Reliable Navigation for Autonomous Vehicles in Connected Vehicle Environments by using Multi-agent Sensor Fusion

Suryansh Saxena

CMU-RI-TR-17-45

August 2017

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee:
Stephen F. Smith
John M. Dolan
Isaac K. Isukapati
Hsu-Chieh Hu

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Robotics.

Abstract

The age of intelligent vehicles is here, and with the rise in autonomous navigation we face a new challenge in finding ways to increase the safety and reliability of navigation for autonomous vehicles in dynamically changing environments.

To date, most methodologies in autonomous navigation mainly rely on the data feed from local sensors. However, erroneous measurements in sensor data might reduce the overall safety and confidence in navigation. These errors may be persistent and/or temporal, and give rise to false predictions made by the autonomous agents, usually classified as either a false positive or a false negative. Though false positives may influence navigation safety in an indirect manner, false negatives directly and significantly affect it.

In this thesis, we explore a methodology for sharing and fusing sensor data of close-proximity autonomous or intelligent vehicles to identify and minimize false negatives in local environment interpretation. For each agent, the methodology compares the local SLAM maps with maps generated by other close proximity agents to identify false negatives in local interpretation.

As per the methodology, each autonomous agent simultaneously localizes and maps its local environment. This map, in turn, is encoded into a low-resolution message and shared over the DSRC communication protocol. Next, the agents distributively fuse this information together to build world interpretation. Each agent then statistically analyses its own interpretation with respect to the world interpretation for the common regions of interest. The proposed statistical algorithm outputs the measure of similarity between local and world interpretations and identifies false negatives (if any) for the local agent. This measure, in turn, can be used to inform the agents to change their kinematic behavior in order to account for the errors in local interpretation. Finally, each agent records the measure and instances of erroneous interpretations, which over a period of time helps analysis and quantifies the sensor health.

We evaluate the qualitative and quantitative efficacy of our proposed methodology and algorithms using a widely accepted traffic simulator called SUMO in scenarios that have a high accident frequency. We use ROS in tandem with SUMO to simulate and quantify the behaviour of autonomous vehicles. Finally, we also propose a communication architecture to enforce the proposed methodology in the real world and quantify the efficacy of the communication architecture by conducting latency testing.

Acknowledgments

I would first like to thank my advisers Dr. Stephen F. Smith, Dr. John M. Dolan, and Dr. Isaac K. Isukapati who have always been around for me. I would also like to thank Hsu-Chieh Hu for his invaluable feedback.

Finally, I must express my very profound gratitude to my parents and brother, family and friends for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis.

This accomplishment would not have been possible without them. Thank you.

Suryansh Saxena
(Author)

Contents

List of Figures

List of Tables

1 Introduction

The age of intelligent vehicles is here, and with the rise in autonomous navigation, we face a new challenges in finding ways to increase the safety and reliability of navigation for autonomous vehicles in dynamically changing environments. To date, much of autonomous driving is dependent upon self-localization and mapping. There are several instances where the autonomy fails due to a bad sensor reading which may be a manifestation of complex environmental conditions or sensor drift etc., such failures can be addressed by collaborative autonomy that can be facilitated by the connected vehicle environments.

Intelligent Transportation Systems (ITS) strive to integrate technologies that increase the safety and efficiency of the transportation system. In that regard, wireless vehicular networks such as **Dedicated Short Range Communications** (DSRC) can play an instrumental role in creating and integrating various safety and mobility ITS applications. DSRC is a two-way short-to-medium-range wireless communications capability that permits very high data transmission critical in communications-based active safety applications The DSRC protocol was instrumented with specific consideration for implementing low latencies and transmission reliability for vehicle-to-vehicle communication (V2V) and vehicle-to-infrastructure communication (V2I), this has resulted in various applications like collision warning, traffic congestion warnings and signal phase coordination etc., improving the overall safety and mobility of daily road navigation. Figure 1 represents this connected environment.

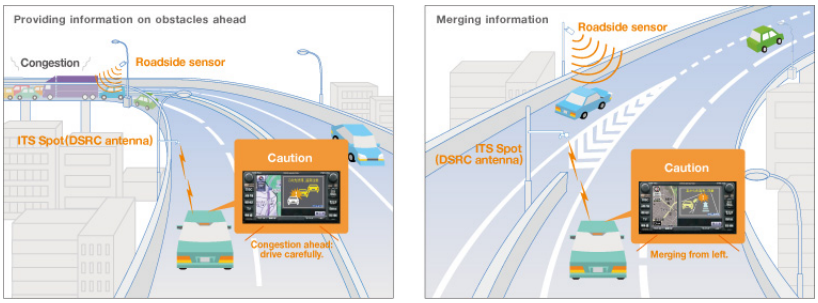

Figure 1: Connected Vehicle Environment Representation

Think of DSRC as a cyber-physical system that behaves like a sensor data sharing framework, and the data can be used for making intelligent decisions and increasing the efficiency of autonomous navigation. However, the areas of autonomous navigation and connected vehicle environment protocols have some major challenges. We address some of these issues in this thesis by; developing an architecture for connecting autonomous vehicles into the ITS grid and leveraging the proposed architecture to connected multiple autonomous vehicles and fusing their data for enhancing the safety of autonomous navigation.

1.1 Motivation : Addressing the Gap

Autonomous navigation has some critical challenges with regard to reliability and safety of navigation. One of the critical challenges in autonomous navigation is major dependence on local sensor feed and the accurate interpretation of the sensing data. A small glitch in sensor data can lead to catastrophic results on a road network. The majority of the autonomous car accidents reported today are due to false negatives, which manifest from either bad sensor data or failure of the navigation algorithms to analyze the sensor accurately [1].

For example one of the Tesla cars crashed in a trailer of the big-rig truck due to bad sensor interpretation [2]. The camera on the Tesla car was unable to identify the white trailer truck due to environmental lighting. The reason for the crash was not a bad sensor rather bad sensing condition which manifested a false negative. Another example is Uber's self-driving car, that failed to recognize at least six red lights in San Francisco during a testing on February 2017 [3]. The self-driving vehicle was unable to detect several red lights and stop signs due to a faulty sensor. Sensors drift over time and therefore it is difficult to prevent bad world interpretation in realtime driving.

The condition arises due to the heavy dependence of autonomy on sensing. Sensors may be prone to several sources of error; therefore, just adding another sensing modality through state-of-the-art connected vehicles (via DSRC, or LTE) would not be able to address the same issues as they are also vulnerable to drift and environmental noise. The data shared over the state-of-the-art connected vehicle technology consist mainly of echo locations of other connected vehicles which is not sufficient to identify the various errors in interpreting the environment. One way to solve the problem is by building better and accurate sensors and algorithms that would reduce the probability of these false negatives immensely. However, this approach is quite expensive both in terms of time of development and overall cost of the system.

Much work has been done in developing algorithms that try to interpret and analyze the environment to increases navigation safety [4] [5] [6]. However, much of these algorithms are not efficient in detecting false negatives because, in order to interpret the environment, these algorithms employ sorting and/or averaging techniques, over the consequent sensor feeds. Therefore, the output of these algorithms ends up rejecting outliers rather than incorporating them. In order to capture false negatives, an autonomous agent must be able to identify the outliers in a sensor feed and map the outliers to either noise or error in interpretation.

On the other hand, there are several challenges with the connected vehicle protocols as well. With the DSRC protocol, one of the issues is that of low penetration within the environment. It is estimated that all cars in the U.S will have DSRC radios embedded by 2060 provided the NHTSA mandate is accepted. Another critical challenge for the DSRC protocol is the low bandwidth, which restricts the agent to share low-resolution data. While other communication protocols like 5G LTE or WiFi have good bandwidth and ever lower latencies, challenges with these protocols are reliability of connectivity

and data transmission, which is critical for any connected environment.

We address some of these challenges by proposing a new sensor sharing methodology for connected autonomous and intelligent vehicles, that shares their location, speed, and echo along with a map encoded with critical information about the obstacles detected. We propose a methodology wherein multiple agents can receive complimentary sensor information and statistically compare their local interpretation with the maps shared by other close-proximity intelligent or autonomous vehicles. Since we are using the connected vehicle framework to share this sensor information, we also address some of the issues with state-of-the-art connected vehicle communication protocols and propose an architecture to enable a wide range of data-sharing, high-bandwidth, and low-latency communication capabilities.

1.2 Methodology Outline

In this thesis, we investigate and address the issues listed in section 1.1, by designing and developing a methodology for autonomous and intelligent vehicles to share and fuse their sensor data to enhance navigation safety. The major focus of this thesis will be to investigate, develop and test the said methodology for identifying and minimizing the false negatives. We also investigate and design an architecture for connecting vehicles to share a wide range of sensor information to enhance intelligence in the traffic grid.

The proposed methodology will enable close-proximity autonomous and intelligent vehicles to share SLAM (view) data with each other. To estimate the ground truth, the data from different agents are fused, in order to compute the world (majority) consensus. This fused consensus is then statistically compared with local interpretation of each agent to identify the outliers. It should be noted that, sensors like Radar, Camera, Lidar etc., mounted on the road-side (if any), will be considered as intelligent agents and will be able to share either data with the relevant autonomous agents through this methodology. The details of the methodology is highlighted below:

1. Sensor Sharing and Fusing Methodology :

    (a) The proposed architecture can be leveraged for enhancing the safety and mobility within the connected environment by exploring Sensor Sharing and Sensor Fusion Methodology. Sensor sharing is a concept we develop in the connected vehicle domain that will allow us to share critical environmental information between multiple agents. Sensor fusion on the other hand, dwells in the domain of robotics and statistical analysis developed for improving the perception of the environment by minimizing the number of false negatives, for autonomous vehicles.

    (b) The agents will be able to share sensor data, like route information, congestion, etc. using their local sensors, which otherwise they would have not been able to share via DSRC. One of the major applications here is tracking of non-connected vehicles by connected agents. These connected

agents then pass the tracking information into the ITS grid, via the proposed architecture, essentially connecting the non-connected vehicles.

(c) Each autonomous agent will be connected to other close-proximity agents and will share their local world interpretations. Each agent will analyze regions of interest in the incoming interpretations and then fuse their local sensor input with the data received. Agents will then statistically evaluate ground truth by comparison. This will enable the autonomous agents to identify false negatives within their local frame by comparison with global consensus.

2. Split-Level Architecture (DSRC $<->$ ROS $<->$ Cellular) :

(a) The Split-Level Architecture is a communication adapter for connecting vehicles. This architecture utilizes the low latency and reliability of the DSRC protocol for safety applications and high penetration and bandwidth of the cellular (4G/5G) LTE protocol for enhancing the mobility of the agents. The two communication protocols are also utilized as redundant modes of communication for avoiding DSRC dead-zones.

(b) The Robot Operating System (ROS) is used as a central processor of information and communicator to the two protocols. ROS is envisioned as a processing and communication interface that fulfills three main objectives:

i. (i) It interacts with the vehicle-level sensors and operational/control sub-systems. The wide acceptance of ROS in the robotics community makes it especially useful in case of autonomous vehicles; (ii) It allows open processing (encoding/decoding) of information from various sources; (iii) It can easily interface with DSRC and cellular radios.

(c) The proposed architecture is able to encode sensory information from different agents and push it into the grid.The proposed architecture is also able to push/pull information from multiple online-databases like Wazes to inform navigation and enhance mobility.

1.3 Thesis Outline

This thesis is organized as follows, In Chapter 2 we formally introduce the concept of connected vehicles and then highlight the safety message protocols embedded in DSRC that we will exploit for this work. In Chapter 3 we discuss the errors and issues with autonomous navigation in detail and propose a methodology to identify and minimize such errors. Next in Chapter 4 we detail the construction of a simulator for testing the methodology and propose two experiments to test the efficacy of the system. The results and observations from the experiments conducted have been highlighted in Chapter 5. Finally we address the issues with connected vehicle protocols and propose an architecture to tackle some of these issues in Chapter 6, and then conclude our work and provide a high-level description of future work in Chapter 7.

2 Connected Vehicle Environment

The fundamental premise of the connected vehicle environment lies in the power of wireless connectivity among vehicles (referred to as vehicle-to-vehicle or V2V communications), the infrastructure (vehicle-to-infrastructure or V2I communications), and mobile devices to bring about transformative changes in highway safety, mobility, and the environmental impacts of the transportation system.

Connected vehicles have the potential to transform the way we travel through the creation of a safe, interoperable wireless communications network, a system that includes vehicles, traffic signals, smartphones, and other devices. Multiple connected vehicle application development efforts and studies have demonstrated the potential for significant safety, mobility, and environmental benefits from V2I applications. The magnitude of the benefits depends on the level of technology deployment at the roadside, in vehicles, or within mobile devices.

2.1 Connected Infrastructure with DSRC

DSRC (Dedicated Short Range Communications) is a two-way short-to-medium-range wireless communications capability that permits very high data transmission critical in communications-based active safety applications. In Report and Order FCC-03-324, the Federal Communications Commission (FCC) allocated 75 MHz of spectrum in the 5.9 GHz band for use by Intelligent Transportations Systems (ITS) vehicle safety and mobility applications.

A set of IEEE standards have proposed a Wireless Access in Vehicular Environments (WAVE) system architecture which utilizes the DSRC band to offer a platform and infrastructure for connected vehicle systems. The message formats and requirements for WAVE services and applications have been defined by the Society of Automotive Engineers in the SAE J2735 message set standard [7] [8].The key functional attributes of DSRC are as follows:

1. Low latency: The delays involved in opening and closing a connection are very short on the order of 0.02 seconds
2. Limited interference: DSRC is very robust in the face of radio interference. Also, its short range (1000 m) limits the chance of interference from distant sources. Additionally, DSRC is protected by the Federal Communications Commission (FCC) for transportation applications.
3. Although purely commercial-convenience DSRC applications are welcome, transportation safety applications take precedence. Hence the protocol is built for safety.
4. Strong performance during adverse weather conditions.

There are two classes of devices in a WAVE system : The on-board unit (OBU) and roadside unit (RSU). They are equivalent to the mobile station (MS) and base station

(BS) in the cellular systems, respectively. There are two classes of communications enabled by the OBUs and RSUs: vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I). The OBU in a vehicle normally directly communicates with other OBUs within the radio coverage area. This direct V2V communication reduces the message latency and low latency is an essential requirement for safety applications, such as collision avoidance. It should be noted that an OBU is more likely to be embedded and connected with other electronic systems of the vehicle via in-vehicle networking, such as a controller area network (CAN) [9].

2.1.1 Safety Message Standards

The 2004 FCC ruling [10] specifies the DSRC will have six service channels and one control channel. The control channel is to be regularly monitored by all vehicles. Safety messages, whether originated by vehicles (OBU) or roadside transmitters (RSU), are to have priority over non-public safety communications. Therefore all safety messages are to be sent in the control channel.It should be noted that connected V2V safety applications are built around the SAE J2735 [11] Basic Safety Message (BSM) protocol.

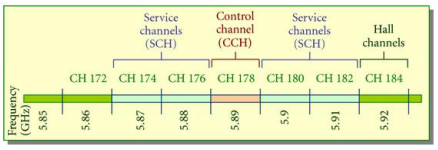

Figure 2: DSRC Protocol Channel Allocation

Basic Safety Message	
BSM Part 1 (frequency : 10x)	BSM Part 2 (frequency : 1x)
Mac ID (Type : String)	Mac ID (Type : String)
Message Number (Type : Float)	Message Number (Type : Float)
Vehicle Length (Type : Float)	Vehicle Length (Type : Float)
Vehicle Location (GPS) (Type : Tuple)	Vehicle Location (GPS) (Type : Tuple)
Vehicle Speed (Type : Float)	Vehicle Speed (Type : Float)
Vehicle Heading (Type : Float)	Vehicle Heading (Type : Float)
Vehicle Acceleration (Type : Float)	Vehicle Acceleration (Type : Float)
Vehicle Time-Stamp (Type : Float)	Vehicle Time-Stamp (Type : Float)
	Vehicle Object Distance (Type : List)

Table 1: Basic Safety Message Structure

Any DSRC unit can transmit or receive two types (Part 1 and Part 2) of BSM. BSM Part 1 contains the core data elements (vehicle size, position, speed, heading acceleration, brake system status)and can be transmitted with a latency of approximately 100 ms (milliseconds), while BSM Part 2 is an extension to the Part 1 message with some

additional data element(s) and is often referred to as a payload BSM. The frequency of the payload BSM depends on the data element being added to the BSM; however, for most cases, the frequency of the payload BSM is within 1000 ms (millisecond). In this thesis, we use the BSM (part 1 and 2) as the standard message for V2X communication; these are highlighted in Table 1. It should be noted that there are several optional data elements for the payload BSM that are optional and have not been highlighted here, please refer to the standard (2016) for a complete description [11].

In this thesis we are using part 1 of the basic safety message for sharing the local kinematic information about the vehicle, while each agent encodes its local surrounding environmental information into the payload BSM. We are using the Object Message Element for encoding this information. In Chapter 3 we will see how to encode and utilize the Object Message Type to encode environmental information using a LIDAR sensor and the ROS adapter.

3 Methodology

This thesis highlights a methodology for resolving the false negatives in sensing and environment perception. As highlighted in Chapter 1, to date, most methodologies in autonomous navigation mainly rely on a data feed from local sensors. However, erroneous measurements in sensor data might reduce the overall safety and confidence in navigation. These errors may be persistent due to failing sensor health, sensor drift, bad calibration, and/or temporary conditions which are mainly due to environmental and/or random effects. This gives rise to false predictions made by the autonomous agents, usually classified as either a false positive or a false negative.

It is cumbersome to address these errors in realtime while navigating because resolving some of them requires, 1) more data from the sensors, which in turn could still be faulty; and 2) intelligent filters that can handle false positives while trying to identify false negatives in a dynamically changing environment. Moreover, this problem does not only exist in autonomous driving but rather in almost any area of robotics and sensor application. One way of addressing these issues is by fusing data for a point of interest (false negative/positive) from multiple sensors or agents to find some level of ground truth in the different interpretations [12] [13] [14]. This gives rise to a collaborative autonomous behaviour between multiple agents and increases the reliability in autonomy.

Figure 3: Fleet of Google's Autonomous Vehicle

Increase in the number of autonomous vehicles will lead to more cases of false negatives/positives collectively. Each error can cause accidents and therefore it becomes increasing importance to enhance navigation safety for autonomous vehicles in dynamically changing environments. However, there exists a unique advantage; more agents on the roads we can employ a methodology for fusing sensor information between agents in close proximity to identify and reduce the number of false negatives/positive.

Therefore we explore and develop a methodology for sharing and fusing sensor data between multiple autonomous or intelligent vehicles that have overlapping views of the environment. This thesis focuses on identifying and minimizing the number of false negatives in sensing and interpretation. Before we propose our model, we make certain assumptions to ease some constraints :

- The road network and environment conditions are such that false positives to not have a significant impact on safety. Thus in this work we only identify and

minimize false negatives.

- The vehicles must have compatible sensors or algorithms, such that data from multiple agents can be fused without considering format or synchronization inconsistency (as this is not the focus of the thesis).

- And finally we assume that only a minority of autonomous vehicles have strong noise associated with their data, and therefore when fusing the data between multiple agents we are certain to have some level of accuracy in world interpretation from agents having sensors with low or acceptable noise. In this thesis, we do not try to address pathological cases like when data from all the agents are highly erroneous.

We propose a methodology that compares the local SLAM maps of each agent with maps generated by other close-proximity agents to identify false negatives in local interpretation. As per the methodology, each autonomous agent simultaneously localizes and maps its local environment. This map, in turn, is encoded into a low-resolution message and then shared over a communication protocol. In this thesis we take DSRC as our connected vehicle communicational protocol. Next the agents distributively fuse this information together to build a world interpretation. Each agent then statistically analyzes its own interpretation with respect to the world interpretation for the common regions of interest using a maximum standard deviation test.

The proposed statistical algorithm outputs the measure of similarity between local and world interpretations, and identify false negatives (if any) for the local agent and its interpretation. We correct or minimize these errors by updating the agent's kinematic behavior to avoid obstacles it could earlier not perceive. The proposed methodology not only identifies the erroneous reading but also provides a measure of sensor health by evaluating the instances of bad readings over a period of time. To understand the methodology we first highlight some critical sources of errors in autonomous navigation in section 3.1 and then provide the methodology details about the process flow and different components in section 3.2.

3.1 Errors in Autonomous Navigation

A false negative in the real world may manifest in the form of a missed obstacle, imperfect reading of the road lane, incorrect speed estimation of other vehicles etc.. It is trivial to understand that either of these cases is a serious safety concern. It should also be noted that often these kinds of errors are the more difficult, based on the local and environmental conditions, to correct in real world applications, as the interpretation is highly influenced by the local sensor and or local environmental conditions [15]. Based on the sensor health and noise model the number of instances of these errors may increase significantly, which would reduce the overall reliability of autonomous navigation.

Traditionally these issues have been addressed by online-sensor calibration routines

which try to correct for persistent errors [16] on the fly. These techniques usually depend upon more robust sensors like odometer to calibrate the others. However, there are some issues with these methodologies; one, since the calibration is happing in a dynamically changing environment, the reliability of correction reduces drastically in certain cases and the routine has to be re-run. Secondly, these routines have little or no ability to correct errors that manifest due to environmental or random effects. Since it is difficult to quantify the source of error in dynamic navigation, some of the techniques presented are not complete answers to false negatives. This makes false negatives difficult to detect. It is critical to identify and minimize the number of false negatives in the autonomous driving, as they significantly affect the safety of navigation. [17].

On the other hand, false positives are relatively easier to trace and may influence safety in only an indirect manner [18]. One obvious reason why they are easier to detect is that they show up as a significant reading in the sensor data or the local algorithmic interpretations [19]. Consider the example of an autonomous vehicle navigating on a highway. One case of false positive could be the vehicle estimating a non-existent obstacle at some distance. The initial reaction of the vehicle will be to either brake or change lanes in order to avoid a collision. This is not necessarily unsafe but it also results in safety issues if there are other vehicles traveling at high speeds that may collide with the autonomous vehicle as a result of braking or lane changing.

3.2 Correct False Negatives in Connected Environment

We observe from the previous section that false negatives strongly influence the safety and reliability of autonomous navigation and are also fairly undetectable through local interpretation. It should be noted that false negatives result mainly from bad sensors or local environmental conditions. However, we can address this issue by proposing multiple (at least two) good sensors that are in different orientations but scanning the same area that may be able to detect a point which in our local model was not observed. An example is represented in Figure 4, where sensor set 1 is unable to detect the obstacle due to a false negative, but it can be seen that the other sensors can observe the obstacle and can inform sensor set 1 with some measure of confidence.

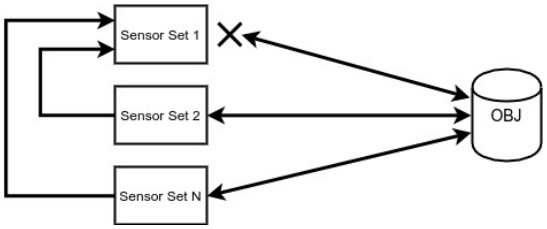

Figure 4: Representation of Multiple Sensors scanning a obstacle

We represent the sensors as sensor sets in Figure 4, because autonomous vehicles fuse data from multiple sensors like LIDAR, Radar, Camera etc., to interpret their local environment. Hence each sensor set in the figure represents one autonomous vehi-

cle. It should be highlighted again that these sensor sets may be oriented differently, as autonomous vehicles navigating in the real world will be at different locations and orientations. A real world example is illustrated in Figure 5, where the different autonomous vehicles are heading towards an orange obstacle while trying to avoid it.

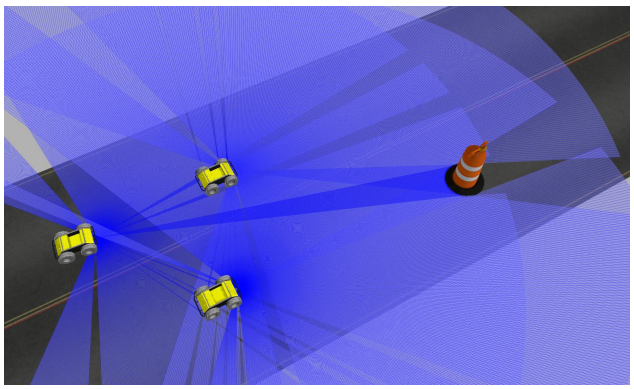

Figure 5: Autonomous Agents Navigating on Road while Avoiding Obstacles

A similar methodology can be implemented for a road network with autonomous and intelligent vehicles. With increasing number of such vehicles, we can say with some generality that soon there will be enough autonomous and intelligent vehicles on the road that in certain situations two or more such vehicles will have overlapping views of the world. This will allows us to fuse the sensor data between multiple autonomous vehicles in close proximity to identify and minimize the false negatives in their local sensing. It should be noted that the multiple agents will not be sharing ground truth values as the information sensed will always be noisy and therefore the algorithmic interpretations will be probabilistic. Hence each agent will get approximate estimations of the perceived environment and therefore must estimate the ground truth to some measure in order to enhance safety of navigation.

The entire methodology can be summarized by the following steps:

1) First an autonomous agent must understand its surrounding and its current location for the purpose of navigation. The agent uses local sensor systems like LIDAR and Camera to build an interpretation of the surrounding world. This is done by scanning the different obstacles and landmarks within the field of view (usually 360 degrees). Usually, a LIDAR returns a 360-degree obstacle point cloud which is a cartesian map of the points detected by the LIDAR. An agent must map a group of points to a particular obstacle. This process is referred to as data segmentation. Bad segmentation algorithms may result in false negatives or positives. Section 3.2.1 details an approach to data segmentation.

2) Once the agents have a local world interpretation they must then self-localize and map the obstacles they interpret. The agent estimates its own location along with the location of the obstacles using a SLAM or Simultaneous Localization

and Mapping algorithm. One popular SLAM algorithm is an Extended Kalman Filter SLAM which is highlighted in Section 3.2.2. The output of a EKF SLAM is a map that encodes the distance to all the perceivable obstacles and a measure of the accuracy in form of variance into an occupancy grid.

3) Next the agent parses the critical information from the occupancy grid and transforms the data into a low-resolution message. The message contains a Gaussian representation for each obstacle where the mean is the estimated distance to the obstacle and the variance is the uncertainty computed by the EKF SLAM. The low-resolution message is then sent over the connected vehicle infrastructure. These data are received by other agents, which can compare their local interpretation with the interpretations received over the communication channel.

4) Before the agents can compare their local interpretations with those of others, they first must identify the regions of overlap in their local world view and the view of the other agents, as a comparison can be made only in regions that are observable to the other agents. Each agent first transforms the different interpretations into its local coordinate frame via euclidean transformation. Then the agent must associate the different data points from each interpretation to its local frame. This data association is usually done using the K-nearest neighbors algorithm, where we cluster the data points based on mean distance and relative angles. The output lets the agent map its own local data points to points in the interpretations from the different agents. This has been highlighted in Section 3.2.3.

5) The data points in each cluster, except the data points from the local view, are fused with each other using a kernel mixture model. Each data point is basically a probability density function (PDF) with mean as the distance from the local sensor. In this thesis we assume all the PDF's to be unimodal and gaussian, hence we fuse the different data points using the normally weighted addition principle highlighted in section 3.2.3.

6) Next the fused data are then compared to the local agent interpretations to identify the error in reading. The fused data give a world interpretation from multiple agents and are most likely close to the ground truth. Therefore the deviation of the local data with respect to the fused data will provide a measure to identify the false negatives in local reading. There are various possible cases :

    a) If the agent cannot associate a data point in the fused map to a local data point, then most likely the local agent has either missed that data point or estimated it somewhere far from the ground truth. This is a false negative, so in this case the agent accepts the values proposed by the fused map.

    b) If the agent cannot associate its own data point with any data point in the fused map, then most likely it is an erroneous interpretation in the local map. This is the case when the sensor reads an obstacle, however, has a bad reading of its location. In this case, the agent still keeps its local interpretation, as it is better to overestimate rather than underestimate in order to be cautious.

c) If the agent is able to associate its own data point to a data point in the fused map, then it should be able to analyze the similarity between its own interpretation with respect to the fused interpretation. This analysis will estimate whether the local reading is a false negative or not. This point is not very intuitive. It should be noted errors that overestimate or underestimate a measurement are also quantified as false negatives, which may include examples like underestimating the velocity of a vehicle etc.. Based on the situation these errors may also be qualified as false positives.

   Hence in order to safely navigate in these situations, the agent must quantify whether the reading is underestimated or overestimated, which can be done by statistically comparing the density functions of the local and fused data points. We analyze the measure of similarity between the fused and local data points using the Maximum Deviation Test, highlighted in Section 3.2.4. The output of the test provides a score of similarity between the two points.

   A high score estimates the local and fused interpretations to be quite similar, in which case the agent adheres to its local interpretation. However, if the score is low, then we compare the confidence of the two readings. If the local variance is lower then that of the fused data, it means we are overestimating the obstacle, and since safety is our main consideration, the agent accepts the local model. However, in the inverse case, the agent underestimates, and therefore the agent accepts the fused interpretation.

This methodology allows each agent with some level of confidence to find false negatives and positives in the local world interpretation and hence use the knowledge to update its instantaneous kinematics models to enhance safety and reliability of its navigation. It should be highlighted that for this methodology to work, we at least need three agents in close proximity to each other.

3.2.1 Local Sensing and Data Segmentation or Clustering

Most autonomous navigation methodologies depend upon data feed from sensors. Today, a modern autonomous vehicle includes sensors like GPS, IMU, Odometer, Radar, Camera and LIDAR sensor (etc.) to help the vehicle localize and navigate safely in an obstacle environment. These sensors allow the autonomous vehicles to localize and build a map of obstacles in the environment. But before the agent builds a map, it must first interpret the data through the sensor. Usually sensors like LIDAR, Radar and Camera collectively generate a point cloud map of the environment such as the one represented in Figure 6.

It can be seen that each obstacle is made up of a certain number of data points. These data points must be associated in order to give a complete representation of the obstacle. There are several methodologies that can cluster data in the point cloud-based on computer vision, geometric analysis, motion patterns etc. [20] [21]. In this thesis we

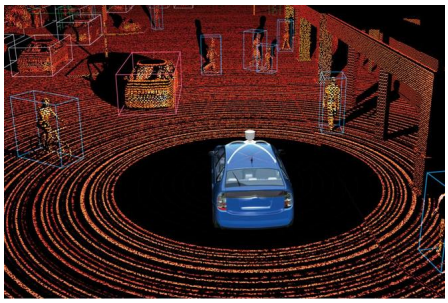

Figure 6: LIDAR point cloud map

simulate a LIDAR sensor to generate a point cloud and we cluster the data based on the sequential compatible nearest neighbour (SCNN) approach [22]. The SCNN associates a particular landmark to a batch of data (point cloud) in a greedy fashion. Since in the thesis we are assuming that all obstacles are vehicles, the vehicles are far apart and some vehicles are connected via DSRC, the SCNN method is quite efficient in clustering the data points.The pseudocode for the implementation of the SCNN algorithm is shown below:

Data: The input is the point cloud data from the LIDAR and Locations of Connected Vehicles
Result: Cluster data points into obstacles
initialization; Set E with all the data points.;
while $E! = 0$ **do**
  Calculate likelihood of data point $e_t = i, j$ using nearest neighbour test, where $i \in measurment$ and $j \in features.$;
  if *Certain combination that maximizes the likelihood function* (f_{ij}) **then**
    E.pop(e_t) ;
  else
    continue;
  end
end

Algorithm 1: SCNN implementation

3.2.2 EKF SLAM

Once we have the data clustered for each obstacle in the agent's sensor map we next translate this map into an occupancy gird that encodes the estimated distances to the obstacles and the measure of confidence associated with them, while localizing itself with respect to the obstacles. In robotics, this process is referred to as Simultaneous Localization and Mapping (SLAM) and one of the popular SLAM approaches utilizes the Extended Kalman Filter [23]. It should be noted that EKF SLAM is a Bayesian technique for estimating the location of the obstacles based on a certain motion model and measurement model. A simple process to follow is highlighted in figure 7.

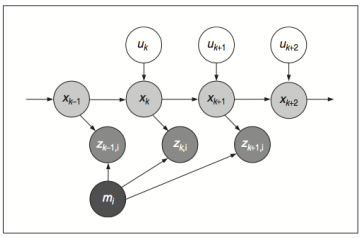

Figure 7: EKF SLAM update cycle

Since it is a Bayesian technique, each data point is independent of the next. We initialize the EKF with a random prediction state, which is then updated by the motion model u_k, which can be estimated using techniques like optical flow etc.. In this thesis we are able to simply parse the motion model for each vehicle using a simulator and a local noise model. This process is called the prediction step. Next the prediction state of the obstacles is corrected using measurements from local the LIDAR sensor m_i. At the end of each step the EKF SLAM will localize the obstacle with respect to the agent with some measure of accuracy or confidence. We implement the EKF SLAM filter as follows:

Assume the state of an obstacle at any given time t, is μ_{t-1}. We find the motion model u_t, that is, the motion incurred in that time step and measurement mode z_t feedback for that time instant. Then we first predict the update state of the object using the motion model :

$$\bar{\mu_t} = g(u_t, \mu_{t-1})$$

Next, we find the prediction variance (Σ) or measure of confidence for that particular state to be:

$$\bar{\Sigma_t} = G_t \Sigma_{t-1} G_t^T + R_t$$

We find the Kalman gain using :

$$K_t = \bar{\Sigma_t} H_t^T (H_t \bar{\Sigma_t} H_t^T + Q_t)^{-1}$$

Finally, we update the prediction using the measurement model to get the final state estimate and the associated confidence.

$$\mu_t = \bar{\mu_t} + K_t(z_t - h(\bar{\mu_t})), \Sigma_t = (I - K_t H_t)\bar{\Sigma_t}$$

After each step, we get an updated estimation of the object state and the associated accuracy. In this thesis the object will be the obstacle, the state will be the distance from the sensing agent and accuracy will be the variance associated with the distance.

3.2.3 Sensor Fusion with other Connected Agents

The agents record the estimated location of each obstacle and the associated variance after every EKF cycle; the record is called a frame. After a certain number of frames,

the agent encodes the critical information with respect to the different obstacle's distances and their respective variances into a super frame and sends it over a communication channel. In this thesis, the agents share super frames at a frequency of 1 Hz over the DSRC framework. The data received by a particular agent are critical as they inform the agent about the environment interpretation of other close-proximity vehicles. Since these vehicles are in close proximity, they are bound to have some region of overlap in environment view, as represented in Figure 8

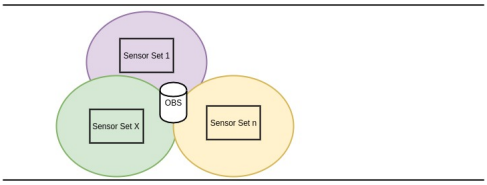

Figure 8: Representation of Overlap of Sensor regions in Multi-agent Navigation

Thus, before we can find the geometric regions of overlap between two or more agents, we must first transform all the data points received over the DSRC framework into one frame of reference, as different agents may have different orientations. We must preserve both the angle and the distance of each data point with its respective agent. Hence we transform the data points received from different agents into the coordinate frame of the local agent by using the euclidean translations and rotation transformation. We implement a translation transform as:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where we translate points $< x, y, z >$ to $< x', y', z' >$ by adding a vector $< p, q, r >$. We implement a rotation transformation in the XY plane by :

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} cos(a) & -sin(a) & 0 & 0 \\ sin(a) & cos(a) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

where a is the relative angle of rotation. In this thesis we translate the data point contributed by interpretations of different agents by the relative distance between the local and proximity agent. Next the data points are now passed through a rotation transform where the angle of rotation is the relative angle between the local and proximity agent. Finally the views between the local and proximity agents are transformed into one view from the perspective of the local agent. A representation from the simulator can be seen in figure 9. The bold RED point is the local agent while the other points are either tracked autonomous agents (bold blue) or data points (obstacles) transformed.

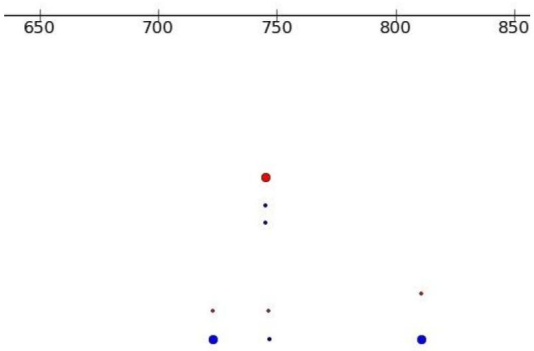

Figure 9: Transformation of the data points in to the local agent coordinate frame

Once the data points have been transformed, next we must fuse each data point from the different interpretations from proximity agents to the data points from the local vehicle. Though the different agents may view the same obstacle, errors in prediction and internal motion and measurement noise models, maps the data points for the same object at two different coordinates, as a result of which we must use a data association algorithm.

Even though there exist errors in each interpretation for an obstacle, we should still find data associated with a particular obstacle within some distance and angle vicinity. Therefore we use the k-nearest neighbour algorithm to associate the data points from multiple interpretations to the local map. We search over the distances between the different data points, and also search over the relative angles between the different data points. We study the impact of the two approaches in Chapter 5. We implement the K-nearest neighbour algorithm as follows:

Classify (X,Y,x) // X : training data, Y : class labels of X, x: unknown sample ;
for *i = 1 to m* **do**
| Compute Parameter d(X_i,x)
end
Compute set I containing indices for the k smallest parameters d(X_i,x). ;
return majority label for $\{Y_i$ where $i \in I\}$;

Algorithm 2: k-NN implementation

Finally data points that are contributed from other agents and are within the same cluster are fused using a mixed kernel model. In this thesis work since we assume all the data points to be gaussian, therefore we can fuse the data points using the weighted normal addition. The main reason we are fusing the data points in this thesis is, since they are gaussian and we also have the knowledge that some of these data points may be contributed by noisy agents, therefore adding some noisy gaussians with more accurate ones will shift the reading towards ground truth. This is represented in Figure 10 : where :

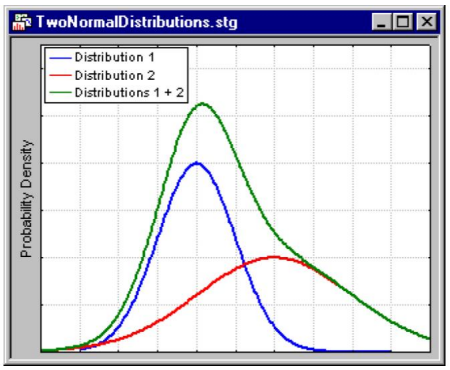

Figure 10: Weighted Gaussian Addition

$$mean(\mu_{new}) = \frac{\sigma_1^2\mu_1 + \sigma_2^2\mu_2}{\sigma_1^2 + \sigma_2^2}, variance(\sigma_{new}^2) = \frac{1}{\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}}$$

As highlighted above that, all noises are modeled as a guassian, and different gaussians for the same data point, when fused, shifts the net mean closer to the ground truth while decreasing uncertainty. In the context of this thesis, the above would mean that the data fused from other agents will be able to estimate the distance to an obstacle better and reduce the uncertainty. However, this is not always the case (as seen in later sections).In certain cases the fused interpretation may not be able to estimate an obstacle with high confidence. These scenarios happen when there are a majority of bad contributors (vehicles with bad sensors) that share SLAM data with each other (interaction). We handle such cases as a part of future works.

3.2.4 Maximum Deviation Test

The agent may associate some of its local data points with the fused interpretation contributed by the vehicles in near proximity. The agent must first identify the deviation of its own interpretation with respect to the fused map (global consensus). It must accept the one that is closest to the ground truth in order to enhance the safety and reliability of navigation. This analysis become nontrivial does not have a measure of the ground truth or the sense of which interpretation is more accurate; local or fuse interpretation.

One way in which the agent can estimate the ground truth is by statistically comparing it own interpretation with the fused world interpretation. This helps the agent analyze how deviated is its local interpretation with respect to the fused data. Since the fused data represents the consensus within other close-proximity agents, it likely that it will be closer to the ground truth versus a single local interpretation. This test is also important because it quantifies the consistencies in local interpretation with respect to fused map.

The maximum deviation test [24], as the name suggests is a statistical technique to

quantify the difference between two density functions, as all the data points are probabilistic with some confidence measure. The idea of the Max Deviation Test (MDT) is to generate the cumulative density functions (CDF's) for the two density function, one associated with the data point of the local and the other with that of the fused map. Now in order to quantify the deviation between the two interpretation, we compute the absolute errors on each percentile values respectively. Effectively we are matching the templates of the two distributions and compute instances where the deviation is higher than the tolerance level.

If these absolute errors for each percentile are within a threshold, then we return a score of 1, otherwise 0. Finally we sum the scores and if it is lesser than a set threshold, then we quantify that the two interpretations do not converge. The non-convergence of the two interpretations is an indicator of a false negative reading. Hence through the MDT we are able to track false negatives in autonomous navigation. We summarize the algorithm below:

1) Pass the density functions of the two data points (fused and local) to the MDT. In this thesis we pass the density function as a tuple which contains the mean distance to an obstacle and the variance associated with that obstacle. The MDT will convert the two density functions into CDF's using :

$$P(x) = \int_{-\infty}^{x} p(t)dt$$

where p(x) is the density function.

2) Next for each CDF we compute values for the range of 0 to 100 percentiles.

3) We compute the absolute error between the respective percentile values

4) If the error is within the threshold we return a score of +1 otherwise 0

5) Repeat for all percentile values

6) Sum the scores.

7) If the score is above a certain acceptable threshold we return that the functions are convergent or quite similar. If the score is below the threshold then the test indicates a false negative.

The MDT results are passed back to the agent, which makes a decision on the interpretations based on confidence (discussed in sections 3.2). An example where the test indicates false negative can be seen in Figure 11. The blue curve corresponds to the data interpretation from the local agent.

The curves are manifested after the MDT converts the data points from fused and local interpretation into the two CDF. In this example we set the tolerance level to be 5 percent (deviation) and we set convergence threshold at the value 95. That means that there should be at least 95 instances where the absolute error between percentiles

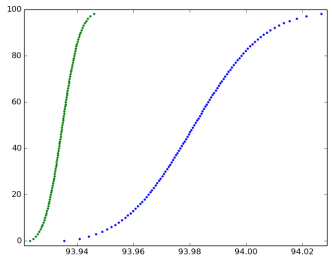

Figure 11: MDT identifies a false negative

values is within the tolerance. As we run the loop, the MDT test returns a score of 25 which is lower than the convergence threshold, indicating a false negative. We can see that this is clearly a false negative as the mean distance to the obstacle is significantly shifted for the blue curve.

The MDT is a powerful tool to compare and match two distributions as it does not depend upon the parametric state of the distributions or their modality. In other words MDT compares two distributions non-parametrically and is not centered around the mean. This is an important property, as it make MDT widely applicable in dynamic conditions. There are other powerful statistical tests like the **Kolmogorov Smirnov (KS) Test** [25] which, traditionally have been used to quantify the deviations between two nonparametric distributions.

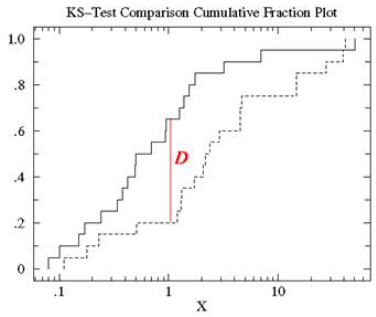

Figure 12: Kolmogorov Smirnov Test

However, the KS test compares the deviation between two distributions by first generating a profile that computes the distance of each point from the mean of the curve for the two distributions and then matches the two profiles. This can be seen in Figure 12. Hence a MDT filter will be more powerful than other statistical tests like the Kolmogorov Smirnov Test in autonomous navigation as usually the data interpretation in real world autonomy will be multimodal distribution that are not centered around

mean.

Another statistical technique that is used for quantifying deviations between two distributions is the **Kullback-Leibler (KL)** Divergence Test. The KL test compares the PDFs (probability density function) of two distributions and quantifies the divergence between two distributions based on entropy comparison. If you have two probability distributions, P and Q the KL divergence measures the similarity of P and Q. If $KL_{Divergence}(P||Q) = 0$, then the two distributions are equal.

They are computed as follows:

$$D_{KL}(P||Q) = \Sigma_i P(i) log \frac{P(i)}{Q(i)}$$

or

$$D_{KL}(P||Q) = \int_{-\infty}^{\infty} P(i) log \frac{P(i)}{Q(i)} dx$$

However, it can be seen that KL Test is non-symmetric. Which in our case means, that the KL Test will produce non-symmetric output for the fused and local interpretation if they are in inverse order. This is critical as for different close-proximity vehicles (with bad sensors) the measure of the divergence might be significantly different. An example of the KL Test is represented in Figure 13.

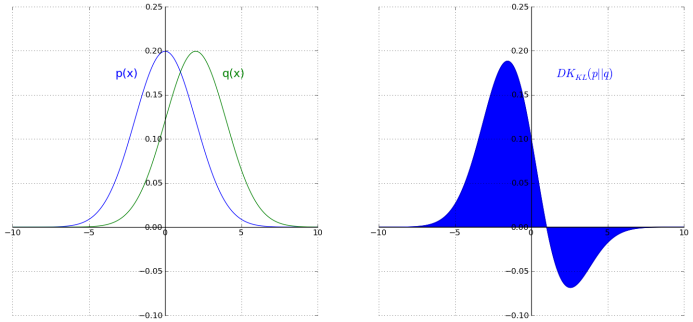

Figure 13: Kullback-Leibler Test

Its obvious that MDT is significantly powerful than KL Test in the case of statistical comparison of SLAM data between close-proximity autonomous agents, as MDT is symmetric and will produce consistent output irrespective of the order of comparison for the different agents.

4 Experimental Outline

The high-level understanding from the foregoing should be that autonomous navigation is most vulnerable to false negatives, which highly influence the navigation safety. We highlight a methodology for autonomous connected vehicles in **Section 3.2** to identify and correct some of these false negatives, this section is discusses the implementation and testing of the proposed methodology.

We use the **SUMO** [26] (Simulation of Urban Mobility), a well-known traffic simulator, to generate a traffic grid, implement traffic rules, and manage and maintain the traffic behaviour and kinematics. SUMO allows different software to generate and manage the traffic network in realtime via a python script called **TRACI**. The TRACI script opens a port to SUMO and allows for the modulation of the different network and traffic parameters. One reason that we chose SUMO over other traffic simulators is that SUMO allows users/software to control and change vehicle parameters in realtime. To simulate any autonomous behaviour, this granularity of control is essential.

Another reason for working with SUMO is that there exist several communication simulators that are built on top of SUMO. One such simulator is **Veins**, which simulates a DSRC framework and allows for V2X (V2V or V2I) communication in SUMO. In this work we will use Veins to simulate the DSRC framework. It should be noted that SUMO is based on real-world traffic theory and models, therefore we will accept the data from **SUMO as ground truth** throughout this thesis work. Next we need a soft-

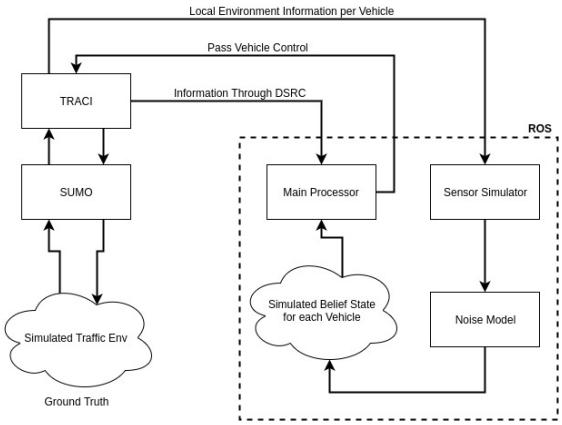

Figure 14: High-Level Schematic of the Simulator

ware package (or tool) to implement the main methodology proposed in **Section 3.2**, which includes generating LIDAR scans, LIDAR data segmentation with respect to agents, building SLAM and finally comparing the fused view with respect to local view via the Max Deviation Test. In this work we chose **ROS** as the software package to compute and simulate all these parameters. ROS is an ideal choice because it has been traditionally used in robotics and has several repositories for computing and

simulating robotic behaviours. Secondly ROS is portable to embedded scale and has repositories that can interface with bare-metal electronics, making it a suitable choice when considering real-world implementation.

The interaction between SUMO and ROS can be seen in Figure 14. SUMO sets up the traffic and vehicular environment and updates vehicle motion models at each simulation step. Some of the vehicles in SUMO are set as autonomous and their control and local data are passed to ROS via TRACI. This includes information about the location and instantaneous kinematics of the autonomous vehicle and data through its local DSRC channel. To simulate local sensor data, TRACI is used to generate an environment (map) grid which is then passed to a Sensor simulator that generates the data with respect to the specifications of the sensor. This data feed is then passed through a Noise unit, which distorts the data based on some noise model and then makes these noisy data available to the ROS-simulated autonomous vehicle.

Details of the simulator and experiments are highlighted in the subsections below.

4.1 Simulator Design and Process Flow

The simulator has been built on top of ROS and SUMO, as highlighted above. The main consideration for implementation of the simulator was to exhibit real-world behaviour. In this section we discuss the different components of the simulator, their interaction with other modules, and finally their role in implementing the methodology.

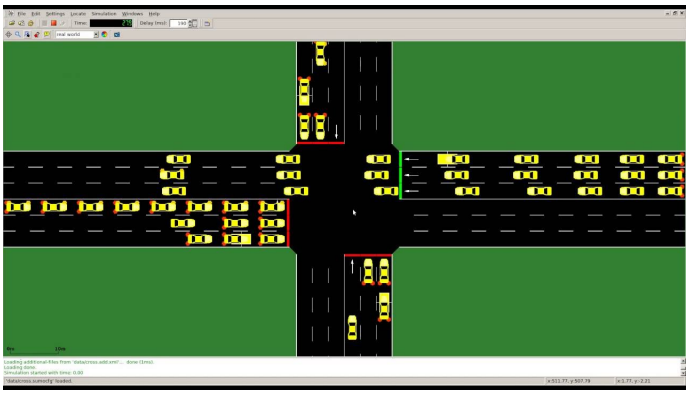

Figure 15: Example of road network simulation by SUMO

First the simulator must be able to simulate a traffic network to test our methodology and its overall efficacy, and for this purpose we chose SUMO. SUMO generates a road network, along with pre-designed routes, vehicle collision models, traffic rules and other traffic-related parameters using configuration files. This parameters are mostly static and cannot be changed in realtime during the simulation. These models are realistic and emulate real-world behaviour. These configuration files are loaded by SUMO

and we simulate a traffic network. An example is shown in Figure 15.

Once the road network is set up we spawn some vehicles and define their routes with respect to the road network simulated. Using TRACI we define number of vehicles to be spawned and randomly assign them routes. TRACI has a python API for SUMO and allows for easy GET/SET operations. Once the vehicles have been set up we randomly choose some number of vehicles and tag them autonomous. After spawning the different vehicles in SUMO, TRACI opens a port to ROS and passes information and control of the autonomous vehicles.

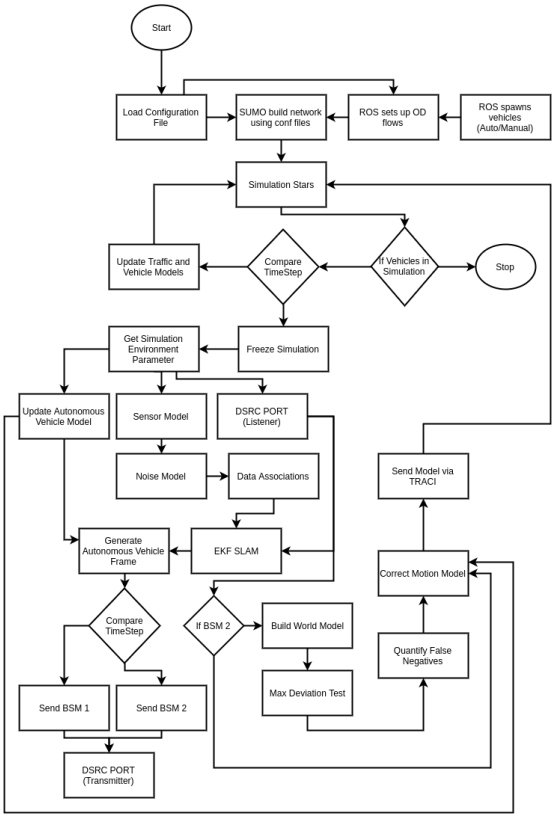

Figure 16: Process Flow for the Simulator

Figure 16 highlights the process flow of the simulator developed. The process flow shows the different levels of interaction in the simulator (between ROS and SUMO) which implement the proposed methodology. It can be seen that once we have the basic road network and vehicles set up, the simulator starts the SUMO activity for one simulation time step, which is basically updating the network and vehicle states with respect to their local models. Each simulation time step is exactly 100 millisecond (ms) in real-world time. The reason we use 100ms granularity is first, we know that

the minimum latency seen in DSRC protocol is 100ms and second, we set the sensors to operate at 10 Hz or 100ms range. After each simulation step, our simulator freezes SUMO to parse the update state of the vehicles, check for DSRC messages, and process the sensor information for each agent.

In order to implement the methodology, each autonomous agent must independently localize the maps in its local environment. For implementing such behaviours the simulator passes the updated location and kinematic state of the marked autonomous vehicles to ROS via TRACI. This simulates the data from the GPS, Odometer and IMU (inertial measurement unit) sensors that are critical to localize the autonomous agent. To perceive the environment, we simulate a LIDAR sensor for each autonomous vehicle. The simulator via TRACI gets the local obstacle grid within a circle of radius equal to the LIDAR range and centred on the coordinate frame of an autonomous vehicle.

We form this grid by first using a LIDAR sensor model to get the basic data specifications. For this work we model the LIDAR data as received by the **Hokuyo UTM-30LX-EW**, which scans for the entire 360 degrees with an angular resolution of 0.5 degrees (720 data points). TRACI builds a circle centred around the coordinates of the autonomous vehicle, with radius 120 meters. We consider the range of the LIDAR sensor to be 120 meters as this is the standard specification for LIDAR used in autonomous vehicles. Then TRACI finds points associated with obstacles within that field and returns their distances and angle of measurement. Figure 17 shows the LIDAR data received plotted in RVIZ.

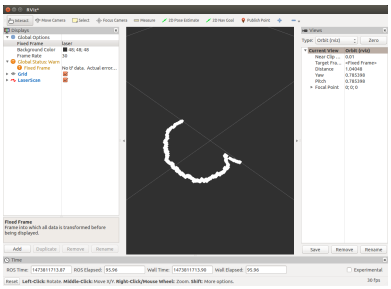

Figure 17: Simulated LIDAR data in ROS

These data are then passed through a NOISE model which adds a Gaussian noise to the distances returned. We assume the system to be uni-modal and to preserve the modality we assume very low error in measuring angles through the LIDAR sensor. This entire process simulates a LIDAR sensor with some in-built noise. We intentionally keep the noise level for some autonomous vehicles very high, as a result of which we will see false negatives in the observations. This noise data are now sent to the ROS port for the respective autonomous vehicle. The LIDAR data are modelled with occlusion, that is, if one obstacle is blocking an other, the LIDAR data will not be able to capture the latter.

The next step in the simulator is for all the autonomous vehicles to associate the LIDAR data to an obstacle. It should be noted that in this work we are assuming that only other vehicles show up in the obstacle map, as in this work we will focus on detecting false negatives, which in reality are vehicles that were missed in the sensor's reading. Each autonomous vehicle simulated in ROS then uses the **sequential compatible nearest neighbour** technique (highlighted in Section 3.2.1) to segment the LIDAR data.

Next ROS tries to find if there were any DSRC messages received in the given simulation time step. ROS does this by listing on a DSRC port maintained by Veins. Veins is used to send and receive DSRC messages over the SUMO simulator. It works on a parallel thread and opens a port to either send or receive DSRC compatible messages. While SUMO is frozen, Veins is also paused and we just load the buffer for sending information or read if any information is received. In certain cases some of the obstacles scanned by the LIDAR sensor will be connected vehicles that will pass their location information through the basic safety message over the DSRC protocol. Each autonomous agent can utilize this information to correct and associate particular clusters with the corresponding connected vehicles, making the association stronger. These LIDAR readings are logged; we refer to them as **Frames**. Each frame is passed to an **Extended Kalman Filter** (highlighted in Chapter 3.2.2) which outputs a belief state estimate of the vehicles and the respective obstacles.

Next the autonomous vehicles use ROS to process their instantaneous kinematic model into a BSM (Basic Safety Message) and then ROS forwards this message to the DSRC port that is maintained by Veins. Next, based on the self-localization information and local instantaneous obstacle understanding, each autonomous vehicle updates its motion and path models. This information is sent by ROS to TRACI, which updates the respective vehicle parameters in SUMO. After the SUMO models have been updated, the simulator unfreezes Veins and SUMO and lets them update their simulation model till the next simulation time step.

After every 10 simulation steps (which corresponds to 1 second in real world), each autonomous vehicle should have processed 10 frames through the extended kalman filter (EKF), and the output of the EKF should be a probabilistic estimate of distance for each seen obstacle. In this work, since we assume a uni-modal system, the output of the EFK is a Gaussian whose mean is the approximate distance to the obstacle and whose variance defines the measure of accuracy or confidence. In Section 4.1.1 we quantify the relationship between the variance and false negative that we use in this work. ROS then encodes the Gaussian for each obstacle into a Payload BSM (Basic Safety Message), and we refer to this probabilistic map as a **Super Frame**. These Super Frames are representations of the surrounding probabilistic world with respect to each autonomous vehicle. The latency of sharing these frames is 1 second. Therefore every other second from the start of the simulation, autonomous vehicles within DSRC range of each other receive these Super Frames, which are used to compare self and fused interpretation.

Thus every 10 simulation time steps, the different autonomous vehicles get Payload

BSM from near-proximity vehicles (if any) after which the autonomous vehicle distributively identifies the region of overlap between its own environment view and the views of the other vehicles. This overlap is then quantified in terms of region of overlap within the Super Frames of proximity vehicles. Once we have regions of interest (ROI), we segment the data maps out with respect to each Super Frame. Finally these segmented data maps are fused with each other using **normal weighted addition**, highlighted in Chapter 3.2.3. Finally the autonomous vehicle tries to associate the data points between the fused maps and its local super frame. The simulator uses the **K-nearest neighbours** approach which is highlighted in Chapter 3.2.3 to associate the data point in the two maps using angles as a metric rather than distances.

After the data association, the autonomous vehicle can identify two kinds of points; first, 1) local data points it can associate with the fused map, and 2) these data points it cannot associate. For the first case the simulator passes the local Super Frame Estimate (mean, standard deviation) and Fused Map estimate to the **Maximum Deviation Test Module** (highlighted in Chapter 3.2.4) to estimate the similarity in the two perspectives. The Maximum deviation test is able to identify the local false negatives for the particular autonomous vehicle. In the second case, the data points that the autonomous vehicle can not associate can either be obstacles that are only visible in the local view of the autonomous vehicle or missed data points that the fused world can identify or interpret. In these cases the local autonomous vehicle accepts the data points generated by the fused map and/or its self data points.

Finally, after analysis of instances and locations of false negatives in the local Super Frame, the vehicle sends updates to its Motion Model to account for the world understanding. ROS sends the motion model to TRACI, which updates the models in SUMO for the particular vehicles. This details the entire working of the simulator and how the methodology proposed is implemented. Simulation finally ends when all the vehicles in the simulator have completed their route and exit the simulator.

4.1.1 Types of False Negatives

The simulator tries to identify and minimize the number of false negatives. This section highlights what is considered a false negative in the simulator. In general a false negative is a critical data point that the sensor missed or that was lost in the interpretation of the sensor data. This is a little difficult to simulate as there needs to be a constant account of what is being missed for understanding the efficacy of the system.

In this thesis we simulate a false negative as a reading with low confidence or accuracy. We are assuming all noise models to be Gaussian, where the mean represents the distance measured while the variance represents the confidence, therefore a high variance for a particular obstacle would be considered a bad reading. If this reading is below a certain threshold or if this reading can be corrected by the fused map data then we consider the reading as a false negative. This model allows us to represent an autonomous vehicle missing an obstacle (true false negative) as a reading with a Gaussian of infinite variance.

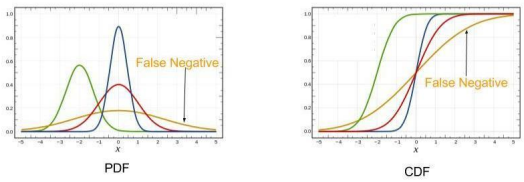

Figure 18: Representation of False Negative (orange curve)

A representation of a False Negative in the simulated data can be seen in Figure 18. The red curve in the figure represents a bad reading and therefore has a high uncertainty, while the green curve represents an accurate prediction. It is also important to highlight the types of false negative that can exist in the simulation. These are as follows:

- Noise models of an autonomous vehicle can be set with high variances. That is, whatever the local sensors measure has a high deviation from the real world. This is one of the primary types of false negative in the simulation. The intent is for these to simulate faulty or drifting sensors that constantly give erroneous readings.
- There exist cases in the simulator wherein the fused map information is not able to resolve the variance of a given data point to below a certain threshold, which means that even after fusing maps from multiple sources, there is little confidence about the existence and position of the obstacle. This could be a manifestation of high relative speeds in the system, multiple bad sensor interpretations, partial occlusion, etc..

4.1.2 Simulator Assumptions

One of the major considerations while developing the system was to ensure that the simulator resembles the real world as closely as possible. However, there were still certain assumptions made to relax some process computations, and some process development which were not in the scope of this thesis.

Some of the assumptions made were as follows :

- Firstly we assume that all interactions within the simulation are event-based. That is, at a certain time-step an event is initiated and agents appropriately interact with the simulator. This assumption was made to relax the issues of clock synchronization, which is beyond the scope of this thesis work.
- Another critical assumption in the simulator is that there exist no False Positives in sensor readings, as this thesis is focused on resolving false negatives.
- Another assumption made by the simulator is that all vehicles no matter what type cannot travel over 80 miles per hour. This assumption was made for au-

tonomous vehicles to have high overlapping regions in consecutive frames generated by the LIDAR data.

- Another assumption the simulator makes is that only vehicle are qualified as obstacles. The question of what is an obstacle in autonomous driving is beyond the scope of this thesis work and hence we just assume that other vehicles act as obstacles.

- We also assume that all noise in the simulation is uni-modal. It is not difficult to resolve multi-modal system, but building SLAM maps and localizing in a multi-modal system is much more computationally intensive. We also assume all noise to be Gaussian in nature, as it is easy to handle and operate on.

- The simulator also assumes that there is low noise in GPS, Odometer and IMU data. This assumption can be easily removed, by implementing an efficient multi-modal localization software; however, this was not the interest of this thesis.

- Another assumption in the simulator is that there are no elevations or depressions in the road network. Also the height of each vehicle is the same. These assumptions resolve into modulating the LIDAR sensor model with one LIDAR beam for building 2D SLAM maps. It is not difficult to deal with multi-beam lidar and build more accurate and complex point cloud, however that is not the interest of this thesis.

- The simulator also assumes that all autonomous vehicles are connected.

- Finally the simulator also makes some basic assumptions like vehicle are at least 5 meter apart from each other and each vehicle only passes an other on the left side.

4.2 Experiments

The proposed methodology is geared towards reducing the number of false negatives to enhance the safety and reliability of navigation for autonomous vehicles. To test the methodology's efficacy we must test it in a high-risk environment where the chances of accidents are high. In these scenarios we will examine the performance of the methodology to avoid an accident even if there exist false negatives in local sensing.

For this task we simulate two scenarios with high accidental rates. The first experiment simulates a lane changing scenario on a high-speed network. The combination of high speed along with several blind spots makes lane changing difficult and highly prone to accidents [27]. The second scenario is the case of an unsignalized intersection. One such example is a stop sign intersection [28]. This scenario is highly prone to accidents mainly due to bad driving behaviour. It has been noted that drivers with usually high speed are sometimes unwilling to stop at an unsignalized intersection, causing high accident rates. These scenarios become even more critical in the case of autonomous navigation because in either case a false negative in sensing may lead to a

severe accident.

We simulate both the scenarios with autonomous, connected and non-connected vehicles. All autonomous vehicles in the simulation are depicted by red color triangles, while any other vehicle is depicted using yellow triangles. ROS only controls and processes for autonomous vehicles, while the rest of the vehicles are controlled and managed by SUMO. There are also different types of vehicle models in the simulation. These models have different lengths, widths, acceleration rates, deceleration rates and navigation error variance. We randomly assign autonomous and manual vehicles from these categories. The intension is to simulate different kinds of vehicles in the real-world traffic.

Each experiment is conducted multiple times with random starting positions, routes and models for each vehicle. The details of the two simulations are given below.

4.2.1 Lane Changing Model

In this experiment we simulate a 2-mile-long straight highway with 3 lanes. We only simulate one-way traffic, with vehicles chaining lanes to avoid congestion and reach their destinations faster. We use the **lcSpeedGain** lane changing model in SUMO, which is a model with eagerness for performing lane changing to gain speed. Higher values result in more lane-changing. The representation of the network can be seen in Figure 19. In the figure we can see that an autonomous vehicle (circled vehicle) is trying to change a lane while there is incoming traffic.

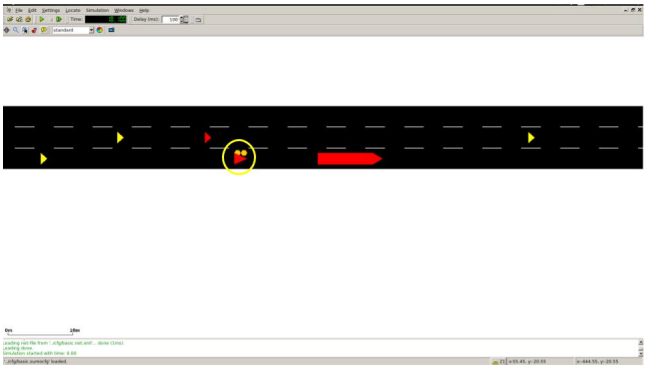

Figure 19: Simulation of Lane Changing Model

The highway has three exits and entry points at equal distances from each other. We simulate various traffic scenarios to test the efficacy of the methodology. One such scenario is that, we make 25 percent of the vehicles in the grid to be autonomous. Within the autonomous vehicles we set 5 percent of the vehicles to have very noisy sensors. This is intended to generate a greater number of false negatives in the simulation. We set the variance for the vehicle with bad sensors to be four times the value of the vehicles with regular sensors. We test the simulation with different numbers of vehicles and

variance measures and compute the efficacy of the proposed methodology in Section 5.

4.2.2 Unsignalized Intersection Model

In this experiment we simulate an unsignalized intersection shown in Figure 20, which has two one-way streets and an intersection. To study the impact of false negatives and the efficacy of the methodology to minimize them we need to increase the chances of interaction which will happen in case of collision at the intersection. Hence we simulate a all-way stop-sign intersection, wherein each vehicle randomly makes a decision to either stop or go at the intersection. In the experiment a vehicle may chose not to stop and run-over the intersection. This simulates a high entropy case for high number of collisions at the intersection.

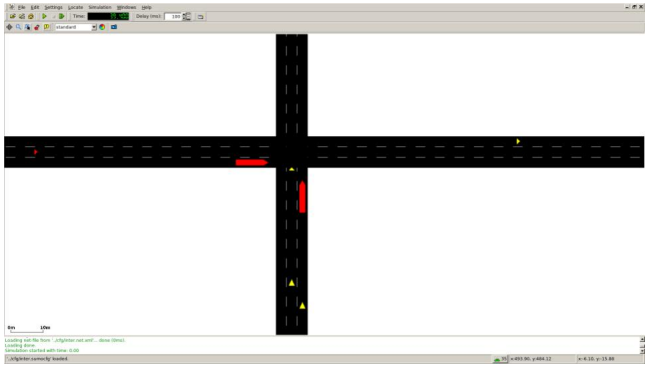

Figure 20: Simulation of Lane Changing Model

The motion model for the vehicles is such that each tries to reach its destination as fast as possible, which is the **Krauss** model in SUMO. Each road length approaching the intersection is 1-mile long. The interest points originate at the intersection, where vehicles try to avoid collision. It should be noted that the manual vehicles are controlled by SUMO to avoid collisions, while ROS tries the same for autonomous vehicles.

This simulation, like the previous one, is run on different traffic scenarios. One such scenario has 25 percent of the autonomous vehicles in the grid . Among the autonomous vehicles we set 5 percent of the vehicles to have very noisy sensors. This is intent to generate a greater number of false negatives in the simulation. We set the variance for the vehicle with bad sensors to be four times the value of the vehicles with regular sensors. We test the simulation with different numbers of vehicles and variance measure and compute the efficacy of the proposed methodology in Section 5.

5 Observations and Results

The two experiments highlighted above were conducted in three different settings, ten times each. The values highlighted below are the aggregated results. The three different settings for the experiments are as follows :

1) We set the total number of cars in the network to be 50. Out of the 50 vehicles, 25% of the vehicles are autonomous. Of the total number of autonomous vehicles, 5% of the vehicles are set with high sensor noise values.

2) We set the total number of cars in the network to be 100. Out of the 100 vehicles, 25% of the vehicles are autonomous. Of the total number of autonomous vehicles, 10% of the vehicles are set with high sensor noise values.

3) We set the total number of cars in the network to be 200. Out of the 200 vehicles, 50% of the vehicles are autonomous. Of the total number of autonomous vehicles, 10% of the vehicles are set with high sensor noise values.

It should be noted again that each of these settings was tested 10 times for the two proposed scenarios highlighted in Chapter 4. In each instance the choice of routes, vehicle type and starting location for each vehicle was random. It was seen that the proposed methodology was able to identify false negatives in a high number of instances, as highlighted below. However, before discussing the actual result we must first highlight the measure for result validation.

The methodology should be to identify the multiple instances of false negatives in navigation and update the local kinematics models of the agent in order to avoid collisions. Each agent logs local navigational information throughout the simulation. To validate the efficacy of the proposed methodology we compare the logged information with the ground truth. The simulator is designed such that SUMO maintains the ground truth states for each agent. Chapter 4 highlighted that SUMO never shares ground truth info with the agents. The info shared is either first passed through a noise model system or is received via the DSRC. Hence each agent maintains its own probabilistic view of the world. We compare the probabilistic interpretation with the ground truth via SUMO to understand the efficacy of the methodology.

1) We log information about when an agent identified a false negative and its corresponding action to counter. This prediction is cross-validated by SUMO. The agent first corrects for a false negative and then estimates the collision time. Based on which it updates its kinematic model to avoid the obstacle. SUMO cross-validates this behaviour and checks for potential collisions between the agent and the obstacle and in turn returns a positive point if the collision time increases after the kinematic model is updated.

2) SUMO records relative distance between pairs of vehicles constantly. When this distance passes below the threshold, it passes a warning of potential collision. SUMO warnings are checked and compared with agent prediction for collision. Cases where the agent predicts no collision, while SUMO does, mean that the

methodology failed to identify and correct the false negative. Hence the simulator also records the number of times the methodology failed.

The results are as follows :

Table 2: Results for Test Setting 1

Parameter	Experiment 1 Value (Avg)	Experiment 2 Value (Avg)
Total Simulation Steps	14000 Steps	14000 Steps
Instances when Methodology Failed	8.4 instances	6.2 instances
Instances when Methodology Passed	122 instances	97.5 instances
Success Percentage	93.5%	94.02%
Percentage of Overestimate	3.8%	4.3%

Table 3: Results for Test Setting 2

Parameter	Experiment 1 Value (Avg)	Experiment 2 Value (Avg)
Total Simulation Steps	30000 Steps	28000 Steps
Instances when Methodology Failed	12.5 instances	16.4 instances
Instances when Methodology Passed	210.2 instances	148.8 instances
Success Percentage	94.3%	90.07%
Percentage of Overestimate	5.2%	4.2%

Table 4: Results for Test Setting 3

Parameter	Experiment 1 Value (Avg)	Experiment 2 Value (Avg)
Total Simulation Steps	80000 Steps	77000 Steps
Instances when Methodology Failed	29.97 instances	39.2 instances
Instances when Methodology Passed	485.7 instances	340.2 instances
Success Percentage	94.18%	89.66%
Percentage of Overestimate	6.23%	8.23%

We have discussed all the rows of the result tables, except for the last row. The percentage of Overestimate is the measure of instances when an autonomous agent identified a false negative or erroneous reading but, when cross-validated with SUMO, there was no existence of any obstacle within the predicted distance. These are what we term as ghost readings or false positives. These readings manifest as a result of imperfect data

association techniques that are not able to resolve a local data point to a point in the fused map and/or can also arise due to fact that the error may grow over propagation and distort the interpretation of the obstacle to an extent that it cannot be associated with any other point through different perspectives or interpretation. However, these errors do not cause concerns for safety, as they just make the system more cautious.

Another important result/observation is the analysis of **interaction sensitivity**, which maps the number of autonomous vehicles that contribute data versus the number of instances when a false negative was resolved due to the interaction. This is an important result as it thresholds a minimum penetration of autonomous vehicles in the traffic network to resolve false negatives. It should be noted again that we still assume that a majority number of autonomous vehicles to have good sensors.

We document the number of autonomous vehicles in any given interaction versus number of times the methodology was able to resolve the false negatives in the given interaction among the different autonomous agents. The aggregated results from the lane changing and unsignalized intersection experiments are represented in the Figure 21. It can be seen that as the number of autonomous vehicles increases in an interaction, the probability of resolving a false negative also increases. Under the current assumptions we can resolve a false negative with a probability of 95 percent or more provided that there are at least four autonomous vehicles in close-proximity (120 meters) of each other.

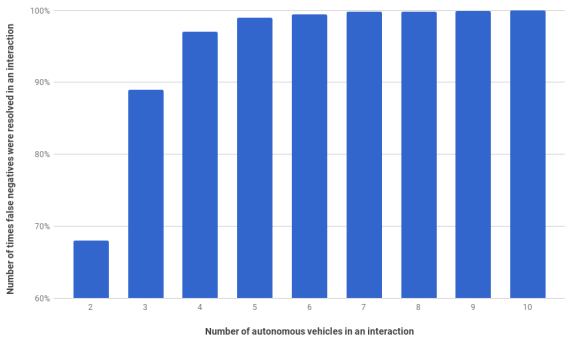

Figure 21: Interaction Sensitivity Analysis

Finally we must also comment about the cases when the current methodology fails to identify a false negative. Throughout the different experiments and scenarios we see that certain times SUMO corrects the navigation for the autonomous vehicles rather than ROS. This happens mainly because ROS is unable to resolve an obstacle close to the ground truth, therefore unable to identify a false negative. Initial investigation reveals the following reasons for the failure of the methodology to resolve false negatives:

- The autonomous vehicles with bad sensors, sometimes are unable to resolve or

associate a particular data point to an obstacle. In a crowded interaction, an autonomous vehicle with bad sensors may significantly distort the data point for an obstacle such that during data association, that point is interpret to be associated with some other obstacle. Hence in rare situations the local interpretation is completely void of certain obstacles.

- There exist cases where the majority of the close-proximity autonomous agents have bad sensors. In certain scenarios we see two autonomous vehicles sharing data with each other to estimate false negatives in their local interpretations. In such scenarios if both the vehicles have bad data, then it is likely that the agents will be unable to identify the false negatives.

It should be noted that there may exist other cases that contributed to the number of instances when the methodology fails. Exploration of such cases will be a part of the future works. It can be concluded from the initial investigation that in order to minimizes the number of instances when the methodology is unable to resolve a false negative, we must use better and robust data association techniques that is efficiently able to resolve the different close-by obstacles.

Also they agents must not myopically compare the their local interpretations with the fused interpretation as the fused interpretation might be significantly shifted from the ground truth, due to contributions from a majority of autonomous vehicles with bad sensors. The agent must be able to dynamically compare its own interpretation with combinations of data contributed by other close-proximity agents. This will certainly reduce the influence of bad contributors and minimize the instances when the methodology is unable to resolve a false negative. It should be noted that this development will also be a part of the future works.

6 Split-Level Architecture

In previous chapters we learned that there has been extensive work produced to develop a robust architecture that can support a fully connected vehicular environment. However as stated there are potential trade-offs between different protocols. While low latencies and reliabilities are essential in the framework, potential for higher penetration and accessibility is also a key factor for setting up the infrastructure. Therefore in these thesis we propose a Split-Level Architecture that utilizes the advantages of different communication protocols to provide desired services and functionality. Without the loss of generality it can be said that the proposed architecture is a communication adapter that can be embedded within vehicles and infrastructure for providing low latencies, high bandwidth and reliable communication channels for safety and mobility applications while increasing the penetration in the environment.

The Split-Level Architecture can be developed with any communication protocols provided the following is satisfied :

- One or more communication protocols must guarantee realtime latencies (100 ms or less) while being highly reliable (dedicated).
- One or more communication protocols must facilitate agents to share high-bandwidth information at low latencies (within a second)
- One or more communication protocols must facilitate agents to remotely connect to the Internet and/or should be accessible to other open-communication channels
- At least two communication protocols must be utilized that are capable of transmitting and receiving a common standard message structure.

It should be noted that features mentioned above are desirable for number of reasons. First, for any connected vehicle application that deals in security we need to have a reliable connection along with low latencies as these applications are time-critical. Second, while safety is time-critical, usually low-latency communication tradeoff bandwidth. However, there are many scenarios where low-bandwidth information is not enough to compute or implement safety protocols, hence in these applications we usually require a more detailed representation. One such example is avoiding pot-holes (which have been one of the major sources of road accidents in the United states [29]), which can be achieved if a message containing its estimated location (map) is passed to respective agents, which usually requires more bandwidth. Thus our proposed architecture should be flexible in certain scenarios to send and receive a higher bandwidth information while maintaining relatively low latency.

Third, it can be said that due to the Internet and the Internet-of-things (IoT) initiatives there is already much information present about traffic and road environment. This information should be accessible to the architecture to not only improve understanding but also enhance penetration within the network. Agents using the proposed architecture should not only be able to benefit from the information present in the grid, but

also contribute to it (hence allowing agents to also crowd-source critical information), making the system even more integrated. Lastly, we contend that the architecture must utilize at least two communication protocols to serve as a redundant data source which is meant to address the dead-zone characteristic of any communication system. This is a critical requirement, as redundancy can prove essential for safety applications. The name **Split-Level Architecture** is derived due to the latter requirement. For the work

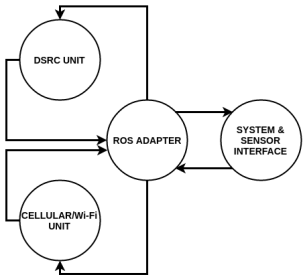

Figure 22: High Level Description of Split-Level Architecture

in this thesis we utilize DSRC and Cellular (5G/4G LTE), because as highlighted in the previous chapters they are the state-of-the-art protocols for servicing connected vehicle environments and also fulfill the desired characteristics mentioned above. We can see a representation of the proposed split level architecture in Figure 22. It can be seen that the two modes of communication are interfaced with a third element (processing unit). So far we have only considered the communication aspect of the architecture ; however, we propose to make a complete system which can run as a stand-alone module, we need to interface the communication modules with a processing element. Also remember that the work of this thesis is to provide methodologies for connecting autonomous vehicles into the connected grid while also improving the understanding or representation of the environment.

Hence we propose a third element to be the Robot Operating System also commonly know as ROS. ROS is a software package built on the C++ and Python programming languages and heavily support libraries for processing sensor data and computing or implementing robot behaviours. It should be noted that the connected vehicle architecture has been designed to be interfaced with vehicular sensors to compute the basic parameters required by the safety and mobility messages. Usually it is assumed that an on-board computer will parse these parameters (odometer,brake status) and set up the message. However, our argument is that these basic parameters can be processed using the standard techniques to generate a more descriptive understanding of the vehicle parameters and the environment. With the rise of intelligent vehicles, semi-autonomous vehicles and autonomous vehicles, we also have access to a wider array of sensors on each vehicle. Therefore ROS seems like an obvious choice given our goals and the wide range of its applicability.

We highlight the behaviors of each component of the architecture below.

6.1 System Architecture

A high-level architectural representation can be seen in Figure 23. The central piece of the architecture is the ROS adapter that is responsible for processing sensor information and encoding it for transmission over the two communication protocols. Again we must highlight that the architecture is open to standards and does not depend upon any particular communication network protocol. However, a critical operating requirement for the architecture is that the system must perform at realtime operational latencies and posses the desired characteristics mentioned previously. It should be noted that

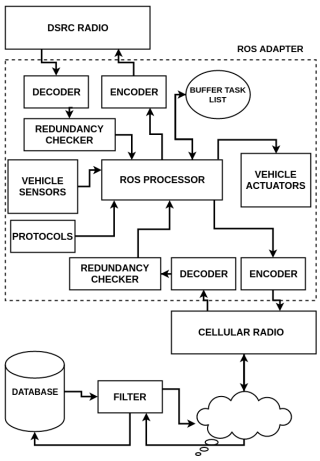

Figure 23: Representation of Systems level design of the Split Level Architecture

the two modes of communication not only act as redundant sources for transmitting and receiving data, but are also utilized by ROS for certain specific applications. For example, DSRC or a low-latency network is preferred by ROS for safety applications, while cellular is mainly used for sending and receiving mobility information. In certain cases that require a higher bandwidth of data transmission, ROS sends the basic low-resolution data from the DSRC port, while the rest is sent via the cellular interface.

6.1.1 ROS

The ROS adapter is a processing and communication interface of our architecture. The architecture utilizes ROS for 1) interacting with the vehicle-level sensors and operational/control sub-systems; 2) processing (encoding/decoding) and generating multiple safety and mobility data structures that can address vehicular states and local sensory parameters; and 3) operating at the embedded level to interact with DSRC and cellular radios.

The ROS adapter enables a seamless integration between robotic systems and the connected environment. As highlighted before, today vehicles are embedded with arrays of sensors like GPS, radar, camera, LIDAR. ROS libraries are able to parse the raw sen-

sor information and process it to generate vehicle inertial and kinematic states along with data to represent the environment from the perspective of the agent. One such example is ROS building occupancy grids using LIDAR point clouds [30], and encoding this information into the BSM part 1 and 2 message structures. Now we can send this information through both cellular and DSRC protocols, as both will comply with the message structure. This is a really powerful feature, as it allows autonomous vehicles to share localization data pertinent to neighboring non-autonomous, non-connected vehicles with other autonomous vehicles.

Also it should be noted that ROS can scale to embedded controls and hence can also be interfaced with vehicle actuators for making semi-autonomous control decisions if need be. One of the major issues with DSRC and DSRC-type systems is to get the drivers to respond to a safety issue. The human mind can perceive information at very high speeds ; however, the process of acting has quite high latencies depending upon the person. In these cases ROS can set or control behavior like obstacle avoidance or speed adjustment by assessing the cruise and or steer control actuators, which any modern vehicle would have.

Finally, the ROS adapter is also responsible for filtering out stale information. Since we have redundant sources of communication that operate at different latencies, we will have stale information about various agents in the connected environment. Thus decisions are made solely based on comparing the time-stamp of a message received to the latest message received by that agent by accessing the internal message logger (buffer). A message is accepted if it is either the latest with respect to the time-stamp or the message type (example safety) requires the system to store older messages (up to a certain relative time-stamp).

6.1.2 DSRC

A low-latency ad hoc network is a critical component of the proposed architecture because it is the primary mode of communication to effectively handle safety-related data. In a road environment, vehicles in close proximity to each other are often at risk of collision due to occlusion between two or more vehicles, high speeds of travel, or dangerous actions on the part of drivers, etc. A low-latency communication network enables vehicles within a close proximity to share time-critical safety data, which may help them avoid any impending collision. The architecture can utilize any ad hoc communication protocol provided the given network to have guaranteed maximum latencies in the 100 millisecond range. These vehicular ad hoc networks will allow: 1) vehicles to share basic safety information (e.g. velocity, brake, direction, mechanical state, etc.); 2) intelligent, semi-autonomous and autonomous vehicles to share their critical sensor data (e.g., localization and tracking information).

6.1.3 Cellular

The role of the cellular communication protocol is three-fold: 1) it provides a platform for third party data integration like Google Maps and Waze. These third party

trusted-sources are reliable low-noise databases that maintain information about the traffic environment and allow for external sources to read from them. A simple example could be the accidental data feed from a provider like Waze, which otherwise might not have entered into our DSRC ad hoc network and might be relevant for mobility and planning or may have safety considerations for our vehicles.

The second role the cellular protocol will play is that of sending and receiving mobility-based messages. These messages will most likely not be time-critical, but will rather be used for planning navigation. Hence, the ROS adapter will push non-critical messages through cellular, hence avoiding crowding the low-bandwidth DSRC network. Finally ; cellular will be used as a redundant means of communication for safety messages. In certain scenarios, where a high-bandwidth safety message needs to be sent, ROS will develop a low-resolution copy of the message generated and send it via DSRC, while sending the entire message through cloud. The ROS adapter at the receiving end will take action using data through DSRC, but will try to correct its analysis/action after the cellular message is received.

This thesis assumes a back-end virtual agent on the cloud that is able to manage incoming data through cellular and make it available to the appropriate agents. Developing the said agent is beyond the scope of this thesis and therefore we will be using an off-the-shelf back-end system like Twitter.

6.2 Sensor Sharing

The term Sensor Sharing refer to a process where two or more agents are sharing their sensor data (raw and/or processed) via some communication protocol. A connected vehicle environment is essentially a sensor sharing application, where multiple agents (V2X) share their local information to enhance safety and mobility within the traffic network. Traditionally the data shared between multiple agents are usually parameters like signal phase timings (SpaT), vehicle speed and heading, warning messages, which usually give information only about the connected agents. We propose that using the split-level architecture including the ROS adapter, we can process and share a much more detailed description of the connected agents and the local environment around them.

As the number of smart (intelligent, semi-autonomous, and autonomous) vehicles increases we find two extremely useful sensor sharing methodologies that would enhance the reliability of autonomous navigation and increas the connected penetration within the traffic network, by exploring sensor sharing in the Split-Level Architecture. We can therefore advance autonomy and penetration without significant investment in hardware and development costs. We highlight the two methodologies below :

6.2.1 SLAM based Map Sharing

SLAM or Simultaneous Localization and Mapping is a probabilistic technique for localizing an autonomous robot with respect to the surroundings while building a local

map. SLAM is a common methodology used by autonomous vehicles to probabilistically estimate their locations and the locations of the objects around them. In a road network, the objects of interest around an autonomous agent are usually other vehicles.

As an example we can propose that these navigating semi or autonomous agents are constantly scanning their surroundings using LIDAR sensors. The data from the sensors are fused with other sensors like Camera, Radar, IMU (Inertial Measurement Unit) to compute self-localization parameters and generate a probabilistic map using some form of SLAM technique like FastSLAM [31].This process is able to generate an occupancy grid map that encodes all localization information. A representation can be seen in Figure 24.

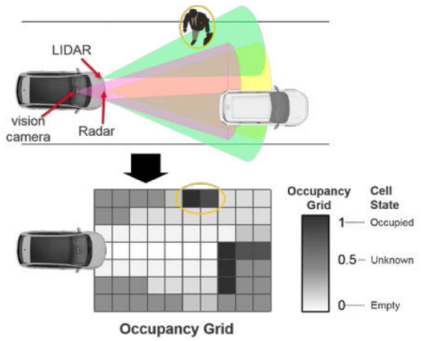

Figure 24: Representation Occupancy Grid created using LIDAR data

These occupancy grids not only contain the self-localization information about the agent, but also encoded information about the local world around the agent from its perspective. Parts of these maps that are critical to safety can be shared over the Split-Level Architecture to multi-autonomous agents within the environment. This information may be treated by spatially distant agents as a mode for mobility data, while near agents compare to the incoming maps for their own safety considerations.

The work proposed in this thesis is mainly utilizing this sensor sharing concept for increasing the reliability of navigation in autonomous vehicles.

6.2.2 Simulated BSM

The idea of the Simulated Basic Safety Message (S-BSM) is that, for the foreseeable future, there will be vehicles that are not in-built with the connected vehicle infrastructure and that therefore will not be connected. However, this problem can be addressed by leveraging the advantages of the Split-Level Architecture and increasing number of intelligent and autonomous vehicles. We propose that connected vehicles with embedded sensor systems can track and locate non-connected vehicles. Using the on-board sensors the agents can then estimate the heading, velocity and location of such non-connected vehicles. This information can in turn be formatted into a BSM message and pushed into the connected grid.

Hence such a methodology can greatly enhance the penetration of the connected environment and the information can be utilized by smart signals, traffic planners, path planning software, to optimize their decisions, now accommodating previously non-connected vehicles.

6.3 Latency Test

The fundamental factor critical to the behavior of the split-level architecture and its overall effect within the system is operational latencies. If the latencies within the architecture are too high, the architecture become inefficient in relaying data between two or more sub-systems. In order to test the feasibility of the proposed architecture, we quantified individual latencies of each sub-system. The test used two DSRC radios (an OBU, and a RSU), manufactured by Arada Systems PVT. LTD; an ODROID XU4 processing board with an on-board ARM processor and installed with ROS; and two android cell-phones connected to a Twitter back-haul network via 4G LTE cellular connectivity.

The process of this experiment was such that the DSRC OBU reads the data from a broadcasting DSRC RSU, and forwards that information via a UDP port to ODROID running ROS. ROS in turn pushes those data to twitter via the 4G Network and using the Twitter HTTP API, and finally a remote agent running ROS parses the same data point using the second android device and the Twitter API. The test computed the over-all latency in the process as well as individual latencies of the sub-systems. The experiment recorded five thousand (5,000) such interactions between each sub-system. Table 5 show the results.

Table 5: Latencies in Sub-Systems

Sub-System	Maximum Latencies (millisecond)
DSRC (RSU)- DSRC (OBU)	100 ms
DSRC (OBU) - ROS	10 ms
ROS - CLOUD	300 ms
CLOUD - CLOUD	100 ms
CLOUD - ROS	250 ms

The observed latencies indicate that sub-systems operate at close to realtime scale and hence make the architecture feasible to operate within the connected vehicle environment.

7 Conclusions and Future Work

The work in this thesis has been dedicated to address the issue of safety and reliability in autonomous navigation. Due to errors in sensing, which could either be persistent or temporary, autonomous agents often make bad decisions which jeopardize the safety of people within the vehicle or outside. False negatives in general have direct adverse affect on safety and reliability and these kinds of errors usually are difficult to identify and track.

To help minimize the number of false negatives in autonomous navigation, we proposed a methodology for sharing and comparing SLAM maps with different intelligent and/or autonomous agents that are in close proximity. The information shared between the agents can be used to analyze the ground truth and identify the false negatives in local world interpretation. This methodology was based on the principle that if multiple agents have overlapping fields of view, then false negatives for some agents are bound to show up as good data points for the other agents. Thus agents can statistically compare their world views in order to estimate and track false negatives.

We tested the proposed methodology in high accident-rate traffic scenarios with some autonomous agents with bad sensors. We found that our proposed methodology has great efficacy, however in certain cases may also manifest ghost readings (false positives). Lastly we discussed the connected vehicle infrastructure and certain issues with the state-of-the-art protocols. We proposed a communication architecture that will enhance the capability of connected vehicles and allow for more intelligent traffic control and management operations. To test the efficacy of the architecture, we ran some latency tests to quantify real-world performance.

There is a huge potential to expand and improve this work. One of the potential future works is addressing false positive errors using statistical analysis of SLAM maps. Another future work is developing methodologies using the proposed framework to quantify sensor health in realtime. This will be of great importance to autonomous vehicle companies and advocates. Another critical future work, could be to integrate the path planning behaviour for autonomous vehicles into the proposed framework to further enhance safety and introduce a platform for collaborative autonomy in traffic environments. Another future work is stress-testing the simulator to quantify the various thresholds (example number of autonomous vehicles, etc.) for optimal performance.

A critical future work is developing algorithms to handle the cases when the fused interpretation is unable to resolve an obstacle or a data point with high certainty. These cases occur when there are a majority of vehicles with bad sensor that contribute their SLAM data in an DSRC interaction. Therefore, we must incorporate some technique to efficiently fuse the data before comparison with the local interpretation.One of these techniques could be the integration of self-sensor health assignment methodologies. These methodologies can define an accuracy weight for the local SLAM interpretation. We will share the weights along with the corresponding SLAM data. The maps will be

fused based on the weights estimated, which might increases the estimation certainty.

Another important methodology can be combinatorial comparison between different agents. Suppose there are three vehicles, two of them have bad sensors while one has accurate sensors. In this case, rather than fusing the data, which would in turn diverge the estimate from the ground truth, we compare the SLAM data between the agents in all possible combinations. This will enable the agent to determine the most accurate estimation. We propose exploration of such methodologies and study of their efficacy, as part of the future works.

Finally the proposed methodology has a significant commercial application in the autonomous driving world. The proposed methodology is able to quantify the errors in sensing in realtime and assist the autonomous agents to navigate safely by leveraging data from other agents. In crude sense the agent is calibrating its own sensor data to avoid false negatives. Thus this methodology can extend towards developing a realtime sensor calibration technique that will calibrate a bad sensor by utilizing accurate data from the other close-proximity agents and/or sensors-embedded infrastructure.

This has high commercial value, as in the future autonomous vehicles will be able to self-calibrate and optimize its own sensor as opposed to having to go to the shop frequently for an expensive calibration by a technician. Hence there may be autonomous vehicles in the future with accurate, optimized, and finely calibrated sensors; their sole purpose will be to assist with realtime calibration of other agents.

In closing, we must highlight that though this work addresses some of the critical problems in autonomous navigation, it is still the tip of the iceberg. It is clear to me that exploring probabilistic robotics with collaborative autonomy will prove essential in realizing and building competently autonomous vehicles and systems.

References

[1] S. Gibbs, "Google's self-driving car in broadside collision after other car jumps red light."

[2] R. Spacek, "Tesla car on autopilot warned driver 7 times before fatal crash, safety regulator says."

[3] D. Z. Morris, "Ubers self-driving systems, not human drivers, missed at least six red lights in san francisco."

[4] F. Thomanek, E. D. Dickmanns, and D. Dickmanns, "Multiple object recognition and scene interpretation for autonomous road vehicle guidance," in *Intelligent Vehicles' 94 Symposium, Proceedings of the*, pp. 231–236, IEEE, 1994.

[5] H. Schneiderman and M. Nashman, "A discriminating feature tracker for vision-based autonomous driving," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 6, pp. 769–775, 1994.

[6] H. Surmann, A. Nüchter, and J. Hertzberg, "An autonomous mobile robot with a 3d laser range finder for 3d exploration and digitalization of indoor environments," *Robotics and Autonomous Systems*, vol. 45, no. 3, pp. 181–198, 2003.

[7] Y. L. Morgan, "Notes on dsrc & wave standards suite: Its architecture, design, and characteristics," *IEEE Communications Surveys & Tutorials*, vol. 12, no. 4, pp. 504–518, 2010.

[8] Y. J. Li, "An overview of the dsrc/wave technology," in *International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness*, pp. 544–558, Springer, 2010.

[9] Y. J. Li, *An Overview of the DSRC/WAVE Technology*, pp. 544–558. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[10] "Federal communications commission. fcc 03-024. fcc report and order february 2004.."

[11] S. International, *Dedicated Short Range Communications (DSRC) Message Set Dictionary*. 2016.

[12] K. Hinckley, J. Pierce, M. Sinclair, and E. Horvitz, "Sensing techniques for mobile interaction," in *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pp. 91–100, ACM, 2000.

[13] J. L. Baxter, E. Burke, J. M. Garibaldi, and M. Norman, "Multi-robot search and rescue: A potential field based approach," in *Autonomous robots and agents*, pp. 9–16, Springer, 2007.

[14] F. Ye, H. Luo, S. Lu, and L. Zhang, "Statistical en-route filtering of injected false data in sensor networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, pp. 839–850, April 2005.

[15] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 2016-01-0128, pp. 15–24, 2016.

[16] B. Douillard, J. Levinson, and G. Sibley, "Calibration for autonomous vehicle operation," May 4 2017. US Patent App. 14/756,996.

[17] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 3354–3361, IEEE, 2012.

[18] U. Franke, D. Gavrila, S. Gorzig, F. Lindner, F. Puetzold, and C. Wohler, "Autonomous driving goes downtown," *IEEE Intelligent Systems and Their Applications*, vol. 13, no. 6, pp. 40–48, 1998.

[19] E. Olson, "Apriltag: A robust and flexible visual fiducial system," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 3400–3407, IEEE, 2011.

[20] M. Thuy and F. P. Leon, "Non-linear, shape independent object tracking based on 2d lidar data," in *Intelligent Vehicles Symposium, 2009 IEEE*, pp. 532–537, IEEE, 2009.

[21] Y. Li and E. B. Olson, "Extracting general-purpose features from lidar data," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1388–1393, IEEE, 2010.

[22] A. J. Cooper, *A comparison of data association techniques for simultaneous localization and mapping*. PhD thesis, Massachusetts Institute of Technology, 2005.

[23] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT press, 2005.

[24] I. K. Isukapati and G. F. List, "Synthesizing route travel time distributions considering spatial dependencies," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, pp. 2143–2149, IEEE, 2016.

[25] R. Wilcox, "Kolmogorov–smirnov test," *Encyclopedia of biostatistics*, 2005.

[26] D. Krajzewicz, J. Erdmann, M. Behrisch, and L. Bieker, "Recent development and applications of SUMO - Simulation of Urban MObility," *International Journal On Advances in Systems and Measurements*, vol. 5, pp. 128–138, December 2012.

[27] B. Sen, J. D. Smith, and W. G. Najm, "Analysis of lane change crashes," tech. rep., 2003.

[28] R. A. Retting, H. B. Weinstein, and M. G. Solomon, "Analysis of motor-vehicle crashes at stop signs in four us cities," *Journal of Safety Research*, vol. 34, no. 5, pp. 485–489, 2003.

[29] D. F. Rudny and D. W. Sallmann, "Analysis of accidents involving alleged road surface defects (i.e., shoulder drop-offs, loose gravel, bumps and potholes)," in *SAE Technical Paper*, SAE International, 02 1996.

[30] S. Kohlbrecher, O. von Stryk, J. Meyer, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pp. 155–160, Nov 2011.

[31] D. Hahnel, W. Burgard, D. Fox, and S. Thrun, "An efficient fastslam algorithm for generating maps of large-scale cyclic environments from raw laser range measurements," in *Proceedings 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003) (Cat. No.03CH37453)*, vol. 1, pp. 206–211 vol.1, Oct 2003.