2018-01-0164

# 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

## 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. Related videos were also accessible online[1, 2].

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 the experimental and 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]. 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 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. 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.

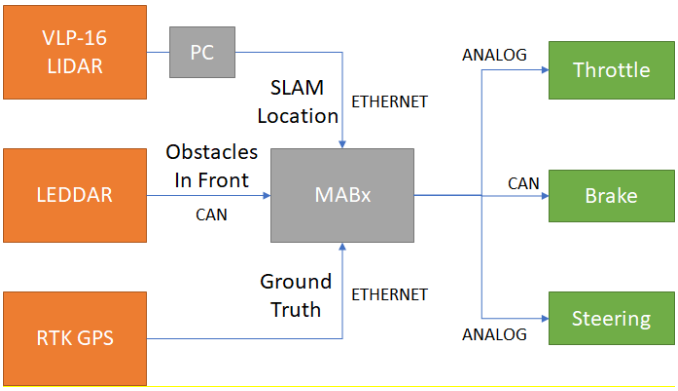


Figure 1. Platform architecture and connections.

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 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 10Hz. 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.

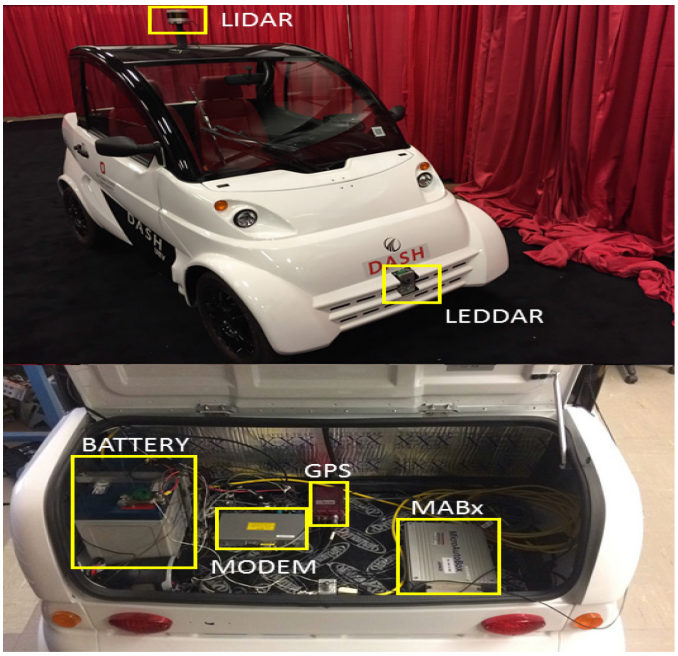


Figure 2. Hardware on the vehicle.

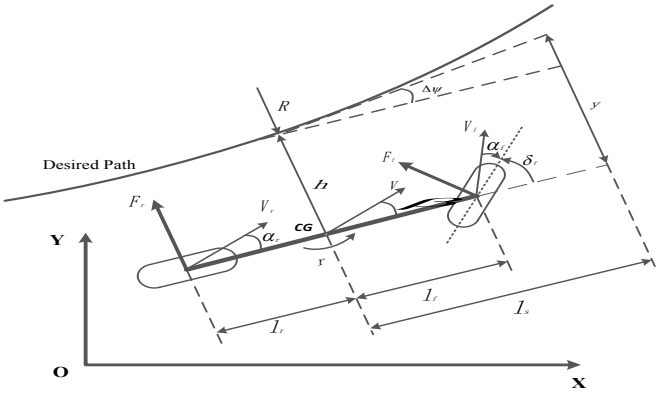


Figure 3. Illustration of single track model.

## *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

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

where $\beta$ 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 $\delta_f$. $\rho_{ref}=1/R$ is the road curvature where $R$ is the road radius. Other terms in the state space model are

$$a_{11}=-\frac{C_r+C_f}{mV}, a_{12}=-1+\frac{C_r l_r-C_f l_f}{mV^2}$$
$$a_{21}=-\frac{C_r l_r-C_f l_f}{J}, a_{22}=-\frac{C_r l_r^2-C_f l_f^2}{JV^2}$$
$$b_{11}=\frac{C_f}{mV}, b_{12}=\frac{C_f l_f}{J} \tag{2}$$

where $m$ is the vehicle mass, $J$ is the moment of inertia, $\mu$ 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

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

(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

$$X_i(1) = X_{i+1}(0)$$
$$Y_i(1) = Y_{i+1}(0)$$

(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 $\lambda$, where $\lambda$ is the variable for the fit which varies across each segment between 0 to 1, resulting in

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

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

(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,

$$y = h + l_s \sin(\Delta\psi)$$

(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 4.

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

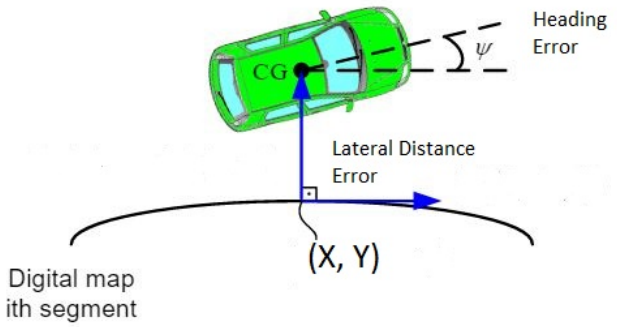


Figure 4. Illustration of error calculation.

## *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 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 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).

*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 0.2m x 0.2m. 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 0.3m in this work), if

$$z_{\max,j} - z_{\min,j} \le h_{thres}$$

(7)

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

$$z_{\max,j} - z_{\min,j} \geq h_{thres}$$

(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:

$$\alpha_i = atan2(y_i, x_i)$$

(9)

where $atan2$ 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:

$$range_i = \sqrt{x_i^2 + y_i^2}$$

(10)

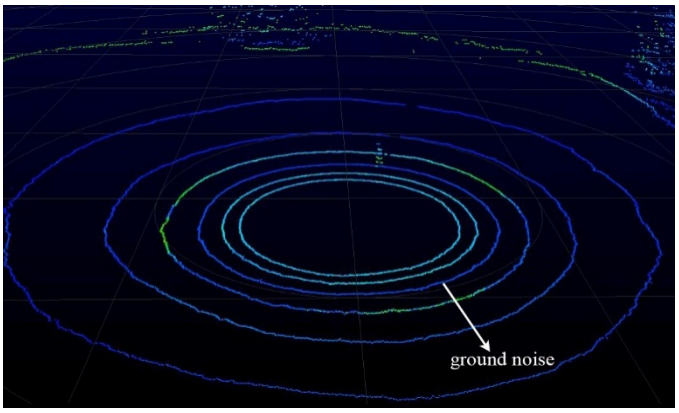


Figure 5. Raw 3D point cloud with ground noise

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.

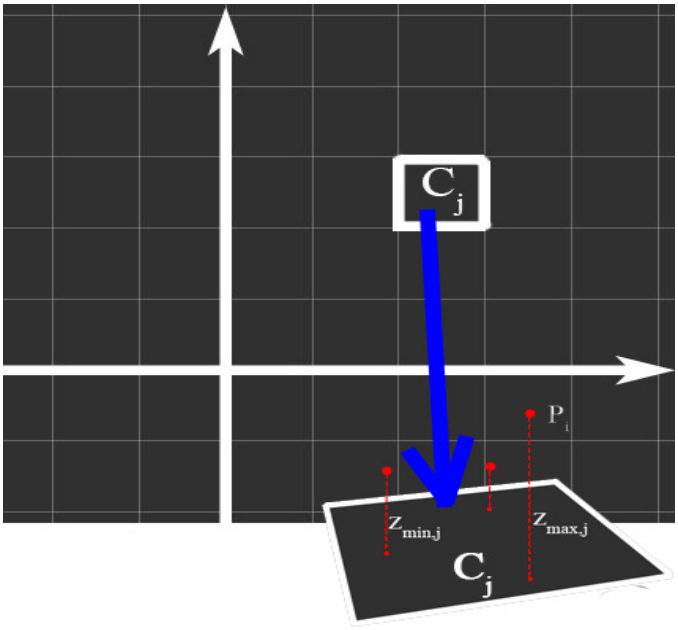


Figure 6. Occupancy height map. $C_j$ is one of the cells. Height of every cell is determined by the maximum height difference in that cell

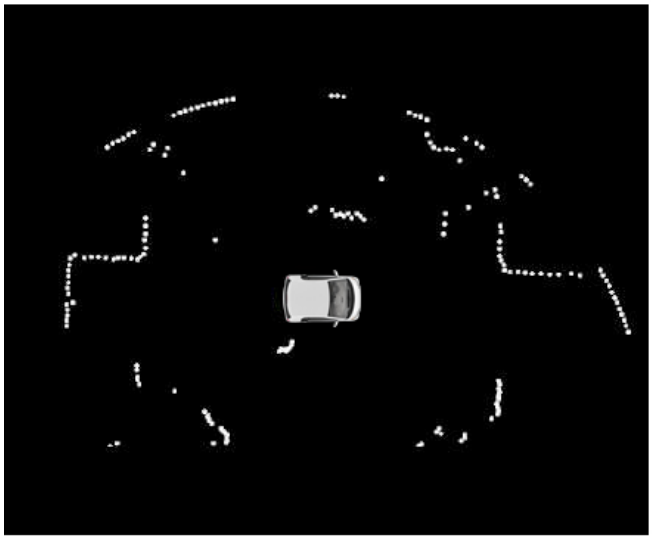

Figure 7. Projected 2D scan end points

*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:

$$E = \min \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2 \tag{11}$$

$$\xi^* = \arg\min_{\xi^*} \sum_{i=1}^{n} [1 - M(S_i(\xi))]^2 \tag{12}$$

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 $\xi$ 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, $\xi_0$, in every iteration, a transformation update $\Delta\xi$ is added to the accumulated transformation so far, $\xi$, 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:

$$E = \sum_{i=1}^{n} [1 - M(S_i(\xi + \Delta\xi))]^2 \to 0 \tag{14}$$

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:

$$\frac{\partial E}{\partial(\Delta\xi)} = 2\sum_{i=1}^{n} [\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}]^T \cdot \ldots$$

$$\cdot \sum_{i=1}^{n} [1 - M(S_i(\xi)) - \nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi} \Delta\xi] = 0 \tag{16}$$

according to Levenberg–Marquardt algorithm, the optimal solution for $\Delta\xi$ can be determined by:

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

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

$$H = w_i \cdot [\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}]^T [\nabla M(S_i(\xi)) \frac{\partial S_i(\xi)}{\partial \xi}] \tag{18}$$

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

$$\xi \leftarrow \xi + \Delta\xi \tag{19}$$

and that makes $\xi$ iteratively move forward to the optimal transformation $\xi^*$.

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:

$$\left\| \Delta\xi \right\| < \varepsilon \tag{20}$$

where operator $\|\cdot\|$ denotes Frobenius norm, $\varepsilon$ is a parameter for threshold and is set to 0.001 in this work. $E_k$ is the cost function in the $k^{th}$ 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 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 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.

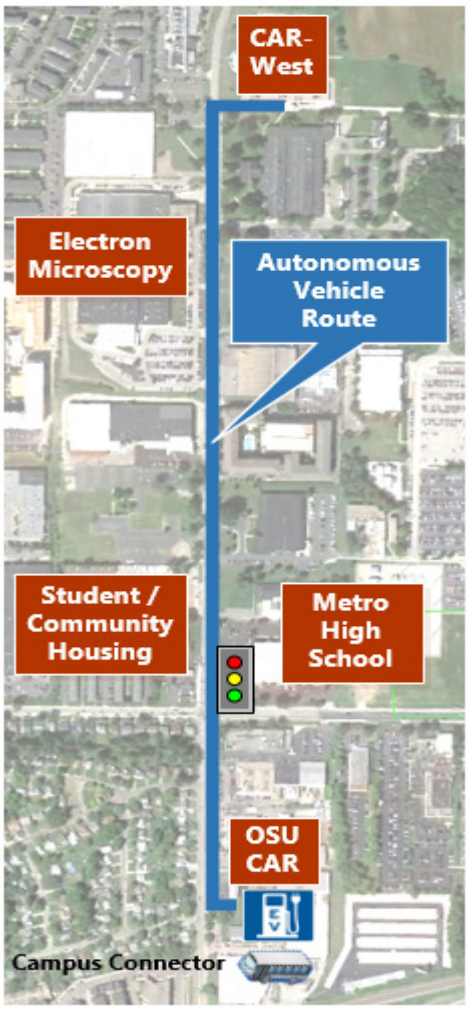


Figure 8. Autonomous vehicle test route from Car-West to Car (scale 1:8000)

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. 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.

### *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 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 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 2 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 2 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.

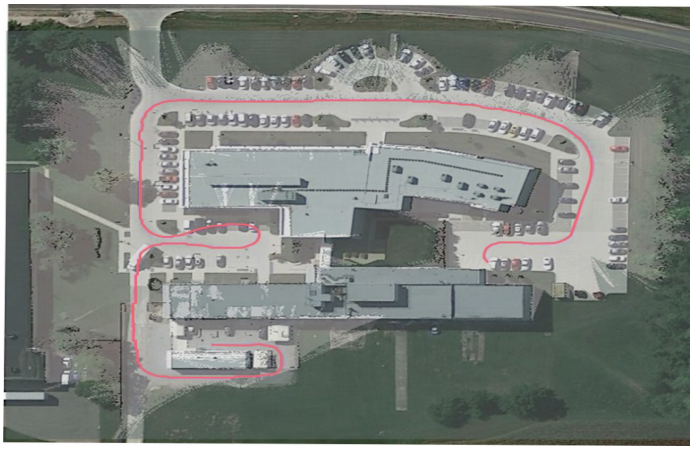

Figure 9. Generated map overlapped with Google Earth

Table 2. 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 |

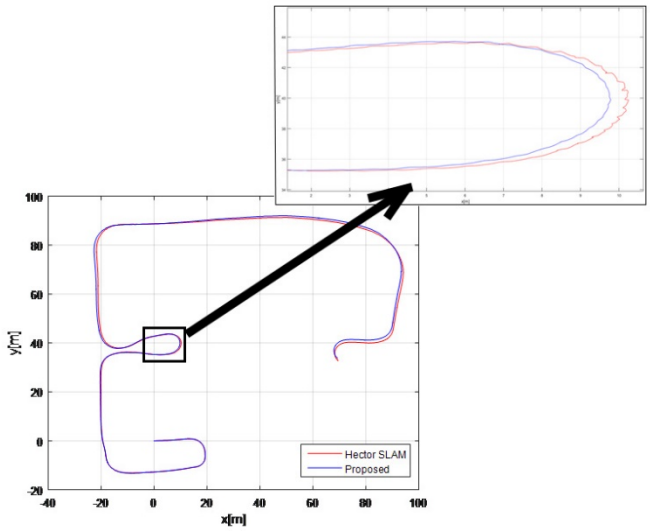


Figure 10. Trajectory comparison between our proposed SLAM (blue) with Hector SLAM (red)

### *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.

Figure 12 and Figure 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 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 $0.7 \times 10^5, 1.5 \times 10^5, 1.8 \times 10^5$ 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.

Figure 11. Trajectory on satellite image.

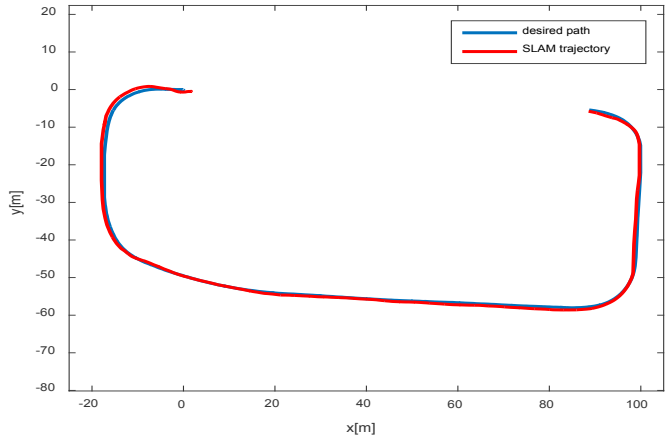


Figure 12. Desired path compared to our proposed SLAM path following trajectory.

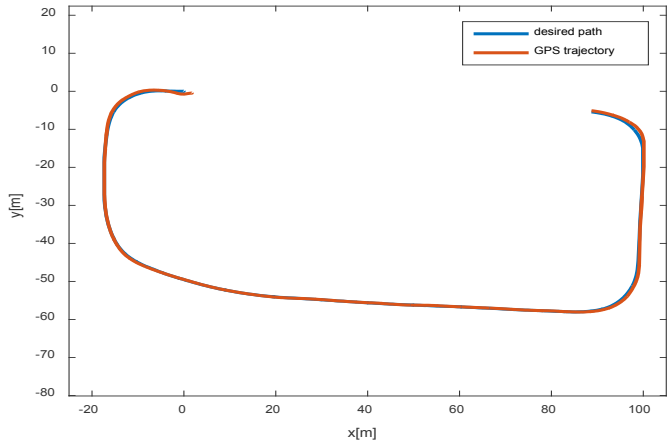


Figure 13. Desired path compared to GPS path following trajectory

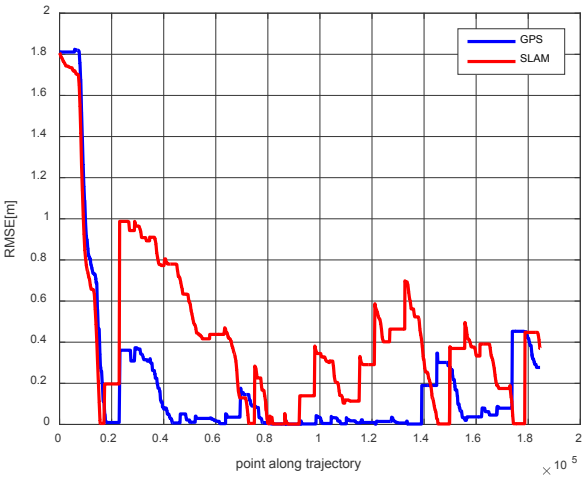


Figure 14. RMSE in lateral direction comparison between our proposed SLAM based path following and GPS based path following.

## 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 15.

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.

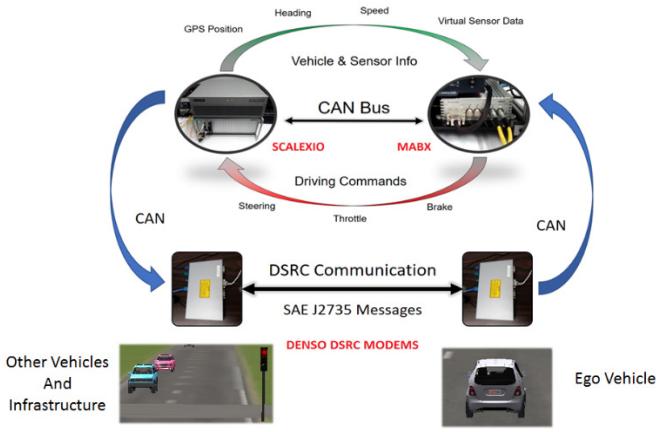


Figure 15. HiL equipment and communication.

## *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.

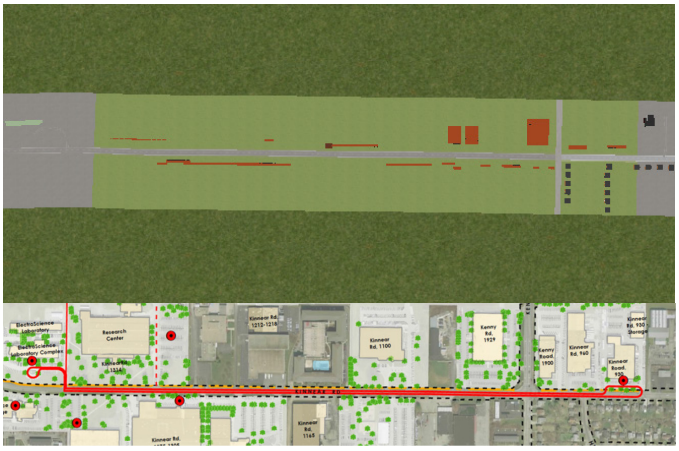

Figure 16. Top-down view of CAR West - CAR AV test route.

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 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 17.

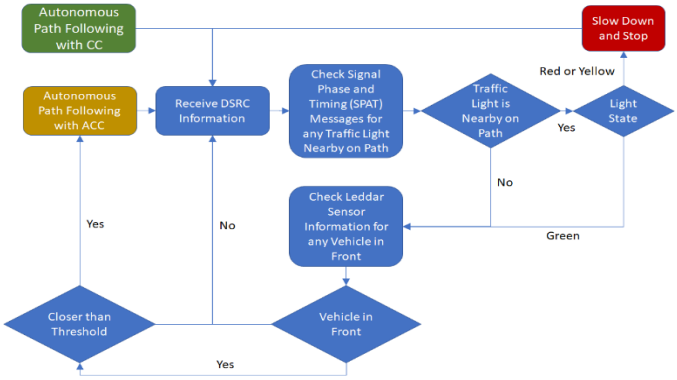


Figure 17. Decision making flowchart.

## *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.

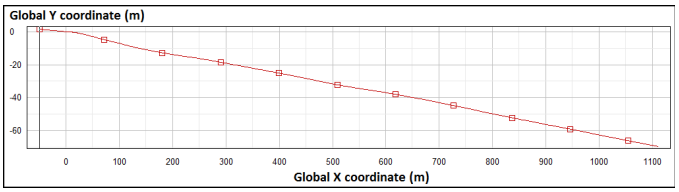


Figure 18. CAR West to CAR route constructed in CarSim.

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 has higher speed and curves, 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, vehicle goes to ACC mode to adapt the speed of preceding vehicle and keep the distance, disregarding the velocity profile.

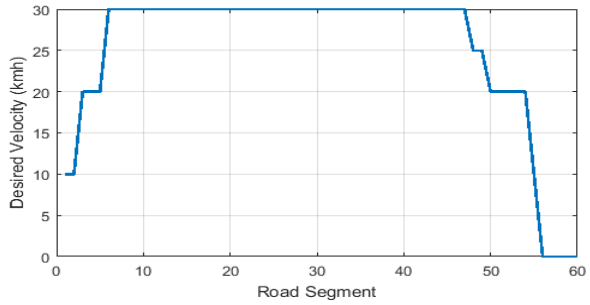


Figure 19. Velocity profile with respect to segment.

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) was plotted with respect to time as shown in Figure 20.

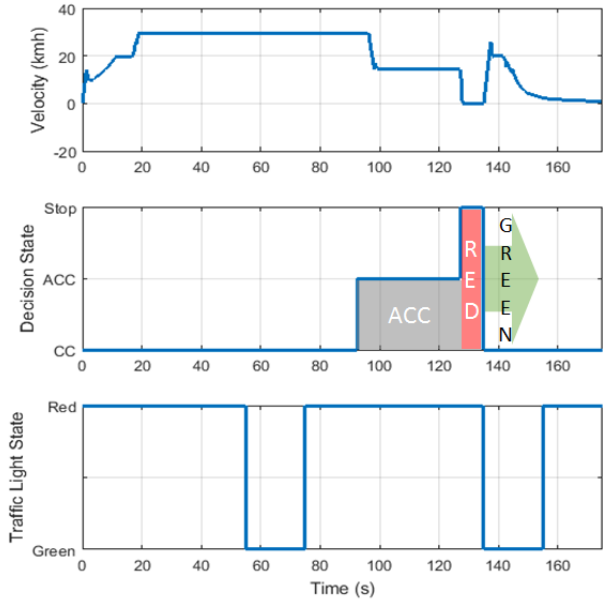


Figure 20. Vehicle velocity, behavior and traffic light state with respect to simulation time.

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, 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. This behavior can be confirmed by comparing the velocity profile (30km/h) with vehicle velocity at that time phase (around 15km/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. This behavior can also be confirmed by looking at the velocity graph, 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 travelling through this route. Some frames from the simulation are shown in Figure 22, while the vehicle is doing autonomous driving.

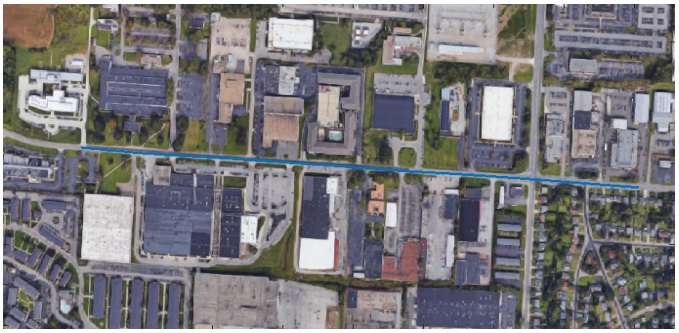

Figure 21. Vehicle trajectory on satellite image.

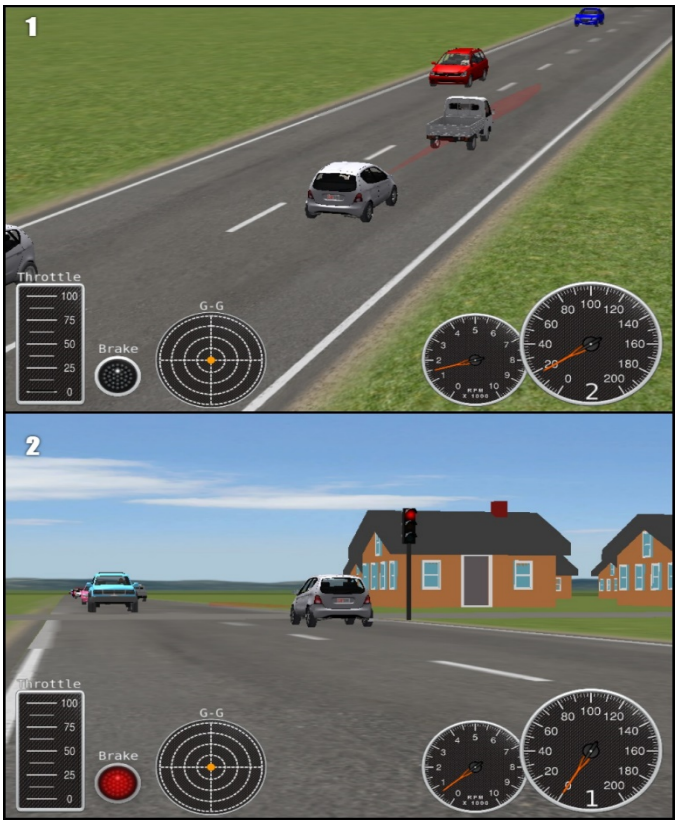


Figure 22. Simulation frames while vehicle is doing ACC (1) and stopping at traffic light (2).

## 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, MWM Gamini, Paul Newman, Steve Clark, Hugh F. Durrant-Whyte, and Michael Csorba. "A solution to the simultaneous localization and map building (SLAM) problem." IEEE Transactions on robotics and automation 17, no. 3 (2001): 229-241.
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, Jorge J. "The Levenberg-Marquardt algorithm: implementation and theory." In Numerical analysis, pp. 105-116. Springer, Berlin, Heidelberg, 1978.
13. Lucas, Bruce D., and Takeo Kanade. "An iterative image registration technique with an application to stereo vision." (1981): 674-679.
14. Meer, Peter, et al. "Robust regression methods for computer vision: A review." International journal of computer vision 6.1 (1991): 59-70.
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.

## 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 |
| **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 |