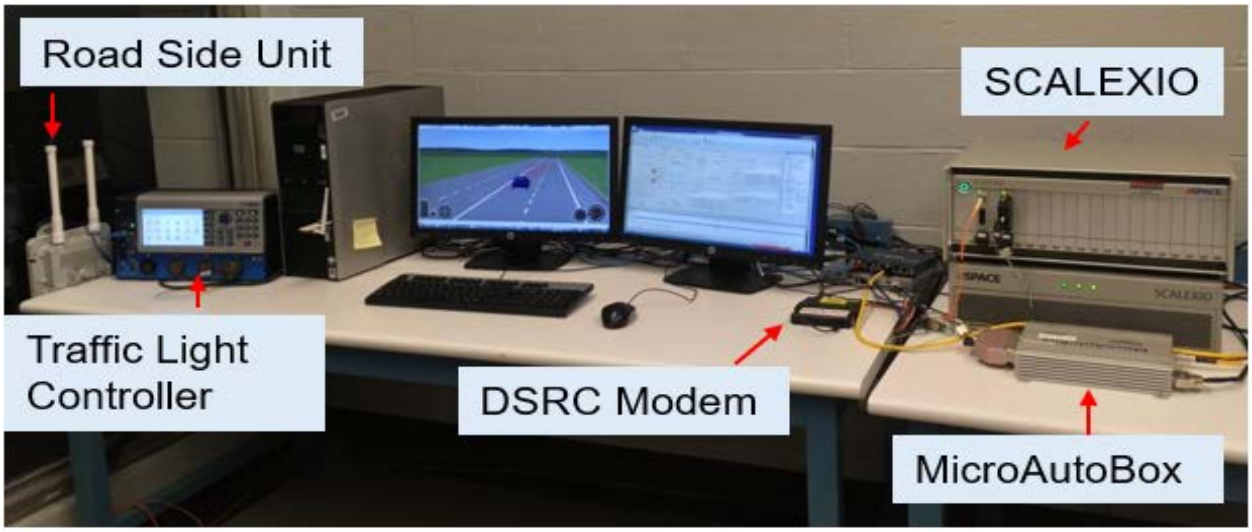


Figure 40. Hardware-in-the-Loop simulator setup.

The OSU AV pilot test route from CAR West (our lab location) to CAR (Center for Automotive Research – our main research center) shown in Figure 37 was chosen and constructed in CarSim to evaluate the vehicle's decision making and low level control performance. To incorporate the real traffic into simulation, information about other vehicles on the road were imported from SUMO software. Placement and field of view (FOV) of the sensors implemented as soft sensors in the HIL simulation can be seen in Figure 41.

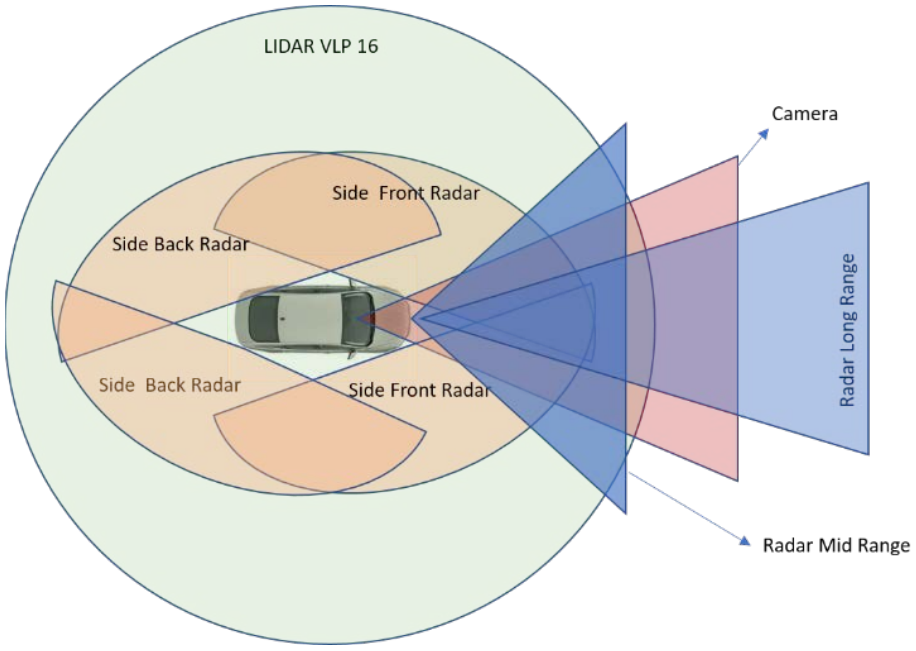


Figure 41. Placement and FOV of the sensors on the car.

Scenarios including stop sign, crossing traffic and traffic lights were simulated on the route in Figure 37 to test the decision-making strategies introduced in previous sections. Figure 42 shows the vehicle speed, steering and tracking error along the route. Since sharp turns appear when entering and exiting the main straight road, lateral error *e* was expectedly high for these two cases, but the look-ahead error *y* which combines the lateral and heading angle error was still relatively small. A video animation of this soft HIL AV test is present at: https://youtu.be/_yWiZWP0Rag.

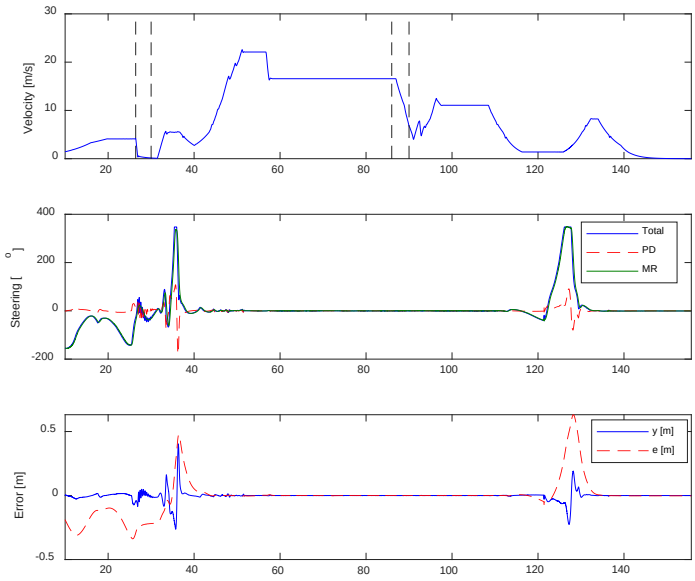


Figure 42. CARWest-to-CAR AV pilot test route HIL simulation results.

Evaluation – Scalability of Path Tracking Experiments

After the controller design, real world experiments were done for the purpose of analyzing the effectiveness of both controller parameters and overall unified approach. The path following algorithm developed for the sedan is scaled to the small electric vehicle and tested as it is driven autonomously on an open field. An oval path is generated for the vehicle to follow and loop two times in order to test both performance and repeatability. First path following experiment is done by using the lateral controller coefficients designed for the sedan vehicle without any change and it works except for the relatively large path tracking error as low speed is used. The second experiment is done with the controller coefficients re-tuned for the small electric vehicle. Results for both of the experiments are shown on the same plots in Figures 43 and 44 to make comparison easier.

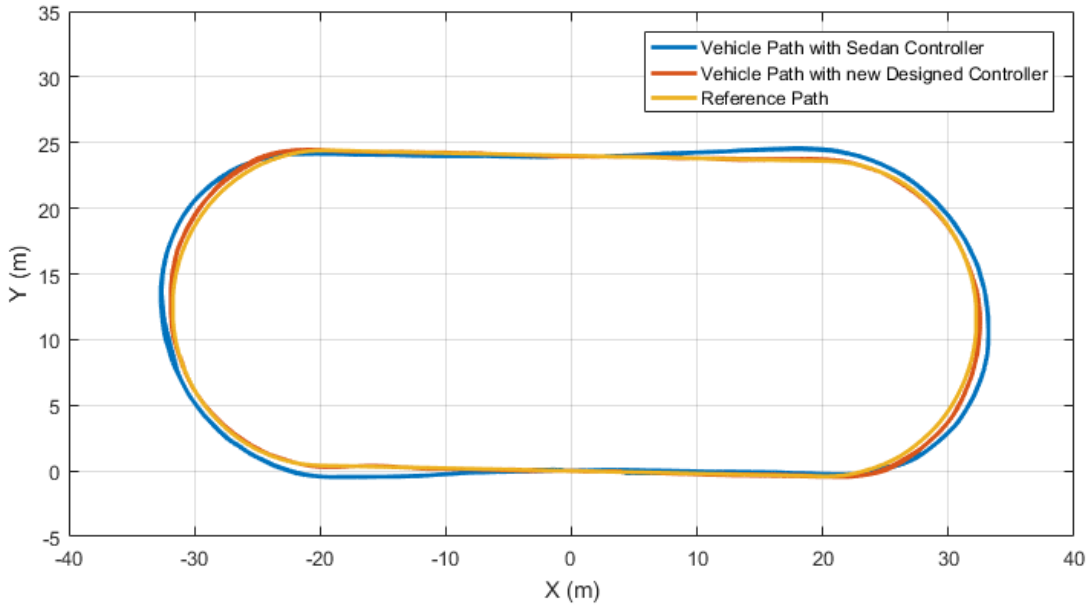


Figure 43. Path following comparison of Dash EV for two different controllers.

In Figure 43, we can see the reference path with comparison to the vehicle path while it is doing autonomous path following. We can also see two different controllers and how the performance is affected by changing coefficients according to the system parameters scaling down from one vehicle to other one. Sedan controller is not able to keep up with path when there is a curve and creates a significant error. There is also a very small error while it is following a straight road. Dash re-tuned controller on the other hand, is better in both of following a straight line or following a curved road. Moreover, the vehicle follows the same path for both its first and second lap in the experiments with both of the controllers, meaning repeatability is good for both of the controllers.

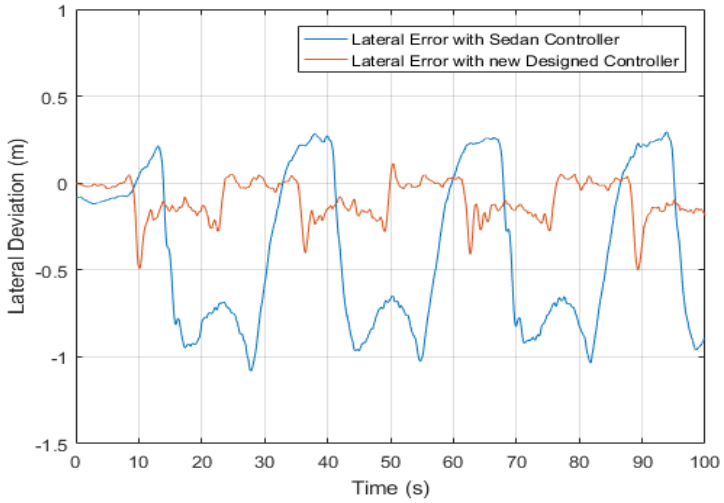


Figure 44. Lateral error comparison for two different controllers.

In Figure 44, we can see the lateral deviations in the course of approximately two laps of the path for both of the controllers. Lateral error from the experiment with the sedan controller yields higher peak values and has RMS value of 0.5636 whereas the other

controller yields much lower peak values and has an RMS error value of 0.1443, which is very close to the value of 0.121. This is similar performance to that of the sedan vehicle with the sedan controller. It should be noted that the re-tuning of the control gains was conducted using a robust parameter space design and took a very short time to formulate and execute. A video of the Dash EV vehicle during this path tacking run is available at: https://youtu.be/WQjOZoPdQh4. The collision avoidance maneuvering application was also scaled successfully from the sedan vehicle to the Dash EV vehicle.

Evaluation – Proof of Concept Testing On-Demand Shuttle

A limited scale proof-of-concept demonstration of on-demand AV shuttle use as a first mile / last mile solution was conducted during the first year of the project in the parking lot around our lab. The video of this demonstration is available at: https://youtu.be/K9dCd4ofYxA.

Publications

The SmartShuttle sub-project resulted in the following publications that acknowledge project support. Copies of the papers (published ones) in this list are appended to the end of this report.

Journal Papers

1. Gelbal, S.Y., Aksun-Guvenc, B., Güvenç, L., "Elastic Band Collision Avoidance of Low Speed Autonomous Shuttles with Pedestrians," International Journal of Automotive Technology, under review.
2. Zhu, S., Gelbal, S.Y., Aksun-Guvenc, B., Güvenç, L., "Parameter-space Based Robust Gain-scheduling Design of Automated Vehicle Lateral Control," IEEE Transactions on Vehicular Technology, *under review*.
3. Bowen, W., Gelbal, S.Y., Aksun-Güvenç, L., Güvenç, L., 2018, "Localization and Perception for Control and Decision Making of a Low Speed Autonomous Shuttle in a Campus Pilot Deployment," SAE International Journal of Connected and Automated Vehicles, paper number 12-01-02-0003, pp. 53-66.

Conference Papers

1. Cantas, M.R., Gelbal, S.Y., Aksun-Güvenç, B., Güvenç, L., 2019, "Cooperative Adaptive Cruise Control Design and Implementation," WCX19: SAE World Congress Experience, April 9-11, Detroit, Michigan, Session AE 504 Intelligent Transportation Systems, SAE Paper Number 2019-01-0496.
2. Wang, H., Güvenç, L., 2019, "Discrete-Time Robust PD controlled system with DOB/CDOB compensation for high speed autonomous vehicle path following," WCX19: SAE World Congress Experience, April 9-11, Detroit, Michigan, Session AE 501 Intelligent Vehicle Initiative, SAE Paper Number 2019-01-674.
3. Li, Xinchen, Zhu, S., Gelbal, S.Y., Cantas, M.R., Aksun-Güvenç, B., Güvenç, L., 2019, "A Unified, Scalable and Replicable Approach to Development,

Implementation and HIL Evaluation of Autonomous Shuttles for Use in a Smart City," WCX19: SAE World Congress Experience, April 9-11, Detroit, Michigan, Session AE 504 Intelligent Transportation Systems, SAE Paper Number 2019-01-0493.

4. Sheng, Zhu, Gelbal, S.Y., Aksun-Güvenç, B., 2018, "Online Quintic Path Planning of Minimum Curvature Variation with Application in Collision Avoidance, IEEE Intelligent Transportation Systems Conference, Hawaii, pp. 669-676.
5. Sheng, Zhu, Gelbal, S.Y., Li, X., Cantas, M.R., Aksun- Güvenç, B., Güvenç, L., 2018, "Parameter Space and Model Regulation based Robust, Scalable and Replicable Lateral Control Design for Autonomous Vehicles," 57th IEEE Conference on Decision and Control, Miami Beach, Florida, pp. 6963-6969.
6. Cantas, M.R., Güvenç, L., 2018, "Camera and GPS Fusion for Automated Lane Keeping Application," WCX18: SAE World Congress Experience, April 10-12, Detroit, Michigan, Active Safety: Systems and Sub Systems, SAE Paper Number 2018-01-0608.
7. Wang, H., Güvenç, L., 2018, "Use of Robust DOB/CDOB Compensation to Improve Autonomous Vehicle Path Following Performance in the Presence of Model Uncertainty, CAN Bus Delays and External Disturbances," WCX18: SAE World Congress Experience, April 10-12, Detroit, Michigan, Intelligent Vehicle Initiative, SAE Paper Number 2018-01-1086.
8. Bowen, W., Gelbal, S.Y., Aksun-Güvenç, L., Güvenç, L., 2018, "Localization and Perception for Control and Decision Making of a Low Speed Autonomous Shuttle in a Campus Pilot Deployment," WCX18: SAE World Congress Experience, April 10-12, Detroit, Michigan, Driver Assistance Systems: Algorithms, Applications and Electronic Sensing, SAE Paper Number 2018-01-1182.

Conclusions and Recommendations

The final project report for the SmartShuttle sub-project of the Ohio State University has been presented in this report. SmartShuttle was a two year project where the unified, scalable and replicable automated driving architecture introduced by the Automated Driving Lab of the Ohio State University was further developed, replicated in different vehicles and scaled between different vehicle sizes. The project approach, some of the results obtained and links to some videos and raw datasets were presented in the report. The readers are referred to our publications for more detailed information.

References

[1] Gelbal, Şükrü Yaren, Mustafa Ridvan Cantaş, Santhosh Tamilarasan, Levent Güvenç, and Bilin Aksun-Güvenç. "A connected and autonomous vehicle hardware-in-the-loop simulator for developing automated driving algorithms." In Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on, pp. 3397-3402. IEEE, 2017.

[2] “ODOT's VRS RTK Network,” [Online]. Available: http://www.dot.state.oh.us/Divisions/Engineering/CaddMapping/Survey/Pages/VRS RTK-.aspx.

[3] P. Biber and W. Strasser, "The normal distributions transform: a new approach to laser scan matching," Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453), Las Vegas, NV, USA, 2003, pp. 2743-2748 vol.3. doi: 10.1109/IROS.2003.1249285.

[4] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda and T. Hamada, "An Open Approach to Autonomous Vehicles," in IEEE Micro, vol. 35, no. 6, pp. 60-68, Nov.-Dec. 2015. doi: 10.1109/MM.2015.133.

[5] PJ Besl, ND McKay, “Method for registration of 3-D shapes,” Sensor Fusion IV: Control Paradigms and Data Structures, vol.1611, pp.586-607.

[6] S. Kato et al., "Autoware on Board: Enabling Autonomous Vehicles with Embedded Systems," 2018 ACM/IEEE 9th International Conference on Cyber-Physical Systems (ICCPS), Porto, 2018, pp. 287-296. doi: 10.1109/ICCPS.2018.00035.

[7] A. Nüchter, K. Lingemann, J. Hertzberg, and H. Surmann, “6D SLAM—3D mapping outdoor environments,” Journal of Field Robotics, vol. 24, no. 8-9, pp. 699–722, 2007.

[8] G. J. L. Naus, R. P. A. Vugts, J. Ploeg, M. J. G. van de Molengraft, and M. Steinbuch, “String-stable CACC design and experimental validation: A frequency-domain approach,” IEEE Trans. Veh. Technol., vol. 59, no. 9, pp. 4268–4279, Nov. 2010.

[9] D. Swaroop, J. K. Hedrick, C. C. Chien, and P. Ioannou, “A Comparision of Spacing and Headway Control Laws for Automatically Controlled Vehicles,” Veh. Syst. Dyn., vol. 23, no. 1, pp. 597–625, 1994.

[10] Ş. Y. Gelbal, M. R. Cantaş, S. Tamilarasan, L. Güvenç, and B. Aksun Güvenç, “A Connected and Autonomous Vehicle Hardware-in- the-Loop Simulator for Developing Automated Driving Algorithms,” in 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC) Banff Center, Banff, Canada, 2017.

[11] M. Treiber, A. Hennecke, and D. Helbing, “Congested traffic states in empirical observations and microscopic simulations,” Physical review E, vol. 62, no. 2, p. 1805, 2000.

[12] M.R. Cantas, S. Y. Gelbal, L. Guvenc, and B Aksun Guvenc. Cooperative adaptive cruise control design and implementation. Technical report, SAE Conference Paper, 2019.

[13] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[14] Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., & Bengio, Y. (2017). The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. Computer Vision and Pattern Recognition Workshops (CVPRW). IEEE.

[15] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. IEEE Conference on Computer Vision and Pattern Recognition, (pp. 4700-4708).

[16] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. IEEE Conference on Computer Vision and Pattern Recognition, (pp. 770-778).

[17] Yu, C., Wang, J., Peng, C., Gao, C., Yu, G., & Sang, N. (2018). Bisenet: Bilateral segmentation network for real-time semantic segmentation. Proceedings of the European Conference on Computer Vision (ECCV), (pp. 325-341).

[18] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. International Conference on Machine Learning, (pp. 448–456).

[19] Glorot, X., Bordes, A., & Bengio, Y. (2011). Deep sparse rectifier neural networks. International Conference on Artificial Intelligence and Statistics, (pp. 315–323).

[20] Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. IEEE Conference on Computer Vision and Pattern Recognition.

[21] Yang, M., Yu, K., Zhang, C., Li, Z., & Yang, K. (2018). DenseASPP for Semantic Segmentation in Street Scenes. IEEE Conference on Computer Vision and Pattern Recognition, (pp. 3684-3692).

[22] Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., & Yuille, A. L. (2018). Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, (pp. 834-848).

SAE INTERNATIONAL

ARTICLE INFO

Article ID: 12-01-02-0003

doi:10.4271/12-01-02-0003

Localization and Perception for Control and Decision-Making of a Low-Speed Autonomous Shuttle in a Campus Pilot Deployment

Bowen Wen, Sukru Yaren Gelbal, Bilin Aksun Guvenc, and Levent Guvenc,
Ohio State University, USA

Abstract

Future SAE Level 4 and Level 5 autonomous vehicles (AV) will require novel applications of localization, perception, control, and artificial intelligence technology in order to offer innovative and disruptive solutions to current mobility problems. This article concentrates on low-speed autonomous shuttles that are transitioning from being tested in limited traffic, dedicated routes to being deployed as SAE Level 4 automated driving vehicles in urban environments like college campuses and outdoor shopping centers within smart cities. The Ohio State University has designated a small segment in an underserved area of the campus as an initial AV pilot test route for the deployment of low-speed autonomous shuttles. This article presents initial results of ongoing work on developing solutions to the localization and perception challenges of this planned pilot deployment. The article treats autonomous driving with Real-Time Kinematic (RTK) GPS (Global Positioning Systems) with an inertial measurement unit (IMU), combined with simultaneous localization and mapping (SLAM) with three-dimensional light detection and ranging (LIDAR) sensor, which provides solutions to scenarios where GPS is not available or a lower cost, and hence lower accuracy GPS is desirable. Our in-house automated low-speed electric vehicle is used in experimental evaluation and verification. In addition, the experimental vehicle has vehicle to everything (V2X) communication capability and utilizes a dedicated short-range communication (DSRC) modem. It is able to communicate with instrumented traffic lights and with pedestrians and bicyclists with DSRC-enabled smartphones. Before real-world experiments, our connected and automated driving hardware-in-the-loop (HiL) simulator with real DSRC modems is used for extensive testing of the algorithms and the low-level longitudinal and lateral controllers. Real-world experiments that are reported here have been conducted in a small test area close to the Ohio State University AV pilot test route. Model-in-the-loop simulation, HiL simulation, and experimental testing are used for demonstrating the feasibility and robustness of this approach to developing and evaluating low-speed autonomous shuttle localization and perception algorithms for control and decision-making.

History

Received: 30 Apr 2018
Revised: 28 Jun 2018
Accepted: 08 Jul 2018
e-Available: 12 Nov 2018

Keywords

Smart shuttle, Automated driving, SLAM, Hardware-in-the-Loop, Path following

Citation

Wen, B., Gelbal, S., Guvenc, B., and Guvenc, L., "Localization and Perception for Control and Decision-Making of a Low-Speed Autonomous Shuttle in a Campus Pilot Deployment," *SAE Int. J. of CAV* 1(2):53–65, 2018, doi:10.4271/12-01-02-0003.

ISSN: 2574-0741
e-ISSN: 2574-075X

Introduction

For the sake of development of smart city, the Ohio State University has designated a small segment in an underserved area of the campus as an initial autonomous vehicle (AV) pilot test route for the deployment of SAE Level 4 low-speed autonomous shuttles. This article presents preliminary work toward proof-of-concept low-speed autonomous shuttle deployment in this AV pilot test route which extends from our research lab through a 0.7 mile public road with a traffic light intersection and low-speed traffic to our main research center. Our approach is to develop and test elements of this autonomous system in the private parking lot right next to our lab and in a realistic virtual replica of the AV pilot test route created within our hardware-in-the-loop (HiL) simulator environment. As we have already reported our work on GPS waypoint following based path tracking in our earlier articles, this article concentrates on light detection and ranging (LIDAR) simultaneous localization and mapping (SLAM)-based localization for path tracking, a simple decision-making logic for automated driving and experimental and simulation results.

SLAM as first proposed by Leonard and Durrant-Whyte [1] is used to build up maps of surrounding environment with the aid of sensors such as LIDAR sensor or camera while also estimating the position of a robot simultaneously. A reliable and accurate solution of SLAM problems lays the foundation for an autonomous navigation and control platform [2, 3]. During the last decade, highly effective SLAM techniques have been developed, and state-of-the-art two-dimensional laser SLAM algorithms are now able to have satisfactory performance in terms of accuracy and computational speed (e.g., GMapping [4] and Hector SLAM [5]). In addition, researchers have successfully extended SLAM applicable scenarios from indoor environment to outdoor environment for AV [6, 7]. Probabilistic map distributions over environment properties followed by Bayesian inference [8] increased robustness to environment variations and dynamic obstacles, which enabled the vehicle to autonomously drive for hundreds of miles in dense traffic on narrow urban roads. A fast implementation of incremental scan matching method based on occupancy grid map was introduced in [9] where data association was also applied to solve the multiple object tracking problems in a dynamic environment. Most of the previous work in the literature in SLAM methods has concentrated on the evaluation of localization performance, whereas SLAM is used and evaluated as part of an automated path following system here.

In this article, both SLAM- and GPS-based localizations are used for localization and path following. The SLAM system used is based on the Levenberg-Marquardt algorithm, and results are compared with the Hector SLAM method. First, a reasonable convergence criterion was provided for the solution to the Levenberg-Marquardt algorithm in contrast to the fixed iteration step setting implemented in Hector SLAM, enabling more accurate and reliable pose estimation when combined with an integrated control system for smooth and comfortable path following performance. LIDAR is the only sensor that this SLAM algorithm depends on, posing effective solutions to scenarios where GPS is not available or a lower-cost and, hence, lower-accuracy GPS is desirable. Both HiL simulations containing different traffic scenarios and relevant real-world experiments were conducted. Results were demonstrated and evaluated to prove the feasibility and robustness of this approach to low-speed autonomous shuttle localization and perception algorithms for control and decision-making. Related videos were also accessible online[1,2].

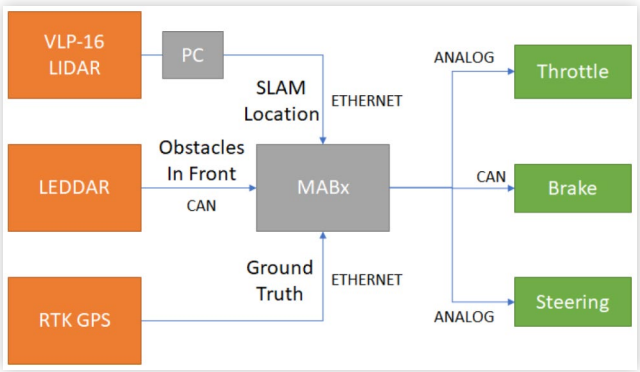
The article continues with an overview of the autonomous shuttle used in this study, the vehicle dynamics, and path tracking error models. The LIDAR SLAM algorithm and experimental GPS and SLAM-based path following results are presented next. This is followed by a description of the HiL simulator and how the AV test pilot route is replicated in the simulator including communication with the traffic light controller. After the experimental and simulation results, the article ends with conclusions and directions of ongoing work.

System Overview

Hardware and Platform

The vehicle used in the experiments for this study is a small, low-speed, fully electric two-seater shuttle used for ride sharing applications (Dash EV). The architecture and hardware presented in this article are general in nature and also implemented on other vehicles in our lab [10]. Architecture and connections are illustrated as a chart in Figure 1. In order to achieve autonomous driving capability, steering, throttle, and brake in this vehicle were converted to by-wire. This is done by adding actuators into the vehicle, since it was not built with them as some of the commercial sedan vehicles. For steering actuation, a smart motor was connected to the steering mechanism through gears. For brake actuation, a linear electric motor was fixed behind the brake pedal that pushes or pulls according to the position command. For

FIGURE 1 Platform architecture and connections.



[1] https://www.youtube.com/watch?v=LwrdPRxHzrg.
[2] https://www.youtube.com/watch?v=-O-H6KUAc8k.

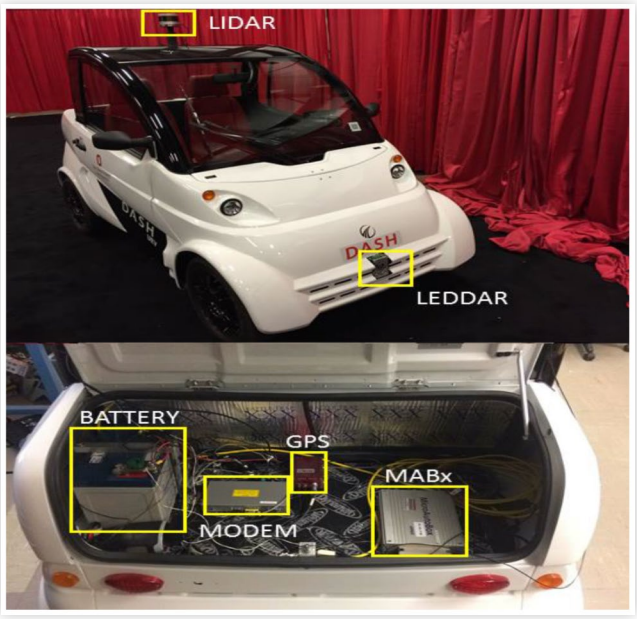
throttle, an electronic bypass circuit was constructed and used to override the throttle signal that is sent to vehicle electronic control unit (ECU) with the throttle command.

Sensors are added for localization and environmental perception after steering, throttle, and brake functions are converted to drive-by-wire. These sensors are GPS, a LIDAR sensor, a Leddar sensor, and a Point Grey camera used in this article as a backup sensor. The Leddar sensor is a solid-state LIDAR which we use to get information about the obstacles in front of the vehicle. These obstacles can be vehicles, pedestrians, bicyclists, etc. It is mainly used for emergency purposes, when there is an obstacle very close to the vehicle which creates a need to stop. It can be also used in low-speed car following applications such as adaptive cruise control (ACC) since its range is 50 m. For localization, GPS and LIDAR sensors were used. We use a differential GPS with Real-Time Kinematic (RTK) correction capability, which provides about 2-5 cm accuracy when RTK correction signals are used. Also with the differential antennas, it provides heading information even while the vehicle is stationary. LIDAR is used for both localization with SLAM and perception. It is a 16 channel Velodyne LIDAR PUCK (VLP-16) which is mounted on the top of the vehicle horizontally to guarantee a horizontal field of view (FoV) of 360 degrees with vertical FoV of 30 degrees from the surrounding environment. A 3D point cloud is generated at a frequency of 10 Hz. Theoretically, the LIDAR's maximum detection range can reach up to 100 m depending on application, while in this work, detection range used for localization was set to 80 m to achieve satisfactory point cloud density and quality. A separate computer is used for processing the point cloud generated by LIDAR sensor to obtain location by SLAM. Computer specifications are mentioned in Real-World Experiments section.

The element between the actuators and sensors is the dSPACE MicroAutoBox (MABx) ECU that is used for rapid prototyping of the low-level lateral and longitudinal direction controllers and basic decision-making algorithms created as Simulink models. Simulink coder is used to convert the model into embedded code, and the code is uploaded to the MABx device. The generated code can later be easily embedded in a series production level ECU at the end of the research and development phase.

Sensors send data to the MABx ECU with a means of communication specific to the sensor, like CAN or User Datagram Protocol (UDP) for most of our sensors. This data is fed to controllers running within the device. Controllers are created in the Simulink, and outputs of the controllers are connected to output blocks that correspond to input/output (I/O) ports of the MABx. These I/O ports are physically connected to actuators or drivers of actuators to provide reference signal and achieve autonomous driving. The experimental vehicle also has a dedicated short-range communication (DSRC) modem to communicate with other vehicles, infrastructure, and pedestrians with DSRC-enabled smartphones. For V2X communication, all messages are sent using the standard messages of the Society of Automotive Engineers (SAE) J2735 DSRC Message Set and use the standard communication rate of 10 Hz. Devices and actuators are powered through a 12V battery placed in the trunk of the vehicle. Some of the hardware discussed in this section is shown in Figure 2.

FIGURE 2 Hardware on the vehicle.



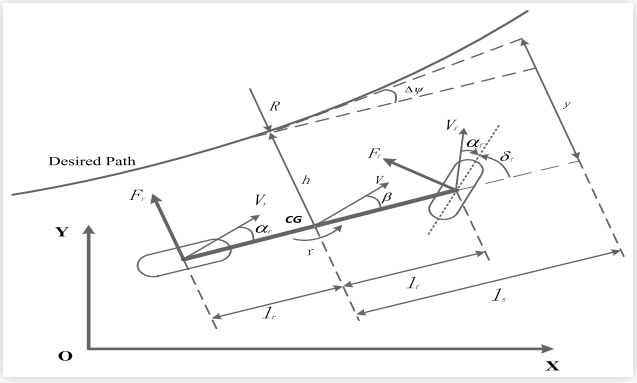
Vehicle Dynamics Model

The vehicle model and path following algorithm used are presented briefly in this and the following section. The lateral dynamics and path tracking error model is illustrated in Figure 3 and given in state space form as

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \Delta\dot{\psi} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ V & l_s & V & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ \Delta\psi \\ y \end{bmatrix} + \begin{bmatrix} b_{11} & 0 \\ b_{21} & 0 \\ 0 & -V \\ 0 & l_s V \end{bmatrix} \begin{bmatrix} \delta_f \\ \rho_{ref} \end{bmatrix} \quad \text{Eq. (1)}$$

where β is side slip angle, r is yaw rate, V is combination of lateral and longitudinal velocity of the vehicle body, $\Delta\Psi$ is

FIGURE 3 Illustration of single track model.



yaw angle relative to the tangent of the desired path, l_s is the preview distance, and y is lateral deviation from desired path with respect to preview distance. The control input is the steering angle δ_f. $\rho_{ref}=1/R$ is the road curvature where R is the road radius. Other terms in the state space model are

$$a_{11}=-\frac{C_r+C_f}{mV},\ a_{12}=-1+\frac{C_r l_r-C_f l_f}{mV^2}$$
$$a_{21}=-\frac{C_r l_r-C_f l_f}{J},\ a_{22}=-\frac{C_r l_r^2-C_f l_f^2}{JV^2} \qquad \text{Eq. (2)}$$
$$b_{11}=\frac{C_f}{mV},\ b_{12}=\frac{C_f l_f}{J}$$

where m is the vehicle mass, J is the moment of inertia, μ is the road friction coefficient, C_f and C_r are the cornering stiffnesses, l_f is the distance from the center of gravity (CG) of the vehicle to the front axle, and l_r is the distance from the CG to the rear axle.

Path Tracking Model

The low-level automated driving tasks are lateral and longitudinal control. The path determination and path tracking error computation are described briefly in this section. The path tracking model consists of two parts, which are offline generation of the path and online calculation of the error according to the generated path. These parts are explained in following subsections.

A. Offline Path Generation The path following algorithm employs a predetermined path to be provided to the AV to follow [11]. This map is generated from GPS waypoints where these points can be pulled from an online map or can be collected through recording during a priori manual driving. These data points are then divided into smaller groups named segments with equal number of data points for ease of the formulation. These segments are both used for curve fitting and velocity profiling through the route. After dividing the road into segments, a process of fitting a third-order polynomial is performed as

$$X_i(\lambda)=a_{xi}\lambda^3+b_{xi}\lambda^2+c_{xi}\lambda+d_{xi}$$
$$Y_i(\lambda)=a_{yi}\lambda^3+b_{yi}\lambda^2+c_{yi}\lambda+d_{yi} \qquad \text{Eq. (3)}$$

where i represents the segment number and terms a, b, c, d are polynomial fit coefficients for the corresponding segment. Fitting the data points provides effective replication of the curvature that the road carries and also eliminates the noise in the GPS data points. To provide a smooth transition from one segment to another by satisfying continuity of the polynomials and their first derivatives in X and Y, we use

$$X_i(1)=X_{i+1}(0)$$
$$Y_i(1)=Y_{i+1}(0) \qquad \text{Eq. (4)}$$

The X and the Y points derived from the GPS latitude and longitude data using a degree to meter conversion are fit using a single parameter λ, where λ is the variable for the fit which varies across each segment between 0 and 1, resulting in

$$\frac{dX_i(1)}{d\lambda}=\frac{dX_{i+1}(0)}{d\lambda}$$
$$\frac{dY_i(1)}{d\lambda}=\frac{dY_{i+1}(0)}{d\lambda} \qquad \text{Eq. (5)}$$

B. Error Calculation After the generation of path coefficients, an error is calculated for the lateral controller to use as input. Heading and position of the vehicle are provided by means of localization, in this case either SLAM or GPS. Using these, the location of the car with respect to the path, in other words the deviation from the path, is calculated. This approach reduces both oscillations and steady-state lateral deviation compared to calculation with respect to position only. In order to find an equivalent distance parameter to add to the first component distance error, a preview distance ls is defined. Then, the error becomes

$$y=h+l_s\sin(\Delta\psi) \qquad \text{Eq. (6)}$$

where $\Delta\psi$ is the net angular difference of heading of the vehicle from the heading tangent to the desired path and y is the total error of the vehicle computed at preview distance ls as is illustrated in Figure 4.

Finally, error is fed to a robust PID controller which controls the actuation of steering of the vehicle.

SLAM Algorithm

The SLAM-based localization algorithm is presented in this section. In this study, ground plane is always assumed to be flat, and hence only 2D mapping and localization are required, while z direction pose information in Cartesian coordinate system is not necessarily considered. In the following algorithm, the pose state vector $(x, y, \theta)^T$, comprised

FIGURE 4 Illustration of error calculation.

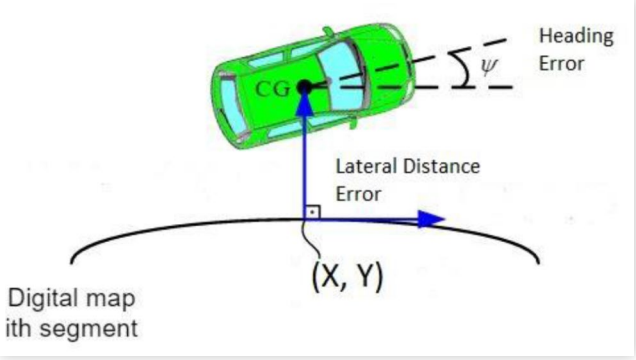
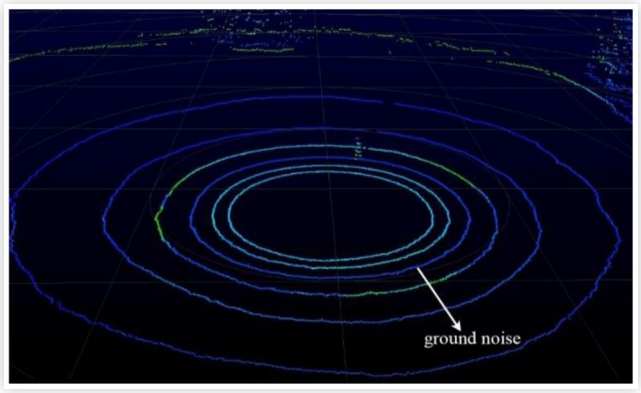


FIGURE 5 Raw 3D point cloud with ground noise.



of 2D Cartesian coordinates and orientation angle, and thus three degrees of freedom (DOF), is used to represent the pose information for the low-speed autonomous shuttle. As has been presented, the 16 channel Velodyne LIDAR can provide 3D point cloud including 360 degree FoV information of the surrounding environment. However, in this context, considering the constraint of the processor in this configuration, additional computational complexity will negatively affect the whole system in terms of real-time performance. Therefore, so as to obtain planar scan information, 3D point cloud is projected into 2D space.

Before the projection, ground noise as seen in Figure 5 needs to be removed by building up occupancy height map (section A). Once the planar scan end points are obtained, scan matching process is used to align the current scan end points either to those in the last frame or to the built-up map in order to derive the pose transformation of the shuttle. A more reliable and accurate optimization framework inspired by Hector SLAM [5] is imposed for the scan matching process, where more reasonable stop criteria are also introduced (section B).

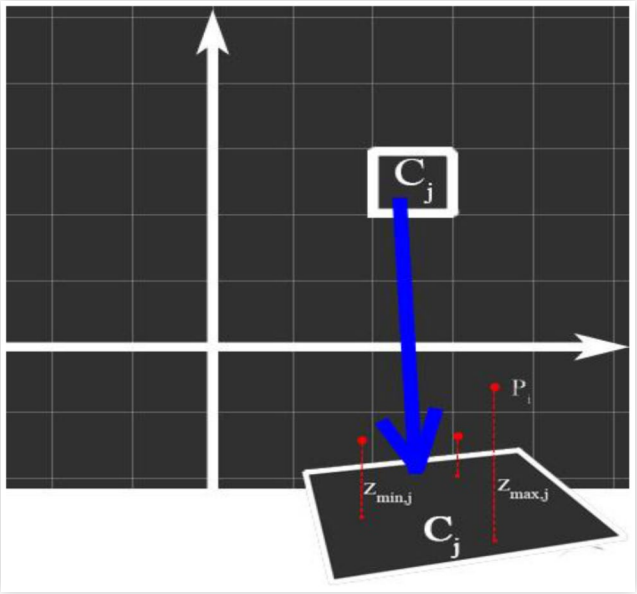
A. Ground Noise Removal and Projection Occupancy height map is built up for ground noise removal. The LIDAR position is selected as the origin, and the Cartesian coordinate system is built with the *x*-*y* plane representing the ground plane and the *z*-axis being vertical to it. As shown in Figure 6, from a top-down view, we divide the x-y plane into many square cells of equal size. In this work, cell size is set to 0.2 m × 0.2 m. For each of the 3D points $P_i = (x_i, y_i, z_i)^{\mathrm{T}}$, we can find a cell C_j that it belongs to. Subsequently, for each of the cells C_j by comparing the heights of the points to a threshold h_{thres} (set to 0.3 m in this work), if

$$z_{\max,j} - z_{\min,j} \le h_{thres} \qquad \text{Eq. (7)}$$

then this cell is defined as not occupied or comprised of ground noise and thus left as empty. If

$$z_{\max,j} - z_{\min,j} \ge h_{thres} \qquad \text{Eq. (8)}$$

FIGURE 6 Occupancy height map. C_j is one of the cells. Height of every cell is determined by the maximum height difference in that cell.



then this cell is defined as occupied and all the 3D points included in it are remained for further projection.

In the projection step, polar coordinate system is used to represent the position of each scan end point in 2D plane. For each 3D point P_i, its angular position in *x*-*y* plane can be expressed as

$$\alpha_i = atan2(y_i, x_i) \qquad \text{Eq. (9)}$$

where *atan*2 is four-quadrant inverse tangent and hence $\alpha_i \in [-\pi, \pi]$. The range of the 2D scan corresponding to the 3D point P_i can be expressed as

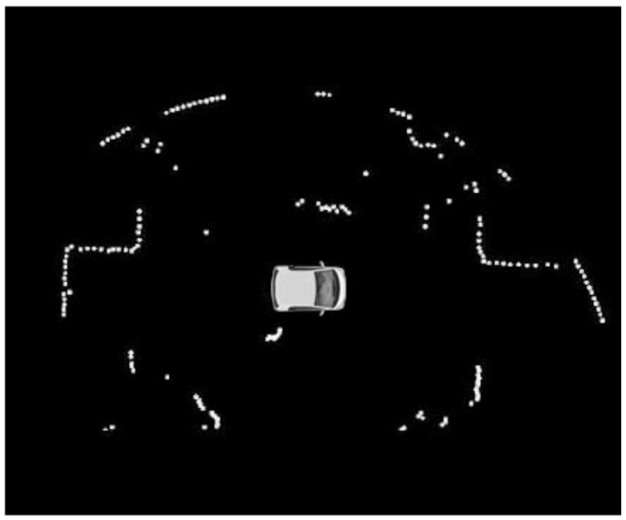
$$range_i = \sqrt{x_i^2 + y_i^2} \qquad \text{Eq. (10)}$$

Note that there can be more than one projected 2D scan point in the same direction with different ranges. The ultimate range of 2D scan end point is the smallest range in that direction. Therefore, every projected 2D scan beams with their associated scan end points can be identified by angular positions, as shown in Figure 7.

B. Map Generation and Scan Matching In this work, the same map access methodology as [5] is employed, which can provide an effective solution to the accuracy limitation caused by discrete property of occupancy grid maps.

Due to the high accuracy and frequency of modern LIDAR, iterative optimization algorithms are now possible to minimize the error between obtained scan end points and built-up maps, delivering the optimal alignment in the scan matching step. In this work, instead of Gauss-Newton optimization performed in Hector SLAM [5], the Levenberg-Marquardt algorithm [12] is applied to provide faster convergence for the same accuracy compared with Gauss-Newton

FIGURE 7 Projected 2D scan end points.



optimization, which can tremendously benefit the real-time system on autonomous shuttles. Given the generated map occupancy value $M(P_m)$ corresponding to the continuous map point location $P_m = (x_m, y_m)^T$, our goal is to find the rigid transformation $\xi = (p_x, p_y, \theta)^T$ which minimizes the overall summation of occupancy error between the current scan end points and the most updated map; consequently, the objective function and desired rigid transformation can be defined as

$$E = \min \sum_{i=1}^{n} \left[1 - M\left(S_i(\xi)\right)\right]^2 \qquad \text{Eq. (11)}$$

$$\xi^* = \arg\min_{\xi^*} \sum_{i=1}^{n} \left[1 - M\left(S_i(\xi)\right)\right]^2 \qquad \text{Eq. (12)}$$

where n is the number of scan end points and $s_i = (s_{i,x}, s_{i,y})^T$ is the world coordinate of the transformed scan end point. $S_i(\xi)$ is a function of ξ that transforms scan end point coordinate into world system, expressed as

$$S_i(\xi) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} s_{i,x} \\ s_{i,y} \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix} \qquad \text{Eq. (13)}$$

and $M(S_i(\xi)) \in [0,1]$ is the occupancy value at the location given by $S_i(\xi)$. Once this is performed, the optimal transformation that best aligns the current frame with the most updated map points is obtained.

This quadratic cost function E can be solved by Levenberg-Marquardt algorithm [13] efficiently. Starting from an initial estimation of the transformation, for example, the optimal transformation provided in last frame, ξ_0, in every iteration, a transformation update $\Delta\xi$ is added to the accumulated transformation so far, ξ, so as to move forward to the minimum point and further minimize the function. Intuitively, by each iteration step, the cost function is closer to 0:

$$E = \sum_{i=1}^{n} \left[1 - M\left(S_i(\xi + \Delta\xi)\right)\right]^2 \to 0 \qquad \text{Eq. (14)}$$

By replacing $M(S_i(\xi + \Delta\xi))$ with its Taylor series expansion, we obtain

$$E \approx \sum_{i=1}^{n} \left[1 - M\left(S_i(\xi)\right) - \nabla M\left(S_i(\xi)\right) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi\right]^2 \to 0 \qquad \text{Eq. (15)}$$

By letting the partial derivative with respect to $\Delta\xi$ equal to 0

$$\frac{\partial E}{\partial(\Delta\xi)} = 2\sum_{i=1}^{n} \left[\nabla M\left(S_i(\xi)\right) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T \cdot \ldots \cdot \sum_{i=1}^{n} \left[1 - M\left(S_i(\xi)\right) - \nabla M\left(S_i(\xi)\right) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi\right] = 0 \qquad \text{Eq. (16)}$$

According to Levenberg-Marquardt algorithm, the optimal solution for $\Delta\xi$ can be determined by

$$\Delta\xi = \left(H^{-1} + \lambda I\right) \sum_{i=1}^{n} w_i \cdot \left[\nabla M\left(S_i(\xi)\right) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T \left[1 - M\left(S_i(\xi)\right)\right] \qquad \text{Eq. (17)}$$

where w_i is weight associated with point P_i, which mainly down weights the low-quality scan end points with big error and hence enhances robustness against noise [14]. λ is a damping parameter (initially set to 0.01 in this work), I is identity matrix, and H is weighted approximate Hessian matrix, defined by

$$H = w_i \cdot \left[\nabla M\left(S_i(\xi)\right) \frac{\partial S_i(\xi)}{\partial \xi}\right]^T \left[\nabla M\left(S_i(\xi)\right) \frac{\partial S_i(\xi)}{\partial \xi}\right] \qquad \text{Eq. (18)}$$

By solving $\Delta\xi$, ξ is updated by

$$\xi \leftarrow \xi + \Delta\xi \qquad \text{Eq. (19)}$$

and that makes ξ iteratively move forward to the optimal transformation ξ^*.

In contrast to the practical implementation in Hector SLAM [5], where fixed iteration step setting is employed to evaluate the Gauss-Newton optimization, in addition to setting a maximum iteration step (10 in this work), we hereby propose a more reasonable stop condition before reaching the maximum iteration step, which has been proven to ensure sufficient convergence while avoiding unnecessary iterations caused by oscillation around the optimal solution:

$$\|\Delta\xi\| < \varepsilon \qquad \text{Eq. (20)}$$

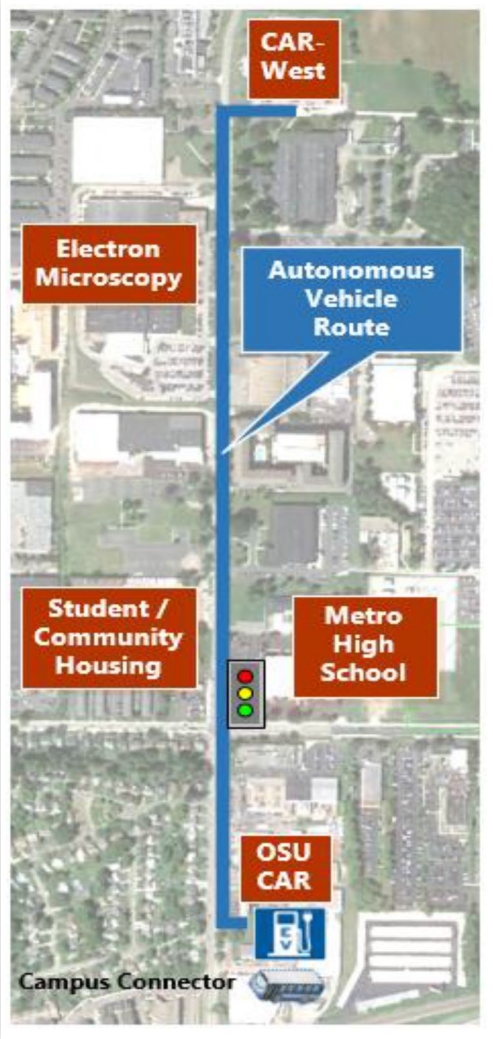
where operator $\|\cdot\|$ denotes Frobenius norm and ε is a parameter for threshold and is set to 0.001 in this work. E_k is the cost function in the kth iteration step.

Real-World Experiments

We conducted extensive experimental validations of our system including offline SLAM system test on collected data as well as real-time field experiment in the area around the initial AV pilot test route, a small segment in an underserved area of the campus designated by the Ohio State University, as shown in Figure 8. All the algorithms relevant to LIDAR data processing and SLAM as described above are implemented in C++ because of its efficiency of real-time performance. Performances are evaluated between the SLAM system proposed in [5] and the extended version proposed in this article. Traditional path following experiment result based on high-accuracy GPS similar to the previous work is compared with this innovative SLAM-based path following experiment result, demonstrating the feasibility and effectiveness of this compounded system. Note that randomness is inevitably introduced by probabilistic occupancy grid map model in the SLAM system. For this reason, the experiment results are reported based on the median performance of several runs.

Real-time SLAM algorithm is carried out with an I7-6700HQ (8 cores @ 2.60 GHz), NVIDIA Titan X (Pascal)/PCIe/SSE2 and 4Gb RAM on the Robot Operating System (ROS) [15], an open-source operating system providing services designed for heterogeneous computer cluster in Linux environment. UDP communication is built up between ROS and MABx for localization information transfer. Regional localization information delivered by SLAM algorithm is sent to MABx for further decision-making and control strategy, for example, longitudinal or lateral control.

FIGURE 8 AV test route from Car West to Center for Automotive Research (CAR) (scale 1:8000).



SLAM Evaluation

In order to quantitatively evaluate our proposed SLAM system against Hector SLAM, both SLAM systems are tested on the same LIDAR data collected around our lab, Car West. Due to the absence of "ground truth," alignment error yielded in both algorithms is reported for comparison. Ideally, with sufficient accuracy, the alignment error (described in Equation 11) should be very small. However, inevitably introduced sensor noise and non-smooth approximation of the optimization model make the solution of pose estimation only able to approach real pose but never perfectly equivalent, and hence total alignment error always exists. Therefore, in the same context, the smaller the alignment error, the higher the accuracy that is achieved, and hereby we evaluate the performance by comparing their alignment error and iterations implemented in each alignment, which can reflect their estimation accuracy as well as their convergence speed. Considering that offline SLAM accuracy is similar to its real-time accuracy, this comparison can effectively validate the overall performance of our proposed SLAM system against the Hector SLAM.

The ultimate map generated by our proposed SLAM system is overlapped with the same location obtained from Google Earth for comparison convenience as shown in Figure 9, where the map generated by our proposed SLAM is in shadow and red line is the test trajectory. It is important to note that the map

FIGURE 9 Generated map overlapped with Google Earth.

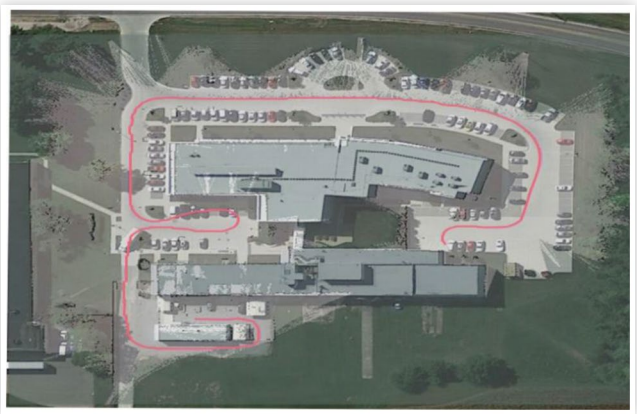
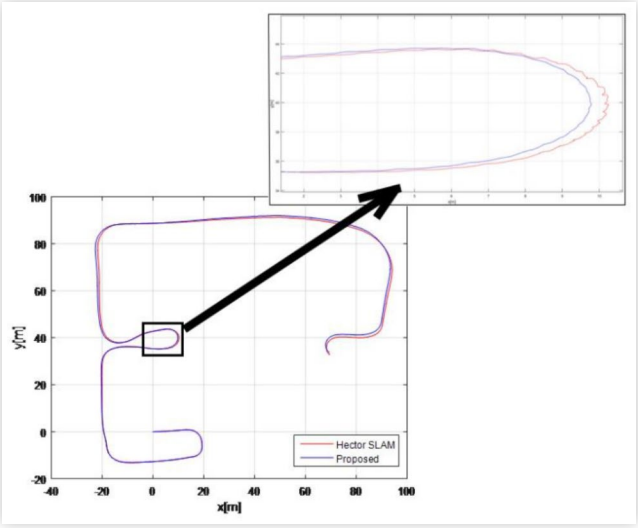


FIGURE 10 Trajectory comparison between our proposed SLAM (blue) and Hector SLAM (red).



from Google Earth is not strictly a top-down view. Thus, here a minor shift is necessarily used to keep the edges of the mapped buildings consistent with their actual corresponding edges in Google Earth. In this experiment, raw LIDAR data is initially collected by VLP-16 along the test trajectory which starts from the backyard of Car West, passing through an open field which is sufficiently challenging because of the limited landscapes for matching alignment and textureless wall. Another challenging part of this test trajectory is a sharp 180 degree turn in the front of the parking lot of the lab building, which demands fast convergence and robustness of the nonlinear optimization model implemented in the SLAM system.

Figure 10 shows both complete and regional localization estimation from the two SLAM systems along the test trajectory. The smoother localization given by our proposed SLAM system with the integrated automated drive control systems can dramatically improve passenger comfort while taking a ride in the shuttle. Table 1 illustrates the average alignment error and average iteration steps required between the two SLAM systems. It can be clearly observed that in some runs, our proposed SLAM can effectively reduce the alignment error to a relatively lower level despite the fact that in almost half of the runs the benefit is not distinct. Results of the average alignment error from Table 1 can further prove this property.

TABLE 1 Performance comparison between our proposed SLAM with Hector SLAM. The results are average values over 10 simulations. Alignment error is accumulated error of occupancy value, which is dimensionless. Number of iterations is number of iterations to converge.

	Proposed SLAM	Hector SLAM
Alignment error	78.759	84.107
Number of iterations	6.557	3.400

This can be attributed to the defect of this optimization-based SLAM system where global minimum cannot be guaranteed and scan end point outliers can inevitably introduce noise to the system. Therefore, a reliable preprocessing model of the scan end points is desired as an extension to this framework, which may be an interesting topic in future work. Although in our proposed SLAM system additional iteration steps are sacrificed for better alignment compared with Hector SLAM, in which the iteration step is set to a fixed value and naturally convergence cannot be guaranteed, the increased iteration step is still in an acceptable range for real-time performance according to our real-time experiments.

Real-Time SLAM Path Following Performance

In addition to quantitative evaluation of our proposed SLAM system, various real-world experiments are also conducted to validate its feasibility and adaptivity of integration with the control system. We first manually drive the shuttle along the predetermined trajectory around our lab building, as shown in Figure 11, to collect GPS points, from which the desired path is then generated for path following reference.

Figures 12 and 13 show the actual path following trajectory performed by our proposed SLAM system and RTK GPS separately compared with the desired path. The coordinate of starting position is set to the origin in the following plots for comparison convenience. It can be observed that similar to GPS, SLAM-based path following can be achieved comparable to GPS-based result, though with occasional minor error, which again proved the supplemental functionality of our proposed SLAM system in GPS not accessible cases. Figure 14 shows the root-mean-square error (RMSE) along the whole path following trajectory performed by SLAM compared with the same experiment setting but performed by differential GPS. The shuttle speed of both path following approaches is kept at an average value of 12 km/h. As can be seen from the experimental results, conventional path following that relies on highly accurate differential GPS has the expected performance with appropriate lateral controller design. The overall

FIGURE 11 Trajectory on satellite image.

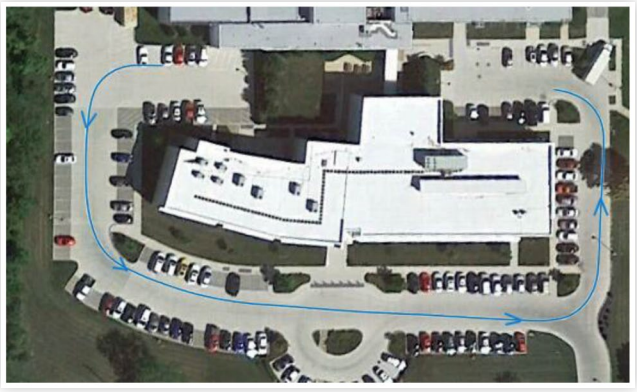


FIGURE 12 Desired path compared to our proposed SLAM path following trajectory.

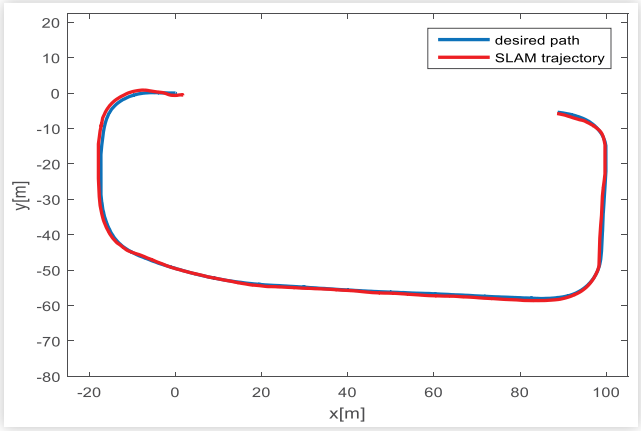


FIGURE 13 Desired path compared to GPS path following trajectory.

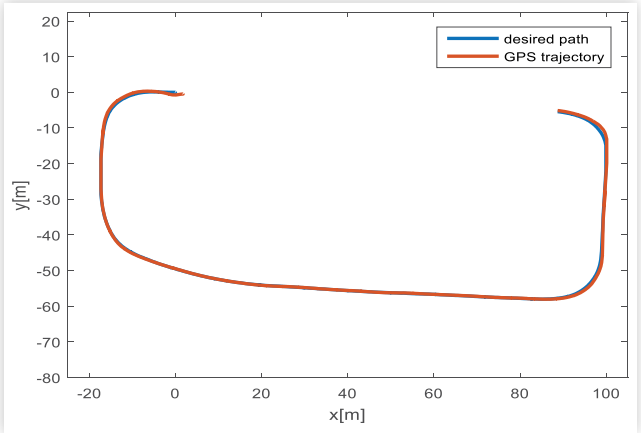
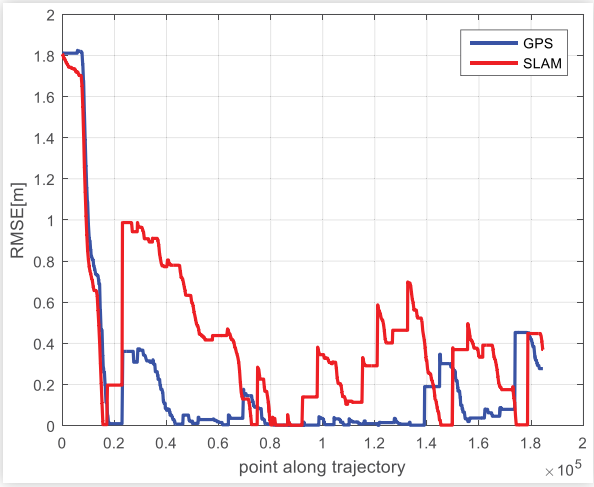


FIGURE 14 RMSE in lateral direction comparison between our proposed SLAM-based path following and GPS-based path following.



performance of GPS is better than SLAM, but SLAM-based path following tends to have even smaller RMSE at some regions, for example, at points of 0.7×10^5, 1.5×10^5, and 1.8×10^5 which are at the corners of the trajectory. The fact suggests that this SLAM system can provide precise estimation of the shuttle orientation while there may exist some delay or inaccuracy in the orientation angle provided by differential GPS, which is computed based on compass. It demonstrates that localization and perception system that purely relies on LIDAR can supplement the cases when GPS is not available or a lower cost, and hence lower accuracy GPS is desirable for intelligent shuttles.

HiL Studies

HiL setup is crucial for faster development of controllers and algorithms, since it provides a realistic virtual proving ground before the implementation and deployment phases. To create this realistic virtual proving ground, real-world scenarios should be replicated with as many aspects as possible. This includes emulation of sensors, addition of traffic, addition of hardware, and replication of real-world routes. For this article, the planned actual real-world AV shuttle deployment route is selected as a virtual proving ground.

Equipment and Setup

The HiL setup is constructed with hardware as close as possible to real-world case. Therefore, MABx is used as a main controller. This ECU is also the device we use in our AV as low-level controller, which is mentioned in the Hardware and Platform section. Since we already develop autonomous driving algorithms which run within this device during the HiL development, it allows us to directly implement the algorithms and controllers that we developed inside the HiL simulation to a real AV. MABx is also connected to a DSRC modem similar to the real-world case in the HiL simulator. Through this modem, it receives the V2X data that is published for the vehicles and infrastructure within the simulation. Again, similar to the real-world case, it is connected to the SCALEXIO computer which mimics the actual vehicle through the Controller Area Network (CAN) bus. The MABx thinks it is connected to a real vehicle while receiving the ego vehicle information from CAN bus and publishing actuation commands for steering, brake, and throttle through the CAN bus.

These commands are picked up by the SCALEXIO real-time vehicle, traffic, and sensor simulator. This simulator runs a Simulink model with CarSim vehicle dynamics. Vehicle model parameters inside CarSim are validated through vehicle dynamics experiments previously performed on the real vehicle. Therefore, vehicle dynamics simulation provides results very close to the real world. While simulating high-fidelity vehicle dynamics for ego vehicle, it can also simulate roads, sensors, and infrastructure through the capabilities of

FIGURE 15 HiL equipment and communication.

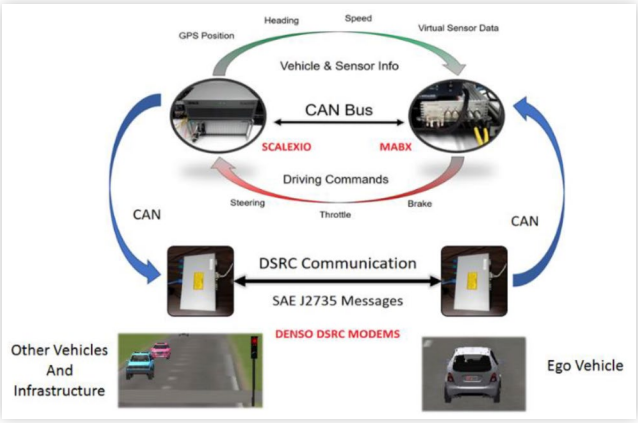
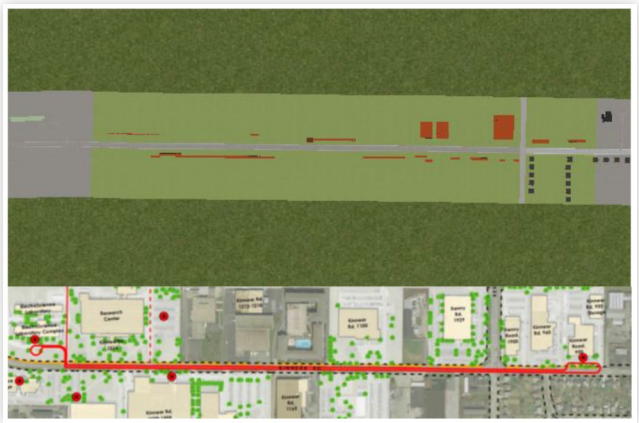


FIGURE 16 Top-down view of Car West-CAR AV test route.



CarSim. This feature provides significant advantage since it allows us to create numerous test scenarios which have applications in real-world autonomous driving. It is also connected to another DSRC modem that publishes V2X information for other vehicles and infrastructure that exist inside the simulation environment. All of the DSRC message packets are sent within a standard format obtained from SAE J2735 DSRC Message Set and using the standard communication rate of 10 Hz. Overall illustration of the HiL setup and communication between components are shown in Figure 15.

With this HiL setup, we are able to test numerous kinds of different scenarios involving other vehicles, pedestrians, and road structures, which involve V2X communication. Moreover, we are able to test our controllers and autonomous driving algorithms and do improvements on them before starting road testing.

In this study, the HiL setup discussed above is used to provide a virtual proving ground for algorithm and controller development before real-world deployment of the autonomous shuttle. A test scenario is created based on a planned real-world deployment route, which is explained within the next section, followed by discussion of the simulation results.

Test Scenario

A replication of the real-world route AV pilot test route was created inside CarSim for autonomous driving simulation. This route starts from the road in front of the parking lot of our research lab building (Car West) and ends about 0.7 miles down the road in front of our main research center (CAR). A traffic light is placed on the intersection and vehicle traffic is generated within CarSim for main route. Buildings are also created as a representation of real ones and placed according to their real-world positions. A top-down view of the road, which is rendered in CarSim, is shown in Figure 16.

The path to be followed is generated from the GPS points on the road, and vehicle is set to autonomously drive on this path, in other words, to follow the route while making decisions according to the situations it comes across during the drive. GPS and Leddar sensors are virtually simulated in CarSim software, while DSRC messages are received through real hardware. Therefore, the virtual simulation vehicle is equipped with a real DSRC radio, soft GPS, and a soft Leddar sensor. In this specific scenario, DSRC radio is mainly used for determination of the traffic light state in the intersection. Leddar sensor is utilized for detection of the distance between ego vehicle and preceding vehicle. Since LIDAR emulation is currently not available as a solution within CarSim, work is still in progress to emulate or simulate LIDAR sensor which provides a 3D point cloud data to simulate LIDAR-based algorithms such as SLAM in the simulator.

A. Decision-Making

The vehicle was commanded to follow the route while handling some of the situations it may come across. For this purpose, a simple decision-making strategy is created with three main states. This decision-making strategy is still work in progress and currently does not take all of the possible real-world cases into account. Instead, the scenario is slightly simplified with respect to real-world conditions in order to use a noncomplex decision-making strategy. These simplifications include the placement of the starting and end position onto the main road and removal of the intersection cross traffic. These simplifications will be removed in further study.

The developed decision-making strategy consists of three main states. In cruise control (CC) state, the vehicle is given a velocity profile to follow as a longitudinal control strategy. The vehicle follows the route while traveling at the desired speed which is decided by this velocity profile, according to the map segment the vehicle is currently in. With this velocity profile, the vehicle can slow down or speed up when necessary, according to the road portion it is currently in, and therefore can safely approach intersections, sharp curved turns, and traffic lights and obey traffic speed limits. While carrying out path following in CC state, it constantly checks for any DSRC messages. In case there is any traffic light nearby on path, according to the state of the light, it can go to stop state or continue. Furthermore, by making use of the Leddar sensor information, the vehicle can determine if there is a preceding vehicle, and according to the distance, it goes

FIGURE 17 Decision-making flowchart.

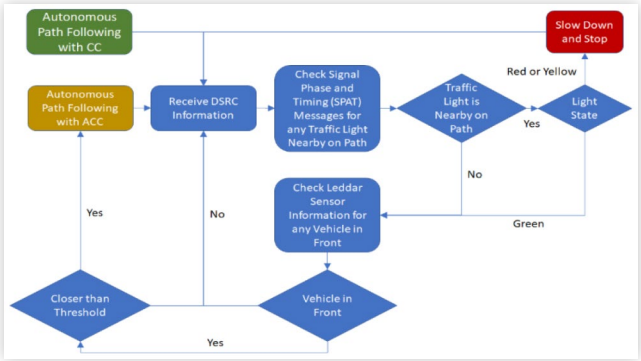
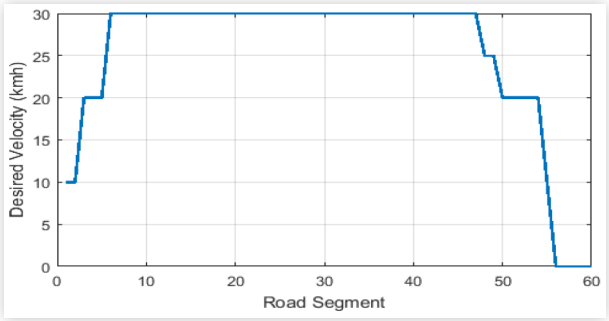


FIGURE 19 Velocity profile with respect to segment.



to ACC state or Cooperative Adaptive Cruise Control (CACC) state in the case of a communicating preceding vehicle for car following. In this state, the vehicle keeps a safe time gap with the preceding vehicle. The flowchart for the simple decision-making used is shown in Figure 17.

HiL Simulation Results

After route is constructed in the simulation environment, path following and decision-making algorithms that are explained in previous sections were implemented in HiL simulation environment. Route that is constructed in CarSim environment is shown in Figure 18 where *X* and *Y* scales are different for better visibility.

With the road constructed and algorithms implemented, a velocity profile was created for vehicle to follow when it is on CC mode, which is shown in Figure 19. Velocity profile is created according to the corresponding road segments where long straight road segments have higher speed and curves and intersections have lower speed. While following the velocity profile with longitudinal controller and following the path with lateral controller described in previous sections, vehicle also takes decisions according to the flowchart shown in Figure 17. For example, in case of any other vehicle coming in front, the vehicle goes to ACC mode to adapt the speed of preceding vehicle and keep the distance, disregarding the velocity profile.

With all the implementation and preparation of the simulation environment completed, HiL simulation was conducted to see the effectiveness of the overall structure. After the simulation, recorded vehicle velocity, vehicle decision state (stop/ACC/CC), and traffic light state (green/red) were plotted with respect to time as shown in Figure 20.

As seen in Figure 20, the vehicle follows the speed profile in CC mode while doing autonomous path following. After some time, starting around 90th second, also marked as gray ACC area, it comes across a non-communicating preceding vehicle which travels at a slower velocity. Instead of following the velocity profile, AV goes to ACC mode and slows down to adapt to the speed and keep the distance between itself and the preceding vehicle constant. This behavior can be confirmed by comparing the velocity profile (30 km/h) with vehicle velocity at that time phase (around 15 km/h). Around 125th second, it comes close to the intersection where there is a traffic light which is at red signal state and it stops. It waits until the light is green and then continues its way.

FIGURE 18 Car West to CAR route constructed in CarSim.

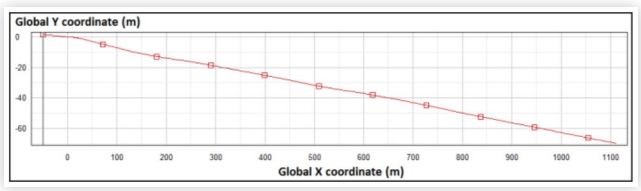


FIGURE 20 Vehicle velocity, behavior, and traffic light state with respect to simulation time.

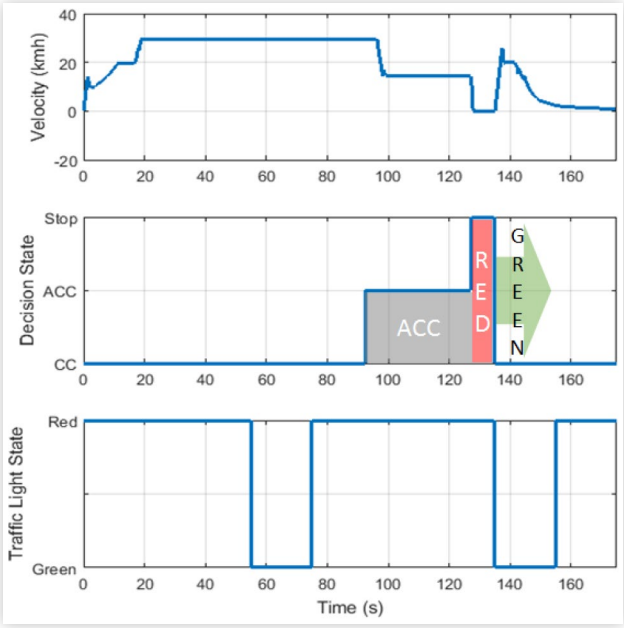
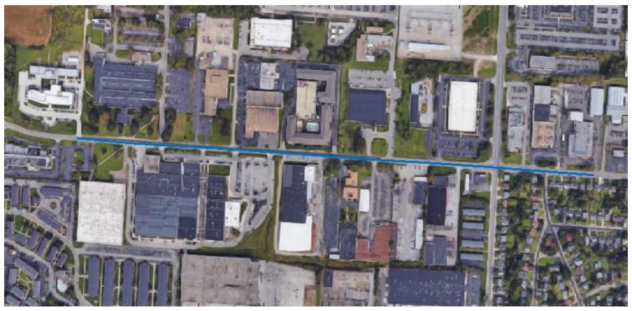


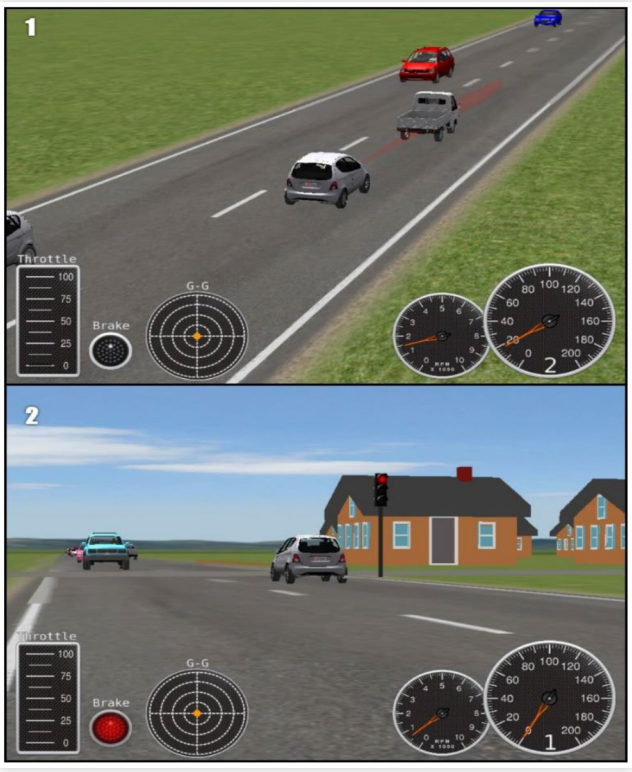
FIGURE 21 Vehicle trajectory on satellite image.



This behavior can also be confirmed by looking at the velocity graph and decision-making graph where it is marked as red and green areas and the traffic light state graph in Figure 20.

After passing the traffic light, it comes closer to the destination, slows down, and stops. The trajectory of the vehicle is also plotted on a satellite image and shown in Figure 21. It is seen that the vehicle is able to follow the route and autonomously handle dynamic driving tasks it can come across while traveling through this route. Some frames from the simulation are shown in Figure 22, while the vehicle is doing autonomous driving.

FIGURE 22 Simulation frames while the vehicle is doing ACC (1) and stopping at traffic light (2).



Summary/Conclusion

This article presented preliminary work for an AV shuttle deployment in the AV pilot test route of the Ohio State University. GPS and LIDAR SLAM are both used for localization and path generation. Since GPS-based localization and path following were presented in our earlier work, this article concentrated on a LIDAR SLAM system which is inherited from the Hector SLAM framework and based on the Levenberg-Marquardt algorithm. It was demonstrated that this LIDAR SLAM algorithm can be used for self-localization of our low-speed autonomous shuttle. Extensive experiments were conducted for offline SLAM performance evaluation as well as real-world experiments for path following in a parking lot for safety. The proposed SLAM system was compared with the state-of-the-art 2D SLAM approach especially in terms of scan alignment accuracy and seen to provide dynamically reasonable pose estimation. As a prerequisite to testing autonomous driving on the actual AV pilot test route, this route was replicated in our HiL simulator for developing and testing low-level controllers and decision-making logic. GPS and Leddar sensors, traffic, and the traffic light were emulated in the HiL simulator, while the low-level control ECU and the DSRC radios used for V2I and V2V communication were real hardware. LIDAR sensor emulation work is in progress and will allow us to implement LIDAR-based algorithms for both localization, for example, SLAM, and obstacle detection and classification within the HiL simulator.

Contact Information

Automated Driving Lab
930 Kinnear Rd
Columbus, OH, 43212
guvenc.1@osu.edu

Acknowledgments

This article is based upon work supported by the National Science Foundation under Grant No. 1640308 for the NIST GCTC Smart City EAGER project UNIFY titled: Unified and Scalable Architecture for Low Speed Automated Shuttle Deployment in a Smart City, by the U.S. Department of Transportation Mobility 21: National University Transportation Center for Improving Mobility (CMU) sub-project titled: SmartShuttle: Model Based Design and Evaluation of Automated On-Demand Shuttles for Solving the First-Mile and Last-Mile Problem in a Smart City and the Ohio State University Center for Automotive Research Membership Project titled: Use of OSU CAR Connected and Automated Driving Vehicle HiL Simulator for Developing Basic Highway Chauffeur and Smart City Autonomous Shuttle Algorithms. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

This article is an updated and significantly revised version of a presentation at WCX18, Detroit, MI, April 10-12, 2018 [16].

Definitions/Abbreviations

AV - Autonomous vehicle
HiL - Hardware-in-the-loop
IMU - Inertial measurement unit
SLAM - Simultaneous localization and mapping
LIDAR - Light detection and ranging
V2X - Vehicle to everything
DSRC - Dedicated short-range communication
GPS - Global Positioning Systems
ECU - Electronic control unit
SAE - Society of Automotive Engineers
RTK - Real-Time Kinematic
UDP - User Datagram Protocol
DOF - Degrees of freedom
PC - Personal Computer
FoV - Field of view

References

1. Leonard, J.J. and Durrant-Whyte, H.F., "Simultaneous Map Building and Localization for an Autonomous Mobile Robot," in *IEEE/RSJ International Workshop on Intelligent Robots and Systems' 91.'Intelligence for Mechanical Systems, Proceedings IROS'91*, IEEE, 1991, 1442-1447.
2. Pritsker, A. and Alan, B., *Introduction to Stimulation and Slam II* (1986).
3. Dissanayake, M.W.M.G., Newman, P., Clark, S., Durrant-Whyte, H.F. et al., "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem," *IEEE Transactions on Robotics and Automation* 17(3):229-241, 2001.
4. Grisettiyz, G., Stachniss, C., and Burgard, W., "Improving Grid-Based Slam with Rao-Blackwellized Particle Filters by Adaptive Proposals and Selective Resampling," in *IEEE International Conference on Robotics and Automation, ICRA 2005. Proceedings of the 2005*, IEEE, 2005, 2432-2437.
5. Kohlbrecher, S., Von Stryk, O., Meyer, J., and Klingauf, U., "A Flexible and Scalable Slam System with Full 3d Motion Estimation," in *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, IEEE, 2011, 155-160.
6. Newman, P., Cole, D., and Ho, K., "Outdoor SLAM Using Visual Appearance and Laser Ranging," in *Robotics and Automation, 2006. Proceedings 2006 IEEE International Conference on ICRA 2006*, IEEE, 2006, 1180-1187.
7. Cole, D.M. and Newman, P.M., "Using Laser Range Data for 3D SLAM in Outdoor Environments," in *Robotics and Automation, 2006. Proceedings 2006 IEEE International Conference on ICRA 2006*, IEEE, 2006, 1556-1563.
8. Levinson, J. and Thrun, S., "Robust Vehicle Localization in Urban Environments Using Probabilistic Maps," in *IEEE International Conference on Robotics and Automation (ICRA), 2010*, IEEE, 2010, 4372-4378.
9. Vu, T.-D., Burlet, J., and Aycard, O., "Grid-Based Localization and Online Mapping with Moving Objects Detection and Tracking: New Results," in *Intelligent Vehicles Symposium, 2008 IEEE*, IEEE, 2008, 684-689.
10. Gelbal, S.Y., Wang, H., Chandramouli, N., Guvenc, L. et al., "A Connected and Autonomous Vehicle Hardware-in-the-Loop Simulator for Developing Automated Driving Algorithms," *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017.
11. Emirler, M.T., Wang, H., Aksun Güvenç, B., and Güvenç, L., "Automated Robust Path Following Control Based on Calculation of Lateral Deviation and Yaw Angle Error," in *ASME Dynamic Systems and Control Conference (DSCC)*, 30, Columbus, OH, USA, Oct. 28, 2015.
12. Moré, J.J., The Levenberg-Marquardt Algorithm: Implementation and Theory, in *Numerical Analysis* (Berlin, Heidelberg, Springer, 1978), 105-116.
13. Lucas, B.D. and Kanade, T., *An Iterative Image Registration Technique with an Application to Stereo Vision* (1981), 674-679.
14. Meer, P. et al., "Robust Regression Methods for Computer Vision: A Review," *International Journal of Computer Vision* 6(1):59-70, 1991.
15. Quigley, M., Conley, K., Gerkey, B., Faust, J. et al., "ROS: An Open-Source Robot Operating System," *ICRA Workshop on Open Source Software* 3(3.2):5, 2009.
16. Wen, B., Gelbal, S.Y., Aksun Güvenç, B., and Güvenç, L., "Localization and Perception for Control and Decision Making of a Low Speed Autonomous Shuttle in a Campus Pilot Deployment," SAE Technical Paper 2018-01-1182, 2018, doi:10.4271/2018-01-1182.

2019-01-0496

Cooperative Adaptive Cruise Control Design and Implementation

Mustafa Ridvan Cantas, Sukru Yaren Gelbal, Levent Guvenc, Bilin Aksun Guvenc

Ohio State University

Abstract

In this manuscript a design and implementation of CACC on an autonomous vehicle platform (2017 Ford Fusion) is presented. The developed CACC controls the intervehicle distance between the target vehicle and ego vehicle using a feedforward PD controller. In this design the feedforward information is the acceleration of the target vehicle which is communicated through Dedicated Short-Range Communication (DSRC) modem. The manuscript explains the detailed architecture of the designed CACC with used hardware and methods for the both simulation and experiments. Also, an approach to overcome detection failures at the curved roads is presented to improve overall quality of the designed CACC system. As a result, the initial simulation and experimental results with the designed CACC system is presented in the paper. The presented results indicate that CACC improves the car following performance of the ego vehicle as compared to the classical Adaptive Cruise Controller.

Introduction

With the recent advancements in automotive sensors, cars are becoming more autonomous making use of these new technologies. The Advanced Driver Assistant systems such as Adaptive Cruise Control (ACC) system do not only ensure the safety but also increase the comfort of travel. A well-known longitudinal control method, Cooperative Adaptive Cruise Control (CACC), which is an ACC system supported by the Dedicated Short-Range Radio Communication (DSRC) technology that allows Vehicle-to-Vehicle (V2V) communication, enables lower time headway. Reducing the time headway distance between two vehicles can significantly increase the capacity of the road. Also, platooning multiple vehicles has the potential to improve the fuel efficiency of the vehicles by avoiding unnecessary accelerations and decelerations, and reducing the air drag. Motivated by these advantages, in this manuscript, we will present a design and implementation of CACC on an autonomous vehicle platform (2017 Ford Fusion) with initial experimental results.

The designed CACC maintains the desired constant-time headway better than the well-known Adaptive Cruise Control (ACC). Thus, it is possible to reduce the headway for the CACC. In truck platooning smaller time gap results in higher fuel efficiencies by reducing the air drag resistance. Similarly reducing the headway time would increase the capacity of the highways significantly by improving the traffic flow rate [1]. Motivated by these advantages of CACC over ACC, in this manuscript, a Cooperative Adaptive Cruise Controller design process for the autonomous vehicle platform will be explained.

Adaptive Cruise Controllers are already being used in the production vehicles under different names. A comprehensive literature review for ACC systems is done in [2]. Adaptive Cruise Controllers aims to maintain the time-headway constant while car following without breaking the string stability. However, they cannot use low time-headway values since it would result in rear end collision in case of sudden speed changes in the traffic (shock-waves) [3]. Therefore, in ACC a small time gap causes string instability by amplifying the disturbances in the upstream direction. Using the DSRC communication, one can improve the car following performance by reducing the time-headway without breaking the string stability [4]. This car-following model is called Cooperative Adaptive Cruise Control. Some of the earlier work on CACC can be seen in [4- 10]. In [4] authors presented their CACC design methodology by considering the string stability requirements and they experimentally validated their design. One of the early implementations of CACC is done under California PATH program [5-6]. In 2011 several research institutes formed a CACC platoon at Grand Cooperative Driving Challenge (GCDC). Two of the CACC implementations in this challenge can be seen in [7-8]. In [9] authors presented their design for car-following with CACC and approaching maneuver controller. In [10] authors presented multi vehicle look ahead CACC simulation results which shows that the multi vehicle look ahead in CACC improves the performance of CACC. In [11-13] the authors presented how to handle the adversarial environment conditions.

The rest of the paper organized as follows. Next section will explain the designed CACC structure. Following that the simulation environment with target vehicle modeling and simulation results for two vehicle car following scenario will be presented. Then, the experimental vehicle set up with the explanation of sensors will be explained. In the Perception section, in-lane vehicle detection algorithm will be explained. Finally, the manuscript will be concluded with experimental results and their comparison with simulation results.

CACC Structure

The control structure of the designed CACC system is shown in the block diagram in Figure 1. The designed control system is similar to the one designed and shown to be string -stable in [4]. Since the vehicle does not have built-in ACC the low-level controller is designed as a gain-scheduled PI controller. As an upper controller, a PD controller with a feed-forward controller is used. To sustain the string stability a constant time headway spacing policy is employed [14]. The input of the feedforward controller is the acceleration of the target vehicle which is transmitted through DRSC radio communication. Formulation of the spacing policy is given in Equations 1-2. Where l is the length of the target vehicle, x_1 and x_2 are the position of the target

and ego vehicle, T_{hw} is the desired time-headway and V_{host} is the speed of the host vehicle. The designed PD controller minimizes the spacing error e which is given in Equation 3. Gains of the PD controller chosen as $k_P = {k_D}^2 = {w_K}^2$ where the w_K is chosen close to the bandwidth of the low-level closed-loop bandwidth.

$$\Delta x = x_1 - x_2 - l \tag{1}$$

$$\Delta x_{desired} = V_{host} T_{hw} + standstill\ distance \tag{2}$$

$$e = \Delta x - \Delta x_{desired} \tag{3}$$

Feedforward controller is designed in the same way where it was designed in [8]. Formulation of the feedforward controller is given in Equation 4, where $1/\tau$ represents the desired closed-loop bandwidth.

$$F = \frac{\tau s + 1}{T_{hw} s + 1} \tag{4}$$

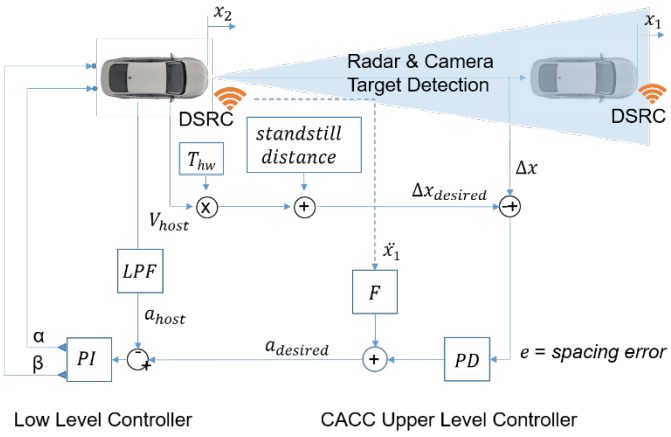


Figure 1. Cooperative Adaptive Cruise Controller (CACC) block diagram.

Simulation Environment

Development of the initial CACC model is done in CarSim-Matlab co-simulation environment [15]. CarSim is a vehicle simulation environment with the capability of simulating the dynamics of the vehicle. It can also simulate the target vehicle as a kinematic object. In the simulation, target vehicle will be driven by an Intelligent Driver model. By changing the desired speed and/or acceleration limits for the target vehicle, one can create different driving scenarios using Intelligent Driver Model (IDM) [16]. The formulation of the IDM is given in Equations 5-7.

$$\dot{x}_\alpha = \frac{dx_\alpha}{dt} = v_\alpha \tag{5}$$

$$\dot{v}_\alpha = \frac{dv_\alpha}{dt} = a\left(1 - \left(\frac{v_a}{v_0}\right)^\delta - \left(\frac{s^*(v_a, \Delta v_a)}{s_a}\right)^2\right) \tag{6}$$

$$s^*(v_a, \Delta v_a) = s_0 + v_a T + \frac{v_a \Delta v_a}{2\sqrt{ab}} \tag{7}$$

where:

v_0: the velocity the vehicle would drive at in free traffic
s_0: a minimum desired net distance
T: the minimum possible time to the vehicle in front
a: the maximum vehicle acceleration
b: (a positive number) the maximum vehicle breaking m/s^2

The IDM car-following model is commonly used in traffic simulations for simulating driving behavior of the human driver in traffic. In this case, the Intelligent driver model is used to model a human driver for the target vehicle. In CarSim one can also create realistic roads by importing GPS trajectory of the rote. The simulation of the radar and camera is also possible by using the virtual sensors offered in CarSim. Figure 2 shows the visualization of the car following scenario simulation with a radar field of view.

Figure 2. CarSim CACC Simulation visualization.

After creating the simulation environment which replicates the structure shown in Figure 1 simulations run for two different scenarios: ACC and CACC. As the initial evaluation, in the created simulation environment target vehicle first accelerates to a set speed of 20 km/h then it changes set speed to 25km/h and finally it stops. In the simulations, the ego vehicle follows the target vehicle with 1 sec time headway. As it can be seen from simulation results in Figure 3 CACC follows the target vehicle much better. Although both of the speed controllers maintain the time headway, CACC time headway follows the set value (1 second) more accurately. Simulation results with 0.6s is overlayered over the experimental results.

In another scenario, in order to show the performance of the CACC in more realistic scenario the target vehicle speed, acceleration profiles over time is collected by driving the experimental vehicle in an urban route environment. By replaying the recorded data during the simulation, the real world driving experience with a sudden acceleration and braking behavior of the target vehicle in an urban environment is simulated. Similar to the previous simulation results CACC follows the desired time headway of 0.6s much better as compared to ACC (Figure 4). Especially for sudden changes in speed of the target vehicle, CACC responds much better and keeps the desired spacing more accurately.

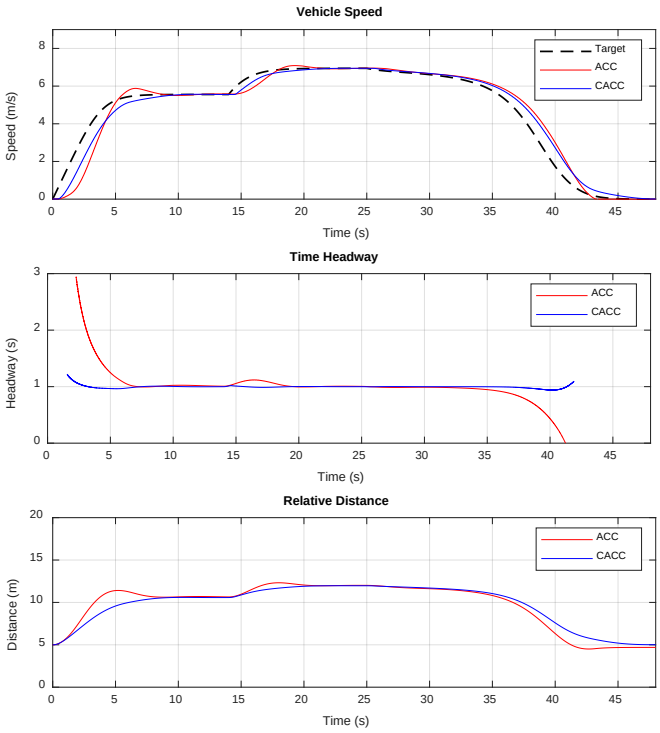


Figure 3. CarSim ACC and CACC simulation results for 1 second time headway with an IDM driven target vehicle.

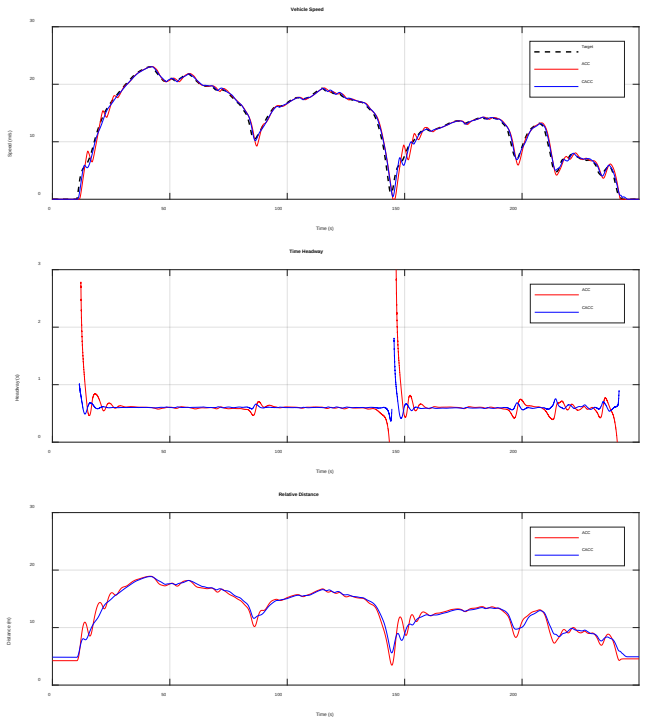


Figure 4. CarSim ACC and CACC simulation results for 0.6 second time headway with a human driven target vehicle data.

Experimental Vehicle

Figure 5. Autonomous vehicle development platform of Automated Driving Lab, at The Ohio State University.

For the experiments, a 2017 Ford Fusion with drive by wire capability is used (Figure 5). Since the vehicle is not equipped with an ACC the throttle and brake actuation are realized through CAN bus messages. For the longitudinal motion measurements, speed and acceleration measurements on the vehicle CAN bus is used.

As the electronic control unit, a dSpace MicroAutoBox II (Figure 6) is employed due to its easy prototyping property with high-performance real-time system implementation capabilities. As it can be seen from the block diagram every equipment in the vehicle is connected to the MicroAutoBox controller. All the measurements coming from the sensors, vehicle CAN bus are processed in the controller and based on the embedded algorithm throttle and brake commands are sent to the vehicle. All the algorithms and the data parsing coming from the sensors are programmed using MATLAB Simulink blocks and they are embedded into the MicroAutoBox controller. As a user interface a portable computer with ControlDesk application is used.

To detect the objects on the road the vehicle is equipped with a 76.5GHz Delphi forward looking radar which can track up to 64 objects and give their positions and relative velocity information. The radar is a combination of both long and middle range radars. Radar is connected to both MicroAutoBox and the laptop. While the data coming from the radar is parsed and processed in the MicroAutoBox, detections can be seen in real time for diagnosis purposes using DataView software. In order to visually validate the radar detections, a forward-looking webcam is connected to the laptop. DataView software can overlay the detections to the video stream acquired from this webcam (Figure 7).

A black and white monocular smart camera from Mobileye (Figure 6) is used to detect lane lines on the road to determine in lane vehicles among detected targets via radar. This camera can detect the lane line markers on the road and provides the lane line information in the form of 3rd order polynomials. Coefficients of the lane line polynomials are available on the CAN bus alongside the road curvature information.

The test vehicle is also equipped with two Denso WSU (Wireless Safety Unit) 5900 DSRC modems to communicate with the target vehicle. In the CACC scenario target vehicle broadcast its acceleration alongside the Basic Safety Message (BSM) [17]. While the first modem is receiving the target vehicle acceleration, the second modem

on the vehicle is used to transmit the acceleration of the virtual target vehicle.

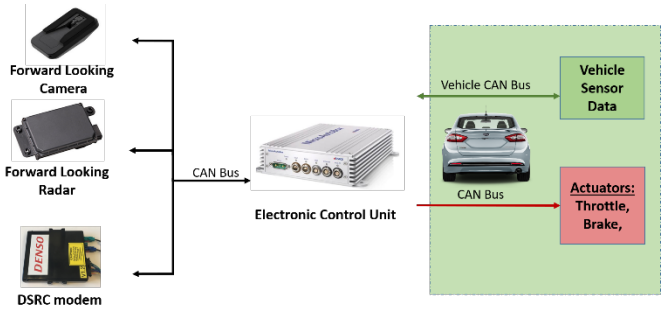


Figure 6. CACC experimental vehicle hardware block diagram.

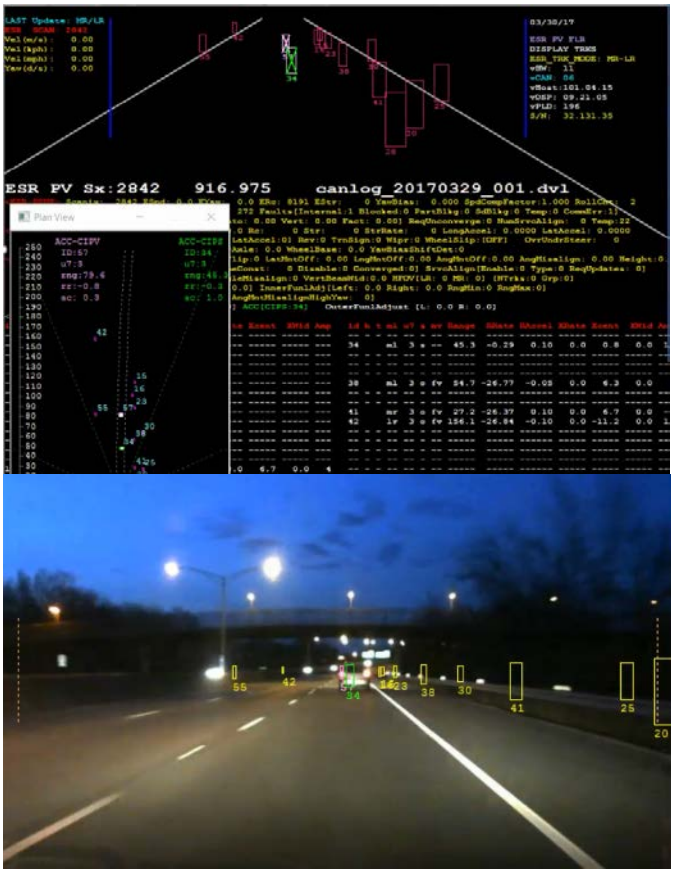


Figure 7. Sample Radar Detection Visualization in real time.

Perception

A radar is used to detect the positions of the target vehicles. The downside of this method the radar only provides the position and speed information of the detected objects. Since the radar does not know the lane information on the road, it cannot distinguish the vehicles which are in the lane and which are not. From the Radar, an ACC target information is available, which is selected by the radar using the speed of the vehicle, steering angle, and yaw rate information of the ego vehicle. Although most of the time this target detection is valid, since the exact algorithm for choosing the ACC target vehicle is not known and availability of the ACC target depends on the algorithm used in the Radar we are forced to develop another method to detect the vehicles in the lane.

According to Zhang et.al [18], there are two main difficulties to detect the target vehicle using radar. First, differentiation of the lane change or curve entry/exit behavior is challenging. Second, when the vehicle in the next lane goes into the curvature it can be misclassified as in the host lane. To overcome these difficulties a camera and radar are used together. While the lane boundary information comes from the camera, objects detections are acquired from the radar. For each time step, acquired detections are sorted by their longitudinal range. Then each detected object is checked to see whether it is in the host lane or not by comparing its lateral position with respect to the lane boundaries. Block diagram of the system can be seen in Figure 8. One should note that if at least one of the lane lines are not visible to the camera, it is required to create the lane boundaries synthetically. If only one of the lane lines is available, the other lane boundary is created using the available lane boundary information and the lane width. If both of the lane lines are not available, it is assumed that the vehicle moves straight, and the target object is searched within a window where the width of the window is equal to the lane width.

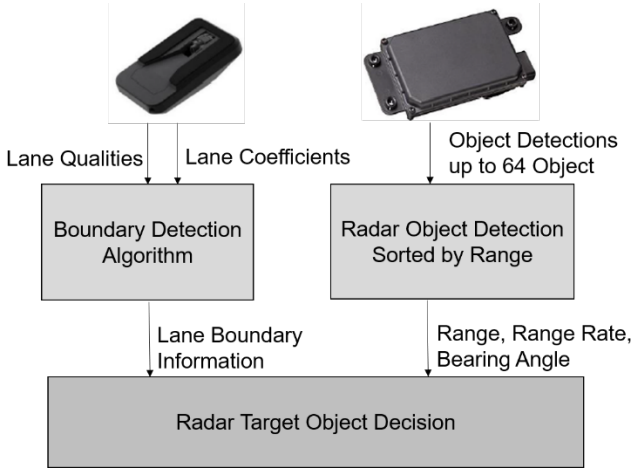


Figure 8. In lane vehicle detection structure with camera and radar combination.

Radar can provide the positions of the detected objects in polar coordinates (Figure 9). The measurements acquired from the radar for each object are listed with their range (r_i), bearing angle (α_i) parameters and range rate. These coordinates are converted into the Cartesian coordinate system using basic trigonometric equations. Since the radar is placed in front of the front bumper and the origin of the radar coordinate system is chosen as the center of the radar, measurements acquired from the radar are converted to the longitudinal distance (x_i) and the lateral distance (y_i) of the target objects from the center of the front bumper (Equations 8 and 9).

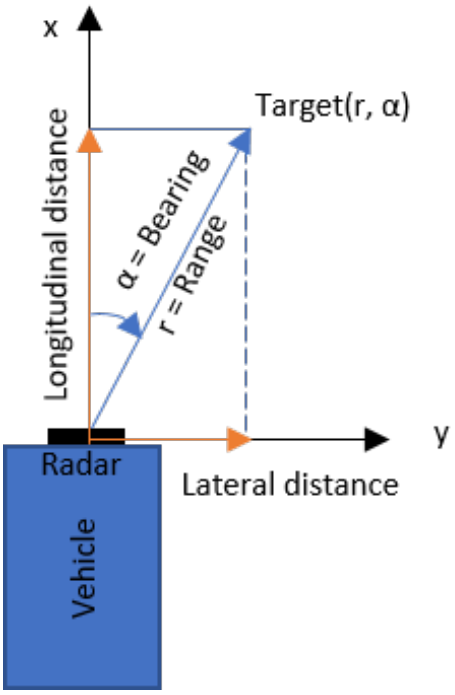


Figure 9. Radar detection coordinate system.

$$Longitudinal\ distance = x_i = r_i * \cos(\alpha_i) \quad (8)$$

$$Lateral\ distance = y_i = r_i * \sin(\alpha_i) \quad (9)$$

The camera in the vehicle is centered at the top of the windshield. It provides the lane line definitions as a third order polynomial in the camera coordinate system, where the origin of the coordinate system is the location of the camera. The curve fitted to the left and right lanes are given in Equations 10 and 11 respectively.

$$y^l(x) = {a_0}^l + {a_1}^l * x + {a_2}^l * x^2 + {a_3}^l * x^3 \quad (10)$$

$$y^r(x) = {a_0}^r + {a_1}^r * x + {a_2}^r * x^2 + {a_3}^r * x^3 \quad (11)$$

where *y*, *x* represents the lateral and longitudinal positions of the points on the fitted curve. Each a_i represents the coefficients of the fitted curve to the lane lines. Here r and l superscripts differentiate the curve fits for left and right lane respectively. At the final step, by knowing the positions of the targets and lane boundaries, the closest target in the lane can be chosen as the in-lane target. For this purpose, firstly, all the objects are sorted by their longitudinal distances. Then each detected object's coordinates are translated to the camera coordinate system by adding the distance between the camera and the radar in the longitudinal direction (Δx) Equations 12 and 13.

$$x_{new\_i} = x_i + \Delta x \quad (12)$$

$$y_{new\_i} = y_i \quad (13)$$

Inserting the new lateral distance of the target objects into the Equations 10 and 11, the left (LB) and right boundaries (RB) of the lane at the distance of the target object is calculated (Equations 14 and 15).

$$LB_i = {a_0}^l + {a_1}^l * x_{new\_i} + {a_2}^l * {x_{new\_i}}^2 + {a_3}^l * {x_{new\_i}}^3 \quad (14)$$

$$RB_i = {a_0}^r + {a_1}^r * x_{new\_i} + {a_2}^r * {x_{new\_i}}^2 + {a_3}^r * {x_{new\_i}}^3 \quad (15)$$

If the lateral distance of the i^{th} detected object is between the calculated lane boundaries for the longitudinal distance of the i^{th} object, this object is considered to be an in-lane possible target (Equation 16).

$$(LB_i < y_{new\_i} < RB_i) \quad (16)$$

Among all in-lane possible targets, the closest vehicle in the longitudinal direction is accepted as the in-lane target vehicle. Sample experimental result for this method is presented in Figure 10. The lateral position of the target vehicle with respect to the center of the vehicle and lane boundaries are shown in the plot. One can see from the experimental result that the target vehicle is detected even in the curved section of the road accurately.

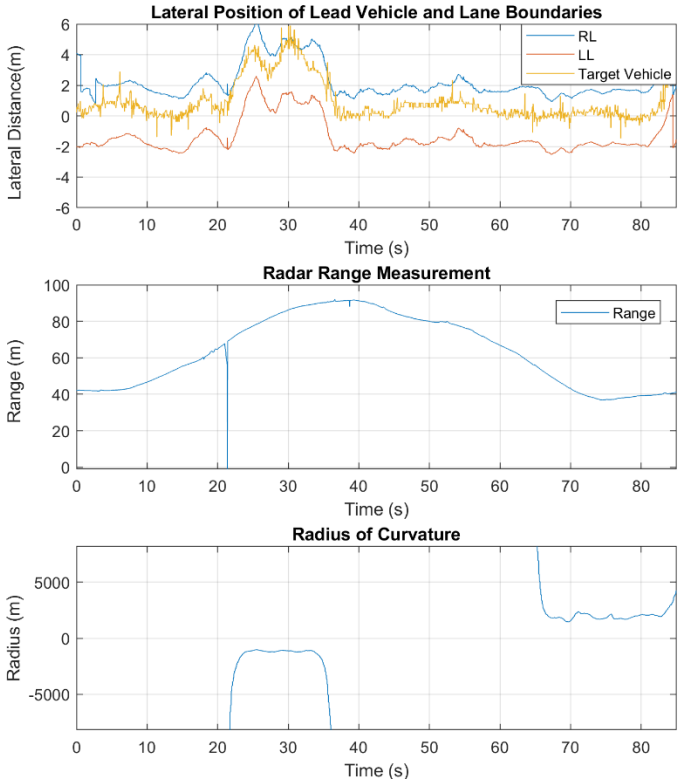


Figure 10. In lane vehicle detection experimental results.

Experimental Results

In the experiments, the target vehicle speed profile is chosen as the same with the simulation target vehicle speed profile. Target vehicle profile is generated in real time with IDM driver similar to the simulation environment. The virtual target vehicle accelerates to 20 km/h and 25 km/h consecutively then it stops. In the CACC scenario, the simulated acceleration values for the target vehicle is broadcasted through DSRC OBU and it is received by another OBU for the ego vehicle. One can see the experimental results for the ACC and CACC

for 1 second headway time overlaid onto the simulation results on Figure 11. The simulation results match with the experimental results. The small mismatches between the experiment and CarSim simulation are caused due to the fact that the CarSim vehicle model is not an exact model of the experimental vehicle. In the CarSim simulation, a generic D class vehicle model is used. In response to the speed changes in the target vehicle speed profile, the CACC speed controllers start accelerating and deceleration faster as compared to ACC using the target vehicle acceleration information coming from the DSRC modem. Thus, the CACC controller can follow the target vehicle more accurately. CACC time headway following performance is much better than the performance of ACC.

Similarly, CarSim simulations and experiments are repeated for desired time headway of 0.6 s. The comparison of the simulation and experiment results are shown in Figure 12. Similar to the previous case the simulation and experimental results are close to each other. And CACC performs better while following the target vehicle with constant 0.6s time headway.

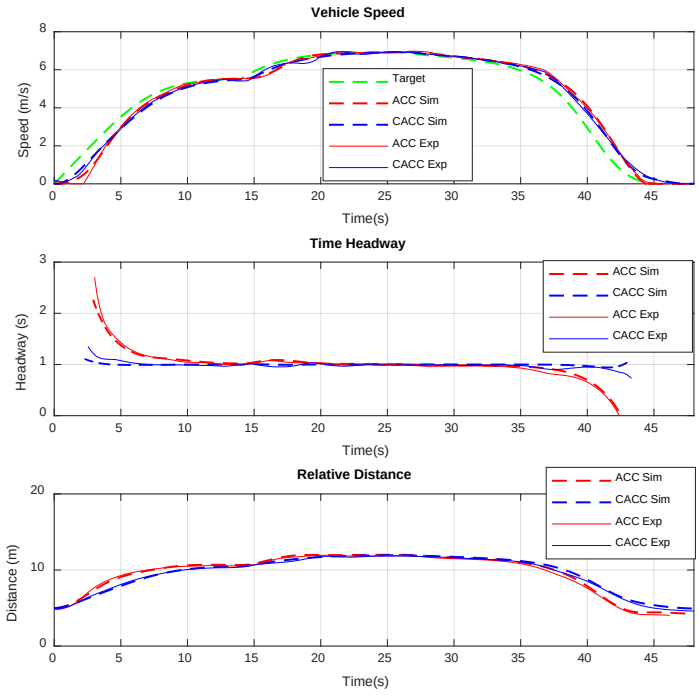


Figure 11. Comparison of ACC and CACC experimental results with simulation results for 1 second desired time headway.

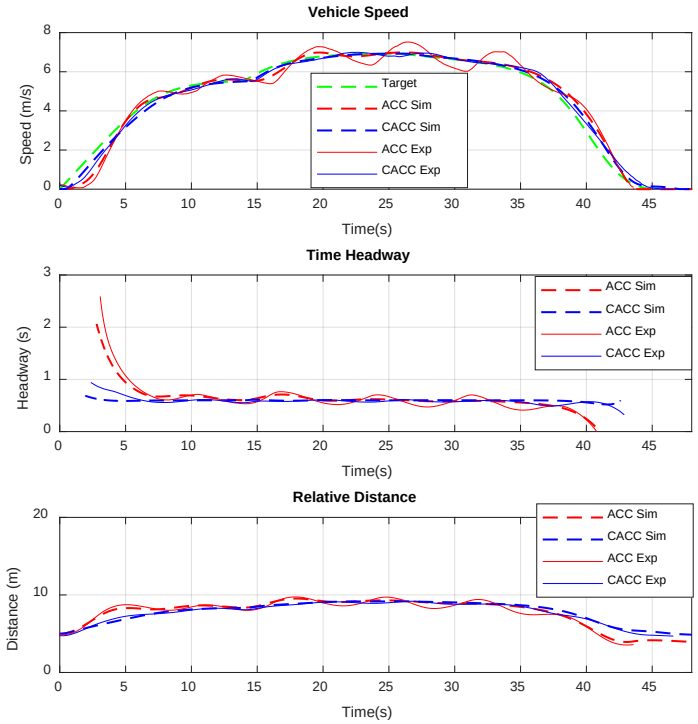


Figure 12. Comparison of ACC and CACC experimental results with simulation results for 0.6 second desired time headway.

Summary/Conclusions

In this manuscript, the design process of ACC and CACC structure of the Automated Driving Lab at The Ohio State University is presented with initial simulation and experimental results. Both simulation and experimental results show that communication between the target vehicle and ego vehicle in CACC increase the car following performance significantly. This performance improvement will lead to better string stability and capacity increase on the roads. As a next step conducted experiments will be repeated with actual target vehicle and higher speed scenarios. The performance of the lower level controllers will be improved. Multi-vehicle look-ahead scenarios will be examined.

References

1. B. Van Arem, J. G. Van Driel, and R. Visser, “The Impact of Cooperative Adaptive Cruise Control on Traffic-Flow Characteristics,” *IEEE Trans. Intell. Transp. Syst.*, vol. 7, no. 4, p. 429, 2006.
2. Lingyun Xiao & Feng Gao (2010) A comprehensive review of the development of adaptive cruise control systems, Vehicle System Dynamics, 48:10, 1167-1192, DOI: 10.1080/00423110903365910
3. M. Minderhoud and P. Bovy,Impact of intelligent cruise control on motorway capacity,Transportation Research Record 1679 Paper No. 99-0049, 1999.
4. G. J. L. Naus, R. P. A. Vugts, J. Ploeg, M. J. G. van de Molengraft, and M. Steinbuch, “String-stable CACC design and

experimental validation: A frequency-domain approach," IEEE Trans. Veh. Technol., vol. 59, no. 9, pp. 4268–4279, Nov. 2010.
5. R. Rajamani, H. S. Tan, B. K. Law, and W. Bin Zhang, "Demonstration of integrated longitudinal and lateral control for the operation of automated vehicles in platoons," *IEEE Trans. Control Syst. Technol.*, vol. 8, no. 4, pp. 695–708, 2000.
6. R. Rajamani and S. E. Shladover, "Experimental comparative study of autonomous and co-operative vehicle-follower control systems," *Transp. Res. Part C Emerg. Technol.*, vol. 9, no. 1, pp. 15–31, 2001.
7. J. Ploeg, B. T. M. Scheepers, E. van Nunen, N. van de Wouw, and H. Nijmeijer, "Design and experimental evaluation of cooperative adaptive cruise control," 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC), Washington, DC, 2011, pp. 260-265. doi: 10.1109/ITSC.2011.6082981
8. L. Güvenç *et al.*, "Cooperative Adaptive Cruise Control Implementation of Team Mekar at the Grand Cooperative Driving Challenge," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 3, pp. 1062–1074, 2012.
9. V. Milanes, S. E. Shladover, J. Spring, C. Nowakowski, H. Kawazoe, and M. Nakamura, "Cooperative adaptive cruise control in real traffic situations," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 1, pp. 296–305, 2014.
10. J. Meier *et al.*, "Implementation and evaluation of cooperative adaptive cruise control functionalities," in *IET Intelligent Transport Systems*, vol. 12, no. 9, pp. 1110-1115, 11 2018. doi: 10.1049/iet-its.2018.5175
11. Dadras, S., Dadras, S., & Winstead, C. (2018). Identification of the Attacker in Cyber-Physical Systems with an Application to Vehicular Platooning in Adversarial Environment. 2018 Annual American Control Conference (ACC), 5560-5567.
12. Dadras, S., Dadras, S., & Winstead, C. (2018). Reachable Set Analysis of Vehicular Platooning in Adversarial Environment. 2018 Annual American Control Conference (ACC), 5568-5575.
13. Dadras, S., Gerdes, R. M., and Sharma R. (2015). Vehicular Platooning in an Adversarial Environment. In Proceedings of the 10th ACM Symposium on Information, Computer and Communications Security (ASIA CCS '15). ACM, New York, NY, USA, 167-178. DOI: https://doi.org/10.1145/2714576.2714619
14. D. Swaroop, J. K. Hedrick, C. C. Chien, and P. Ioannou, "A Comparision of Spacing and Headway Control Laws for Automatically Controlled Vehicles," *Veh. Syst. Dyn.*, vol. 23, no. 1, pp. 597–625, 1994.
15. Ş. Y. Gelbal, M. R. Cantaş, S. Tamilarasan, L. Güvenç, and B. Aksun Güvenç, "A Connected and Autonomous Vehicle Hardware-in- the-Loop Simulator for Developing Automated Driving Algorithms," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC) Banff Center, Banff, Canada*, 2017.
16. M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," Physical review E, vol. 62, no. 2, p. 1805, 2000.
17. SAE International, "(R) Dedicated Short Range Communications (DSRC) message set dictionary." [Online]. Available: https://saemobilus.sae.org/content/j2735_200911
18. D. Zhang, K. Li, and J. Wang, "Radar-based target identification and tracking on a curved road," *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.*, vol. 226, no. 1, pp. 39–47, 2012.

Contact Information

M. R. Cantas is with the Department of Electrical and Computer Engineering, The Ohio State University, Columbus, OH 43210 USA (e-mail: cantas.1@osu.edu).

Acknowledgments

This work was partially supported the National Science Foundation under Grant 1640308, and by the U.S. Department of Transportation Mobility 21: National University Transportation Center for Improving Mobility (CMU) sub-project titled: Smart Shuttle: Model Based Design and Evaluation of Automated On-Demand Shuttles for Solving the First-Mile and Last-Mile Problem in a Smart City. The authors would like to thank to Nitish Chandramouli and Santhosh Tamilarasan for their valuable discussions and help.

Definitions/Abbreviations

ACC	Adaptive Cruise Control
CACC	Cooperative Adaptive Cruise Control
IDM	Intelligent Driver Model
BSM	Basic Safety Message
OBU	On Board Unit

2019-01-0674

Discrete-time Robust PD Controlled System with DOB/CDOB Compensation for High Speed Autonomous Vehicle Path Following

Haoan Wang, Levent Guvenc

Ohio State University

Abstract

In recent years, there has been increasing research on automated driving technology. Autonomous vehicle path following performance is one of significant consideration. This paper presents discrete time design of robust PD controlled system with disturbance observer (DOB) and communication disturbance observer (CDOB) compensation to enhance autonomous vehicle path following performance. Although always implemented on digital devices, DOB and CDOB structure are usually designed in continuous time in the literature and also in our previous work. However, it requires high sampling rate for continuous-time design block diagram to automatically convert to corresponding discrete-time controller using rapid controller prototyping systems. In this paper, direct discrete time design is carried out. Digital PD feedback controller is designed based on the nominal plant using the proposed parameter space approach. Zero order hold method is applied to discretize the nominal plant, DOB and CDOB structure in continuous domain. Discrete time DOB is embedded into the steering to path following error loop for model regulation in the presence of uncertainty in vehicle parameters such as vehicle mass, vehicle speed and road-tire friction coefficient and rejecting external disturbance like crosswind force. On the other hand, time delay from CAN bus based sensor and actuator command interfaces results in degradation of system performance since large negative phase angles are added to the plant frequency response. Discrete time CDOB compensated control system can be used for time delay compensation where the accurate knowledge of delay time value is not necessary. A validated model of our lab's Ford Fusion hybrid automated driving research vehicle is used for the simulation analysis while the vehicle is driving at high speed. Simulation results successfully demonstrate the improvement of autonomous vehicle path following performance with the proposed discrete time DOB and CDOB structure.

I. Introduction

During the past decades, autonomous vehicle driving technology has been developing rapidly. Researchers are investigating different steering control methods to improve path following performance of autonomous vehicle. In [1], a double loop PD-PID controller is designed for the vehicle steering control. The inner loop is a PID controller which performs to control the position of steering wheel while the outer loop is a PD controller which aims at vehicle's heading control. Similar to double loop PD-PID controller, nested PI and PID controller was proposed in [2]. PI steering controller at inner loop that reduces yaw rate tracking error is used to improve the vehicle steering dynamics. A PID controller is employed at external control loop to reject the lateral deviation from the desired path due to road curvature disturbance. Sliding model control has been widely used due to its benefits of fast and good transient response and robustness with respect to system uncertainties and external disturbances [3]. As another classic control method, model predictive control has the capability to deal with a wide variety of process control constraints systematically and is applied to the socially acceptable collision free path following system in [4]. Parameter space approach based robust PID controller design is presented in [5-6] and it has the advantage of dealing with variable vehicle parameters such as vehicle mass, vehicle velocity and road-tire friction coefficient.

In order to further improve autonomous vehicle path following performance in the existence of uncertain parameters and external disturbance, a disturbance observer (DOB) is added into the control system to achieve insensitivity to modeling error and disturbance rejection. The disturbance observer was firstly proposed by Ohnishi [7] and further developed by Umeno and Hori [8]. Later, DOB has been applied in mechatronic applications in the literature. In [9], robustness of disturbance observer is added to the model of electrohydraulic system considering the case in which the plant has large parametric variation. Two-degrees-of-freedom control architecture known as the model regulator (disturbance observer) is proposed in [10] as a robust steering controller for improving yaw stability in a driver-assist system.

In the autonomous vehicle path following system, CAN bus delay in the steering system is another important issue. Time delay causes large negative phase angles which lead to performance degradation or even instability of the system. The Smith predictor has been firstly introduced and used in many different cases such as [11-12]. It has the advantage of easy implementation. However, time delay model and model accuracy in the knowledge of time delay are required to ensure no degradation of compensation performance. Communication disturbance observer is proposed as another time delay compensation approach. It was firstly applied in the bilateral teleoperation systems [13] and has been extended to robust time delayed control system in [14-15]. Compared with Smith predictor, the accurate knowledge of the time delay value is not necessary in communication disturbance observer and also it can be used for plants with variable time delay.

Disturbance observer and communication disturbance observer are usually designed in continuous time domain and conducted on the

digital platform using a very high sampling rate. This requires high speed processors and may not be achievable in many situations. Therefore, it is worthwhile to investigate direct discrete time design for DOB and CDOB compensated system. [16] discusses application of different discretization methods in discrete implementation of the DOB based control and analyzes three very popular discretization methods: backward difference, bilinear transform, and Al-Alaoui method. It shows that bilinear transform method and Al-Alaoui method provide significantly better performance than backward difference method. A state-space analysis of discrete-time DOB for a class of sampled-data control systems is presented in [17], where discrete-time singular perturbation theory is used to make uncertain sampled-data control system with the discrete-time DOB behaves as the nominal model without disturbance. [18] analysis robust stability condition for discrete time DOB designed by using forward difference discretization method where it is observed that the ratio between time constant of *Q* filter and sampling time is of significant importance in discrete time DOB and the ratio is suggested to be one for stability. In [19], a discrete-time communication disturbance observer is applied in a network-based gait rehabilitation system for compensating time delay which exist in both sensor-controller and controller actuator channels.

This paper is an extension of our previous work about DOB and CDOB compensated autonomous vehicle path following control system from continuous time domain to discrete time domain. Digital robust PD controller is designed based on the parameter space approach. Uncertainty box illustrating vehicle parameter variations is formed where the vehicle is operating at high speed. DOB structure and CDOB structure are discretized using zero-order-hold method. Simulation results show that discrete time DOB compensated system deals with model regulation and external disturbance rejection and discrete time CDOB compensated system realizes time delay compensation. Both of them present better path following performance than PD feedback controlled system.

The rest of this paper is organized as follows. Section II presents the validated single track vehicle model used for the autonomous vehicle path following. The structures of disturbance observer and communication disturbance observer in discrete time domain are illustrated in Section III and Section IV, respectively. Section V presents parameter space approach based digital robust PD controller design and discretization of DOB and CDOB structures through ZOH method is given in section VI. Section VII shows simulation results of autonomous vehicle path following using discrete time PD with DOB compensation and CDOB compensation, respectively. The paper ends with conclusions in Section VIII.

II. Vehicle Model

A single track vehicle model presented in Figure 1 is used to model the steering dynamics. The state space model can be described as:

$$\begin{bmatrix}\dot{\beta}\\ \dot{r}\\ \Delta\dot{\psi}\\ \dot{y}\end{bmatrix}=\begin{bmatrix}a_{11} & a_{12} & 0 & 0\\ a_{21} & a_{22} & 0 & 0\\ 0 & 1 & 0 & 0\\ V & l_s & V & 0\end{bmatrix}\begin{bmatrix}\beta\\ r\\ \Delta\psi\\ y\end{bmatrix}+\begin{bmatrix}b_{11}\\ b_{21}\\ 0\\ 0\end{bmatrix}\delta_f+\begin{bmatrix}0\\ 0\\ -\mathrm{V}\\ -l_s\mathrm{V}\end{bmatrix}\rho_{ref} \quad (1)$$

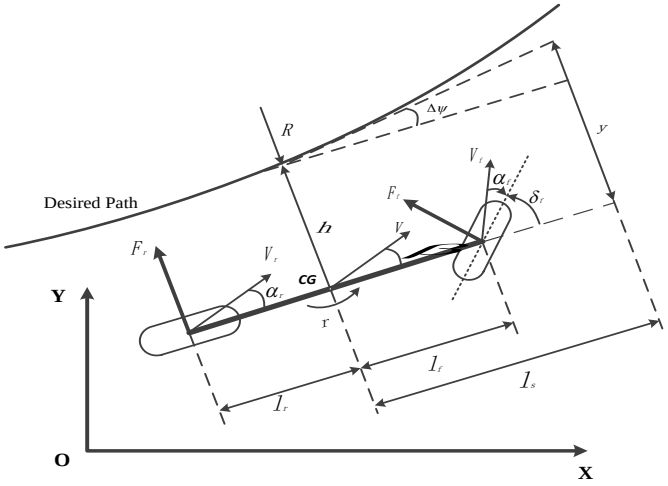


Figure 1. Single track vehicle model diagram

As crosswind force has some influence on dynamics of the vehicle system, the extended vehicle model is given as:

$$\begin{bmatrix}\dot{\beta}\\ \dot{r}\\ \Delta\dot{\psi}\\ \dot{y}\end{bmatrix}=\begin{bmatrix}a_{11} & a_{12} & 0 & 0\\ a_{21} & a_{22} & 0 & 0\\ 0 & 1 & 0 & 0\\ V & l_s & V & 0\end{bmatrix}\begin{bmatrix}\beta\\ r\\ \Delta\psi\\ y\end{bmatrix}+\begin{bmatrix}b_{11}\\ b_{21}\\ 0\\ 0\end{bmatrix}\delta_f+\begin{bmatrix}0\\ 0\\ -\mathrm{V}\\ -l_s\mathrm{V}\end{bmatrix}\rho_{ref}+\begin{bmatrix}\frac{1}{mV}\\ 0\\ 0\\ 0\end{bmatrix}F_{wind}+\begin{bmatrix}0\\ \frac{l_{wind}}{J}\\ 0\\ 0\end{bmatrix}F_{wind} \quad (2)$$

where

$$a_{11}=-(c_r+c_f)/\tilde{m}V,\ a_{12}=-1+(c_r l_r-c_f l_f)/\tilde{m}V^2 \quad (3)$$

$$a_{21}=(C_r l_r-c_f l_f)/J,\ a_{22}=-(c_r l_r^2+c_f l_f^2)/JV^2$$

$$b_{11}=c_f/\tilde{m}V,\ b_{21}=c_f lf/J$$

Writing vehicle steering dynamics in standard form according to (2):

$$\dot{x}=Ax+Bu \quad (4)$$

The transfer function from front wheel steering angle δ_f to the lateral deviation *y* in continuous time domain is calculated as equation (5):

$$\frac{y}{\delta_f}=G_{\delta_f}=[0\ 0\ 0\ 1](sI-A)^{-1}\begin{bmatrix}b_{11}\\ b_{21}\\ 0\\ 0\end{bmatrix} \quad (5)$$

Figure 2. Experiment vehicle

A Ford Fusion hybrid sedan shown in Figure 2 is used as the vehicle under consideration for controller design and simulation. The values of single track model parameters given in Table 1 are measured from experimental vehicle and the vehicle model is validated as seen in reference [20]. Vehicle virtual mass $\tilde{m}(m/\mu)$, vehicle velocity V and road friction coefficient μ are taken as three uncertain parameters of interest. The nominal values of these three parameters are 2,000 kg, 60 km/hr and 1, respectively. The uncertainty box showing maximum parametric variation in velocity V and virtual mass $\tilde{m}$ of this vehicle is shown in Figure 3. Four vertices labeled by a, b, c, d in the uncertainty box are used to evaluate the performance improvement of the disturbance observer compensated system.

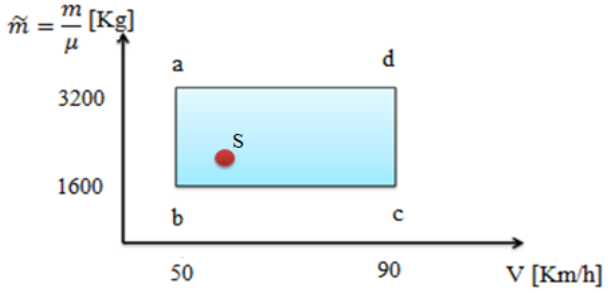


Figure 3. Parametric Uncertainty Box

Table I. Ford Fusion vehicle model parameters

β	vehicle side slip angle [rad]
subscript f	front tires
V	vehicle velocity [m/s]
δ_f	front wheel steering angle [rad]
J	yaw moment of inertia [3728 kgm^2]
C_r	rear cornering stiffness [50,000 N/rad]
l_f	distance from CG to front axle [1.3008 m]
l_r	distance from CG to rear axle [1.5453 m]
$\rho_{ref}=1/R$	curvature of path [1/m]
r	vehicle yaw rate [rad/s]
subscript r	rear tires
$\Delta\psi$	yaw orientation error with respect to path [rad]
y	lateral deviation [m]
C_f	front cornering stiffness [195,000 N/rad]
m	vehicle mass [2,000 kg]
l_s	preview distance [2m]
F_{wind}	crosswind force [500N]
l_{wind}	horizontal distance from crosswind force acting point to vehicle's CG

III. Discrete Time Disturbance Observer

The block diagram of the discrete time closed-loop control system with disturbance observer compensation is depicted in Figure 4. In the block diagram, robust digital PD feedback controller is used as a baseline controller which is designed based on the nominal discrete model of the vehicle. $Q(z)$ is the low pass filter to be selected and its bandwidth determines the bandwidth of model regulation and disturbance rejection. System plant $G(z)$ is formulated by taking both model uncertainty $\Delta_m(z)$ and external disturbance d into account. The vehicle input - output relation becomes:

$$y = G(z)u + d = (G_n(z)(1 + \Delta_m(z)))u + d \quad (6)$$

where $G_n(z)$ is the desired model of plant and $G(z)$ represents the actual plant. The goal in disturbance observer design is to obtain:

$$y = G_n(z)u_1 \quad (7)$$

as the input-output relation in the presence of model uncertainty $\Delta_m(z)$ and external disturbance d. u_1 is regarded as a new steering input which is derived as follows. By considering model uncertainty and external disturbance as an extended disturbance e, equation (6) can be rewritten as (8):

$$y = (G_n(z)(1 + \Delta_m(z)))u + d = G_n(z)u + e \quad (8)$$

Combining equation (7) with equation (8), the new control input u_1 is represented as:

$$u_1 = u + \frac{e}{G_n(z)} \quad (9)$$

and

$$u = u_1 - \frac{e}{G_n(z)} = u_1 - \frac{y}{G_n(z)} + u \quad (10)$$

In order to limit the compensation to a low frequency range to avoid stability robustness problem at high frequency, the feedback signals in (10) are multiplied by the low pass filter $Q(z)$ and implementation equation becomes:

$$u = u_1 - \frac{Q}{G_n(z)} y + Qu \quad (11)$$

According to discrete time disturbance observer structure in Figure 4, we have the following equation (z is omitted in transfer function for simplicity):

$$Y(z) = \frac{CGG_n}{G_n(1-Q)+G(CG_n+Q)} R(z) + \frac{G_n(1-Q)}{G_n(1-Q)+G(CG_n+Q)} D(z) \quad (12)$$

In DOB, Q is chosen as a unity low pass filter, with $Q \rightarrow 1$,

$$\frac{Y(z)}{R(z)} = \frac{CGG_n}{G_n(1-Q)+G(CG_n+Q)} \rightarrow \frac{CG_n}{(1+CG_n)} \quad (13)$$

From (13), we can see that DOB augmented plant behaves like its nominal plant G_n, which realizes the model regulation. Also, with $Q \rightarrow 1$,

$$\frac{Y(z)}{D(z)} = \frac{G_n(1-Q)}{G_n(1-Q)+G(CG_n+Q)} \rightarrow 0 \quad (14)$$

which realizes disturbance rejection.

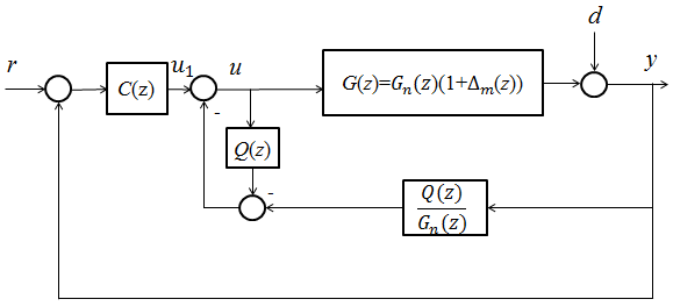


Figure 4. Digital Disturbance Observer Compensated Control System

IV. Discrete Time Communication Disturbance Observer

In order to deal with path following performance degradation or even destabilization caused by the time delay from CAN bus based sensor and actuator command interfaces, communication disturbance observer is applied to compensate the time delay. Figure 5 illustrates block diagram of time delay estimation for CDOB design in z domain. Time delay is considered as a disturbance d which is injected on the system and the aim is to obtain disturbance estimation $\hat{d}$. Equation (15) is obtained from Figure 5 and it can be rewritten as (16). Then, the estimated disturbance $\hat{d}$ is derived by multiplying d with $Q(z)$ to ensure causality as shown in equation (17).

$$y = G_n(z)(u - d) \quad (15)$$

$$d = u - G_n(z)^{-1} y \quad (16)$$

$$\hat{d} = Q(z)\,(u - G_n(z)^{-1} y) \quad (17)$$

According to network disturbance concept as depicted in Figure 6, $\hat{d}$ can be also expressed as equation (18):

$$\hat{d} = u - uz^{-N} \quad (18)$$

where u is system input and N is the time delay.

In this way, the estimated disturbance $\hat{d}$ is used to compensate the time delay effect in the feedback signal.

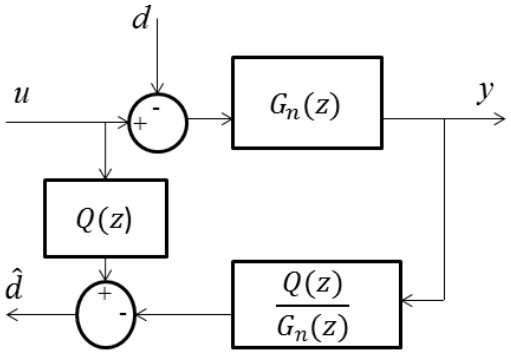


Figure 5. Classic disturbance observer in z domain

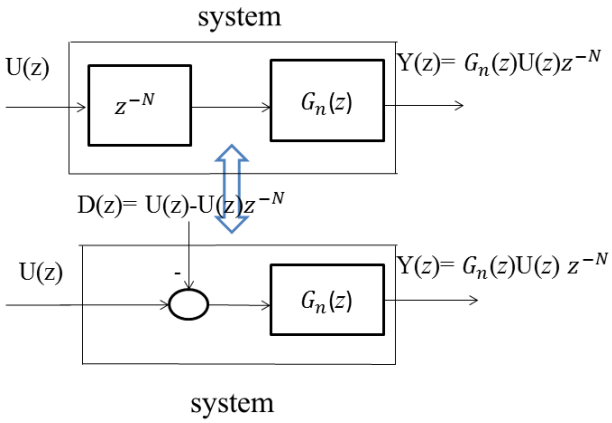


Figure 6. Conceptual block diagram of network disturbance

The block diagram of discrete-time CDOB compensated system is shown as Figure 7. Consider the time delay in the plant, $G(z) = G_n(z)z^{-N}$, z^{-N} represents the time delay of discrete time vehicle

model, where $G_n(z)$ is the nominal plant of $G(z)$. $C(z)$ is the digital robust PD controller which stabilizes the nominal plant $G_n(z)$. $Q(z)$ is the low pass filter. Based on the block diagram, the command regulation and disturbance rejection transfer functions can be derived in z domain as equation (19) (20), respectively.

$$\frac{Y(z)}{R(z)} = \frac{CGz^{-N}}{1+CG_nQ+CGz^{-N}(1-Q)} \tag{19}$$

$$\frac{Y(z)}{D(z)} = \frac{1+CG_nQ}{1+CG_nQ+CGz^{-N}(1-Q)} \tag{20}$$

In CDOB, Q filter is chosen as a low pass filter, with $Q \to 1$ and we can see that the time delay effect is eliminated from the closed-loop characteristic equation. In this way, the time delay is compensated by the CDOB.

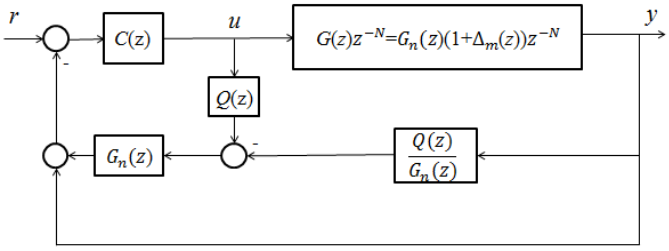


Figure 7. Digital CDOB compensated control system

V. Digital Multi-Objective Robust PD Controller Design

Parameter space approach is developed from continuous time domain to discrete time domain for digital robust PD controller design. The details of parameter space approach based discrete time multi-objective robust PD controller design can be found in reference [21]. For digital multi-objective robust PD controller design, phase margin constraint and mixed sensitivity constraint are taken into account simultaneously. Phase margin is defined as $PM \in [20,80]$ deg and the parameters for mixed sensitivity constraint are: low frequency bound l_s =0.5, the high frequency bound h_s =4, and the approximate bandwidth is $\omega_s = 5\text{rad/sec}$ for sensitivity weight function W_s; low frequency gain l_T =0.2, the high frequency gain h_T =1.8, and the frequency of transition to significant model uncertainty $\omega_T = 120$ rad/sec for complementary sensitivity weight function W_T. Figure 8 illustrates the k_d-k_p solution region and (k_d, k_p) design point is selected as (0.07, 0.2) as shown in red dot. It can be seen from Figure 9 that the corresponding frequency responses satisfy the phase margin constraint. Figure 10 shows the mixed sensitivity constraint is also satisfied with the chosen controller parameters as the magnitude plot is below 0dB ($(|W_sS| + |W_TT|) = 1$) line.

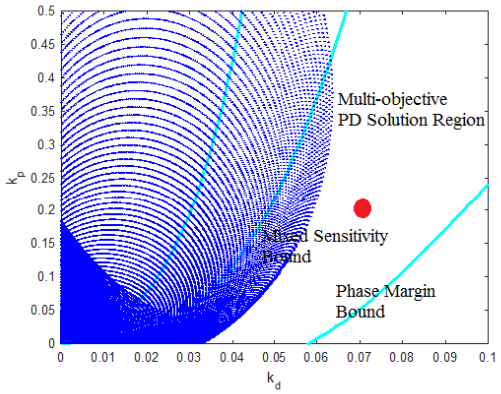


Figure 8. Multi-objective discrete time PD controller design

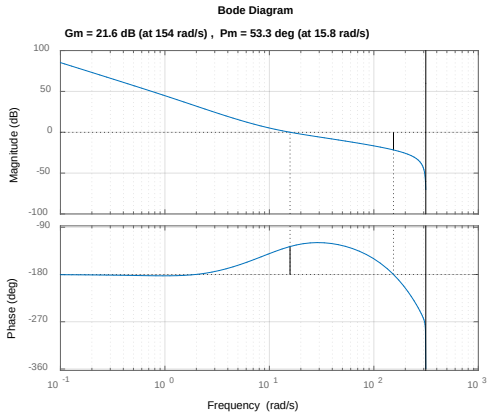


Figure 9. Phase margin constraint

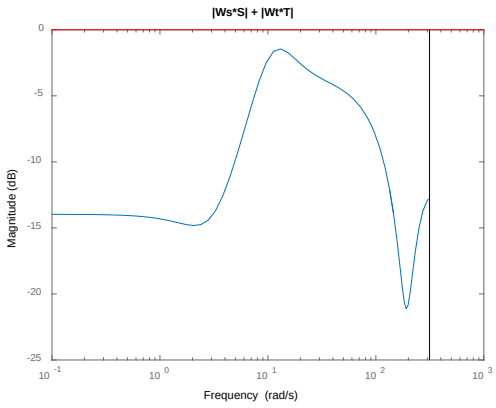


Figure 10. Robust performance plot

VI. Discretization of DOB and CDOB Structure

Discrete time DOB structure and CDOB structure can be derived using zero-order-hold (ZOH) method. The nominal plant G_n is the transfer function from front wheel steering angle δ_f to the lateral deviation y. According to Equation (5), the nominal plant G_n in continuous time domain is calculated as:

$$G_n(s) = \frac{4713s^2 + 1.598\times10^5 s + 7.51\times10^5}{s^2(1.242\, s^2 + 933.8s + 10610)}$$ (21)

For DOB structure, Q_{DOB} is designed in our previous work [22] in continuous time domain as following equation:

$$Q_{DOB}(s) = \frac{1}{0.25s^2+s+1} \quad (22)$$

By using ZOH method, sampling time T_s is given as 0.01 sec, $G_n(z)$ and $Q_{DOB}(z)$ are obtained as:

$$G_n(z) = \frac{0.04867\, z^3 - 0.07432\, z^2 + 0.02046\, z + 0.005954}{z^4 - 2.892\, z^3 + 2.784\, z^2 - 0.8927\, z + 0.0005429} \quad (23)$$

$$Q_{DOB}(z) = \frac{0.0001974z + 0.0001974}{z^2 - 1.96z + 0.9608} \quad (24)$$

For CDOB structure, the cutoff frequency of $Q_{CDOB}(s)$ is determined as 50 rad/s [22] and $Q_{CDOB}(s)$ can be expressed as equation (25), discrete time $Q_{CDOB}(z)$ is derived as equation (26):

$$Q_{CDOB}(s) = \frac{1}{0.0004s^2+0.04s+1} \quad (25)$$

$$Q_{CDOB}(z) = \frac{0.0902\, z + 0.06461}{z^2 - 1.213\, z + 0.3679} \quad (26)$$

VII. Simulation Studies

This section investigates autonomous vehicle path following performance of the proposed discrete time robust PD controlled system with DOB and CDOB compensation, respectively. An elliptical route is chosen as the desired path as shown in Figure 11 and the profile of 500 N crosswind force which acts as external disturbance is given in Figure 12. Vehicle initial position is at (0, 1) with 90° heading angle. The corresponding block diagrams for autonomous vehicle path following are given in Figure 4 and Figure 7, respectively. Matlab/Simulink is used as the simulation software. Sample time is 0.01 sec. Nominal plant *G*(*z*) is given as equation (23) for both Figure 4 and 7. The designed digital robust PD feedback controller in Section V is applied as baseline controller for all simulations. In the DOB compensated closed loop control structure, low pass filter *Q*(*z*) is determined as equation (24), and *Q(z)* for robust PD controlled system with CDOB compensation is given as equation (26).

Figure 13-17 present the autonomous vehicle lateral deviation of discrete time robust PD feedback controller system with DOB compensation at four corners of the parametric uncertainty box and in the existence of crosswind disturbance, respectively. For comparison purpose, corresponding simulation results of PD only feedback control system are also shown in each figure. It can be seen that discrete time DOB compensated system has less path following errors compared with PD only feedback control system, which verifies that discrete time robust PD with DOB compensated system effectively deals with model regulation and disturbance rejection. Table II compares root-mean-square (RMS) errors of discrete time PD control with DOB compensation and discrete time PD control only, which also shows better path following performance of DOB structure.

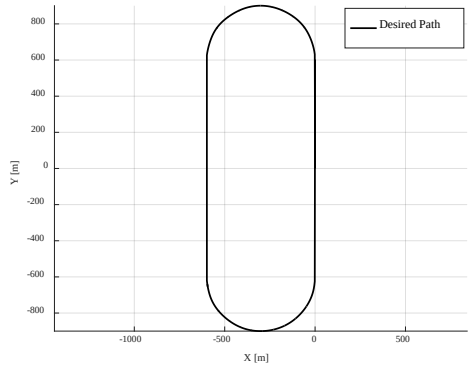


Figure 11. Desired path to be followed

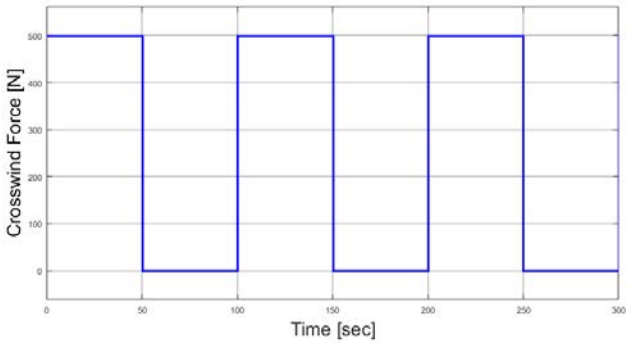


Figure 12. Crosswind force profile

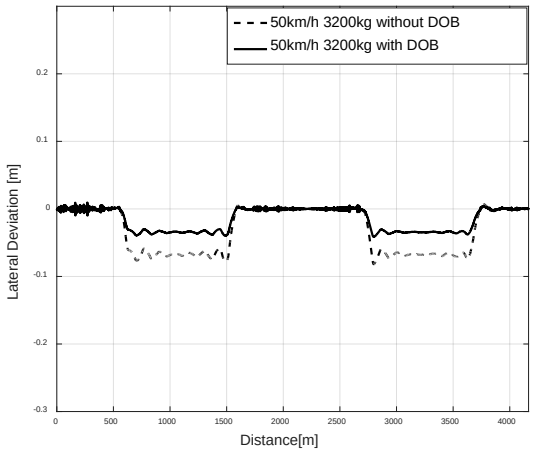


Figure 13. Lateral deviation with and without discrete DOB at corner *a*

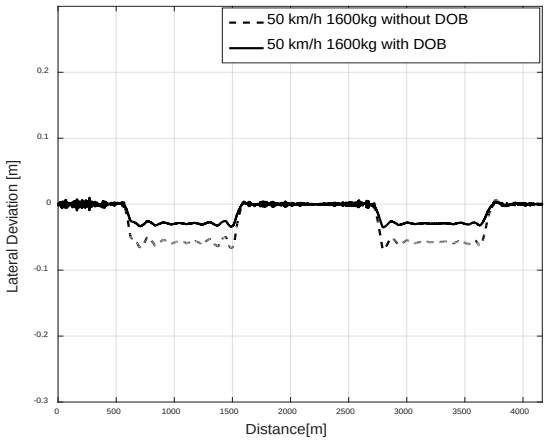


Figure 14. Lateral deviation with and without discrete DOB at corner *b*

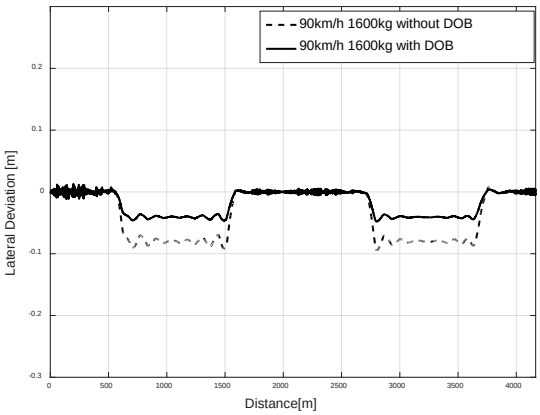


Figure 15. Lateral deviation with and without discrete DOB at corner *c*

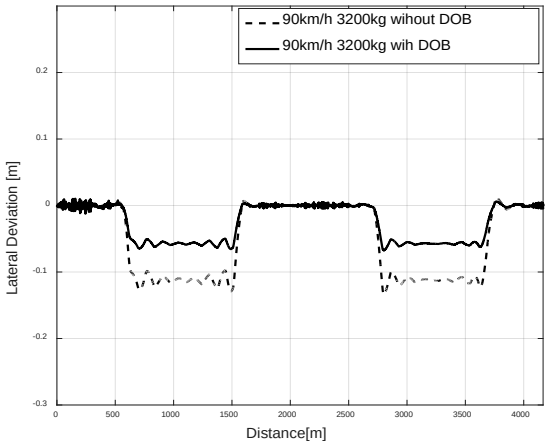


Figure 16. Lateral deviation with and without discrete DOB at corner *d*

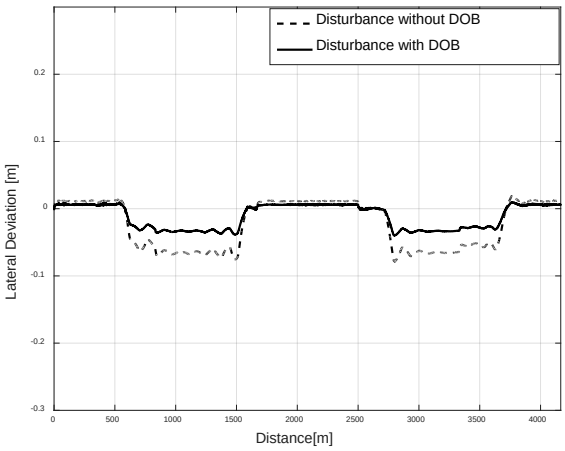


Figure 17. Lateral deviation with and without discrete DOB for crosswind input

Table II Comparison of RMS tracking errors between digital PD and digital PD with DOB

Operating Condition Control	50km/h 1600kg	50km/h 3200kg	90km/h 1600kg	90km/h 3200kg
PD	0.0384m	0.0451m	0.0525m	0.0739m
PD+DOB	0.0196m	0.023m	0.0268m	0.0377m

Figure 18 compares the autonomous vehicle lateral deviation of discrete time robust PD feedback controlled system with and without communication disturbance observer compensation by taking time delay N=1 sec into account. It is shown in Figure 18 that in the existence of time delay, PD feedback control system is not stable and oscillates, while CDOB compensation stabilizes the system and its lateral deviation is close to the one from nominal plant. Figure 19 presents the lateral deviation of CDOB compensated system for different time delay values. It is seen that path following errors do not increase with the increase of delay time. It can be concluded that discrete time CDOB compensates time delay and improves autonomous vehicle path following performance.

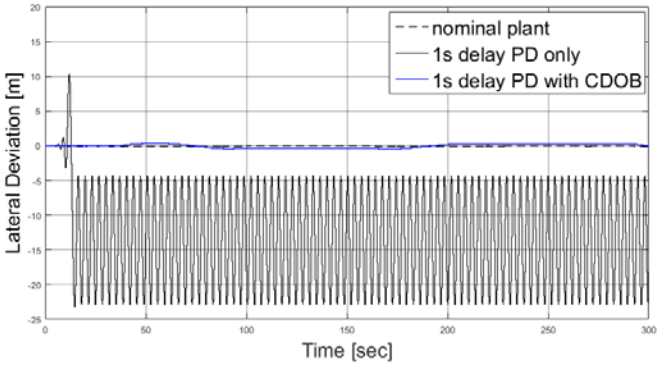


Figure 18. Lateral deviation with and without CDOB for time delay

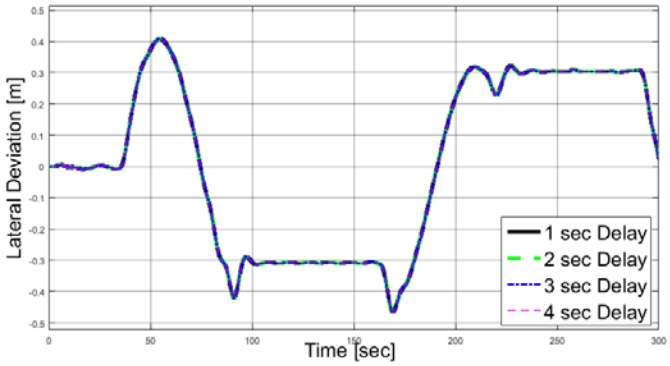


Figure 19. Lateral Deviations of CDOB compensation for different time delay

VIII. Conclusion

This paper applies robust PD feedback control with DOB/CDOB compensated system in discrete time domain and simulations are carried out to verify the effectiveness of the proposed structure. Digital robust PD controller is designed based on parameter space approach. Discrete-time DOB and CDOB structures are obtained by discretizing continuous time DOB and CDOB structures using zero order hold method. Autonomous vehicle path following driving at high speed is performed to validate the proposed discrete time DOB and CDOB structure. Simulation results show that the proposed discrete time DOB structure realizes model regulation and disturbance rejection, and discrete time CDOB effectively deals with time delay compensation. These prove the successful implementation of robust PD with DOB and CDOB compensated system in discrete domain. In the future work, hardware-in-the-loop simulations and experiments will be performed to further test the designed discrete-time DOB and CDOB systems.

References

[1] Suppachai, H., Silawatchananai, C., Parnichkun, M., et al. “Double loop controller design for the vehicle's heading control”[C]//Robotics and Biomimetics (ROBIO), 2009 IEEE International Conference on. IEEE, 2009: 989-994.

[2] Marino,R., Scalzi, S., Netto,M.,2011,“Nested PID steering control for lane keeping in Vision Based autonomous vehicles,” Control Engineering Practice,19, pp. 1459

[3] Utkin, V. I. "Sliding mode control design principles and applications to electric drives." IEEE transactions on industrial electronics 40.1 (1993): 23-36.

[4] Wang, H, Cao, Y. Aksun Güvenç,B., and Güvenç, L. "MPC Based Automated Steering of a Low Speed Shuttle for Socially Acceptable Accident Avoidance." In ASME 2016 Dynamic Systems and Control Conference, pp. V002T30A004-V002T30A004. American Society of Mechanical Engineers, 2016.

[5] Ackermann, J., 2012. “Robust control: the parameter space approach”. Springer Science & Business Media.

[6] Emirler, M.T., Wang, H., B. Aksun Güvenç, B., and Güvenç, L. "Automated robust path following control based on calculation of lateral deviation and yaw angle error." In ASME Dynamic Systems and Control Conference (DSCC), Columbus, OH, USA, October 28, vol. 30. 2015.

[7] Ohnishi, K., “A new servo method in mechatronics,” Trans. Japanese Soc.Elect.Eng.vol.107-D, pp.83-86,1987

[8] Senjyu, T., Shingo, A., and Katsumi, U., "Robust speed control of DC servomotors using fuzzy reasoning." In Industrial Electronics, Control, and Instrumentation, 1996., Proceedings of the 1996 IEEE IECON 22nd International Conference on, vol. 3, pp. 1365-1370. IEEE, 1996.

[9] Aksun Güvenç, B., and Güvenç, L. "Robustness of disturbance observers in the presence of structured real parametric uncertainty." In American Control Conference, 2001. Proceedings of the 2001, vol. 6, pp. 4222-4227. IEEE, 2001.

[10] Guvenc, Bilin Aksun, et al. "Robust two degree-of-freedom vehicle steering controller design." IEEE Transactions on Control Systems Technology 12.4 (2004): 627-636.

[11] Alevisakis, G., and D. E. Seborg. "An extension of the Smith predictor method to multivariable linear systems containing time delays." International Journal of Control 17, no. 3 (1973): 541-551.

[12] Matausek, M. R., and A. D. Micic. "A modified Smith predictor for controlling a process with an integrator and long dead-time." IEEE transactions on automatic control 41, no. 8 (1996): 1199-1203.

[13] Natori, K., Toshiaki T., Kouhei O., Aleš H., and Karel, J., "Robust bilateral control with internet communication." In Industrial Electronics Society, 2004.IECON 2004. 30th Annual Conference of IEEE, vol. 3, pp. 2321-2326. IEEE, 2004.

[14] Natori, K., Roberto, O., and Kouhei, O., "Stability analysis and practical design procedure of time delayed control systems with communication disturbance observer." IEEE Transactions on Industrial Informatics 4, no. 3 (2008): 185-197.

[15] Emirler, M.T., Aksun Güvenç,B., and Güvenç, L., "Communication disturbance observer approach to control of integral plant with time delay." In Control Conference (ASCC), 2013 9th Asian, pp. 1-6. IEEE, 2013.

[16] Uzunovic, T., Sariyildiz, E., & Sabanovic, A. (2018, March). "A discussion on discrete implementation of disturbance-observer-based control". In Advanced Motion Control (AMC), 2018 IEEE 15th International Workshop on (pp. 613-618). IEEE,2018

[17] Yun, H., Park, G., Shim, H., & Chang, H. J. (2016, July). "State-space analysis of discrete-time disturbance observer for sampled-data control systems". In American Control Conference (ACC), 2016 (pp. 4233-4238). IEEE,2016

[18] Lee, C., Joo, Y., & Shim, H. (2012). "Analysis of discrete-time disturbance observer and a new Q-filter design using delay function".

[19] Zhang, W., & Tomizuka, M. (2013). "Compensation of time delay in a network-based gait rehabilitation system with a discrete-time communication disturbance observer". IFAC Proceedings Volumes, 46(5), 555-562.

[20] Wang, H., Tota, A., Aksun-Guvenc, B., & Guvenc, L. (2018). "Real time implementation of socially acceptable collision avoidance of a low speed autonomous shuttle using the elastic band method". Mechatronics, 50, 341-355.

[21] Wang, H., Ph.D Dissertation, 2018

[22] Wang, H., & Guvenc, L. (2018). "Use of Robust DOB/CDOB Compensation to Improve Autonomous Vehicle Path Following Performance in the Presence of Model Uncertainty, CAN Bus Delays and External Disturbances" (No. 2018-01-1086). SAE Technical Paper.

Contact Information

Automated Driving Lab

930 Kinnear Road, Columbus, OH 43214

guvenc.1@osu.edu

Acknowledgments

This paper is based upon work supported by the National Science Foundation under Grant No.:1640308 for the NIST GCTC Smart City EAGER project UNIFY titled: Unified and Scalable Architecture for Low Speed Automated Shuttle Deployment in a Smart City, by the U.S. Department of Transportation Mobility 21: National University Transportation Center for Improving Mobility (CMU) sub-project titled: SmartShuttle: Model Based Design and Evaluation of Automated On-Demand Shuttles for Solving the First-Mile and Last-Mile Problem in a Smart City.

Definitions/Abbreviations

ZOH	Zero-order-hold
DOB	Disturbance observer
CDOB	Communication disturbance observer

2019-01-0493

A Unified, Scalable and Replicable Approach to Development, Implementation and HIL Evaluation of Autonomous Shuttles for Use in a Smart city

Xinchen Li, Sheng Zhu, Sukru Yaren Gelbal, Mustafa Ridvan Cantas, Bilin Aksun Guvenc, Levent Guvenc

Ohio State University

Abstract

As the technology in autonomous vehicle and smart city infrastructure is developing fast, the idea of smart city and automated driving has become a present and near future reality. Both Highway Chauffeur and low speed shuttle applications are tested recently in different research to test the feasibility of autonomous vehicles and automated driving. Based on examples available in the literature and the past experience of the authors, this paper proposes the use of a unified computing, sensing, communication and actuation architecture for connected and automated driving. It is postulated that this unified architecture will also lead to a scalable and replicable approach. Two vehicles representing a passenger car and a small electric shuttle for smart mobility in a smart city are chosen as the two examples for demonstrating scalability and replicability. For this purpose, the architecture in the passenger car is transferred to the small electric vehicle and used in its automation for demonstrating both scalability and replicability. High Level control and low level lateral control are presented in this paper. The parameter space based parametric control design approach that we are using to achieve scalable automated driving controllers is presented in the paper along with a discussion of how to evaluate performance and a brief description of the planned proof-of-concept test deployment.

Introduction

Solutions to autonomous vehicle and automated driving increasingly draw interest both from academia and industry, which largely stimulate the development of autonomous vehicles. The focus of Original Equipment Manufacturers (OEM) are mainly passenger vehicles due to its large market share, while small companies are concentrating more on small shuttles that can help in solving first-mile, last-mile problems. However, there are similarities behind the difference in applications on different platforms in some perspectives. The architecture which can be used on both high speed passenger vehicles and low speed shuttles is really beneficial for products as autonomous vehicles will be more demanding in the near future, with the development of autonomous vehicle technology and the needs for reducing fatalities in transportation related accidents.

Therefore, the various research efforts that have been conducted on the autonomous vehicle architecture that can be applied on high speed and low speed applications provide useful experience for application to new vehicle platforms. In our work, a unified architecture is proposed for autonomous driving and is improved based on our previous work upon applications on high speed passenger vehicle and low speed shuttle [1]. This unified approach shares the same hardware architecture and similar software library of different application functions, sensors, actuators and controls.

The essential part of this unified architecture is the replicability and scalability, especially the controller for these vehicles. Considering the supervisory controller to make sure that the right decision is made in different scenarios, it needs to meet the following requirements: feasible and adjustable for different vehicle platforms that have different parameters and quick response for handling inputs from different scenarios. Therefore, a standard decision-making framework with quick response is needed for such a unified architecture to realize its replicability and scalability. Also, a robust lateral controller is required for realizing the autonomous path following function in our unified architecture on high speed passengers vehicle and low speed shuttle.

In this paper, the unified architecture is introduced for high speed passengers vehicle and low speed smart shuttle. To meet the requirement for supervisory controller, a rule-based decision making framework is proposed here for decision making. This decision making framework processes sensor information and actuator information to generate supervisory control commands for different scenarios that the vehicles will meet during autonomous driving. We also talk about the robust lateral controller in this unified architecture based on parameter space approach for dealing with the challenge of parameter uncertainties on different vehicles for low level control. The feasibility of this unified architecture is demonstrated with simulations. A brief introduction is given upon our software library as well about the function and hardware integration.

The organization of this paper is as follows. In the second section, the library used in the unified architecture is introduced. It gives detailed information about both hardware integration and software library of available autonomous functions during autonomous driving. The third section presents the decision making framework and explains how it works in response to various scenarios. The fourth section describes the robust lateral controller for path following in the unified architecture. The next section shows the simulation results for implementing the controller on the unified architecture. High Level controller and low level controller are deployed separately and results are explained in detail related to the controller design. The final section gives conclusion and future goals for our unified architecture.

Unified Architecture and Library

In this section, we present the hardware library and software library on the unified architecture. An autonomous vehicle needs sensors to perceive its environment, processing units to process the data from these sensors and actuators to drive the vehicle without a human driver. All of these elements should be chosen properly to obtain the best result possible according to the determined goal. The main idea of the unified architecture is to design an autonomous vehicle structure that can be easily modified or adapted and implemented on another vehicle platform regardless size and type to achieve replicable and scalable autonomous driving. After the implementation, only necessary modifications to optimize the performance would be tuning the low-level controller parameters according to the vehicle and fine tuning of the autonomous driving functions for the new platform. Such an autonomous vehicle hardware architecture also showing the flow of information is shown in Figure 1 and has been easily adapted from our earlier work on driving automation.

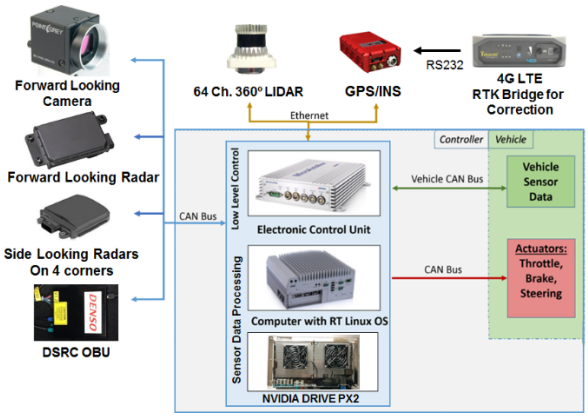


Figure 1. Unified Architecture.

As seen in Figure 1, sensors such as camera, radar, lidar and Leddar are chosen for perception. Data from these sensors are processed in a real time Linux Computer (mainly lidar data processing) and Drive PX2 (mainly image processing) using ROS and neural networks such as YOLO and SEGnet, then processed data is sent to the lower level control unit which in our case is a Microautobox. Pedestrians, other vehicles and obstacles are detected by these sensors and actions can be taken accordingly. An OBU is also added to provide DSRC communication capabilities so that V2X applications can be implemented. These capabilities can also be used to have solutions for NLOS scenarios or improving applications such as ACC to CACC. An important function in autonomous driving is localization. Localization is mainly taken care of by an OXTS RTK GPS sensor in our vehicles, which can provide up to 50 cm accuracy in non-RTK differential GPS operation. In addition, with the corrections coming from an RTK Bridge-X, this accuracy can go up to 4 cm, which is very accurate for this type of application. Since this architecture aims for robust autonomous driving, lidar sensor is also used with algorithms such as SLAM [2] and map-mapping matching to localize the vehicle, in case of GPS signal loss.

The main lower level control unit for the vehicle is the microautobox. Processed data from sensors are transferred into this real-time processing unit to go through decision making, low level control computations and converted into actuator commands such as steering angle, throttle and brake percentages. Software for this controller is designed in MATLAB Simulink and converted to C code, then compiled, using Simulink Coder and lastly embedded into the microautobox electronic control unit. Designing the software structure in Simulink provides advantages such as faster prototyping capability, flexibility, better readability and allows us to create a unified library with a similar idea of a unified architecture. The aim is to combine these two together to create a base that will allow any vehicle platform to have autonomous driving capabilities by using the unified software library together with the unified hardware architecture. Moreover, since the blocks are separately designed in Simulink, if there is a need for modification such as one sensor missing or an algorithm that needs to be improved, this modification can be directly managed through that block separately. The unified software library structure designed in Simulink is shown in Figure 2.

This Simulink software library is designed under several main titles. Localization blocks are used to receive localization data from different sensors, where some of them are processed and coming from processing units and some of them are directly measured and transferred to the microautobox.

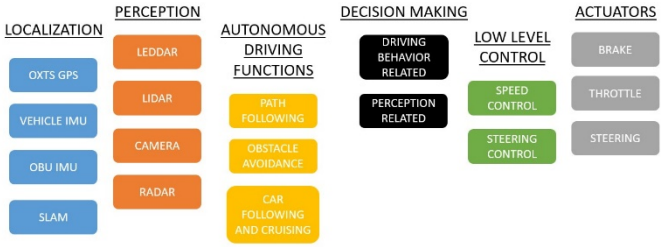


Figure 2. Unified Library.

Similarly, perception blocks are used to receive perception data from sensors or processed data from their processing units. Autonomous behavior algorithm blocks make necessary computations for autonomous driving algorithms such as path following and obstacle avoidance for lateral driving and car following and cruising for longitudinal driving. Decision making part has blocks that use all the information available to the computers such as localization, perception to determine the autonomous driving state in a finite state machine implementation of decision making. Deep learning algorithms used for decision making are also under that heading. Low-level control blocks are for calculating throttle, brake and steering commands required for executing the autonomous driving functions selected by decision making. Lastly, actuator blocks are used to transfer outputs from these low-level controllers to the real actuators inside the vehicle through communication such as CAN or analog signals.

This design is implemented with the hardware architecture and library part into two different sized vehicles to demonstrate, in a limited sense, replicability and scalability. One full size sedan hybrid vehicle (Ford Fusion) and one small size fully electric vehicle (Dash EV) are used for this purpose. While the sedan vehicle is aimed more for high speed autonomous driving applications, the smaller neigborhood electric vehicle is aimed for lower speed autonomous shuttle type driving

applications. These two vehicles with their implemented hardware are displayed in Figure 3. It can also be seen that the sensor and hardware placement are similar for both implementations.

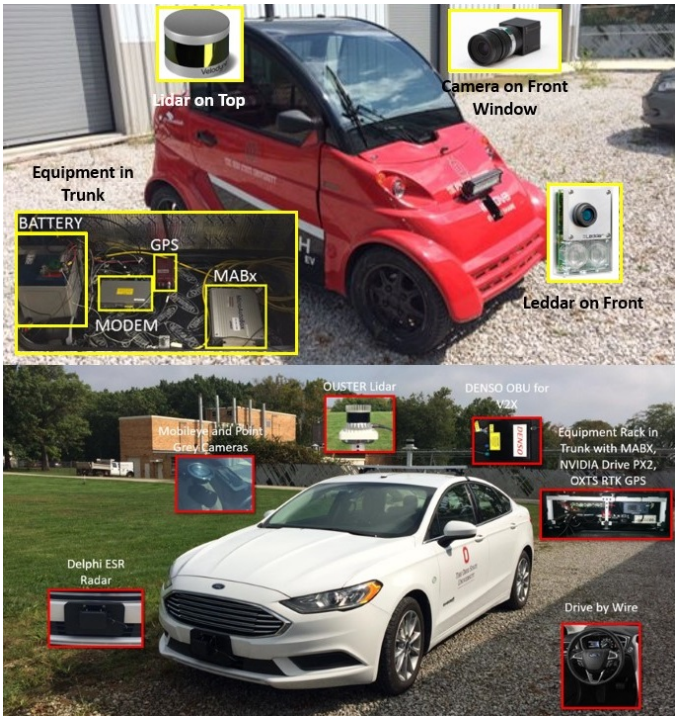


Figure 3. Unified Implementation

Decision Making Framework

At the core of autonomous vehicle control system, the supervisory controller plays a very important role in determining the appropriate state and subsequent course of action when faced with different situations. Decision making has been a research topic that has drawn broad attention and great progress is being made with the various methods that are deployed.

To take care of the uncertainties in the environment, many research efforts in decision making have utilized the probabilistic method for errors that occur in the perception part of autonomous vehicle. Markov Decision Process is one of the most popular methods. A lot of work has been done in decision making based on Markov Decision Process or POMDP [3,4] to deal with the uncertainties existing in perception and situational awareness. Also, since artificial intelligence is also commonly used and well developed, decision making based on machine learning has also been studied and implemented widely in autonomous driving [5].

Due to the heavy computation and uncertainties in the probabilistic method and machine learning method, the platform for supporting decision making has to have a powerful computer like the NVIDIA GPUs used in our hardware architecture for good performance. The rule-based decision making framework is used in this paper because of lower computation complexity and ease of implementation on most of the vehicle platforms we use. Work on replicable and scalable probabilistic or machine learning based decision making is in progress and will be reported in future papers. References [6,7] report such rule-based framework in the DARPA Urban Challenge for example.

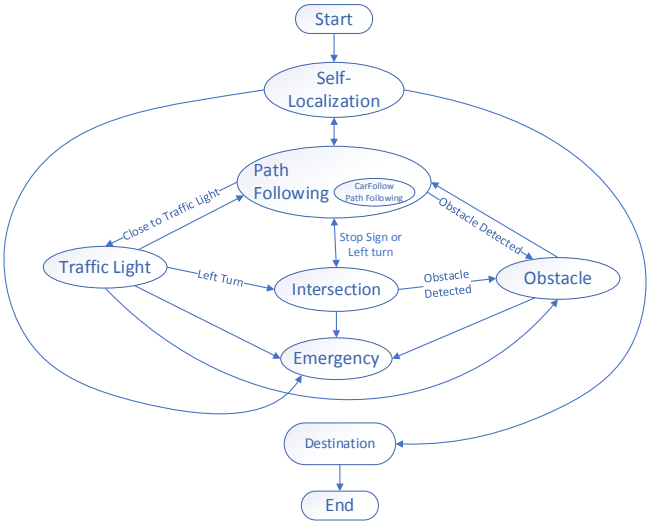


Figure 4. Diagram of Framework for Decision Making in the Unified Autonomous Vehicle Architecture w/ some of switching conditions shown

Figure 4 shows the diagram of our decision-making framework as a Finite State Machine (FSM). It is used to switch between different driving states while doing the task of shuttle or urban driving in a passengers vehicle. Switching between states is triggered by input from perception part generated by the sensors introduced into the hardware library of unified architecture, including desired path, ego-vehicle location referred to the path, traffic light signal, road signs and objects in the surrounding environment on the path. This FSM process has six driving states that will occur in urban driving and one emergency states. They are explained below.

- Start: This is the beginning of the autonomous driving task. The vehicle will receive information about the pre-planned path.
- Self-localization: This is the initial state of the vehicle in an autonomous driving task. It allows the vehicle to estimate its location on the designated route and learn about the knowledge of where to start in order to do path following. The self-localization is realized in two methods: the first is GPS localization using differential RTK GPS and RTK bridge. When the GPS signal is correctly received, it can accurately locate the vehicle on the path. When GPS localization fails or is not desired, pointcloud based localization will be utilized such as SLAM or map matching. Self-localization is done periodically to make sure that the path following task is accomplished.
- Path Following: This is the most commonly called state for autonomous driving. In this state, the vehicle follows the planned path at a certain speed based on the robust lateral controller. There is a sub-state in this path following, the Car follow path following. The function of this sub-state is similar to adaptive cruise control in highway driving. If a vehicle is detected in front of the ego-vehicle, this state will be triggered, then ego-vehicles will follow the front vehicle at its speed or our planned speed if its speed is higher than speed limit of the road. This state is a dominant state that we will be in most of the time to ensure our ego vehicle can do different autonomous driving tasks successfully and to successfully cooperate with other vehicles in the same lane. The function of path following

is realized by implementing robust controller with parameter space method on our unified architecture for autonomous vehicles. The control deployed in the unified architecture is replicable and scalable so that it can work with vehicles with different parameter and ensure the robust performance in path following maneuver.

- Intersection: The intersection state will be triggered from path following when a cross road and a stop sign is detected or when a left turn or right turn entry into a road is to be executed. In the intersection state, if there is no stop sign detected, ego-vehicle will go back to previous state and wait for another input for switching. If a stop sign is detected, the ego-vehicle will automatically wait for a few seconds and then do the crossing traffic check. Side radars and lidar sensors will be used in such a condition to detect crossing traffic when waiting at the stop sign. The waiting time can be determined by algorithm mentioned in [8]. After input of crossing traffic checking shows no vehicle is crossing within the range of sensor detection, FSM will be triggered back to path following state to continue vehicle motion. The other scenario that can triggered by switching into intersection state is that a left turn needs to be taken for path following. The way of determining whether we are going to take a left turn or not is by the curvature of upcoming path. Knowing the current location of ego-vehicles and the planned path, the curvature can easily be derived as an input of this framework. When left turn is needed, we will wait for 2 seconds and do crossing traffic detection. It will be switched back to path following once crossing traffic information is gained and no vehicle is passing by.
- Traffic Light: Traffic light state will be switched into as information of traffic light position is received through input and it is found that the ego-vehicle is closed to a traffic light. Apart from the traffic light position information, signal information is also collected and work as input for the traffic light state. Several sub-states lie within the traffic light state. Red light state is on when red light signal and yellow light signal is received. Then, the vehicle will stop and wait for several seconds then check the red light signal again. Green light state is triggered if green light signal is received. Green light state is an intermedia state between Traffic Light state and Path following state. As the green light state is triggered, our state will switch back to the Path Following state. Traffic light State helps the autonomous vehicle to pass the traffic light without stopping if possible. Further improvement can be made for this state with the help of other sensors, such as the modem on vehicle. In the near future, V2I communication will be widely implemented on the road such that road infrastructure can communicate with autonomous vehicle through “RSU Modem” communication. It will be easier and faster for detecting traffic light position information and traffic light signals while doing autonomous driving.
- Obstacle: The state of Obstacle is a maneuver for obstacle avoidance. There are possibilities that different objects are encountered on the road and block the lane in which the ego vehicle is moving. The decision making framework will switch into this state when an obstacle is detected in front of the vehicle. The distance of detecting is related with moving speed. The distance for detecting is the safe distance for braking. In this framework, a pedestrian is regarded as an obstacle as well. When a pedestrian is detected, the obstacle state is triggered and a stop maneuver is taken to slow down the vehicle. In this paper, obstacle avoidance maneuver is doing the brake and stop since the test route for this paper has only a single lane road. Later on, collision avoidance algorithm will be implemented to realize a better action for avoiding the obstacle.
- Emergency stop: This state is the emergency state to make sure nothing goes wrong while doing autonomous driving task. Sometimes, the controller may have unexpected error and send wrong low level control command to the vehicle. Thus, action will be made by human driver to switch the FSM to Emergency Stop state. In this state, the vehicle will be switching back to manual driving and will be stopped by the human driver. This function is realized if the emergency button is pushed or steering wheel, brake pedal or throttle pedal are touched by the human driver in the driver seat. This state can be triggered from any one of the other states in the FSM.

The transition in Finite State Machine triggered by input information make state of driving switch for decision making. It gives high level command to the vehicle so that the vehicle can react to different scenarios while autonomously driving. The sub states within different states can do maneuvers when specific sensor information and vehicle state are sent to the controller. Also, the emergency stop state can take care of exceptions to improve the safety of the autonomous driving task. This rule-based decision making framework has the advantage of low computational complexity and ease of modification and improvement.

States and rules can be added in the near future to deal with more complicated situations and the performance of this framework can also be improved with more sensing function added in the unified architecture. For example, the state “Stop for passenger” can be added in the future when the function of an on-demand ordering service is added to the shuttle vehicle. With 3D mapping and map matching technology, the accuracy of self-localization will be improved when the GPS signal is weak. A collision avoidance algorithm will enable the Obstacle state to make a maneuver to make a detour around the obstacle rather than stop and wait for the stopped obstacle to move.

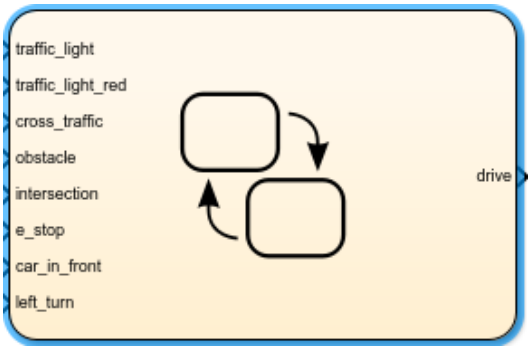
Figure 5 shows the deployment of finite state machine with state flow chart. Inputs of the chart are index of scenarios, including “traffic_light”, “red_light”, “cross_traffic”, “obstacle”, “obstacle” “e-stop”, “car_in_front” and “left_turn” from environment perception and vehicle path following controller. With the inputs, switching will occur as the switching condition is satisfied in the finite state machine. The output of this chart is the index of driving state corresponding to the current scenario. The output index is for one of the three basic driving states for autonomous path following function: path following, car follow with path following and the stop state.

Lateral Controller on Unified Architecture

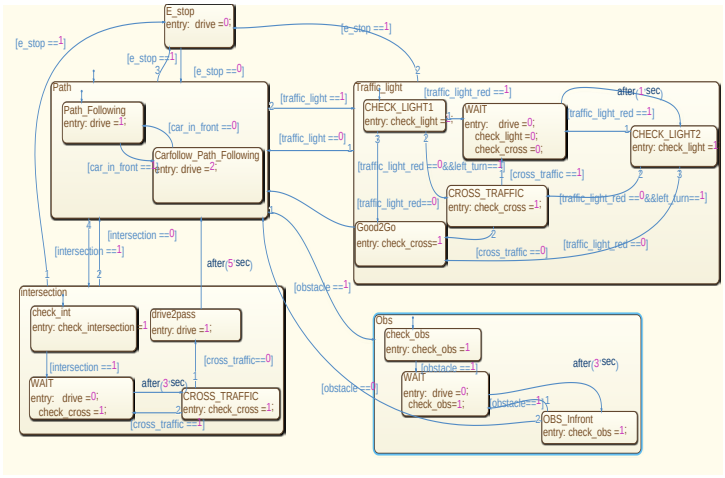
This section introduces the robust lateral control algorithm that we use on the unified architecture. Path following algorithm have been studied in different research [11]. With the lateral control algorithm, vehicles with our unified architecture can do autonomous path following function. A robust PD controller was designed as the lateral steering controller for the vehicle to track the planned path, as seen in our previous work [9]. A bicycle model of the vehicle presented in [10] was used to derive its deviation from the planned path. The controller has the following form with the look-ahead error *y* as feedback:

$$C(s) = k_p + k_d s \tag{1}$$

The uncertainty area of our two experimental platforms, Dash and Fusion, are shown in Figure 6 using vehicle specifications and working conditions.



(a)



(b)

Figure 5. (a) State Flow chart of the decision making framework. (b) Structure of the Finite State Machine in State Flow chart

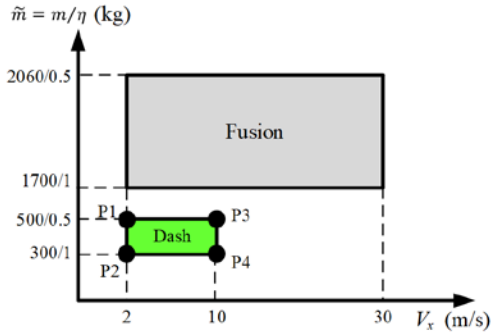


Figure 6. Uncertainty region of vehicle mass m, longitudinal speed V_x and parameter η for Dash and Fusion experiment platforms.

The design process takes Dash as an example with both D-stability and mixed sensitivity criterion into consideration. Satisfaction of the mixed sensitivity criterion ensures the robust performance, i.e. desired tracking performance and robustness to model uncertainties:

$$\left\| |W_S S| + |W_T T| \right\|_\infty < 1, \qquad (2)$$

where S is sensitivity function, T is complementary sensitivity function, W_S and W_T are weights for S and T respectively.

The D-stability boundaries and mixed sensitivity point conditions are reflected onto the design space of controller parameters k_p and k_d using the parameter space approach [9]. Figure 7 shows the parameter space at vertex P1 for the Dash vehicle. The colored lines in the plot are reflections of D-stability boundaries while the points of blue envelope curves were obtained by substituting a range of frequency values at critical condition of (2). Similarly the parameter space at the other three vertices were drawn. The PD controller parameters $k_p = 0.5$, $k_d = 0.035$ (red cross in Figure 7) were picked in the overlapped selectable region, satisfying the design requirement at all four uncertain vertices. Figure 8 shows the magnitude plot of $|W_S S| + |W_T T|$ at each vertex of the uncertainty region. It can be seen that the mixed sensitivity constraint (2) is met at those extremal points.

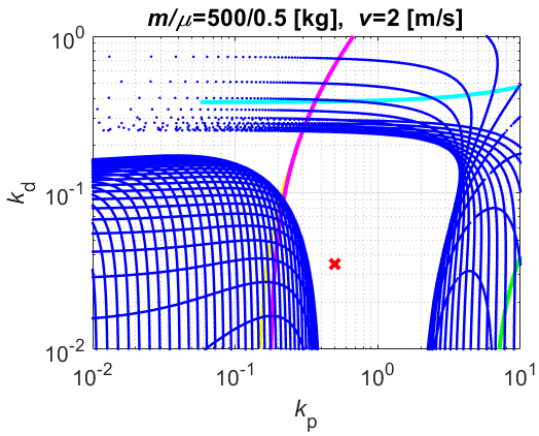


Figure 7. Parameter space at vertex P1

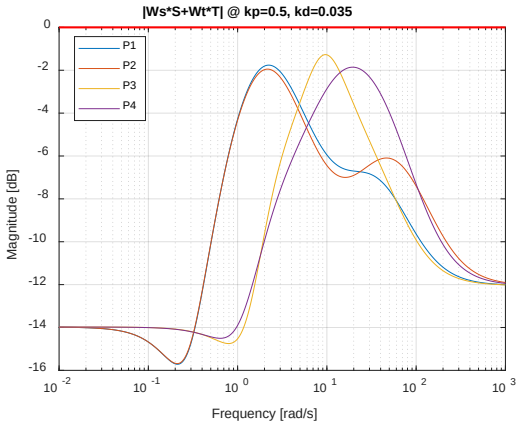


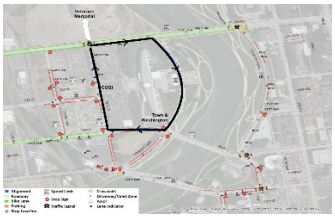
Figure 8. Robust performance satisfied with selected PD parameters at each apex

Simulation and result explanation

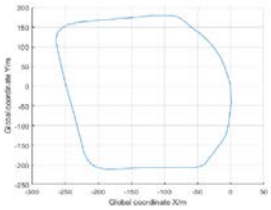
To evaluate the performance of the developed control strategy, along with both low level control, decision making and sensor placement, a simulation of controller deployment is required.

Simulation Environment of Decision Making Framework

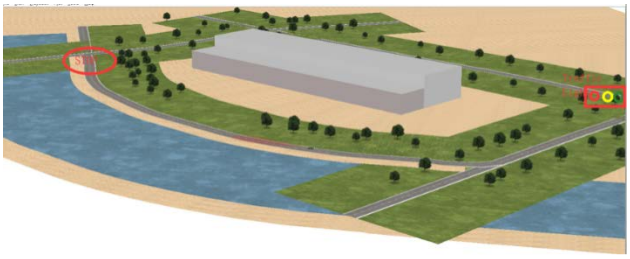
The simulation is done in the Carsim Environment with Carsim simulink co-simulation. In Carsim, the test route can be set up with other virtual vehicles to simulate the road environment that the vehicle will meet while driving in the real world. In this paper, the simulation is made on the route COSI from one of the smart city test routes in Columbus as shown in Figure 9 and using simulation vehicles as in Figure 10. Simulation results are presented with the simulation of both the low speed vehicle model and the high-speed passenger vehicle model while also showing the decision making FSM state.



(a)



(b)



(c)

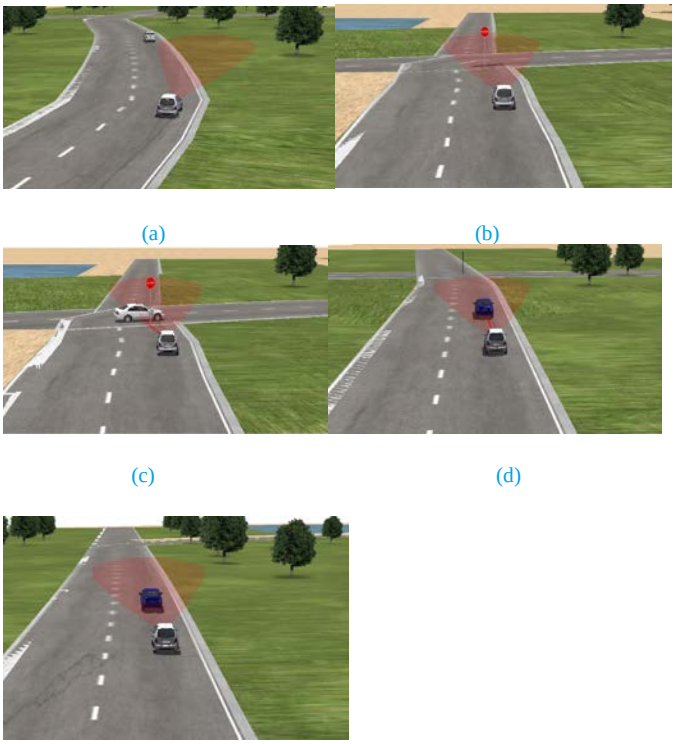
Figure 9. (a) COSI route in map (test route is enlightened); (b) plot of route in global frame; (c) Test route constructed in Carsim Simulation.

(a)

(b)

Figure 10. Carsim Simulation environment on different vehicle models; (a) Low speed vehicle model; (b) High speed vehicle model

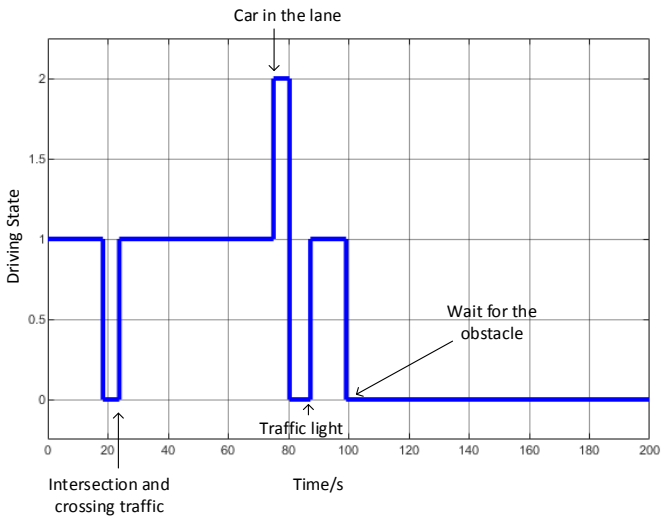
In the simulation of high-level controller, we tested the decision making framework with four scenarios: intersection with stop sign, traffic light with red light state, moving vehicle in the lane and obstacle encountered in the same lane. These scenarios in the simulation environment are presented in Figure 11.



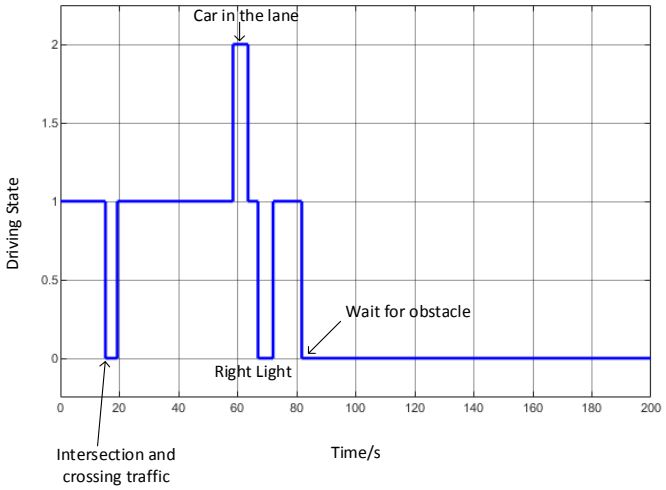
(a) (b) (c) (d) (e)

Figrue 11. Carsim scenarios: (a) Path Following in lane; (b) Intersection with stop sign; (c) Crossing traffic passing by at intersection;; (d) Car follow while doing path following; (e) Obstacle in the lane (use stopping vehicle represented as an obstacle in the lane)

While driving on the road facing different scenarios, the decision making framework will generate high level commands for the vehicle to respond to. As introduced in the previous section, inputs are index of the scenarios, and outputs are the driving states related to the commands. Here in the simulation, the outputs are set to be “1” for path follow, “2” for car follow with path following and “0” for stop and wait for other commands from the decision making framework. At the intersection, the vehicle waits for 3 seconds to check crossing traffic. When a vehicle is crossing, the Stop state will be triggered again for waiting. Then, a car in the same lane with the speed lower than speed limit triggers the decision making state flow switch to Car follow with path following state. The Stop drive state is switched to again when approaching a red traffic light as well as being close to the obstacle vehicle in front. The state vs time plot is shown in Figure 12. It shows how the simulation vehicles with our unified architecture respond to different scenarios. First, it comes to an intersection with the stop sign, the vehicle waits for 3 seconds after which it detects crossing traffic. The vehicle will keep waiting until the crossing traffic has left, then the drive state switches back to normal path following. Stop state is used due to red traffic light being on right after the Car follow with path following state. In the end, the vehicle stops and waits for the obstacle in the same lane.



(a)



(b)

Figure 12. Drive State vs time plot; (a) Drive State in low speed vehicle model simulation; (b) Drive State in high speed vehicle model simulation; (Due to the faster speed, driving time is shorter than the previous simulation)

The velocity for the low speed vehicle is set to be 10 m/s and the high speed vehicle velocity is set to be 12 m/s. Distance for vehicle speed decreasing is 25m and the distance for following a vehicle is $10 \sim 15$ m.

Simulation result for robust lateral control

Figure 13 shows the simulation results for a full run around the COSI route. Combining the lateral controller and decision making framework, simulation is made on the low shuttle vehicle on the test route for low speed application In this simulation, normal path following speed is set to be 5 m/s and speed will decrease to 1 m/s during cornering. The tracking performance of the vehicle is acceptable.

Conclusions and Future Work

From the simulation results and demonstration, we have shown that the controller, especially the Finite State Machine based Decision Making Framework can work well for the Unified Architecture for high speed autonomous vehicle and low speed autonomous shuttle.

However, more work needs to be done in the future. Functionality corresponding to the high level command needs to be improved such as better car following algorithm and collision avoidance. Localization based on 3D mapping will be studied as well.

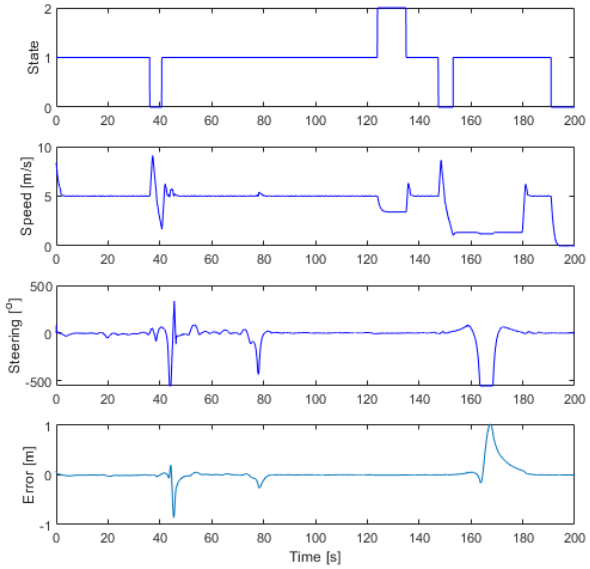


Figure 13. COSI pilot test route simulation results

References

1. Gelbal, S. Y., Chandramouli, N., Wang, H., Aksun-Guvenc, B., and Guvenc, L., "A unified architecture for scalable and replicable autonomous shuttles in a smart city." In Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on, pp. 3391-3396. IEEE, 2017.
2. Wen, B., Gelbal, S.Y., Aksun-Guvenc, B., and Guvenc, L., "Localization and Perception for Control and Decision Making of a Low Speed Autonomous Shuttle in a Campus Pilot Deployment." *SAE International Journal of Connected and Automated Vehicles*, JCAV-2018-0011R1
3. Ulbrich, S., and Markus M., "Probabilistic online POMDP decision making for lane changes in fully automated driving." In *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, pp. 2063-2067. IEEE, 2013.
4. Brechtel, S., Tobias G., and Rüdiger D., "Probabilistic decision-making under uncertainty for autonomous driving using continuous POMDPs." In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pp. 392-399. IEEE, 2014.
5. Chen, C., Seff, A., Kornhauser, A., and Xiao J., "Deepdriving: Learning affordance for direct perception in autonomous

driving." In Proceedings of the IEEE International Conference on Computer Vision, pp. 2722-2730. 2015.
6. Montemerlo, M., Becker, J., Bhat, S., Dahlkamp, H., Dolgov, D., Ettinger., Haehnel et al. "Junior: The stanford entry in the urban challenge." *Journal of field Robotics* 25, no. 9 (2008): 569-597.
7. Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., Hong. D. et al. "Odin: Team victortango's entry in the darpa urban challenge." *Journal of field Robotics* 25, no. 8 (2008): 467-492.
8. Cantas, M.R., Kavas, O., Tamilarasan, S., Gelbal, S.Y., Guvenç, L, 2019, "Use of Hardware in the Loop (HIL) Simulation for Developing Connected Autonomous Vehicle (CAV) Applications," WCX19: SAE World Congress Experience, April 9-11, Detroit, Michigan, Session AE 503 Vehicle to Infrastructure, SAE offer/paper number 19AE-0150, *under review*.
9. Zhu, S., Gelbal, S.Y., Li, X., Cantas, M.R., Aksun-Guvenc, B., and Guvenc, L., "Parameter space and model regulation based robust, scable and replicable lateral control design for Autonomous Vehicles," 2018 IEEE 57th Annual Conference on Decision and Control (CDC), Miami Beach, 2018.
10. Guvenc, L., Aksun-Guvenc, B., Demirel, B., and Emirler M.T., Control of Mechatronic Systems, Institution of Engineering and Technology, 2017.
11. Dadras, S., "Path Tracking Using Fractional Order Extremum Seeking Controller for Autonomous Ground Vehicle," SAE Technical Paper 2017-01-0094, 2017, https://doi.org/10.4271/2017-01-0094.

Contact Information

Xinchen Li li.6312@osu.edu

Automated Driving Lab

1320 Kinnear Rd, Columbus, OH, 43212

guvenc.1@osu.edu

Acknowledgments

This work was supported in part by the National Science Foundation under Grant 1640308, and by the U.S. Department of Transportation Mobility 21: National University Transportation Center for Improving Mobility (CMU) sub-project titled: SmartShuttle: Model Based Design and Evaluation of Automated On-Demand Shuttles for Solving the First-Mile and Last-Mile Problem in a Smart City. The authors acknowledge the support of NVIDIA for the donation of the two Drive PX2 systems used in our vehicles and for the donation of a Titan Pascal GPU that we use in training dep learning networks.

2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Online Quintic Path Planning of Minimum Curvature Variation with Application in Collision Avoidance*

Sheng Zhu, Sukru Yaren Gelbal, and Bilin Aksun-Guvenc, *Member, IEEE*

Abstract— **Path planning is a crucial task in automated driving. Due to the complexity of dynamically changing driving environment, the planned path generally requires the capability to adjust itself in real time to avoid obstacles detected in its way. The use of an optimization method is able to generate a collision-free and smooth path. However, its high computation burden limits its direct application to online path planning. This paper proposed a time-efficient online table-lookup approach to deal with this dilemma. Given discrete target points, this approach is capable to form a quintic-spline path with second-order geometric (G^2-) continuity using a look-up table. The look-up table was generated beforehand in the reference space with minimization on curvature variation. The paper demonstrates the application of this online approach in collision avoidance, with a geometry-based method to decide new target points when obstacles are detected in the original path. These new target points are fed to the table-lookup online path planning algorithm to generate a collision-free path with minimum curvature variation.**

I. INTRODUCTION

Path planning is a challenging task for autonomous driving in a dynamically changing environment. The planned path should be collision-free with surrounding obstacles in the environment, while also smooth enough for the benefits of path following and passenger comfort. A poorly designed path would not only make the subsequent path following difficult, but may also cause passenger discomfort with exceedingly large lateral acceleration and yaw oscillation. To satisfy this multi-objective task, the optimization approach seems a good fit here, which is capable to weigh the relative importance of each objective within imposed constraints of kinematics and obstacles [1-3].

The collision-avoidance scenario is extremely time sensitive and demands a quick path re-planning especially at high speed. However, optimization approach is known not time-efficient since the optimization of the objective function takes place at each motion state [4]. The complexity of the optimization problem greatly increases if the optimal objective and constraints imposed become complicated. For this reason, online implementation of optimization is still restricted with a trade-off between explicit representation and computational efficiency, and usually requires a high-performance in-vehicle PC for calculation [5, 6]. Meanwhile, the global optimum is difficult to find for a non-convex optimization problem. The optimization solution generally reduces to a local optimum in this case and may require another path selector to decide whether the planned path is acceptable or not [6, 7].

Instead of online optimization, path planning is usually divided into two phases, smooth path generation and afterward collision-free evaluation [8]. The optimization cost in the former phase can be reduced by precomputing lookup tables of path parameters offline. For example, [9, 10] parameterized control inputs and generated model-based trajectory from initial states to target ones via the 5D precomputed tables. The parameterization of these control inputs, however, is not inclusive sometimes [9].

This paper similarly adopts the table-lookup approach but instead describes the path as quintic polynomials with parameters optimized to minimize the curvature variation. The dimension of lookup tables required is reduced with the introduction of a reference space. In the reference space where initial and target endpoints are fixed, lookup tables of optimal path parameters to minimize the curvature variation are constructed offline using optimization method at different sets of orientation angles and curvature values. In online implementation, two endpoints in each segment are transformed into the reference space where linear interpolations of the lookup tables are completed. A de-transformation is followed afterwards to obtain the optimal path with G^2-continuity in real space. In this way, lookup tables up to four dimensions are sufficient within a small storage space.

Collision avoidance scenario is chosen to demonstrate the applicability of the table-lookup path planning technique. A low-computation, geometry-based method is proposed to realize collision prediction and new target points generation based on an occupancy grid map. The generated four new target points are connected by the table-lookup approach to form a smooth collision-free path with minimum curvature variation.

The paper is organized as follows: Section II demonstrates the table-lookup path planning approach. Section III introduces the geometry-based method to generate discrete target points in case of collision prediction. Section IV focuses on the steering controller for path following. This is followed by simulation results and discussion in a collision-avoidance scenario in Section V, combining the aforementioned techniques together.

*This work was supported in part by the National Science Foundation under Grant 1640308, and by the U.S. Department of Transportation Mobility 21: National University Transportation Center for Improving Mobility (CMU) sub-project titled: SmartShuttle: Model Based Design and Evaluation of Automated On-Demand Shuttles for Solving the First-Mile and Last-Mile Problem in a Smart City.

S. Zhu and B. Aksun-Guvenc are with Center for Automotive Research and with Department of Mechanical and Aerospace Engineering, Ohio State University, OH 43212 USA (e-mail: zhu.1473@osu.edu; aksunguvenc.1@osu.edu).

S. Y. Gelbal is with Center for Automotive Research and with Department of Electrical and Computer Engineering, Ohio State University, OH 43212 USA (e-mail: gelbal.1@osu.edu).

II. PATH PLANNING

A. G^2-quintic Splines

The path in this paper is described by a polynomial spline, with each segment expressed as a quintic polynomial $p(\lambda)$, $\lambda\in[0, 1]$:

$$\mathbf{p}(\lambda)=\begin{bmatrix}x(\lambda)\\ y(\lambda)\end{bmatrix}:=\begin{bmatrix}x_0+x_1\lambda+x_2\lambda^2+x_3\lambda^3+x_4\lambda^4+x_5\lambda^5\\ y_0+y_1\lambda+y_2\lambda^2+y_3\lambda^3+y_4\lambda^4+y_5\lambda^5\end{bmatrix}, \quad (1)$$

where x_i, y_i, (i=0, 1, …, 5) are polynomial coefficients to be determined. To determine these coefficients for each segment, the position, orientation and curvature information of the two endpoints are required (Fig. 1):

$$\mathbf{p}(0)=\mathbf{p}_A, \quad \mathbf{p}(1)=\mathbf{p}_B, \quad (2a\text{-}b)$$

$$\hat{\mathbf{e}}(0)=\begin{bmatrix}\cos\theta_A\\ \sin\theta_A\end{bmatrix}, \quad \hat{\mathbf{e}}(1)=\begin{bmatrix}\cos\theta_B\\ \sin\theta_B\end{bmatrix}, \quad (2c\text{-}d)$$

$$\kappa(0)=\kappa_A, \quad \kappa(1)=\kappa_B. \quad (2e\text{-}f)$$

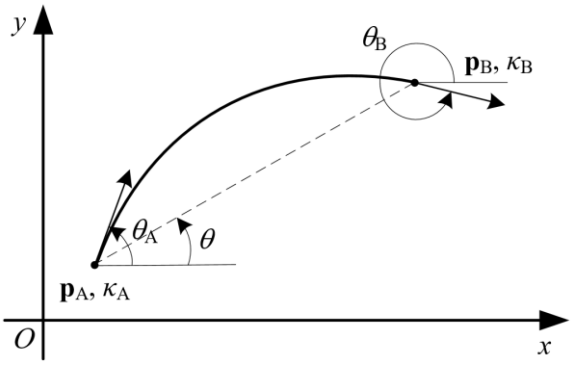


Figure 1. Interpolation conditions of the Endpoints

It has been shown in [11] that if the polynomial coefficients could be expressed with respect to $\boldsymbol{\eta}:=[\eta_1\ \eta_2\ \eta_3\ \eta_4]^T\in\Omega:=(0,\infty)\times(0,\infty)\times(-\infty,\infty)\times(-\infty,\infty)$ as follows, then the interpolation constraints are always fulfilled for any $\boldsymbol{\eta}\in\Omega$:

$$x_0=x_A, \quad (3a)$$

$$x_1=\eta_1\cos\theta_A, \quad (3b)$$

$$x_2=\frac{1}{2}\left(\eta_3\cos\theta_A-\eta_1^2\kappa_A\sin\theta_A\right), \quad (3c)$$

$$x_3=10(x_B-x_A)-\left(6\eta_1+\frac{3}{2}\eta_3\right)\cos\theta_A-\left(4\eta_2-\frac{1}{2}\eta_4\right)\cos\theta_B+\frac{3}{2}\eta_1^2\kappa_A\sin\theta_A-\frac{1}{2}\eta_2^2\kappa_B\sin\theta_B, \quad (3d)$$

$$x_4=-15(x_B-x_A)+\left(8\eta_1+\frac{3}{2}\eta_3\right)\cos\theta_A+(7\eta_2-\eta_4)\cos\theta_B-\frac{3}{2}\eta_1^2\kappa_A\sin\theta_A+\eta_2^2\kappa_B\sin\theta_B, \quad (3e)$$

$$x_5=6(x_B-x_A)-\left(3\eta_1+\frac{1}{2}\eta_3\right)\cos\theta_A-\left(3\eta_2-\frac{1}{2}\eta_4\right)\cos\theta_B+\frac{1}{2}\eta_1^2\kappa_A\sin\theta_A-\frac{1}{2}\eta_2^2\kappa_B\sin\theta_B, \quad (3f)$$

$$y_0=y_A, \quad (4a)$$

$$y_1=\eta_1\sin\theta_A, \quad (4b)$$

$$y_2=\frac{1}{2}\left(\eta_3\sin\theta_A+\eta_1^2\kappa_A\cos\theta_A\right), \quad (4c)$$

$$y_3=10(y_B-y_A)-\left(6\eta_1+\frac{3}{2}\eta_3\right)\sin\theta_A-\left(4\eta_2-\frac{1}{2}\eta_4\right)\sin\theta_B-\frac{3}{2}\eta_1^2\kappa_A\cos\theta_A+\frac{1}{2}\eta_2^2\kappa_B\cos\theta_B, \quad (4d)$$

$$y_4=-15(y_B-y_A)+\left(8\eta_1+\frac{3}{2}\eta_3\right)\sin\theta_A+(7\eta_2-\eta_4)\sin\theta_B+\frac{3}{2}\eta_1^2\kappa_A\cos\theta_A-\eta_2^2\kappa_B\cos\theta_B, \quad (4e)$$

$$y_5=6(y_B-y_A)-\left(3\eta_1+\frac{1}{2}\eta_3\right)\sin\theta_A-\left(3\eta_2-\frac{1}{2}\eta_4\right)\sin\theta_B-\frac{1}{2}\eta_1^2\kappa_A\cos\theta_A+\frac{1}{2}\eta_2^2\kappa_B\cos\theta_B, \quad (4f)$$

In this way, the quintic polynomial curve in (1) given the interpolation constraints is reduced to four design freedoms with the parameter $\boldsymbol{\eta}\in\Omega$. The optimization problem is formulated to minimize the maximum curvature variation:

$$\min_{\eta\in\Omega}\left(\max_{\lambda\in[0,1]}\left|\frac{d\kappa}{ds}(\lambda)\right|+w\cdot s(1)\right), \quad (5)$$

subject to

$$\left\|\dot{p}(\lambda)\right\|>0, \quad \forall\lambda\in[0,1], \quad (6a)$$

$$\eta_1-\eta_2=0, \quad (6b)$$

$$\eta_3+\eta_4=0, \quad (6c)$$

where s is the curvilinear length measured along the path $p(\lambda)$,

$$s(\lambda)=\int_0^\lambda\left\|\dot{\mathbf{p}}(\zeta)\right\|\mathrm{d}\zeta. \quad (7)$$

The latter item in the objective function adds penalty to the planned path length $s(1)$ weighted by w. The constraint (6a) is added to guarantee the regularity of the curve [11]. It is not difficult to show that the parameter $\boldsymbol{\eta}$ reflects the rate of change of curvilinear length s (or ds/dλ) with respect to λ:

$$\eta_1=\frac{\mathrm{d}s}{\mathrm{d}\lambda}(0)=\left\|\dot{\mathbf{p}}(0)\right\|, \quad \eta_2=\frac{\mathrm{d}s}{\mathrm{d}\lambda}(1)=\left\|\dot{\mathbf{p}}(1)\right\|, \quad (8a\text{-}b)$$

$$\eta_3=\frac{\mathrm{d}^2s}{\mathrm{d}\lambda^2}(0)=\frac{\mathrm{d}}{\mathrm{d}\lambda}\left\|\dot{\mathbf{p}}(0)\right\|, \quad \eta_4=\frac{\mathrm{d}^2s}{\mathrm{d}\lambda^2}(1)=\frac{\mathrm{d}}{\mathrm{d}\lambda}\left\|\dot{\mathbf{p}}(1)\right\|. \quad (8c\text{-}d)$$

Constraints (6b and 6c) are added to expect symmetric distribution of ds/dλ in $u\in[0, 1]$ intuitively and simplify the solution to the above optimization problem.

The minimization of curvature variation is set as the objective in order to reduce the steering variation, since the steering angle δ_f is directly related to the travelled curvature κ for a single-track vehicle model,

$$\delta_f=\left(L+k_{us}V_x^2\right)\kappa, \quad (9)$$

where L is the vehicle wheelbase, k_{us} is the understeer gradient coefficient determined by vehicle mass distribution and tire cornering stiffness coefficients, and V_x is the vehicle longitudinal speed. Therefore, a planned path with less varying curvature is desired with smoother steering and few lateral acceleration perturbations.

The minimax optimization problem could be transformed to a semi-infinite programming form as seen in [11]. The solution to this problem can be pursued by converting semi-infinite constraints to a set of finite constraints with peak values and then using the sequential quadratic programming. However, the solution to this semi-infinite programming problem comes with heavy calculation load and is hence not time-efficient. It also may give local optimum and sometimes even doesn't guarantee a solution if the initial condition is poorly selected. These negative factors together limit its direct application in online path planning even though the path generated is ideal with minimum curvature variation.

B. Table-lookup Path Planning

Due to the difficulty of direct online implementation of the optimization method, the table-lookup approach is raised with the idea to obtain suboptimal path parameters from the lookup tables constructed beforehand with extensive offline programming. However, the different combination of coordinates, orientation and curvature values of the two end points (Fig. 1) requires a look-up table up to six dimensions with infinite choices of values.

To reduce the complexity of dimensionality and save storage space, a reference space is formed where coordinates of endpoints A' and B' are fixed, for example at (0, 0) m and (10, 0) m respectively. Lookup tables of path parameters were constructed with respect to four-dimensional inputs of orientation angles and curvature values of the endpoints. The orientations $\theta_{A'}$ and $\theta_{B'}$ are defined in the range of [-π/4, π/4] rad, and the curvature $\kappa_{A'}$ and $\kappa_{B'}$ in the range of [-0.1, 0.1], each with a number of evenly divided grid of 20. The semi-infinite optimization problem (5) was solved offline at each combination of the defined orientations and curvature values. Fig. 2 shows the results of path trajectories, curvature and curvature variation value along the path at five different combinations of $\theta_{A'}$, $\theta_{B'}$, $\kappa_{A'}$, and $\kappa_{B'}$. Larger ranges of these lookup parameter values are unnecessary since the original endpoints problem can be divided with more medium points to fit inside the range of $\theta_{A'}$, $\theta_{B'}$, $\kappa_{A'}$, and $\kappa_{B'}$ after transformation into the reference space.

From these optimization solutions, the path parameter $\boldsymbol{\eta}$' instead of the polynomial coefficients, x_i' and y_i' (i=0, …, 5) were stored into a total of four four-dimensional (4D) lookup tables to reduce table size.

Fig. 3 shows the table-lookup path planning procedure. A transformation of the original endpoints into the reference space is required first. The constraints, i.e. orientation angles and curvature values, of the transformed endpoints A' and B' are then referred to the 4D lookup tables to obtain parameter values $\boldsymbol{\eta}$' through linear interpolation. This path found in the reference space is then reflected back to the original coordinate system through de-transformation process.

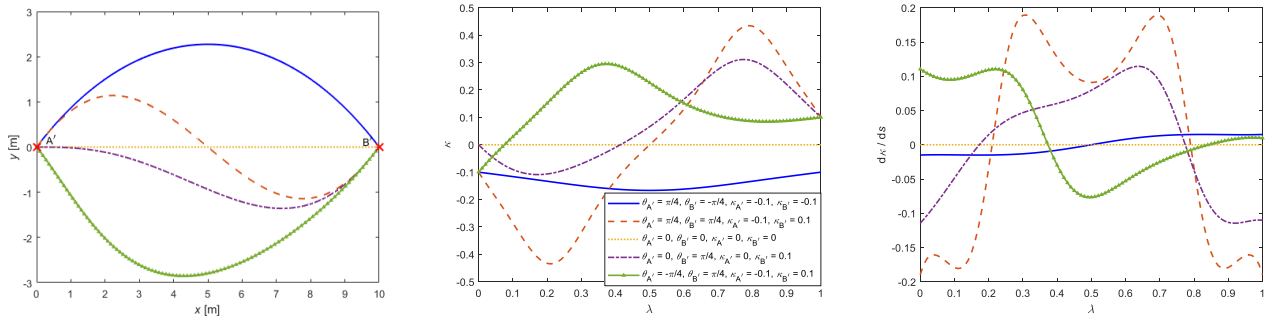


Figure 2. (a) Optimal polynomial curve, (b) curvature κ(λ), and (c) curvature variation dκ/ds(λ) for some orientation angles and curvature of endpoints A' and B' in the reference space (legend shown in (b))

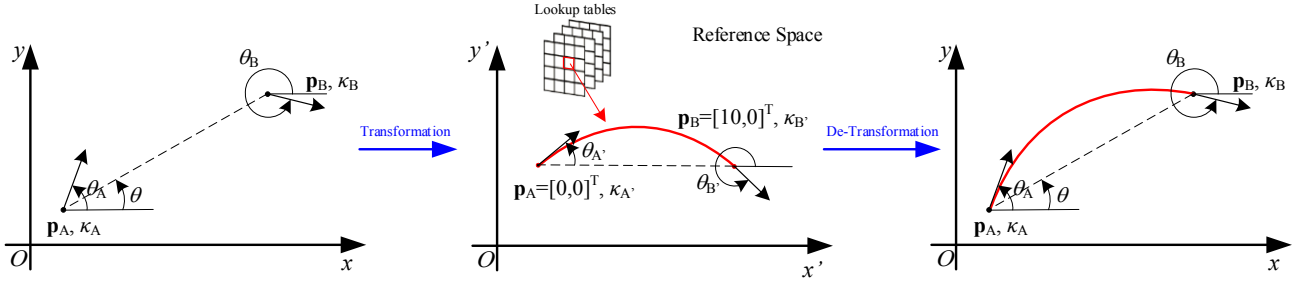


Figure 3. Table-lookup path planning procedure

1) Transformation

To reflect the endpoints A and B to their counterparts A' and B' fixed in the reference space, coordinate scaling and rotation are required. The orientation angles and curvature values of the endpoints after transformation are:

$$\theta_{A'} = \theta_A - \theta\,, \qquad \theta_{B'} = \theta_B - \theta\,, \tag{10a-b}$$

$$\kappa_{A'} = \kappa_A \cdot \frac{\|AB\|}{\|A'B'\|}\,, \quad \kappa_{B'} = \kappa_B \cdot \frac{\|AB\|}{\|A'B'\|}\,, \tag{10c-d}$$

where θ is the angle of line AB with respect to the abscissa axis (Fig. 3), and the Euclidean distance ||AB|| and ||A'B'|| are used to scale the curvature values into the reference space.

2) Table lookup

The $\theta_{A'}$, $\theta_{B'}$, $\kappa_{A'}$, and $\kappa_{B'}$ after transformation are fed to the 4D lookup tables to get the path parameter values of $\boldsymbol{\eta}$' through linear interpolation. The linear interpolation includes merely simple algebraic calculation and is not detailed here. Polynomial coefficients, x_i' and y_i' (i=0, 1, …, 5), can then be determined from parameter $\boldsymbol{\eta}$' with (3) and (4) respectively.

3) De-transformation

A de-transformation is necessary to reflect the polynomial coefficients, x_i' and y_i' (i=0, 1, …, 5), in the reference space back to the original coordinate systems. The polynomial curve in the real space has the relation below with its counterpart in the reference space:

$$\mathbf{p}(\lambda) = \begin{bmatrix} x(\lambda) \\ y(\lambda) \end{bmatrix} = \begin{bmatrix} x_A \\ y_A \end{bmatrix} + \frac{\|AB\|}{\|A'B'\|} \cdot \mathbf{R}_\theta \begin{bmatrix} x'_0 + x'_1\lambda + x'_2\lambda^2 \\ y'_0 + y'_1\lambda + y'_2\lambda^2 \end{bmatrix} + \begin{bmatrix} x'_3\lambda^3 + x'_4\lambda^4 + x'_5\lambda^5 \\ y'_3\lambda^3 + y'_4\lambda^4 + y'_5\lambda^5 \end{bmatrix}, \tag{11}$$

where $\mathbf{R}_\theta$ is the rotation matrix

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}. \tag{12}$$

The path parameters, x_i and y_i (i=0, 1, …, 5), for the original two endpoints problem can then be found easily by expressing (11) as

$$\mathbf{p}(\lambda) = [\mathbf{M}]_{2\times 6} \begin{bmatrix} 1 & \lambda & \cdots & \lambda^5 \end{bmatrix}^{\mathrm{T}}, \tag{13}$$

where $\mathbf{M}$ is the 2×6 matrix which contains the polynomial coefficients x_i and y_i (i=0, 1, …, 5).

Fig. 4 shows that the suboptimal solution of the table-lookup path planning approximate very close to the optimization result with endpoints given at $\mathbf{p}_A = [11, 99]^T$, $\mathbf{p}_B = [-10, -10]^T$, $\theta_A = -86.5^o$, $\theta_B = -117.2^o$, $\kappa_A = 0$, and $\kappa_B = 0$ as an example. Similar results are obtained at other endpoints conditions, validating the applicability of the table-lookup approach in path planning. Besides, it takes much shorter time to implement, i.e. 7.3 μs on average in Simulink Desktop Real-Time™ environment, compared to semi-infinite programming (Table I).

TABLE I. TIME COST COMPARISON

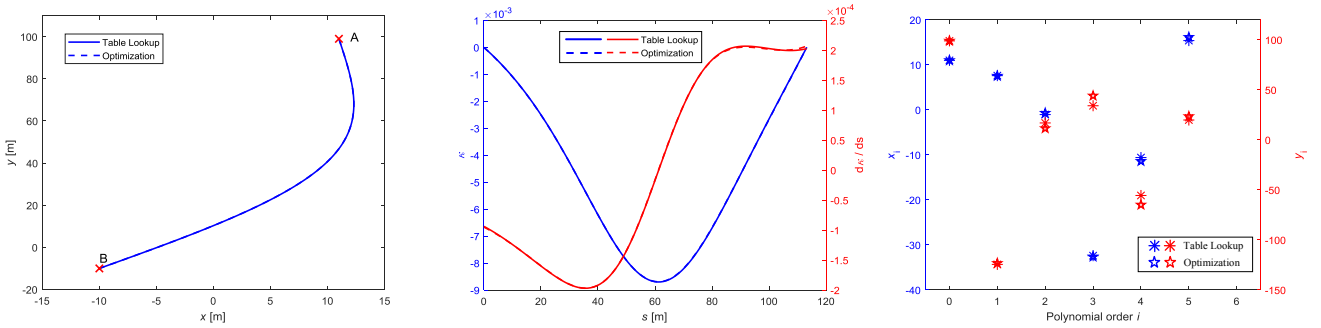
	Semi-infinite programming	Table-lookup approach
Time cost	178.8 ms	7.3 μs

III. GEOMETRY-BASED COLLISION-FREE TARGET POINTS GENERATION

The proposed table-lookup path planning requires the information of discrete target points. These preview target points need to be generated dynamically to provide the vehicle the freedom to avoid obstacles in real time. There are several ways of generating these target points, such as elastic band [12, 13], and potential field methods [7, 14]. These two methods maintain the vehicle at a relative safe distance from the obstacle based on the defined elastic force or potential field. However, the absolute distance of the path from the obstacles are generally uncertain in the plan phase. Depending on the tuned algorithm parameters, the generated target points could be too conservative or close to the obstacles, neither of which is desirable for a collision-avoidance behavior. For this reason, a geometry-based method is proposed with ascertained absolute distance from obstacles and have the benefit of low computation cost.

A. Collision Prediction

A two-dimensional local occupancy grid map needs to be formed first with each grid cell size of 0.5m×0.5m. In real world case, this grid map can be formed using sensors such as camera, radar and LIDAR. Information from these sensors could be used to update the map continuously while the vehicle is on the move, with the use of different packages



(a) (b) (c)

Figure 4. Comparison of table-lookup approach and optimization solution: (a) Path trajectory; (b) curvature κ and its variation dκ/ds; (c) polynomial coefficients x_i and y_i.

available in ROS. Some of the packages that can be used are the simultaneous localization and mapping (SLAM) algorithm [15, 16] packages such as Gmapping and Hector SLAM packages. A visualization of the LIDAR data that can be used for grid mapping is shown in Fig. 5 as an example, with data recorded with a Velodyne 16-channel LIDAR. This sensor will be used in real world implementation and experiments in future work.

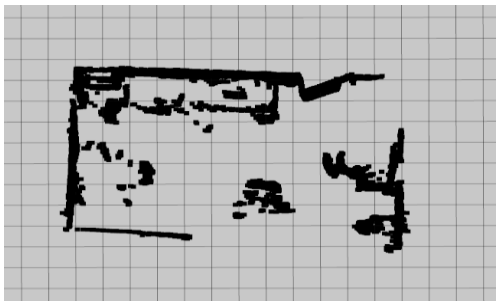


Figure 5. ROS visualization of LIDAR Data with surrounding environment

The planned path assumes the vehicle as a point mass. To account for the size of the vehicle itself to avoid collisions, the grid areas of the objects in the grip map are enlarged to provide some conservativeness. The controller calculates a preview path trajectory with 100 m look-ahead distance and judges if potential collision is expected. The collision prediction algorithm first evaluates the grid cells where the future path goes through, and then checks if any of these pass-through grid cells coincide with any cell the obstacles occupy. If any coincidence is observed as seen in Fig. 6, potential collision would be expected and a new path is in need to avoid the obstacle.

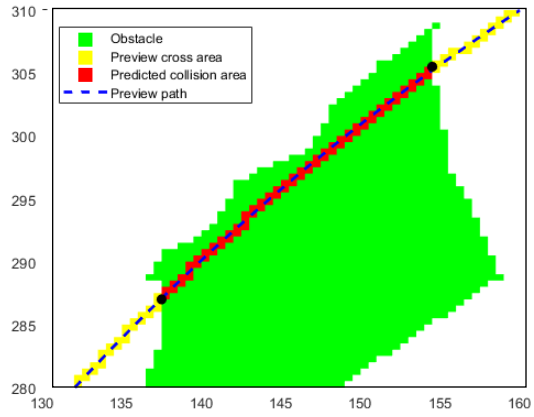


Figure 6. Preview path collides with obstacles

B. Collision-free Target Points Generation

If potential collision is expected through the collision prediction module, then new collision-free target points need to be generated. Assuming P and Q are the corners where the future path enters and exits the obstacle (Fig. 7), the start and end target points A and B can be found in the original path with a travel distance s before P and after Q respectively:

$$s = \max\{V_x \Delta t,\ 5m\}, \tag{14}$$

where Δt is a time constant and 5m is the minimum allowable reserved distance. Notice that in application the decision-making module also needs to be designed and decide whether to stop or pass the obstacle considering other traffic. Longitudinal speed profile should be designed accordingly. This paper mainly focuses on online path planning so decision-making and speed profile design are not covered here.

The intermediate target points are found as follows. The idea is to shift line PQ parallelly with step size 0.5m until a collision-free parallel line l is found (Fig. 7). [17] directly chose a single target point E closest to the final parallel line l to form a two-segment evasive path together with points A and B. However, if the obstacle is not convex as in the case of Fig. 7, the planned path A-D-B might still collide with the obstacle and requires another or more path planning procedures afterwards. These extra computations can be avoided by designing a three-segment evasive path instead. The idea is to find the widest potential collision points C' and E' along the shift direction and reflects these two points to the parallel line l as C and E. The evasive path A-C-E-B is then ascertained to avoid the obstacle with one computation.

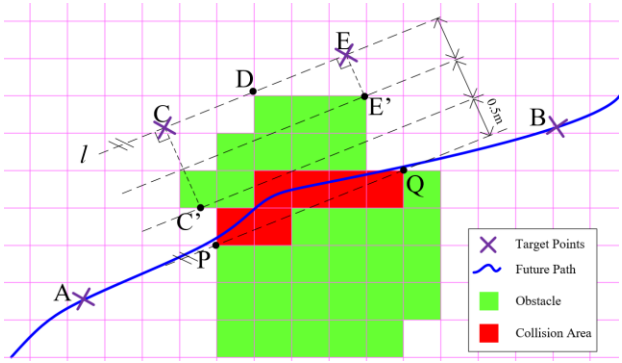


Figure 7. Illustration of collision-free target points generation

Besides the locations of the collision-free target points, orientation angles and curvature information at these points are still needed before linear interpolation to the lookup tables generated in Section II to form the path. The orientation and curvature values for points A and B are known from the original planned path. For points C and E, orientation angles are set to the angle of line l with respect to the abscissa axis and curvatures to 0 in order to form a straight path segment CE.

C. Online Path Planning and Update

Given the target points A, B, C and E generated with the geometry-based method, each segment could be formed with the two neighboring endpoints through the table-lookup procedure: transformation, lookup, and de-transformation (Fig. 3). The three segments together form a spline with second-order geometric (G^2) continuity, because curvature at the endpoints and in between are both continuous.

To update the path information, the neighboring segments before target point A and after point B are also re-planned because the original path segments (the mth and nth segment shown in Fig. 8) were broken apart by A and B. Therefore, a total of only five segments are constructed with the table-lookup online path planning method. The five segments are inserted between the original (m-1)th and (n+1)th

segments to form the new path with updated segment order (Fig. 8).

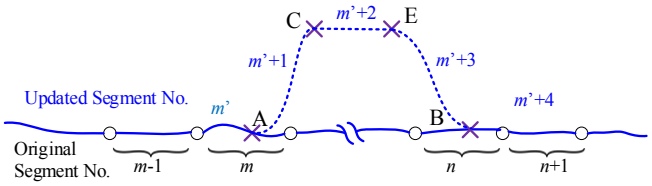


Figure 8. Original and updated path segments along with their orders

The updated path information is then used to form a new preview path trajectory. The process of collision prediction and re-planning is then continued based on the new preview path. Therefore, dynamically moving object and more obstacles could be taken into account with the proposed table-lookup online path planning approach.

IV. Path Following Control

The path following control determines the steering angle δ_f at the front wheel and is comprised of three parts,

$$\delta_f = \delta_{\text{feedforward}} + \delta_{\text{proportional}} + \delta_{\text{integral}} \quad . \tag{15}$$

where

$$\delta_{\text{feedforward}} = \left(L + k_{\text{us}} V_x^2\right)\kappa \ , \tag{16}$$

$$\delta_{\text{proportional}} = -k_p y = -k_p \left[e + l_s \sin\left(\Delta\psi\right)\right], \tag{17}$$

$$\delta_{\text{integral}} = -k_i \int_0^t e \, \mathrm{d}t \ , \tag{18}$$

L is the vehicle wheelbase length, e is lateral error, $\Delta\psi$ is heading angle error, y is the look-ahead error, k_p is the proportional feedback gain, and k_i is the integral coefficient to reduce the steady-state tracking error.

The proportional gain k_p and the look-ahead distance l_s were selected with the parameter space approach [18, 19] at each speed to ensure robustness over uncertainty range of vehicle mass $m \in [1700,\ 2060]$ kg and tire saturation coefficient $\eta \in [\eta(V_x),\ 1]$. Fig. 9 shows the selected values of (l_s, k_p) at speed 5m/s in the parameter space where D-stability region, steering overshoot and peak lateral deviation contours are shown at uncertainty apices. Similarly we choose the values of (l_s, k_p) at other speeds (Fig. 10) to form a speed-dependent gain-scheduling path following control strategy.

V. Results

Simulation was carried in Simulink to validate the table-lookup online path planning technique along with the geometry-based target points generation method in the scenario of collision avoidance. A high-fidelity vehicle model in CarSim with parameters customized to match our experiment vehicle Ford Fusion was used during the simulation. The scenario was created at a typical road curve with an obstacle in the original planned path for the vehicle. The obstacle has an irregular shape in order to evaluate the

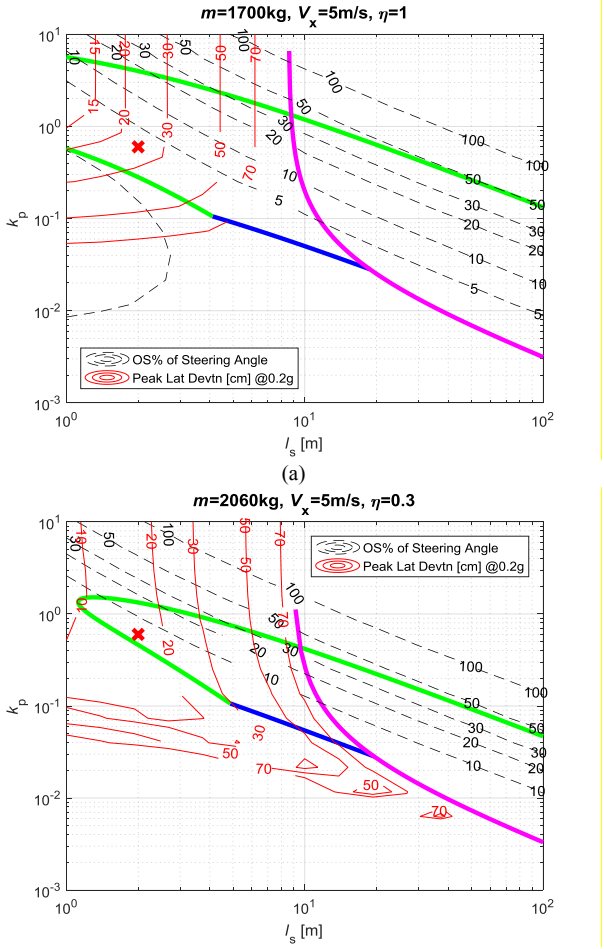


Figure 9. D-stability and performance contours mappings in the parameter space of (l_s, k_p) of two uncertainty apices at 5m/s as an example (Red cross represents the picked values of (l_s, k_p) at this speed)

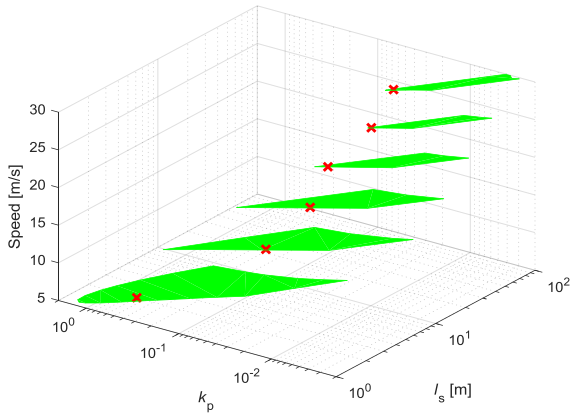


Figure 10. Robust D-stability region and (l_s, k_p) chosen at each speed

capability of the online path planning to decide an appropriate surpass distance.

Fig. 11 shows the result of the aforementioned online path planning technique with the original and updated path, vehicle trajectory, and the obstacle highlighted in an occupancy grid map. The vehicle followed its original planned path and online path planning was inactive until the

vehicle reached the asterisk position (Fig. 11) where the preview path was found in collision with the detected obstacle. Four new targets labeled in cross (Fig. 11) were generated based on the obstacle shape with the geometry-based method along with their orientation and curvature information. These target points were interpolated into the lookup tables generated before to form a collision-free path with minimum curvature variation. A total of five new segments in solid line (Fig. 11) were constructed with the table-lookup approach and in replacement of the original path along with an updated segment order.

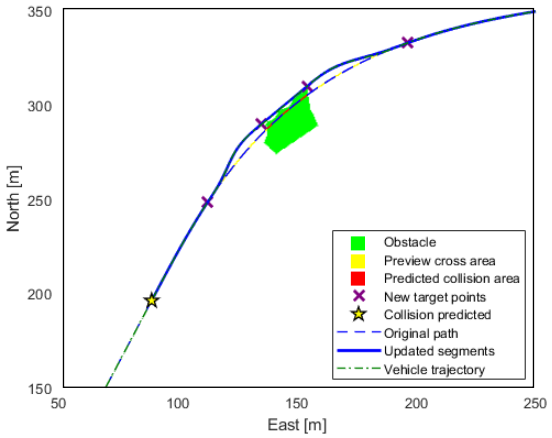


Figure 11. Online path planning result with an obstacle at a road curve

Fig. 12 shows a zoom-in of Fig. 11 where the cross area of the original preview path and the predicted collision area with the obstacle are highlighted. It can be seen that the two generated intermediate target points have accounted for the widest potential collision in the path shift direction with the proposed geometry-based method. The vehicle trajectory also shows good following performance with steering control at the speed of 15 m/s.

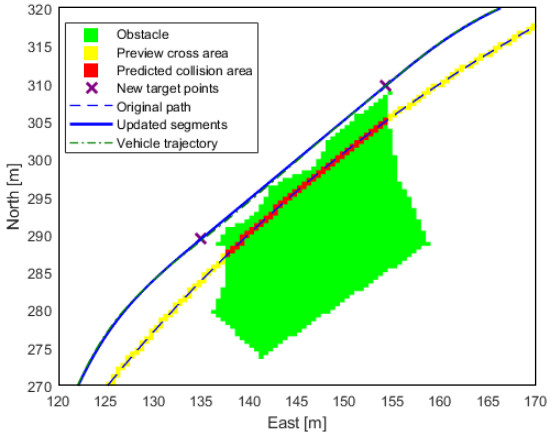


Figure 12. A zoom-in of Fig. 12 showing the widest potential collision points were accounted for an irregular-shaped obstacle with the geometry-based method

The updated path shows a smooth transition from the original path to the updated segments to evade the obstacle (Fig. 11). Curvature change is continuous along the updated path (Fig. 13), with a minimization of curvature variation ensured by the beforehand generated lookup tables with optimization approach. The smooth curvature change is desired as it directly affects steering as seen in (9). The segment nodes before and after path updating were also shown in Fig. 13 to illustrate the path updating process in Fig. 8.

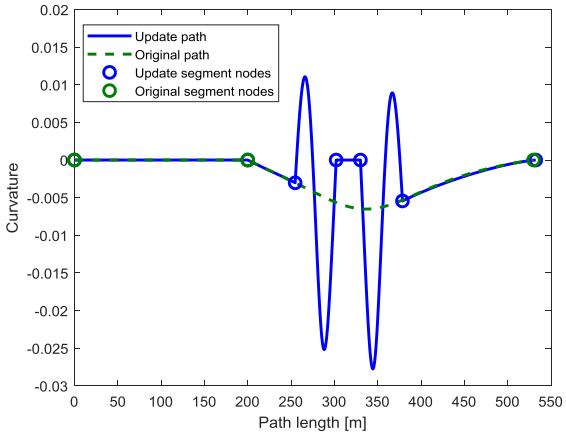


Figure 13. Curvature of the original and updated path and their corresponding segment nodes

Fig. 14 shows the steering and path following performance for the collision avoidance scenario. The steering along with yaw rate and lateral acceleration is smooth thanks to the G^2-continuity of the updated path, which improves passenger comfort. Besides, the lateral error and heading angle error are within acceptable range, validating the effectiveness of the designed path following controller.

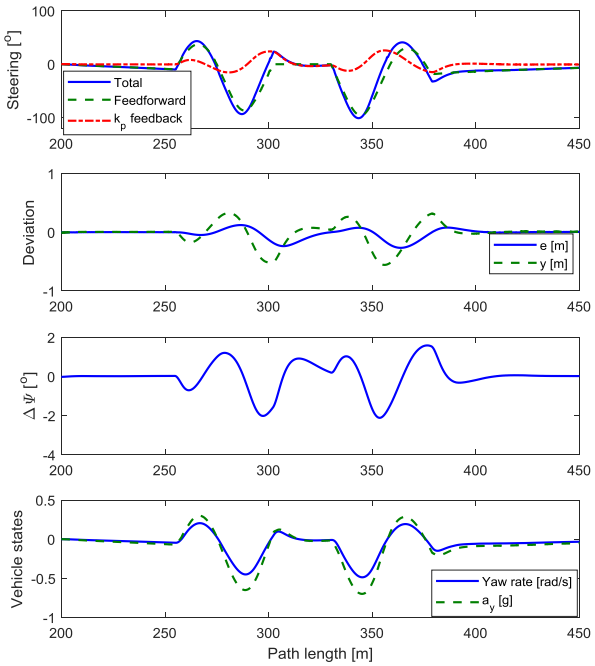


Figure 14. (a) Steering control, (b) lateral and look-ahead error, (c) heading angle error, (d) vehicle yaw rate and lateral acceleration during the collision avoidance at 15 m/s

VI. CONCLUSION

A time-efficient table-lookup path planning technique was proposed and developed. In combination with a geometry-based target points generation method, this approach is capable to update a collision-free path to avoid obstacles. Although not shown in the simulation here, the technique is potential to expand the scenarios with more obstacles or dynamically changing obstacles. A future experiment will be conducted with our experiment vehicle to validate the real-time performance of the table-lookup path planning approach.

REFERENCES

[1] J. Ziegler, P. Bender, T. Dang and C. Stiller, "Trajectory planning for Bertha—A local, continuous method," in *Intelligent Vehicles Symposium Proceedings*, 2014 IEEE, pp. 450-457, 2014.

[2] T. Gu and J.M. Dolan, "On-road motion planning for autonomous vehicles," in *International Conference on Intelligent Robotics and Applications*, pp. 588-597, 2012.

[3] X. Li, Z. Sun, D. Cao, D. Liu and H. He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles," *Mechanical Systems and Signal Processing*, vol. 87, pp. 118-137, 2017.

[4] D. González, J. Pérez, V. Milanés and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 1135-1145, 2016.

[5] C. Katrakazas, M. Quddus, W. Chen and L. Deka, "Real-time motion planning methods for autonomous on-road driving: State-of-the-art and future research directions," *Transportation Research Part C: Emerging Technologies*, vol. 60, pp. 416-442, November 2015. 2015.

[6] L.B. Cremean, T.B. Foote, J.H. Gillula, G.H. Hines, D. Kogan, K.L. Kriechbaum, J.C. Lamb, J. Leibs, L. Lindzey and C.E. Rasmussen, "Alice: An information- rich autonomous vehicle for high- speed desert navigation," *Journal of Field Robotics*, vol. 23, pp. 777-810, 2006.

[7] D. Dolgov, S. Thrun, M. Montemerlo and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *The International Journal of Robotics Research*, vol. 29, pp. 485-501, 2010.

[8] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny and G. Hoffmann, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, pp. 661-692, 2006.

[9] D. Ferguson, T.M. Howard and M. Likhachev, "Motion planning in urban environments," *Journal of Field Robotics*, vol. 25, pp. 939-960, 2008.

[10] X. Li, Z. Sun, D. Cao, D. Liu and H. He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles," *Mechanical Systems and Signal Processing*, vol. 87, pp. 118-137, 2017.

[11] C.G.L. Bianco and A. Piazzi, "Optimal trajectory planning with quintic G/sup 2/-splines," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, pp. 620-625, 2000.

[12] H. Wang, A. Tota, B. Aksun-Guvenc and L. Guvenc, "Real time implementation of socially acceptable collision avoidance of a low speed autonomous shuttle using the elastic band method," *Mechatronics*, vol. 50, pp. 341-355, 2018.

[13] M.T. Emirler, H. Wang and B. Aksun-Guvenç, "Socially acceptable collision avoidance system for vulnerable road users," in *IFAC Control in Transportation Systems*, vol. 49, pp. 436-441, 2016.

[14] L. Tang, S. Dian, G. Gu, K. Zhou, S. Wang and X. Feng, "A novel potential field method for obstacle avoidance and path planning of mobile robot," in *3rd IEEE International Conference on Computer Science and Information Technology*, pp. 633-637, 2010.

[15] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J.Z. Kolter, D. Langer, O. Pink and V. Pratt, "Towards fully autonomous driving: Systems and algorithms," in *IEEE Intelligent Vehicles Symposium*, pp. 163-168, 2011.

[16] T. Luettel, M. Himmelsbach and H. Wuensche, "Autonomous ground vehicles—Concepts and a path to the future," in *Proceedings of the IEEE*, vol. 100, no. Special Centennial Issue, pp. 1831-1839, 13 May 2012.

[17] L. Han, H. Yashiro, H.T.N. Nejad, Q.H. Do and S. Mita, "Bezier curve based path planning for autonomous vehicle in urban environment," in IEEE Intelligent Vehicles Symposium, pp. 1036-1042, 2010.

[18] L. Guvenc, B. Aksun-Guvenc, B. Demirel and M.T. Emirler, *Control of Mechatronic Systems*, Institution of Engineering and Technology, 2017.

[19] J. Ackermann, *Robust control: the parameter space approach*, Springer Science & Business Media, 2012.

2018 IEEE Conference on Decision and Control (CDC)
Miami Beach, FL, USA, Dec. 17-19, 2018

Parameter Space and Model Regulation based Robust, Scalable and Replicable Lateral Control Design for Autonomous Vehicles*

Sheng Zhu[a], Sukru Yaren Gelbal[b], Xinchen Li[b], Mustafa Ridvan Cantas[b], Bilin Aksun-Guvenc[a], *Member, IEEE* and Levent Guvenc[a,b], *Member, IEEE*

Abstract—**This paper introduces a unified, scalable and replicable approach to make implementation of the autonomous system on a new vehicle faster while preserving its autonomous performance. The main idea of this approach is to create a standard hardware architecture, along with a Simulink or similar library and templates for autonomous driving for a unified approach to vehicle autonomy, making it easier to scale the solution and replicate it on other vehicle platforms. However, this scaling and replicating of the autonomous driving system between vehicles remains difficult especially for low-level controller design due to parametric difference between vehicles. This paper, hence, demonstrates a sequential controller design procedure with specific example of lateral control for a chosen vehicle. The same design process can be replicated to adapt controller parameters for other vehicles. The parameter space approach is applied here to ensure robust path following performance of a proportional-derivative (PD) steering controller, considering uncertainties of vehicle load, speed and tire cornering stiffness. To further reduce the tracking error and handle unmodeled dynamics and reject disturbances, a model regulator was added based on overall system analysis. To evaluate the control strategy, a validated high-fidelity model of an autonomous research vehicle is used within a hardware-in-the-loop (HIL) simulation environment. Soft sensors were also connected to the soft automated vehicle in the HIL environment to test high-level control and decision making mechanisms. The road used for the simulations is a replica of a designated real world short AV pilot route in the Ohio State University West campus. Traffic is generated with Simulation of Urban MObility (SUMO) software in order to analyze the problems due to the presence of other vehicles and evaluate performance more realistically in the HIL simulator.**

I. Introduction

Solutions to autonomous driving or advanced driver assistance systems (ADAS) continues to grow interest from research and industry. Traditional Original Equipment Manufacturers (OEMs) and software-based technology companies have mainly focused on passenger vehicle solutions due to its largest market share, while start-up companies have been working mostly on niche markets like low-speed autonomous shuttles operating on a fixed route. Regardless of different applications, their research and product development involve similarity and redundancy in some aspects. Meanwhile, as autonomous driving technology is advancing and series production would be expected in the near future, extensive testing processes for autonomous vehicles will be necessary. A standardized in-the-lab testing process is crucial to ensure that all autonomous driving functions operate as planned before proceeding with public road testing and deployment.

Therefore, a common and unified architecture of road vehicle autonomy that is easily scalable and replicable is beneficial for fast product development, saving development resources, and facilitating the validation process. There are examples of unified architectures that have already been investigated by several researchers [1-3]. In our work, these goals are achieved by defining a standard hardware architecture, by forming a software library and developing generic autonomous driving functions at different levels of vehicle autonomy [4].

Replicability of this unified architecture and scalability of the high-level decision making and low-level controllers are crucial to ensure easier implementation on various vehicle platforms [5]. Along with the unified structure, control algorithms also need to be replicable and scalable. This is challenged by significant parametric differences between vehicles. A standardized design procedure for vehicle control is, therefore, needed, to realize replicability and scalability.

In this paper, the design procedure for robust vehicle lateral control to follow the planned path is focused upon and the efficacy of our proposed approach is demonstrated by using our neighborhood electric vehicle (Dash), an experimental autonomous shuttle, as an example. The same control architecture and design procedures can be replicated for other vehicle platforms to adapt controller parameters. Indeed, the hardware and software libraries and basic autonomous driving functions were borrowed, scaled down and replicated from our 2017 Ford Fusion Hybrid research autonomous vehicle. In the steering control application considered here, path following performance should be ensured despite parameter variations like vehicle load, speed and tire cornering stiffness. This demands a robust design for the lateral control by taking these uncertainties into consideration. This paper proposes a robust proportional-derivative (PD) controller design using the parameter-space approach. D-stability and mixed-sensitivity requirements were imposed to ensure its robust performance. To further reduce tracking error, a model regulator was also designed in combination with the robust PD controller.

The organization of the paper is as follows. Section II describes the unified architecture and automation library.

*This work was supported in part by the National Science Foundation under Grant 1640308, and by the U.S. Department of Transportation Mobility 21: National University Transportation Center for Improving Mobility (CMU) sub-project titled: SmartShuttle: Model Based Design and Evaluation of Automated On-Demand Shuttles for Solving the First-Mile and Last-Mile Problem in a Smart City.

Authors are with Center for Automotive Research and with [a] Department of Mechanical and Aerospace Engineering or [b] Department of Electrical and Computer Engineering, Ohio State University, OH 43212 USA (e-mail: zhu.1473@osu.edu; gelbal.1@osu.edu; li.6312@osu.edu; cantas.1@osu.edu; aksunguvenc.1@osu.edu; guvenc.1@osu.edu).

Section III shows the design procedures of the robust PD controller and add-on model regulator by taking the Dash vehicle as an example. Section IV depicts the high-level decision making strategies. Section V explains the validation and evaluation of the control strategy along with sensor placement and autonomous decision making within the hardware-in-the-loop environment. This is followed by the simulation validation of the overall unified architecture and our conclusions.

II. UNIFIED ARCHITECTURE AND AUTOMATION LIBRARY

Our previous work [4,6] on unifying the structure with scalability and replicability is to create a standard base for hardware structure along with a library to be used by developers for faster and easier automation of vehicles. Hardware structure includes different types of sensors to achieve enough coverage, resolution and also robustness to external disturbances. Data from these sensors is being processed by a high processing power computer to create meaningful information, which is used by a low-level controller, i.e. a dSpace MicroAutobox in our vehicles, to drive the vehicle autonomously by interfacing with actuators and sending necessary commands. The unified architecture is shown in Fig. 1.

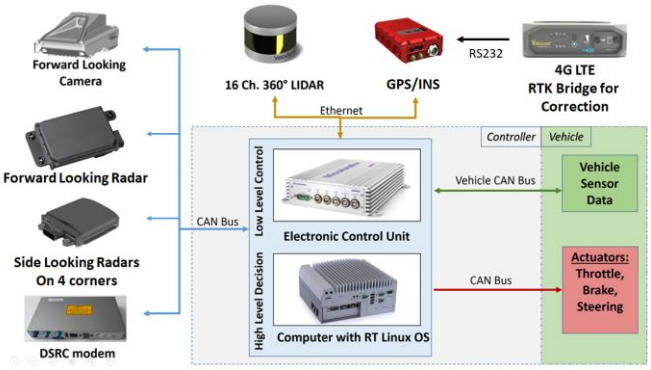


Figure 1. Unified architecture.

Using this unified architecture, two different sized vehicles were automated. Perception sensors such as Lidar, Camera, Radars are implemented as well as GPS Sensor for localization. The dSpace MicroAutobox unit is used for low level controls and an in-vehicle Linux PC with a GPU is used for sensor data computation. Moreover, DSRC (dedicated short-range communications) radios are added to have the capability of communicating with other vehicles, pedestrians, bicyclists and infrastructure. Pictures of the vehicles and the implemented hardware were shown in our previous papers as well as the evaluation of the scalability approach.

Along with the unified architecture, a unified Simulink library was created. This library shown in Fig. 2 consists of different types of blocks, including low-level control blocks for steering, throttle, brake, shift; sensor blocks for receiving data from the sensors in order to have environment perception and localization; and finally control and decision-making blocks for low and high level control of the autonomous vehicle. We are currently extending this library for use with NVIDIA Drive PX 2 GPUs. It is slightly modified for CarSim soft sensors and then used in the HIL simulations reported in this paper.

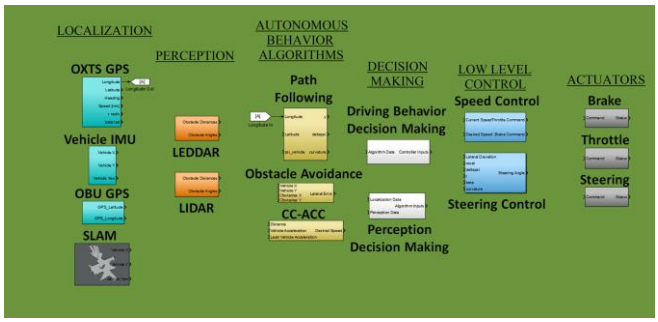


Figure 2. Simulink vehicle automation library.

III. VEHICLE LATERAL CONTROL

Vehicle longitudinal control is realized by tuning a PID controller to follow the speed profile and so is not covered here for the sake of brevity. This paper describes the design of lateral control in detail as it is crucial for path following performance. The vehicle lateral control is comprised of a robust proportional-derivative (PD) controller and an add-on model regulator. While this section took the Dash vehicle as an example, the same lateral control structure and design procedures could easily be applied to other vehicles.

A. Vehicle Model

The bicycle model was used to design the lateral control, as depicted in Fig. 3.

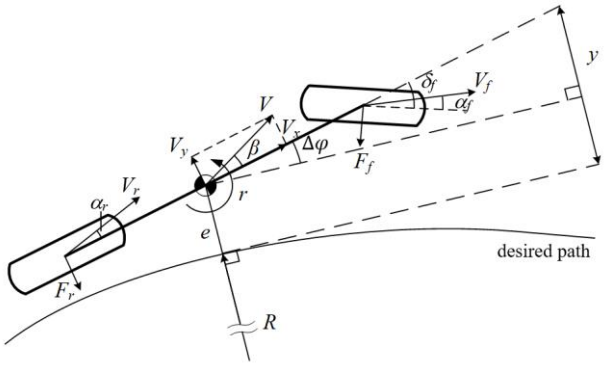


Figure 3. Bicycle model and its deviation from the path

The state-space equation to describe the vehicle states and its deviation from the planned path is [7]:

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \Delta\dot{\psi} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ V & l_s & V & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ \Delta\psi \\ y \end{bmatrix} + \begin{bmatrix} b_{11} & 0 \\ b_{21} & 0 \\ 0 & -V_x \\ 0 & -l_s V_x \end{bmatrix} \begin{bmatrix} \delta_f \\ \rho_{ref} \end{bmatrix}, \quad (1)$$

where the coefficients used are

$$a_{11} = -\frac{C_{\alpha f} + C_{\alpha r}}{\tilde{m} V_x}, \qquad a_{12} = -1 - \frac{C_{\alpha f} l_f - C_{\alpha r} l_r}{\tilde{m} V_x^2},$$

$$a_{21} = -\frac{C_{\alpha f} l_f - C_{\alpha r} l_r}{\tilde{I}_z}, \qquad a_{22} = -\frac{C_{\alpha f} l_f^2 + C_{\alpha r} l_r^2}{\tilde{I}_z V_x},$$

$$b_{11} = \frac{C_{\alpha f}}{\tilde{m} V_x}, \qquad b_{21} = \frac{C_{\alpha f} l_f}{\tilde{I}_z}.$$

$C_{\alpha f}$, $C_{\alpha r}$ are lumped tire cornering stiffness for front and rear sets of tires respectively; V_x is the vehicle longitudinal speed; l_f, l_r are the lengths from the vehicle center of gravity to its front and rear axle respectively; β is the side slip angle; r is the yaw rate of the vehicle; $\Delta\psi$ is the vehicle heading angle error from the path; δ_f is the steering angle of the front wheel; ρ_{ref} is the planned path curvature; y is the look-ahead error from the path at forward distance l_s and can be calculated from lateral error e and heading error $\Delta\psi$:

$$y = e + l_s \Delta\psi \ , \tag{2}$$

$\tilde{m} = m/\eta$, and $\tilde{I}_z = I_z/\eta$ are defined as virtual mass and virtual yaw moment inertia, η is the tire saturation parameter [8]. In this way, the vehicle mass and the tire coefficient η are lumped together for the convenience of uncertainty analysis.

Since the Dash vehicle is operated at low-speeds without extreme behaviors, the uncertainties in the vehicle lateral dynamics are defined accordingly. The vehicle mass is estimated to be inside the range of [300, 500] kg ranging from its curb mass to full-load with two passengers, the velocity is within [2, 10] m/s, and the tire saturation parameter varies between [0.5, 1]. Its uncertainty region is depicted in Fig. 4, along with the platform 2017 Ford Fusion Hybrid which has higher mass and speed range.

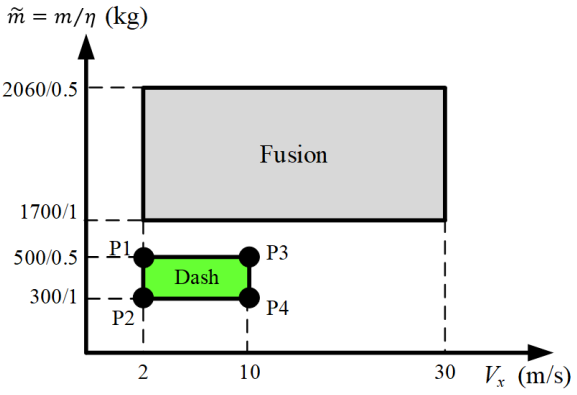


Figure 4. Uncertainty region of vehicle mass m, longitudinal speed V_x and parameter η for Dash and Fusion experiment platforms.

B. Robust Proportional-Derivative Steering Control

A robust proportional-derivative (PD) controller was designed with feedback of look-ahead error y to follow the planned path in the presence of model uncertainties:

$$C(s) = k_p + k_d s \tag{3}$$

D-stability requirement was raised to ensure converged system response within setting time of 8 sec and damping ratio larger than 0.4. These requirements were reflected in the D-stability region in s-plane (Fig. 5) with $\sigma = 0.5$, $R = 100$ and $\theta = 66.2°$.

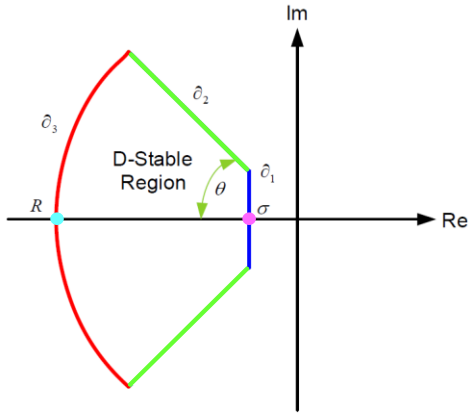


Figure 5. Illustration of D-stability region in the complex plane.

To take both tracking performance and robustness to model uncertainties into account, the mixed sensitivity criterion was considered. The robust performance can be ensured by satisfying:

$$\left\| |W_S S| + |W_T T| \right\|_\infty < 1 \ , \tag{4}$$

or equivalently

$$|W_S S| + |W_T T| < 1 \qquad \forall \omega \ , \tag{5}$$

where S is sensitivity function, T is complementary sensitivity function, W_S and W_T are weights for S and T respectively. The inverse of the sensitivity function weight is chosen as

$$W_S^{-1}(s) = h_s \frac{s + \omega_s l_s}{s + \omega_s h_s}, \tag{6}$$

with $l_s = 0.5$ being the low-frequency sensitivity bound, $h_s = 4$ being the high-frequency sensitivity bound, and $\omega_s = 3$ rad/s. The complementary sensitivity function weight is chosen as

$$W_T(s) = h_T \frac{s + \omega_T l_T}{s + \omega_T h_T}, \tag{7}$$

where the low-frequency gain is $l_T = 0.2$, the high-frequency gain is $h_T = 2$ (corresponds to uncertainty up to 200% at high frequencies), and the frequency of transition to significant model uncertainty is $\omega_T = 20$ rad/s.

The parameter space approach [9] is able to reflect the D-stability boundaries and mixed sensitivity point conditions to the design space of controller parameters k_p and k_d . The parameter space for the four vertices of the uncertainty region is shown in Fig. 6 (a-d). The colored lines are reflections of D-stability boundaries of same colors as in Fig. 5. The points of blue envelop curves were obtained by substituting different frequency values at critical condition of (5). The PD controller parameter was hence selected to be $k_p = 0.5$, $k_d = 0.035$ as indicated with the red cross in the overlapped selectable region (Fig. 6 (e)), satisfying the design requirements at all four uncertainty vertices. The correspond-

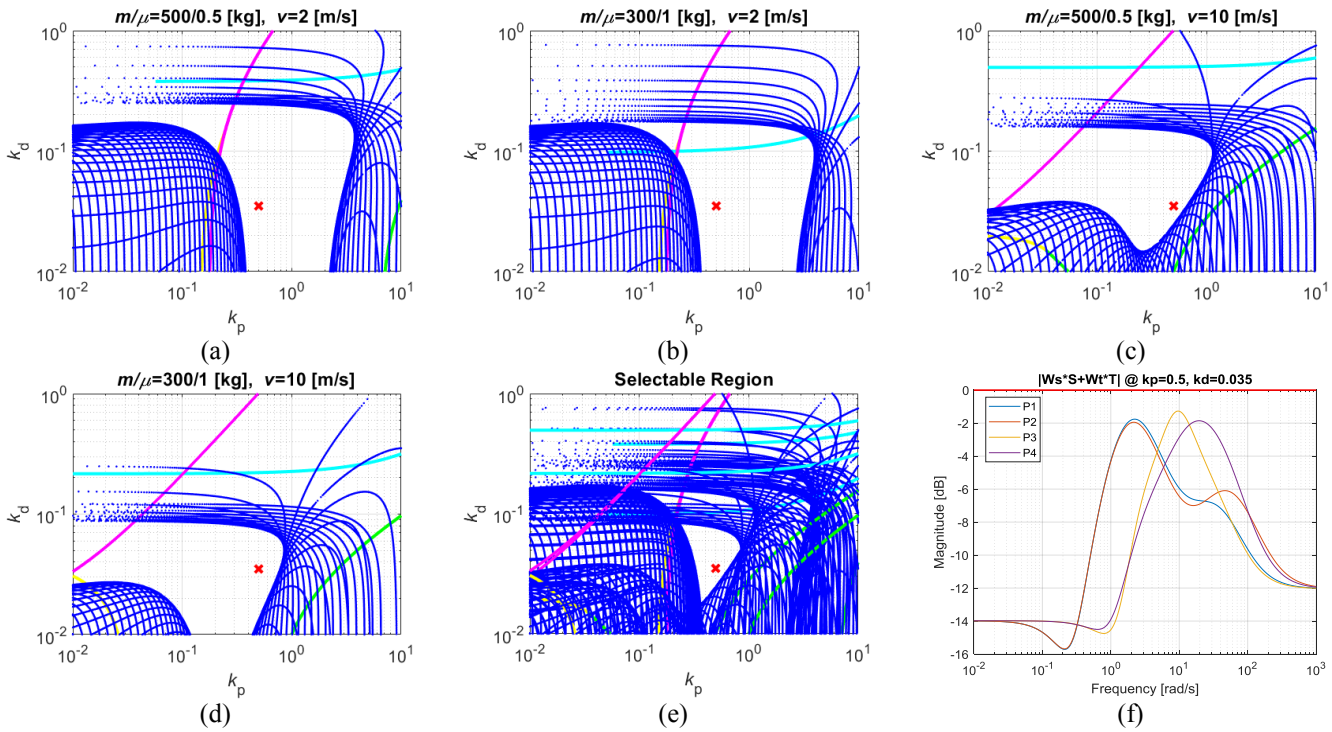


Figure 6. Parameter Space region at apex (a) P1 (b) P2 (c) P3 (d) P4 and (e) overlapped selectable region; (f) Robust performance with selected PD parameters (red cross) at each apex.

ing $|W_S S| + |W_T T|$ magnitude plot at each vertex of the uncertainty region is shown in Fig. 6(f) and validates that the mixed sensitivity constraint is met.

C. *Add-on Model Regulator*

To further reject the look-ahead error, a model regulator was added together with the previously designed robust PD controller (Fig. 7). The model regulator, also referred to as disturbance observer, is proven effective in disturbance rejection and in achieving insensitivity to modelling errors. Its applications in for example in direct drive positioning [10] and friction compensation [11] are successful.

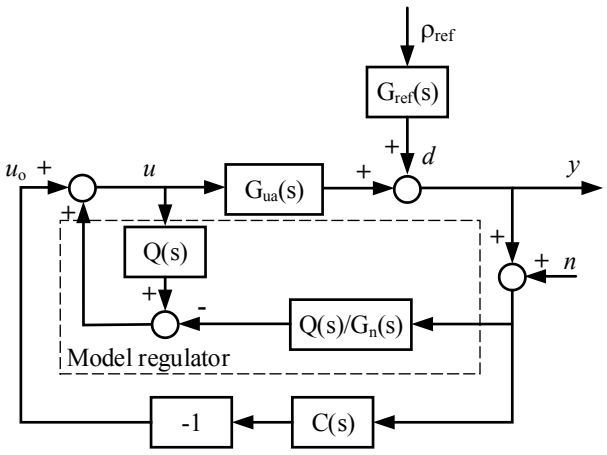


Figure 7. System diagram with the PD controller and model regulator.

The filter Q(s) is chosen to make inverse of nominal model $Q/G_n(s)$ causal with a cutoff frequency at 10 rad/s:

$$Q(s) = \frac{1}{(0.1s+1)^2}. \tag{8}$$

Since the steering is also affected by the look-ahead error through the PD control C(s), the design of model regulator should consider the overall system. The loop gain is:

$$L(s) = \frac{G_{ua}(C + Q/G_n)}{1-Q}. \tag{9}$$

Transfer function of look-ahead error over path curvature and noise can be expressed as:

$$\frac{y}{\rho_{ref}} = \frac{G_{ref}}{1+L} = \frac{G_{ref}(1-Q)}{1-Q+CG_{ua}+G_{ua}Q/G_n}, \tag{10}$$

$$\frac{y}{n} = \frac{-(C+Q/G_n)G_{ua}}{(1-Q)(1+L)} = \frac{-G_{ua}(C+Q/G_n)}{1-Q+CG_{ua}+G_{ua}Q/G_n}. \tag{11}$$

Curvature of the path usually only presents low-frequency characteristic. Therefore to further reject the response of look-ahead error due to curvature, $|y/\rho_{ref}|$ should approach zero at low frequency. To reject noise influence, $|y/n|$ should have small amplitude at high frequency. Considering the characteristic of the filter Q(s), if the nominal model G_n is chosen to approximate G_{ua} closely at high frequency, the stated requirements can be satisfied.

The transfer function of $G_{ua}(s)$ and $G_{ref}(s)$ have the form:

$$G_{ua} = \frac{n_2 s^2 + n_1 s + n_0}{(d_2 s^2 + d_1 s + d_0)s^2}, \tag{12}$$

$$G_{ref}(s) = -\frac{l_s V_x \left(s + \frac{V_x}{l_s} \right)}{s^2}, \qquad (13)$$

where the coefficients n_i, d_i (i =1,2,3) are related to vehicle parameters. Frequency responses of $G_{ua}(s)$ at the uncertainty vertices of Fig. 4 are shown in Fig. 8 and exhibit large differences at low frequencies.

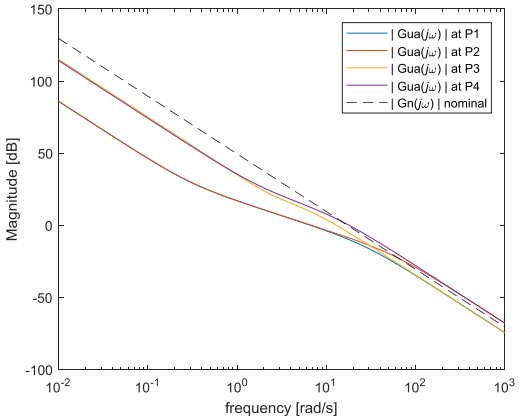


Figure 8. Magnitude $G_{ua}(j\omega)$ for the four vertices.

It could be observed that the magnitude of $G_{ua}(j\omega)$ are similar at high frequency for the vertices of the uncertainty region. Therefore, we could use a single nominal model $G_n(s)$ to approach $G_{ua}(s)$ at high frequency for all the conditions inside the uncertainty region:

$$G_n(s) = \frac{k_n}{s^2}, \qquad (14)$$

where magnitude of $G_n(s)$ at k_n =300 was also shown as the dashed line in Fig. 8.

The magnitude responses $|y/u|$, $|y/\rho_{\text{ref}}|$ and $|y/n|$ before and after adding the designed model regulator are shown in Fig. 9. The magnitude $|y/u|$ converges especially below cut-off frequency of filter Q(s), suggesting good model regulation effect. $|y/\rho_{ref}|$ showed effective rejection of path curvature at low frequency, meaning steady-state tracking error will be greatly reduced. Meanwhile, the noise rejection still remains satisfactory at high frequency as seen from $|y/n|$.

IV. Rule Based Decision Making

To accomplish the function of auto-driving, considering road traffic and infrastructure, we design a control logic based on rule-based decision-making method. Information about desired path, ego-motion, traffic sign and traffic light is provided by the in-vehicle Linux PC or sensors directly. The decision making used in the autonomous drive in our AV test pilot route (from Car to Car West) is represented as a FSM (Finite Sate Machine) in Fig. 10.

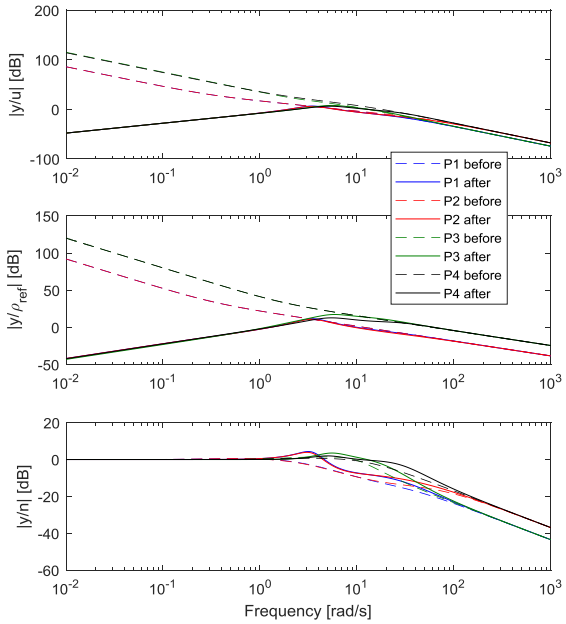


Figure 9. Magnitude $|y/u|$, $|y/\rho_{\text{ref}}|$ and $|y/n|$ before and after applying model regulator.

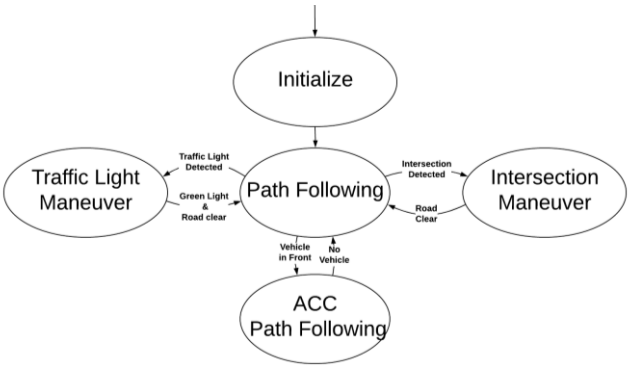


Figure 10. Decision making chart for autonomous drive.

At the Initialize state, the vehicle checks whether all sensors are working properly. Other states are explained below:

- Path Following: the vehicle goes along the planned path with the aforementioned lateral control algorithm.
- ACC (Adaptive Cruise Control) Path Following: When a vehicle is detected in front, the ego-vehicle adapts its peed to the target while following the planned path.
- Traffic Light Maneuver: Trigger when receiving traffic signal phase and timing (SPaT) information: If the light is Green, our vehicle will check crossing traffic until the road is clear and switch back to path following; if the light is yellow or red, it will be triggered to stop at the traffic light to wait for the red light turning into green.
- Intersection Maneuver: Triggered when the vehicle comes to an intersection or a stop sign, ego-vehicle will wait at the intersection, detecting crossing traffic until the road is clear.

Based on the control logic, the soft version of our Dash autonomous vehicle was able to run the shuttle task with the designed low level robust steering and speed controllers, without running into problems in repeated simulations with random traffic in our HiL simulator.

V. HIL Environment and Simulations

To extensively evaluate the performance of the developed control strategy, along with high level control, decision making and sensor placement, a hardware-in-the-loop (HIL) simulator is employed (Fig. 11). A high-fidelity CarSim vehicle model runs in dSpace Scalexio HIL platform to simulate vehicle lateral and longitudinal motions alongside the perception sensors in real time, while the MicroAutobox controller implements the actual control scheme at 100Hz sampling frequency.

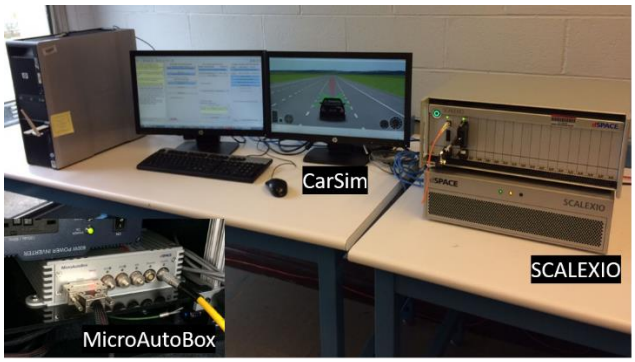


Figure 11. Hardware-in-the-Loop setup

A. Slalom Path

Fig. 12 shows the reference slalom path and the trajectory of Dash at 10m/s with the designed lateral control, along with another of Ford at 30m/s following the same lateral control design procedure. Both vehicles were able to follow the slalom path of tracking error within 0.15 m, suggesting the replicability of the lateral control design procedure.

The effect of the added model regulator was also proved to be effective in reducing look-ahead error as compared to the use of the PD steering controller alone (Fig. 13). The steering angle and yaw motion during the process were without apparent oscillation (Fig. 14). Contributions from PD controller and model regulator respectively are also shown in Fig. 14.

B. CARWest-to-CAR Route

The OSU AV pilot test route from CAR West (our lab location) to CAR (Center for Automotive Research – our main research center) shown in Fig. 16 was chosen and constructed in CarSim to evaluate the vehicle's decision making and lateral control performance. To incorporate the real traffic into simulation, information about other vehicles on the road were imported from SUMO software (Fig. 15).

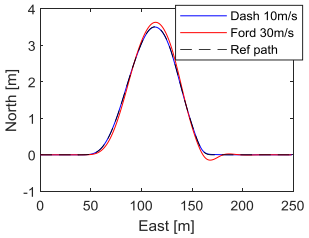


Figure 12. Slalom trajectory of Dash at 10m/s and Ford at 30m/s.

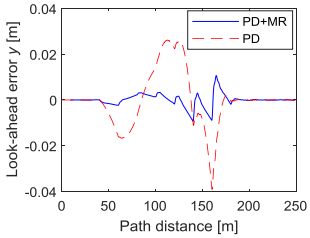


Figure 13. Look-ahead error with and without model regulator (Dash 10m/s).

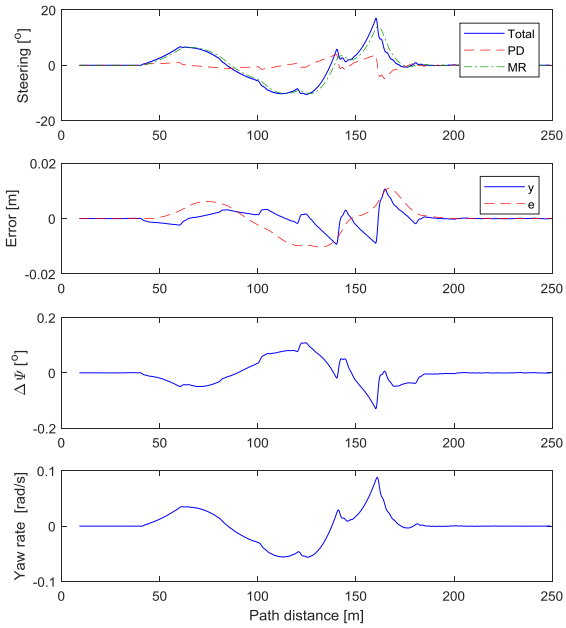


Figure 14. Steering, tracking error and yaw rate of slalom test (Dash 10m/s).

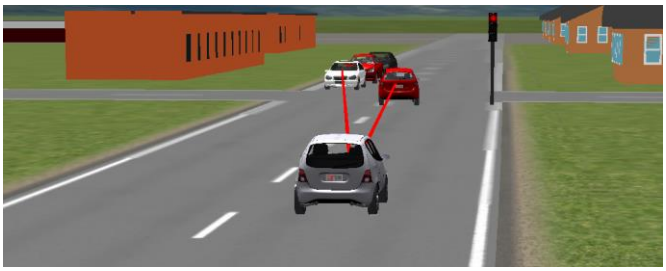


Figure 15. Constructed road with traffic in Carsim.

Placement and field of view (FOV) of the sensors described in section II can be seen in Fig. 17. These were implemented as soft sensors in the HIL simulator.

Figure 16. CARWest-to-CAR route.

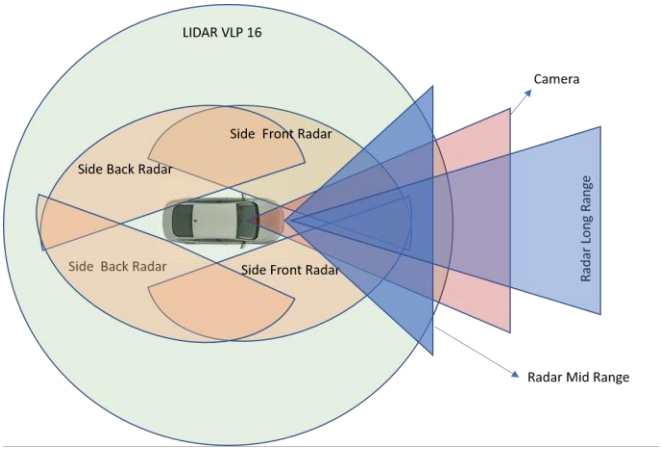


Figure 17. Placement and FOV of the sensors on the car.

Scenarios including stop sign, crossing traffic and traffic lights were simulated on the route in Fig. 16 to test the decision-making strategies introduced in Section IV. Fig. 18 shows the vehicle speed, steering and tracking error along the route. Since sharp turns appear when entering and exiting the main straight road, lateral error e was expectedly high for these two cases, but the look-ahead error y which combines the lateral and heading angle error was still relatively small.

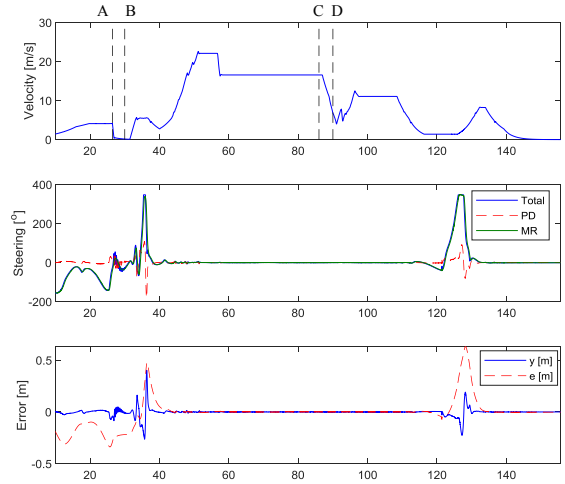


Figure 18. CARWest-to-CAR AV pilot test route simulation results. (A-close to stop sign; B-traffic crossing; C-traffic light nearby and turns red; D-traffic light turns green)

CONCLUSION

A unified and replicable approach on lateral control design procedure based on robust PD controller with parameter space design and add-on model regulator is introduced. HIL test on slalom path suggested good path following performance of the lateral control and its replicability to another larger vehicle platform. The lateral control was integrated with our previous developed unified, scalable and replicable autonomous driving solution. Decision-making strategies, sensor perception and lateral control were evaluated in the Ohio State University AV pilot test route with random traffic in a simulation study. Autonomous shuttles are planned to be used on this short route and then extended to the rest of the university campus. The approach used in this paper presented a method of in-lab evaluation in a realistic traffic environment for identifying and fixing possible problems before an actual deployment. A future experiment will be conducted with our experimental vehicles to evaluate our unified architecture.

REFERENCES

[1] S. Behere and M. Törngren, "A functional architecture for autonomous driving," in *Proceedings of the First International Workshop on Automotive Software Architecture*, pp. 3-10, 2015.

[2] K. Jo, J. Kim, D. Kim, C. Jang and M. Sunwoo, "Development of autonomous car—Part II: A case study on the implementation of an autonomous driving system based on distributed architecture," *IEEE Trans.Ind.Electron.*, vol. 62, pp. 5119-5132, 2015.

[3] J. Ziegler, P. Bender, M. Schreiber and H. Lategahn, "Making Bertha Drive- An Autonomous Journey on a Historic Route," *IEEE Intell. Transport. Syst. Mag.*, vol. 6, no. 2, pp. 8-20, 2014.

[4] S.Y. Gelbal, B.A. Guvenc and L. Guvenc, "SmartShuttle: a unified, scalable and replicable approach to connected and automated driving in a smart city," in *Proceedings of the 2nd International Workshop on Science of Smart City Operations and Platforms Engineering*, pp. 57-62, 2017.

[5] F. Bonsignorio, and A. P. del Pobil, "Toward replicable and measurable robotics research," *IEEE Robot. Automat. Mag.*, vol. 21, no. 1, pp. 22-25, 2015.

[6] S.Y. Gelbal, N. Chandramouli, H. Wang, B. Aksun-Guvenc and L. Guvenc, "A Unified Architecture for Scalable and Replicable Autonomous Shuttles in a Smart City," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 3391-3396, 2017.

[7] L. Guvenc, B. Aksun-Guvenc, B. Demirel and M.T. Emirler, *Control of Mechatronic Systems*, Institution of Engineering and Technology, 2017.

[8] K.L. Talvala, K. Kritayakirana and J.C. Gerdes, "Pushing the limits: From lanekeeping to autonomous racing," *Annual Reviews in Control*, vol. 35, pp. 137-148, 2011.

[9] J. Ackermann, Robust control: the parameter space approach, Springer Science & Business Media, 2012, .

[10] Y.A.I. Mohamed, "Design and implementation of a robust current-control scheme for a PMSM vector drive with a simple adaptive disturbance observer," *IEEE Trans.Ind.Electron.*, vol. 54, pp. 1981-1988, 2007.

[11] Z. Jamaludin, H. Van Brussel and J. Swevers, "Friction compensation of an XY feed table using friction-model-based feedforward and an inverse-model-based disturbance observer," *IEEE Trans.Ind.Electron.*, vol. 56, pp. 3848-3853, 2009.

2018-01-0608 Published 03 Apr 2018

Camera Based Automated Lane Keeping Application Complemented by GPS Localization Based Path Following

Mustafa Ridvan Cantas and Levent Guvenc The Ohio State University

Citation: Cantas, M.R. and Guvenc, L., "Camera Based Automated Lane Keeping Application Complemented by GPS Localization Based Path Following," SAE Technical Paper 2018-01-0608, 2018, doi:10.4271/2018-01-0608.

Abstract

Advances in sensor solutions in the automotive sector make it possible to develop better ADAS and autonomous driving functions. One of the main tasks of highway chauffeur and highway pilot automated driving systems is to keep the vehicle between the lane lines while driving on a pre-defined route. This task can be achieved by using camera and/or GPS to localize the vehicle between the lane lines. However, both sensors have shortcomings in certain scenarios. While the camera does not work when there are no lane lines to be detected, an RTK GPS can localize the vehicle accurately. On the other hand, GPS requires at least 3 satellite connections to be able to localize the vehicle and more satellite connections and real-time over-the-air corrections for lane-level positioning accuracy. If GPS localization fails or is not accurate enough, lane line information from the camera can be used as a backup. In this paper, a vision based lane keeping system is aided by a GPS based path following application to overcome the shortcomings of the GPS and camera sensors when used alone in highway driving path following applications. The developed system has a parameter space based robust steering controller which can handle lateral motion control of the vehicle based on path tracking error detected using the GPS or camera sensor. The designed control system works for both low speed and high-speed driving scenarios and is robust to changes in vehicle mass. The results are demonstrated using the validated model of our 2017 Ford Fusion Hybrid research automated driving vehicle in our hardware-in-the-loop simulator. Experimental verification is also planned.

Introduction

There are six automated driving levels defined in new SAE International standard J3016 varying from 0 to 5 [1], where 0 represents the no automation case and 5 represents the fully autonomous driving case with no human intervention. This paper will cover a lane keeping application for a vehicle already equipped with an adaptive cruise control. This automation level falls within Level 2 which is partial automation where the steering and acceleration of the vehicle are handled by the automated driving system but the driver is still in the loop. This is an initial part of our work aimed at developing a Level 3 Highway Chauffeur and a Level 4 Highway Autopilot in a realistic hardware-in-the-loop simulation (HIL) environment.

In the literature and in production level vehicles, there are many lane-keeping and lane departure warning applications. For instance, Tuncer et al. worked on developing a lane keeping system when the driver is inattentive [2]. In their application, a camera based lane keeping controller is designed and simulated in the HIL simulator. Kang et al. proposed a solution for estimating the lane positions for short term lane information lost from the camera [3]. Although lane keeping applications and path fallowing applications are thoroughly studied in the literature, the failure of the existing systems would not be acceptable for a fully autonomous vehicle system. Considering many of these systems are using the camera to detect the lane lines and localize the vehicle in the lateral direction, the failure of the camera detection would result in failure to keep being within the lane. As highlighted in the work of the Yenikaya et al. [4], some of the camera detection failures can be caused by the absence of the lane lines, poor lane line quality, shadow on the lane lines, or other vehicle occlusions. The camera may also completely fail to work or communicate with the controller. Today high accuracy GPS units are also available for accurate localization. For example, the GPS unit used in our experimental vehicle OXTS xNAV550 has 1.6 m accuracy with single antenna, 0.4 m for DGPS mode and up to 2 cm for RTK mode using a base station. Also with the use of online RTK correction services and RTK Bridge units it is possible to have RTK corrections without a base station. In the case of using RTK bridge unit, accuracy of the system is around 5 cm.While today RTK GPS units are very expensive as compared to the cameras, they are getting cheaper with the advance of the technology. Therefore, usage of a GPS based lane level path fallowing algorithm is suggested as one of the backup solutions for the camera failure cases. One might ask why GPS system is not used solely for the lane keeping application. This is because the GPS system also has its own shortcomings. If the RTK corrections for the GPS are not available or the lane level map of the environment is not

available, it is not possible to localize the vehicle within lane level accuracy. Therefore, a combination of the camera and GPS solution is preferred over using them alone by themselves.

The rest of the paper has sections in the following order: Lateral vehicle model, lane detection, path generation, lateral deviation calculation, lateral controller design, simulation results and summary/conclusions.

Lateral Vehicle Model

In this paper, lateral dynamics of the vehicle is modeled using the nonlinear vehicle model (Bicycle Model). In this model, the two front wheels are represented as single front wheel and similarly, the two rear wheels of the vehicle are represented as a single rear wheel. As our test vehicle is only steerable from the front wheels, the test vehicle is modeled to be only steerable from the front wheel [5]. Forces acting on the vehicle in this model are shown in Figure 1. Lateral forces generated by the front/rear wheels, vehicle center of gravity, distance of the center of gravity from the wheels and the preview distance are represented in the figure as F_f/ F_r, CG, l_f/ l_r, l_s respectively.

The lateral direction steering controller for the automated lane-keeping application is designed using a linearized version of the nonlinear vehicle model. Linearized state space model of the lateral motion of the vehicle is given in Equation 1 where β is the vehicle side slip angle at the vehicle center of gravity, r is vehicle yaw rate, V is velocity, $\Delta\psi$ is yaw angle of the vehicle with respect to desired path's tangent, ρ_{ref} is the road curvature, δ_f is the steering wheel angle and μ is the friction coefficient of the road. The entries a_{11}, a_{12}, a_{21}, a_{22}, b_{11}, b_{12} used in Equation 1 are given in Equations 2-7, where c_r, c_f are the cornering stiffness of the rear and front wheels, $\tilde{J} = J/\mu$ is the virtual mass moment of inertia and the $\tilde{m} = m/\mu$ is the virtual mass.

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \Delta\dot{\psi} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ V & l_s & V & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ \Delta\psi \\ y \end{bmatrix} + \begin{bmatrix} b_{11} & 0 \\ b_{21} & 0 \\ 0 & -V \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_f \\ \rho_{ref} \end{bmatrix} \quad (1)$$

$$a_{11} = \frac{-(c_r + c_f)}{\tilde{m}V}, a_{12} = -1 + \frac{-(c_r l_r - c_f l_f)}{\tilde{m}V^2}, \quad (2)$$

$$a_{11} = \frac{-(c_r + c_f)}{\tilde{m}V}, a_{12} = -1 + \frac{-(c_r l_r - c_f l_f)}{\tilde{m}V^2}, \quad (3)$$

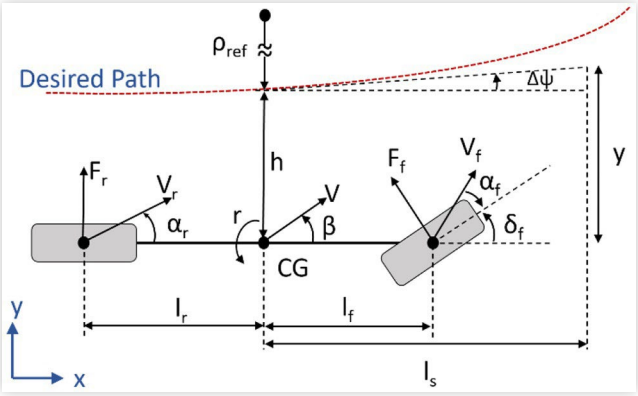
$$a_{21} = \frac{(c_r l_r - c_f l_f)}{\tilde{J}}, a_{22} = \frac{-(c_r l_r^2 + c_f l_f^2)}{\tilde{J}V^2}, \quad (4)$$

$$a_{21} = \frac{(c_r l_r - c_f l_f)}{\tilde{J}}, a_{22} = \frac{-(c_r l_r^2 + c_f l_f^2)}{\tilde{J}V^2}, \quad (5)$$

$$b_{11} = \frac{c_f}{\tilde{m}V}, b_{12} = \frac{c_f l_f}{\tilde{J}}, \quad (6)$$

$$b_{11} = \frac{c_f}{\tilde{m}V}, b_{12} = \frac{c_f l_f}{\tilde{J}}, \quad (7)$$

FIGURE 1 Lateral vehicle model for lane keeping application



Lane Detection

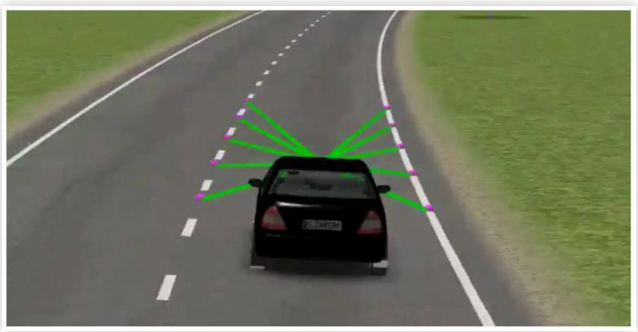
Lane lines on the road can be used to localize the ego vehicle on the road Cartesian coordinates. Our research automated driving vehicle that we plan to use in the future for experimental evaluation has a Mobileye camera which can provide the coefficients of the polynomial fit for the lane detections, lane detection availability and quality information and a GreyPoint camera with our own algorithms for the same outputs. In this paper, we are using our connected and automated driving HIL simulator which has CarSim Real Time with Sensors and Traffic. The CarSim soft camera sensor in the HIL simulator provides the lane detection in the form of x, y coordinates (Figure 2). To simulate the real sensor output and extrapolate the lane detection points, two second order curves denoted by *yl(x)* and *yr(x)* are fitted to the left and right lane detection points respectively, coming from the CarSim software (Equations 8 and 9).

$$y^l(x) = a_0^{\,l} + a_1^{\,l} * x + a_2^{\,l} * x^2 \quad (8)$$

$$y^r(x) = a_0^{\,r} + a_1^{\,r} * x + a_2^{\,r} * x^2 \quad (9)$$

By inserting a longitudinal distance x into the Equations 8 and 9, one can calculate the lateral distance of the vehicle from the right and left lane lines at that longitudinal distance.

FIGURE 2 CarSim soft camera sensor visualization.



Path Generation

For generating the lane level path following map/path, the method presented in [5, 6] is used. This method requires to drive the car at a constant speed at the center of the road and collect accurate GPS data points. These GPS waypoints can also be automatically extracted from a realistic map. We use both approaches. Our future work will be based on an e-Horizon system once we add this capability to our research automated driving vehicle. Collected GPS waypoints are divided into a predetermined number of polynomial segments to capture the different characteristics of the road. These segments are represented as 3rd order parametric polynomials of a distance parameter λ, where λ changes between *i-1* to *i*, according to the number of the segment used. These polynomials are given below as:

$$X_i(\lambda) = a_{xi}\lambda^3 + b_{xi}\lambda^2 + c_{xi}\lambda + d_{xi} \tag{10}$$

$$Y_i(\lambda) = a_{yi}\lambda^3 + b_{yi}\lambda^2 + c_{yi}\lambda + d_{yi} \tag{11}$$

where X_i and Y_i are the path centerline coordinates for the *i*th segment. Since the polynomials fitted to two consecutive segments need to have continuity at their intersection, the polynomials can be fitted to the GPS waypoints using the constrained least squares method. However, to solve the constrained least squares problem, first, the unconstrained problem needs to be solved. The unconstrained problem can be formed in matrix form as shown below.

$$x_{data} = \Lambda n_{x,uncs} \tag{12}$$

$$y_{data} = \Lambda n_{y,uncs} \tag{13}$$

$$\Lambda = \begin{bmatrix} \bar{\lambda}^3 & \bar{\lambda}^2 & \bar{\lambda} & 1 & 0 & 0 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \bar{\lambda}^3 & \bar{\lambda}^2 & \bar{\lambda} & 1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \tag{14}$$

$$n_{x,uncs} = \begin{bmatrix} a_{x1} & b_{x1} & c_{x1} & d_{x1} & \cdots & a_{xm} & b_{xm} & c_{xm} & d_{xm} \end{bmatrix}^T \tag{15}$$

$$n_{y,uncs} = \begin{bmatrix} a_{y1} & b_{y1} & c_{y1} & d_{y1} & \cdots & a_{ym} & b_{ym} & c_{ym} & d_{ym} \end{bmatrix}^T \tag{16}$$

In the Λ matrix, $\bar{\lambda}$ represents the entire λ vector which ranges from *i-1* to *i*, where *i* is the number of the segment. The number of elements in $\bar{\lambda}$ is equal to the number of the data points in the i^{th} segment. For the given equations, the solution of the unconstrained least square problem is given in Equations 17 and 18 as

$$n_{x,uncs} = \left(\Lambda^T\Lambda\right)^{-1}\Lambda^T x_{data} \tag{17}$$

$$n_{y,uncs} = \left(\Lambda^T\Lambda\right)^{-1}\Lambda^T y_{data} \tag{18}$$

To sustain the continuity and smoothness (continuity of the first derivative) at the segment boundaries, the constraints given below are defined.

$$X_i(i) = X_{i+1}(i) \tag{19}$$

$$Y_i(i) = Y_{i+1}(i) \tag{20}$$

$$\frac{dX_i(i)}{d\lambda} = \frac{dX_{i+1}(i)}{d\lambda} \tag{21}$$

$$\frac{dY_i(i)}{d\lambda} = \frac{dY_{i+1}(i)}{d\lambda} \tag{22}$$

$$\frac{d^2X_i(i)}{d\lambda} = \frac{d^2X_{i+1}(i)}{d\lambda} \tag{23}$$

$$\frac{d^2Y_i(i)}{d\lambda} = \frac{d^2Y_{i+1}(i)}{d\lambda} \tag{24}$$

From the equations 10 and 11, these constraints can be rewritten as shown in Equations 25-30.

$$a_{xi}i^3 + b_{xi}i^2 + c_{xi}i + d_{xi} = a_{xi+1}i^3 + b_{xi+1}i^2 + c_{xi+1}i + d_{xi+1} \tag{25}$$

$$a_{yi}i^3 + b_{yi}i^2 + c_{yi}i + d_{yi} = a_{yi+1}i^3 + b_{yi+1}i^2 + c_{yi+1}i + d_{yi+1} \tag{26}$$

$$3a_{xi}i^2 + 2b_{xi}i + c_{xi} = 3a_{xi+1}i^2 + 2b_{xi+1}i + c_{xi+1} \tag{27}$$

$$3a_{yi}i^2 + 2b_{yi}i + c_{yi} = 3a_{yi+1}i^2 + 2b_{yi+1}i + c_{yi+1} \tag{28}$$

$$6a_{xi}i + 2b_{xi} = 6a_{xi+1}i + 2b_{xi+1} \tag{29}$$

$$6a_{yi}i + 2b_{yi} = 6a_{yi+1}i + 2b_{yi+1} \tag{30}$$

These defined constraint equations are used in matrix form to convert the unconstrained problem into the constrained problem. These equations are combined into a matrix form as shown in Equations 31 and 32.

$$Fn_{x,cs} = 0 \tag{31}$$

$$Fn_{y,cs} = 0 \tag{32}$$

Finally, the solution of the constrained problem is given in Equations 33 and 34.

$$n_{x,cs} = n_{x,uncs} - \left(\wedge^T\wedge\right)^{-1}F^T\left[F\left(\wedge^T\wedge\right)^{-1}F^T\right]^{-1}F_{n,xuncs} \tag{33}$$

$$n_{y,cs} = n_{y,uncs} - \left(\wedge^T\wedge\right)^{-1}F^T\left[F\left(\wedge^T\wedge\right)^{-1}F^T\right]^{-1}F_{n,yuncs} \tag{34}$$

Lateral Deviation Calculation

The lateral controller takes the lateral deviation at a predefined preview distance as input and calculates the corresponding steering angle. As mentioned earlier, two different methods are used to calculate the lateral deviation in this application. The first method uses lane detections acquired from the camera and the second method uses the GPS localization and map based waypoint information.

Lateral Deviation from the Lane Line Detections:

The polynomials representing the lane lines must be parallel to one another as the lane lines are parallel to each other in a real road. Knowing this, the centerline of the road can be represented with the polynomial below in vehicle coordinates.

$$y^c(x) = a_0{}^c + a_1{}^c * x + a_2{}^c * x^2 \tag{35}$$

where coefficients of the polynomial which represents the centerline are given below. Here the superscript "*c*" indicates that the polynomial is a fit for the centerline of the road, and the coefficients of the polynomial are given in Equations 36-38.

$$a_0{}^c = \left(a_0{}^l + a_0{}^r\right)/2 \tag{36}$$

$$a_1{}^c = \left(a_1{}^l + a_1{}^r\right)/2 \tag{37}$$

$$a_2{}^c = \left(a_2{}^l + a_2{}^r\right)/2 \tag{38}$$

Inserting the preview distance l_s into Equation 35 gives the lateral distance of the vehicle at the preview distance.

Lateral Deviation Calculation from the Map and GPS Measurements:

When the lane detections are not available or reliable, the lateral deviation at the preview distance is calculated using the current lateral deviation and the yaw angle error with respect to the generated map. Based on the geometry shown in Figure 1 the lateral deviation at the preview distance l_s can be calculated as

$$y = \mathrm{h} + \mathrm{l}_s \sin(\Delta\Psi) \tag{39}$$

where h is the lateral deviation from the desired path at the vehicle center of gravity, l_s is the preview distance and the $\Delta\psi$ is the yaw angle of the vehicle with respect to the desired path.

First, the lateral deviation of the vehicle from the generated map is calculated. Assuming the radius of the curvature is much larger than the lateral deviation of the vehicle, the shortest distance to the path can be calculated by finding the perpendicular vector to the path from the vehicle center of gravity. This means that the tangent vector of the path will be orthogonal to the shortest vector between the generated path and the vehicle center of gravity as shown in Figure 3. Here the center of the gravity of the vehicle is represented using east and north map coordinates P_E and P_N respectively. Using the fact that dot product of two orthogonal vectors is zero, the solution of Equation 40 for λ_c gives the closest segment position to the vehicle. One can evaluate the x, y coordinates of the closest point on the path and the distance of the vehicle from the path by inserting λ_c into the Equation 41.

$$\left(X(\lambda) - P_E, Y(\lambda) - P_N\right)\left(\dot{X}(\lambda), \dot{Y}(\lambda)\right) = 0 \tag{40}$$

$$h = \rho\sqrt{\left(X(\lambda_c) - P_E\right)^2 + \left(Y(\lambda_c) - P_N\right)^2} \tag{41}$$

where

$$\rho = \mathrm{sgn}\left(\vec{U}(3)\right) \tag{42}$$

$$\vec{U} = \left(\left(X(\lambda_c) - P_E\right), \left(Y(\lambda_c) - P_N\right), 0\right) \mathrm{x} \left(\dot{X}(\lambda), \dot{Y}(\lambda), 0\right) \tag{43}$$

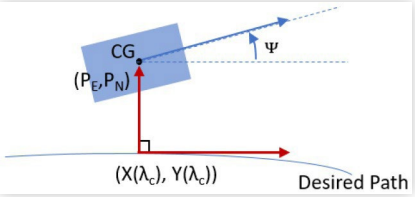
If the third component of the cross product of the path tangent and distance vector is negative, it shows that the vehicle is in the inner side of the desired path and vice-versa.

After finding the h, $\Delta\psi$ is calculated by subtracting the slope of the road at the closest point on the reference path from the yaw angle of the vehicle. Calculation of $\Delta\psi$ can be seen in Equation 44.

$$\Delta\Psi = \Psi - \frac{\dot{Y}(\lambda_c)}{\dot{X}(\lambda_c)} \tag{44}$$

Finally, the lateral deviation at the preview distance can be calculated by inserting h, l_s and $\Delta\psi$ into equation 39.

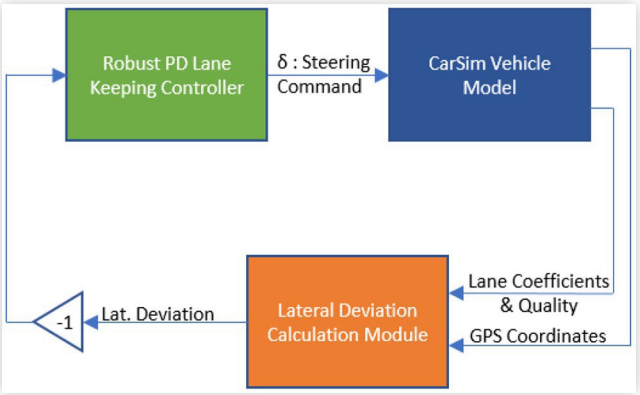
FIGURE 3 Position and orientation of the vehicle with respect to the desired path.



Lateral Controller Design

The parameter space based design approach given in [7] is used to design the PD controller for the robust lane-keeping controller for the overall system shown in Figure 4. The input and outputs of the system can be listed as the steering command and the lateral deviation at the preview distance respectively. The test vehicle modeled in the state space model of Equation 1 has the numerical parameter values $J = 3{,}728\ \mathrm{kgm}^2$, $C_f = 1.2\mathrm{e}5$ N/rad, $C_r = 1.9\mathrm{e}5$ N/rad, $l_r = 1.5453$ m and $l_f = 1.30$ m where the weight of the vehicle varies between 1,700 kg and 2,000 kg.

FIGURE 4 Lateral controller system block diagram.



Since the vehicle operates in different load and speed conditions, the controller is designed to be able to work under these different operating conditions as is shown in Figure 5 as an uncertainty box. Also, the preview distance for higher speeds is increased as $l_s = \max(k_s v, l_{smax})$ where v is vehicle speed, k_s is a proportional factor and l_{smax} is the upper bound on the preview length. In this paper, k_s is adjusted such that preview distance changes linearly between 4 m to 7 m for the chosen operating speed range 5 m/s to 30 m/s.

As a D-stability requirement, desired settling time, damping ratio and maximum bandwidth are chosen as 0.5 seconds, 0.7 and 19 rad/sec respectively. PD controller coefficients (K_p and K_d) are chosen as free parameters to find a solution region using the parameter space approach. D-stability solution region is constructed for each corner of the uncertainty box in Figure 5 and they are overlaid on top of each other to find the overall solution region as shown in Figure 6. In this figure blue, green, red, cyan, magenta colored lines show Settling Time Constraint Complex Root Boundary (CRB), Damping Constraint CRB, Bandwidth Constraint CRB, Bandwidth Constraint Real Root Boundary (RRB), Settling Time Constraint RRB respectively. Since blue line is covered by the magenta, it is not clearly visible. Calculation of these boundaries are shown in detail in [7]. By choosing a point in this solution region, one set of K_p and K_d values for the PD controller are chosen as shown in the right plot in Figure 6.

FIGURE 5 Lateral dynamics uncertainty box.

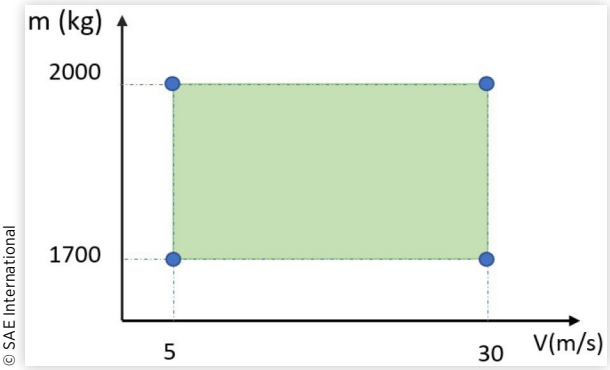


FIGURE 6 Left: Solution regions for the corners of the uncertainty box is plotted on top each other. Right: The zoomed version of the left figure where intersection of the solution regions is highlighted with a gray fill and chosen solution point is shown with a red dot.

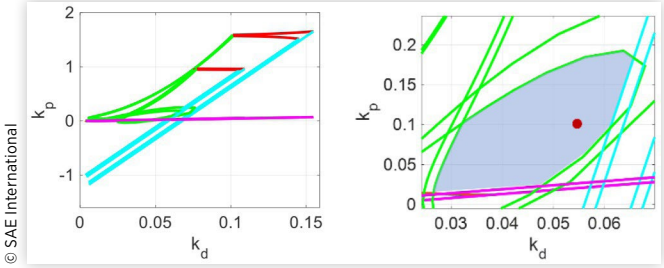
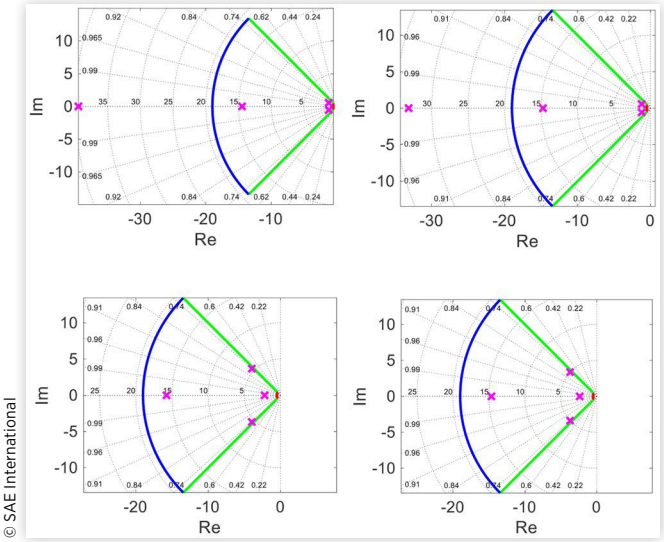


FIGURE 7 D-Stable region and the system pole positions in complex plane for the chosen K_p and K_d. Top Left: 5 m/s, 1700 kg l_s = 4 m, Top Right: 5 m/s, 2000 kg l_s = 4 m, Bottom Left: 30 m/s, 1700 kg /s = 7 m, Bottom Right: 30 m/s, 2000 kg l_s = 7 m.



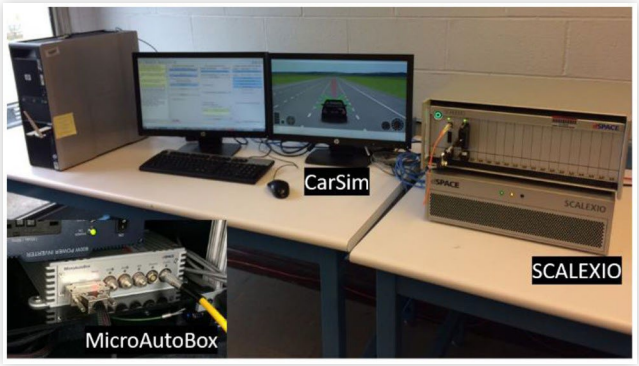
To be considered as a D-stable system, the poles of the system should lie within the D Stable region where it is defined by the desired settling time, the desired minimum damping ratio and the desired maximum bandwidth, all given earlier. As it can be seen from Figure 7, all of the dominant poles of the system which are marked as "x" lie in the D-Stable region for the chosen K_p and K_d coefficients of the PD controller.

Hardware in the Loop Simulator

Testing the developed algorithms in the Hardware in the Loop simulator is a prerequisite to road testing. While testing the vehicles on the road may take extensive time and money, by using simulators, these adverse effects can be minimized with the ease of repeating the simulations in an accident-free environment. Running in real time and being able to connect to the hardware used in the vehicle makes the Hardware in the Loop simulator more advantageous over the regular Model in the Loop simulators. The setup of the HIL simulator used in this paper can be seen in the Figure 8. The setup consists of three main components. The following paragraph will give brief information about these components.

The first component in the system is the computer with CarSim software. This computer is used to design controller algorithms in Matlab and prepare the model of the test vehicle, sensors, and roads. This computer is also used as an interface to communicate with the real-time simulation computer and the controller during the simulation. Secondly, the dSPACE SCALEXIO Processing Unit, which is the real-time computation unit in the HIL, is used to run the validated vehicle model, sensors and traffic information based on the information

FIGURE 8 Hardware in the Loop Simulator setup.



coming from its input ports. Finally, a MicroAutoBox controller unit is connected to HIL, as the control task of the test vehicle is handled by MicroAutoBox. Like the test vehicle setup, in this system computer is connected to MicroAutoBox and the SCALEXIO via ethernet port and the communication between the MicroAutoBox and the SCALEXIO is handled by the CAN Bus. While not used in this paper, we also have DSRC radio units connected as hardware in our HIL simulator. In the next section, simulation results of the lane keeping system are given.

Simulation Results

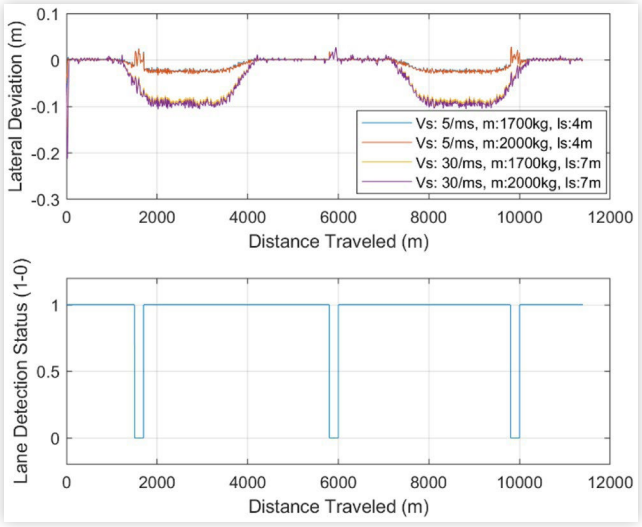
To evaluate the performance of the system a designed lane keeping controller is tested in the CarSim environment described in the previous section. As a test track, a simple model of the high-speed test track in the Transportation Research Center proving ground is constructed in CarSim using its pre-recorded GPS waypoints. The top view of the TRC testing track can be seen in the Google Maps image in Figure 9 which has two curved section between 1000 m-4200 m and 7200 m-10100 m. Since the GPS based path following localization is used as a backup solution, camera based lane keeping system is considered as the active system. In the experimental setup, vehicle is equipped with a Mobileye camera where it can output the quality of the lane line detections. So, in the experiments the mode switching between the camera and GPS will be done based on this lane line detection quality information by the Lateral Calculation Module show in in Figure 4. If the there is no reliable lane detections, vehicle is going to switch to GPS based path following mode. Based on the road conditions lane quality of the system can fail anytime. To simulate the cases where the camera detection fails, the system switches to the GPS based lane keeping mode for pre-defined distance intervals (1500-1700, 5800-6000, and 9800-10000 meters). In the first and third sections vehicle is travelling in the curved parts of the test track while it travels at the straight part of the road in the second interval.

The designed system is tested for the different speed and vehicle mass conditions which are defined as the corners of the uncertainty box given in Figure 5.

The simulation results for the designed system are shown in Figure 10.

FIGURE 9 Top view of the TRC test tracks from Google Maps.

FIGURE 10 Simulation Results for the designed lane keeping system. Top: lateral deviation vs distance traveled for different operating conditions. Bottom: Availability of lane detection vs distance traveled.



When the lateral deviation graph (Figure 10) is analyzed, it can be seen that the designed system still keeps the vehicle in the lane even when the lane detection status goes to zero. Although both the vision based and the GPS based solutions have a higher error for the curved sections of the road, which increases for high speeds, this error is less than 12 cm.

Summary/Conclusions

This paper presented the use of a GPS based lane keeping/path following application as a backup to the camera based lane keeping application. By using these two methods, lane level

control for the vehicle is sustained even when one of the sensor inputs is not available or is not reliable. This happens for instance when lane markings are missing or are very vague or not observable due to weather conditions in certain parts of the road. As it can be seen from the simulation results in Figure 10 both of the designed systems keep the vehicle in the lane accurately. Although increasing the operating speed increases the lateral deviation, especially for the curved parts still the deviation is under 12 cm for the highest vehicle speed of 30 m/s. Also, the system works in different operating conditions where the vehicle weight and the speed are varied over the uncertainty box. As a future work, this simulator will be improved for level 3 to 4 autonomous driving scenario simulations and presented work will be tested in the TRC test track shown in Figure 9.

References

[1]. SAE International, "Automated Driving: Levels of Driving Automation Are Defined in New SAE International Standard J3016," https://www.sae.org/misc/pdfs/automated_driving.pdf, accessed Oct. 2017.

[2]. Ö. Tunçer, L. Güvenç, F. Coşkun and E. Karsligil, "Vision Based Lane Keeping Assistance Control Triggered by a Driver Inattention Monitor, "*2010 IEEE International Conference on Systems, Man and Cybernetics*, Istanbul, 2010, 289-297. doi:10.1109/ICSMC.2010.5642254

[3]. C. M. Kang et al., "Lateral Control for Autonomous Lane Keeping System on Highways," *15th International Conference on Control, Automation and Systems (ICCAS)*, Busan, 2015, 1728-1733. doi:10.1109/ICCAS.2015.7364643

[4]. S. Yenikaya, G. Yenikaya, and E. Duven, "Keeping the vehicle on the road: A survey on on-road lane detection systems," *ACM Comput. Surv.*, 46, 2:1-2:43, July 2013.

[5]. M. Emirler, H. Wang, B. Aksun Güvenç, L. Güvenç. "Automated Robust Path Following Control Based on Calculation of Lateral Deviation and Yaw Angle Error," *ASME. Dynamic Systems and Control Conference*, Volume 3, Columbus, OH, 2015. doi:10.1115/DSCC2015-9856.

[6]. Rossetter, E.J., "A Potential Field Framework for Active Vehicle Lanekeeping Assistance," PhD thesis, Stanford University, 2003.

[7]. L. Guvenc, B. Aksun-Guvenc,. B. Demirel, M.T. Emirler, Control of Mechatronic Systems. IET, 2017.

Contact Information

Automated Driving Lab
930 Kinnear Road, Columbus, OH 43214
cantas.1@osu.edu, guvenc.1@osu.edu

Acknowledgments

This paper is based upon work supported by the National Science Foundation under Grant No.:1640308 for the NIST GCTC Smart City EAGER project UNIFY titled: Unified and Scalable Architecture for Low Speed Automated Shuttle Deployment in a Smart City, by the U.S. Department of Transportation Mobility 21: National University Transportation Center for Improving Mobility (CMU) sub-project titled: Smart Shuttle: Model Based Design and Evaluation of Automated On-Demand Shuttles for Solving the First-Mile and Last-Mile Problem in a Smart City and the Ohio State University Center for Automotive Research Membership Project titled: Use of OSU CAR Connected and Automated Driving Vehicle HiL Simulator for Developing Basic Highway Chauffeur and Smart City Autonomous Shuttle Algorithms.

Authors would like to thank to NVIDIA for their GPU donation to Automated Driving Lab. Simulations presented in this work is done with a computer equipped with a GeForce - GTX TITAN graphics card.

The first author of this paper would like to thank his colleagues in the Automated Driving Lab at the Ohio State University for their support and the Ministry of National Education of the Republic of Turkey for partially supporting his education.

Definitions/Abbreviations

ADAS - Advanced Driver Assistance Systems

GPS - Global Positioning System

RTK - Real-Time Kinematic

HIL - Hardware in the Loop

CRB - Complex Root Boundary

RRB - Real Root Boundary

PD - Proportional, Derivative

CAN - Controller Area Network

DSRC - Dedicated Short Range Communication

TRC - Transportation Research Center

ISSN 0148-7191

2018-01-1086 Published 03 Apr 2018

Use of Robust DOB/CDOB Compensation to Improve Autonomous Vehicle Path Following Performance in the Presence of Model Uncertainty, CAN Bus Delays and External Disturbances

Haoan Wang and Levent Guvenc The Ohio State University

Citation: Wang, H. and Guvenc, L., "Use of Robust DOB/CDOB Compensation to Improve Autonomous Vehicle Path Following Performance in the Presence of Model Uncertainty, CAN Bus Delays and External Disturbances," SAE Technical Paper 2018-01-1086, 2018, doi:10.4271/2018-01-1086.

Abstract

Autonomous vehicle technology has been developing rapidly in recent years. Vehicle parametric uncertainty in the vehicle model, variable time delays in the CAN bus based sensor and actuator command interfaces, changes in vehicle sped, sensitivity to external disturbances like side wind and changes in road friction coefficient are factors that affect autonomous driving systems like they have affected ADAS and active safety systems in the past. This paper presents a robust control architecture for automated driving systems for handling the abovementioned problems. A path tracking control system is chosen as the proof-of-concept demonstration application in this paper. A disturbance observer (DOB) is embedded within the steering to path error automated driving loop to handle uncertain parameters such as vehicle mass, vehicle velocities and road friction coefficient and to reject yaw moment disturbances. The compensation of vehicle model with the embedded disturbance observer forces it to behave like its nominal model within the bandwidth of the disturbance observer. A parameter space approach based steering controller is then used to optimize performance. The proposed method demonstrates good disturbance rejection and achieves stability robustness. The variable time delay from the "steer-by-wire" system in an actual vehicle can also lead to stability issues since it adds large negative phase angle to the plant frequency response and tends to destabilize it. A communication disturbance observer (CDOB) based time delay compensation approach that does not require exact knowledge of this time delay is embedded into the steering actuation loop to handle this problem. Stability analysis of both DOB and CDOB compensation system are presented in this paper. Extensive model-in-the-loop simulations were performed to test the designed disturbance observer and CDOB systems and show reduced path following errors in the presence of uncertainty, disturbances and time delay. A validated model of our 2017 Ford Fusion Hybrid research autonomous vehicle is used in the simulation analyses. Simulation results verify the performance enhancement of the vehicle path following control with proposed DOB and CDOB structure. A HiL simulator that uses a validated CarSim model with sensors and traffic will be used later to verify the real time capability of our approach.

I. Introduction

With the rapid development of autonomous vehicles, automatic steering technique plays an important role in autonomous research area. Many different steering control methods have been proposed in the literature. A path following algorithm named Circular Look Ahead (CLA) steering control was proposed in [1] which can control a car to precisely follow a path even on a curvy road. The waypoint tracking method of autonomous navigation is presented in [2] using the Point to Point algorithm with position and heading measurements from GPS receivers. Model predictive control based vehicle front wheel steering is applied to track the collision free path in [3] and has the capability to deal with a wide variety of process control constraints systematically. However, regular controllers are usually designed without considering external disturbances and model uncertainty in mind, which may lead to performance degradation in path tracking. To solve such problem, a disturbance observer (DOB) is added into the control system to achieve insensitivity to modeling error and disturbance rejection. The disturbance observer was firstly proposed by Ohnishi [4] and further developed by Umeno and Hori [5]. Later, DOB has been applied in mechatronic applications in the literature. In [6], robustness of disturbance observer is added to the model of electrohydraulic system considering the case in which the plant has large parametric variation. A new active front steering controller design for electric vehicle stability using disturbance observer was proposed in [7].

Time delay is another significant issue which generally exists in the network-based control system. With the occurrence of time delay, large negative phase angles are added to the frequency response of vehicle plant which may lead to instability of the system. The Smith predictor has been widely used for a long time and extended for different cases such as

[8, 9]. Smith predictor has the advantage of easy implementation and simplicity in understanding. However, time delay model and model accuracy in the knowledge of time delay are required to ensure no degradation of compensation performance. Communication disturbance observer was proposed as another time delay compensation approach. This method was firstly applied in the bilateral teleoperation systems [10] and has been extended to robust time delayed control system in [11, 12]. The communication disturbance observer can be implemented to a wider range of applications since the accuracy of time delay is not necessary and also can be used for plants with variable time delay.

Motivated by the limitations of single DOB and CDOB compensated system. [13] proposed a double disturbance observer (DDOB) structure in the wireless motion control system design, which embedded both DOB and CDOB in one control system. The proposed approach effectively realized time delay compensation and external disturbance rejection simultaneously.

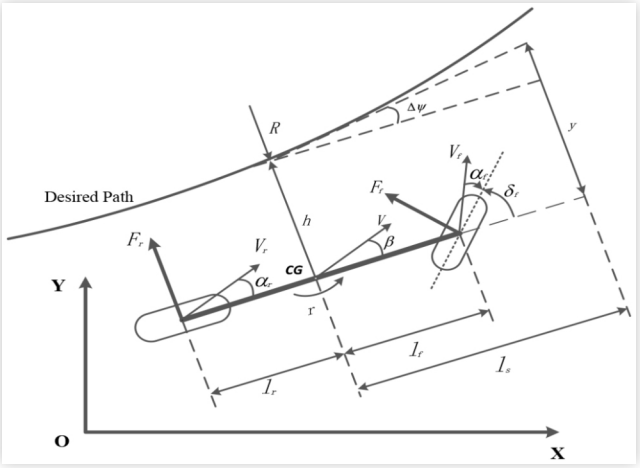
Although DOB and CDOB have been applied to many different applications in the literature, there are few DOB and CDOB applications in autonomous vehicle system which will be a potential area of progress. Furthermore, DOB, CDOB and DDOB compensated structure investigated in this paper was applied in the autonomous vehicle path following control system separately as a new topic in the field of automated vehicle. Uncertain parameters including vehicle mass, vehicle velocities and road friction coefficient and disturbances like road curvatures are firstly focused on. A disturbance observer (DOB) is embedded within the steering to path error automated driving loop to reject disturbances and handle model uncertainty. Then, time delay was taken into account and CDOB was embedded into the steering actuation loop to handle the problem. Robustness of stability of both structures is analyzed and validated. In order to deal with time delay and external disturbances simultaneously, DDOB compensated structure was used. Simulation results show that DDOB works better than DOB or CDOB compensated systems and all three compensated systems demonstrate good path following performance compared with PD feedback control system.

The rest of this paper is organized as follows. The vehicle steering model and vehicle parameters are presented in Section II. Disturbance observer and communication disturbance observer and are introduced in Section III and Section IV respectively. In Section V, robust PD controller and Q filter are designed. Also, robust stability analysis of both DOB and CDOB design are demonstrated. Section VI proposed double disturbance observer and Section VII shows autonomous vehicle path following simulation results using DOB compensation system, CDOB compensation system and results comparison between DDOB and CDOB. The paper ends with conclusion and recommendations for future work in Section VII.

II. Vehicle Model

By combining the two front wheels together and two rear wheels together of a four wheel car, a single track vehicle model is formed as shown in Figure 1 to model the steering dynamics.

FIGURE 1 Diagram of the vehicle model



The parameters of the vehicle model are given in Table 1. The state space model can be described as:

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \dot{\Delta\psi} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} a11 & a12 & 0 & 0 \\ a21 & a22 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ V & l_s & V & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ \Delta\psi \\ y \end{bmatrix} + \begin{bmatrix} b11 \\ b21 \\ 0 \\ 0 \end{bmatrix} \delta_f + \begin{bmatrix} 0 \\ 0 \\ -V \\ 0 \end{bmatrix} \rho_{ref} \quad (1)$$

where

$$\begin{aligned} &a11 = -\left(c_r + c_f\right)/\tilde{m}V, a12 = -1 + \left(c_r l_r - c_f l_f\right)/\tilde{m}V^2 \\ &a21 = \left(C_r l_r - c_f l_f\right)/J, a22 = -\left(c_r l_r^2 + c_f l_f^2\right)/JV^2 \\ &b11 = c_f/\tilde{m}V, b12 = c_f lf/J \end{aligned} \quad (2)$$

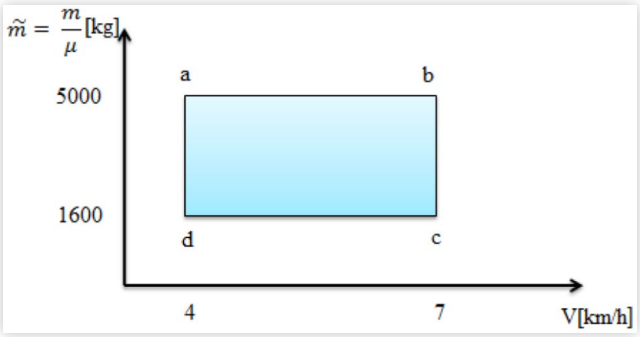
The standard form of vehicle steering dynamics can be written as (3) according to (1):

$$\dot{x} = Ax + Bu \quad (3)$$

TABLE 1 Parameters of the vehicle model

β	vehicle side slip angle [rad]
subscript f	front tires
V	vehicle velocity [m/s]
δ_f	front wheel steering angle [rad]
J	yaw moment of inertia [3728 kgm^2]
C_r	rear cornering stiffness [50,000 N/rad]
l_f	distance from CG to front axle [1.3008 m]
l_r	distance from CG to rear axle [1.5453 m]
$\rho_{ref} = 1/R$	curvature of path [1/m]
r	vehicle yaw rate [rad/s]
subscript r	rear tires
$\Delta\psi$	yaw orientation error with respect to path [rad]
y	lateral deviation [m]
C_f	front cornering stiffness [195,000 N/rad]
m	vehicle mass [2,000 kg]

FIGURE. 2 Parametric Uncertainty Box



The transfer function from front wheel steering angle δ_f to the lateral deviation y is represented by equation (4). Note that front wheel steered vehicle is considered in this paper so that δr = 0.

$$\frac{y}{\delta_f} = G_{\delta_f} = [0\,0\,0\,1](sI - A)^{-1}\begin{bmatrix} b11 \\ b21 \\ 0 \\ 0 \end{bmatrix} \tag{4}$$

The curvature ρ_{ref} of the desired path is taken as an external disturbance. The transfer function from the road curvature ρ_{ref} to the lateral deviation from the desired path can be represented as:

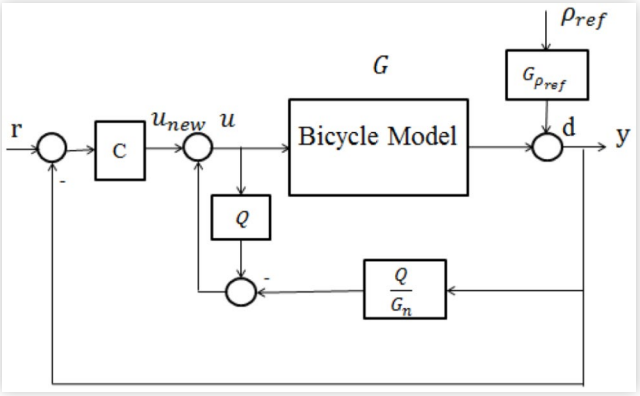
$$\frac{y}{\rho_{ref}} = G_{\rho_{ref}} = [0\,0\,0\,1](sI - A)^{-1}\begin{bmatrix} 0 \\ 0 \\ -V \\ 0 \end{bmatrix} \tag{5}$$

The vehicle velocity V, vehicle virtual mass m and road friction coefficient μ are regarded as uncertainty parameters with nominal parameter values of $V_n = 5km/h$, $\mu_n = 1$ and $m_n = 2{,}000\ kg$. The operating ranges used were $V \in [4, 7]km/h$, $\mu \in [0.4,1]$ and mass $m \in [1600, 2000]$ kg from no load to full load. The virtual mass $\tilde{m} = \frac{m}{\mu}$ is then within the range of $\tilde{m} = \frac{m}{\mu} \in [1600,5000]$ kg. The uncertainty parameters are illustrated in the uncertainty box shown in Figure 2. Four vertices labeled by a, b, c, d in the uncertainty box are used to evaluate the performance and robustness of the disturbance observer compensated system.

III. Disturbance Observer

The block diagram of the closed-loop control system with disturbance observer compensation is depicted in Figure 3. In the block diagram, robust PD feedback controller is used as a baseline controller which is designed based on the nominal model of the vehicle. Q is the low pass filter to be selected and its bandwidth determines the bandwidth of model regulation and disturbance rejection. System plant G is formulated by

FIGURE 3 Disturbance observer compensated control system



taking both model uncertainty Δ_m and external disturbance d into account. The vehicle input - output relation becomes

$$y = Gu + d = \left(G_n(1 + \Delta m)\right)u + d \tag{6}$$

where G_n is the desired model of plant and G represents the actual plant. The goal in disturbance observer design is to obtain

$$y = G_n u_{new} \tag{7}$$

as the input-output relation in the presence of model uncertainty Δ_m and external disturbance d. u_{new} is regarded as a new steering input which is derived as follows. By considering model uncertainty and external disturbance as an extended disturbance e, equation (6) can be rewritten as (8)

$$y = \left(G_n(1 + \Delta m)\right)u + d = G_n u + e \tag{8}$$

Combining equation (7) with equation (8), the new control input u_{new} is represented as

$$u_{new} = u + \frac{e}{G_n} \tag{9}$$

and

$$u = u_{new} - \frac{e}{G_n} = u_{new} - \frac{y}{G_n} + u \tag{10}$$

In order to limit the compensation to a low frequency range to avoid stability robustness problem at high frequency, the feedback signals in (10) are multiplied by the low pass filter Q and implementation equation becomes

$$u = u_{new} - \frac{Q}{G_n}y + Qu \tag{11}$$

Based on the block diagram, the model regulation and disturbance rejection transfer function can be derived as equations (12) (13). It can be seen that Q should be a unity gain low pass filter to make sure as Q→1, $\frac{y}{u_{new}} \to G_n$ for model regulation and $\frac{y}{d} \to 0$ to achieve disturbance rejection.

$$\frac{y}{u_{new}} = \frac{G_n G}{GQ + G_n(1-Q)} \tag{12}$$

$$\frac{y}{d} = \frac{G_n(1-Q)}{GQ + G_n(1-Q)} \tag{13}$$

IV. Communication Disturbance Observer

Although disturbance observer shows good performance in model regulation and disturbance rejection, performance will degrade when there exists time delay in the system. Communication disturbance observer is applied to compensate the time delay. For CDOB design, time delay is considered as a disturbance d that is acting on the system as illustrated in Figure 4 and the aim is to obtain disturbance estimation $\hat{d}$. From Figure 4, we can get equation (14) and it can be rewritten as (15). Then, the estimated disturbance $\hat{d}$ is obtained by multiplying d with Q to ensure causality as shown in equation (16).

$$y = G_n(u - d) \tag{14}$$

$$d = u - G_n^{-1} y \tag{15}$$

$$\hat{d} = Q(u - G_n^{-1} y) \tag{16}$$

According to network disturbance concept as depicted in Figure 5, $\hat{d}$ can be also expressed as equation (17)

$$\hat{d} = u - ue^{-T} \tag{17}$$

where u is system input and T is time delay.

In this way, the estimated disturbance $\hat{d}$ is used to compensate the time delay effect in the feedback signal. Figure 6 shows the structure of the communication disturbance observer compensated control system. There is a 0.08 sec time delay between actual steering wheel input and desired steering wheel input, which is compensated by the proposed CDOB. It is seen that there are two blocks in the structure: the left block is time delay compensation and the right block is network disturbance estimation.

Therefore, the closed loop transfer function of the system is written as (18):

FIGURE 4 Classic disturbance observer

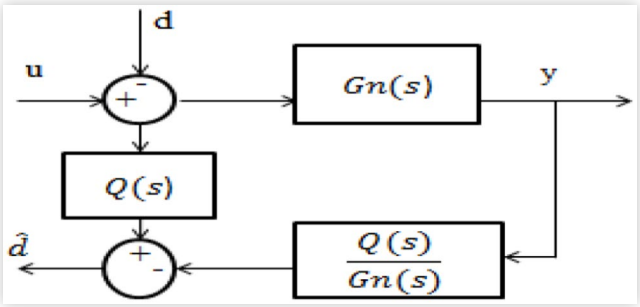


FIGURE 5 Concept of network disturbance

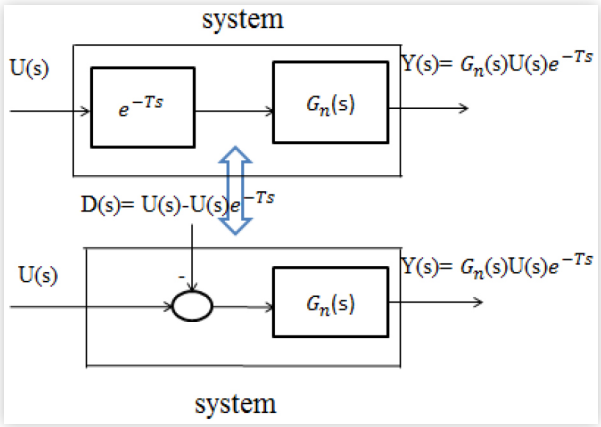
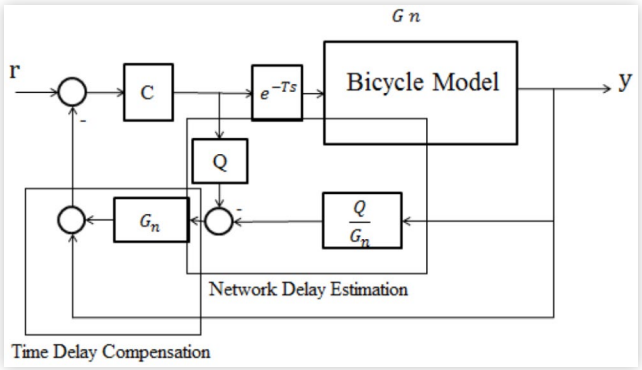


FIGURE 6 Communication disturbance observer compensated control system



$$\frac{y}{r} = \frac{CG_n(s)e^{-Ts}}{1 + CG_nQ + CG_n(1-Q)e^{-T}} \tag{18}$$

The Q filter is usually chosen as a low pass filter due to the fact that reference operates in low frequency. From equation (18), we can see that it is ideal to make $Q = 1$ in low frequency so that the denominator of the transfer function will have no time delay elements.

VI. Design Analysis

A. Robust PD Controller Design

In the proposed robust control system, a parameter space approach based PD controller is designed. The details of parameter space method can be found in [14, 15, 16]. Robust PD controller is designed based on the nominal plant G_n. Using the parameter space method, D-stability boundaries are depicted in Figure 7, where settling time constraint σ is set to be 0.3, damping constraint θ is 135° and bandwidth constraint R is assigned as 1.3 rad/sec. The overall solution

FIGURE 7 D-stability boundary

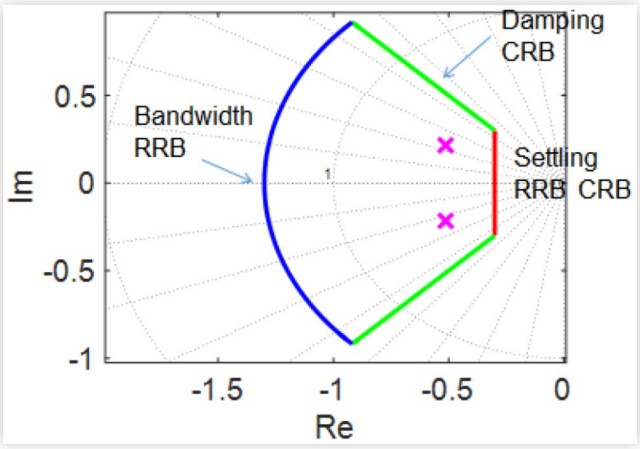
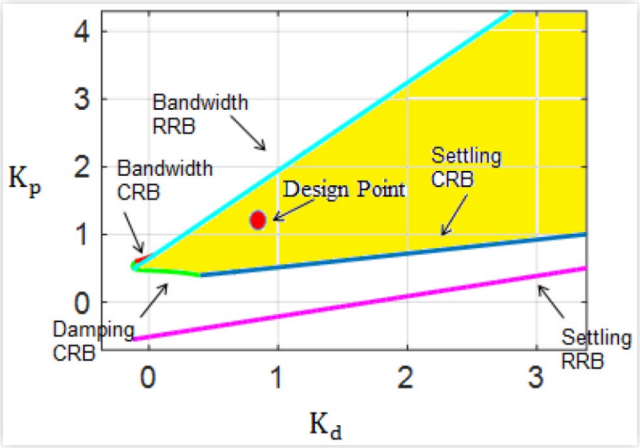


FIGURE 8 D-stability solution region



region which satisfies the stability requirements are calculated and plotted as illustrated in Figure 8. In Figure 8, K_p and K_d are two free design parameters and we select K_p=1.0596, K_d=0.939.

B. Q Filter Design and Verification of Robust Stability

Q filter is designed to be a low pass filter as discussed before for model regulation, disturbance rejection and time delay compensation. For appropriate orders of the Q filter, since the relative degree of low pass filter Q is chosen to be at least equal to the relative degree of G_n for causality of Q/G_n. The vehicle path following transfer function model G_n obtained from equation (4) is calculated as equation (19). Therefore, a second order filter Q is designed as defined in equation (20). For the cutoff frequency of Q filter, it should be appropriately selected in order to make ascertain the stability robustness of the system.

$$\mathrm{Gn}(s)=\frac{227.6\,s^2+8.479*10^4\,s+3.627*10^4}{s^4+459.2s^3+3.329e04\,s^2} \tag{19}$$

$$Q(s)=\frac{1}{(\tau s+1)^2} \tag{20}$$

where $\tau=1/\omega_c$.

B.1. DOB Compensation System Robust Stability Analysis We have obtained that Q must go to unity for model regulation and disturbance rejection. According to the characteristic equation (21), equation (22) is derived since $Q\to 1$, $G_n(1-Q)\to 0$.

$$G_n(1-Q)+G_n(1+\Delta_m)Q=0 \tag{21}$$

$$G_n(1+\Delta_m Q)=0 \tag{22}$$

Based on the small gain theory [17], the sufficient condition for robust stability can be written as equation (23). Combining variations covering all vertices from uncertainty box in Figure 2, real parametric variation of vehicle mass m, vehicle velocity V and road friction coefficient μ are converted to an approximate unstructured multiplicative uncertainty Δ_m. Figure 9 illustrates the satisfaction of disturbance observer design requirement when the cutoff frequency ω_c of Q is 5 rad/s.

$$|Q|<\left|\frac{1}{\Delta_m}\right|,\forall\omega \tag{23}$$

The frequency responses of four corners of the uncertainty box are also studied to illustrate the robustness of DOB compensated system. PD feedback controller was applied to both systems with and without DOB compensation, the input-output behavior |y/r| are shown below. It can be seen that at low frequency there are larger variations in figure 10 as the operating point is varied than in second figure. In figure 11, the frequency response magnitudes are close to each other at low frequency.

B.2. CDOB Compensation System Robust Stability Analysis According to the Nyquist stability criterion, robust stability of uncertain system can be

FIGURE 9 Magnitude of Q and $\frac{1}{\Delta_m}$ for stability of robustness

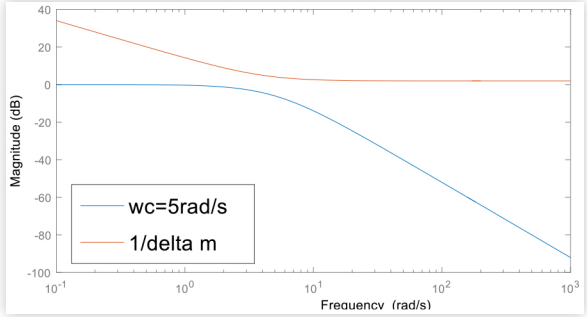


FIGURE 10 |y/r| for the four vertices of uncertainty box without disturbance observer

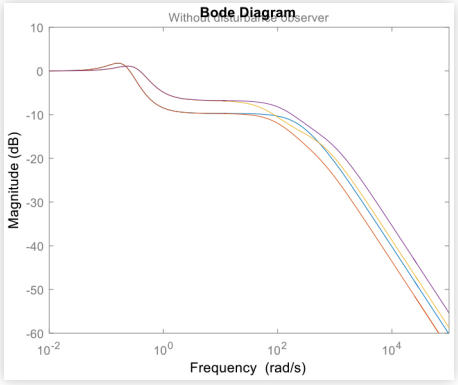
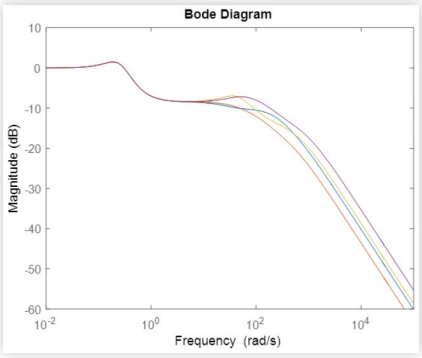


FIGURE 11 |y/r| for the four vertices of uncertainty box with disturbance observer



guaranteed if L does not encircle point (−1, 0), which can be expressed as equation (24):

$$\left|\Delta_m(j\omega)L_n(j\omega)\right| < \left|1+L_n(j\omega)\right|, \forall\omega \tag{24}$$

or equivalently,

$$\left|\frac{\Delta_m(j\omega)L_n(j\omega)}{1+L_n(j\omega)}\right| < 1, \forall\omega \leftrightarrow \left|\frac{L_n(j\omega)}{1+L_n(j\omega)}\right| < \left|\frac{1}{\Delta_m(j\omega)}\right|, \forall\omega \tag{25}$$

where L_n is represented as in equation (26) in this system, which is the nominal loop transfer function.

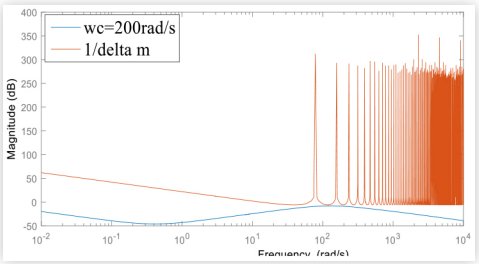
$$L_n = \frac{C(1-Q)G_n e^{-Ts}}{1+CG_nQ} \tag{26}$$

Consider time delay e^{-Ts} as the source of unmodeled dynamics, the model uncertainty Δ_m is then given by equation (27):

$$\Delta_m(s) = e^{-Ts} - 1 \tag{27}$$

Figure 12 illustrates that with the choice of $\omega_c = 200 rad/s$, the system is stable as blue line is below the red one with no intersection.

FIGURE 12 Magnitude of $\frac{L_n(j\omega)}{1+L_n(j\omega)}$ and $\frac{1}{\Delta_m}$ for stability of robustness



VII. Double Disturbance Observer

In order to deal with disturbance rejection and time delay simultaneously, DDOB compensated control system was used and its structure was depicted in figure 13. The lower block has the same structure as the CDOB and the upper block is a disturbance observer for disturbance rejection.

VIII. Simulation Studies

Simulations are performed to check the performance enhancement in the autonomous vehicle path following control with proposed DOB and CDOB structure. The desired path to be followed is an elliptical route as shown in Figure 14 and the curvature of the path is depicted in Figure 15. Figures 16 to 20 compares the path following errors of robust PD feedback controller system with and without disturbance observer compensation. For uncertain parameters, Figure 16 to Figure 19 takes the four corners of parametric uncertainty box into account. In Figure 20, external disturbance is added into the system due to road curvature input ρ_{ref}. It can be seen that with DOB added into the control system, the path following error decreases obviously as shown in Figure 16-20, which verify that DOB effectively deals with model regulation and disturbance rejection. Comparison about

FIGURE 13 Double disturbance observer compensated control system

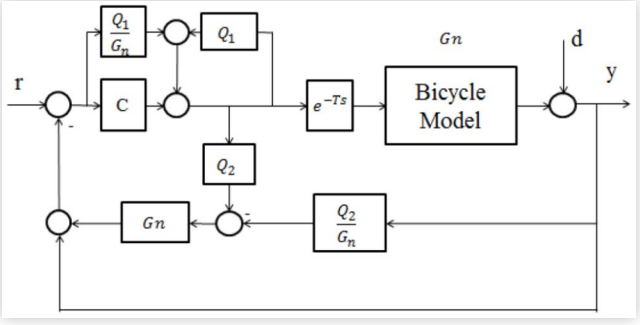


FIGURE 14 Desired path used in the simulation

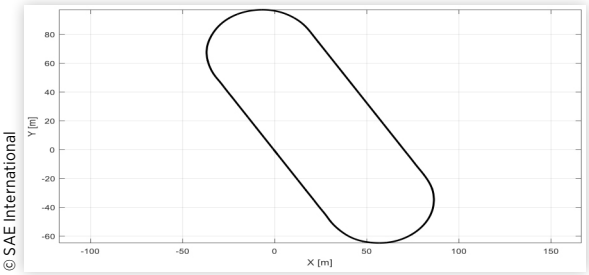


FIGURE 15 Curvature of the desired path

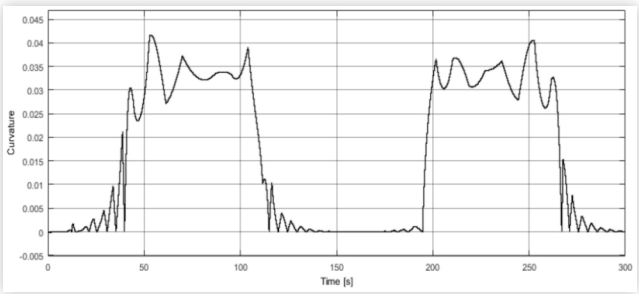


FIGURE 16 Lateral deviation with and without DOB at corner *a* for model uncertainty

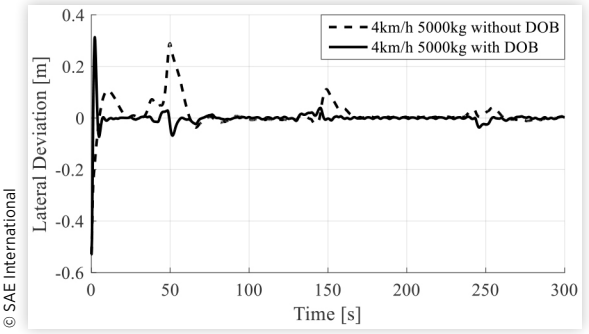


FIGURE 17 Lateral deviation with and without DOB at corner *b* for model uncertainty

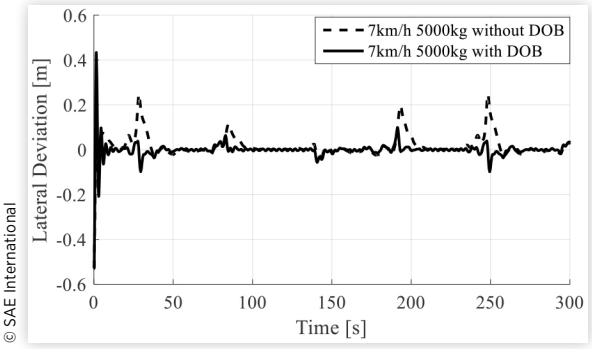


FIGURE 18 Lateral deviation with and without DOB at corner *c* for model uncertainty

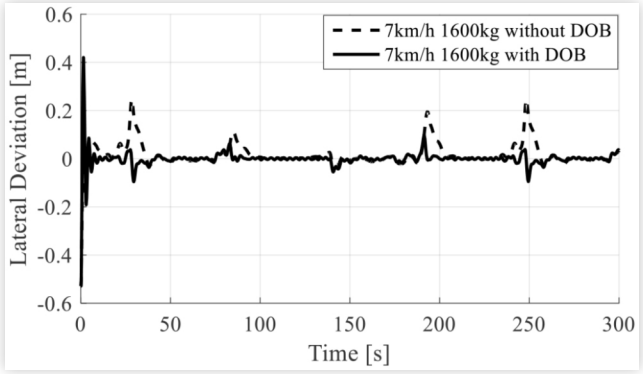


FIGURE 19 Lateral deviation with and without DOB at corner *d* for model uncertainty

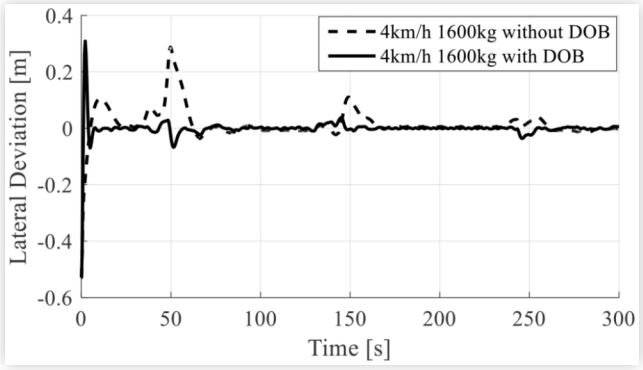


FIGURE 20 Lateral deviation with and without DOB for disturbance input

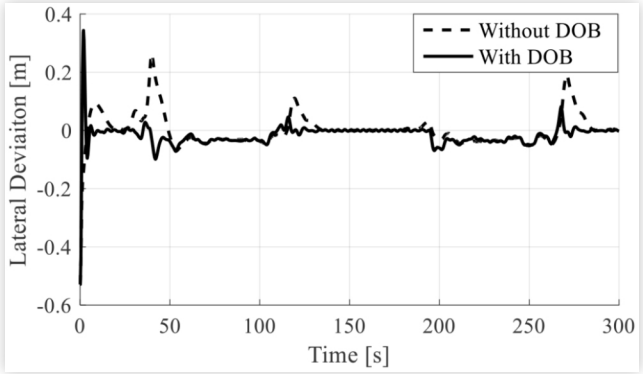


TABLE II Comparison of RMS tracking errors between PD and PD with DOB

	4 km/h 1600 kg	4 km/h 5000 kg	7 km/h 1600 kg	7 km/h 5000 kg
PD	0.0580	0.0581	0.0523	0.0526
PD + DOB	0.0320	0.0336	0.0359	0.0370

root-mean-square (RMS) errors of feedback control with DOB and feedback control only tabulated in Table II also illustrates the smaller errors in the presence of disturbance observer.

Figure 21 compares the lateral deviation of system with and without communication disturbance observer compensation by considering 0.08 sec CAN bus delay for steering actuation. It shows that CDOB compensates the time delay effect in the closed loop system and has reduced errors. From Figure 22, we can see that CDOB compensated control

FIGURE 21 Lateral deviation with and without CDOB for time delay

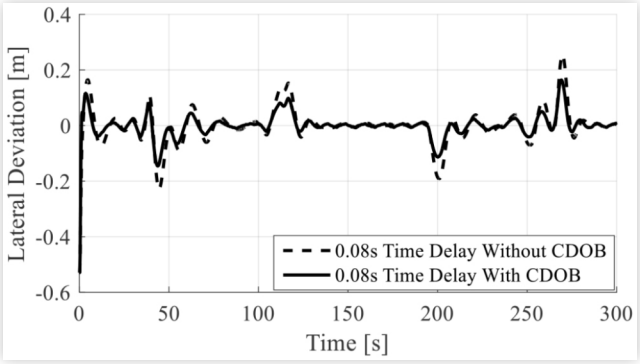


FIGURE 22 Steering angle with and without CDOB for time delay

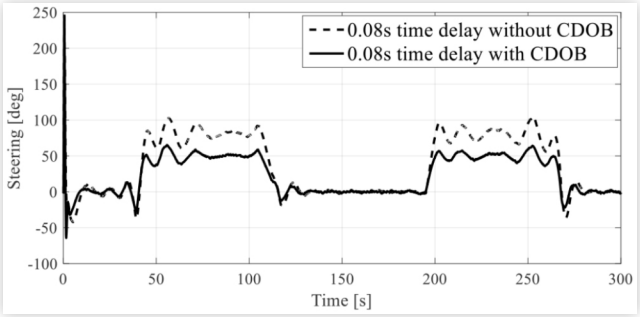
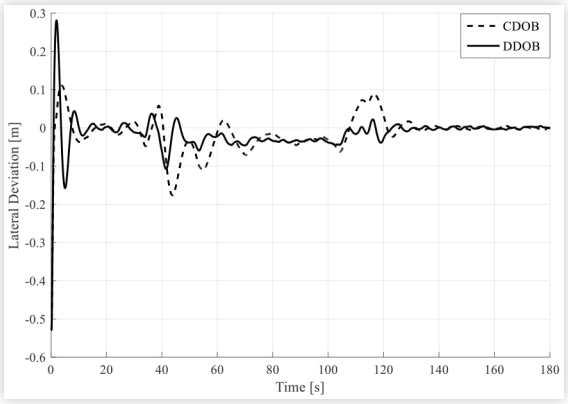


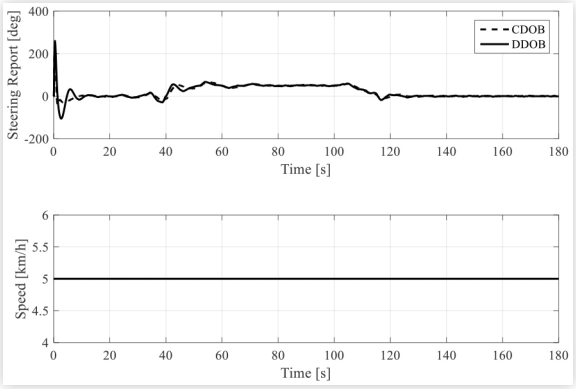
FIGURE 23 lateral deviation of CDOB and DDOB compensated system



system has smaller steering angle compared with PD only controlled system. These results show a better path following performance of CDOB compensation control system.

Figures 23 and 24 compare the lateral deviation and steering angle and speed of CDOB and DDOB compensated system when both 0.08 sec time delay and disturbance input exist in the system simultaneously. It can be seen that both systems have similar steering angle and DDOB works better than CDOB with reduced path following errors.

FIGURE 24 Steering angle and speed of CDOB and DDOB compensated system



IX. Conclusion and Future Work

In this paper, the disturbance observer was applied to deal with model uncertainty and external disturbance and communication disturbance observer was used to handle CAN bus delay in order to realize performance enhancement of autonomous vehicle path following control. Also, double disturbance observer was applied in the vehicle path following control system to achieve model regulation, disturbance rejection and time delay simultaneously. Robust PD controller was designed based on the nominal model and Q filter design was presented. Robust stability of DOB and CDOB was studied analytically and verified. Simulation results were given to evaluate the vehicle path following performance and verify the proposed control algorithm.

In the future work, varying time delay will be studied with CDOB compensated system. More model-in-the-loop and hardware-in-the-loop (HiL) simulations will be performed to further test the designed DOB, CDOB and DDOB systems.

References

1. Hsieh, M.F. and Ozguner, U., "A Path Following Control Algorithm for Urban Driving," *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, IEEE, 2008, 227-231.
2. Cha, Y.C., Lee, K.S., Lee, D.S., Park, H.G. et al., "A Lateral Controller Design for an Unmanned Vehicle," *Advanced Intelligent Mechatronics (AIM), 2011 IEEE/ASME International Conference on*, IEEE, 2011, 283-286.
3. Wang, H., Cao, Y., Güvenç, B.A., and Güvenç, L.,"MPC Based Automated Steering of a Low Speed Shuttle for Socially Acceptable Accident Avoidance," *ASME 2016 Dynamic Systems and Control Conference*, American Society of Mechanical Engineers, 2016, V002T30A004-V002T30A004.
4. Ohnishi, K., "A New Servo Method in Mechatronics," *Trans. Japanese Soc. Elect. Eng.* 107-D:83-86, 1987.

5. Senjyu, T., Ashimine, S., and Uezato, K., "Robust Speed Control of DC Servomotors Using Fuzzy Reasoning," *Industrial Electronics, Control, and Instrumentation, 1996, Proceedings of the 1996 IEEE IECON 22nd International Conference on*, Vol. 3, IEEE, 1996, 1365-1370.
6. Guvenc, B.A. and Guvenc, L.," "Robustness of Disturbance Observers in the Presence of Structured Real Parametric Uncertainty," *American Control Conference, 2001. Proceedings of the 2001*, Vol. 6, IEEE, 2001, 4222-4227.
7. Lee, S.-H., Lee, Y.O., Son, Y., and Chung, C.C., "Robust Active Steering Control of Autonomous Vehicles: A State Space Disturbance Observer Approach," *Control, Automation and Systems (ICCAS), 2011 11th International Conference on*, IEEE, 2011, 596-598.
8. Alevisakis, G. and Seborg, D.E., "An Extension of the Smith Predictor Method to Multivariable Linear Systems Containing Time Delays," *International Journal of Control* 17(3):541-551, 1973.
9. Matausek, M.R. and Micic, A.D., "A Modified Smith Predictor for Controlling a Process with an Integrator and Long Dead-Time," *IEEE Transactions on Automatic Control* 41(8):1199-1203, 1996.
10. Natori, K., Tsuji, T., Ohnishi, K., Hace, A. et al., "Robust bilateral control with internet communication," *Industrial Electronics Society, 2004. IECON 2004. 30th Annual Conference of IEEE*, Vol. 3, IEEE, 2004, 2321-2326.
11. Natori, K., Oboe, R., and Ohnishi, K., "Stability Analysis and Practical Design Procedure of Time Delayed Control Systems with Communication Disturbance Observer," *IEEE Transactions on Industrial Informatics* 4(3):185-197, 2008.
12. Emirler, M.T., Güvenç, B.A., and Güvenç, L., "Communication Disturbance Observer Approach to Control of Integral Plant with Time Delay," *Control Conference (ASCC), 2013 9th Asian*, IEEE, 2013, 1-6.
13. Zhang, W. et al., "Robust Time Delay Compensation in a Wireless Motion Control System with Double Disturbance Observers," *American Control Conference (ACC), 2015*, IEEE, 2015.
14. Ackermann, J., "*Robust Control: The Parameter Space Approach*," (Springer Science & Business Media, 2012).
15. Emirler, M.T., Uygan, İ.M.C., Güvenç, B.A., and Güvenç, L., "Robust PID Steering Control in Parameter Space for Highly Automated Driving," *International Journal of Vehicular Technology* 2014, 2014.
16. Emirler, M.T., Wang, H., Aksun Güvenç, B., and Güvenç, L., "Automated Robust Path Following Control Based on Calculation of Lateral Deviation and Yaw Angle Error," *ASME Dynamic Systems and Control Conference (DSCC)*, Vol. 30, Columbus, 28 Oct 2015.
17. Skogestad, S. and Postlethwaite, I., "*Multivariable Feedback Control: Analysis and Design*," Vol. 2 (New York, Wiley, 2007).

Contact Information

Automated Driving Lab
930 Kinnear Road, Columbus, OH 43214
guvenc.1@osu.edu

Acknowledgments

This paper is based upon work supported by the National Science Foundation under Grant No.:1640308 for the NIST GCTC Smart City EAGER project UNIFY titled: Unified and Scalable Architecture for Low Speed Automated Shuttle Deployment in a Smart City, by the U.S. Department of Transportation Mobility 21: National University Transportation Center for Improving Mobility (CMU) sub-project titled: SmartShuttle: Model Based Design and Evaluation of Automated On-Demand Shuttles for Solving the First-Mile and Last-Mile Problem in a Smart City.

Definitions/Abbreviations

AV - Autonomous Vehicle

HiL - Hardware-in-the-Loop

ISSN 0148-7191

2018-01-1182 Published 03 Apr 2018

Localization and Perception for Control and Decision Making of a Low Speed Autonomous Shuttle in a Campus Pilot Deployment

Bowen Wen, Sukru Yaren Gelbal, Bilin Aksun Guvenc, and Levent Guvenc The Ohio State University

Citation: Wen, B., Gelbal, S.Y., Aksun Guvenc, B., and Guvenc, L., "Localization and Perception for Control and Decision Making of a Low Speed Autonomous Shuttle in a Campus Pilot Deployment," SAE Technical Paper 2018-01-1182, 2018, doi:10.4271/2018-01-1182.

Abstract

Future SAE Level 4 and Level 5 autonomous vehicles will require novel applications of localization, perception, control and artificial intelligence technology in order to offer innovative and disruptive solutions to current mobility problems. This paper concentrates on low speed autonomous shuttles that are transitioning from being tested in limited traffic, dedicated routes to being deployed as SAE Level 4 automated driving vehicles in urban environments like college campuses and outdoor shopping centers within smart cities. The Ohio State University has designated a small segment in an underserved area of campus as an initial autonomous vehicle (AV) pilot test route for the deployment of low speed autonomous shuttles. This paper presents initial results of ongoing work on developing solutions to the localization and perception challenges of this planned pilot deployment. The paper treats autonomous driving with real time kinematics GPS (Global Positioning Systems) with an inertial measurement unit (IMU), combined with simultaneous localization and mapping (SLAM) with three-dimensional light detection and ranging (LIDAR) sensor, which provides solutions to scenarios where GPS is not available or a lower cost and hence lower accuracy GPS is desirable. Our in-house automated low speed electric vehicle is used in experimental evaluation and verification. In addition, the experimental vehicle has vehicle to everything (V2X) communication capability and utilizes a dedicated short-range communication (DSRC) modem. It is able to communicate with instrumented traffic lights and with pedestrians and bicyclists with DSRC enabled smartphones. Before real-world experiments, our connected and automated driving hardware in the loop (HiL) simulator with real DSRC modems is used for extensive testing of the algorithms and the low level longitudinal and lateral controllers. Real-world experiments that are reported here have been conducted in a small test area close to the Ohio State University AV pilot test route. Model-in-the-loop simulation, HiL simulation and experimental testing are used for demonstrating the feasibility and robustness of this approach to developing and evaluating low speed autonomous shuttle localization and perception algorithms for control and decision making.

Introduction

For the sake of development of smart city, the Ohio State University has designated a small segment in an underserved area of campus as an initial Autonomous Vehicle (AV) pilot test route for the deployment of SAE Level 4 low speed autonomous shuttles. This paper presents preliminary work towards proof-of-concept low speed autonomous shuttle deployment in this AV pilot test route which extends from our research lab through a 0.7 mile public road with a traffic light intersection and low speed traffic to our main research center. Our approach is to develop and test elements of this autonomous system in the private parking lot right next to our lab and in a realistic virtual replica of the AV pilot test route created within our Hardware-in-the-Loop (HiL) simulator environment. As we have already reported our work on GPS waypoint following based path tracking in our earlier papers, this paper concentrates on LIDAR SLAM based localization for path tracking, a simple decision making logic for automated driving and experimental and simulation results.

Simultaneous localization and mapping (SLAM) as first proposed by Leonard and Durrant-Whyte [1] is used to build up maps of surrounding environment with the aid of sensors such as light detection and ranging (LIDAR) sensor or camera, while also estimating the position of a robot simultaneously. A reliable and accurate solution of SLAM problems lay the foundation for an autonomous navigation and control platform [2, 3]. During the last decade, highly effective SLAM techniques have been developed and state-of-the-art two dimensional laser SLAM algorithms are now able to have satisfactory performance in terms of accuracy and computational speed (e.g. GMapping [4] and Hector SLAM [5]). In addition, researchers have successfully extended SLAM applicable scenarios from indoor environment to outdoor environment for autonomous vehicles [6, 7]. Probabilistic map distributions over environment properties followed by Bayesian inference [8] increased robustness to environment variations and dynamic obstacles, which enabled the vehicle to autonomously drive for hundreds of miles in dense traffic on narrow urban

roads. A fast implementation of incremental scan matching method based on occupancy grid map was introduced in [9] where data association was also applied to solve the multiple object tracking problem in a dynamic environment. Most of the previous work in the literature in SLAM methods has concentrated on the evaluation of localization performance whereas SLAM is used and evaluated as part of an automated path following system here.

In this paper both SLAM and GPS based localization are used for localization and path following. The SLAM system used is based on the Levenberg-Marquardt algorithm and results are compared with the Hector SLAM method. First, a reasonable convergence criteria was provided for the solution to the Levenberg-Marquardt algorithm in contrast to the fixed iteration step setting implemented in Hector SLAM, enabling more accurate and reliable pose estimation when combined with an integrated control system for smooth and comfortable path following performance. LIDAR is the only sensor that this SLAM algorithm depends on, posing an effective solutions to scenarios where GPS is not available or a lower cost and hence lower accuracy GPS is desirable. Both HiL simulations containing different traffic scenarios and relevant real world experiments were conducted. Results were demonstrated and evaluated to prove the feasibility and robustness of this approach to for low speed autonomous shuttle localization and perception algorithms for control and decision making.

The paper continues with an overview of the autonomous shuttle used in this study, the vehicle dynamics and path tracking error models. The LIDAR SLAM algorithm and experimental GPS and SLAM based path following results are presented next. This is followed by a description of the HiL simulator and how the AV test pilot route is replicated in the simulator including communication with the traffic light controller. After simulation results, the paper ends with conclusions and directions of ongoing work.

System Overview

Hardware and Platform

The vehicle used in the experiments for this study is a small, low speed, fully-electric two seater shuttle used for ride sharing applications (Dash EV). The architecture and hardware presented in this paper is general in nature and also implemented on other vehicles in our lab [10]. In order to achieve autonomous driving capability, steering, throttle and brake in this vehicle were converted to by-wire. This is done by adding actuators into the vehicle, since it was not built with them as some of the commercial sedan vehicles. For steering actuation, a smart motor was connected to the steering mechanism through gears. For brake actuation, a linear electric motor was fixed behind the brake pedal, that pushes or pulls according to the position command. For throttle, an electronic by-pass circuit was constructed and used to override the throttle signal that is sent to vehicle Electronic Control Unit (ECU) with the throttle command.

Sensors are added for localization and environmental perception after steering, throttle and brake functions are converted to drive-by-wire. These sensors are GPS, a LIDAR sensor, a Leddar sensor and a Point Grey camera used in this paper as a backup sensor. The Leddar sensor is a solid-state LIDAR which we use to get information about the obstacles in front of the vehicle. These obstacles can be vehicles, pedestrians, bicyclists etc. It is mainly used for emergency purposes, when there is an obstacle very close to the vehicle which creates a need to stop. It can be also used in low speed car following applications such as Adaptive Cruise Control (ACC) since its range is 50 m. For localization, GPS and LIDAR sensors were used. We use a differential GPS with Real-Time Kinematic (RTK) correction capability, which provides about 2-5 cm accuracy when RTK correction signals are used. Also with the differential antennas, it provides heading information even while the vehicle is stationary. LIDAR is used for both localization with SLAM and perception. It is a 16 channel Velodyne LIDAR PUCK (VLP-16) which is mounted on the top of the vehicle horizontally to guarantee a horizontal Field of View (FOV) of 360 degrees with vertical FOV of 30 degree from the surrounding environment. A 3D point cloud is generated at a frequency of 10 Hz. Theoretically, the LIDAR's maximum detection range can reach up to 100 m depending on application while in this work, detection range used for localization was set to 80 m to achieve satisfactory point cloud density and quality.

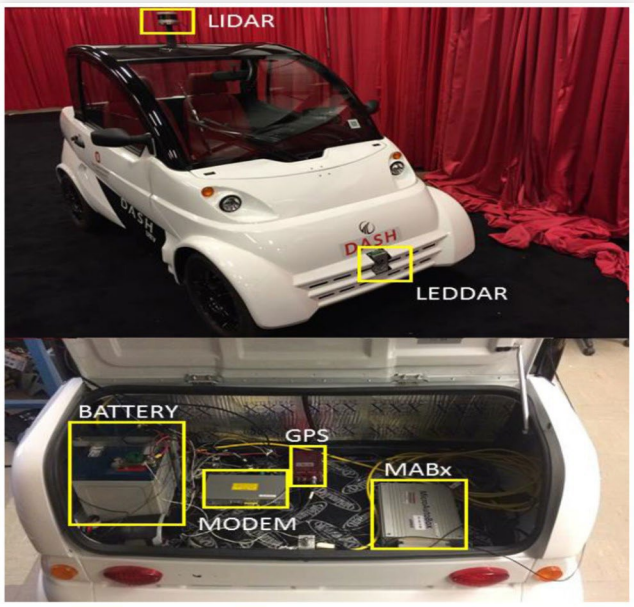
The element between the actuators and sensors is the dSPACE Microautobox (MABx) electronic control unit that is used for rapid prototyping of the low-level lateral and longitudinal direction controllers and basic decision-making algorithms created as a Simulink models. Simulink coder is used to convert the model into embedded code and the code is uploaded to the MABx device. The generated code can later be easily embedded in a series production level electronic control unit at the end of the research and development phase.

Sensors send data to the Microautobox electronic control unit with a means of communication specific to the sensor, like CAN or User Datagram Protocol (UDP) for most of our sensors. This data is fed to controllers running within the device. Controllers are created in the Simulink and outputs of the controllers are connected to output blocks that correspond to I/O ports of the Microautobox. These I/O ports are physically connected to actuators or drivers of actuators to provide reference signal and achieve autonomous driving. The experimental vehicle also has a Dedicated Short Range Communication (DSRC) modem to communicate with other vehicles, infrastructure and pedestrians with DSRC enabled smartphones. For V2X communication, all messages are sent using the standard messages of the Society of Automotive Engineers (SAE) J2735 DSRC Message Set and use the standard communication rate of 10 Hz. Devices and actuators are powered through a 12 V battery placed in the trunk of the vehicle. Some of the hardware discussed in this section is shown in Figure 1.

Vehicle Dynamics Model

The vehicle model and path following algorithm used are presented briefly in this and the following section. The lateral

FIGURE 1 Hardware on the vehicle.

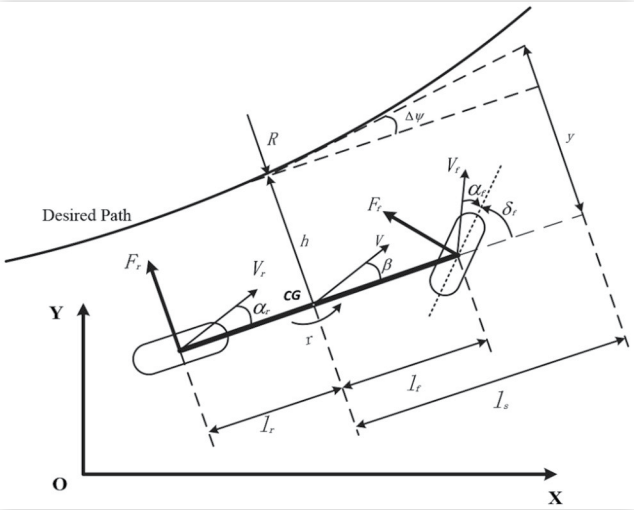


dynamics and path tracking error model is illustrated in Figure 2 and given in state space form as

$$\begin{bmatrix} \dot{\beta} \\ \dot{r} \\ \Delta\dot{\psi} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ a_{21} & a_{22} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ V & l_s & V & 0 \end{bmatrix} \begin{bmatrix} \beta \\ r \\ \Delta\psi \\ y \end{bmatrix} + \begin{bmatrix} b_{11} & 0 \\ b_{21} & 0 \\ 0 & -V \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \delta_f \\ \rho_{ref} \end{bmatrix} \quad (1)$$

where β is side slip angle, r is yaw rate, V is combination of lateral and longitudinal velocity of the vehicle body, $\Delta\Psi$ is yaw angle relative to the tangent of the desired path, l_s is the preview distance and y is lateral deviation from desired path with respect to preview distance. The control input is the steering angle δ_f. $\rho_{ref} = 1/R$ is the road curvature where R is the road radius. Other terms in the state space model are

FIGURE 2 Illustration of single track model.



$$a_{11} = -\frac{C_r + C_f}{mV}, a_{12} = -1 + \frac{C_r l_r - C_f l_f}{mV^2}$$
$$a_{21} = -\frac{C_r l_r - C_f l_f}{J}, a_{22} = -\frac{C_r l_r^2 - C_f l_f^2}{JV^2} \quad (2)$$
$$b_{11} = \frac{C_f}{mV}, b_{12} = \frac{C_f l_f}{J}$$

where m is the vehicle mass, J is the moment of inertia, μ is the road friction coefficient, C_f and C_r are the cornering stiffnesses, l_f is the distance from the Center of Gravity of the vehicle (CG) to the front axle and l_r is the distance from the CG to the rear axle.

Path Tracking Model

The low level automated driving tasks are lateral and longitudinal control. The path determination and path tracking error computation are described briefly in this section. The path tracking model consists of two parts, which are offline generation of the path and online calculation of the error according to the generated path. These parts are explained in following subsections.

A. Offline Path Generation The path following algorithm employs a pre-determined path to be provided to the autonomous vehicle to follow [11]. This map is generated from GPS waypoints where these points can be pulled from an online map or can be collected through recording during a priori manual driving. These data points are then divided into smaller groups named segments with equal number of data points for ease of formulation. These segments are both used for curve fitting and velocity profiling through the route. After dividing the road into segments, a process of fitting a third order polynomial is performed as

$$X_i(\lambda) = a_{xi}\lambda^3 + b_{xi}\lambda^2 + c_{xi}\lambda + d_{xi}$$
$$Y_i(\lambda) = a_{yi}\lambda^3 + b_{yi}\lambda^2 + c_{yi}\lambda + d_{yi} \quad (3)$$

Where i represents the segment number and terms a, b, c, d are polynomial fit coefficients for the corresponding segment. Fitting the data points provides effective replication of the curvature that the road carries and also eliminates the noise in the GPS data points. To provide a smooth transition from one segment to another by satisfying continuity of the polynomials and their first derivatives in X and Y, we use

$$X_i(1) = X_{i+1}(0)$$
$$Y_i(1) = Y_{i+1}(0) \quad (4)$$

The X and the Y points derived from the GPS latitude and longitude data using a degree to meter conversion, are fit using a single parameter λ, where λ is the variable for the fit which varies across each segment between 0 and 1, resulting in

$$\frac{dX_i(1)}{d\lambda} = \frac{dX_{i+1}(0)}{d\lambda}$$
$$\frac{dY_i(1)}{d\lambda} = \frac{dY_{i+1}(0)}{d\lambda} \quad (5)$$

B. Error Calculation After the generation of path coefficients, an error is calculated for the lateral controller to use as input. Heading and position of the vehicle is provided by means of localization, in this case either SLAM or GPS. Using these, the location of the car with respect to the path in other words the deviation from the path is calculated. This approach reduces both oscillations and steady state lateral deviation compared to calculation with respect to position only. In order to find an equivalent distance parameter to add to the first component distance error, a preview distance l_s is defined. Then, the error becomes,

$$y = h + l_s \sin(\Delta\psi) \tag{6}$$

Where $\Delta\psi$ is the net angular difference of heading of the vehicle from the heading tangent to the desired path and y is the total error of the vehicle computed at preview distance l_s as is illustrated in Figure 3.

Finally, error is fed to a robust PID controller which controls the actuation of steering of the vehicle.

SLAM Algorithm

The SLAM based localization algorithm is presented in this section. In this study, ground plane is always assumed to be flat and hence only 2D mapping and localization are required while z direction pose information in Cartesian coordinate system is not necessarily considered. In the following algorithm, the pose state vector $(x, y, \theta)^T$, comprised of 2D Cartesian coordinates and orientation angle, and thus three degrees of freedom (DOF), is used to represent the pose information for the low speed autonomous shuttle. As has been presented, the 16 channel Velodyne LIDAR can provide 3D point cloud including 360 degree FoV information of the surrounding environment. However, in this context, considering the constraint of the processor in this configuration, additional computational complexity will negatively affect the whole system in terms of real time performance. Therefore, so as to obtain planar scan information, 3D point cloud is projected into 2D space.

Before the projection, ground noise as seen in Figure 4 needs to be removed by building up occupancy height map (section A). Once the planar scan end points are obtained, scan matching process is used to align the current scan end points either to those in last frame or to the built up map in order to derive the pose transformation of the shuttle. A more reliable and accurate optimization framework inspired by Hector SLAM [5] is imposed for the scan matching process, where more reasonable stop criteria is also introduced (section B).

FIGURE 3 Illustration of error calculation.

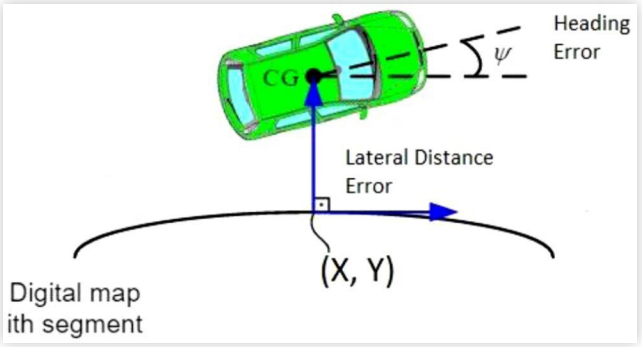
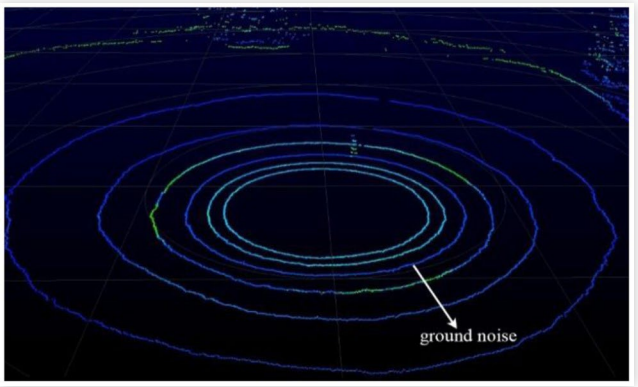


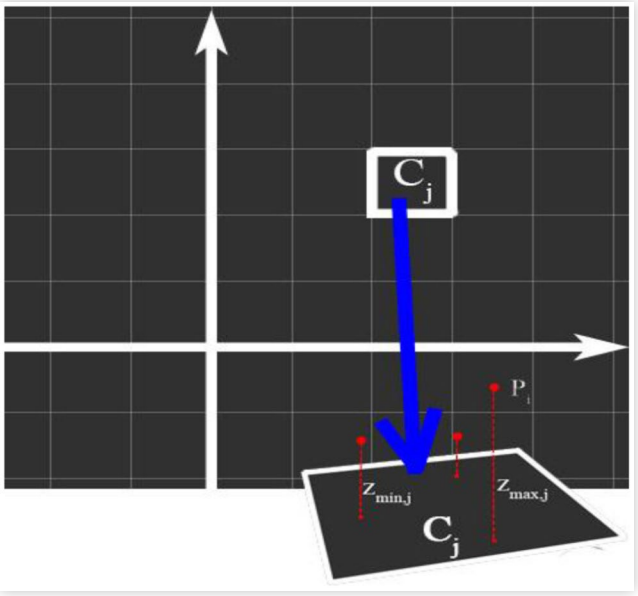
FIGURE 4 Raw 3D point cloud with ground noise.



A. Ground Noise Removal and Projection Occupancy height map is built up for ground noise removal. The LIDAR position is selected as the origin and the Cartesian coordinate system is built with the x-y plane representing the ground plane and the z axis being vertical to it. As shown in Figure 5, from a top-down view, we divide the x-y plane into many square cells of equal size. In this work, cell size is set to 0.2 m × 0.2 m. For each of the 3D points $P_i = (x_i, y_i, z_i)T$, we can find a cell C_j that it belongs to. Subsequently, for each of the cells C_j by comparing the heights of the points to a threshold h_{thres} (set to 0.3 m in this work), if

$$z_{\max,j} - z_{\min,j} \leq h_{thres} \tag{7}$$

FIGURE 5 Occupancy height map. C_j is one of the cells. Height of every cell is determined by the maximum height difference in that cell.



then this cell is defined as not occupied or comprised of ground noise and thus left as empty. If

$$z_{\max,j} - z_{\min,j} \geq h_{thres} \tag{8}$$

then this cell is defined as occupied and all the 3D points included in it are remained for further projection.

In the projection step, polar coordinate system is used to represent the position of each scan end point in 2D plane. For each 3D point P_i, its angular position in x-y plane can be expressed as:

$$\alpha_i = a\tan 2(y_i, x_i) \tag{9}$$

where *atan*2 is four-quadrant inverse tangent and hence $\alpha_i \in [-\pi, \pi]$. The range of the 2D scan corresponding to the 3D point P_i can be expressed as:

$$range_i = \sqrt{x_i^2 - y_i^2} \tag{10}$$

Note that there can be more than one projected 2D scan point in the same direction with different ranges. The ultimate range of 2D scan end point is the smallest range in that direction. Therefore, every projected 2D scan beams with their associated scan end points can be identified by angular positions, as shown in Figure 6.

B. Map Generation and Scan Matching In this work, the same map access methodology as [5] is employed, which can provide an effective solution to the accuracy limitation caused by discrete property of occupancy grid maps.

Due to the high accuracy and frequency of modern LIDAR, iterative optimization algorithms are now possible to minimize the error between obtained scan end points and built up maps, delivering the optimal alignment in the scan matching step. In this work, instead of Gauss-Newton optimization performed in Hector SLAM [5], the Levenberg-Marquardt algorithm [12] is applied to provide faster convergence for same accuracy compared with Gauss-Newton optimization, which can tremendously benefit the real time system on autonomous shuttles. Given the generated map occupancy value $M(P_m)$ corresponding to the continuous map point location $P_m = (x_m, y_m)^T$, our goal is to find the rigid transformation $\xi = (p_x, p_y, \theta)^T$ which minimizes the overall summation of occupancy error between the current scan end points and the most updated map, consequently the objective function and desired rigid transformation can be defined as:

$$E = \min \sum_{i=1}^{n} \left[1 - M\left(S_i(\xi)\right)\right]^2 \tag{11}$$

$$\xi^* = \arg\min_{\xi^*} \sum_{i=1}^{n} \left[1 - M\left(S_i(\xi)\right)\right]^2 \tag{12}$$

where n is the number of scan end points, $s_i = (s_{i,x}, s_{i,y})^T$ is the world coordinate of the transformed scan end point. $S_i(\xi)$ is a function of ξ that transforms scan end point coordinate into world system, expressed as:

$$S_i(\xi) = \begin{pmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{pmatrix} \begin{pmatrix} s_{i,x} \\ s_{i,y} \end{pmatrix} + \begin{pmatrix} p_x \\ p_y \end{pmatrix} \tag{13}$$

and $M(S_i(\xi)) \in [0, 1]$ is the occupancy value at the location given by $S_i(\xi)$. Once this is performed, the optimal transformation that best aligns the current frame with the most updated map points is obtained.

This quadratic cost function E can be solved by Levenberg-Marquardt algorithm [13] efficiently. Starting from an initial estimation of the transformation, e.g. the optimal transformation provided in last frame, ξ_0, in every iteration, a transformation update $\Delta\xi$is added to the accumulated transformation so far, ξ, so as to move forward to the minimum point and further minimize the function. Intuitively, by each iteration step, the cost function is closer to 0:

$$E = \sum_{i=1}^{n} \left[1 - M\left(S_i(\xi + \Delta\xi)\right)\right]^2 \to 0 \tag{14}$$

By replacing $M(S_i(\xi + \Delta\xi))$ with its Taylor series expansion, we obtain

$$E \approx \sum_{i=1}^{n} [1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi]^2 \to 0 \tag{15}$$

By letting the partial derivative with respect to $\Delta\xi$ equal to 0:

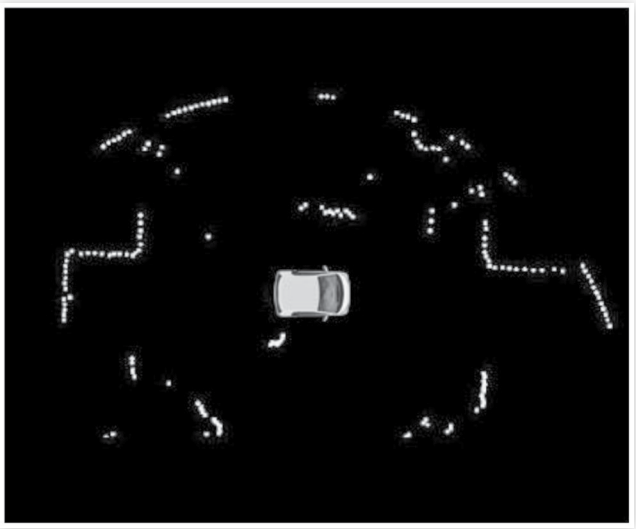
$$\begin{aligned} \frac{\partial E}{\partial(\Delta\xi)} &= 2 \sum_{i=1}^{n} \left[\nabla M\left(S_i(\xi)\right) \frac{\partial S_i(\xi)}{\partial \xi} \right]^T \cdot \ldots \\ &\cdot \sum_{i=1}^{n} \left[1 - M\left(S_i(\xi)\right) - \nabla M\left(S_i(\xi)\right) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi \right] = 0 \end{aligned} \tag{16}$$

according to Levenberg-Marquardt algorithm, the optimal solution for $\Delta\xi$ can be determined by:

$$\Delta\xi = \left(H^{-1} + \lambda I\right) \sum_{i=1}^{n} w_i \cdot \left[\nabla M\left(S_i(\xi)\right) \frac{\partial S_i(\xi)}{\partial \xi} \right]^T \left[1 - M\left(S_i(\xi)\right)\right] \tag{17}$$

where w_i is weight associated with point P_i, which mainly down weights the low quality scan end points with big error

FIGURE 6 Projected 2D scan end points.



and hence enhance robustness against noise [14]. λ is a damping parameter (initially set to 0.01 in this work), I is identity matrix, H is weighted approximate Hessian matrix, defined by:

$$H = w_i \cdot \left[\nabla M\left(S_i(\xi)\right)\frac{\partial S_i(\xi)}{\partial \xi}\right]^T \left[\nabla M\left(S_i(\xi)\right)\frac{\partial S_i(\xi)}{\partial \xi}\right] \quad (18)$$

By solving $\Delta\xi$, ξ is updated by:

$$\xi \leftarrow \xi + \Delta\xi \quad (19)$$

and that makes ξ iteratively move forward to the optimal transformation ξ^*.

In contrast to the practical implementation in Hector SLAM [5], where fixed iteration step setting is employed to evaluate the Gauss-Newton optimization, in addition to setting a maximum iteration step (10 in this work), we hereby propose a more reasonable stop condition before reaching the maximum iteration step, which has been proven to ensure sufficient convergence while avoiding unnecessary iterations caused by oscillation around the optimal solution:

$$\| \Delta\xi \| < \varepsilon \quad (20)$$

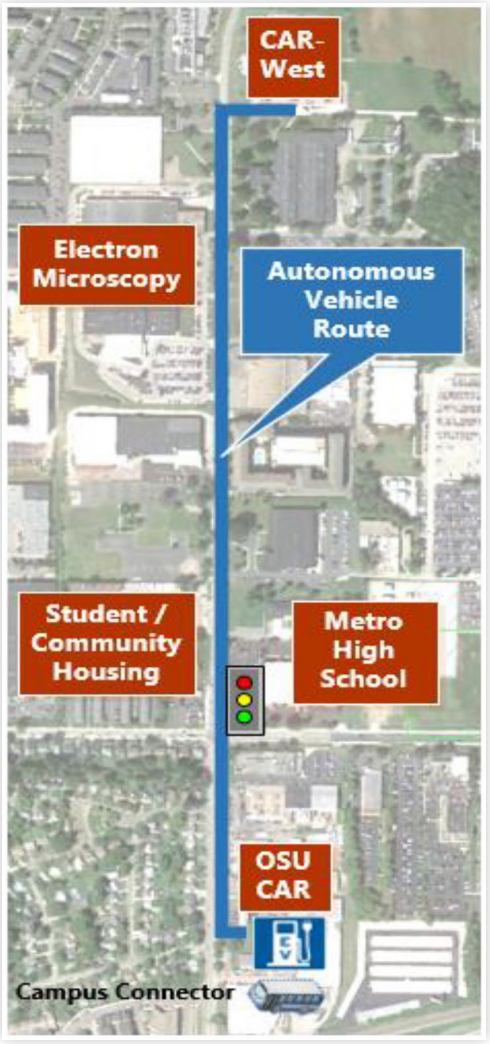
where operator $\|\bullet\|$ denotes Frobenius norm, ε is a parameter for threshold and is set to 0.001 in this work. E_k is the cost function in the *kth* iteration step.

Real World Experiments

We conducted extensive experimental validations of our system including offline SLAM system test on collected data as well as real time field experiment in the area around the initial autonomous vehicle (AV) pilot test route, a small segment in an underserved area of campus designated by The Ohio State University, as shown in Figure 7. All the algorithms relevant to LIDAR data processing and SLAM as described above are implemented in C++ because of its efficiency of real time performance. Performances are evaluated between the SLAM system proposed in [5] and the extended version proposed in this paper. Traditional path following experiment result based on high accuracy GPS similar to the previous work is compared with this innovative SLAM based path following experiment result, demonstrating the feasibility and effectiveness of this compounded system. Note that randomness is inevitably introduced by probabilistic occupancy grid map model in the SLAM system. For this reason, the experiment results are reported based on the median performance of several runs.

Real time SLAM algorithm is carried out with an I7-6700HQ (8 cores @ 2.60 GHz), NVIDIA Titan X (Pascal)/PCIe/SSE2 and 4 Gb RAM on the Robot Operating System (ROS) [15], an open source operating system providing services designed for heterogeneous computer cluster in Linux environment. User Datagram Protocol (UDP) communication is built up between ROS and MABx for localization information transfer. Regional localization information delivered by SLAM algorithm is sent to MABx for further decision making and control strategy, e.g. longitudinal or lateral control.

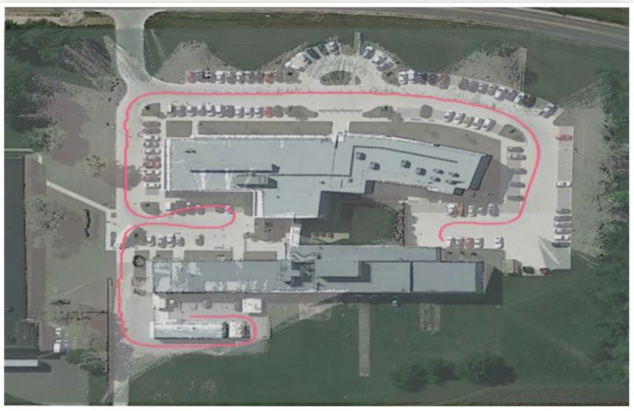
FIGURE 7 Autonomous vehicle test route from Car-West to Car (scale 1:8000).



SLAM Evaluation

In order to quantitatively evaluate our proposed SLAM system against Hector SLAM, both SLAM systems are tested on the same LIDAR data collected around our lab, Car-West. Due to the absence of "ground truth", alignment error yielded in both algorithms is reported for comparison. Ideally, with sufficient accuracy, the alignment error (described in equation (11)) should be very small. However, inevitably introduced sensor noise and non-smooth approximation of the optimization model make the solution of pose estimation only able to approach real pose but never perfectly equivalent and hence total alignment error always exists. Therefore, in the same context, the smaller the alignment error, the higher the accuracy that is achieved and hereby we evaluate the performance by comparing their alignment error and iterations implemented in each alignment, which can reflect their estimation accuracy as well as their convergence speed. Considering that offline SLAM accuracy is similar to its real time accuracy, this comparison can effectively validate the overall performance of our proposed SLAM system against the Hector SLAM.

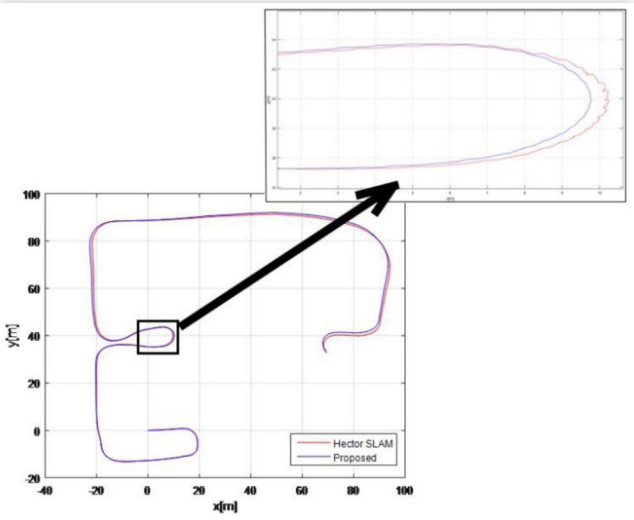
FIGURE 8 Generated map overlapped with Google Earth.



The ultimate map generated by our proposed SLAM system is overlapped with the same location obtained from Google Earth for comparison convenience as shown in Figure 8, where the map generated by our proposed SLAM is in shadow and red line is the test trajectory. It is important to note that the map from Google Earth is not strictly top down view. Thus here a minor shift is necessarily used to keep the edges of the mapped buildings consistent with their actual corresponding edges in Google Earth. In this experiment, raw LIDAR data is initially collected by VLP-16 along the test trajectory which starts from the backyard of Car-West, passing through an open field which is sufficiently challenging because of the limited landscapes for matching alignment and textureless wall. Another challenging part of this test trajectory is a sharp 180 degree turn in the front of the parking lot of the lab building, which demands fast convergence and robustness of the nonlinear optimization model implemented in the SLAM system.

Figure 9 shows both complete and regional localization estimation from the two SLAM systems along the test trajectory. The smoother localization given by our proposed SLAM system with the integrated automated drive control systems can dramatically improve passenger comfort while taking a ride in the shuttle. Table I illustrates the average alignment error and average iteration steps required between the two SLAM systems. It can be clearly observed that in some runs, our proposed SLAM can effectively reduce the alignment error to a relatively lower level despite the fact that in almost half of the runs the benefit is not distinct. Results of the average alignment error from Figure 10 can further prove this property. This can be attributed to the defect of this optimization based SLAM system where global minimum cannot be guaranteed and scan end point outliers can inevitably introduce noise to the system. Therefore, a reliable preprocessing model of the scan end points is desired as an extension to this framework, which may be an interesting topic in future work. Although in our proposed SLAM system additional iteration steps are sacrificed for better alignment compared with Hector SLAM, in which the iteration step is set to a fixed value and naturally convergence cannot be guaranteed, the increased iteration step is still in an acceptable range for real time performance according to our real-time experiments.

FIGURE 9 Trajectory comparison between our proposed SLAM (blue) with Hector SLAM (red).



Real Time Path Following Performance

In addition to quantitative evaluation of our proposed SLAM system, various real world experiments are also conducted to validate its feasibility and adaptivity of integration with the control system. We first manually drive the shuttle along the pre-determined trajectory around our lab building, as shown in Figure 11, to collect GPS points, from which the desired path is then generated for path following reference.

TABLE 1 Performance comparison between our proposed SLAM with Hector SLAM. Alignment error is accumulated error of occupancy value, which is dimensionless.

	Proposed SLAM	Hector SLAM
Average alignment error	78.759	84.107
Average iteration step	6.557	3.400

FIGURE 10 Trajectory on satellite image.

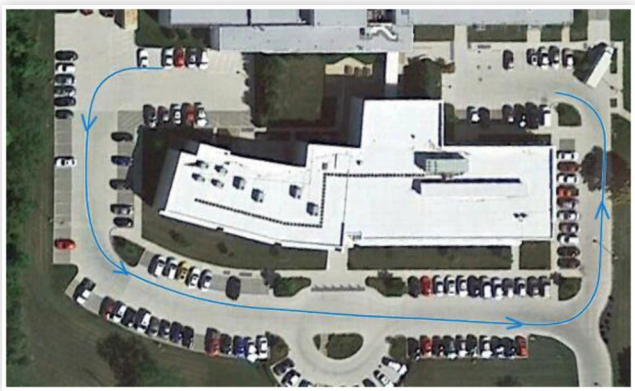


FIGURE 11 Desired path compared to our proposed SLAM path following trajectory.

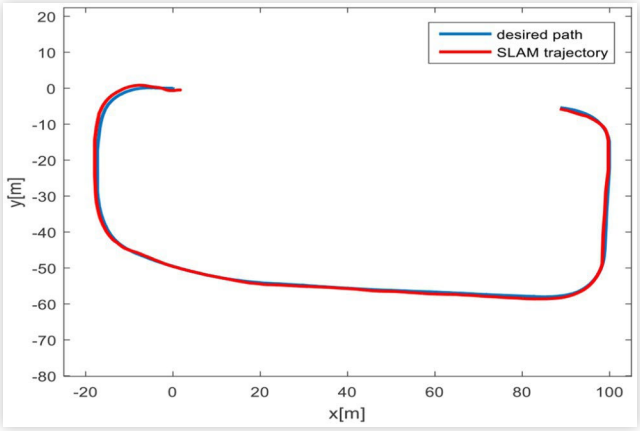
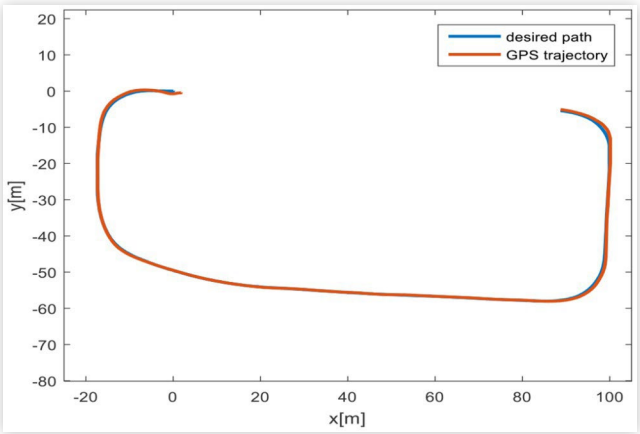
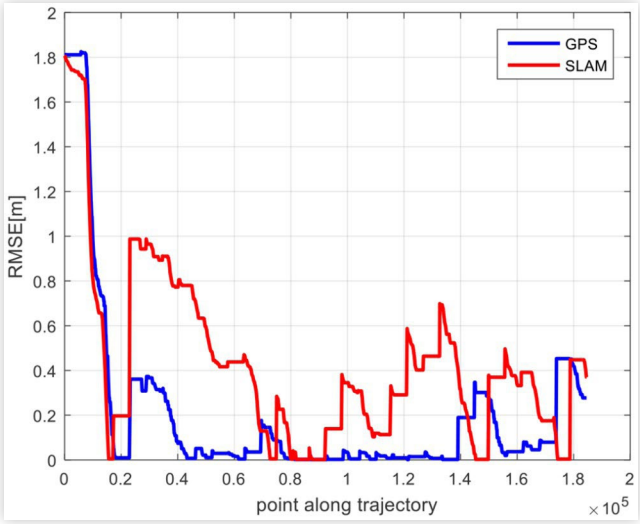


FIGURE 12 Desired path compared to GPS path following trajectory.



Figures 12 and 13 show the actual path following trajectory performed by our proposed SLAM system and RTK GPS separately compared with the desired path. The coordinate of starting position is set to the origin in the following plots for comparison convenience. It can be observed that similar to GPS, SLAM based path following can be achieved comparable to GPS based result, though with occasional minor error, which again proved the supplemental functionality of our proposed SLAM system in GPS not accessible cases. Figure 13 shows the root-mean-square error (RMSE) along the whole path following trajectory performed by SLAM compared with the same experiment setting but performed by differential GPS. The shuttle speed of both path following approaches are kept at an average value of 12 km/h. As can be seen from the experimental results, conventional path following that relies on highly accurate differential GPS has the expected performance with appropriate lateral controller design. The overall performance of GPS is better than SLAM, but SLAM based path following tends to have even smaller RMSE at some regions, e.g. at points of 0.7×10^5, 1.5×10^5, 1.8×10^5 which are at the corners of the trajectory. The fact suggests that this SLAM system can provide precise estimation of the shuttle

FIGURE 13 RMSE in lateral direction comparison between our proposed SLAM based path following and GPS based path following.



orientation while there may exist some delay or inaccuracy in the orientation angle provided by differential GPS, which is computed based on compass. It demonstrates that localization and perception system that purely relies on LIDAR can supplement the cases when GPS is not available or a lower cost and hence lower accuracy GPS is desirable for intelligent shuttles.

HiL Studies

Hardware in the Loop (HiL) setup is crucial for faster development of controllers and algorithms, since it provides a realistic virtual proving ground before the implementation and deployment phases. To create this realistic virtual proving ground, real world scenarios should be replicated with as many aspects as possible. This includes emulation of sensors, addition of traffic, addition of hardware and replication of real world routes. For this paper, the planned actual real-world AV shuttle deployment route is selected as a virtual proving ground.

Equipment and Setup

The HiL setup is constructed with hardware as close as possible to real-world case. Therefore, MABx is used as a main controller. This ECU is also the device we use in our autonomous vehicles as low-level controller, which is mentioned in the Hardware and Platform section. Since we already develop autonomous driving algorithms which runs within this device during the HiL development, it allows us to directly implement the algorithms and controllers that we developed inside the HiL simulation to a real autonomous vehicle. MABx is also connected to a DSRC modem similar to the real world case in the HiL simulator. Through this modem, it receives the V2X data that is published for the vehicles and infrastructure within the simulation. Again, similar to the real world case, it is connected to the Scalexio computer which mimics the actual vehicle through the

Controller Area Network (CAN) bus. The MABx thinks it is connected to a real vehicle while receiving the ego vehicle information from CAN bus and publishing actuation commands for steering, brake and throttle through the CAN bus.

These commands are picked up by the Scalexio real-time vehicle, traffic and sensors simulator. This simulator runs a Simulink model with CarSim vehicle dynamics. Vehicle model parameters inside CarSim are validated through vehicle dynamics experiments previously performed on the real vehicle. Therefore, vehicle dynamics simulation provides results very close to the real-world. While simulating high-fidelity vehicle dynamics for ego vehicle, it can also simulate roads, sensors, infrastructure through the capabilities of CarSim. This feature provides significant advantage since it allows us to create numerous test scenarios which have applications in real world autonomous driving. It is also connected to another DSRC modem that publishes V2X information for other vehicles and infrastructure that exist inside the simulation environment. All of the DSRC message packets are sent within a standard format obtained from SAE J2735 DSRC Message Set and using the standard communication rate of 10 Hz. Overall illustration of the HiL setup and communication between components are shown in Figure 14.

With this HiL setup, we are able to test numerous kinds of different scenarios involving other vehicles, pedestrians and road structures, which involves V2X communication. Moreover, we are able to test our controllers and autonomous driving algorithms and do improvements on them before starting road testing.

In this study, the HiL setup discussed above is used to provide a virtual proving ground for algorithm and controller development before real world deployment of the autonomous shuttle. A test scenario is created based on a planned real world deployment route, which is explained within the next section, followed by discussion of the simulation results.

Test Scenario

A replication of the real world route AV pilot test route was created inside CarSim for autonomous driving simulation. This route starts from the road in front of the parking lot of our research lab building (CAR West) and ends about 0.7 miles

FIGURE 14 HiL equipment and communication.

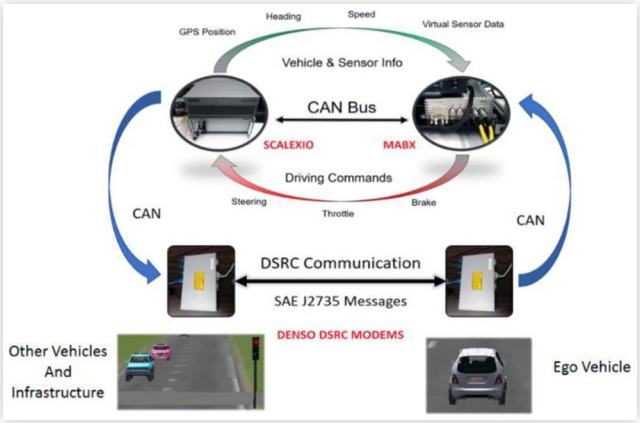
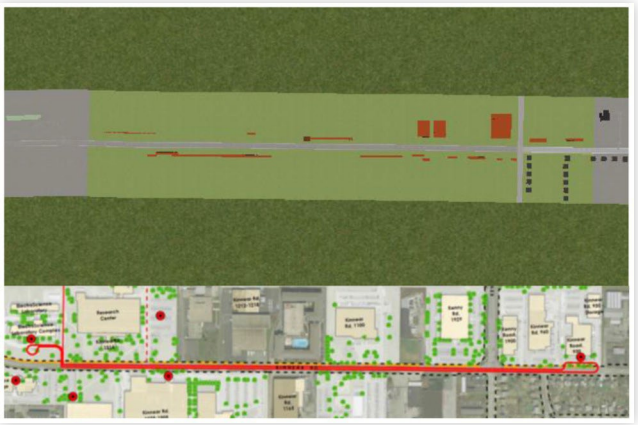


FIGURE 15 Top-down view of CAR-CAR West AV test route.



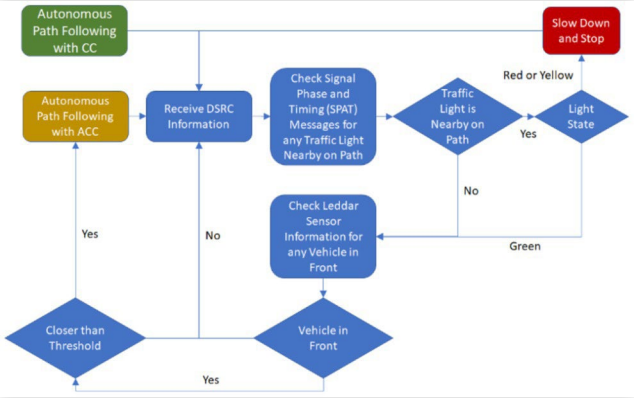
down the road in front of our main research center (CAR). A traffic light is placed on the intersection and vehicle traffic is generated within CarSim for main route. Buildings are also created as a representation of real ones and placed according to their real-world positions. A top-down view of the road which is rendered in CarSim, is shown in Figure 15.

The path to be followed is generated from the GPS points on the road and vehicle is set to autonomously drive on this path, in other words, to follow the route while making decisions according to the situations it comes across during the drive. GPS and Leddar sensors are virtually simulated in CarSim software while DSRC messages are received through real hardware. Therefore, the virtual simulation vehicle is equipped with a real DSRC radio, soft GPS and a soft Leddar sensor. In this specific scenario, DSRC radio is mainly used for determination of the traffic light state in the intersection. Leddar sensor is utilized for detection of the distance between ego vehicle and preceding vehicle. Since LIDAR emulation is currently not available as a solution within CarSim, work is still in progress to emulate or simulate LIDAR sensor which provides a 3D point cloud data to simulate LIDAR based algorithms such as SLAM in the simulator.

A. Decision Making The vehicle was commanded to follow the route while handling some of the situations it may come across. For this purpose, a simple decision-making strategy is created with three main states. This decision-making strategy is still work in progress and currently does not take all of the possible real-world cases into the account. Instead, the scenario is slightly simplified with respect to real world conditions in order to use a non-complex decision-making strategy. These simplifications include the placement of the starting and end position onto the main road and removal of the intersection cross traffic. These simplifications will be removed in further study.

The developed decision making strategy consists of three main states. In Cruise Control (CC) state, the vehicle is given a velocity profile to follow as a longitudinal control strategy. The vehicle follows the route while traveling at the desired speed which is decided by this velocity profile, according to the map segment the vehicle is currently in. With this velocity

FIGURE 16 Decision making flowchart.



profile, the vehicle can slow down or speed up when necessary, according to the road portion it is currently in, and therefore can safely approach intersections, sharp curved turns, traffic lights and obey traffic speed limits. While carrying out path following in CC state, it constantly checks for any DSRC messages. In case there is any traffic light nearby on path, according to the state of the light it can go to stop state or continue. Furthermore, by making use of the Leddar sensor information, the vehicle can determine if there is a preceding vehicle and according to the distance, it goes to Adaptive Cruise Control (ACC) state or Cooperative Adaptive Cruise Control (CACC) state in the case of a communicating preceding vehicle for car following. In this state, the vehicle keeps a safe time gap with the preceding vehicle. The flowchart for the simple decision making used is shown in Figure 16.

HiL Simulation Results

After the route is constructed in CarSim and algorithms and decision-making is implemented in Simulink, simulation testing begins. The vehicle was commanded to follow the route while handling the states as is necessary. The speed profile shown in Figure 17 is provided to the vehicle to follow while it is in CC state.

Speed is decided according to the road segment, where these segments are obtained from path generation algorithm part. After the simulation, recorded vehicle velocity, vehicle

FIGURE 17 Velocity profile with respect to segment.

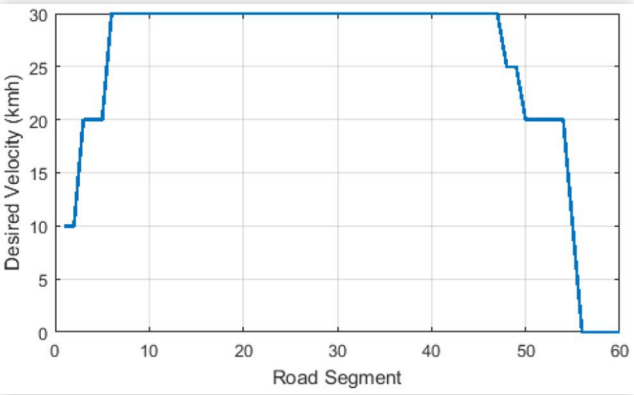
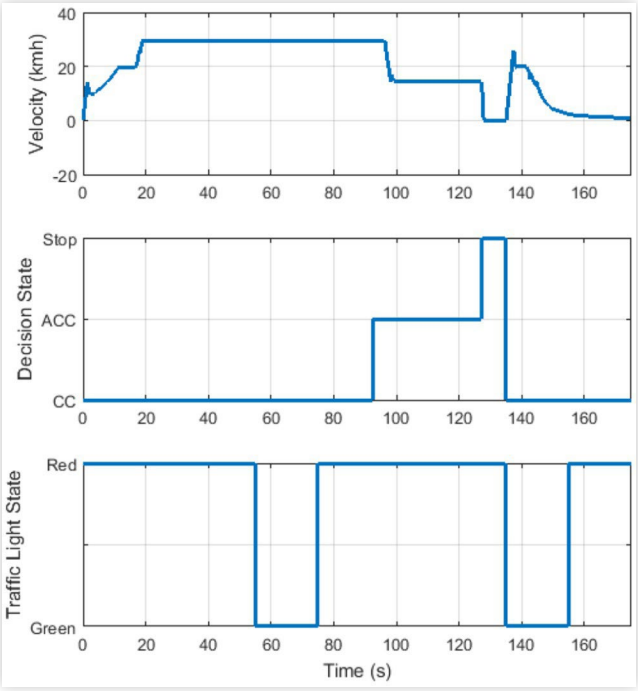


FIGURE 18 Vehicle velocity, behavior and traffic light state with respect to simulation time.



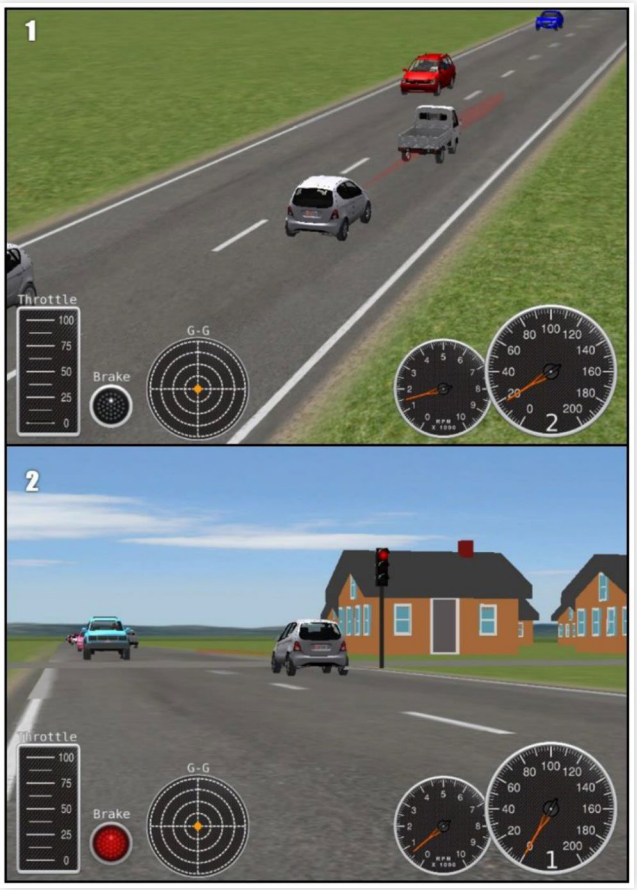
decision state (Stop/ACC/CC) and traffic light state (green/red) are plotted with respect to time as shown in Figure 18.

As seen in Figure 18, the vehicle follows the speed profile in CC mode while doing autonomous path following. After some time, it comes across a non-communicating preceding vehicle which travels at a slower velocity. Instead of following the velocity profile, autonomous vehicle goes to ACC mode and slows down to adapt to the speed and keep the distance between itself and the preceding vehicle constant. Around 125 second, it comes close to the intersection where there is a traffic light which is at red signal state. It waits until the light is green and continues its way. This behavior can also be confirmed by looking at the velocity and the state of the traffic light in Figure 18.

After passing the traffic light, it comes closer to the destination, slows down and stops. The trajectory of the vehicle is also plotted on a satellite image and shown in Figure 19. It is

FIGURE 19 Vehicle trajectory on satellite image.

FIGURE 20 Simulation frames while vehicle is doing ACC (1) and stopping at traffic light (2).



seen that the vehicle is able to follow the route and autonomously handle dynamic driving tasks it can come across while travelling through this route. Screenshots from the simulation are shown in Figure 20, while the vehicle is doing autonomous driving.

Summary/Conclusion

This paper presented preliminary work for a AV shuttle deployment in the AV pilot test route of the Ohio State University. GPS and LIDAR SLAM are both used for localization and path generation. Since GPS based localization and path following was presented in our earlier work, this paper concentrated on a LIDAR SLAM system which is inherited from the Hector SLAM framework and based on the Levenberg-Marquardt algorithm. It was demonstrated that this LIDAR SLAM algorithm can be used for self-localization of our low speed autonomous shuttle. Extensive experiments were conducted for offline SLAM performance evaluation as well as real world experiments for path following in a parking lot for safety. The proposed SLAM system was compared with the state of art 2D SLAM approach especially in terms of scan alignment accuracy and seen to provide dynamically reasonable pose estimation. As a pre-requisite to testing autonomous driving on the actual AV pilot test route, this route was replicated in our HiL simulator for developing and testing low level controllers and decision making logic. GPS and Leddar sensors, traffic and the traffic light were emulated in the HiL simulator while the low level control ECU and the DSRC radios used for V2I and V2V communication were real hardware. LIDAR sensor emulation work is in progress and will allow us to implement LIDAR based algorithms for both localization, e.g. SLAM, and obstacle detection and classification within the HiL simulator.

References

1. Leonard, John J., and Hugh F. Durrant-Whyte. "Simultaneous Map Building and Localization for an Autonomous Mobile Robot."In Intelligent Robots and Systems' 91.' Intelligence for Mechanical Systems, Proceedings IROS'91. IEEE/RSJ International Workshop on, pp. 1442-1447. Ieee, 1991.
2. Pritsker, A. Alan B. "Introduction to Stimulation and Slam II." (1986).
3. Dissanayake, M.W.M.G., Newman, P., Clark, S., Durrant-Whyte, H.F., and Csorba, M., "A Solution to the Simultaneous Localization and Map Building (SLAM) Problem," *IEEE Transactions on Robotics and Automation* 17(3):229-241, 2001.
4. Grisettiyz, Giorgio, Cyrill Stachniss, and Wolfram Burgard. "Improving Grid-based Slam with Rao-blackwellized Particle Filters by Adaptive Proposals and Selective Resampling." In Robotics and Automation, 2005. ICRA 2005. *Proceedings of the 2005 IEEE International Conference on*, pp. 2432-2437. IEEE, 2005.
5. Kohlbrecher, Stefan, Oskar Von Stryk, Johannes Meyer, and Uwe Klingauf. "A flexible and scalable slam system with full 3d motion estimation."*In Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, pp.155-160. IEEE, 2011.
6. Newman, Paul, David Cole, and Kin Ho. "Outdoor SLAM using Visual Appearance and Laser Ranging." In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pp. 1180-1187. IEEE*, 2006.
7. Cole, David M., and Paul M. Newman. "Using Laser Range Data for 3D SLAM in Outdoor Environments." In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 1556-1563. IEEE, 2006.
8. Levinson, Jesse, and Sebastian Thrun. "Robust Vehicle Localization in Urban Environments using Probabilistic Maps." In *Robotics and Automation (ICRA), 2010 IEEE International Conference on, pp. 4372-4378. IEEE*, 2010.
9. Vu, Trung-Dung, Julien Burlet, and Olivier Aycard. "Grid-based localization and online mapping with moving objects detection and tracking: new results."*In Intelligent Vehicles Symposium, 2008 IEEE*, pp. 684-689. IEEE, 2008.
10. Gelbal, S. Y., Wang, H., Chandramouli, N., Guvenc, L. et al. "A Connected and Autonomous Vehicle Hardware-in-the-loop Simulator for Developing Automated Driving Algorithms," *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2017.

11. Emirler, Mümin Tolga, Haoan Wang, B. Aksun Güvenç, and Levent Güvenç. "Automated robust path following control based on calculation of lateral deviation and yaw angle error." In *ASME Dynamic Systems and Control Conference (DSCC)*, Columbus, OH, USA, October 28, vol. 30. 2015.
12. Moré, J.J., "The Levenberg-Marquardt Algorithm: Implementation and Theory," . In: *Numerical Analysis*. (Berlin, Heidelberg, Springer, 1978), 105-116.
13. Lucas, Bruce D., and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision." (1981): 674-679.
14. Meer, P. et al., "Robust Regression Methods for Computer Vision: A Review," *International Journal of Computer Vision* 6(1):59-70, 1991.
15. Quigley, Morgan, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. "ROS: An Open-Source Robot Operating System."In ICRA Workshop on Open Source Software, vol. 3, no. 3.2, p. 5. 2009.

Contact Information

Automated Driving Lab
930 Kinnear Rd, Columbus, OH, 43212
guvenc.1@osu.edu

Acknowledgments

This paper is based upon work supported by the National Science Foundation under Grant No.:1640308 for the NIST GCTC Smart City EAGER project UNIFY titled: Unified and Scalable Architecture for Low Speed Automated Shuttle Deployment in a Smart City, by the U.S. Department of Transportation Mobility 21: National University Transportation Center for Improving Mobility (CMU) sub-project titled: SmartShuttle: Model Based Design and Evaluation of Automated On-Demand Shuttles for Solving the First-Mile and Last-Mile Problem in a Smart City and the Ohio State University Center for Automotive Research Membership Project titled: Use of OSU CAR Connected and Automated Driving Vehicle HiL Simulator for Developing Basic Highway Chauffeur and Smart City Autonomous Shuttle Algorithms. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

Definitions/Abbreviations

AV - Autonomous Vehicle
HiL - Hardware-in-the-Loop
FoV - Field Of View
RTK - Real-Time Kinematic
UDP - User Datagram Protocol
DOF - Degrees Of Freedom
IMU - Inertial Measurement Unit
SLAM - Simultaneous Localization And Mapping
LIDAR - Light Detection And Ranging
V2X - Vehicle to Everything
DSRC - Dedicated Short-Range Communication
GPS - Global Positioning Systems
ECU - Electronic Control Unit
SAE - Society of Automotive Engineers

ISSN 0148-7191