

# UC San Diego

## UC San Diego Electronic Theses and Dissertations

### Title

Autonomous Vehicles: Their Capabilities and Limitations

### Permalink

<https://escholarship.org/uc/item/60n5n5sc>

### Author

Paz Ruiz, David Fernando

### Publication Date

2020

### Supplemental Material

<https://escholarship.org/uc/item/60n5n5sc#supplemental>

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA SAN DIEGO

**Autonomous Vehicles: Their Capabilities and Limitations**

A thesis submitted in partial satisfaction of the  
requirements for the degree  
Master of Science

in

Electrical Engineering (Intelligent Systems, Robotics and Control)

by

David Paz Ruiz

Committee in charge:

Professor Henrik I Christensen, Chair  
Professor Todd Hylton, Co-Chair  
Professor Nikolay Atanasov

2020



Copyright  
David Paz Ruiz, 2020  
All rights reserved.

The thesis of David Paz Ruiz is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

---

---

Co-Chair

---

Chair

University of California San Diego

2020

## DEDICATION

To my parents, Catalina Ruiz and Bertin Paz.

## TABLE OF CONTENTS

Signature Page . . . . .	iii
Dedication . . . . .	iv
Table of Contents . . . . .	v
List of Figures . . . . .	vii
List of Tables . . . . .	viii
Acknowledgements . . . . .	ix
Vita . . . . .	xi
Abstract of the Thesis . . . . .	xii
Chapter 1	
Introduction . . . . .	1
1.1 Related Work . . . . .	3
1.2 Thesis Overview . . . . .	4
Chapter 2	
System Architecture and Vehicles . . . . .	6
2.1 Sensor Configurations and Drive-by-Wire System . . . . .	6
2.2 Module Abstractions and Containerization . . . . .	8
Chapter 3	
Core Perception, Planning and Controls . . . . .	10
3.1 Open Source Modules . . . . .	10
3.2 Localization, Mapping and HD Maps . . . . .	12
3.3 Perception . . . . .	14
3.3.1 LiDAR Detection . . . . .	14
3.3.2 Multi-Object Detection using Cameras . . . . .	16
3.3.3 LiDAR/Camera Fusion . . . . .	20
3.4 Planning . . . . .	23
3.4.1 Global and Motion Planning . . . . .	24
3.4.2 Cruising and Speed Keeping . . . . .	25
3.4.3 Vehicle Following and Planned Stops . . . . .	27
3.5 Path Tracking and Controls . . . . .	30
3.5.1 Vehicle Geometry for Steering Control . . . . .	30
3.5.2 Acceleration and Braking Controls . . . . .	32

Chapter 4	Benchmarking and Performance Evaluation for Autonomous Vehicles . .	35
	4.1 Safety . . . . .	36
	4.2 UMAV: Unbiased Metrics for Autonomous Vehicle Benchmarking .	37
	4.3 Vehicle Performance for Mail Delivery . . . . .	40
	4.4 Autonomous vs Manual Driving . . . . .	41
Chapter 5	Achieving Full Autonomy - L5 . . . . .	44
	5.1 Overview . . . . .	44
	5.2 Intent for Unstructured Environments . . . . .	46
	5.3 Moving Away from High Definition maps . . . . .	49
	5.4 Future Work . . . . .	53
Bibliography	. . . . .	56

## LIST OF FIGURES

Figure 2.1:	Sensor and System Configuration onboard GEM e6 . . . . .	7
Figure 2.2:	Docker Containers and Services running onboard autonomous vehicle platforms	9
Figure 3.1:	Extended Autoware software stack comprised of localization, perception, global planner, motion planner and control modules. . . . .	11
Figure 3.2:	3D Point Cloud Maps (top-down view) from UC San Diego. . . . .	13
Figure 3.3:	Gilman/Voigt Intersection road network at UC San Diego. Trajectories shown in gray define lanes while the trajectory shown in blue corresponds to a planned trajectory. . . . .	14
Figure 3.4:	Three hyper planes are defined for ground removal. . . . .	15
Figure 3.5:	Blue LiDAR scans correspond to points removed while red scans are considered as potential obstacles. . . . .	16
Figure 3.6:	Realtime multi-object detection using YOLOv3 and the COCO dataset. . .	17
Figure 3.7:	Image frame correspondences for front two cameras (a and b) and side cameras (c and d). . . . .	19
Figure 3.8:	Projection from image frame (left) to ground plane (right) using Homographies.	19
Figure 3.9:	LiDAR Cluster Classification: Bounding boxes on the left (a and c) are used to provide labels for the LiDAR clusters shown on the right (b and d). . . .	22
Figure 3.10:	Planning FSA . . . . .	25
Figure 3.11:	Estimating distance to obstacles . . . . .	27
Figure 3.12:	Planning for Stops . . . . .	29
Figure 3.13:	Bicycle Model Approximation . . . . .	31
Figure 3.14:	PID Controller . . . . .	32
Figure 3.15:	Target Speed (m/s) shown in red and Current Vehicle Speed (m/s) shown in blue as functions of time(s) . . . . .	34
Figure 4.1:	Emergency Stop on AV dashboard . . . . .	36
Figure 4.2:	Enable signal as a function of time. . . . .	38
Figure 5.1:	Mail delivery route intervention map. . . . .	45
Figure 5.2:	Scenes of Warren college mail delivery route. . . . .	47
Figure 5.3:	Active traffic in Gilman Dr. and Voigt Dr. Intersection . . . . .	49
Figure 5.4:	Active construction across Warren college mailing center. . . . .	50
Figure 5.5:	Point Cloud Representation of construction fence and route from Figure 5.2a.	51
Figure 5.6:	Global Planner and Dynamic Motion Planner. . . . .	55

## LIST OF TABLES

Table 4.1:	MDBI and MTBI Summary Interventions for summer and fall quarters. . .	40
Table 4.2:	MDBI and MTBI Overall Summary Interventions. . . . .	41
Table 4.3:	UMAV Metrics . . . . .	42

## ACKNOWLEDGEMENTS

I would like to express my most sincere appreciation to my research advisors, Dr. Henrik I. Christensen and Dr. Todd Hylton, for providing me with indispensable support and resources throughout my undergraduate and graduate education. With their constant support, I have learned how to contribute and develop collaborations with the research and tech communities. I am grateful for all of the meaningful discussions and constructive feedback that I have received over the past two years.

I would also like to extend my deepest gratitude to UCSD facilities, operations, mailing center, police station and risk management offices for collaborating with us as part of the active campus-wide autonomous vehicle project.

Special thanks to all of my lab mates at the Cognitive Robotics Lab and the Autonomous Vehicle Laboratory for allowing me to collaborate with them on multiple projects: Po-Jung Lai, Hengyuan Zhang, Nathan Chan, Dominique Meyer, Sumukha Harish, Francis Joseph, Shawn Winston, Yuqing Jiang, Emily Le, Carlos Nieto, Shengye Wang, Ruffin White, Ploy Temiyasathit, and Shixin Li.

Thank you.

The following describes author and co-author contributions as well material being prepared for publication.

Chapter 3 includes material as it appears in Lessons Learned From Deploying Autonomous Vehicles at UC San Diego in Field and Service Robotics, Tokyo, JP, August 2019. David Paz, Po-Jung Lai, Sumukha Harish, Hengyuan Zhang, Nathan Chan, Chun Hu, Sumit-Binnani, and Henrik Christensen. The thesis author was the primary author of this paper.

The methods introduced on LiDAR-based detection (section 3.3.1) and LiDAR/Camera fusion (section 3.3.3) were the result of a collaborative effort between Hengyuan Zhang, Sumukha Harish and the primary author. With Zhang's work on multi-plane fitting methods and Harish's early work on Camera/LiDAR fusion, these modules were incorporated into the Autoware



software stack and deployed using the two autonomous vehicle platforms introduced in this study.

Chapter 4 includes material as it appears in Lessons Learned From Deploying Autonomous Vehicles at UC San Diego in Field and Service Robotics, Tokyo, JP, August 2019. David Paz, Po-Jung Lai, Sumukha Harish, Hengyuan Zhang, Nathan Chan, Chun Hu, Sumit-Binnani, and Henrik Christensen. The thesis author was the primary author of this paper.

The Unbiased Metrics for Autonomous Vehicle Benchmarking (UMAV) introduced in section 4.2 are the result of a year-long study in which Po-Jung Lai, the primary author and principal investigator Henrik Christensen participated in. Po-Jung Lai and the primary author developed the logging devices and performed system evaluation on the two autonomous vehicle platforms introduced in this study. Lai participated as the project lead for the UCSD/TuSimple study discussed in section 4.4.

In addition, chapter 4 contains pre-publication material and statistics obtained during the UCSD mail delivery project being prepared for review.

## VITA

2015	A.A.S, Physics, San Diego Mesa College
2016	Undergraduate Research Assistant, i-Trek, Massachusetts Institute of Technology
2017	Undergraduate Research Assistant, San Diego Supercomputer Center
2017	Undergraduate Research Assistant, Computation Structures Group, Massachusetts Institute of Technology
2018	Bachelor of Science in Computer Engineering, University of California, San Diego
2018-Present	Research Assistant/Project Lead, Autonomous Vehicle Laboratory, University of California, San Diego
2020 (Expected)	Master of Science in Electrical Engineering (Intelligent Systems, Robotics and Control)

## PUBLICATIONS

**David Paz**, Po-Jung Lai, Sumukha Harish, Hengyuan Zhang, Nathan Chan, Chun Hu, SumitBinnani, and Henrik Christensen. Lessons learned from deploying autonomous vehicles at UC San Diego. In *Field and Service Robotics*, Tokyo, JP, August 2019

Emily Le and **David Paz**. Performance analysis of applications using singularity container on sdsccomet. In *Proceedings of the Practice and Experience in Advanced Research Computing 2017 on Sustainability, Success and Impact*, PEARC17, pages 66:1–66:4, New York, NY, USA, 2017. ACM

ABSTRACT OF THE THESIS

**Autonomous Vehicles: Their Capabilities and Limitations**

by

David Paz Ruiz

Master of Science in Electrical Engineering (Intelligent Systems, Robotics and Control)

University of California San Diego, 2020

Professor Henrik I Christensen, Chair  
Professor Todd Hylton, Co-Chair

Despite the latest breakthroughs and technological advancements in state of the art autonomous vehicle systems, many of these systems still experience challenges in a variety of realistic scenarios and environments that prevent them from generalizing to new situations. In this study, two development platforms are designed, tested, deployed and benchmarked to analyze the implications of micro-transit applications and the autonomy needed for fully autonomous systems.

# Chapter 1

## Introduction

From advanced driver-assistance systems to level-4 and level-5 autonomous systems\*, a significant number of industry leaders and institutions have undertaken the task to attempt to make autonomous vehicles a reality. Given the diverse number of constraints experienced from day to day driving, an ideal autonomous vehicle system must be capable of navigating through dynamic environments including intersections, crowded city streets, highway driving and unstructured environments in a similar fashion as a human drivers.

Although many of these system requirements may seem a reality today with today's technology, these requirements are simply basis of what is required and do not fully reflect long term implications on autonomy, generalizability, system robustness, and scalability. In other words, performing the same route plan and iterations under a well controlled environment, may not generalize well under different road conditions and unseen environments. This introduces an entire field of research in autonomous driving system evaluation and benchmarking.

In the state of California alone, the Department of Motor Vehicles (DMV) requires autonomous vehicle companies with a valid testing permit to submit annual reports with a summary of system disengagements †. While a publicly available summary of these disengagement reports

---

\*<https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety>

†<https://www.dmv.ca.gov/portal/dmv/detail/vr/autonomous/auto>

provide an understanding on the number of annual interventions each self-driving car company is generating, temporal and spatial information is not included. Without the time elapsed during trips, the distance driven in autonomous mode, testing location, and the number of vehicles involved in testing, the disengagement reports alone do not fully encompass system performance and robustness. As a result, data normalization is required to characterize autonomous vehicle system performance in order to be compared with human driver performance and analyze safety statistics as a whole.

With these notions in mind, this study aims to shed light on autonomous system technology performance, safety and short-comings through the implementation, testing, deployment, and evaluation of autonomous vehicle systems. As part of a collaborative effort with UC San Diego's Autonomous Vehicle Laboratory (AVL), Mailing Center, Fleet Services, and Police Department, two vehicles were retrofitted with complete drive-by-wire systems and full sensor suites to conduct field tests at the UC San Diego campus. Throughout this collaboration, I worked at the AVL with multiple UCSD students to develop new perception, planning and control models as well benchmarking tools to extend the functionality of an open-source software stack and evaluate its performance as discussed in Chapter 3 and 4. This work has been made possible by the support from the advisors, students, and staff aforementioned in the acknowledgement section.

The vehicles used in this study operated in diverse environments to collect data while serving practical purposes. During summer 2019, the autonomous vehicle platforms were retrofitted to serve as mail delivery vehicles as part of a partnership between the Mailing Center at UC San Diego and the AVL. Using a complete set of unbiased performance metrics, the data collected was analyzed with the purpose of understanding the challenges that autonomous vehicle systems experience today. A number of conclusions are formulated from the results obtained with an emphasis on future work and solutions needed to address system scalability, robustness and generalizability.

## 1.1 Related Work

While autonomous vehicle technology has most recently received a vast amount of popularity, the core concepts, fundamentals and pioneering work were conceived in the early 1980s by Ernst Dickmanns. Dickmanns' fundamentals are built on the notion of using vision as a primary sensing modality without the computational benefits from high performance CPUs or GPUs available today. With these computational constraints, adaptive computer vision methods were applied with implications on dynamic scene understanding, obstacle detection, tracking, navigation and control [Dic07]. Dickmanns has deployed multiple autonomous systems over the past three decades with highway and intersection navigation capabilities.

Clearly, autonomous vehicle systems have been under development over the course of 30 years [Pom95] [JPKA95] [Lei86] [TMD<sup>+</sup>06]. With the rapid computational power evolution since the 1980s, today's performance benefits have facilitated the development of complex algorithms that would have been impossible without dedicated hardware in past decades such as the latest deep learning and statistical based methods for perception, localization and planning.

Along these computational developments, better performing camera sensors have been introduced with capabilities of achieving over 120dB in high dynamic range (HDR). With a wide range of lighting conditions experienced in field robotics, these camera sensors are addressing the short-comings experienced under low-light conditions and high contrast outdoor environments. While cameras have drastically improved over the years, different sensing modalities have also been introduced that work remarkably well.

One type of such devices is based on electromagnetic pulses that measure the time elapsed between pulse returns to estimate distances with a high degree of accuracy; these are widely known as LiDARs—Light Detection and Ranging. These sensors can generate very detailed 3D representations of space with varying resolution that can be used to accurately measure distances to objects or even build dense point cloud maps often used for vehicle localization.

By pairing these high resolution sensors with the latest developments on deep learning, state of the art perception methods have been proposed in recent years for object 3D pose estimation using computer vision [MAFK16] [MSM<sup>+</sup>19], LiDAR [QSMG16] [HKLS12] or fusion [XAJ17]. Additional areas that benefit from these sensor technology breakthroughs and developments include Simultaneous Localization and Mapping (SLAM). Although SLAM covers a broad set of methodologies, dense point cloud mapping and localization using LiDAR technology have become essential for many industry leaders in the areas of autonomous driving due to the accuracy and precision provided by these methods while remaining invariant to lighting conditions. Despite the fact that exceptional performance and reliability can be achieved from dense point clouds, they introduce an excessive overhead and generate scalability limitations.

To measure the advantages and shortcomings of today's available technology including core LiDAR technology, this study covers the development and testing of multiple vehicles while addressing the last-mile problem for micro-transit.

## **1.2 Thesis Overview**

This thesis begins by covering different design considerations, algorithm implementation details, testing, and safety. Once the technical implementations are covered, attention is shifted towards benchmarking and comparing human drivers to autonomous vehicle technology while performing mail delivery. With the key takeaways and lessons learned from deploying mail delivery carts at UC San Diego, a number of considerations are made for achieving full autonomy and the active problems in the field.

System architecture, sensors configurations, as well as the module abstractions used on the development platforms are discussed in Chapter 2 to provide a foundation on the ecosystem that served as a basis for development. This chapter incorporates some recent software engineering practices for encapsulation and containerization; the autonomous vehicles used in this study

are among the first systems to apply these concepts for robotics applications [WC17] [WC18] [PLH<sup>+</sup>19]

In Chapter 3, algorithm design and implementations for the perception, planning, and control systems are introduced. During the initial experiments at UC San Diego, the perception stack was entirely LiDAR based. Additionally, key takeaways from live-testing and using open source modules are reported.

As previously introduced, reported disengagement data requires spatial and temporal normalization for it to be interpreted as an unbiased statistic. The material covered in Chapter 4 focuses on proposing unbiased metrics for benchmarking autonomous vehicle performance using intervention maps, Mean Distance Between Interventions (MDBI) and Mean Time Between Interventions (MTBI). Additional material in regards to safety and vehicle testing are included in this chapter.

With the results discussed in Chapter 4, conclusions are formulated in Chapter 5 to make note of the limitations experienced with today's technology. While scalability is one of the major factors that influences this technology, additional research directions are proposed to address the generalized problem for self driving cars.

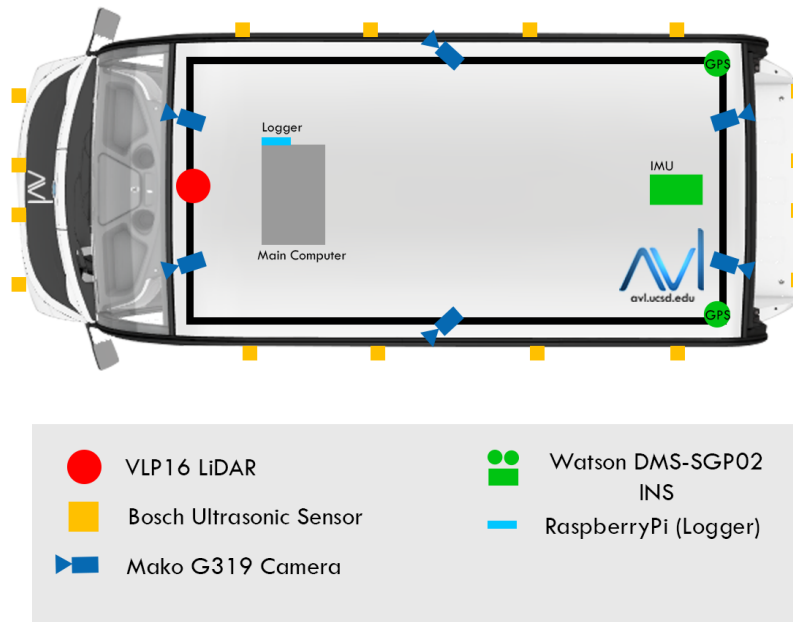


# Chapter 2

## System Architecture and Vehicles

### 2.1 Sensor Configurations and Drive-by-Wire System

Two development vehicles have been retrofitted by ranging and vision sensors for perception applications along with a complete drive-by-wire (DBW) system to control steering, acceleration and braking. The locations of the sensors and cameras are shown in Figure 2.1: 16 ultrasonic sensors, six cameras, one LiDAR and one complete Inertial Navigation Unit. Additional systems on board include the main computer with 32GB RAM, one 1TB solid-state drive, and a high-end graphics card. The logger device shown on board has been actively used to record vehicle signals such as vehicle speed reports, location, target velocities, and vehicle control inputs over time. A combination of these signals are being actively used characterize system performance; additional details are discussed in Chapter 4.



**Figure 2.1:** Sensor and System Configuration onboard GEM e6

The actuation of steering and brake controls is controlled using two servo motors capable of controlling angular position and velocity. Given the steering wheel motor mounting position, the state of the steering wheel can be tracked over time in manual and autonomous mode. This provides additional data that can be used for comparing human steering behavior against the control generated by the automated system. The brake system on the other hand has been retrofitted with a pulley system to actuate the brake pedal in order to engage the front and rear brakes. Given this design implementation, the steel-braided wire over pulley system can not provide a tension when manual input is being applied, and as a result, measuring human braking behavior is not as trivial as steering. Since the GEM e6 vehicles used are fully electric, control input for the accelerator is entirely drive-by-wire and does not require additional mechanical adjustments.

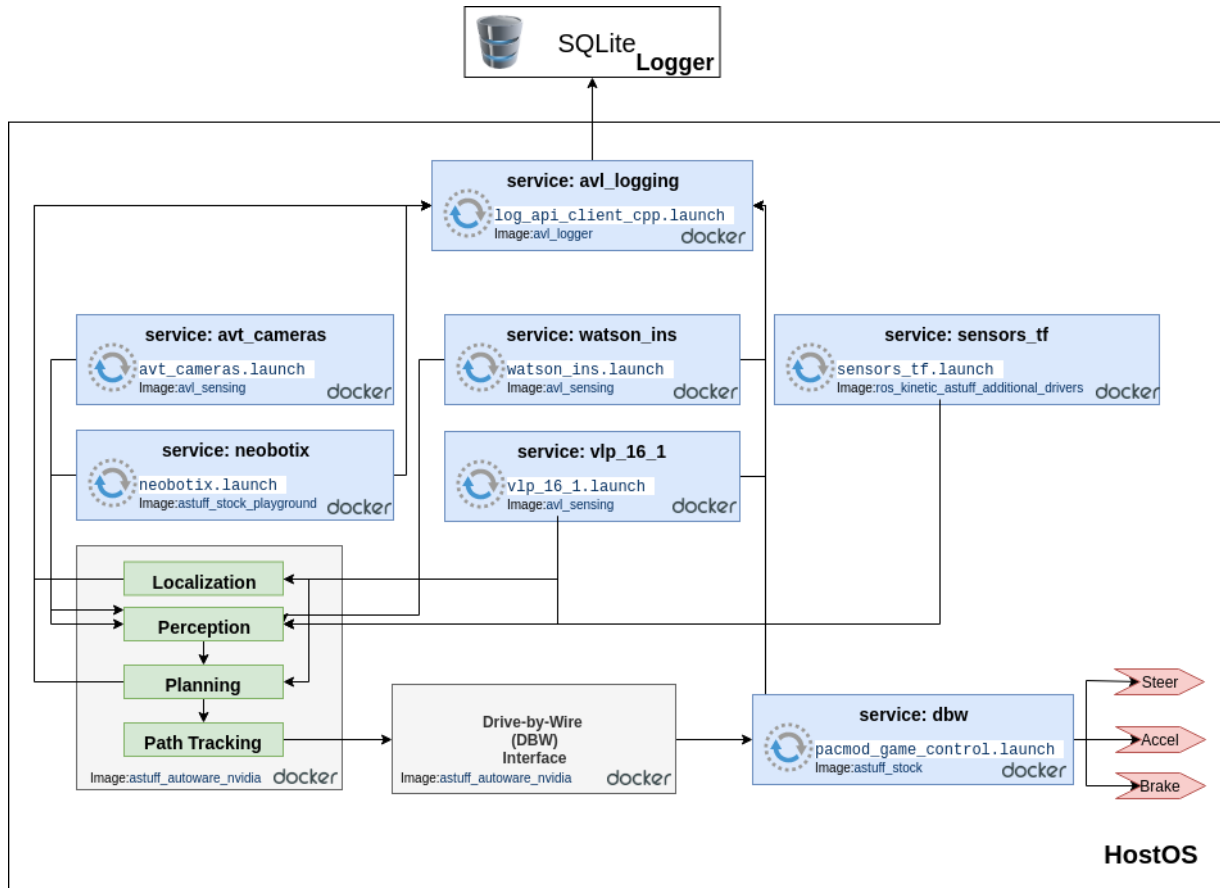
While steering control inputs can readily be computed as steering angle and angular velocity inputs due to finer grain control over the servo motor, accelerator and brake inputs are

treated as unitless intensity inputs that range between zero and one instead of values with units of  $m/s^2$ .

## 2.2 Module Abstractions and Containerization

Although the drive-by-wire system is one major part of controls, device and sensor drivers, logging modules, as well as the diverse high level decision making modules must be installed in a way that it is easy to maintain and keep track of the changes over time using version control. Given these constraints as well as package interdependencies that many libraries often require, one tool that facilitated software integration is Docker. Docker provides a set of tools including Docker Compose that has allowed the software modules to be isolated into different environments. With these abstractions introduced, issues related to compiling source code with incompatible software versions are bypassed making the software integration process faster and more efficient among multiple platforms. Additional benefits of using Docker include automatic module instantiation during boot-time and version control to guarantee that the software running among multiple vehicles behaves the same. The module abstractions actively used are shown in Figure 2.2. In the diagram, the modules shown in blue correspond to ROS nodes that are instantiated automatically using Docker Compose, while the modules shown in gray are manually operated by the user. While the latter is under user control to provide the flexibility of choosing which global plans to execute, the entire system is encapsulated using multiple Docker images as shown by the different labels. To the best of my knowledge, the systems used in the course of this study are among the first to utilize the concept of containerization for robotics applications [WC17] [WC18][PLH<sup>+</sup>19] With all of the data generated from vehicle position and reliability score, speed reports, steering angle positions, acceleration, and brake inputs, the `avl_logging` service collects this information and transmits it over HTTPs to a logging device. This logging device consists of a RaspberryPi based on Flask and a RESTful API that stores the collected information in

an SQLite database automatically. The data stored in the logger can then be transferred for offline processing without interfering with the host operating system or increasing computational overhead. Additional details on how the data collected is processed are covered in Chapter 4.



**Figure 2.2:** Docker Containers and Services running onboard autonomous vehicle platforms

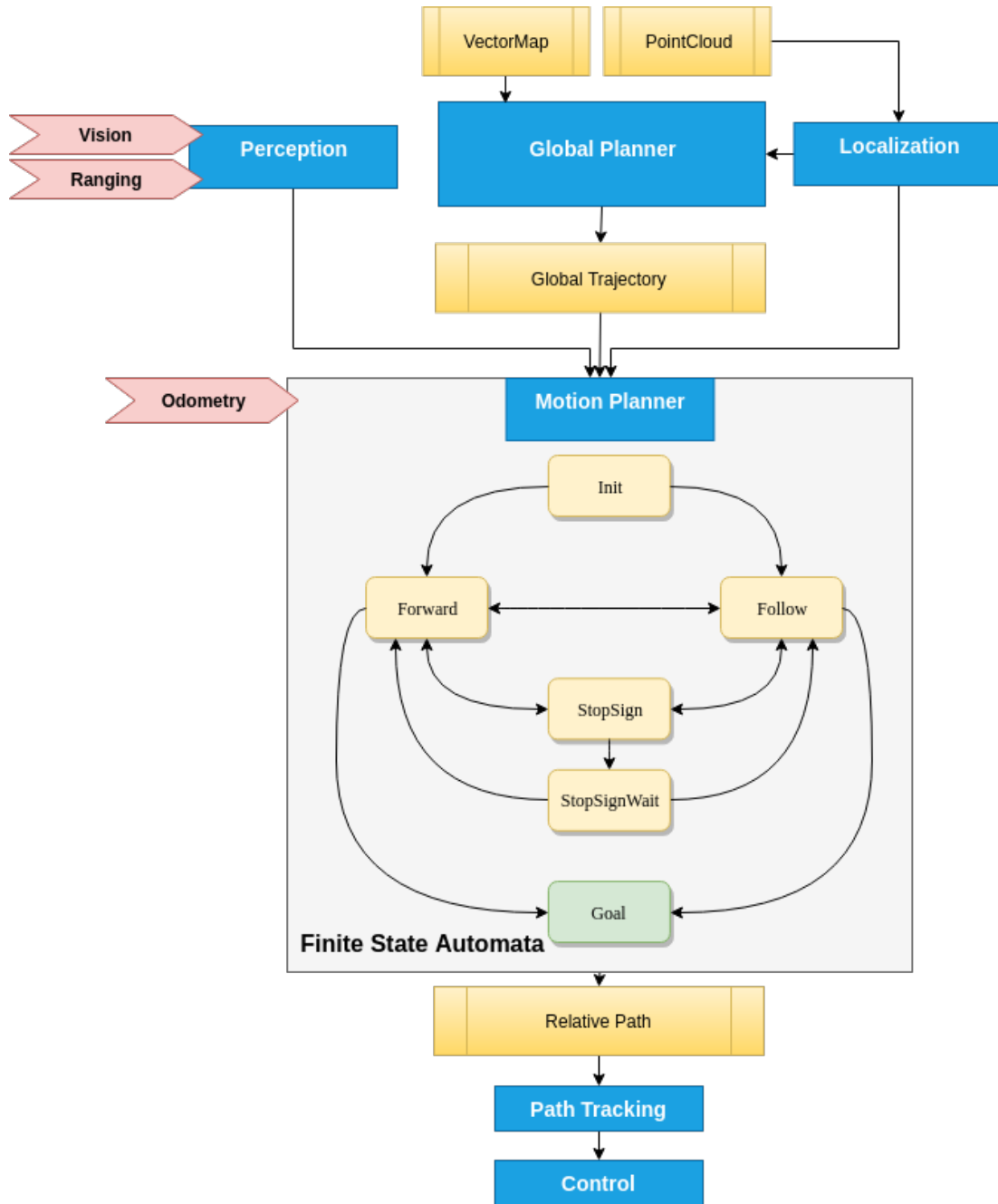
# Chapter 3

## Core Perception, Planning and Controls

### 3.1 Open Source Modules

Localization, perception, planning, and controls modules discussed in this chapter have been written in C++ and tested using the ROS ecosystem [QCG<sup>+</sup>09]. ROS provides the flexibility of integrating a large number of modules with a trivial communication framework that can be shared among many modules at once. Given that many ROS drivers and modules are open-source, this framework has become the ideal tool for fast prototyping in the robotics community. One popular ROS based framework geared towards self driving functionality is Autoware.AI [KTI<sup>+</sup>15]. This framework provides a platform of open source modules that provide certain features for localization, mapping, perception, and planning.

Although Autoware provides many tools that work great for specific applications, during the course of this study, a number of shortcomings were identified in perception, planning and localization. The methods introduced in this chapter aim to improve generalization and robustness while Autoware was used as the basis for the development framework. The software stack with extended functionality and modules introduced in this chapter is represented in Figure 3.1



**Figure 3.1:** Extended Autware software stack comprised of localization, perception, global planner, motion planner and control modules.

## 3.2 Localization, Mapping and HD Maps

Without accurate localization, motion planning and navigation becomes a cumbersome task. Even for human subjects, to be able to navigate through the diverse environments, the concept of relative location within a given environment is needed to execute an action and must be well understood. This is an area of research that has been actively explored over the years in robotics and has given rise to many robust implementations that work well under dynamic scenarios using both visual and LiDAR based methods [MMT15] [KRD08].

The approach taken in this study incorporates the concepts of Three Dimensional Normal Distributions Transform (NDT) [Mag09] that are applied for both mapping and localization. Using the 16-channel Velodyne LiDAR affixed to each development platform, the process of mapping and location consists of three steps: data collection, processing and downsampling.

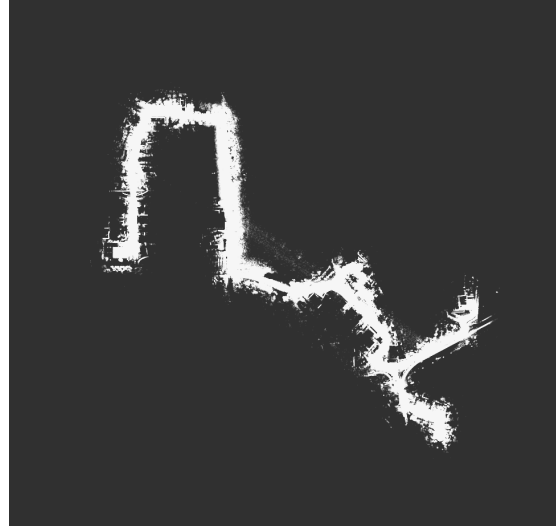
In the data collection process, a vehicle was driven along the roads and areas of interest. Given that NDT does not provide loop closure guarantees, special consideration has to be made to ensure that reasonable scan alignments are made at intersections by driving through overlapping scans multiple times. The data generated in this process was saved for offline postprocessing.

For small maps, the Point Cloud Library (PCL) [RC11] implementation of NDT is capable of generating point clouds close to real time sequentially or using a GPU; however, as the maps grow in size and complexity, LiDAR data processing must be performed sequentially. For point cloud maps with mile-long distances, the process of mapping can last several hours. The point cloud in Figure covers an extensive part of the UC San Diego campus and took approximately 28 hours to complete. Multiple point-cloud maps generated for the UCSD campus are shown in Figure 3.2.

Once the point cloud map has been generated, it is downsampled using the Voxel 3D Grid filter with one meter leaf size. While the data collection, mapping and post-processing process can take a couple of days and is sufficient for localization, manual annotation must be performed



(a) 3.9 mi dense-point cloud map.

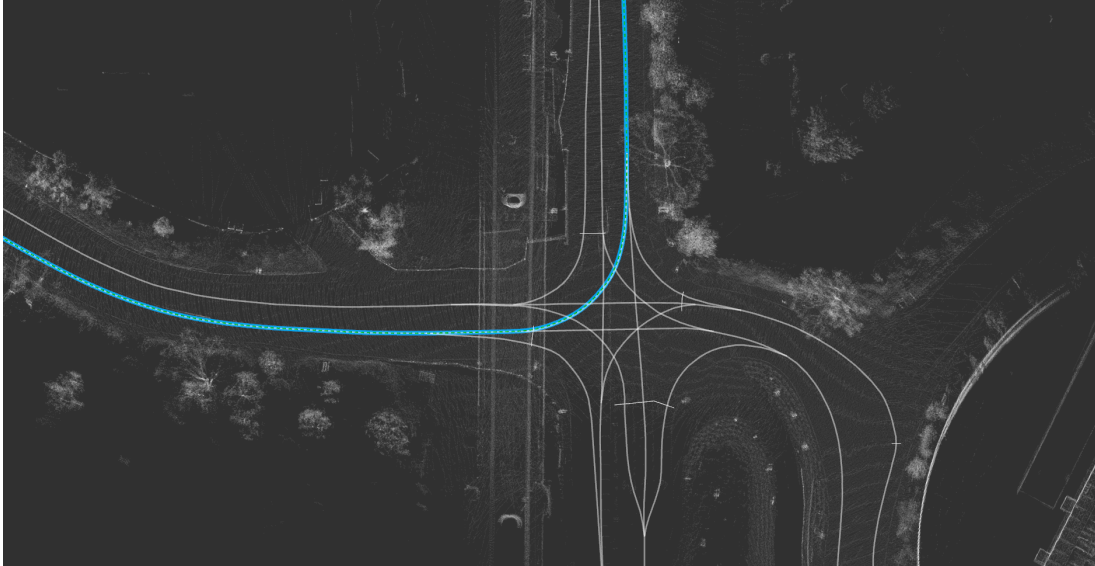


(b) Mail delivery route dense-point cloud map.

**Figure 3.2:** 3D Point Cloud Maps (top-down view) from UC San Diego.

to define lanes and intersection networks in order for the planner to generate a global plan and execute lower level actions during motion planning and control. Although aspects of this process can be automated to reduce overhead, certain portions of map maintenance and updates on road networks cannot be entirely delegated to automated systems and, as such, impact scalability; an example of a complex road network is shown in Figure 3.3 that uses the OpenPlanner format [DTT<sup>+</sup>17]. In the figure, a four-way intersection is defined by multiple lanes and stop lines. Due to constant campus construction, permitted turns and lane definitions changed multiple times in the course of a year and were redefined manually multiple times. Mitigations being actively explored will be revisited in Chapter 5.





**Figure 3.3:** Gilman/Voigt Intersection road network at UC San Diego. Trajectories shown in gray define lanes while the trajectory shown in blue corresponds to a planned trajectory.

## 3.3 Perception

The perception modules explored in this section utilize a combination of camera and LiDAR data. Although this is still an active area in development, some important aspects that impact perception are resolution, sensor configuration and location, and external environment conditions.

### 3.3.1 LiDAR Detection

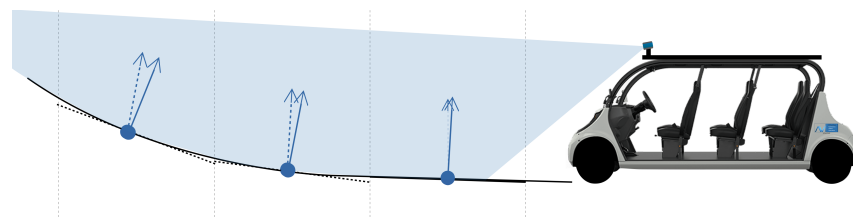
Although good results have been achieved using end-to-end learning and probabilistic methods, important factors to consider include LiDAR scan sparsity and range. With the 16-channel LiDARs mounted on the test vehicles, object definition and classification at distances above 20-30m becomes less than trivial. Given that many of the LiDAR detection algorithms utilize higher resolution LiDARs with 32 or 64 channels, a geometric approach is instead considered here.

LiDAR data is filtered and post-processed to define unlabeled clusters of objects based on distance and thresholds for the number of points each cluster can have. While an additional step for shape fitting can be performed for defining bounding boxes using L-shape fitting, the performance may depend on the perspective [Rac17]. Although the process for object clustering is trivial at a high level, the filtering step requires special attention to road conditions.

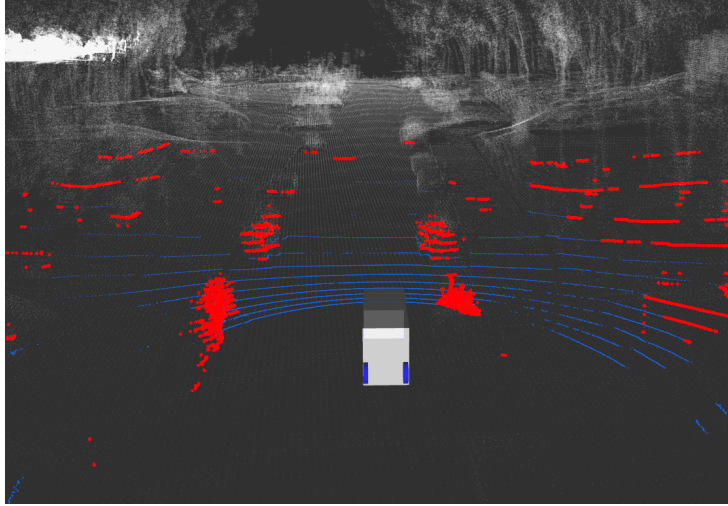
In the raw LiDAR data, relevant object information may be combined with objects with minimal interest including road surfaces. Without removing ground and road points, road obstacles are hard to distinguish during planning. Thus, in order to isolate objects of interest, one goal during the filtering step is ground removal.

One possible implementation can be readily implemented by identifying the LiDAR's pitch angle, defining a plane normal to the ground, and eliminating any points that fall below the plane. Although this approach works well on flat roads, in practice, false detections are experienced while operating the vehicle along roads with high degree of inclination such as the roads at UC San Diego.

Instead, the method implemented uses multi-plane definitions to separate ground points from potential obstacles[PLH<sup>+</sup>19]. In Figure 3.4, three planes are defined dynamically to adjust for road conditions using RANSAC and any points that fall below the defined region are removed. The remaining LiDAR points are clustered based on their relative distance and size as shown in Figure 3.5.



**Figure 3.4:** Three hyper planes are defined for ground removal.



**Figure 3.5:** Blue LiDAR scans correspond to points removed while red scans are considered as potential obstacles.

Even though consistent LiDAR based object detection for certain applications can be achieved with the approach discussed, critical information such as object and class types are often needed for planning and full scene understanding. As discussed in later sections, robust and consistent perception often requires multi-sensor fusion.

### 3.3.2 Multi-Object Detection using Cameras

As previously discussed, the LiDAR-only based approach discussed may miss road scene information including object classification. To provide additional information, camera data can be processed using multi-object detectors such as Faster-RCNN[RHGS15] and YOLOv3[RF18]. Given YOLO's real-time capabilities and fast inference times, this architecture was chosen as the primary deep-learning driven multi-object detector trained with the COCO dataset[LMB<sup>+</sup>14].

Figure 3.6 consists of a image frame extracted and classified in real-time at UC San Diego. This portrays the dynamic environment that is often overlooked from LiDAR information alone.



**Figure 3.6:** Realtime multi-object detection using YOLOv3 and the COCO dataset.

With this additional information, object behavior can be characterized and used to infer pedestrian and vehicle actions. However, introducing image data without depth information presents the task of estimating object pose in a world coordinate frame. Although there exist multiple methods for estimating depth, a simple approach explored involves perceptive transformations assuming plane-to-plane mappings.

To characterize a linear operator that can perform a perspective transformation, the Direct Linear Transformation for over-determined solutions is applied [HZ00]. This process consists of finding four or more correspondences on image frame and on an external plane that satisfy  $\mathbf{x}' = \mathbf{H}\mathbf{x}$ .

To minimize the errors and discrepancies between correspondences, it is often preferred to introduce additional data and optimize. This leads to an over-determined solution that can be solved using Singular Value Decomposition on  $\mathbf{A}$  as defined in Equation 3.1 where  $\mathbf{h}^{j\top}$  corresponds to the  $j$ -th row of the homography matrix  $\mathbf{H}$ . Given  $\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$ , the solution,  $\begin{bmatrix} \mathbf{h}^{1\top} & \mathbf{h}^{2\top} & \mathbf{h}^{3\top} \end{bmatrix}^\top$ , corresponds to the last vector of  $\mathbf{V}$  if the singular values of  $\mathbf{D}$  are sorted

in descending order.

$$\mathbf{A} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = \begin{bmatrix} \mathbf{0}^\top & -w'_i \mathbf{x}_i^\top & y'_i \mathbf{x}_i^\top \\ w'_i \mathbf{x}_i^\top & \mathbf{0}^\top & -x'_i \mathbf{x}_i^\top \end{bmatrix} \begin{pmatrix} \mathbf{h}^1 \\ \mathbf{h}^2 \\ \mathbf{h}^3 \end{pmatrix} = 0 \quad (3.1)$$

$$\mathbf{H} = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \quad (3.2)$$

In order to apply this method, four or more correspondences are needed. This can be readily performed as shown in Figure 3.7 making the process of estimating homography matrices trivial. By combining a camera's intrinsic properties, a location on an image can be projected onto the ground plane. Although this provides relatively good accuracy for pixel coordinates with ground plane correspondences, the ground is rarely flat: the formulation considered assumes plane-to-plane projections. Therefore, if an object protrudes above the ground or if the ground is curved, the estimate will be off.

In Figure 3.8, a point of interest on image plane is obtained using multi-object detection and it is projected on the ground plane. Even though the bounding box extracted provides a region for searching on image frame, choosing a pixel coordinate that best describes the object becomes a task for objects too close to the camera. Thus, the advantage of using perspective transformations for estimating locations relative to the vehicle works better for objects that are further away or that are represented at a smaller scale with respect to the image frame.





(a) Front driver side camera.



(b) Front passenger side camera.

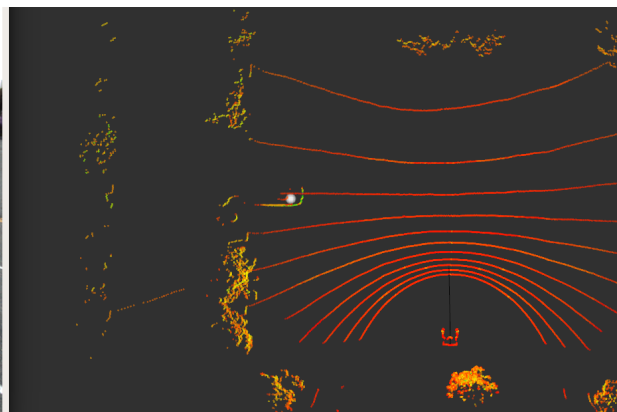


(c) Left side camera.



(d) Right side camera.

**Figure 3.7:** Image frame correspondences for front two cameras (a and b) and side cameras (c and d).



**Figure 3.8:** Projection from image frame (left) to ground plane (right) using Homographies.

### 3.3.3 LiDAR/Camera Fusion

LiDARs and cameras alone each solve perception tasks in different ways. LiDARs can provide information about whether there is an object around the vehicle or not but may not be able to help infer with full confidence if it is a vehicle or simply a ground-removal cluster of points misclassified as a potential obstacle. On the other hand, multi-object detection using a monocular camera can provide these details on image frame with higher confidence but without 3D pose information. Thus, sensor fusion becomes indispensable for robust sensing and estimation in dynamic environments such as autonomous driving. Camera and LiDAR fusion is discussed in this section but—in general—multi-sensor fusion can extend to other ranging and imaging sensors.

To determine LiDAR and camera correspondences and associations, calibration must be performed to identify the relative transformation between the camera and the LiDAR: extrinsics. This process assumes that the cameras used have been individually calibrated to determine their intrinsic characteristics. In the course of this study, intrinsic parameters for each camera were estimated using a checkerboard and the ROS camera calibration tools. Once camera intrinsics have been determined, the process of estimating the relative transformation between each camera and the LiDAR consists of solving the Perspective-n-Point problem (PnP). Although different algorithms have been proposed over the years, a non-iterative approach with  $O(n)$  complexity has been applied [LMNF09]. This process requires identifying at least four correspondences that can be matched between camera and LiDAR frames without needing a calibration target. Once enough correspondences are determined, an estimate for the relative sensor rotation is given by  $\mathbf{R} \in \text{SO}(3)$  and a translation vector  $\mathbf{t} \in \mathbb{R}^2$ . These two quantities effectively specify the relative sensor transformation known as extrinsics and can be expressed as  $\mathbf{T} \in \text{SE}(3)$  as shown in Equation 3.3. It should be noted that the transformations determined are converted into their corresponding quaternion representations as ROS provides a convenient abstraction for

quaternions.

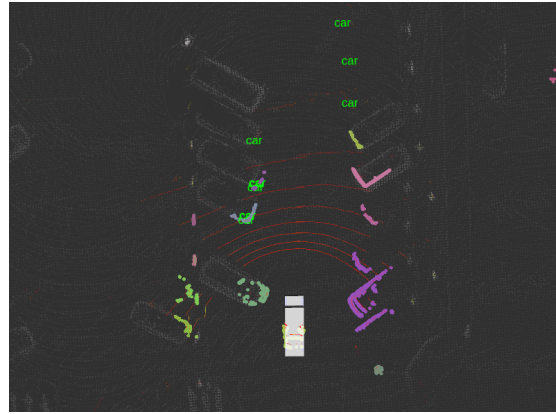
$$\mathbf{T} := \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0}^\top & 1 \end{bmatrix} \quad (3.3)$$

These extrinsics calculated can then be applied for multiple problems such as LiDAR cluster classification and LiDAR-camera based mapping. In Figure 3.9, clusters extracted from LiDAR data are projected on image frame and matched to the closest bounding box label provided by multi-object detection. For this data, a postprocessing step is often needed to eliminate LiDAR clusters that project to the same bounding box as described in Algorithm 1 by effectively matching a LiDAR cluster centroid to the closest LiDAR point projected on the center of its corresponding image bounding box using the L2 norm as a distance metric.





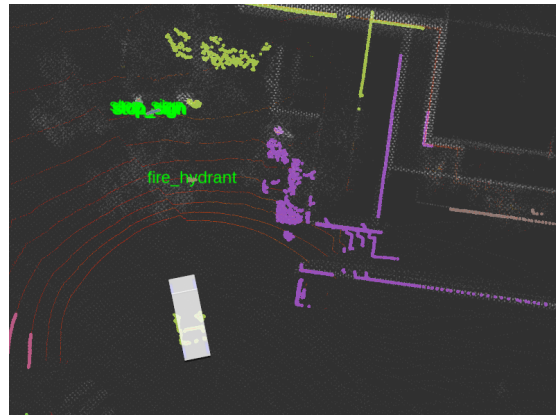
(a) Multi-object detection for parking lot scene.



(b) Labeled LiDAR clusters for parking lot scene using (a) (top-down view).



(c) Multi-object detection for road features and markings.



(d) Labeled LiDAR clusters for road features and markings using (c) (top-down view).

**Figure 3.9: LiDAR Cluster Classification:** Bounding boxes on the left (a and c) are used to provide labels for the LiDAR clusters shown on the right (b and d).

---

**Algorithm 1: Fusion and Association**

---

**Data:** LiDAR cluster centroids  $\mathbf{C}$ , multi-object detection bounding boxes  $\mathbf{B}$ , image projected LiDAR points  $\mathbf{L}_{img}$ , LiDAR points  $\mathbf{L}$ , camera intrinsics matrix  $\mathbf{K}$ , camera-LiDAR extrinsics matrix  $\mathbf{T}$

**Result:** Labeled cluster centroids  $\mathbf{C}^*$

```
for  $c_i \in \mathbf{C}$  do
   $\mathbf{c}_{img} \leftarrow \mathbf{K}\mathbf{T}c_i$ ;
  for  $b_i \in \mathbf{B}$  do
    if  $\mathbf{c}_{img}$  is within  $b_i$  then
       $\mathbf{L}_{img,i} \leftarrow \text{closest\_lidarpoint\_pixel}(b_i)$ ;
       $\mathbf{L}_i \longleftrightarrow \mathbf{L}_{img,i}$ ;
      if  $\text{distance}(\mathbf{L}_i, c_i) < \Delta$  then
         $c_i.\text{label} \leftarrow b_i.\text{label}$ ;
      end
    end
  end
end

 $\mathbf{C}^* \leftarrow \mathbf{C}$ 
```

---

### 3.4 Planning

While the ability to determine the relative position of an AV with respect to obstacles and roads is indispensable, the robot must also generate a plan for reaching an intended goal and revise existing plans based on real-time information to prevent collisions and execute actions as simple as making intersection stops or turns. Performing these actions requires a global planner and a motion planner to be in place to find an optimal path for the intended goal and to execute immediate short term actions, respectively.

In this section, the OpenPlanner [DTT<sup>+</sup>17] is used as a basis for implementing state-specific actions for dealing with vehicle following, cruising, and planned stops as well as incorporating complete road network functionality for global planning and replanning. Once a global plan is generated, the motion planner enforces speed limits and short term actions based on obstacles and publishes a final set of speed-encoded waypoints for path tracking.

### 3.4.1 Global and Motion Planning

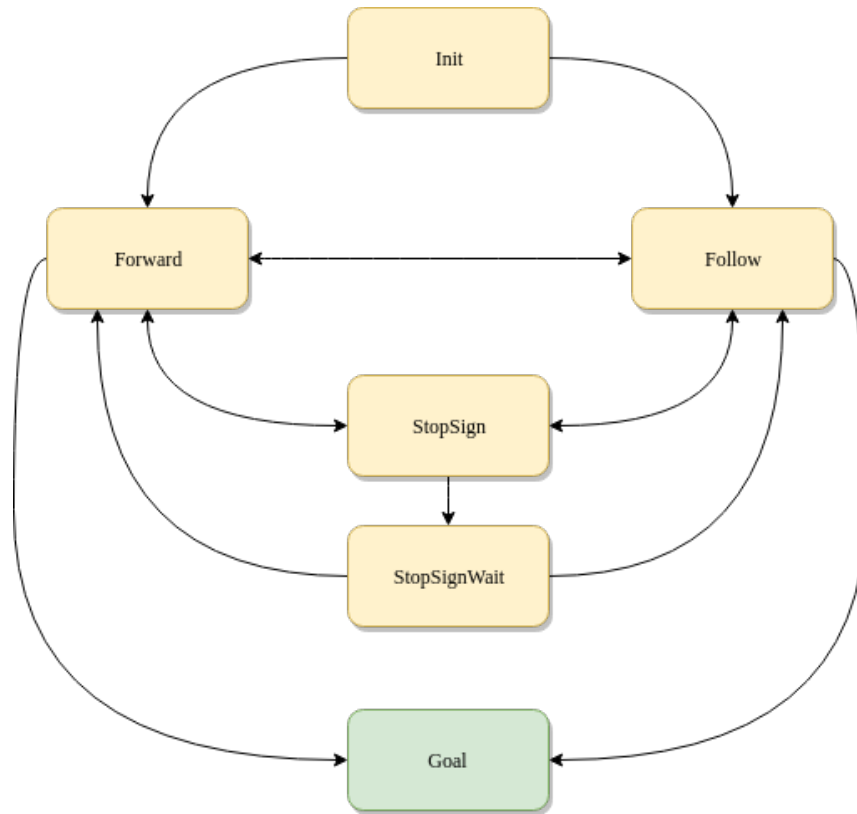
The core functionality for global planning starts with the design and annotation of vector maps [TKLD19]. As described in section 2, the process of annotation requires hours and often days to complete the detailed annotations for definition road networks, crosswalks, trajectories and stop lines. In practice, an additional step to encode speed limits on the predefined trajectories must be done.

With the required maps completed, the optimal shortest path between a starting point and a destination is determined recursively as long as road network connectivity exists. Although this path generated can be used for path tracking and executing low-level control actions, obstacles and rules of traffic must be incorporated. These constraints are enforced by a motion planner with a limited horizon to handle critical spatiotemporal tasks.

To process the continuous information provided from the perception stack while enforcing traffic rules, six of the OpenPlanner vehicle states are extended: Init, Forward, Follow, StopSign, StopSignWait and Goal. With this basis, target speeds are imposed based on speed limits, obstacles and intersections. Figure 3.10 reflects the Finite State Automata used while performing mail delivery trials at UC San Diego.\*

---

\*Due to limited testing with ObstacleAvoidance, this state is not included in live testing and this study.



**Figure 3.10:** Planning FSA

### 3.4.2 Cruising and Speed Keeping

Cruising and speed keeping is handled in the Forward state. In this state, the target speeds are a function of vehicle speed, speed limits and the drive-by-wire enable signal. When the vehicle’s drive-by-wire is disabled, the planner relaxes the previous target speed to avoid integral windup effects that often cause erratic behavior once the drive-by-wire system is re-enabled. When the vehicle’s drive-by-wire is enabled for the first time or re-enabled after a disengagement, the last known vehicle speed is used to initialize an incremental speed signal depending on the acceleration required. This design approach was taken to enforce speed requirements while maintaining invariance to road condition effects and reducing the overall dependence on low level

controllers. The implementation details are shown in Algorithm 2.

---

**Algorithm 2: Speed Keeping**

---

**Data:** drive-by-wire enable signal **auto\_enabled**, vehicle speed **v**, previous target

speed **v\_prev**

**Result:** target speed **v\_target**

**v\_target** = **v**;

**if** *auto\_enabled* **then**

**v\_target** = **v\_prev**;

**end**

**if** *v\_target* > *speed\_limit* +  $\Delta$  **then**

$a = a_{decel}$ ;

**else**

$a = a_{accel}$ ;

**end**

**v\_target** = **v\_target** +  $a \cdot dt$

---

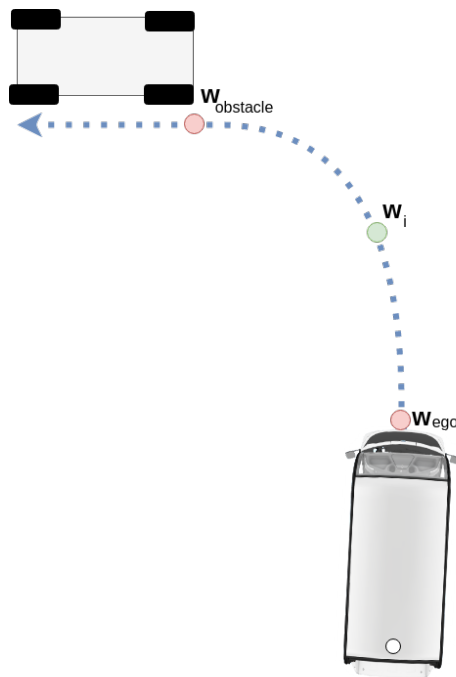
A trivial implementation initially explored was entirely dependent on the vehicle's last known speed report along with a desired acceleration to update the target speed as shown in Equation 3.4.

$$v_{\text{target}} = v_{\text{current}} + a \cdot dt \quad (3.4)$$

Although this approach works well along flat roads or roads with low degree of inclination, speed keeping becomes highly dependent on the low level controllers and their ability to react to small errors. As a result, this shifts the overall robustness to the controllers and their ability to recover from cumulative effects such as the integral sum from a PID controller.

### 3.4.3 Vehicle Following and Planned Stops

In the Follow state, the perception information is used to measure the relative distances between the ego vehicle and potential obstacles. Using the intended trajectory and the dimensions of the vehicle, if the lateral distance between a potential obstacle and the reference trajectory is larger than half of the ego vehicle's width, the trajectory is marked as blocked by assigning a high cost. This approach is shown in detail in Figure 3.11 with additional lateral and longitudinal safety distances. While it is possible to generate alternative lateral trajectory rollouts for obstacle avoidance, this implementation is still under active development and testing. As a result, the details discussed here focus on the special case in which selected trajectory rollouts are fully blocked.



**Figure 3.11:** Estimating distance to obstacles

For fully blocked trajectory rollouts, the distance to the next obstacle is measured by finding the closest rollout waypoint  $w_{\text{obstacle}}$  with respect to the obstacle and the closest rollout waypoint  $w_{\text{ego}}$  with respect to the ego vehicle. With these trajectory waypoints, the obstacle

distance is estimated by iteratively measuring the adjacent waypoint distances between  $w_{\text{obstacle}}$  and  $w_{\text{ego}}$  and adding them up as represented by  $d_{\text{obstacle}}$  in Equation 3.5—where  $w_{\text{ego}} \rightarrow w_i$  and  $w_{\text{obstacle}} \rightarrow w_j$  for  $i \leq k < j$ . The accuracy of these estimates are clearly dependent on waypoint discretization and density between points.

$$d_{\text{obstacle}} = \sum_{k=i}^j \sqrt{(w_{k+1} - w_k)^\top (w_{k+1} - w_k)} \quad (3.5)$$

Given the estimate for the distance to the closest obstacle  $d_{\text{obstacle}}$ , the initial approach consists of generating target speeds entirely based on linear kinematics under the assumption that a constant follow distance is to be achieved. Equation 3.6 represents the inverse relationship between the distance to the closest obstacle  $d_{\text{obstacle}}$  and the acceleration required for the ego vehicle with velocity  $v_{\text{ego}}$  to reach  $v_{\text{obstacle}}$ .

$$2a \cdot d_{\text{obstacle}} = (v_{\text{ego}}^2 - v_{\text{obstacle}}^2) \rightarrow a = (v_{\text{ego}}^2 - v_{\text{obstacle}}^2)/2d_{\text{obstacle}} \quad (3.6)$$

Although constant acceleration is assumed, sampling the ego vehicle's speed and updating the estimate for the distance to the closest obstacle continuously can provide good acceleration estimates that can be used to estimate a target speed by applying Equation 3.4. This approach works well for lower speed driving and scenarios where obstacles' speeds do not differ considerably with respect to the ego vehicle's speed; however, the bottleneck for this approach is generally experienced during high speed driving due to the assumption that a constant follow distance must be kept. Faster moving vehicles require the following distance to be relaxed in order to ensure a safe stopping distance in the event that an emergency stop needs to be made.

To mitigate the effects of using constant following distances, the vehicle following state is entered as soon as the distance to the closest obstacle comes less than the expected distance for a complete stop  $d_{\text{stop}}$  as estimated based on Equation 3.7, where  $a_{\text{brake}}$  is a precalculated deceleration rate in  $m/s^2$  based on an average of worst-case braking performance. Estimates

for  $a_{\text{brake}}$  were determined by performing a series of stops at different speeds while applying constant braking force and measuring the complete stopping distance. This vehicle deceleration rate estimate can be generalized for other behaviors such as planning for intersections.

$$d_{\text{stop}} = v_{\text{ego}}^2 / 2a_{\text{brake}} \quad (3.7)$$

While intersections with stop signs require different logical actions and maneuvers than intersections with traffic lights, the details for stopping at intersection lines generalize to both, stop-line intersections and traffic-light intersections along different road conditions.

Given the ego vehicle speed  $v_{\text{ego}}$  and a stopping distance estimate under a precalculated constant deceleration estimate that is given by  $d_{\text{stop}} = v_{\text{ego}}^2 / 2a_{\text{brake}}$ , planning for stops and intersections becomes a special case of vehicle following. See Figure 3.12. The same planning approach can then be defined for both, vehicle following and stopping where  $d_{\text{obstacle}} \rightarrow d_{\text{stopline}}$ ,  $w_{\text{obstacle}} \rightarrow w_{\text{stopline}}$  and  $v_{\text{obstacle}} = 0$  for stops and intersections.



**Figure 3.12:** Planning for Stops



While the precalculated deceleration  $a_{\text{brake}}$  is simply an estimate that roughly describes the ego vehicle's braking performance, the measured distance to the next obstacle or in the case of intersections, stop line distances, provide a larger weight for the acceleration estimated in Equation 3.6 as it is inversely proportional to the distance from Equation 3.7. As the distance to the next intersection or obstacle decreases, the acceleration's magnitude approaches infinity. Hence, while the precalculated deceleration rate  $a_{\text{brake}}$  is used as a trigger for switching states, the accuracy and robustness of vehicle following and stopping relies on accurate measurements of obstacle distance and stop/intersection distance. During live testing, the combination of the precalculated deceleration rate and measured obstacle/intersection distances provided the most constant and accurate results.

## **3.5 Path Tracking and Controls**

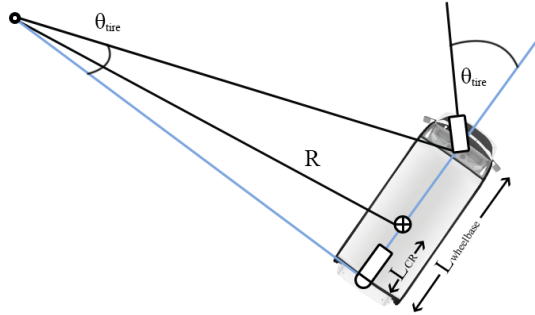
### **3.5.1 Vehicle Geometry for Steering Control**

#### **Path Tracking and Steering Angle**

With the final set of speed-encoded waypoints generated during motion planning with a limited horizon, the radius of curvature and tire angles can be approximated by Pure Pursuit [Con92]. Pure Pursuit is a path tracking algorithm that uses a lookahead distance to determine a moving target waypoint along the ego vehicle's intended trajectory. With the waypoint determined, an arch is fit between the moving waypoint and the location of the rear axle of the vehicle. This arch essentially determines the curvature  $K$  required for the ego vehicle to reach the target waypoint.

Given that the radius of curvature is inversely proportional to curvature,  $R = 1/K$  the vehicle's tire angle can be modeled using a basic bicycle kinematics model. Equation 3.8 and Figure 3.13 describe the relationship between the radius of curvature and the geometric

approximations needed to estimate the tire angle. With the vehicle-specific steering-rack ratio  $C_{rack}$ , the steering angle  $\theta_{tire}$  can be determined by Equation 3.9.



**Figure 3.13:** Bicycle Model Approximation

$$\theta_{tire} = \arctan \left( \frac{L_{wheelbase}}{(R^2 - L_{CR})^{1/2}} \right) \quad (3.8)$$

$$\theta_{steer} = C_{rack} \theta_{tire} \quad (3.9)$$

### Steering Angular Velocity

The angular velocity for the steering wheel is modeled under the notion that minimal steering should be performed at high speeds while offering high maneuverability at low speeds. Thus, a trivial relationship that offers this flexibility is given by Equation 3.10. where  $\beta$  is a constant greater than 1.0 and smaller than 2.0. By analyzing the boundary conditions, it can be observed that when the vehicle is operating at high speeds, the vehicle-speed to max-speed ratio ( $\frac{v_{current}}{v_{max} \cdot \beta}$ ) becomes close to one and as a result the angular speed drops. On the other hand, when the vehicle is driving at low speeds, the ratio drops close to zero and a maximum angular velocity

can be achieved. This relationship is linear on the speed of the vehicle.

$$w = w_{\max} \left[ 1 - \frac{v_{\text{current}}}{v_{\max} \cdot \beta} \right] \tag{3.10}$$

### 3.5.2 Acceleration and Braking Controls

#### Acceleration and Braking

The drive by wire system consists of floating point control inputs for acceleration and braking between zero and one. Given that target speeds and accelerations set by the planner must be mapped to unitless quantities, two traditional PID controllers are employed for the task as shown in Figure 3.14.

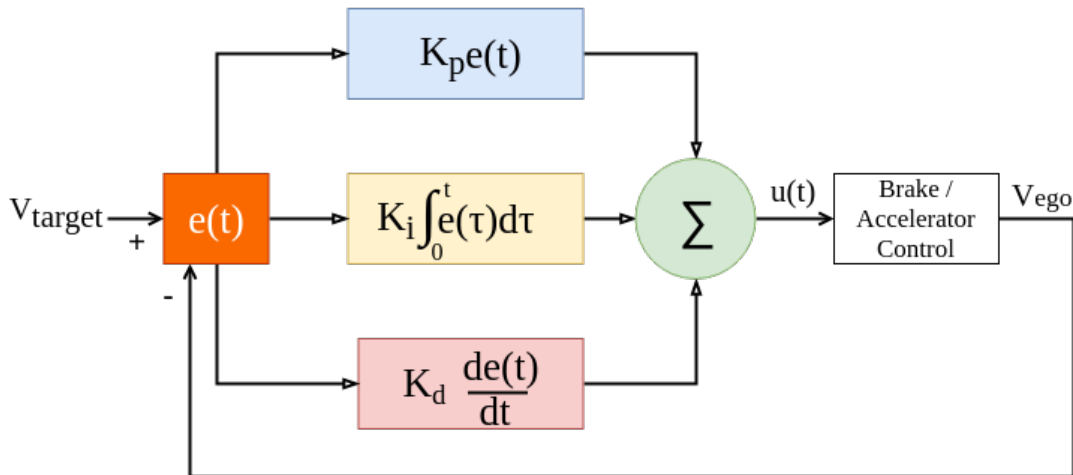


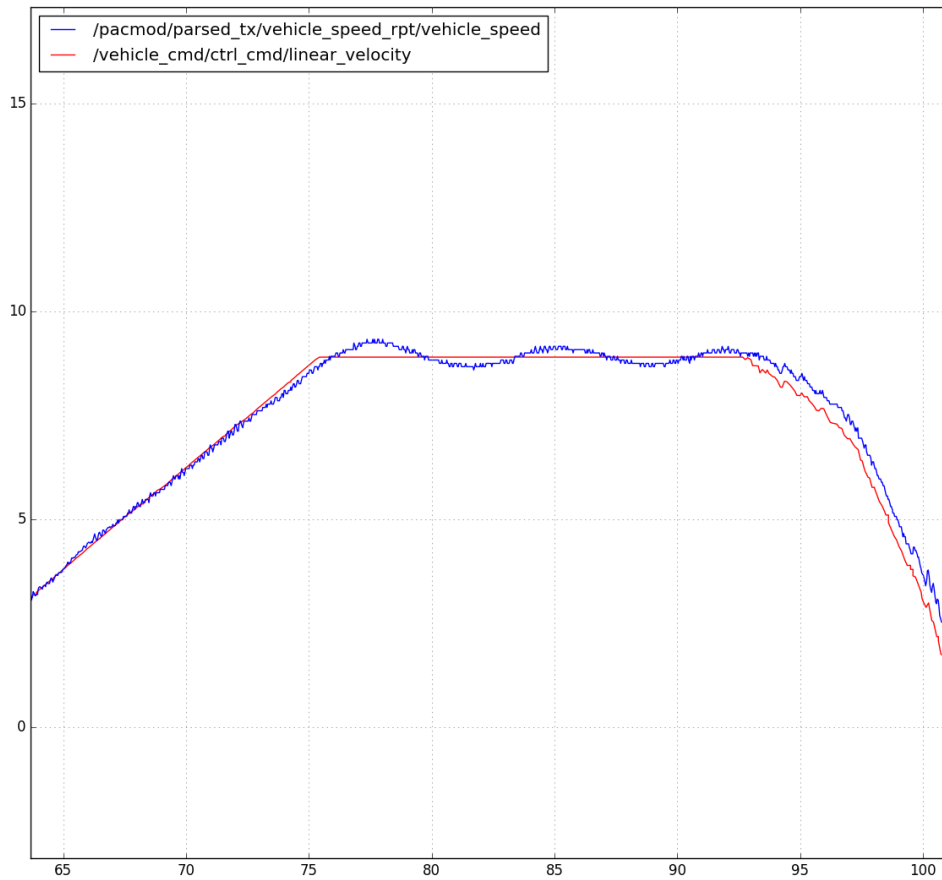
Figure 3.14: PID Controller

The error  $e(t)$  is defined as the difference of the target speed and the current vehicle’s speed as reported by odometry measurements. To keep the signs and units consistent.  $e_{\text{accel}} = e(t)$  and  $e_{\text{brake}} = -e(t)$  are defined for each PID controller. Given this convention, the control equations

are trivially applied as shown in Equation 3.11.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de(t)}{dt} \quad (3.11)$$

It should be noted that as with other applications of the PID controller, there are cases in which the integral sum must be reset. For this control strategy, the integral sums are reset every time the vehicle's speed is relatively close to zero. An additional step taken involves integral sum thresholding. Thresholding prevents the integral sums from generating a greater weight than other terms while providing faster response times and less overshooting. Adequate values for the threshold were determined based on the minimum and maximum control values: 0.0 and 1.0. In Figure 3.15, the target speed and current vehicle speed are shown as a result of vehicle acceleration, followed by an intersection stop under the control strategy discussed.



**Figure 3.15:** Target Speed (m/s) shown in red and Current Vehicle Speed (m/s) shown in blue as functions of time(s)

Chapter 3 includes material as it appears in Lessons Learned From Deploying Autonomous Vehicles at UC San Diego in Field and Service Robotics, Tokyo, JP, August 2019. David Paz, Po-Jung Lai, Sumukha Harish, Hengyuan Zhang, Nathan Chan, Chun Hu, Sumit-Binnani, and Henrik Christensen. The thesis author was the primary author of this paper.

The methods introduced on LiDAR-based detection (section 3.3.1) and LiDAR/Camera fusion (section 3.3.3) were the result of a collaborative effort between Hengyuan Zhang, Sumukha Harish and the primary author. With Zhang’s work on multi-plane fitting methods and Harish’s early work on Camera/LiDAR fusion, these modules were incorporated into the Autoware software stack and deployed using the two autonomous vehicle platforms introduced in this study.

# Chapter 4

## Benchmarking and Performance

### Evaluation for Autonomous Vehicles

The UC San Diego campus covers diverse terrain conditions with an area that spans 2,178 acres. With 10,625 academic staff, 35,827 non-academic staff, and over 38,000 undergraduate and graduate students enrolled for fall 2018, the campus experiences active traffic\*. Taking into consideration these factors, along with the healthcare system and remote campus locations, transportation and mobility becomes an important factor for students, staff, faculty and visitors. While this project is conducted in a campus scenerario, last mile transportation generalizes to many scenarios in which mobility may be restricted due to parking or walking-distance constraints. With these logistic challenges in mind, this project is motivated by exploring alternative solutions such as on-demand automated shuttle service and mail delivery services that can enhance mobility for the last-mile without impacting congestion. In the initial phase of this project, the two GEM e6 vehicles introduced in earlier chapters were used for software development and testing between fall 2018 and mid-spring 2019. During this phase, mail delivery logistics were discussed with the UC San Diego operations to determine the routes of interest that the mailing center covers. This

---

\*<https://ucpa.ucsd.edu/campus-profile/>

led to the automated mail delivery pilot phase which began summer 2019 and has been an active project since.

## 4.1 Safety

Given the complexity of the autonomous system actively deployed during standard tests and mail delivery runs, precautions are actively taken to ensure that the vehicles operate safely. In the event that the vehicle requires manual interventions to be performed, a trained safety driver<sup>†</sup> will take control of the vehicle by applying a torque on the steering wheel or engaging on the brake pedal. This drive-by-wire provides convenient functionality that allows the safety driver to immediately regain control of the vehicle at any point in time. An emergency physical disengagement button is also made available as shown in Figure 4.1.



**Figure 4.1:** Emergency Stop on AV dashboard

---

<sup>†</sup>All safety drivers participating in this project have been trained by the UC San Diego Risk Management's Office. Each safety driver is required to maintain his or her attention on the road at all times and must be capable of performing actions as if him or her were operating the vehicle.

## 4.2 UMAV: Unbiased Metrics for Autonomous Vehicle Benchmarking

As of fall 2019, the vehicles have driven more than 89.9km in autonomous mode at the UC San Diego campus; this corresponds to driving 6.9 hours with the autonomous system engaged. For every trip in which the system was engaged, the logger introduced in Section 2.1 recorded position and reliability information, vehicle speed reports, steering control inputs and reports, acceleration control inputs and reports, braking control inputs and reports, system enable signals, as well as the target speeds imposed by the planner. In addition to the data recorded by the logger, these signals were also recorded along with camera and LiDAR data using the ROSBAG file format provided by ROS.

While system uptime and distance data provides an idea of the amount of data collected during mail delivery, information in regards to the number of interventions performed within these periods of time is highly relevant for characterizing performance. As introduced in Chapter 1, without spatial and temporal information, unbiased statistics cannot be readily assessed. To generate a comprehensive evaluation, an initial set of statistics are proposed for measuring robustness in terms of the number of manual interventions performed: Mean Distance Between Interventions (MDBI) and Mean Time Between Interventions (MTBI). By definition, these statistics can be computed by Equation 4.1 and Equation 4.2, respectively.

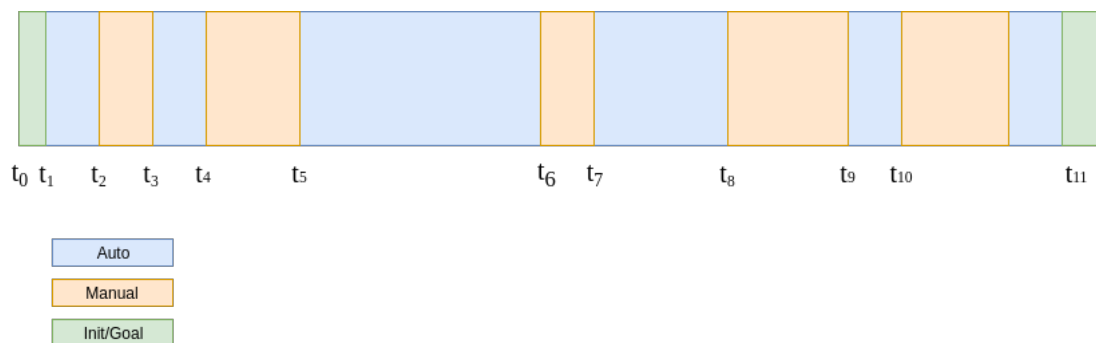
$$\text{MDBI} = \frac{\text{Total Distance}}{\text{Number of Interventions}} \quad (4.1)$$

$$\text{MTBI} = \frac{\text{Total Uptime}}{\text{Number of Interventions}} \quad (4.2)$$

To measure the number of interventions performed over time, the enable/disable signal is recorded over time. This signal is set to “True” when the vehicle is operating in autonomous



mode and it is toggled to “False” as soon as an intervention is performed: the safety driver applies a torque on the steering wheel or taps the brake or accelerator pedals. With the ability of measuring these enable/disable signals, the number of interventions per trip can be accurately measured by the number of enable/disable signal toggles. This idea can be visualized by Figure 4.2—where the blue segments correspond to autonomous operation and the orange segments to manual operation. The green segments correspond to mission start and mission end segments; during the mail delivery deployment at UC San Diego, these segments correspond to manual operation since the vehicle had started recording data before any planning or execution began. This was often the case when packages and mail were loaded onto the vehicle and an enable signal had not been set. For this reason, these starting and ending segments are relaxed and are not considered as interventions.



**Figure 4.2:** Enable signal as a function of time.

To estimate MTBI and MDBI, distance and time are calculated as given by Equation 4.3 and Equation 4.4, respectively. Given the time sequence  $t_1, t_2, \dots, t_{2N}$ , the distances and times measured correspond to the regions in blue from Figure 4.2. This effectively measures how much time elapsed and how much distance was covered while the autonomous system was engaged.

$$d = \sum_{i=1}^N (d(t_{2i}) - d(t_{2i-1})) \quad (4.3)$$

$$t = \sum_{i=1}^N (t_{2i} - t_{2i-1}) \quad (4.4)$$

In the ideal system, the number of interventions will approach zero. Therefore, estimates for MDBI and MTBI follow a  $\frac{C}{0}$  limit form. While this can be expressed as a limit, it is more convenient to analyze the inverses  $\text{MDBI}^{-1}$  and  $\text{MTBI}^{-1}$  to prevent divide-by-zero errors. Following this convention, the new expressions are given by Equation 4.5 and Equation 4.6; where optimal values for  $\text{MDBI}^{-1}$  and  $\text{MTBI}^{-1}$  approach zero.

$$\text{MDBI}^{-1} = \frac{\text{Number of Interventions}}{\text{Total Auto Distance}} \quad (4.5)$$

$$\text{MTBI}^{-1} = \frac{\text{Number of Interventions}}{\text{Total Auto Uptime}} \quad (4.6)$$

Depending on how these quantities are measured, additional information about safety driver dependability can be measured. Considering the case in which, very long interventions are happening, it is useful to know additional statistics that describe the time elapsed and distance covered while the interventions were performed. To do this, MDBI and MTBI can be measured for both, the time and distances in which the vehicle is operating autonomously and the time and distances in which the safety driver is operating the vehicle manually. It should be noted that special attention must be paid when estimating the manual operation statistics; in the ideal system the statistics measured for manual operation will have an indeterminate limit form. To denote the difference between these statistics, subscripts are used to differentiate these statistics as shown in Equation 4.7, Equation 4.8, Equation 4.9, and Equation 4.10.

$$\text{MDBI}_A^{-1} = \frac{\text{Number of Interventions}}{\text{Total Auto Distance}} \quad (4.7)$$

$$MTBI_A^{-1} = \frac{\text{Number of Interventions}}{\text{Total Auto Uptime}} \quad (4.8)$$

$$MDBI_M^{-1} = \frac{\text{Number of Interventions}}{\text{Total Manual Distance}} \quad (4.9)$$

$$MTBI_M^{-1} = \frac{\text{Number of Interventions}}{\text{Total Manual Uptime}} \quad (4.10)$$

### 4.3 Vehicle Performance for Mail Delivery

Using the metrics proposed, the data collected from mail delivery runs at UC San Diego is actively being used to measure vehicle performance over time. Table 4.1 corresponds to the baseline collected over a period of 13 days which include summer and fall quarter 2019. From the table that corresponds to Summer 2019, it can be inferred that—on average—the vehicle drove 414.2m autonomously before an intervention was made. In a similar way, the vehicle drove autonomously, on average, for 113.8 seconds before an intervention was made. During the same period, the dependency on the safety driver can be characterized by the metrics generated during manual driving. On average, each intervention performed lasted 24m or 12.77 seconds. These statistics extend for fall 2019 and have been combined in Table 4.2.

**Table 4.1:** MDBI and MTBI Summary Interventions for summer and fall quarters.

<b>Summer 2019</b>	MDBI	MDBI <sup>-1</sup>	MTBI	MDBI <sup>-1</sup>
Autonomous	414.201	0.0024	113.82	0.00878
Manual	24.00	0.0416	12.77	0.07829
<b>Fall 2019</b>				
Autonomous	283.08	0.0035	84.44	0.0118
Manual	19.25	0.0519	11.54	0.0866

**Table 4.2:** MDBI and MTBI Overall Summary Interventions.

<b>Overall</b>	MDBI	MDBI <sup>-1</sup>	MTBI	MDBI <sup>-1</sup>
Autonomous	380.42	0.0026	106.25	0.0094
Manual	22.77	0.0439	12.46	0.08028

It can be observed that the statistics significantly vary between summer and fall quarters. This significant difference can be explained by campus traffic and ongoing activities experienced early in fall quarter. In the fall, the UC San Diego campus experiences higher traffic and foot activity from students moving in or starting classes. Separating these results by quarterly basis and testing location can potentially help explain trends and traffic patterns. At the same time, it is important to estimate collective averages to make note of the impact of the software release versions on the performance. To facilitate this process, manual notes are taken during mail delivery runs that include safety driver, co-pilot, route information, weather conditions, and the type of interventions defined.

## 4.4 Autonomous vs Manual Driving

While the data and statistics introduced in this chapter represent early results from an active project, the metrics defined effectively quantify the performance in terms of the number of interventions performed of an autonomous vehicle system independently of the vehicle type. By simply collecting speed information, drive-by-wire enable signals, and vehicle status reports, a complete system characterization can be performed. UCSD's AVL has started using this system along with an additional suite of tools for benchmarking control, fuel/energy efficiency and even cost of ownership as shown in Table 4.3.

**Table 4.3: UMAV Metrics**

<b>Trigger</b>	<b>Metric</b>	<b>Type</b>
Intervention	Mean Distance Between Interventions	Event Driven
Intervention	Mean Time Between Interventions	Event Driven
Energy	Miles per Gallon (MPG) Or Charge Consumed (A)	Continuous
Maintenance Cost	Brakes and Tire Wear	Continuous
Up-time	Time Elapsed Per Trip	Event Driven
Control	Speed, Acceleration, Steer Angle Fourier Transform	Continuous

These additional metrics provide vehicle specific information for comparison between autonomous vs manual modes. As previously introduced, MDBI and MTBI permit system evaluation at the software level. On the other hand, the introduction of an energy consumption metric is geared towards vehicle specific evaluation such as autonomous trucks, but it also permits a direct comparison with manual driving fuel consumption. Given the variety of vehicle-specific powertrains, fuel consumption models or battery power measurement devices may be required. With the logging devices introduced in Section 2.2, a fuel consumption model was explored as part of a UC San Diego/TuSimple collaboration to evaluate the performance of level-4 autonomous trucks. These studies show that the use of autonomous driving trucks can improve fuel efficiency by approximately 10% and up to 20% compared to manual driving<sup>‡</sup>. In addition to modeling and evaluating fuel consumption, there are a number of implications on energy consumption and cost of ownership as a result from autonomous systems' control strategies. For example, depending on the steering, acceleration and braking controls, more energy may be required to drive along the same routes if a system overcompensates for small errors. As a result, this can impact fuel consumption, and brake and tire wear. These cumulative effects can affect the overall cost of ownership of a vehicle, as well as the environmental impact. For benchmarking purposes, the measured steering, acceleration and braking status reports can be compared in the frequency domain for autonomous and manual driving.

<sup>‡</sup><https://finance.yahoo.com/news/study-finds-tusimple-trucks-least-153452983.html>

Chapter 4 includes material as it appears in Lessons Learned From Deploying Autonomous Vehicles at UC San Diego in Field and Service Robotics, Tokyo, JP, August 2019. David Paz, Po-Jung Lai, Sumukha Harish, Hengyuan Zhang, Nathan Chan, Chun Hu, Sumit-Binnani, and Henrik Christensen. The thesis author was the primary author of this paper.

The Unbiased Metrics for Autonomous Vehicle Benchmarking (UMAV) introduced in section 4.2 are the result of a year-long study in which Po-Jung Lai, the primary author and principal investigator Henrik Christensen participated in. Po-Jung Lai and the primary author developed the logging devices and performed system evaluation on the two autonomous vehicle platforms introduced in this study. Lai participated as the project lead for the UCSD/TuSimple study discussed in section 4.4.

In addition, chapter 4 contains pre-publication material and statistics obtained during the UCSD mail delivery project being prepared for review.

# Chapter 5

## Achieving Full Autonomy - L5

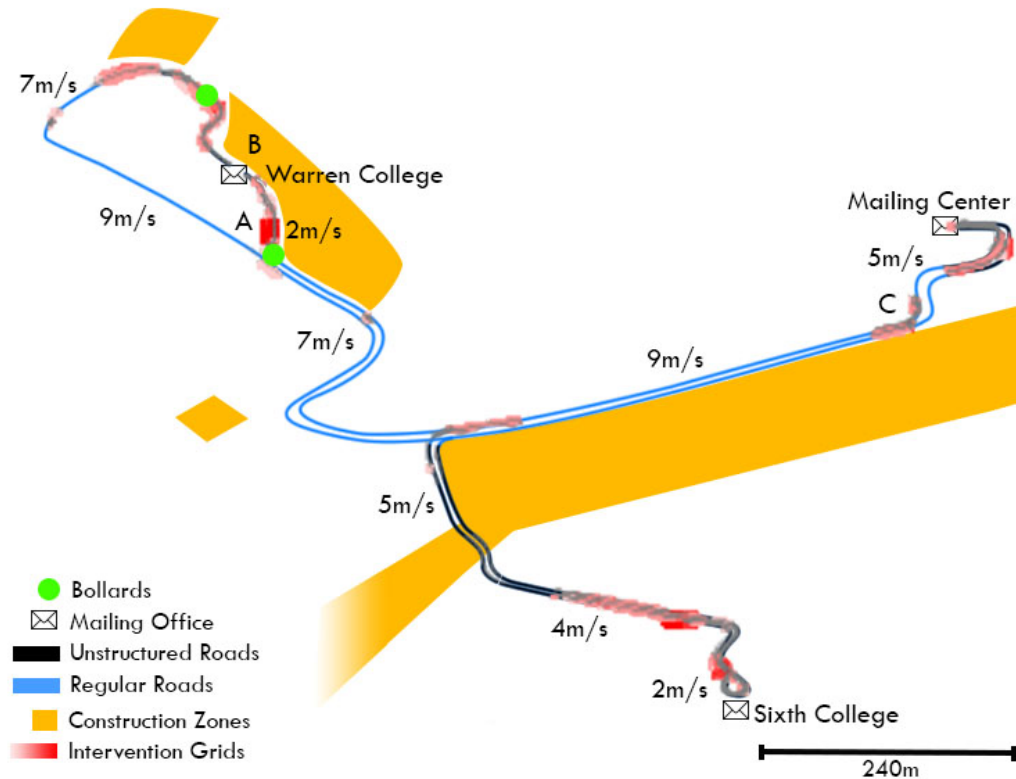
### 5.1 Overview

The design considerations introduced in Chapter 2 and 3 have been crucial for collecting and benchmarking autonomous vehicle data at UC San Diego. While a comprehensive study is actively being performed, the UMAV system introduced and discussed in Chapter 4 provide a comprehensive set of benchmarks for autonomous cars in general.

During the initial mail delivery trial on the UC San Diego campus, the results obtained using UMAV have provided a comprehensive means for benchmarking control robustness and safety driver dependability—these are statistics that are continuously varying as system development continues and overall system robustness improves.

With 89.9km of autonomous driving data collected during initial mail delivery runs, one important tool that has facilitated the analysis of traffic patterns and dynamic planning bottlenecks involves intervention maps. Given that each autonomous vehicle navigates with respect to a fixed frame of reference continuously, it is possible to associate each intervention performed by the safety driver with a pose. This leads to the notion of building complete normalized intervention maps as shown in Figure 5.1. Figure 5.1 contains normalized intervention information for two

mail delivery routes on campus. Each route has been normalized and superimposed with a  $1m^2$  grid resolution based on location and intervention count. It should be noted that the statistics reported in Tables 4.1 and 4.2, correspond to the intervention map introduced in this section.



**Figure 5.1:** Mail delivery route intervention map.

In general, it can be seen that the highest number of interventions occur at intersections and unstructured environments such as walkways: locations A, B and C. Some of these locations and places of interest will be analyzed closely in the upcoming sections.

The patterns observed at these locations shed light on some of the bottlenecks experienced but at the same time provide a sense of the challenges in autonomous driving in general. Autonomous driving systems must be capable of negotiating the right-of-way at road intersections just as well as detecting pedestrian traffic and determining intent in unstructured environments. While many regular vehicles are not allowed to navigate through pedestrian walkways such as the



ones at UCSD\*, there exist other unstructured environments that are continuously changing and become dynamic problems that must be solved in real-time such as SLAM. With these cases in mind, potential research directions will be discussed closely in the following sections.

## 5.2 Intent for Unstructured Environments

In this section, the challenges related to scene understanding and intent for dynamic planning will be discussed whereas the implications of fast-changing environments on the generalized problem of autonomous driving will be left for the next section.

The road segments from locations A, B and C in Figure 5.1. correspond to some locations in which a higher number of interventions were experienced. These segments in particular demonstrate multiple challenges including navigating through tight environments, analyzing pedestrian intent and determining the appropriate order for navigating through a four-way intersection.

Location A corresponds to the beginning of a large pedestrian walkway that forks off from the main road as shown in Figure 5.2a and is protected by bollards as seen in Figure 5.2b—the bollards are represented on Figure 5.1 by green circles. The bollards on campus are not always equally spaced apart and may not always be traversable with the GEM e6 vehicles; the bollards protecting the walkway shown in the images range between 163.8cm and 168.9cm. For comparison, each GEM e6 vehicle measures 141cm in width—which leaves approximately 11cm of clearance on each side.

Since a lateral safety margin of 50cm was integrated for additional safety in the planning modules, the vehicle automatically comes to a stop and waits for a safety driver intervention when it reaches these bollards; these types of interventions were captured on the intervention map and the UMAV system. During testing, these bollards were not always present due to different

---

\*At UCSD, the golf carts used for mail delivery operate in different scenarios including side walks, pedestrian walkways and off-road paths. While UCSD vehicles—including the self-driving vehicles used in this study—are authorized to operate in these areas, many of these paths are not intended for public use.

construction patterns and as a result the bollard-protected trajectories shown on the intervention map do not have the same intervention counts. Additional perception and planning improvements such as adaptive lateral and longitudinal safety distances are being performed to address these corner cases; however, this becomes a more challenging task to perform when there are human subjects or other stochastic events are in the loop.



(a) Mail delivery route fork.



(b) Walkway protection bollards.

**Figure 5.2:** Scenes of Warren college mail delivery route.

Navigation through shared pathways such as parking lots, crosswalks and even side walks require additional attention. In the case of the pathway to follow location A, it can be seen that consistent interventions occur along the broad shared walkway. Many of these interventions experienced occur due to inaccurate expectations for pedestrian movement. During early tests, the autonomous vehicles utilized traditional tracking methods such as the Kalman Filter but fell short while attempting to negotiate with pedestrians and determine the correct right-of-way. As an example, a pedestrian waiting to cross the road should be given the right-of-way at crosswalks; however, if a tracking methodology based on traditional methods is applied, the pedestrian will be treated as stationary and the correct right-of-way may not be enforced. With this notion in mind, the surroundings, context and bodily language becomes highly relevant. For the same case, if the stationary human subject standing by the side of the road is looking down, it is highly unlikely that he or she will cross the road. In contrast, if the same subject is looking directly at the vehicle

while standing close to the road, it is likely that he or she is preparing himself or herself to cross and the autonomous vehicle should slow down and stop.

In any of the cases described, safety guarantees are needed. Statistically speaking, even though a pedestrian looking down on the side of the road is very unlikely to cross the road, the probability that he or she will step off the curb still exists. For this reason, pedestrian intent may be used as a prior but any expectations should be updated dynamically based on real time information.

Another pattern captured in the intervention map introduced involves road intersections. Road intersections in particular can encompass multiple intricacies that encapsulate some of the patterns described such as pedestrian intent but also introduce additional latent dynamics. An example of this complexity can be illustrated by location C from Figure 5.1 with its corresponding image shown in Figure 5.3. This four-way intersection is one of the busiest intersections on campus protected by stop signs. During the morning mail delivery runs, a high volume of vehicles is constantly experienced due to regular commuter traffic, and as such, one of the biggest challenges involves obeying the right-of-way and analyzing pedestrian behavior. In the state of California, drivers are required to perform a three-second stop at stop signs before proceeding; however, in the presence of traffic, the right-of-way is enforced based on a first-in-first-out order. Although the correct right-of-way can be determined based on a queuing mechanism for intersections of this type, additional attention must be paid to human drivers. Even with the correct right-of-way determined by updating an internal queue, other drivers' behaviors can be stochastic. In many cases observed, other drivers were unable to determine the correct queuing order based on the DMV right-of-way rule which often resulted in confusion among human drivers. This implies that a robust planning strategy must be capable of adjusting dynamically for human error—even if an autonomous vehicle is capable of generating the correct plans 100% of the time.



(a) Camera View of intersection.



(b) Point Cloud view of intersection

**Figure 5.3:** Active traffic in Gilman Dr. and Voigt Dr. Intersection

### 5.3 Moving Away from High Definition maps

In the last section, a number of shortcomings were discussed while considering pedestrian and human driver behavior. Many of the scenarios correspond to scene specific challenges that, if solved, would allow the vehicle to significantly improve based on the UMAP system. Nevertheless, there are additional considerations to keep in mind when autonomous vehicle scalability and generalizability is in question.

The software architecture introduced in Chapter 3 utilizes dense point clouds for vehicle pose localization and vector maps for path planning and tracking that are essential for autonomous vehicle operation. One drawback of these types of maps also known as HD maps is robustness, scalability and generalizability. Without having a prior point cloud map and manually annotated trajectories on the map, a vehicle cannot navigate through unseen environments. In a similar way, if the software modules utilize outdated HD maps, detrimental errors are likely to occur. To better illustrate this, Figure 5.1's construction sites will be used as a reference and multiple cases will be discussed.

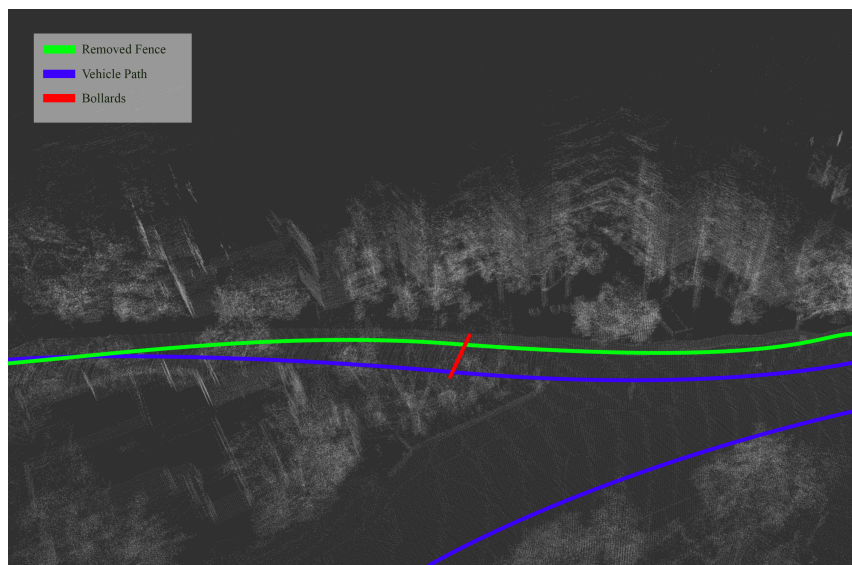
In the intervention map introduced in Figure 5.1, a number of construction sites can be observed around the time the mail delivery project started on campus. Many of these rapidly changed or shifted. In Figure 5.4, one of the autonomous vehicles is shown performing mail

delivery across from a construction site shown in Figure 5.1's location B.



**Figure 5.4:** Active construction across Warren college mailing center.

At the time the vehicle started performing mail delivery along this route, construction was quite active, and as a result, the point cloud and vector maps were built with many of these fast-changing features encoded. While the construction changed rapidly over time, the localization system introduced in Section 3.2 remained robust to many of these changes even after the fences shown in Figure 5.4 were entirely removed. The point cloud representation is also shown in Figure 5.5.



**Figure 5.5:** Point Cloud Representation of construction fence and route from Figure 5.2a.

These results are highly encouraging but there are other factors to consider such as dynamic point cloud updating, vector map updates, scalability and unseen environments. Although the localization module remained invariant to many of the changes due to construction, localization robustness can be impacted if most of the prominent features are lost. The first case to consider involves Figures 5.4 and 5.5; even though the construction fence moved over time and was eventually removed, the buildings around the walkway remained in place. Since the localization methodology is based on optimal grid-based Gaussian distributions, the vehicle pose is estimated based on the optimal LiDAR scan alignment probabilities and is robust to small map changes. The tolerance to small errors is certainly a good characteristic of robust localization but these errors should be accounted for through progressive map updates or pose and estimation related failures are likely to occur and can be detrimental to planning and safety. Minor localization related errors were experienced during testing and are included as part of Figure 5.1's intervention map.

The second case to consider involves vector map updates. For context, the lane definitions for the four-way intersection introduced shown in Figure 5.3 and Figure 5.1's location C changed



multiple times in a matter of weeks due to construction. Even with partially automated updates, if these are not updated at the time new lane definitions or changes take place, the vehicle will operate with outdated information and potentially cause unsafe scenarios. Some examples of this include lane repainting and shifting, and turn-lane logic changes.

This leads to the third case in which HD maps experience troublesome scenarios: unseen environments and scalability. Without an up-to-date point cloud map or previously annotated vector maps, the vehicle will be unable to localize itself and identify where it should and should not drive. Unseen environments without annotations imply that an HD map does not exist, and as a result, a self-driving vehicle that requires these maps cannot operate. With these HD map methodologies in mind, it becomes apparent that solving the general autonomous driving that would allow any car to navigate anywhere autonomously requires real-time updates when the environment and logic changes, and a mechanism to handle the the size of all of these maps. In the cases that involve smaller scale deployment such as micro-transit applications, the bottlenecks experienced from HD maps can be addressed with significant manual work for maintenance. However, as the scale of the problem increases, it becomes challenging to keep track of the changes being performed to logic and the environment without manual annotation in the loop. For larger scale deployments, alternative methodologies need to be considered that can fulfill the requirements of scalability, adaptability and robustness. In order for a self-driving car to scale and navigate long distances, it must be capable of understanding its relative location without requiring highly detailed maps such as HD maps. At the same time, the vehicle must be capable of adjusting any prior expectations on the environment such as road and lane definitions which ultimately influences how robust the system is. These are among some of the topics that have motivated this work. Future research directions that align with these shortcomings will be discussed in the next section.

## 5.4 Future Work

While there are constant engineering developments being performed such as robust sensor fusion and obstacle avoidance for the autonomous vehicle project introduced in this study, the unsolved problems are research oriented and involve pedestrian and vehicle intent, as well as scalability and generalizability.

With the introduction of pedestrian and vehicle intent models, improvements can be performed to the perception stack that can ultimately facilitate the interaction between an autonomous vehicle and other agents. A combination of deep learning and traditional filtering methods can be used to explore solutions for pedestrian and vehicle intent. For pedestrian intent, deep learning methods such as OpenPose [CHS<sup>+</sup>18] can be used as additional features to better interpret pedestrian behavior. With additional behavioral models based on Conditional Random Field methods [NHD<sup>+</sup>19] and Generative Adversarial Neural network methods [GJFF<sup>+</sup>18] [FDSF18] [AHP19] [SKS<sup>+</sup>18] [KSMM<sup>+</sup>19], an abstraction can be developed to learn and capture the different types of behaviors expected and assign probabilities on how likely a pedestrian is to cross a road.

To analyze the behavior of other drivers, similar deep learning and traditional computer vision methods can be used to determine if a driver is using his or her turn signals, emergency lights or braking. For road intersections, it is often important to estimate the direction of travel for each vehicle. This can be performed by analyzing turn signals [CHX<sup>+</sup>17], but also by the orientation of the other vehicles and even tire angles. Since these are heavily dependent on perception robustness, sensor fusion and tracking is inherently important.

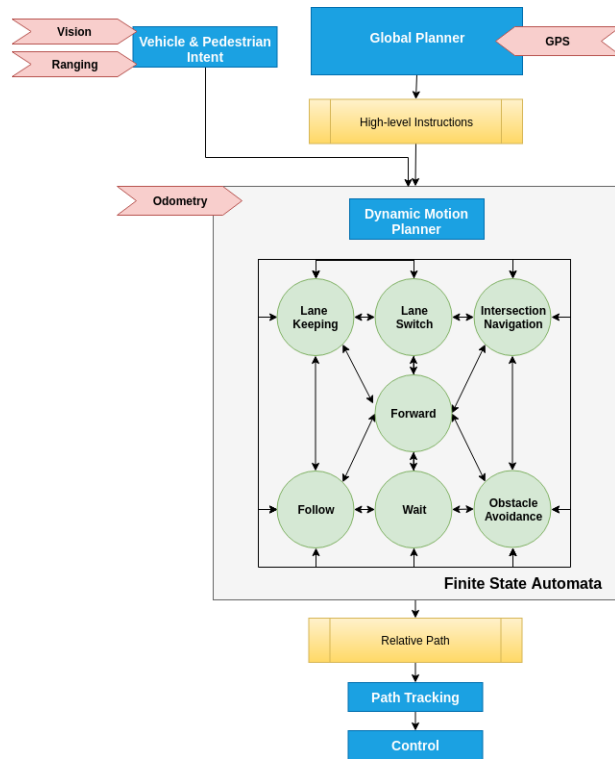
An equally important topic in self-driving technology involves scalability and generalizability as previously introduced in the last section. With continuous maintenance and constant updates, HD maps are ideal for small scale applications such as mail delivery or micro-transit. However, as the scale of the intended application drastically increases, it becomes troublesome to keep track of all of the changes that occur and to update the maps in real-time. To address these



shortcomings, a new planning strategy that consists of a high level global planner and a dynamic motion planner may need to be adopted. In this planning strategy, a high level global planner is introduced to generate a high level set of instructions that describe how long does a vehicle have to navigate until the next intersection and whether it should make a left turn, right turn or continue driving straight. With these high level instructions, a motion planner must handle immediate actions such as dynamic path generation based on real-time information for lane changing, lane keeping, obstacle avoidance, intersection logic and parking. Figure 5.6 corresponds to a revised architecture with the proposed changes that incorporate the new planners.

With these high-level dynamic planners, an alternative approach for navigation is taken that removes the dependency of dense point clouds and HD maps by working on a local frame of reference and estimating road features and path generation dynamically. In the methods discussed in Chapter 3, the localization method as shown in Figure 3.1 utilizes dense point cloud information to determine the location of the vehicle over time. This position information is then used during planning for obstacle detection and avoidance to update a previously defined reference path.

On the other hand, in order to move away from dense point clouds and avoid the overhead associated with manual path annotations and road features, the planner must be capable of recognizing lanes, intersections, crosswalks, traffic signs and lights, as well as estimating its relative location within a local frame of reference. This methodology follows a similar approach as human drivers in which given a set of high level instructions, local planning is performed dynamically and in real-time. While this method focuses on generalizing autonomous driving techniques by removing the inter-dependencies on HD maps, its overall robustness relies on the capabilities of perception and motion planning: lane detection, dynamic path generation, traffic sign recognition, and feature extraction. This proposes a new set of research topics to address for future work.



**Figure 5.6:** Global Planner and Dynamic Motion Planner.

With the autonomous vehicle platforms introduced in Chapter 2 and their corresponding software architectures in Chapter 3, applications for mail delivery were explored. Even though there are active developments on robust perception and fusion, dynamic point cloud map updating, obstacle avoidance, and pedestrian and vehicle intent, the architectures introduced have shown to be ideal for small scale environments such as campus scenarios. With the introduction of the UAV system in Chapter 4, the development cycles can be tracked over time to better understand how system performance improves as a function of time and where most of the challenging scenarios take place. For applications that require large-scale deployments, other techniques are needed such as the architecture introduced in Figure 5.6 that can generalize for fast-changing environments.

# Bibliography

- [AHP19] Javad Amirian, Jean-Bernard Hayet, and Julien Pettre. Social ways: Learning multi-modal distributions of pedestrian trajectories with gans, 2019.
- [CHS<sup>+</sup>18] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: Realtime multi-person 2d pose estimation using part affinity fields. *CoRR*, abs/1812.08008, 2018.
- [CHX<sup>+</sup>17] Long Chen, Xuemin Hu, Tong Xu, Hulin Kuang, and Qingquan Li. Turn signal detection during nighttime by cnn detector and perceptual hashing tracking. *IEEE Transactions on Intelligent Transportation Systems*, PP:1–12, 04 2017.
- [Con92] R. Craig Conlter. Implementation of the pure pursuit path tracking algorithm, 1992.
- [Dic07] Ernst D. Dickmanns. *Dynamic Vision for Perception and Control of Motion*. Springer, London, 2007.
- [DTT<sup>+</sup>17] Hatem Darweesh, Eijiro Takeuchi, Kazuya Takeda, Yoshiki Ninomiya, Adi Sujiwo, Y. Morales, Naoki Akai, Tetsuo Tomizawa, and Shinpei Kato. Open source integrated planner for autonomous navigation in highly dynamic environments. *Journal of Robotics and Mechatronics*, 29:668–684, 08 2017.
- [FDSF18] Tharindu Fernando, Simon Denman, Sridha Sridharan, and Clinton Fookes. Gd-gan: Generative adversarial networks for trajectory prediction and group detection in crowds, 2018.
- [GJFF<sup>+</sup>18] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks, 2018.
- [HKLS12] Jaehyun Han, Dongchul Kim, Minchae Lee, and Myoung-ho Sunwoo. Enhanced road boundary and obstacle detection using a downward-looking lidar sensor. *IEEE Transactions on Vehicular Technology - IEEE TRANS VEH TECHNOL*, 61:971–985, 03 2012.

- [HZ00] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, USA, 2000.
- [JPKA95] Todd Jochem, Dean Pomerleau, B. Sarath Chandra Kumar, and J. Scott Armstrong. Pans: a portable navigation platform. *Proceedings of the Intelligent Vehicles '95 Symposium*, pages 107–112, 1995.
- [KRD08] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Incremental smoothing and mapping. *IEEE Trans. on Robotics (TRO)*, 24(6):1365–1378, December 2008.
- [KSMM<sup>+</sup>19] Vineet Kosaraju, Amir Sadeghian, Roberto Martín-Martín, Ian Reid, S. Hamid Rezatofighi, and Silvio Savarese. Social-bigat: Multimodal trajectory forecasting using bicycle-gan and graph attention networks, 2019.
- [KTI<sup>+</sup>15] shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, and Kazuya Takeda. An open approach to autonomous vehicles. *IEEE Micro*, vol.48, 11 2015.
- [Lei86] Robert D Leighty. Darpa alv (autonomous land vehicle) summary. Technical report, ARMY ENGINEER TOPOGRAPHIC LABS FORT BELVOIR VA, 1986.
- [LMB<sup>+</sup>14] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollár. Microsoft coco: Common objects in context, 2014.
- [LMNF09] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. Eppn: An accurate o(n) solution to the pnp problem. *International Journal of Computer Vision*, 81, 02 2009.
- [MAFK16] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. *CoRR*, abs/1612.00496, 2016.
- [Mag09] Martin Magnusson. *The Three-Dimensional Normal-Distributions Transform — an Efficient Representation for Registration, Surface Analysis, and Loop Detection*. PhD thesis, 12 2009.
- [MMT15] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31(5):1147–1163, Oct 2015.
- [MSM<sup>+</sup>19] Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. Xnect: Real-time multi-person 3d human pose estimation with a single rgb camera, 2019.
- [NHD<sup>+</sup>19] Satyajit Neogi, Michael Hoy, Kang Dang, Hang Yu, and Justin Dauwels. Context model for pedestrian intention prediction using factored latent-dynamic conditional random fields, 2019.

- [PLH<sup>+</sup>19] David Paz, Po-Jung Lai, Sumukha Harish, Hengyuan Zhang, Nathan Chan, Chun Hu, Sumit Binnani, and Henrik Christensen. Lessons learned from deploying autonomous vehicles at UC San Diego. In *Field and Service Robotics*, Tokyo, JP, August 2019.
- [Pom95] D. Pomerleau. Ralph: rapidly adapting lateral position handler. In *Proceedings of the Intelligent Vehicles '95. Symposium*, pages 506–511, Sep. 1995.
- [QCG<sup>+</sup>09] Morgan Quigley, Ken Conley, Brian P. 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*, 2009.
- [QSMG16] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.
- [Rac17] Arya S. Abdul Rachman. 3d-lidar multi object tracking for autonomous driving: Multi-target detection and tracking under urban road uncertainties. 2017.
- [RC11] Radu Bogdan Rusu and Steve Cousins. 3D is here: Point Cloud Library (PCL). In *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.
- [RF18] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.
- [RHGS15] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2015.
- [SKS<sup>+</sup>18] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, S. Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints, 2018.
- [TKLD19] Wai Tun, Sangho Kim, Jae-Woo Lee, and Hatem Darweesh. Open-source tool of vector map for path planning in autoware autonomous driving software. pages 1–3, 02 2019.
- [TMD<sup>+</sup>06] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, Charles Markey, Carlo Rummel, Joe van Niekerk, Eric Jensen, Philippe Alessandrini, Gary Bradski, Bob Davies, Scott Ettinger, Adrian Kaehler, Ara Nefian, and Pamela Mahoney. Stanley: The robot that won the darpa grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006.

- [WC17] Ruffin White and Henrik Christensen. *ROS and Docker*, pages 285–307. Springer International Publishing, Cham, 2017.
- [WC18] S. Wang and H. I. Christensen. Tritonbot: First lessons learned from deployment of a long-term autonomy tour guide robot. In *RoMan*, Nanjing, China, August 2018. IEEE/RSJ.
- [XAJ17] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. *CoRR*, abs/1711.10871, 2017.