Vehicular Clouds and Networking for Semi-Supervised Alignment of Manifolds of Stereo and LIDAR for Autonomous Vehicles

A Dissertation

Presented in Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

with a

Major in Electrical Engineering

in the

College of Graduate Studies

University of Idaho

by

Yassine Maalej

Major Professor: Mohsen Guizani, Ph.D.

Co-Major Professor: Sameh Sorour, Ph.D.

Committee Members: Ahmed Abdel-Rahim, Ph.D.; Suat U. Ay, Ph.D.; Feng Li, Ph.D.

Department Administrator: Joseph Law, Ph.D.

August 2018

Authorization to Submit Dissertation

This thesis of Yassine Maalej, submitted for the degree of Doctor of Philosophy with a Major in Electrical Engineering and titled "Vehicular Clouds and Networking for Semi-Supervised Alignment of Manifolds of Stereo and LIDAR for Autonomous Vehicles," has been reviewed in final form. Permission, as indicated by the signatures and dates below is now granted to submit final copies for the College of Graduate Studies for approval.

Major Professor:	Mohsen Guizani, Ph.D.	Date
Co-Major Professor:	Sameh Sorour, Ph.D.	Date
Members:	Ahmed Abdel-Rahim, Ph.D.	Date
	Suat U. Ay, Ph.D.	Date
	Feng Li, Ph.D.	Date
Department Chair:	Joseph Law, Ph.D.	Date

Abstract

For decades, researchers on vehicular ad-hoc networks (VANETs) and autonomous vehicles presented various solutions for vehicular autonomy and safety. Yet, the work developed in these areas has been mostly conducted in their own separate worlds, and barely affect one-another despite their direct relationship. In later years, the Internet of Vehicles (IOV), encompassing sensing, communications, connectivity, processing, networking, and computation is expected to bridge many technologies to offer value-added information for the navigation of self-driving vehicles, to reduce vehicle on board computation, and to deliver desired functionalities. Potentials for bridging the gap between these two worlds and creating synergies of these two technologies have started to attract significant attention from many companies and governmental agencies. In this Dissertation, a comprehensive survey is first presented with an overview of the emerging key challenges related to the two worlds of Vehicular Clouds (VCs) including communications, networking, traffic modeling, medium access, Vehicular Cloud Computing (VCC), VC collation strategies, security issues, and Autonomous Driving (AD). This will include 3D environment learning approaches, AD enabling deep-learning, computer vision and Artificial Intelligence (AI) techniques. The Dissertation then discusses recent related work and potential trends on merging these two worlds in order to enrich vehicle cognition of its surroundings, and enable safer and more informed and coordinated AD systems.

Modern vehicles are equipped with advanced communication, computation and storage capabilities in On-Board Units (OBUs). These are used to form a Vehicular Cloud (VC) as coalitions of affordable resources to host infotainment applications. With the limitation of static vehicular communication schemes and the computational capabilities constraints in vehicular micro-datacenters, VCs have overcome these technological limitations. VCs are supposed to maximize the usage of Vehicle to Infrastructure (V2I) communications over Service Channels (SCHs) for non-safety applications while maintaining reliable short-lived safety applications Vehicle-to-Vehicle (V2V) communications in the Dedicated Short

Range Communication (DSRC) technology. This Dissertation presents a novel Advanced Activity-Aware (AAA) scheme to enhance Multi-Channel Operations based on IEEE 1609.4 standard in MAC Protocol implemented in Wireless Access for Vehicular Environment (WAVE) for a more dynamic model. The developed AAA scheme rely on the awareness of the vehicular safety load. It aims at dynamically finding an optimal setup for switching between Service Channel Interval (SCHI) and Control Channel Interval (CCHI) by decreasing every inactivity in the network. The developed scheme is implemented using NS3 and maintains the default Synchronization Interval (SI), as defined by the standard in Vehicular Ad hoc Networks (VANETs).

Furthermore, we evaluate both sequential and parallel CUDA-accelerated Markov Decision Process (MDP) based schemes using a fast greedy heuristics algorithm to optimize the problem of vehicular task placement with both IEEE 1609.4 and opportunistically available V2I of AAA scheme. We derive the system reward of VCC by considering the overall utilization of the virtualized resources of the distributed Vehicular Clouds (VCs) as well as the optimality of the solution of placement of the vehicular Bag-of-Tasks (BOTs).

The Dissertation then discusses a vision to create a beneficial link by designing a multimodal scheme for object detection, recognition, and mapping based on the fusion of stereo camera frames, point cloud Velodyne LIDAR scans, and Vehicle-to-Vehicle (V2V) Basic Safety Messages (BSMs) exchanges using VANET protocols. Exploiting the high similarities in the underlying manifold properties of the three data sets, and their high neighborhood correlation, the proposed scheme employs semi-supervised manifold alignment to merge the key features of rich texture descriptions of objects from 2D images, depth and distance between objects provided by 3D point cloud, and awareness of self-declared vehicles from BSMs' 3D information including the ones not seen by camera and LIDAR. The proposed scheme is applied to create joint pixel-to-point-cloud and pixel-to-V2V correspondences of objects in frames from the KITTI Vision Benchmark Suite, using a semi-supervised manifold alignment, to achieve camera-LIDAR and camera-V2V

mapping of their recognized objects. Then, the alignment accuracy results in 2 different driving sequences that are presented while illustrating additional acquired knowledge of objects from various input modalities. The effect of the number of neighbors employed in the alignment process and accuracy is also studied. Finally, this Dissertation is concluded with a list of future research directions that will be useful to researchers in this field.

Acknowledgements

I would like to thank my major Professor Dr. Mohsen Guizani and co-major Professor Dr. Sameh Sorour for their guidance throughout this work and for giving me the opportunity to conduct research on VANETs and Autonomous Vehicle. Their valuable guidance and support in solving difficult problems throughout my doctoral studies and research, for coping with critical situations, and for their patience and encouragement throughout my doctoral research. Their wisdom helped me overcome many difficulties and challenges that made this work possible.

I would like also to thank my committee members Dr. Ahmed Abdel-Rahim, Dr. Suat U. Ay and Dr. Feng Li for their valuable comments that helped focus on my research direction.

I would also like to thank my colleagues in the Wireless Communication Research Lab and National Institute for Advanced Transportation Technology (NIATT) Research Center at the University of Idaho for their support and help throughout this journey.

Dedication

This dissertation is lovingly dedicated to my parents Mohamed and Najoua, my brother Amine and my sister Yosra.

Their support, encouragement, and constant love have been the main motivation throughout my life.

Table of Contents

List of Tables

List of Figures

List of Acronyms

VANETs vehicular ad-hoc networks

IOV Internet of Vehicles

VC Vehicular Cloud

VCC Vehicular Cloud Computing

AD Autonomous Driving

AI Artificial Intelligence

V2V Vehicle-to-Vehicle

V2I Vehicle-to-Infrastructure

NSF National Science Foundation

USDoT US Department of Transportation

OBU Onboard unit

DSRC Dedicated Short Range Communication

LTE Long Term Evolution

CCH Control Channel

SCH Service Channel

MAC Media Access Control

SLAM Simultaneous Localization And Mapping

BSM Basic Safety Message

NPRM Notice of Proposed Rule-Making

NHTSA National Highway Traffic Safety Administration

SI Synchronization Interval

QoS Quality of Service

MDP Markov Decision Process

FCC Federal Communication Commission

AC Access Categorie

WSM WAVE Short Message

PHY Physical Layer

CCHI Control Channel Interval

SCHI Service Channel Interval

Chapter 1: Introduction

1.1 Motivation

The race among vehicle manufacturers and IT companies to develop, test and market a reliable fully autonomous vehicular technology is motivated by the big expected annual revenue stream of smart autonomous and connected vehicles. Researchers have increasingly infused more cyber capabilities to vehicles. Starting from basic communications to cell towers, passing through V2V and Vehicle-to-Infrastructure (V2I) short range communications (a.k.a. VANETs), and reaching the recent era of VCs. On the other hand, many researchers have taken significant steps in vehicular automation with a long-term vision of reaching a full driving autonomy. Despite their obvious relevance to one another, the bridging and synergies between these two worlds have not been properly investigated. But this has recently started to attract massive interest from many companies and government agencies. For instance, the US National Science Foundation (NSF) and the US US Department of Transportation (USDoT) have expressed tremendous need and importance to relate these two worlds together in several of their call for research proposals in 2017 [112]. They clearly emphasized the importance of integration and fusion of data from various input modes in order to create a deeper understanding of the surroundings for safer and more coordinated AD applications. More precisely, enriched 3D scene reconstruction by different input technologies and deep learning techniques are of paramount importance to allow autonomous vehicle systems to perform effectively and safely. These directions are strongly supported by the few accidents and fatalities [114] and traffic light violations [52] made by autonomous vehicle prototypes from top industries in the market (e.g., Tesla, Uber). These incidents that could have easily been mitigated if there existed strong connectivity, computing and distributed coordination resources among vehicles and and the necessary road infrastructure. Luckily, VANET applications for both safety and non-

safety messaging have been approved by many government agencies (e.g., USDOT) and are ready for deployment. Forecasters estimate more than 250 million wireless-enabled vehicles by 2020 [112]. In addition, vehicle manufacturers, such as Cadillac, have already introduced commercial products (e.g., 2017 Cadillac CTS). Many modern vehicles will soon be equipped with sophisticated computing and storage capacities in their Onboard unit (OBU). In addition, vehicles are getting more and more communication technology options, such as Dedicated Short Range Communication (DSRC), WiFi, 3G or Long Term Evolution (LTE), which allow them to efficiently communicate and exchange information about their positions and intended steering commands. Furthermore, these wireless communication technologies can be used for awareness and safety through V2V beacons, as well as to enrich vehicular knowledge about their status and timings through V2I connections to static infrastructure entities such as traffic signals, street signs, etc. In addition to the available computing capabilities, such as micro datacenters in modern RSUs as an example, modern vehicles are equipped with computing resources in their OBUs. All of these distributed resources together offer a great opportunity to form VCs. Advanced VC technologies efficiently combined with recent advances in AD can revolutionize the latter. A VC does not only provide much more enriched and informed 3D understanding of the vehicle surroundings through communications among the vehicles and the infrastructure. But also migrates current AD decision-making paradigms from single isolated operations per vehicle to a more cooperative and coordinated operation through VCC. However, it is a difficult task to create a real-time 3D map that is enriched from the mapping of data derived from distinct sensing inputs as well as premapped data that usually provides static references on streets from the cloud. Reaching this stage requires not only merging and finding the synergies of VC and AD systems, but also getting separate data from the various technologies constituting each of them. For instance, the DSRC technology enables safety and non-safety applications for VANETs, which constitute a key component in VCs and ADs. Safety applications are time critical applications that are based on V2V

communications that enrich the knowledge of AD systems, while non-safety applications rely mostly on V2I communications to enable the connectivity of vehicles to VCs. The Wireless Access for Vehicular Environment (WAVE) standard partitions the bandwidth into seven channels of 10 MHz each, one Control Channel (CCH) to serve safety applications, and six Service Channels Service Channel s (SCHs) to serve non-safety applications. DSRC specifies a channel switching scheme to allow vehicles to alternate between these two types of applications. Moreover, the channel switching scheme has a lot of limitations and needs to be improved and adjusted depending on the network density and the requirements of the applications.

Thus, there is a need for constructing robust single and multi hop communications for VCs in order to perform the desired coordinated role for multiple non-safety or commercial applications. This creates many challenges to optimize their coalition formation, manage the cloud resources efficiently and distributively, and make the technology more cost effective.

On the other hand, many deep learning and artificial intelligence systems and algorithms have been developed and tailored to enhance autonomous vehicles' 3D surrounding reconstruction and image flow recognition from both camera feeds and LIDAR point clouds. End-to-End recognition and driving decisions frameworks have also been recently studied. However, many challenges still exist in terms of the accuracy of the resulting learned environment and decisions taken by autonomous vehicles, especially in very low visibility (e.g., foggy weather, camera facing the sun) and out-of-sight scenarios (e.g., vehicles hidden by buildings/trucks at intersections), in which camera and LIDAR technologies can easily fail.

1.2 Research Significance and Market Opportunity

Connected and autonomous vehicles offer a promising market opportunity for on board communications equipment manufacturers, autonomous steering, Internet Service

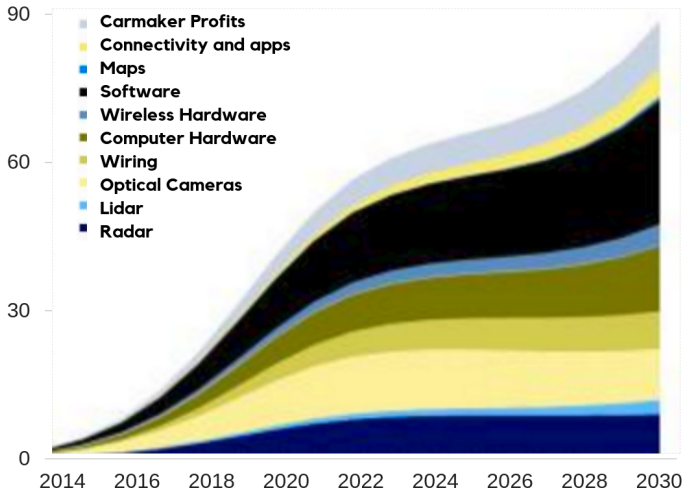

Figure 1.1. Projected Market Share of Autonomous Vehicles Technology Through 2030 [42].

Providers (ISPs), LIDAR and RADAR scanners, AI vehicle computing platforms, etc.

Appealed by the estimated $87 billion market opportunity by 2030 [42], traditional vehicle manufacturers are thriving to put the pedal to the medal while they contest with technology companies to put driver assisting features of fully automated self-driving technology in vehicles. The software systems, computation hardware, and optical cameras are projected to form the most significant economic impact in the self-driving vehicles market, as presented in Fig. 1.1.

The National Highway Traffic Safety Administration (NHTSA) has issued a notice of proposed rule-making Notice of Proposed Rule-Making (NPRM) to mandate vehicle manufacturers to mount DSRC radios for V2V communications in new vehicles sold in the USA market starting in 2020 [10]. Consequently, this is expected to produce a rapid expansion of ubiquitous V2V and V2I connectivity, as the number of projections shown in Fig. 1.2 makes the leap to self-driving vehicles even smaller. Regardless of the on board communications, connectivity to the cloud has a substantial and profound impact on businesses and the society. It enables a variety of key vehicular applications such as software updates, real-time traffic with parking information, charging stations location and availability, and vehicular apps that evolve the deployment of autonomous vehicles.

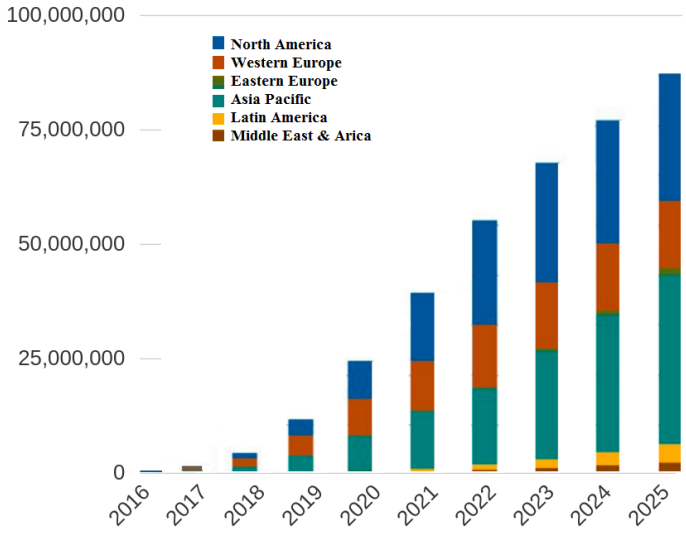

Figure 1.2. Projected Numbers of Vehicles Equipped With V2X Connectivity Through 2025 [11].

Specifically by 2020, projections estimate that the connected vehicle market is expected to create a total revenue of $141 Billion [2]. In addition to the aforementioned driving related features, connected vehicles would gain from cloud services including advanced infotainment systems, commercial applications, and deep analysis and diagnostic systems. Based on different connectivity solutions, the annual global sales are expected to gain a tremendous growth especially for the embedded and integrated solutions, as shown in Fig. 1.3. The embedded connectivity solutions enable connectivity, intelligent computation and built in services and applications running inside the vehicle. While tethered solutions rely on the separation of the applications, remain inside the vehicle, and the connectivity is delivered externally (e.g., using a phone). The integrated solution (usually called bring-you-own-device) enables connecting a device that has the computation, applications and connectivity to the vehicle's dashboard.

US consumer trust and interest in simple, partial, and fully automated driving systems have increased as detailed in Fig. 1.4 presented by the study in [30]. Among all the levels of vehicle automation technologies, younger consumers in the US have shown strong interest in systems with emphasis on basic automation, adaptive safety, limited

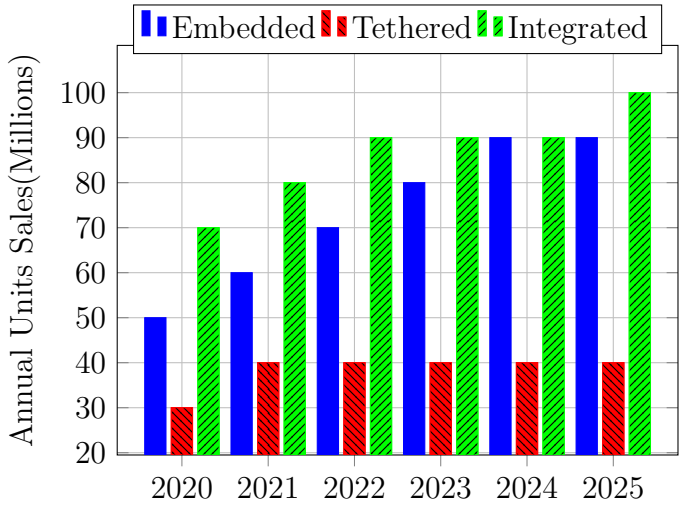

Figure 1.3. Forecast Worldwide Sales of Vehicle Connectivity Solutions.

autonomous driving and full autonomy. These survey findings presented more details about the trust and the interest of US customers between 2014 and 2016, which definitely point to a fast-pace growth demand towards the AD systems.

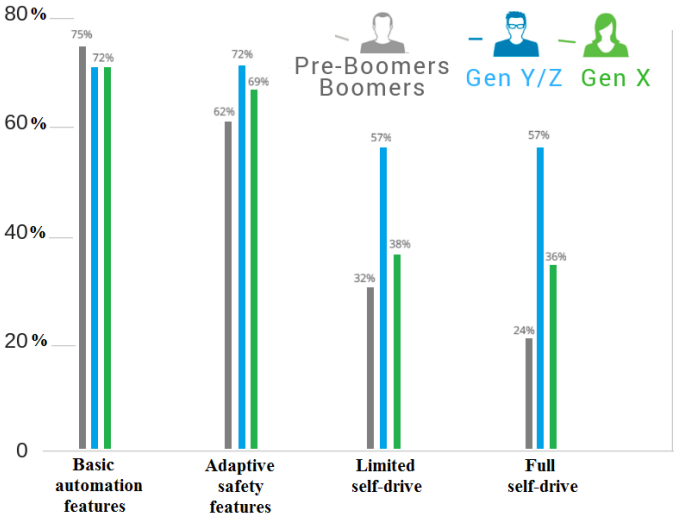

Figure 1.4. US Consumer Interest of Various Levels of Vehicle Automation [30].

1.3 Goals

Based on the importance of the application of self-driving cars and connected vehicles, the goals of this work can be summarized as follows:

- Investigate current vehicular technologies and enhancement of the static WAVE standard.
- Improvements of throughput of the 802.11 p standard over non-safety channels while maintaining the reliability of safety communication.
- Vehicular Cloud creation and simulation of throughput and VM migration.
- CUDA-accelerated and sequential scheduling study of tasks to maximize the overall reward of the VCC.
- Application of deep learning on camera frames and LIDAR point cloud of real driving sequences.
- Fusion and alignment between the output of both perception systems.
- LIDAR data processing, cleaning and tracking with Extended Kalman Filters application.

1.4 Related Work

The IOV includes different types of Vehicular Clouds Computing (VCCs) platforms to support various commercial or non-safety applications. It offers an interesting market opportunity and raises many technological challenges to make it cost effective. VCs combine and manage the available resource of micro data centers in connected vehicles and RSUs in infrastructure and are deployed by incorporating cloud-based services into VANETs.

VCs are non-conventional cloud architecture with critical requirements for efficient computation and storage resources utilization and expected to manage communication and networking limitations as opposed to traditional clouds. A wide-range of on-demand vehicular applications and services, including multimedia streaming, content sharing, urban sensing and safety and surveillance systems are offered by VCC [49]. One way of implementing VCs is by consolidating the OBUs of geographically co-located parked vehicles as proposed in [50]. This type of cloud model is easy to establish since the vehicles are in fixed positions do not require safety messages and consequently we can maximize the usage of all DSRC channels for vehicle to the cloud communication. Olariu *et al.* [19] presented a novel VCs model that is linked via wires or cellular communication to a backbone that plays the major role of management of the cloud system. The model itself is not cost efficient because all cloud resources are rented and none of the affordable microdatacenters in RSUs and OBUs are being included. In [106], the developed VC is established to host vehicles' infotainment applications by using the offered microdatacenters in on-street RSUs. A challenging VC model is formed by dynamically merging the OBUs of moving vehicles, forming an Ad Hoc Networks, to offer non-safety services [13]. Maalej *et al.* [75] developed a VC model that is built by merging the micro data centers in OBUs of on the move vehicles under RSUs coverage and without computational contribution. This VC modeling is complex to manage with respect to the high mobility of vehicles which leads to not only an unpredictable availability of computational resources but also variable performance of V2V communications and Vehicle to cloud. Vehicles with dual-radios are used in [67] by having one DSRC radio continuously tuned at CCH and one radio continuously being used for SCH. Even though this approach avoids the necessity of using Multi-Channel Switching, but induces high interference cost caused by radio channels adjacency. Authors in [131], implemented an independent adaptive channel switching mechanism to achieve selfish switching optimality of each vehicle aside since safety timers end immediately after safety related packets ends. This methods results in a significant dropped BSMs since it

does not keep synchronization between vehicles while sending and receiving over CCH. In [53], the authors proposed to adapt the Minimum Contention Window of the CCH's packet Access Classes based on a certain CCHI that guarantees a given transmission rate of safety messages, communication range changes, vehicle density, safety message rates and sizes. A fair amount of research in the literature are focusing making of the value iteration algorithm calculation run in parallel. These research efforts are mostly related on the theoretical formulation, and if implemented are employed on cluster systems or specialized shared memory multi-CPU systems as illustrated in [104]. The parallelization limitation of these system is due to the fact of expensive shared memory in multi-CPU systems and additions to increase parallelism is very costly, while cluster systems incur a high communication cost because of missing shared memory [121]. The cost of communication gets higher considerably with each computational resource addition it outweighs the benefit in performance. These developed systems offer parallelism for a few heavy processes working one problem at a time and does not use the key benefit of parallelism offered by GPUs for executing light-weight threads simultaneously. The acceleration of the decision making of task scheduling in vehicular clouds has never been developed as a full solution. Additionally, a computation resource allocation scheme is being applied to VCC in [133] with the emphasis on energy cost and income of the cloud by applying the problem as a semi-Markov decision Process (SMDP), while Yu *et al.* [127] focused only on the resource reservation, allocation and management for the VC versus virtual machine migration cost.

Analyzing camera frames using various techniques has been one of the mostly used techniques for autonomous vehicle surrounding recognition. One suggested technique is semantic segmentation, which labels each pixel in an image with the category of the belonging objects. To determine that a certain pixel belongs to a vehicle, a person or to any other class of objects, a contextual window that is wide enough is defined to show the surrounding of the pixel and consequently make an informed decision of the pixel's

object class. Techniques based on Markov Random Fields (MRF), Conditional Random Fields (CRF) and many graphical models are presented in [55], [66], [89] to guarantee the consistency of pixel labeling in the context of the overall image. In addition, the authors in [116], [63], [51] developed various methods for image pre-segmentation into super-pixels, which are used to extract the categories and features from both individual segments and combinations of neighboring segments.

Alternatively, the authors in [65] attempted to create 3D reconstruction of dynamic scenes by achieving a long-range spatio-temporal regularization in semantic video segmentation, due to the fact that both the camera and the scene are in motion. The developed idea is to integrate deep convolutional neural networks (CNNs) and CRF to perform sharp pixel-level boundaries of objects. The proposed solution minimized the distances between the features associated with corresponding points in the scene, and consequently optimized the feature space that is used by the dense CRF. To this end, deep learning has shown the best performance in inferring objects from previously untrained scenes. In [119], the segmentation of the input images was achieved by representing the dynamic scene as a collection of rigidly moving planes and jointly recovering the geometry/3D motion when over-segmenting the scene. The developed piece-wise rigid scene is intended to represent real world scenes with independent object motions rather than pixel-based representations like partially used in [86].

Joseph *et al.* [101] developed a general purpose object detection system characterized by a resolution classifier and the usage of a two fully connected networks. These two networks are built on top of a 24-layer convolutional network, followed by two fully connected layers. Additionally, a unified muti-scale deep CNN for real-time object detection is developed in [24] with many sub-network detectors and multiple output layers for multiple object class recognition.

Another widely-used sensing technique for 3D environment reconstruction is LIDAR scanners. 3D LIDAR-generated point clouds were already used in distance ranging, ob-

stacle detection and avoidance, path planning, and were thus imported to autonomous driving systems. 2D convolutional neural networks (CNNs) [56] have been designed for processing and recognizing objects from 3D LIDAR point cloud. However, this solution is not considered optimal since it requires a model to recover the original geometric relationships. Vote3Deep is developed in [41] for fast point cloud object detection using 3D CNN, in order to keep the key power of LIDAR as distance and objects 3D shapes and depth detection. The KITTI Vision Benchmark Suite [48] offers raw LIDAR and labeled objects from point cloud.

As clarified above, most autonomous driving systems rely on LIDAR, stereo cameras or radar sensors to achieve object detection, and scene flow estimation of objects on roads. Despite the great advancement in both technologies, they are still incapable of detecting hidden elements, such as hidden vehicles or pedestrians. Camera based systems may also fail in detecting geometrically line-of-sight entities due to limited visibility conditions, such as bad weather conditions (e.g., very bright sun, fog, heavy rain/snow) and close colours to surrounding nature (e.g., the cause of the prototype accident in [115]). Finally, the camera and LIDAR techniques fail in detecting road/traffic conditions, (e.g., red traffic lights, change in speed limit, etc), which may cause traffic violations (e.g., [52]) and even fatal problems. As aforementioned, all such problems may be resolved if these sensing technologies are complimented with actual in-flow information from both close vehicles and traffic infrastructure through VANETs. Google's self driving car project called WAYMO [83] is collaborating with Intel in order to create powerful chips responsible for the processing and fusion of data retrieved from radars, cameras, and LIDARs. Researchers in [111] presented an approach to geometrically align points from LIDAR scans to points captured from a 360 degree camera.

In addition, VANETs offers various types of V2V and vehicle-to-infrastructure (V2I) safety messages on 7 control channels operating over a dedicated 75 MHz spectrum band

around 5.9 GHz [61, 76]. Our aim in this work is to present an augmented scene flow understanding and object mapping by considering not only LIDAR and cameras, but also DSRC-based V2V beacons exchanged between vehicles.

The traditional localization approach using Global Positioning Systems (GPS) is not only inaccurate but also limited since it gives the position of the vehicle itself without distances of their surroundings. Knowing that the automated driving instructions need to be precise and requires that the steering commands to be in tune with the surrounding moving and static objects. Most research attempts that are used by vehicle automation systems to gain better insight about textural and spatial details, use Simultaneous Localization And Mapping (SLAM) techniques [39] for self-driving cars localization through odometry [93]. This uses data irrespective of sensing modality to approximate the estimate changes of a vehicle's position over time according to their surrounding. SLAMs techniques' heavy computation and limitation to guarantee only positioning is the incentive to this work by offering alignment of dynamic surrounding objects fused from different input sensors, instead of aligning the whole scene without knowing the objects and mining them.

1.5 Contributions

Our work contributions can be divided into three major pillar projects, namely AAA scheme and VC scheduling, manifold alignment and sensor fusion and tracking and LIDAR 3D scene understanding.

Cloud computing and vehicular Communication represent the corner stones for building smart cities with safer roads, intelligent transportation management and on the air infotainment applications. The key advantage of Vehciular Clouds (VCs) is to host applications that are frequently requested and deployed on VANETs. The virtualized computational resources in OBUs of vehicles and in RSUs offer a promising environment to place non-safety computation hungry applications. The vehicular network reliability and

the clouds performance are of paramount importance because of the complex challenges including the increase of the offloaded data from the cloud, the rapidly changing topology of the network and the safety of vehicles. The Dedicated Short Range Communication (DSRC) [62] standard is considered as a key vehicular communication technology that was created by the Federal Communication Commission (FCC) [7]. DSRC spans over a dedicated 75 MHz spectrum band around 5.9 GHz in USA [88]. The used bandwidth is partitioned into seven different channels of 10 Mhz each. The channels contain one Control Channel (178) to serve event-driven and safety related applications, and six Service Channels (172,174,176,180,182,184) to serve packets of cloud-based non-safety applications in vehicular networks. Time critical Safety applications are based on V2V communication and delay-tolerant non-safety applications based on the V2I communication and may include V2V communication during infrastructure coverage limitations. Consequently, VANETs deployment is expected to guarantee a reliable dissemination of safety messages in V2V as well as to take advantage of abundant resources in a VCs to host applications with a reasonable downtime on clients' side when Virtual Machine Migration (VMM) occurs while leaving the coalition forming the cloud.

The main contributions developed in the advanced scheme based on IEEE 1609.4 and the VC task scheduling are summarized as follows:

- Develop a DSRC channel inactivity awareness and highly dynamic automated interval switching scheme based on the calculated average effective interval utilization.
- VCs connectivity improvement by maximizing the SCH utilization and reducing V2I end-to-end delays.
- Reducing the probability of the dropped Virtual Machine (VM) duing migration by improving the VC throughput and enhancing the reliability applications through a decrease in the end-to-end delay of packets between vehicles and RSUs.
- Formulation of an optimal vehicular task placement scheme as a Markov Decision

Process and comparison with a fast greedy heuristic algorithm for scheduling vehicular tasks in a VCC system that is formed of various VCs with opportunistically available V2I communications.

- Accelerate the iteration value algorithm used to calculate the optimal decisions of the MDP problem using the CUDA parallel programming model.

The goal of this Dissertation is to merge the key features of LIDAR in giving accurate distances, camera with object textural details, and V2V beacons for the awareness of both hidden out-of-sight vehicles or vehicles not observed by the two other means. We first adapt the camera and LIDAR learning and object recognition schemes to prepare the resulting data for alignment. We also generate BSMs for vehicles detected in the KITTI Vision Benchmark Suite for alignment purposes. Exploiting the physical neighborhood correlation within the three data sets, and their natural correspondences in the 3D physical space, we then cast the merging problem of these three sets of data as a semi-supervised manifold alignment. Our proposed approach is to first identify few clear correspondences between data points from each pair of data sets, and employ them to align (i.e., pair) the rest of the points between the camera-LIDAR and camera-V2V data sets, thus establishing a full object correspondence among the three sets. To perform this alignment, we compute the neighborhood correlations and Laplacian matrix for in each data set using local linear embedding. The alignment problem is then formulated as an eigenvalue problem over a compounded Laplacian matrix. Once the mapping of paired points is done, the other points from each data set can be easily paired and the non-paired points can be added in the aligned 3D environment, thus significantly enriching the vehicle knowledge of its surroundings. We test this work using the scene flow, 3D LIDAR point clouds, and generated BSMs of the KITTI Vision Benchmark Suite, and perform the camera-LIDAR and camera-V2V alignment.

1.6 Organization of the Dissertation

The remainder structure of this Dissertation is organized into six major chapters. Chapter 2 provides a comprehensive survey of the key challenges in VANETs, VCs, stereo perception and LIDAR understanding and the fusion techniques. In chapter 3, we present the WAVE 802.11p MAC with the multi-channel operations defined in the IEEE 1609.4 standard and the developed AAA advanced scheme. Afterwards, chapter 4 sheds light on the vehicular workload modeling and vehicular cloud computing scheduling with both sequentially and parallel schemes. The developed technique to post-align pre-learned scenes for camera feeds and LIDAR scans between the created manifolds are presented in Chapter 5. In Chapter 6, we present the extended kalman filters applied to the 3D CNN for the raw lidar scans. Finally, we conclude the Dissertation by summarizing the important take-away contributions and open research directions in chapter 7.

Chapter 2: The Integration of Vehicular Clouds and Autonomous Driving

2.1 Key Challenges in VANETs and VCs

The major studied building blocks of vehicular networks can be classified into four major sections; namely: communications technologies, cellular and DSRC architectures, network architectures and VANET MAC protocols. The vehicular networks are intended to manage the connectivity of vehicles from and to VCs and to support access independently of the clouds' architectures and challenges. This section focus on the overall challenges in VCs and provides a thorough summary of the most relevant architectures developed in the literature.

2.1.1 Vehicular Communications Technologies

Modern vehicles are mostly equipped with various communications technologies such as WiFi, DSRC, 3G, and LTE. This is considered a valuable advancement, but ensuring that vehicles use the most appropriate communication technologies is a very challenging problem. This is the case for each of these technologies and the trade-off between their pros and cons. The maximum range of WiFi-direct can extend to 200 meters with a highest data rate of 11 Mbps. Whereas LTE is a paid mobile communication technology that has a wider connectivity coverage than DSRC and WiFi, but provides a lower data rate that ranges from 5 to 12 Mbps. Even though DSRC has a maximum data rate of up to 27 Mbps, its maximum range is 1000 meters.

2.1.1.1 Dedicated Short-Range Communications (DSRC)

The DSRC [61] standard was coined by the Federal Communications Commission (FCC) [7] to operate over a dedicated 75 MHz spectrum band around 5.9 GHz in the USA [88] and partitions the bandwidth into seven channels. The DSRC has been adopted to enable two classes of applications (Safety and Non-Safety) for VANETs. One control channel CCH number 178 which are used to serve safety applications. Six SCHs (172,174,176,180,182,184) to serve non-safety applications.

DSRC specifies a channel switching scheme that allows vehicles to alternate between the transmission of safety and non-safety packets. The DSRC standard recommends that, during a SI which is 100ms long, vehicles should visit the CCH to exchange their status messages with neighboring vehicles. The SI is composed of Control Channel Interval (CCHI) and Service Channel Interval (SCHI), both intervals are used to access the CCH and the SCHs for sending safety and non-safety application packets, respectively. Many research studies have been conducted to optimize the CCHI and the SCHI to guarantee a fair split of the SI between the safety and non-safety applications and to prioritize applications based on their QoS requirements. A higher time share of the CCH from the SI leads to more reliable safety applications. In [53], the authors developed two algorithms to optimize the length of the CCHI so that non-safety applications have a fair share of the SI. In [77], the authors developed an advanced awareness scheme of channel inactivity to dynamically achieve optimal interval switching based on the average effective interval utilization.

2.1.1.2 Cellular Technology and DSRC-Cellular hybrid Architectures

Due to the aforementioned limitations of DSRC, many research attempts have raised interest to investigate the support of cellular technologies (e.g., 3G, LTE, 5G) for vehicle to any (V2X) communications and usage of both DSRC and cellular communications, as

shown in Fig. 2.1. Cellular technology is gaining momentum in the research community over DSRC because of the following enablers:

- The capacity of dense networks to support high bandwidth demand of vehicles and cellular users at the same time. In such scenario, the performance evaluation of the downlink with unicast and broadcast vehicular messages is studied in [81] for LTE based systems.

- Reduced frequency of horizontal handoff of vehicles between BSs is guaranteed through the large cellular coverage.

- Unlike DSRC poor Non-Line-Of-Sight receptions, cellular LTE systems provide potentially better coverage as demonstrated in [82].

- Cellular is widespread and mature technology that accelerates the deployment of V2X communications and does not require new infrastructure facilities like DSRC.

On the contrary, current research studies that investigate sending vehicular safety messages using LTE [25], and cross-traffic assistance of vehicular safety messages using UMTS and LTE [82], do not account for traditional cellular network traffic. Even though the downlink and uplink transmission rates between the BS, as the access network connecting to the Internet, and vehicles in LTE (up to 300 and 75 Mb/s, respectively) are higher than in DSRC, LTE cannot provide the same awareness update rate and latency for Cooperative Awareness Messages (CAM) like "offered" using DSRC [82]. To put it differently, the centralized nature of cellular networks imposes that messages should be sent from vehicles to BSs that unicast or broadcast messages to other vehicles. In other words, instead of directly sending safety messages using DSRC, in cellular networks safety messages have to pass by the BS. Therefore, the use of the cellular technology results in lower awareness update rate and higher latency for disseminating delay critical safety-related applications. Additionally, several challenges limit transmission of the messages received by the BS:

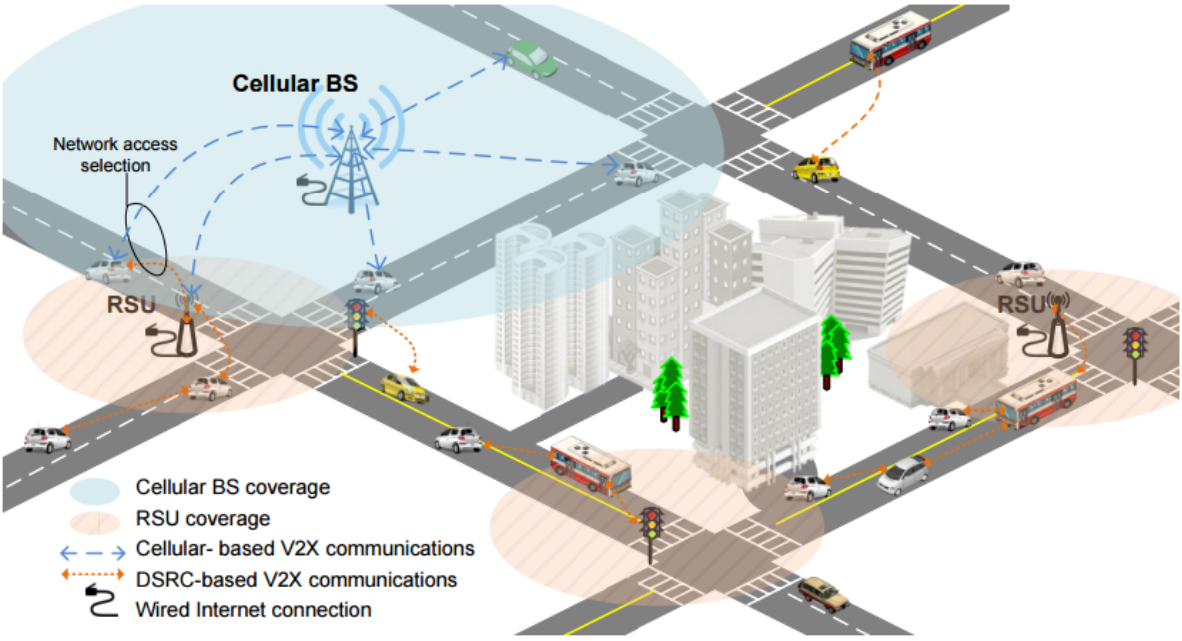

Figure 2.1. DSRC-Cellular Hybrid Scenario for V2X Communications [9].

- Unicasts the received safety messages to the relevant vehicles subject to the the initial sender's safety. The Work in [14] shows that in case of unicast of messages from the BS, the uplink has lower traffic load than the downlink and it becomes a bottleneck. One affordable solution that is available in 3GPP and higher standards is the use of multimedia broadcast and multicast services (MBMS), and the evolved MBMS (eMBMS) for safety message dissemination over cellular communications.

- Broadcasts the received safety messages to all the vehicles in its cell coverage area. Ultimately every vehicle receives all the safety messages sent by the BS, regardless of whether or not it belongs to the same zone of relevance to the sending vehicle. An alternative solution that might have a high latency is to create multicast groups [5] with eMBMS that are concerned with the safety messages that the BS transmits.

2.1.1.3 Traffic modeling and networking of Safety and Non-Safety Applications

To better understand how the traffic density in vehicular networks affects the reliability of the safety applications in event-driven and periodic applications, many studies [53] allow safety applications to optimize the CCHI while applying different types of workloads composed of various safety applications. Many types of safety applications depend on different types of data [15].

In the VANET WAVE Protocol, every CCH period supports sending only one beacon packet for every vehicle in every channel period. The key point is how to prevent and detect the collision caused by hidden or exposed terminals in the broadcast mode especially that there are no Request To Send (RTS), Clear To Send (CTS), Acknowledgements (ACKs) and re-transmission packets as in the case of IEEE 802.11 wireless networking protocols. There are many types of safety applications that are used to increase the safety of roads by exchanging information regarding the traffic, vehicles' statuses, road conditions and many other details. These applications include:

- V2I safety applications [37] related to the communications between vehicles and road infrastructure. They include Stop Sign Violation Warning (SSVW) [95], Railroad Crossing Violation Warning (RCVW), Spot Weather Information Warning (SWIW), Oversize Vehicle Warning (OVW) and Reduced Speed Zone Warning (RSZW).

- V2V safety applications [47] that are executed by vehicles and received by their neighboring ones. They include Forward Collision Warning (FCW), Electronic Emergency Brake Light, Do Not Pass Warning, Left Turn Assist (LTA), Intersection Movement Assist (IMA), Blind Spot Warning (BSW) and Lane Change Warning (LCW).

The authors in [15] studied the packet reception rate of various scenarios each composed of a mixture of different safety applications. Packet reception rates in VANETs charac-

terizes the successful transfer probability of safety packets sent between vehicles and it is calculated to achieve the overall successful probability of transfer. The probability is based on the following three conditions: successful transfer probability of a packet when no other vehicle is sending, the successful transfer probability of a packet when left side senders are sending packets, and the successful transfer probability of a packet when right side senders are sending packets.

Asgari *et al.* [15] proposed that Based on the Poisson Distribution rules, the workload of a running application on a vehicle can be characterized by one parameter λr which is the exponential distribution parameter of the time between packets received by other vehicles and time of the presented mathematical model. A very challenging problem is to allow vehicles to access the channel with a derived optimal probability in a way that maximizes the successful transmission rates. It also modifies DSRC parameters based on the conditions of the road and network conditions to allow the co-existence of safety and non-safety applications with lower congestion and higher successful transmission rate.
In [125], authors presented a cloud-based network selection scheme to ensure that vehicles on the move are assisted and are able to make decisions to select the best network after having a wider network awareness scope. The developed scheme considers the high mobility of vehicles and the fast changing network topology as well as the changes adapted to the communications channel leading to unstable Received Signal Strength (RSS). Xu *et al.* [125] developed a coalition formation game model for vehicular networks with a fast convergence algorithm tested with three vehicular scenarios that demonstrate a realistic range of network topology, resource availability and mobility patterns of vehicles. It uses a distributed database that holds and periodically updates the networks' statistics, road traffic flow information and network maps. Vehicles can consequently apply a network selection algorithm to determine the most suitable network that maximizes their flow of application packets from and to the cloud.

2.1.1.4 Vehicular Network Architectures

In this section, we survey the most recent research progress on vehicular network architectures, as detailed in [70]. This can be classified into three major categories based on the parts intervening in the network:

- Stationary cellular and/or RSU based networks used as access points to provide services to vehicles, such architecture is detailed in [27].
- Pure VANET formed by parked [50] or moving vehicles [33].
- Hybrid network architecture that combines cellular and RSU with mobile vehicles as ad-hoc networks, as described in [92].

Vehicular networks based on fixed infrastructure are eventually unfeasible because of the high cost associated with infrastructure deployment. Most of the research attempts focus on VANETs and hybrid networks. VANETs are created by applying the principles of mobile ad-hoc networks (MANETs) for spontaneously creating a wireless network of moving vehicles to increase safety through DSRC communications or collaboration to better utilize the communications capabilities of vehicles in the same area. VANETs support a wide range of applications that vary from simple one hop [73] information dissemination of CAMs [109] to multi-hop [70] dissemination of messages over vast distances [54]. VANETs can be divided into three types depending on their building blocks; namely, connected vehicles that rely only on V2V communications, coalition of RSUs and OBUs with V2V and V2I communication, and interconnected infrastructure equipment (e.g., Traffic light, pavement markers, cameras, etc.). In contrast to MANETs, modeling the moving nodes in VANETs is less complex especially that most of vehicles are restricted in their range of motion by paved highways/roads with traffic lights. The key issue of network selection is when vehicles are connected to many types of infrastructure nodes like RSUs with DSRC communications access and/or cellular nodes with 3G/LTE access

and/or WiFi. Thus, it renders the decision making regarding the best type of connectivity very critical given vehicular application requirements, cost, availability expectations, etc. These infrastructure nodes have to be utilized efficiently in the best manner that reduces handover latency from one network to another and to avoid bottleneck network congestion. In [84], authors studied the use of IPv6 based on its efficient Internet traffic flow management over heterogeneous technologies. Their model is based on using the RSUs to act as IPv6 routers and to handle the convergence of DSRC and cellular technologies while vehicles get in or out of their coverage.

2.1.2 Vehicular Cloud Computing (VCC)

Realizing a standardized VCC is not an easy task to achieve due to the various proposed architectures and the many challenges related to various provenances of computing and virtualization capabilities.

2.1.2.1 Vehicular Cloud (VC) Architecture

Based on the description in [123] and in [12], the overall VCC architecture relies on three layers:

- Inside-vehicle or Tier-1 cloud formed by physical resources.
- Vehicular and infrastructure communications or Tier-2 cloud.
- The abstraction of computational resources and the formation of VCC similar to a traditional Back-End Cloud (BEC).

The inside-vehicle layer [58] is responsible for monitoring the health and mood of the driver and collecting information inside the vehicle. The collected data can be pressure and temperature by using body sensors, environmental sensors, smart phone sensors, the vehicle's internal sensors, Inertial Navigation Sensors (INS), and driver behavior recognition to predict the driver's reflexes and intentions.

Later, all the information collated via sensors should be sent to the cloud for storage or for use as input for various software programs in the application layer. It is assumed that each vehicle is equipped with an OBU and that is considered as the key-stone for computation in the VC. Equally important, the OBUs have broadband wireless communications capabilities to transfer data through 3G or LTE cellular communications devices, Wi-Fi, WiMAX, or WAVE. The OBU is connected to the sensors, GPU, GPS, controller, etc. through a central gateway and form a complex in-vehicle network [20]. The next layer of this architecture is responsible for the communications. This layer covers V2V and V2I communications. The V2I component of this layer is used to augment the safety level of vehicles on highways by reducing the percentage of crashes, delays and congestion, improve mobility, and provide Wireless Roadside Inspection (WRI) to automatically inspect commercial vehicles. The cloud is the last layer of the VCC architecture that allows for massive and complex computations in minimal time. The cloud layer consists of three internal sub-layers: application, cloud infrastructure, and cloud platform.

2.1.2.2 Vehicular Cloud Computing (VCC) Challenges

The computing resources of vehicles that remain underutilized are combined to create a vehicular cloud composed of vehicles in parking lots [50]. In [106], authors propose forming a vehicular cloud by using the microdatacenters in the RSUs so that every RSU offers its capabilities to run infotainment applications. In [13] the VC is able to merge autonomously the moving OBUs to offer services. Whether the VC is static or in motion, as presented in 2.2, many challenges still remain on how to manage these types of clouds with unpredictable availability of computational resources as vehicles enter and leave the cloud. Moreover, handling VMMs with the lowest cost possible and across heterogeneous VCs is a requirement. The technical challenge also lies in the fact that the vehicles come and go, which makes the assignment of computing resources to tasks difficult. In addi-

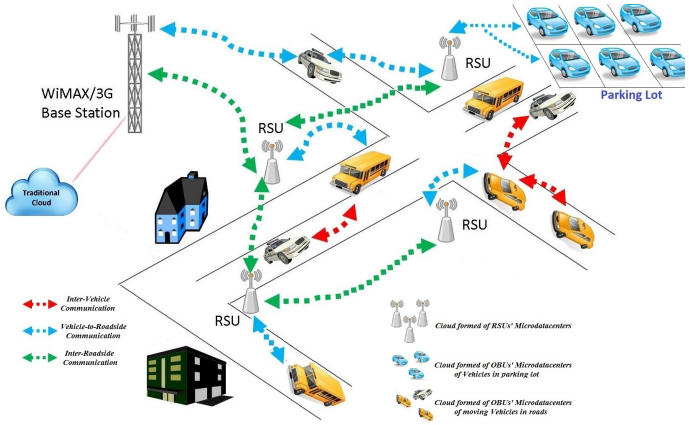

Figure 2.2. Different Architectures of Vehicular Clouds.

tion, there is a tremendous need for fault tolerant job assignment strategies to mitigate the effects of resource volatility and unavailability in VCs, as detailed in [50].

This can be achieved by offering a theoretical analysis and prediction of the expected job completion time in both cases when vehicles do not leave during a checkpoint operation and when vehicles leave while checkpointing is in progress, which leads to a loss of VMs context and to the failure of the system. One of the solutions proposed in the literature is referred to as checkpointing where the state of the computation is saved periodically while jobs are still in progress and the most recent image, or also called copy, of VM is used to rollback the computation in case of a system failure. The contribution of the authors in this paper is focused on saving the state of the computation of a VMM by a vehicle only when needed.

This puts less overhead on the system, especially that most of the checkpoints are not necessary which may increase the efficiency of the vehicular cloud. Ghazizadeh *et al.* [50] designed a model for deterministic environment in terms of vehicle departures and arrivals to the network. The authors explored also the idea of assigning each job to two vehicles for fault-tolerance purposes and taking a checkpoint only when one of them leaves the parking lot.

The proposed solution seems efficient in terms of minimizing the packet losses of appli-

cations when vehicles leave the cloud but it has a significant disadvantage as each task is being executed twice at the same time which decreases the effective throughput of the cloud.

In [126] the authors investigate the resource management and sharing problem for bandwidth communication and computing resources to handle mobile applications in cloud-enabled vehicular networks. Cloud SPs cooperate to form coalitions of cloud resources to share their idle resources with each other. The coalition game model or scheme based on two-sided matching theory for cooperation among cloud SPs. The resources can be better utilized with an improved QoS for users.

The Cloud Service Provider (CSP) reserves a certain amount of long-term bandwidth from network providers and reserves some long-term computing resources (eg., CPU, Memory, Storage, Bandwidth, etc.) from data-centers, which are owned by CSPs. When the CSPâs available resources are limited and do not satisfy the vehicular application requirements, the service quality and user experience degrade. With this in mind, the solution is that vehicles share their storage, bandwidth, and energy resources with each other to secure, store, carry and forward network data for the mutual enhancement of the overall performance of a network of the cloud environment.

The Vehicular Cloud Service Providers (VCSPs) using the computation and communications capabilities of the underlying OBUs have to employ efficient resource management and sharing schemes to meet the requirements of the applications on the street, while taking into account the main differences between Mobile Cloud Computing (MCC) and VCs especially the high mobility of vehicles. It is considered as a compelling business model in the cloud market to know the cost of resources to be rented and the reward of resources offered to the cloud. Having a combination of prices with resource demands in cloud-enabled vehicular networks is implemented as a coalition formation application using the two sided matching theory.

This application also helps to improve the fairness of transactions and optimizes the

resource utilization and cost of the VCSP. Therefore, building a win-win situation by rewarding, as in [68], vehicles agreeing to become a cloud member.

Additionally, some applications have strict deadline constraints, which causes a problem with decision-making for VC selection to host the applications especially with unstable VC resources. In [13], two major issues that are considered related to the cloud availability and lifetime, and the applications processing time. The high mobility of vehicles over time reflects the dynamic changes of resource availability over time, which similarly implies many issues regarding the management of the the dynamic VC's resources to guarantee that tasks are completed with minimum cost before their deadlines and within the lifetime of the VC. In contrast to the previously discussed work, authors in [13] assume that the computing resources of the OBUs are limited and could merge autonomously, flexibly and dynamically to offer services to authorized users. The proposed architecture of the dynamic VC consists of a Cloud Directory Application Repository (CDAR) that contains a list of all applications, all tasks that compose applications and application deadlines, vehicular cloud communications (e.g., WAVE, 4G or 5G Based).
In addition, the architecture is based on a broker that first announces invitations for participation in the cloud, receives responses of the vehicles containing the type and the amount of the available resources and finally registers the vehicular cloud in a CDAR in its area and specifies the properties of resources, lifetime of cloud and costs of resources.

In addition, one major concern in VC is the cloud formation and VMM specifically with the most challenging type of VC that has mobile micro datacenters mounted on vehicles' OBUs and with RSUs acting as head of cloud. In [102], authors have developed dynamic VMM for dynamic VCs. When a vehicle is moving it may exit one network and enter another. If it is still in the same area or still in the communications range of the same RSU, there is no need for VMM. Otherwise, there will be a pressing need to effectively migrate the virtual machines running those applications in order to keep them in the same network. The authors proposed a new scheme called vehicular VMM

which is used to better achieve an effective handling of frequent changes in the data center topology and host heterogeneity while keeping minimal RSU intervention.

2.2 Autonomous Driving: Perception and LIDAR Scene understanding and Reconstruction

In this section, we discuss the most successful machine learning techniques for AD approaches using different input modality sensors, as presented in Fig. 2.3. With this in mind, the classification of recent work falls into the following broad categories: mediated direct perception approaches, behavior approaches, reflex approaches, and LIDAR perception approaches. Then, we present recent visual-based and LIDAR-basd odometry for vehicle localization and mapping based on raw data with non-recognized objects in scene, based on camera-only feeds, LIDAR-only point cloud data, and mixed approaches combining visual and LIDAR inputs for joint vehicle localization and mapping within its surrounding objects.

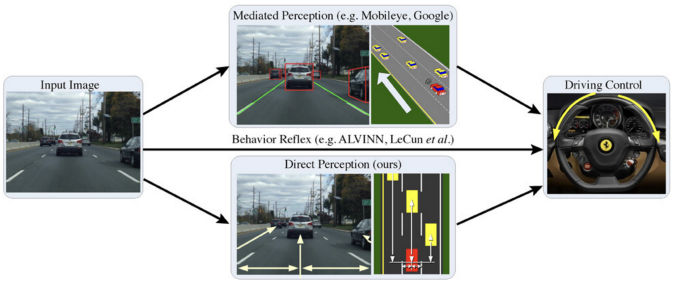

Figure 2.3. Illustration of Three Paradigms of the Autonomous Driving Systems [26].

2.2.1 Mediated Perception Approaches for Autonomous Driving Systems

Mediated perception approaches process the entire scene given by a front-facing camera and parse it in order to recognize driving-relevant objects that impact the driving decisions. This kind of approach depends on independent modules that are used for dif-

ferent specialized recognition tasks, such as lanes, vehicles, pedestrians, traffic lights and signs, free space, etc. [48].

The recognition results of these modules are integrated and combined in a unique world representation to represent the surroundings of the vehicle and from which the final driving decisions are made, as detailed in most of the state-of-the-art systems [94]. The complexity and cost of mediated perception approaches stems from the need to make a total scene parsing and understanding, while only a small number of recognized objects are relevant to make driving decisions from one frame to another.

2.2.1.1 Pixel by Pixel Full Scene Labeling (FSL)

One approach to understand the scene around vehicles is semantic segmentation that labels each pixel in an image with the category of the object to which it belongs. Labeling each pixel of the scene independently from its surrounding pixels is a very hard task to achieve. In order to know the category of a pixel, the labeling process relies on relatively short-range surrounding information and long-range information to understand the context while an object covers more than a pixel. Fig. 2.4 shows the result of FSL on a camera frame from the SIFT flow Dataset.

In other words, to determine that a certain pixel belongs to a vehicle, a pedestrian or to any other class of object, FSL needs to have a contextual window that is wide enough to show the surrounding of the pixel. Consequently, an informed decision can be made of the class of the object that contains the pixel. Techniques based on Markov Random Fields (MRF), Conditional Random Field (CRF) and other graphical models are presented in [55], [66], [89] to guarantee the consistency of labeling the pixels in the context of the overall image. In addition, authors in [51], [63] and [116] developed various methods for pre-segmentation into super-pixels or segment candidates that are used to extract the categories and features characterizing individual segments and combinations of neighboring segments.

Farabet *et al.* [46] developed a FSL technique that includes detection, segmentation and recognition of all objects in the scene. It uses a large contextual window that labels pixels and reduces the requirements for postprocessing methods by using a multiscale Convolutional Neural Network (CNN) that is trained from raw pixels to extract dense feature vectors that are encoding regions of multiple sizes centered on each pixel of the image.

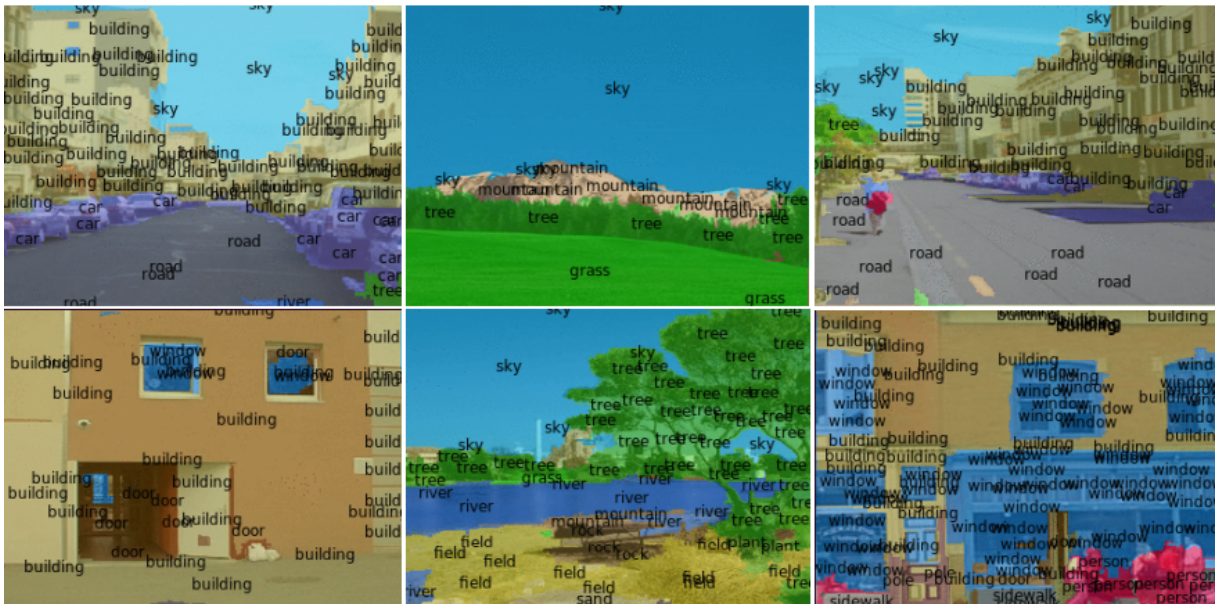

Figure 2.4. Results of FSL on SIFT Flow Dataset [45].

A complete detailed version of the same scene parsing architecture is developed in [45]. The key idea behind the proposed mutiscale dense feature extractor is to generate a series of feature vectors of regions with multiple sizes and centered around every pixel. The CNNs that are fed with raw pixels with end to end training have copies of a single network with the same weights. These networks are applied to multiple scales of a Laplacien pyramid version of the input image. The trained networks offer features that produce efficient multiscale representations for FSL to capture texture, shape and contextual information. Even though the learned multiscale representations allow for the detection and

recognition of regions and objects contained in the scene, they do not draw the boundaries of the regions accurately.

In [18], the authors were the first to utilize the CNN architecture for semantically segmenting the pixel of the images using a pixel-wise classification layer on top of the the decoder network that follows the encoder network. Impressive results of the pixe-lwise classification are presented in Fig. 2.5.

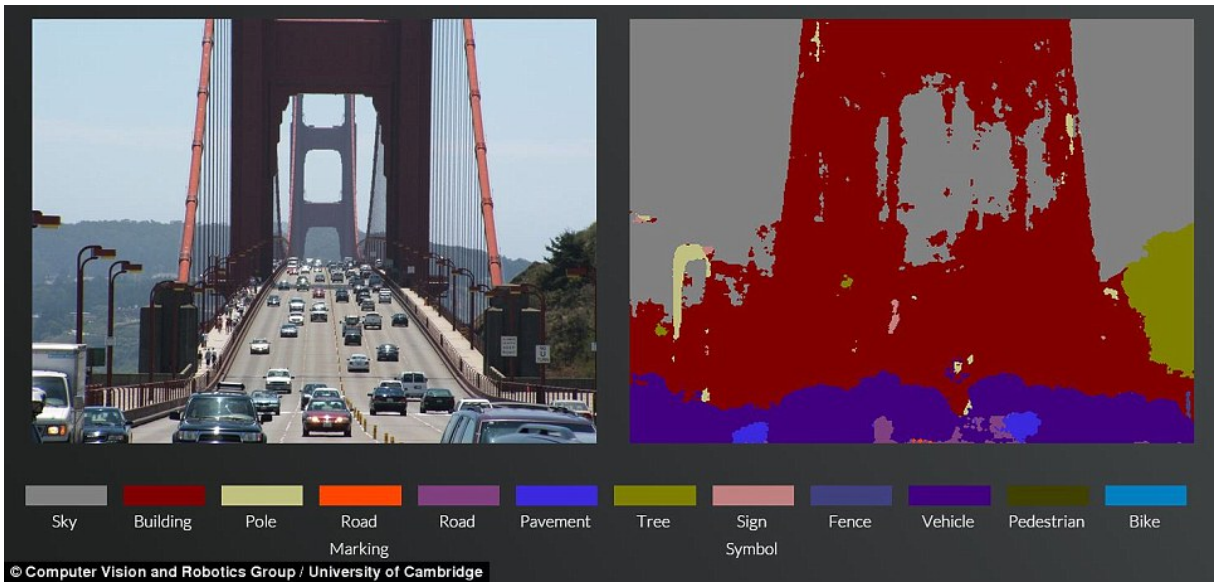

Figure 2.5. Pixels Labeling Based on Different Class of Objects [18].

2.2.2 Direct Perception Approaches for Autonomous Driving Systems

The direct perception approach was initially introduced by [26] to predict the driving decisions, instead of learning to map images containing different situations with the steering action or parsing and recognizing the scene and deciding the control actions of the vehicle. The developed model is built on top of a CNN that automatically derives indicators of situation of the road, such as the distance between the vehicle and the lane marking, angle of the vehicle relative to the road and distances to other surrounding ve-

hicles. Additionally, the adopted and trained CNN learns to extract image features and makes meaningful predictions and description for scene understanding and autonomously drive the vehicle by the help of a simple controller.

2.2.3 Behavior Reflex Approaches for Autonomous Driving Systems

Behavior reflex approaches [99] are drastically different from mediated perception approaches and direct perception approaches by constructing a direct mapping of the sensed input to the corresponding driving action that has been taken. One former technique is developed in [8] that generated a neural network to create a direct mapping from input images to corresponding steering angles in order to be directly applied to learn driving actions. To learn the model, the system uses the images and the actions representing the steering angles as the training data. It has been proven that this technique can handle lane navigation, but failed to realize accurate decisions in a complicated traffic scenario. As opposed to the Autonomous Land Vehicle in a Neural Network (ALVINN) system developed in [99] that uses fully-connected networks, the end-to-end DAVE-2 system proposed by NVIDIA in [21] uses a CNN that directly maps the raw pixels of images from cameras to the steering commands. DAVE-2 makes use of CNNs that offers more than pattern recognition, as it is trained to learn the whole processing pipeline, to find steering decisions based on training video data collected while driving from two cameras and coupled with left and right steering decisions.

2.2.4 LIDAR Perception Approaches for Autonomous Driving Systems

The work in [34] developed a solution based on LIDAR data only to explore the vehicle surroundings. This solution involves three consecutive phases; namely, segmentation, fragmentation detection and clustering, and tracking. The developed solution is built

based on combining multiple artificial intelligence techniques and it is conceived for detecting and tracking objects of any shape that are going to be used by autonomous vehicles to find and understand the scene flow of its surroundings as shown in Fig 2.6.

Figure 2.6. LIDAR-Based Scene Understanding.

2.2.5 Mixed Mediated and LIDAR Perception Approaches for Autonomous Driving Systems

The autonomous driving platform developed in [87] presents an interesting and challenging real-world computer vision benchmarks that include stereo, optical flow, visual odometry, 3D object detection and 3D tracking of real time surroundings of autonomous vehicles. The data includes the LIDAR and the camera recordings of driving sequences as shown in Fig. 2.8.

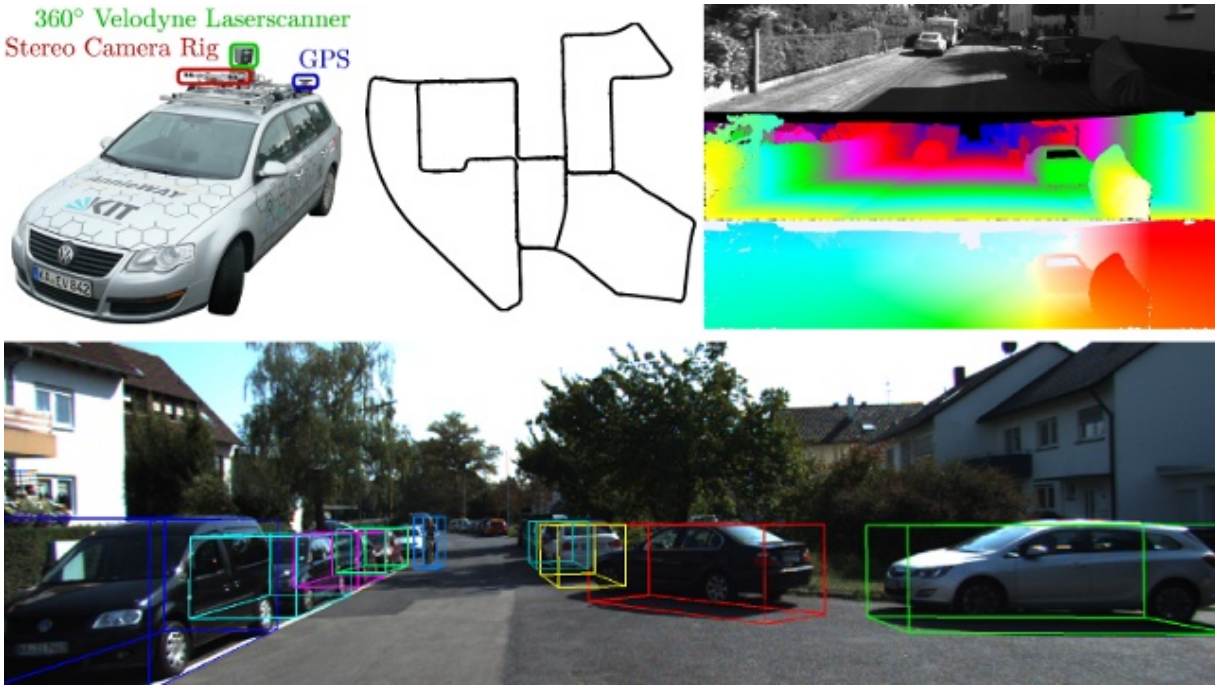

Figure 2.7. KITTI Benchmark Suite for creating the Dataset [87].

2.3 Localization and Mapping For Self-Driving Vehicles

Traditional localization approaches such as GPS are inaccurate with a significant localization error in addition to the total unreliability because of the insufficient network coverage. Digital driving instructions have to be ultra-precise since driving requires steering commands to be in consensus with the deep-learning and mapping of the outdoor environment.

Consequently, fully AD systems opt to use high-precision maps or also called HD maps to localize a vehicle in the space and within its surrounding objects in the scene. Automatically created high-precision decision maps put together various enriched multi-model sensed data and a pre-mapped environment that can be from raw LIDAR, Video or mixed sources.

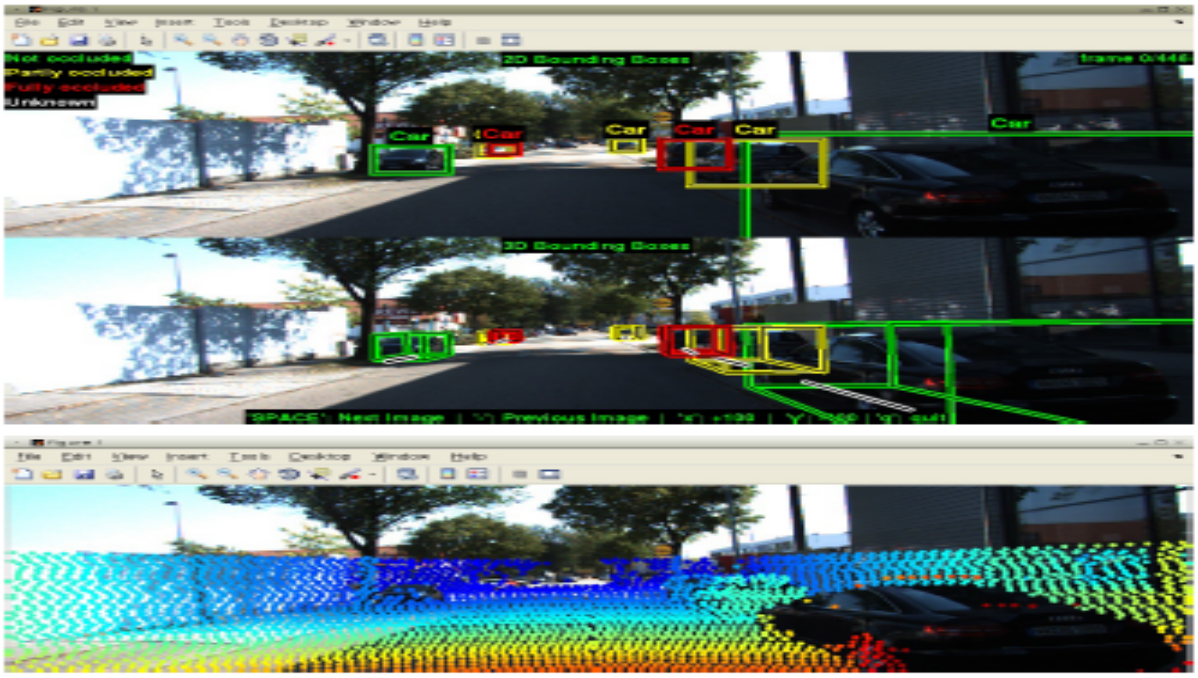

Figure 2.8. Real-Time 3D Object Detection and Tracking.

As opposed to standard definition maps that are based on GPS positioning such as Google Maps and used for machine to human communications, HD maps are fundamentally different with a better resolution and are meant for machine-to-machine communications. Its relative localization systems are supposed to provide an approximate maximum localization precision of 25 cm, in comparison with a 2 to 10 meter accuracy in GPS, without the need for external error detection and correction systems.

Major research attempts are discussed in this section to gain a better insight into the real practicality of SLAM [39] for localization of self-driving vehicles through odometry [93] that uses data irrespective of sensing modality to approximate the changes in a vehicle's position over time.

2.3.1 Visual Odometry for Vehicle Localization

Scanned roads by stereo cameras overlaid over high precision positional data produce a highly detailed representation and localization. In this context, Raul *et al.* developed

in [90] a novel feature-based monocular SLAM system that requires creation of initial maps since depths may not be found from a single image frame in order to provide wide baseline loop closing detection and to serve motion clutter. The developed solution covers the tracking, mapping, relocalization and loop closing based on a novel strategy that selects specific key important points and keyframes to generate trackable maps that grow with scene content changes.

Every Map point holds the 3D position of the world coordinate system, the viewing direction consisting of rays joining the point with the optical center of the keyframes observing it, and a representative ORB descriptor holding the smallest hamming distance among all associated descriptors in keyframes where the same point is observed.

In addition, the proposed scheme ensures that every keyframe stores camera principal points and focal length, ORB features that were extracted from the frame independently of whether or not they are associated with map points and the camera pose that represents rigid body transformation of points from real-world coordinates to camera coordinates. Then the co-visibility graph, the essential graph, ORB descriptors extraction and the automatic map initialization are achieved dynamically in order to satisfy finding initial correspondences and making the parallel computation to find inliers between models. Fig. 2.9 shows points and keyframe trajectory, ground truth and trajectory and after many iterations of full bundle adjustment.

A continuous estimation of vehicle's current position and orientation depending on the observed environment from the camera is presented in [118] as shown in Fig. 2.10. It enables SLAM with rigidly coupled frames of Multi-Camera Systems (MCS) by taking advantage of the MultiCol Model and introducing:

- The usage of Multi-Key Frames (MKFs) with identification and match points across them and mask outliers.
- Hyper-graph formulation of MultiCol for different tracking and modeling pipeline

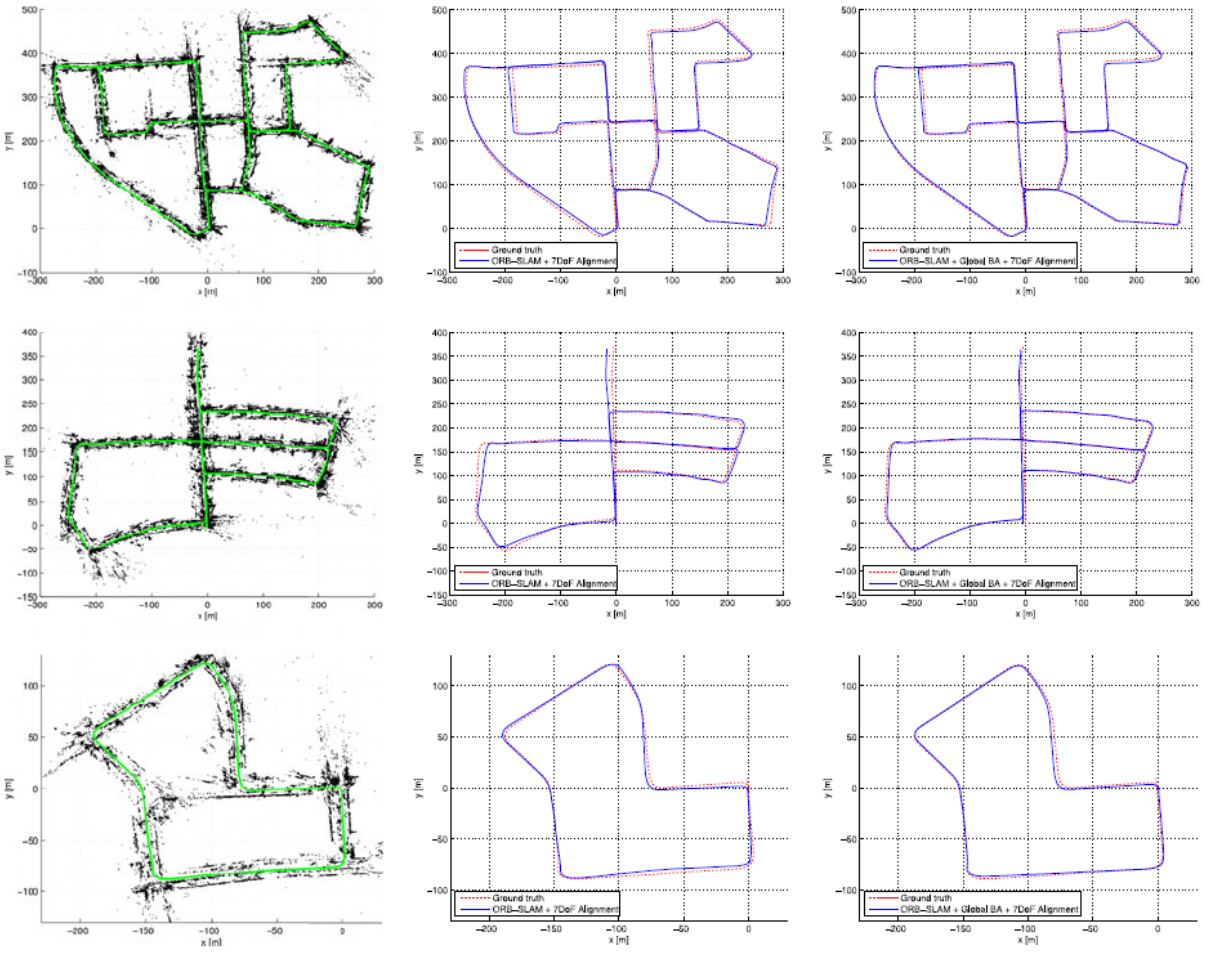

Figure 2.9. Sequences 00, 05, and 07 From the Odometry Benchmark of the KITTI dataset. (Left) Points and Keyframe Trajectory. (Center) Trajectory and Ground Truth. (Right) Trajectory After 20 Iterations of Full BA [90].

than used in [90] where graph edges connect two vertexes only and now edges can connect to any arbitrary number of vertexes.

- Advanced Multi-camera loop closing that searches eventual loop closures from newly presented MKF in order to figure out if a place has been visited.

Depiction of the loop closing, applying similarity transformation to possible alignment of points and usage of the alignment error to correct the remaining MKF poses and points mapping by projection to map are generalized in Fig. 2.10.

Figure 2.10. Loop Detection Example on an Uncoinciding Spatial Alignment of Local Map With Historic Map [118].

The authors in [98] presented a stereo SLAM localization system called S-PTAM that uses only a stereo camera to avoid the monocular bootstrapping problem as in [90]. To improve robustness, the authors used to enforce stereo constraints on the pose- and map-refinment algorithms and applied a maintenance process for each independent thread by running a map bundle adjustment refinement algorithm in local area to improve global consistency.

Fig. 2.11 draws the estimated trajectory discovered by S-PTAM compared to ground truth and to generated maps while appiled to the sequence 00 of the KITTI dataset.

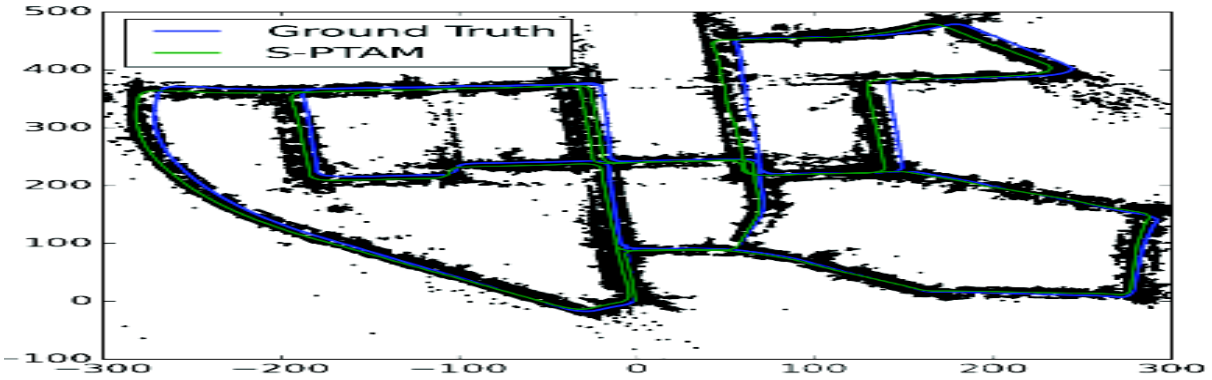

Figure 2.11. Estimated Trajectory Performed by S-PTAM on KITTI Dataset 00 Compared to the Ground Truth as well as the Generated Map [98].

To achieve the creation of reliable visual environment systems, the authors in [44]

present a scheme for tracking keypoint and estimation of egomotion and the structure of the environment from the trajectories of the selected keypoints.

Fig. 2.12 presents an overview of the pipeline of the solution that first consists of decomposing a sequence of images from a monocular camera setup into consecutive frames and applying a propagation based tracking method to find the 2D trajectories of the keypoints and not through finding keypoint correspondences like in [98].

The search of the keypoints matches from one frame is achieved by propagating the estimated 3D position of each keypoint from one frame to another and by making sure that the photometric consistency is verified.

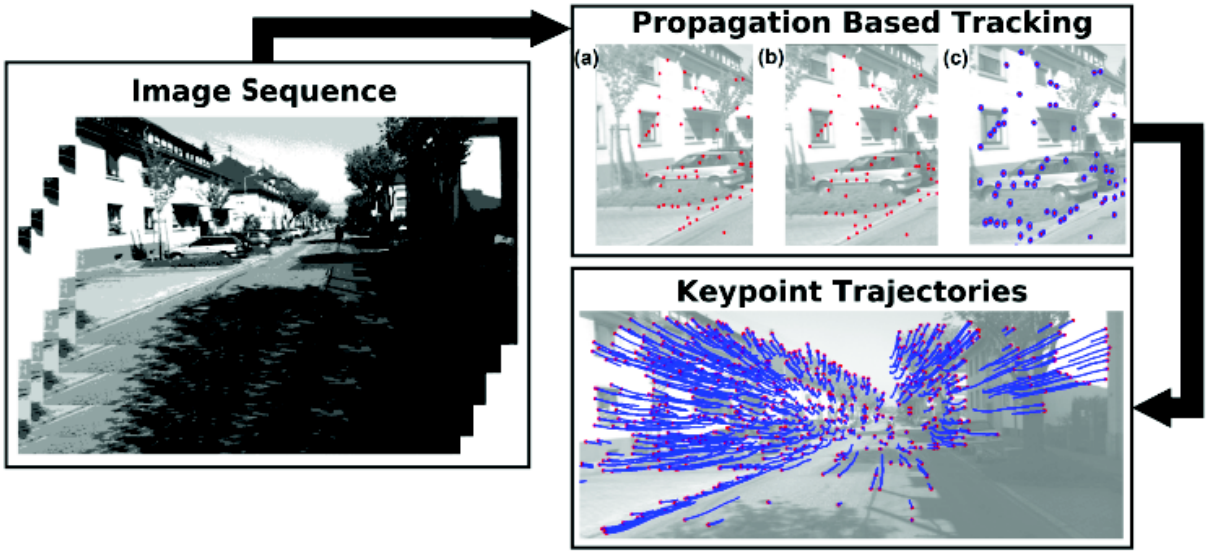

Figure 2.12. System Steps Modules From Sequence of Images to Propagation Based Tracking and Finally Keypoint Trajectories. The Keypoints (a) and (b) in Frame n-1 and Frame n, While Keypoints (c) is the Predicted Keypoints in Frame n+1 [44].

2.3.2 LIDAR-Based Vehicle Localization

In general, the rotational movement of LIDAR scanners and having the vehicle on the move too, make the registration and localization of the point cloud very difficult.

The authors in [129] proposed a solution that uses odometry measurements to achieve low-drift and low-computational complexity by applying two algorithms. The first algo-

rithm is applied to perform the odometry with low fidelity and high frequency for LIDAR velocity estimation. The second algorithm is applied to run with a lower frequency to perform both fine matching and registration of the point cloud data.

The LIDAR odometry algorithm uses last sweep point cloud, current sweep growing point cloud and the pose transform from last recursion as inputs parameters. It extracts feature points, finds its correspondence and applies motion estimation to a robust fitting, as presented in Fig. 2.13. The mapping algorithm is used to match and register the re-projected point cloud to a specific time stamp, based on the undistorted point cloud and the pose transform that contains the LIDAR motion during the sweep generated by the LIDAR odometry algorithm. LOAM solution is ranked #3, with a 0.70 % , on the KITTI odometry benchmark for the average computed translational and rotational errors.

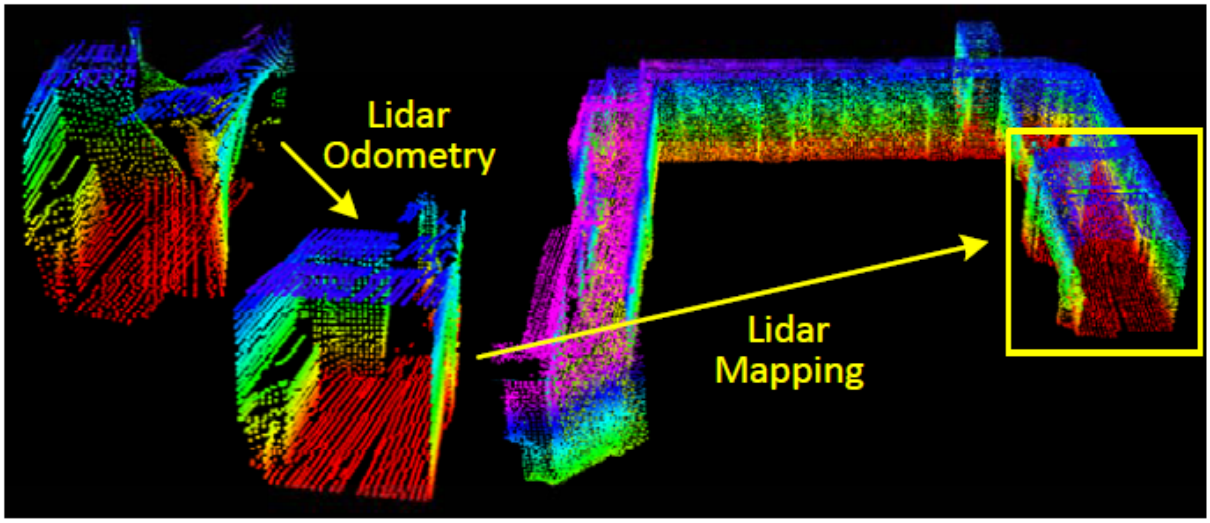

Figure 2.13. The Developed Solution Decomposes The Problem Into Two Parallel Algorithms: The Odometry Algorithm Used to Estimate the Velocity of LIDAR and to Correct Distortion in the Point Cloud, then, a Mapping Algorithm Matches and Registers the Point Cloud to Create a Map [129].

The authors in [29] developed a novel idea of synthetic 2D LIDAR in order to achieve precise vehicle localization on a virtual 2D plane. The presented Monte Carlo Localization to estiamte the position of the vehicle replies on the synthetic LIDAR measurements and odometry information. A demonstration of the accuracy and robustness of this approach

is demonstrated through carrying out real-time localization of the driving test in the NUS campus area, as presented in Fig. 2.14.

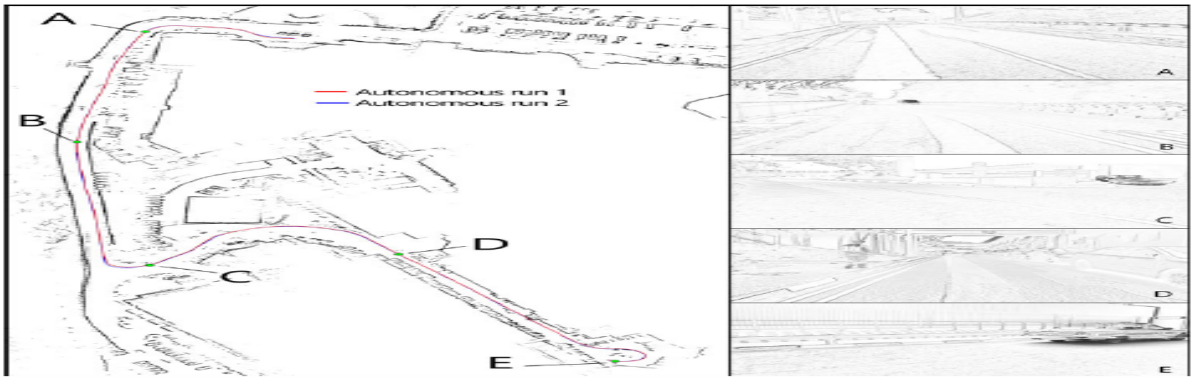

Figure 2.14. Autonomous Navigation With Synthetic Virtual LIDAR. Images on the Right from Top to Bottom Correspond to Visual Validation of Localization Repeatability From Checkpoint A to E [29].

In [22], the authors studied an accurate and reliable vehicle localization system by using the landmarks acquired from LIDAR mapping systems. Since the standard Global Navigation Satellite System (GNSS) solutions lacks high reliability, the developed solution relies on relative localization using LIDAR sensors and a pre-mapped environment. Consequently, the creation of detailed environment maps is a necessary step to produce accurate localization. The obtained landmark pairs are then associated with each others using an estimation approach, which leads to accurate position estimations of the robotic vehicle, as presented in Fig. 2.15.

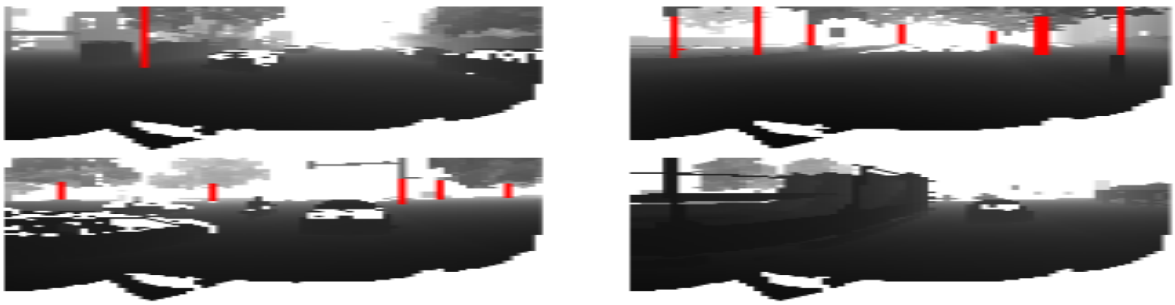

Figure 2.15. Pole Detection Results With Black is Near, White is Far (or error), and Detected Poles are Marked in Red [22].

Additionally, more advanced methods were incorporated, such as using dead reckoning based on multiple matching successive scans as presented in SLAM approaches. Alternatively, the authors discussed other landmark incorporating examples like planar patches, as presented in Fig. 2.16. Then, they investigated in details the matching algorithm with the RMS variations, and the weak configuration points that were automatically detected. This yield smaller error bounds that met the requirement of the design in terms of reliable and automatic association procedure.

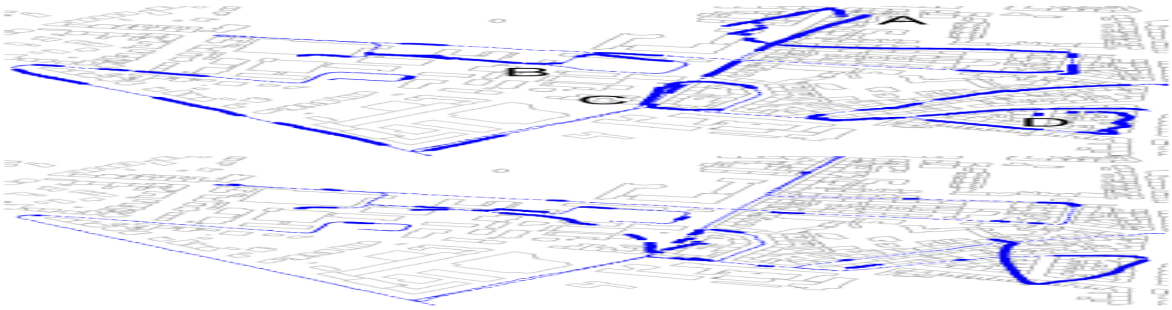

Figure 2.16. GPS Trajectory on Top and Result of Matching on Buttom [22].

2.3.3 Combined Visual-LIDAR Odometry for Vehicle Localization

It is widely common to utilize the dense point clouds with measures of surface reflectivity obtained by LIDAR scans to obtain localization results of centimeter-level accuracy. However, this comes at a prohibitive cost in terms of sensor suites and complex state-of-the-art localization techniques. This is why, the authors in [124] investigated a cheaper model that finds comparable localization accuracy using cameras.

The developed model is built to localize a single monocular camera within an already created 3D ground map from a surveying vehicle equipped with 3D LIDAR scanners, as shown in Fig. 2.17. To realize that, a GPU is used to create several synthetic views of the visited environment in order to achieve maximal normalized mutual information between the GPU and the real camera measures.

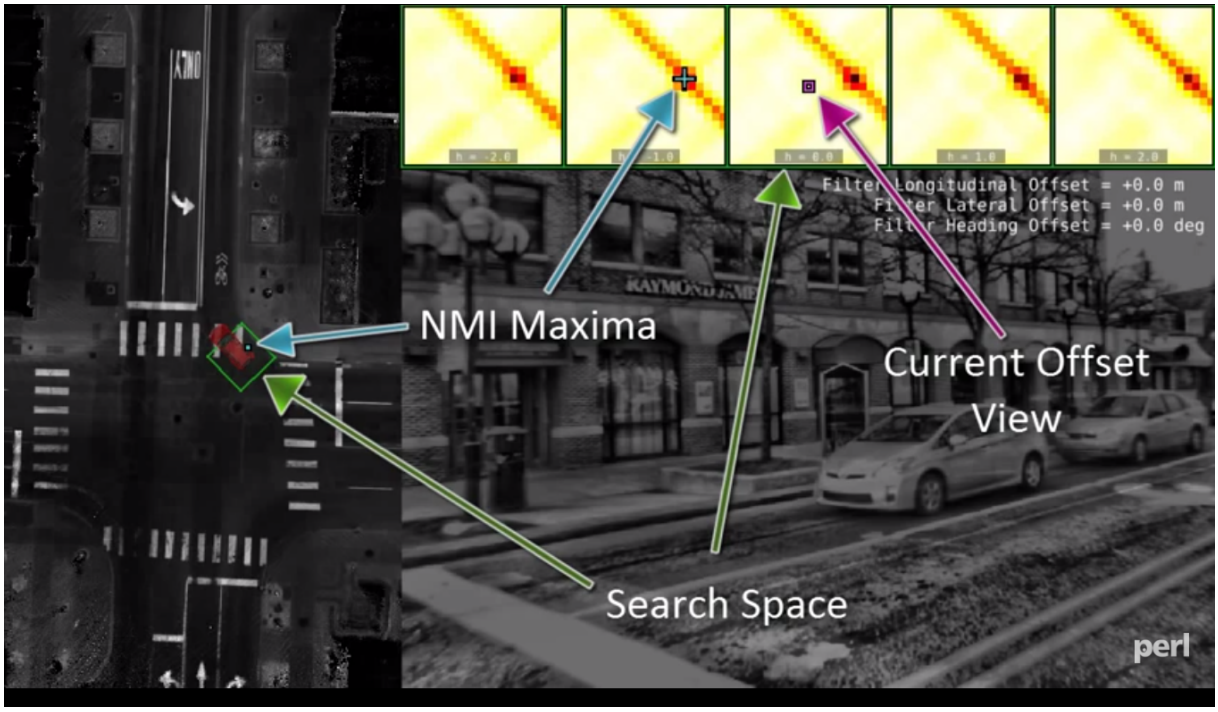

Figure 2.17. Localization Within LIDAR Maps with Birdâs Eye View on Left, Normalized Mutual Information Cost Surface On Top and Alpha Blended Camerasynthetic View on Bottom [124].

In [130], the authors present a general framework for combining visual and LIDAR odometries in a fundamental and novel method. The method shows improvements in performance over the state of the art, particularly in robustness to aggressive motion and temporary lack of visual features. The proposed on-line method starts with a visual odometry to estimate the ego-motion and to register point clouds from a scanning LIDAR at a high frequency but low fidelity. Then, scan matching based LIDAR odometry refines the motion estimation and point cloud registration, simultaneously. In addition to the comparison of the motion estimation accuracy, the authors evaluated the robustness of the method when the sensor suite moves at a high speed and is subject to significant ambient lighting changes.

The overall system is divided into two sections. The visual odometry section estimates frame to frame motion of the sensor at the image frame rate, using visual images with assistance from LIDAR clouds. In this section, the feature tracking block extracts and

matches visual features between consecutive images, as shown in Fig. 2.18.

The depth map registration block registers LIDAR clouds on a local depthmap, and associates depth to the visual features. The frame to frame motion estimation block takes the visual features to compute motion estimates. When applied to the KITTI odometry benchmark for assessing the translational and rotational errors, V-LOAM achieves 0.75% of relative position drift and is ranked #2.

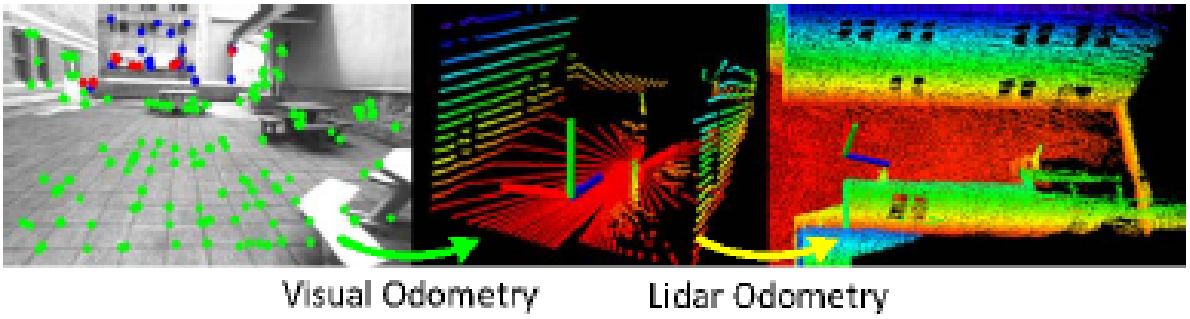

Figure 2.18. Motion Estimation and Mapping are Achieved Using a Monocular Camera Combined with 3D LIDAR [130].

Visual odometry can be augmented with depth information as provided by RGB-D cameras, or from LIDARs associated with cameras. However, such depth information can be limited by the sensors, leaving large areas in the visual images where depth is unavailable.

The authors in [128] propose a method to utilize the depth, even if sparsely available, in recovery of camera motion. In addition, the method utilizes depth by triangulation from the previously estimated motion, and salient visual features for which depth is unavailable. The core of the proposed method is a bundle adjustment that refines the motion estimates in parallel by processing a sequence of images, in a batch optimization.

The method in [128] for the three sensor setups, one using an RGB-D camera, and two using combinations of a camera and a 3D LIDAR. The presented method is rated #2 on the KITTI odometry benchmark irrespective of the sensing modality, and is rated #1 among visual odometry methods, as presented in Fig. 2.19.

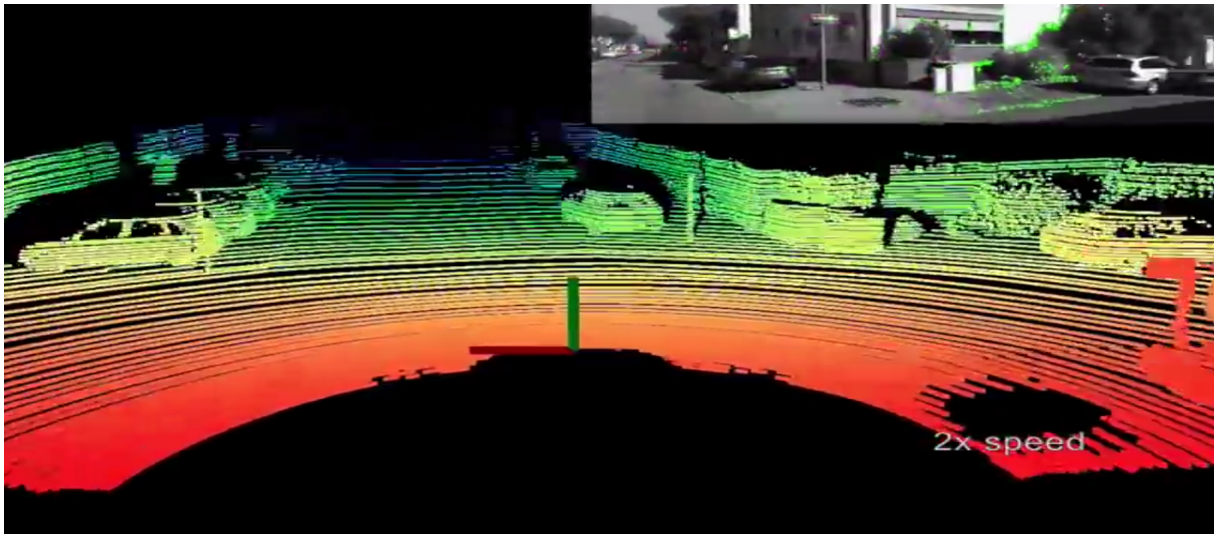

Figure 2.19. Point Cloud Perceived by Velodyne Scanner on KITTI Dataset [128].

In [69] the authors developed a complete solution for high-accuracy localization of vehicles on the move using urban environments' maps. The presented solution aims to generate high-resolution environment maps by integrating GPS, IMU, wheel odometry, and LIDAR point cloud data obtained from the vehicle's recorded data. The solution allows the localization of the moving vehicle within the maps by mainly using a particle filter method for calculating the correlation of LIDAR measurements with the map, as shown in Fig. 2.20.

The map is created by the reconstruction the 3D structure of the environment using infrared reflected laser beams of the LIDAR. More precisely, the experimental results relative accuracy are better by more than an order of magnitude than the ones using traditional methods (e.g., GPS, IMU and odometry). Therefore, the presented solution achieved a reliable real-time localization within 10 centimeter range accuracy during bad whether and without GPS services, as opposed to the non accurate outdoor localization research work based only on GPS.

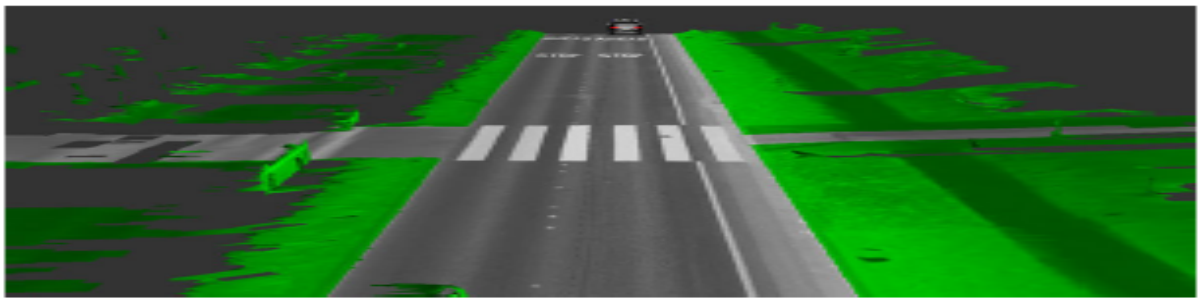

Figure 2.20. Visualization of the Ground Plane Extraction with the Measurements that Coincide with the Ground Plane Are Retained and in Green All Others Are Discarded [69].

2.4 Towards VCs Meeting Autonomous Vehicles

In full alignment with the USDoTâs ITS-JPO vision stated in the RFP, we innovatively blend connected vehicle technology with autonomous vehicle systems. These systems will leverage the power of V2V and V2I communications, to significantly enrich the vehicle understanding of its surroundings and collectively perform driving decisions given the exchanged dynamics, movements, and intents of all vehicles.

In addition to the communications impact, VCs are expected to play a major role in supporting the limited capabilities of vehicles with on the move add-on functionalities. The VC will contribute to creating solid learning formulations to build more accurate 3D scenes of the vehicle surroundings, using the acquired information from multiple input modes, such as cameras, LIDARs, and exchanged V2V/V2I messages.

The targeted multimodal solutions will insert all the information acquired by the input modes into one mapping and learning setting, which is very likely to be the revolutionary HD live maps. It encodes one accurate 3D space representing the surrounding scene including V2V-enabled out-of-sight vehicles and V2I-conveyed sign/signal information, 2D images with rich texture description and accurate object depth in 3D LIDAR point clouds.

2.5 Post Alignment and Fusion

Maalej *et al.* presented in [78] a multimodal framework for object detection, recognition and mapping based on the fusion of stereo camera frames, point cloud Velodyne LIDAR scans, and V2V exchanged BSMs over DSRC. Based on the adapted Darknet's CNN, the pixel-wise adjacency coordinates of moments are derived from the bounding boxes of recognized objects of KITTI frames. In addition, objects are retrieved from point cloud scans to the same camera capture. The generated BSMs corresponding to the vehicles from LIDAR have position (x,y,z) triplet as a replacement to the real-world positioning from (Latitude, Longitude, Elevation). The authors presented a semi-supervised manifold alignment technique to achieve camera-LIDAR and camera-V2V mapping of recognized objects that have the same underlying manifold.

The post-alignement consists of matching correpsonding pairs of recognized objects to reconstruct a 3D world as a virtual map surrounding the self-driving vehicles. Finally, the presented enriched points are the ones that were not mapped as they might correspond to out-of-sight objects, or BSMs of distant cars, etc.

Alternatively in [6], the idea of a cloud-to-car mapping system is presented by Archer Software to have from the cloud the most updated details about road networks like lane curvature, roadway profile, static positions of traffic signs and then recognized objects from the car itself are post-added in order to achieve automatic maneuvering.

In [64], the authors present a novel cloud-assisted design for autonomous driving that is called Carcel, where all the sensed data from self-driving cars and from static roadside sensors are post-aggregated and managed in the cloud. Precisely, the cloud is responsible for conveying obstacles to vehicles after analyzing the received information, which means that the cloud will assist vehicles on roads to make decisions to avoid obstacles that are not necessarily sensed by their sensors. Moreover, the longer range information that the cloud builds about the traffic patterns enables more efficient path planning. This method

is not realistic due to the huge amount of data that is sent from and to the cloud, which delays the steering decisions arriving to vehicles.

2.6 Pre-fusion between VC and AD Data Modalities

As opposed to the post-fusion and mapping systems, the pre-fusion systems do not rely on having seperate recognized data from sensors or cloud that are mapped together. For example, Mobileye's REM system [43] relies first on a pre-existing rich HD maps that are supplemented with to-the-minute data from real-world driving. An advanced faster version for accurate real-time mapping of data is currently under development.

An issue with the pre-fusion of cloud data of in-vehicle sensors is the threat of cyber attacks which may lead to a disastrous reconstruction of surrounding world and consequently to a failing AD system. As detailed in [3] having self-driving vehicles relying on offline system might be the solution to hinder hackers.

More simplistic and secure scheme that does not rely on incoming cloud data is developed in [28], where a Multi-View 3D networks (MV3D) is developed to guarantee accurate 3D object detection applied to autonomous driving scenarios. The developed sensory-fusion framework is used to predict the orientation of 3D bounding boxes representing an object by taking both raw RGB images and LIDAR point cloud without applying any prior recognition. The generated candidates of 3D boxes for localization and detection in the KITTI benchmark outperforms the up-to-date systems by approximately 25% and 30%.

Similarly in [117], the 3D point cloud and 2D front view images are fused via a CNN with the usage of a Region Proposal Network (RPN) that is used in the network's multiple layers to generate proposal region objects in the front view.

2.7 Cooperative and Coordinated Autonomous Driving

In order to fully leverage the autonomy of self-driving vehicles, it is obvious that developing intelligent systems built on top of coordinated autonomy would guarantee additional safety with more trusted manoeuvres, reduce fuel consumption and balance network traffic while guaranteeing a cooperative dispatching of vehicles depending on time and positions. In this context many ongoing research projects on vehicles coordination are being developed.

EPFL's research team working on the AutoNet2030 [23] presented in the design of distributed algorithms among vehicles with different levels of intelligence on borad ranging from fully autonomous to standard vehicles for cooperation among vehicles. The demonstration included an autonomous truck, an autonomous vehicle and an ordinary car equipped with a human machine interface that was able to satisfy cooperatively a merging scenario on a highway by offering enough space to a vehicle intending to merge in-between the two others. The robustness of the algorithms allows its deployability even before having all the vehicles being autonomous.

Another key feature of coordinated driving is being implemented by the authors in [31] consisting of a decentralized trajectory coordination for autonomous vehicles without the usage of widely applied synchronization points that force all vehicles to pass by synchronously. The developed system assumes that a unique priority is assigned to every vehicle and each one periodically informs other vehicles about its planned future trajectory through V2V communications. Consequently, every vehicle will be responsible for maintaining the possible trajectory that will be collision-free with regard to other vehicles with higher priority. The collision-free path is obtained by the trajectory planned that takes into account space, time and a pre-designed network of paths while modeling higher-priority vehicles when moving. A real time scenario validating the developed

method is presented where every conflict between desired trajectories triggers automatic conflict resolution.

2.8 Examples of Integration Systems

NVIDIA's Drive PX [60] AI car computer for self-driving vehicles is considered as a pioneering effort among all the computing platforms for real-time scene understanding, HD mapping, energy efficiency, automated highway driving, accelerated deep learning, sensor fusion, HD mapping, etc. Its GPU data parallel architecture accelerates the perception task of detecting and classifying objects using deep neural networks that are pre-trained in cloud datacenters and deployed in the vehicle.

In addition NVIDIA's recently developed Co-Pilot [110] is considered as an extra AI system built toward fully autonomous systems that is either driving the vehicle or looking out for the driver. In other words, for tricky traffic, changing roads without up-to date maps or highly fuzzy uncertain perception then the Co-Pilot makes sure to increase passenger driver awareness with surrounding environmental awareness. It notifies the driver about objects that she might need to be concerned about, increase the cautiousness of driver about where to look by focusing on the driver's face recognition, head tracking, gaze tracking, lips reading, etc.

NVIDIA's DriveWorks [36] is a powerful Software Development Kit (SDK) for researchers and industry to implement any level of autonomous driving accelerated on Drive PX. Its layered architecture and run-time pipeline framework is detailed in Fig. 2.21 and offers the following:

- Detection Libraries used for various types of sensors processing, fusion for objects segmentation, detection and classification.
- Localization Libraries used for odometry mapping, map localization and HD maps interfacing.

- Planning Libraries to ensure functions of vehicle automatic control, overall scene understanding and intelligent path planning.
- Visualization Libraries offering applications for cluster display streaming, ADAS and debug rendering.

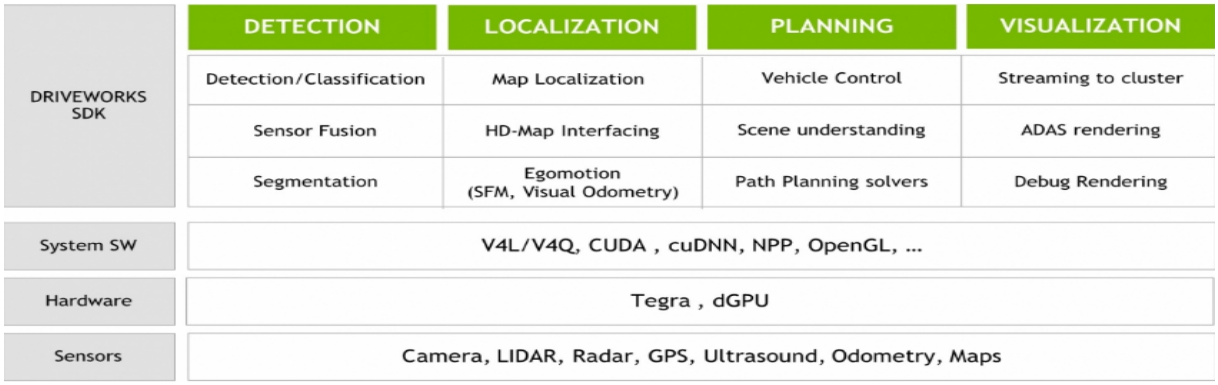

Figure 2.21. DriveWorks SDK Architecture [60].

Partnering with NVIDIA on AI technology, HERE [57] is moving forward to accelerate its HD Live Maps [4] by using DRIVE PX 2 for its self-driving vehicle solutions that are delivered as a cloud service for a digital mapping content. The HD Live Maps is a layered complex solution, visualized in Fig. 2.22, can be depicted as:

- Lane Model: is considered as the key model of the HD maps since it provides a highly accurate representation of the road networks with different lanes' separation and roads' delimiters, all mapper with a 10 to 20 cm precision.
- Localization Model: allows finding the exact position of the vehicle in the lane where it is driving on the HD Map. Interestingly, it allows cars to be map makers since as long as they are being on the road, they collect data about the perceived environment and provide it to the cloud which aggregates all received information to update the maps.

- Activity Layer: plays a major role in understanding dynamic events while driving on the road network such as heavy traffic, accidents, road closures, and other dynamic events that may impact the driving strategies and paths of the vehicle.
- Analytic Layer is used to adapt to tricky scenarios on how to drive by looking at various driving styles that inform the vehicles on how to behave under similar conditions.

As an illustration, HERE HD Live Maps offer intelligent complete solutions needed to pave the road to raise the trust in the commercial sustainability and reusability of automated driving systems by vehicle manufacturers for automated driving solutions.

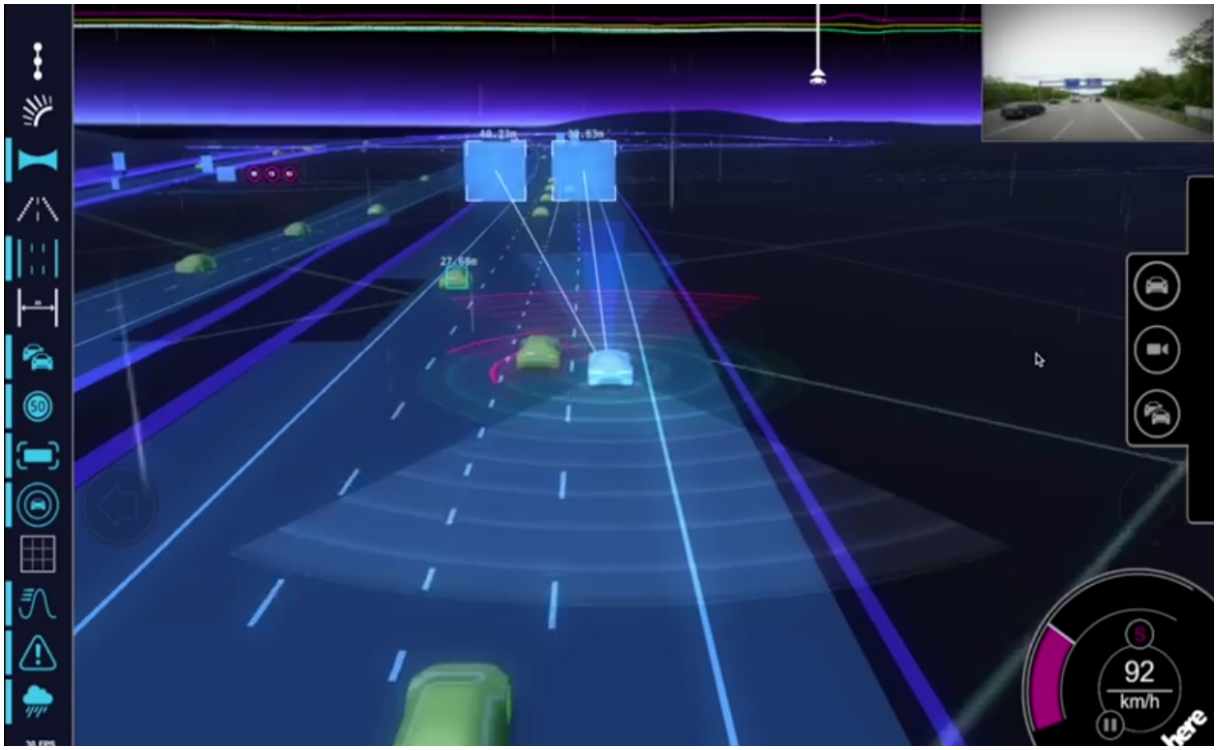

Figure 2.22. Visualization of HERE's Live HD Map.

With cameras, RADAR, LIDAR, unreliable GPS and more sensors, Waymo [122] as formerly known as the Google self-driving car has achieved self-driven 3 million miles by using complex heavy computation mapping and fusion of sensed data while using minimal

access to cloud-based services. Up to now, the self-driving is not commercialized yet even though there have been different prototypes varying from modified traditional vehicles requiring human inputs to vehicles without steering wheels.

Tesla's Autopilot [1] is considered as the most popular full standalone hardware with software solution for autonomous cars that is being actively updated and refined as long as the number of driven miles get higher, more than 1.3 billion miles of data [59]. All cars built by Tesla after 2014 are equipped with hardware autopilot that are capable of receiving updates from the cloud of the core software for self-driving, required drivers for the sensors, etc.

With its particular technology that relies on 8 cameras, surrounding both Model S and X, along with twelve ultrasonic sensors, RADAR and GPS, the Autopilot creates a 3D view of the vehicle's surrounding in every direction that is beyond human access. The enhanced Autopilot offers new capabilities varying from simple speed to traffic condition adjustment to complex lane changing without driver's input. Despite these new technological advancements and market solutions for autonomous driving, most of the aforementioned solutions have at certain places, moments or situations of low confidence of driving autonomously. With such low confidence, self-driving systems require the driver to take-over the commands for driving.

Chapter 3: Wireless Access for Vehicular Environment and IEEE 1609.4

The Wireless Access for Vehicular Environment (WAVE) standard partitions the bandwidth into seven channels of 10 MHz each, one control channel (CCH) to serve safety applications, and six Service Channels (SCHs) to serve non-safety applications. DSRC specifies a channel switching scheme to allow vehicles to alternate between these two types of applications. Moreover, the channel switching scheme has a lot of limitations and needs to be improved and adjusted depending on the network density and the requirements of the applications. WAVE is divided into four categories of 1609 family of standards:

- IEEE 1609.1: standard used as a guide to manage the resources in VANETs like RSUs, OBUs etc.
- IEEE 1609.2: standard defined as the WAVE security services for applications and management of messages that is responsible for ensuring the anonymity, authenticity and confidentiality of the vehicular safety and non-safety packets.
- IEEE 1609.3: standard developed as a guide for WAVE network configuration management and WAVE Short Message (WSM) transmission and reception.
- IEEE 1609.4: standard responsible for WAVE Multi-Channel Operations that it provides DSRC frequency band coordination and management of multiple connection access by adopting IEEE 802.11p for Wireless LAN Media Access Control (MAC) and Physical Layer (PHY) specifications

DSRC specifies a channel switching scheme to allow Vehicles to alternate between different packets of safety and non-safety applications. DSRC standard recommends that for Synchronization Interval (SI) which is equal to 100ms, vehicles should visit the CCH to exchange their different types of status messages with the neighboring Vehicles. The SI

is composed of Control Channel Interval (CCHI) and Service Channel Interval (SCHI), both used to respectively access the CCH and the SCH for sending safety and non-safety applications packets. Many challenges are studied to optimize the CCHI and the SCHI to guarantee a fair share of SI between the safety and non safety applications and to prioritize applications based on their QoS requirements. The more we increase the time share of the CCH from the SI the more we increase the reliability of safety applications.

3.1 WAVE 802.11p MAC

Most of the Medium Access Control(MAC) protocols developed in the literature and used in IEEE 802.11p are employed in earlier standards of Wireless LANs. The medium access is very challenging in VANETs because of its rapidly changing network topologies due to vehicles' mobility. Frequency Division Multiple Access (FDMA) [71], Time Division Multiple Access (TDMA) [132] and Code Division Multiple Access (CDMA) [91] are very complex to implement because of the permenant need of coordination to associate frequency channels, time slots or codes with vehicles. The IEEE 802.11p MAC protocol uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol [120] that is highly considered the best adapted by VANETs. The IEEE 802.11p offers to transmit packets on one channel and it employs a continuous mechanism [72] for DSRC channel access to send both non-safety and safety packets, as presented in Fig. 3.1. WAVE recommends that, for a fixed SI equal to 100ms, every vehicle have to be tuned to the CCH in order to exchange their safety related messages with the neighboring vehicles [53]. WAVE employs Quality of Service (QoS) differentiation between the packets sent over CCH or SCH by supporting the contention-based Enhanced Distributed Channel Access (EDCA) mechanism [72]. EDCA implements four different access classes by using different transmission queue for each, as presented in Fig. 3.2, as well as various Arbitration Inter-Frame Spacing (AIFS) period, maximum and minimum Contention Windows (CWs) for traffic priority. To guarantee channel switching and coordination mechanisms,

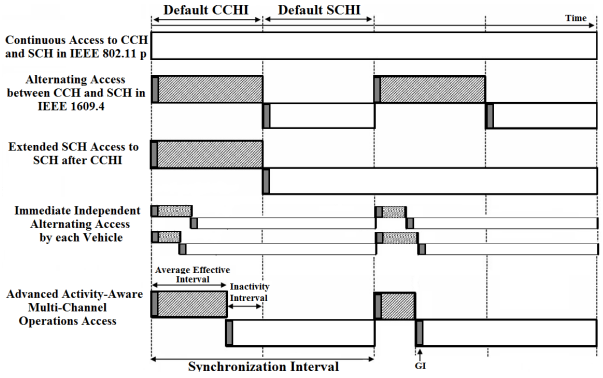

Figure 3.1. Wave Different Channel Access Mechanisms.

the IEEE 1609.4 standard for Multi-Channel Operations [100] is developed on top of IEEE 802.11p MAC as Fig. 3.2 presents. To support Multi-Channel Operations, channel switching procedures have been proposed in IEEE 1609.4. The IEEE 802.11p MAC uses Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) In this technology, there are four access classes (ACs) with different Arbitration Inter Frame Space Numbers (AIFSN) to ensure less waiting time for high priority packets, where CWmin and CW-max respectively represent the minimum and maximum contention windows that an AC can use in the IEEE 802.11 back-off process IEEE 1609.4 describes a concept of channel intervals in which time is divided into alternating Control Channel (CCH) and Service Channel (SCH) intervals. The general concept calls for each interval to be 50ms long. A pair of a CCH and SCH intervals forms a Sync interval. There are ten Sync intervals per second, the clearest issue with IEEE 1609.4 is channel utilization. While a device is allowed to stay on the CCH and send and receive vehicle safety messages at all times, it is generally expected that it should schedule its safety message transmission during the CCH intervals. This is to accommodate the safety needs of vehicles nearby that are actively channel switching to engage in other applications in SCHs. A major challenge in VANET is the frequent topology changes in network topology. VANET MAC protocols have to reduce the medium access delay for safety applications, to handle all the vehicles entering or leaving the network.

3.2 IEEE 1609.4 Standard for Multi-Channel Operations

The IEEE 1609.4 supports Multi-Channel wireless radio operations for WAVE by providing the coordination and management of the DSRC channel switching and routing by adopting the specifications of IEEE 802.11p MAC and Physical Layer(PHY). Fig. 3.1 describes the channel intervals by dividing channel access times into fixed alternating CCH and SCH intervals that are dedicated respectively for sending safety and non-safety packets. When it comes to VANETs with low density of vehicles, the static switching scheme of IEEE 1609.4 does not guarantee an optimal channel utilization. It presumes that vehicles have to be tuned to CCH (resp SCH) during CCHI (resp SCHI) no matter the existence of messages that need to be sent. Low density networks are characterized with important inactivity for safety message transmissions during the specified static timer in IEEE 1609.4. It represents a challenging incentive to enable a dynamic intelligent time share to access the different channel types based on the VANETs' safety messages and Cloud applications. This optimized access, if exists, will increase the reliability of VC to offer more throughput, handle efficiently the task placement and allows a higher probability of successful virtual machine migration from vehicles before leaving the network forming the VC.

3.3 Advanced Activity-Aware (AAA) Multi-Channel Operations

A detailed description of the variables used in the formulation of the AAA scheme is presented in Table 3.1. We describe the algorithm AAA Multi-Channel Operations Scheme that consist in two main parts of updating MAC interval timers after finding the effective CCH utilization. Moving vehicles under the same RSU communication range,

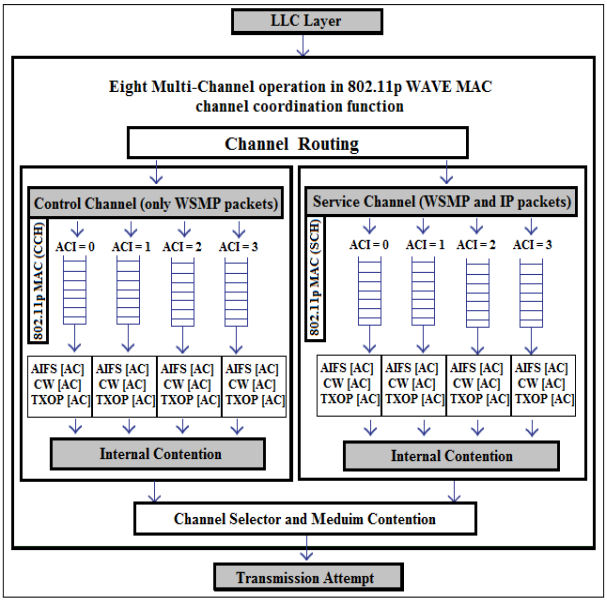

Figure 3.2. IEEE 1609.4 MAC With Multi-Channel Operations.

will have time-changing distances between each other, and the network will end up having a varying end-to-end delays of generated and transmitted BSMs during CCHI. Additionally, newly accepted vehicles to the network and VC will send non-safety applications packets but broadcast their safety beacons. Every vehicle will contribute by calculating the average end-to-end delay and the number of exchanged BSMs in order to dynamically find and evaluate the overall inactivity during the used CCHI and whether or not it can be reduced. The preliminary phase consist of collecting the characteristics about the V2V communication usage during BSM exchange, and later the RSU collects the details from all the vehicles under its management and execute the AAA algorithm. Based on the safety packet rates RG_{BSM}, the total number of generated BSMs by every vehicle in one SI can be found by using Eq.3.1 and are considered ready in the queue at the beginning of every Guard Interval (GI) during every CCHI.

$$NG_{BSM} = RG_{BSM} * GI \tag{3.1}$$

During repeated CCH access based on the CCHI, the number of the broad-casted BSM related packets by the DSRC antenna in vehicle's OBU is expressed in Eq. 3.2 by calculating the difference between the originally generated and the still queued BSMs Qv_h after channel switching.

$$Sv_h = NG_{BSM} - Qv_h \tag{3.2}$$

Successfully received BSMs have and average end-to-end delay denoted by R_{vhBSM}, can be found by every vehicle vh by using Eq. 3.3. Let t_{Rijh} and t_{Sijh} be respectively the time of i^th BSM received by vehicle vh and sent by vehicle vj.

$$Dv_h = \frac{\sum_{\substack{i=1 \\ i \neq h}}^{N_V} \sum_{j=0}^{N_{ih}} t_{Rijh} - t_{Sijh}}{R_{vhBSM}} \tag{3.3}$$

The inactivity interval during current switching scheme can be calculated as the difference between the default CCHI and the average time during which the CCH is effectively used, which is given by Eq. 3.4, where D_{V2V}, given by Eq. 3.5, represents the average delay of BSM in V2V communication and S_{V2V}, given by Eq. 3.6, is the number of exchanged BSM in the network.

$$U_{CCH} = S_{V2V} * D_{V2V} \tag{3.4}$$

$$D_{V2V} = \frac{D_{V2V} + \sum_{h=1}^{N_V} D_{vh}}{} \tag{3.5}$$

$$S_{V2V} = \frac{S_{V2V} + \sum_{v_h=1}^{N_V} Sv_h}{1 + N_V} \tag{3.6}$$

The increased average of VC related packets that are sent by vehicles to the RSU is formulated by Eq. 3.9, during the enhanced SCHI that is expressed in Eq. 3.8 based on the calculated inactivity interval given by Eq. 3.7 as the difference between the default interval and the effective utilization interval.

$$I_{CCH} = CCHI - U_{CCH} \tag{3.7}$$

$$ESCHI = DSCHI + ICCH \tag{3.8}$$

$$S_{VC} = \sum_{h=1}^{N_V} NG_{VC} - Qv_{iVC} \tag{3.9}$$

The algorithm collects after every SI the information related to the successful sent packets,

Table 3.1. Symbols and Notations

Varaible	Description
N_V	Total Number of Vehicles
ST	Total Simulation Time
SI	Default Synchronisation Interval
N_{Sync}	Number of SIs in Total Simulation Time
SI_i	Current Snchronisation Interval
I_{CCH}	Inactivity Interval of CCH in CCHI
U_{CCH}	Average effective utilisation of CCH in CCHI
RG_{BSM}	BSM packet generation rate
RG_{VC}	VC non-safety application packet generation rate
NG_{BSM}	Number of generated BSM messages per vehicle
NG_{VC}	Number of VC applications packets per vehicle
S_{vi}	Number of BSM sent by vehicle vi
D_{vi}	Average end-to-end delay of all BSMs received by vi
Q_{vi}	Number of BSM queued packets of vi
Qvi_{VC}	Number of VC queued packets of vi
Rv_{iBSM}	Number of BSM received of vehicle vi
t_{Sijh}	Time j^{th} BSM is sent by vi and received by vh
N_{ih}	Total number of BSMs sent by vi and received by vh
$tSVC_{ijRSU}$	Time when packet j is sent from vi to the RSU
$tRVC_{ijRSU}$	Time when packet j sent from vi is received by RSU
$NiRSU$	VC packets sent by vi and received by RSU
S_{V2V}	Average number of BSM messages sent by all the vehicles
S_{VC}	Average VC packets sent by vehicles to the RSU
D_{V2V}	Average end-to-end delay of BSM message
D_{VC}	Average end-to-end delay of VC packets
t_{Rijh}	Time j^{th} BSM sent by vi is received by vh
$DCCHI$	Default Control Channel Interval
$DSCHI$	Default Service Channel Interval

delays, vehicles' number and other parameters. It updates the characteristics of WAVE every 10*SIs=1s by figuring out whether the used CCHI is longer than needed or not.

Giving more time to the SCHI will increase the successful transmission rate and decrease the average end to end delay of up-link data from vehicles to the RSU. For simplification purposes, we will not consider any prioritization of any kind of application packets over others during the transmission attempt in CCH or SCH. BSMs are the only type of V2V communication in the CCH and IP packets from vehicles to the RSU. We will consider the same Access Categories Access Categories (ACs) for all the Queues either in CCH or SCHs, for that we assign the same minimum and maximum for Contention Windows (CW) and AIFS as shown in Table 3.1. We do not intend to study the effect of service prioritization and the differentiation between ACs of the different channel access parameters of EDCA used in IEEE 802.11p. We would rather identify the overall inactivity interval of using the CCH between all the vehicles during the default CCHI and use it to increase the time share of the SCH to fulfill the enhanced VC performance. Algorithm 1 presents more details for the part of finding the average end-to-end delay between exchanged BSMs between every SI transition, as well details about inactivity and increased SCHI in order to keep the default SI equal to 100 ms as predifined by the standard.

Algorithm 1 Advanced Activity-Aware Multi-Channel Operations

Result: Adjusted CCHI and SCHI

$\mathrm{N}_{Sync} = \frac{ST}{SI}, SI_i = 1$

while $SI_i < N_{Sync}$ **do**

  if *Current Time != UTC second* **then**

    $S_{V2V} = \frac{S_{V2V} + \sum_{v_h=1}^{N_V} Sv_h}{1+N_V}$

    $D_{V2V} = \frac{D_{V2V} + \sum_{h=1}^{N_V} D_{vh}}{1+\sum_{v_h=1}^{N_V} Rv_{h\,BSM}}$

  else

    {Average Effective CCH Utilization}

    $U_{CCH} = S_{V2V} * D_{V2V}$

    {Inactivity Interval in current CCHI}

    $I_{CCH} = CCHI - U_{CCH}$

    if $U_{CCH} < D_{CCHI}$ **then**

      {*Update MAC Interval Timers* }

      $CCHI = U_{CCH}$

      $SCHI = SCHI + I_{CCH}$

      { *The* NG_{VC} *are ready at the new SCHI*}

      $NG_{VC} = RG_{VC} * SI$

      {*Increased* S_{VC} *in new longer SCHI*}

      $S_{VC} = S_{VC} + \sum_{h=1}^{N_V} NG_{VC} - Qv_{iVC}$

      {*Reduced* D_{VC} *in the new longer SCHI*}

      $D_{VC} = \frac{D_{VC} + \sum_{i=1}^{N_V} \sum_{j=0}^{NiRSU} tRVC_{ijRSU} - tSVC_{ijRSU}}{1+\sum_{i=1}^{N_V} \sum_{j=0}^{NiRSU} tRVC_{ijRSU} - tSVC_{ijRSU}}$

    else

      {*The number of vehicles is high and CCHI is not sufficient*}

      $CCHI = D_{CCHI}$

      $SCHI = D_{SCHI}$

    end

    $S_{V2V} = 0$

    $D_{V2V} = 0$

    SI_i= SI_i + *1*

  end

end

Chapter 4: Vehicular Task Workload Modeling and Vehicular Cloud Computing Model

4.1 Vehicular Task Workload Modeling

We consider a set of n Bag-of-Tasks (BOTs), as presented in Fig. 4.1, each forming a collection of identical and independent tasks that can be placed and executed in any order. Every BOT has its own requirements in terms of delay, throughput and number of VMs that have to be met in accordance with the VC performance metrics like delay and throughput. In the context VCC, we will not consider the periodicity and the temporal burstiness of task arrivals and we assume that all the BOTs are available before any scheduling decision.

4.2 Vehicular Cloud Computing Model

Based on our developed AAA scheme and the default IEEE 1609.4, we consider the simulation results shown in Fig. 4.7 of the throughput of different VCs composition of microdatacenters. For simplicity, we consider that each vehicle contribute to the formation of the VC with one virtualised VM from its OBU's microdatacenter. The overall VCC system is built on top of various VCs that differ from one vehicular network to another. Each coalition of vehicles under the same RSU constitute a single VC, as shown in Fig. 4.3, with specific OBUs, virtualized as VM, density and VM throughput and delay. The AAA scheme calculates the average end-to-end delay and the number of successfully exchanged BSM, it is the key part to dynamically evaluate whether or not the used CCHI can be reduced for the sake of an increased SCHI as detailed in Fig. 3.1. Significant improvement in the each VC metrics are achieved by using the AAA in comparison to the

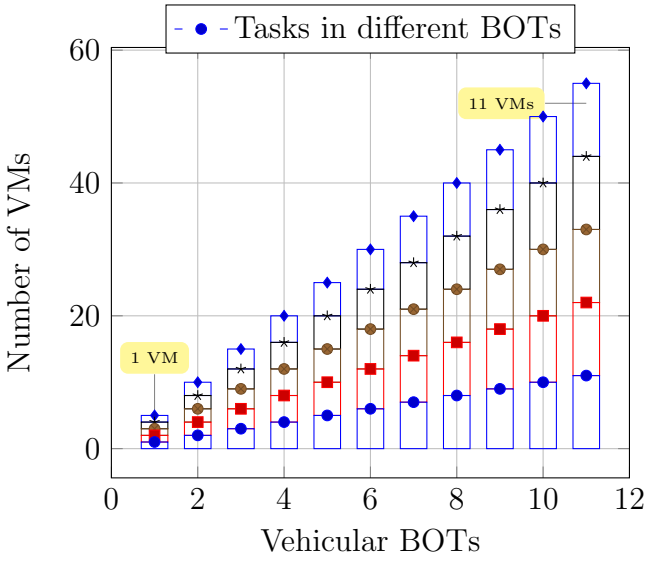

Figure 4.1. Vehicular Cloud-Based BOTs Requirements in VMs.

default IEEE 1609.4. The scheduling problem consists of placing a set of BOTs into the available resources of various VCs while taking into account the uniqueness of each one of them in one hand and maximizing the utilization of VMs on the other hand. Moreover, the TCC resources offer VMs that are pay-per-use. Therefore, a cost effective scheduling policy cannot greedily acquire as many resources as possible.

4.3 Fast Greedy Heuristics for scheduling vehicular tasks in Vehicular Cloud Computing

A fast greedy heuristics algorithm framework developed in [96] is used to find the sub-optimal solutions for the virtual machine allocation problem. The greedy scheduling algorithms never consider its choices after a scheduling decision has been made, which may lead to a non-optimal final decisions since one policy may have the highest reward among all the possible decisions, but can deteriorate the reward of the next decisions. We consider the total VCC formed by all the VCs described and simulated above and are considered as data centers with different throughput and delay. There is no general optimal decision-making scheme for task scheduling into VCs. Therefore, optimality of

policies vary depending on the abundance and the distribution of the VMs in all the VCs as well as the requirements of the vehicular applications in every BOT. For this, the greedy-assignment of task to VMs of specific VC is considered as central scheduler [35] that sorts the VMs of all the VCs based on their performance metrics and assign task to the most efficient first VM suitable for hosting. It then continues allocation until no task remains in all the BOTs or else, VMs queues of every VC are full. The fast greedy heuristics scheme is described by the following algorithm.

Algorithm 1 : Greedy scheme for most Efficient VM first

VCC ← Set of vehicular clouds forming the VC
S_{BOT}← *Set of different BOT*
T_i← *Set of tasks in the* i^{th} *BOT*
most efficient VM (mevm) ← *NULL*
for all $bot_i \in S_{BOT}$ ***do***
  for all $T_{ij} \in bot_i$ ***do***
    isScheduled=Flase
    for all $vc_i \in VCC$ ***do***
      if *isScheduled=True* ***then***
        Break
      end if
      for all $VM_i \in vc_i$ ***do***
        if VM_i *is available and meets the requirements of the task* T_{ij} ***then***
          $mevm = VM_i$, *isScheduled=True,* $T_{ij} \notin bot_i$ ***end if***

if *isScheduled=Flase* ***then***
  Task T_{ij} *is placed in paid cloud*
  T_{ij} *is removed from* bot_i
end if

4.4 Sequential and CUDA-accelerated MDP-Based scheme for scheduling tasks in Vehicular Cloud Computing

4.4.1 MDP-Based Scheme Model

Abstractly, the Markov Decision Process(MDP) can be formulated by a state space S of the VCC, action set A of scheduling decisions, immediate reward function r(s,a,s') of the VCC and probabilistic transition P (s'|s,a) from a state s to s' under the action a.

4.4.2 System States

As stated earlier, we will focus on the scheduling multiple set of BOTs with different VMs requirements into the VCs of the VCC system. For an efficient task scheduling, there is a trade-off to achieve between minimizing the cost related to paid hosting services and maximizing the overall reward of the VCC. The MDP system states S reflects the distributed VCs with the available virtualized resources of the vehicles' OBUs, the hosted applications and the vehicular tasks remaining to be placed. So we can denote the state set as:

$$S = \{s|s = (n1, n2, ..., n_{TT}, M, e)\} \tag{4.1}$$

where n_i is the number of vehicular applications that were allocated to the i^{th} VC, M represents the set of available VMs per VCs and e is the task placement request.

4.4.3 Actions

In our MDP model for task scheduling we have many possibilities of scheduling actions a that can be executed from the set A:

$$A = \{-1, 1\} \tag{4.2}$$

When an application scheduling request occurs, the VCC decides among the set of actions which one should to be launched. Placing in TCC due to communication delay or throughput limitation or VM insufficiency represent -1. Whereas a(s)=1 represents the case that the selected VC from the VCC the minimum delay and VM required by the application.

4.4.4 Transition function

Our problem has a probabilistic transition function P (s'|s,a) $\longrightarrow$ [0,1] that indicates the probability of transitioning from a specific state s to another state s' of VCC after an action a is taken. For every application placement request, each of the VCs verifies whether or not its average Vehicle to Infrastructure (V2I) packet delay meets the requirement of the application itself and it has available non occupied VMs that meet the required throughput for hosting it. When a new service request needs to be placed, the number of occupied VMs is given by Eq. 4.3 independently of the taken action by the overall VCC system.

$$N_v = \sum_{i=1}^{N_c} \sum_{j=1}^{N_a} u_{ij} + v_j \tag{4.3}$$

When a scheduling request of an application j, requiring n_j VMs, is served and placed on $i^{th}VC$, the number of occupied VMs in the corresponding cloud becomes

$$T_{VMi} = \sum_{i=1}^{R_c} u_i + n_j \tag{4.4}$$

4.4.5 Rewards

Based on the taken action a, the instant system reward under the current state of the VCC s can be expressed as:

$$r(s, a, s') = k(s, a, s') - g(s, a, s') \tag{4.5}$$

k(s,a,s') represents the immediate reward of the VCC system after making the scheduling action a under the current state s when the event e of task placement happens. It holds the income gained by VC after hosting the task. *g(s,a,s')* is the instant cost of the current decision epoch of the task placement request. When a task scheduling request is admitted to the VC, the immediate reward that can be earned is expressed by Eq. 4.6 and the cost related to this policy is defined by Eq. 4.7.

$$k(s, a, s') = \beta_{vc} * n_j \tag{4.6}$$

$$g(s, a, s') = \beta_{tc} * n_j + \gamma_{vc} * \mu_i \tag{4.7}$$

μ_i is the underutilized VMs in the i^{th} VC. β_{vc}, β_{tc} and γ_{vc} represent respectively the induced reward per VM in VCC, cost of placement per VM in TCC and cost of underutilized VM in VCC left behind after scheduling. The VCC system does not gain any reward under the states in which all the VMs of every VC have been exhausted. Thus, the next scheduling epochs of the remaining application will only induce a cost that represents a negative contribution to the reward.

4.4.6 Total Reward of the Solution

The main objective of the MDP-based task scheduling in VCC is to optimize the task placement of various BOTs over distributed VCs that increases the function of the total cloud reward. The function is the maximum expected long-term cumulative reward of the VCC is expressed as:

$$\begin{aligned} R &= \max_{s \in S}[\sum_{s' \in S, a \in A} r(s, a, s')] \\ &= \max_{s \in S}[\sum_{s' \in S, a \in A} k(s, a, s') - g(s, a, s')] \\ &= \max_{s \in S}[\sum_{s' \in S, a \in A} (1 + \beta_{vc}) * n_j - (\beta_{tc} * n_j + \gamma_{vc} * VC_{ij})] \end{aligned} \tag{4.8}$$

4.4.7 Optimal Solution with MDP Scheduling

An optimal solution of scheduling decisions is found through the iteration algorithm so that it maximizes the expected long-term overall reward of the VCC system. The solution to the MDP is a policy π: $S \longrightarrow A$ that maximizes the long-term sum of expected rewards achievable starting from any state s. The iterative algorithm tabulates the cumulative optimal reward in the function V^*, which represents the unique solution for the Bellman equation given in Eq. 4.9:

$$V^*(s) = \max_{s \in S}[r(s,a,s') + \sum_{s' \in S} P(s'|s,a)V^*(s')] \tag{4.9}$$

Since our model is formulated as finite MDP model with sequential decision making under uncertainty, with finite spaces of states and actions, the value iteration algorithm is used to solve the maximization problem that is presented by Eq. 4.8. A detailed description of the steps executed by the iteration algorithm are given by the following algorithm.

Algorithm 2 : Value Iteration Algorithm

Step 1: We set V(s)=0 for every state s of the VCC, $k = 0$ and $\epsilon > 0$.

Step2: For each state s, compute $\mathrm{V}^{k+1}(s) = \max_{s \in S}[r(s,a,s') + (\sum_{s' \in S} P(s'|s,a)V^k(s'))]$

Step 3: $If ||V^{k+1}(s) - V^k(s)|| < \epsilon$, go to ***Step 4***. Otherwise, increase k by 1 and go back to ***Step2***.

Step 4: For each s ε S, we compute the stationary optimal policy of scheduling in the VCC and stop and output the deterministic policy of scheduling, $\pi = \mathrm{argmax}_{a \epsilon A_s}[\max_{s \in S}[r(s,a,s') + \sum_{s' \in S} P(s'|s,a)V^{k+1}(s')]$

4.5 CUDA Accelerated Blocks of State Divided Iteration (BSDI) Algorithm for Value Iteration

General purpose GPUs have become specialized for highly parallel intensive Single Instruction Multiple Data (SIMD) computation, as shown in Fig. 4.2. These modern GPUs are therefore being designed to guarantee that more transistors are devoted to effective data processing rather than to flow control and data caching unlike CPU. Every Streaming Multiprocessor (SM) contains multiple streaming processors (SPs), also called CUDA cores [107], and highly energy efficient special function units, which are simple (without scalar and only vector instructions, simple register renaming, simple branch prediction and prefetching, etc.). CUDA-enabled GPUs accelerate data-parallel applications by running thread groups in lock-step to execute the same instruction in a SIMD mode [38] and mapping them to parallel execution units. When a CUDA program on the host CPU invokes a kernel grid, the set of thread blocks are grouped and distributed to SMs with the available execution capacity. Multiple thread blocks are concurrently executed on a single SM and when it finishes new blocks of threads are launched on vacated SMs. In contrast, while using fully hyper-threaded CPUs the core's clock speed decreases [105] so that the chip does not melt. We sort of have a tradeoff between multithreading and the clock speed. In the GPUs we do not have these design tensions and we only consider a highly data parallel programs to be executed by thousands of threads and we are not interested in how long it takes any thread to complete but, rather, how long to complete the kernel in order to evaluate the the computational throughput performance in single/-double precision Floating-point Operations Per Second (flops). GPUs architecture with multiple cores is driven by the high memory bandwidth in order to

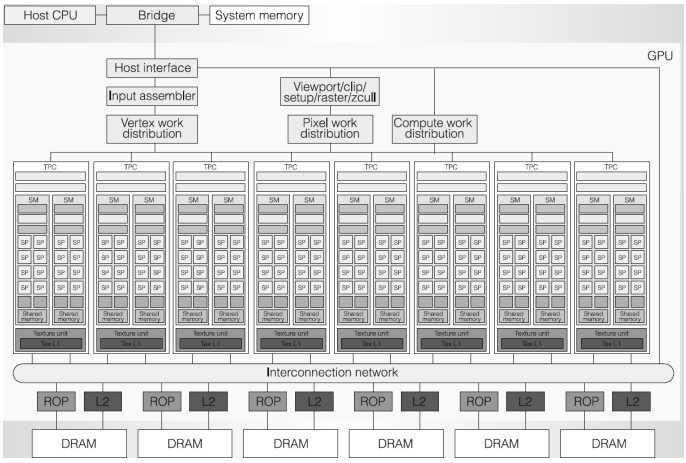

Figure 4.2. Block Diagram of CUDA Enabled GPU.

offer massive parallel processing resources, and are very-suited to address problems that can be expressed as data-parallel programming. The NVIDIA's Quadro K1100M GPU, as detailed in Table 4.1 is used to leverage the parallel capabilities of the CUDA programming model by running the kernel program that is the parallel implementation of the value iteration algorithm responsible for finding the optimal solution of the MDP scheduling model, as detailed in [74].

Table 4.1. CUDA Enabled GPU Characteristics

Varia, ble	Description
Device 0/CUDA Driver/Routine	Quadro K1100M with 7.5/6.5
Architecture	Kepler
CUDA Compute Capability	3.0
Total amount of gloal memory	2048 MBytes
Number of MPs	2
Number of CUDA cores/MP	192
Total number of cores	384 CUDA cores
Clock Rate of GPU	706 MHz (0.71 GHz)
Clock Rate of Memory	1400 MHz
Memory Bus Width	128 bits
Size of L2 Cache	262144 bytes
Total Amount of Constant Memory	65536 bytes
Total Shared Memory per Block	49152 bytes
Total Registers per Block	65536
Size of Warp	32
Maximum Threads Number per MP	2048
Maximum Threads Number per Block	1024
Texture alignment	512 bytes
Host to Device Bandwidth	9735 MB/s
Device to Host Bandwidth	9302 MB/s
Device to Device Bandwidth	3200 MB/s
Peak single precision performance	550 Gigaflops

The Block of State Divided Iteration (BSDI) is presented in Algorithm 3 to parallelize

the state space of the VCC and to adapt the value iteration algorithm to the CUDA programming model. It partitions the state space S into equal sized blocks and we assign a thread within each block to be responsible for the calculation of the rewards under different transitions of possible scheduling decisions while residing the successor states of the states' actions within the same block.

Algorithm 3 : Parallel Blocks of State Dvivided Iteration

cudaMemcpy($V_g(s), h_{Vg}(s)$, lengthbytes, cudaMemcpyHostToDevice)

$V_g(s) \leftarrow$ State space residing in global memory

__shared__ $V_s(s) \leftarrow$ State space residing in shared memory

for all s $\in$ S in parallel **do**

$V_s(s) = V_g(s)$

Synchronize threads : __syncthreads()

$v \leftarrow V_s(s)$

for iterations $<$ number of episodes **do**

$\Delta_s \leftarrow 0$

for all a $\in$ A **do**

$V_s(s) \leftarrow \max_{s \in S}[r(s,a,s') + \sum_{s' \in S} P(s'|s,a)V_{InBlock}(s')]$

end for

$\Delta_s \leftarrow \max(\Delta_s, |v - V_s(s)|)$

$V_s(s) \leftarrow v$

Synchronize threads : __syncthreads()

end for

$V_g(s) = V_s(s)$

end for

for all Δ_s of s $\in$ S in parallel **do** $\pi \leftarrow \max(\Delta_s)$

end for

cudaMemcpy(h_π,π,lengthbytes,cudaMemcpyDeviceToHost)

4.6 Simulation of AAA and static IEEE 1609.4 scheme and performance metrics comparison of the VCC and the safety applications

Fig. 4.3 represents a screenshot of the NS-3 simulated VC. It is formed via an abstracted layer of computing from the microdatacenters of the fixed RSU as well as of the random vehicles that are moving in way point mobility with random speed under the RSU area. Packets of the safety messages are generated by every vehicle once every 100 ms, which corresponds at the beginning of Guard Intervals of every CCHI. Additionally, the packet generation rate of the non-safety applications hosted in the VC that are also queued in each vehicle during the beginning of the Guard Interval corresponding to every SCHI. Only periodic V2V BSMs with equal packet size of 200 Bytes are considered in the study and the VC non-safety packets are with equal size of 1500 Bytes. The detailed parameters used in the simulation with NS3 are presented in Table 4.2. In addition, we consider for simplification reason the usage of equal priorities between packets by using the background traffic (BK) defined in the access category in EDCA for the CCH. In zone Z0 in Fig. 4.4, it is characteriszed with few number of distant vehicles that are introduced randomly to the network under the RSU communication range. The AAA scheme offers and average end-to-end delay of V2V safety packets that very much comparable to the the one given by using default static IEEE 1609.4, even though the AAA used a reduced CCHI as presented in Fig. 4.8. When a larger number of vehicle is introduced under the RSU communication area, as highlighted in zone Z1, the inter-distances between vehicles gets smaller and the CCHI gets increased in order to give more activity time in CCH for vehicles so that they successfully exchange their safety related messages, as presented in Fig. 4.8.

Table 4.2. Parameters of the Simulation in NS3

Parameters	Values
Number of Vehicles	5 - 50
Simulation Time	1000 ms
Number of RSUs	1
Virtual Machine size	500 kbits
Synchronisation Interval (SI)	100 ms
Transceiver Communication Range	200m
Default SCHI and CCHI	50 ms
RSU area	50*50m
Default Guard Interval (GI)	4 ms
VC application packet generation rate	10 Hz
VC application packet size	1500 Byte
BSM packet generation rate	10 Hz
BSM packet size	200 Byte
Modulation and Data rate	QPSK 6 Mbps
CCH and SCH Bandwidth	10 Mhz
CWmin of BK	15
EDCA Access Class	Background Traffic (BK)
CWmax of BK	1023
Number of SCH and CCH	1
AIFSN of BK	9
Propagation Model	Nakagami

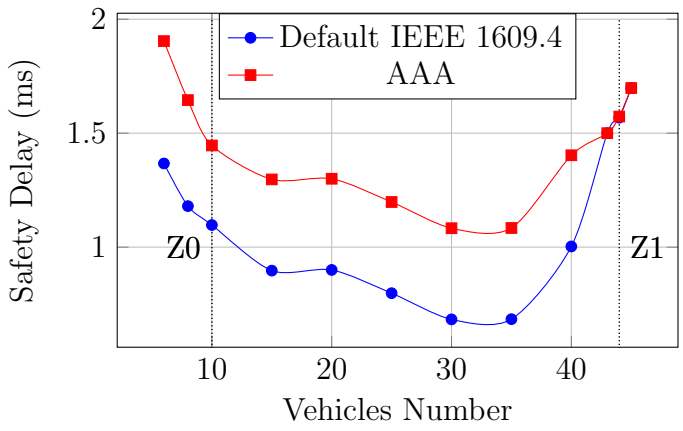

Figure 4.4. Average End-To-End Delay of BSM Versus Number of Vehicles.

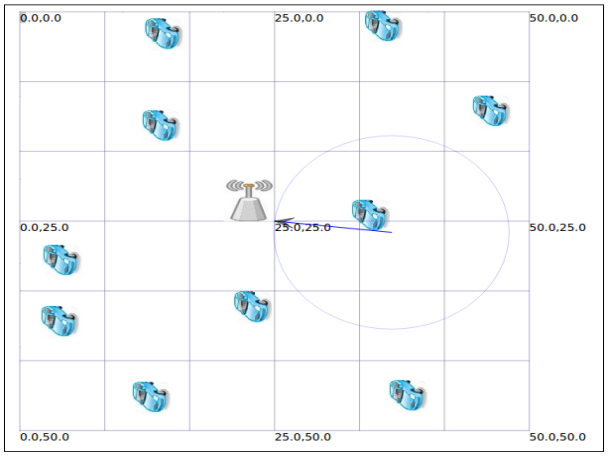

Figure 4.3. NS3 Simulation of the VC Build Around the RSU.

The average end-to-end delay of packets delivered in the VC is acceptable in low dense VANETs, as shown in Fig. 4.5. As long as the number of vehicles increases, more BSMs are queued to be sent and so the CCHI has to get increased, which induces a reduction in the Inactivity Interval progressively and consequently will lead to a decrease in the SCHI for cloud related packets, as presented in Fig. 4.8, and increases of the average end-to-end delay of VC applications. Our Advanced Activity-Aware Multi-Channel Operations

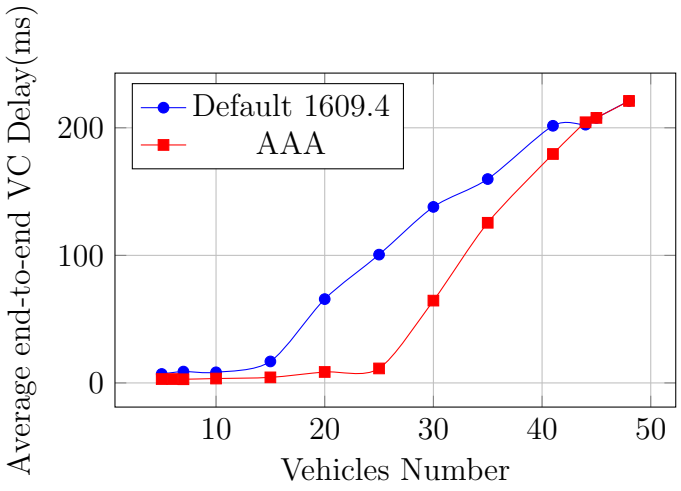

Figure 4.5. Average End-To-End Delay to the RSU of Non-Safety Messages Versus Number of Vehicles.

scheme offers a very much comparable successful number of exchanged BSMs and V2V

throughput as in the default IEEE 1609.4 with static channel access switching, as compared in Fig. 4.6, even though that the CCHI was significantly decreased as presented in Fig. 4.8.

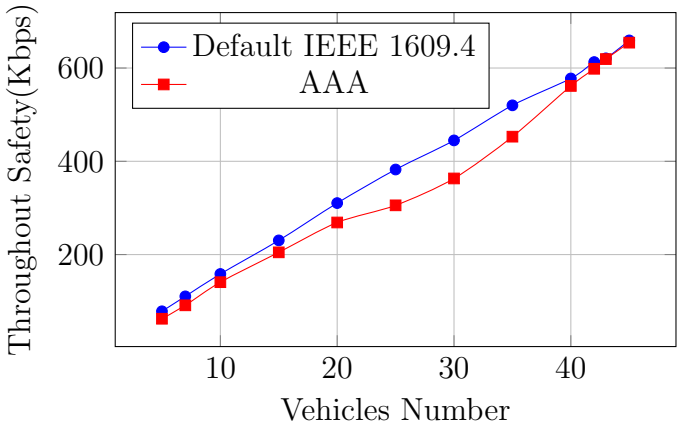

Figure 4.6. Throughput of V2V BSM Safety Messages Versus The Number of Vehicles.

Significant improvement is achieved in the throughput of the VC by using the AAA Multi-Channel Operations scheme over the static IEEE 1609.4 as detailed in Fig. 4.11. The conisderable improvement is caused by automatically considering that the CCHI is minimized by the inactivity interval that is very significant in VANETs with low vehicle density, which induce an extended SCHI compared to the default for the VC applications. Therefore, more vehicles will find slots in CW to successfully send cloud related packets before switching to V2V safety communication, which consequently generates a higher throughput of the VC.

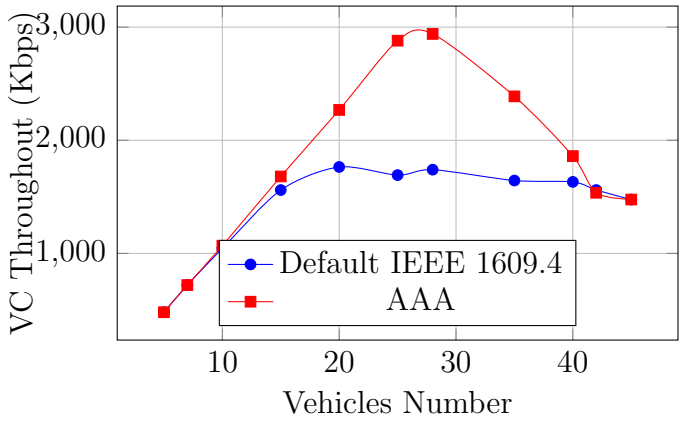

Figure 4.7. VC Throughput Versus Number of Vehicles.

Fig. 4.8 shows the impact of the AAA scheme to finding the inactivity interval and automatically adapting the interval timers of IEEE 1609.4 by decreasing the CCHI for BSM dissemination and increasing the SCHI for VC usage for both vehicular type of communiaction in WAVE.

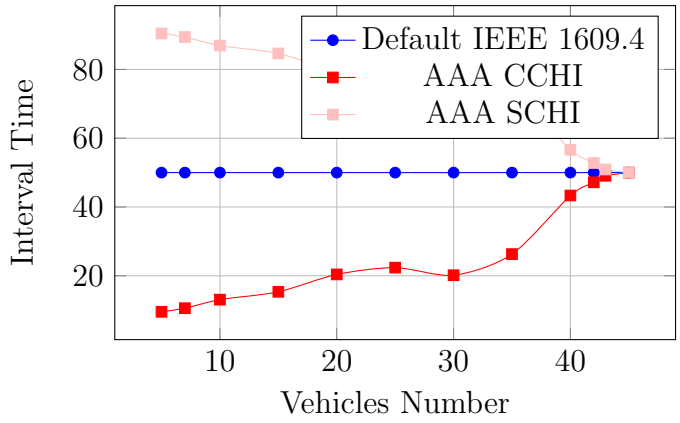

Figure 4.8. Channels Intervals Adjustment of AAA and IEEE 1609.4 Versus the Number of Vehicles.

The formulation of the average throughput per vehicle of non-safety applications that are hosted in the VC is given by Eq. 4.10 as follows:

$$Th_{ij} = \frac{Th_{VCij}}{j} \tag{4.10}$$

where Th_{VCij} represented the calculated throughput of the VC found while using the i^{th}

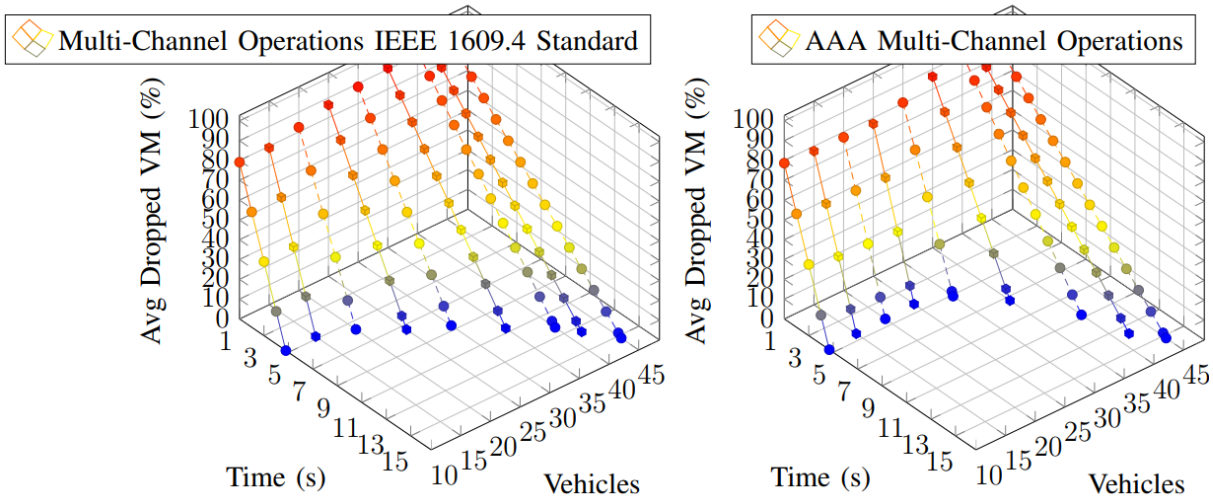

Figure 4.9. Average Percentage of Dropped VMs in Different Network Density Versus Time.

mode of either the default IEEE 1609.4 or the AAA scheme for a number of j OBUs forming it.

The scenario of the migration consist of a vehicle that has a total of $T_{VMsize} = 500$kbits as the size of all its VMs running in its OBU and should do migration to the RSU before leaving the network and the VC. The average percentage of dropped VMs, are presented in Fig. 4.9, at an instant t(s) from the beginning of migration is defined as shown in Eq. 4.11.

$$AVM_t = 1 - \sum_{k=0}^{t} \frac{T_{VMsize} - k * Th_{VCij}}{Th_{VCij}} \tag{4.11}$$

4.7 Numerical Results and Analysis of the scheduling schemes in VCC

The evaluation of the scheduling schemes of tasks for the VCC mainly targets the derived performance of the scheduling decisions based on MDP and greedy heuristic by jointly considering the resource utilization of the virtual machines offered by the VCC and

the acquired total reward of the scheduling decisions. Fig. 4.10 presents the histogram of the overall total percentage of VMs utilization after placing the tasks from the BOTs by using the fast greedy and the MDP-based schemes. We note that the fact of having equal VM utilization in VCs from the VCC under the greedy scheme and MDP scheduling does not really reflect the same placement decisions have been made of tasks. Most of the VCs forming the VCC system contain many left behind unused VMs under the greedy algorithm since it does not calculate the impact of every decision on the next placement decisions. The VM utilization achieved by using the MDP scheduling scheme is maximized through the selection of the highest total reward of a sequence of placement decisions and not only based on the reward per task scheduling.

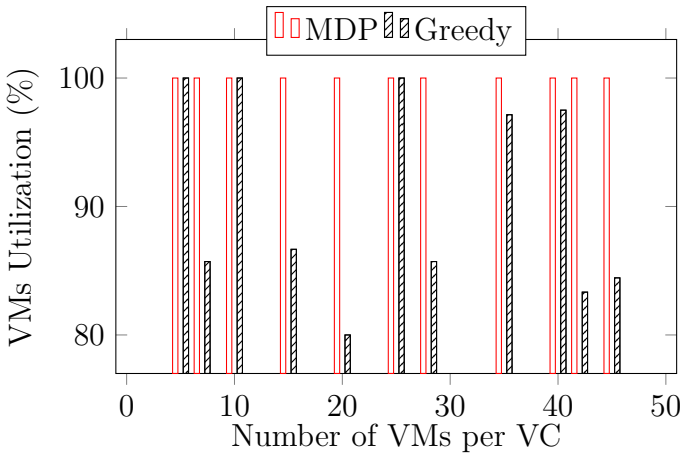

Figure 4.10. VCC VMs Utilization for Greedy and MDP Schemes.

Based on the VCC utilization and the scheduling decisions made by the greedy algorithm, Fig. 4.11 shows that various VC did not reach their optimal throughput under both of the IEEE 1609.4 and the AAA schemes.

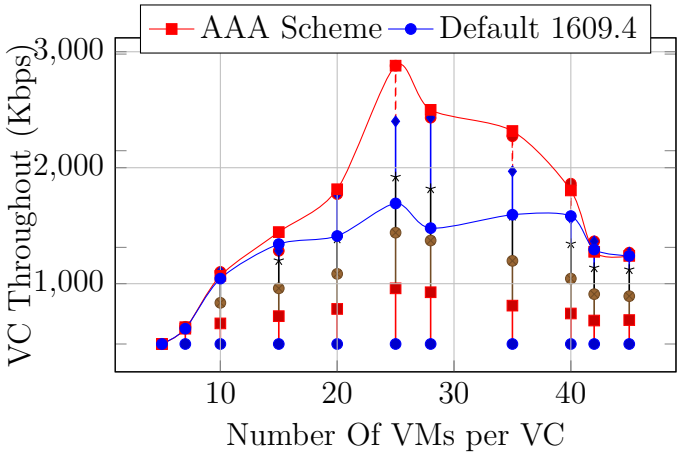

Figure 4.11. Vehicular Cloud Throughput Under Greedy Scheduling.

In addition, the evaluation of the VCs throughput by using the MDP scheme presents, as described in Fig. 4.12, a maximized effective utilization of the VMs. Moreover, the task hosted in the VMs do not follow a specific rule as in the Greedy scheme and highly depends on decisions with the highest total reward of the cloud and fill up the maximum available VMs in every VCs.

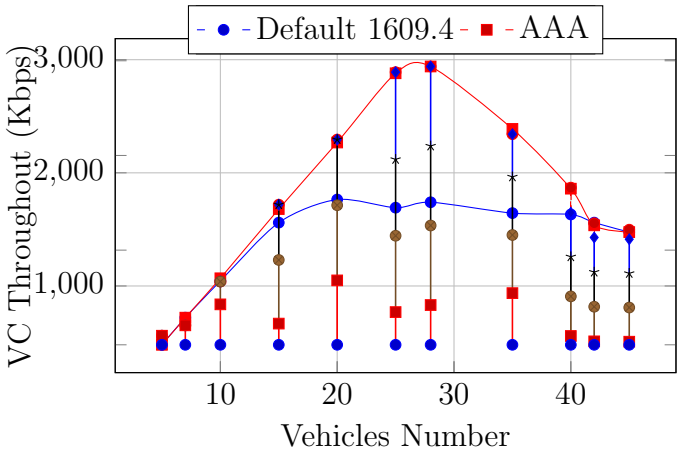

Figure 4.12. VC Throughput Under MDP Scheduling.

Fig. 4.13 and Fig. 4.14 show how the tasks of the BOTs are distributed along free VMs in VCs and paid VMs in TCC. In addition, we compare the total underutilized VMs left by the Greedy and MDP-based scheme in the VCC and total required VMs for both

of the scheme. The results also describe the improvements that MDP scheme offers in comparison with the Greedy scheme in terms of number of occupied VMs in both of the cloud environments.

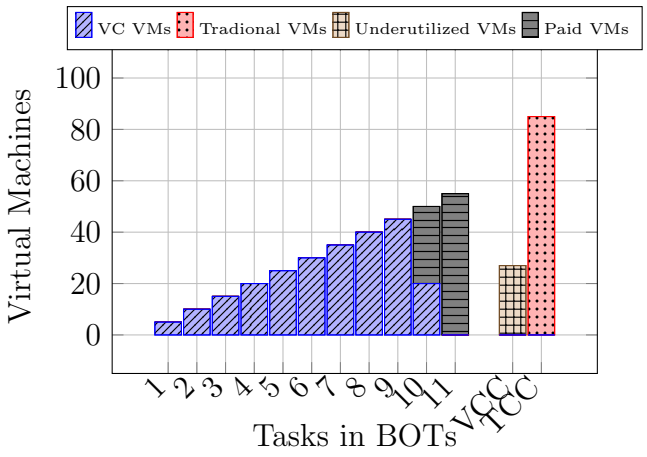

Figure 4.13. BOTs Scheduling Under Greedy Scheme.

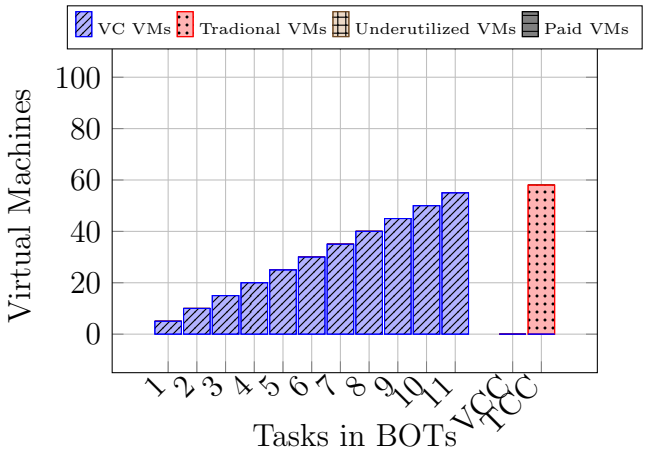

Figure 4.14. BOTs Scheduling Under MDP-Based Scheme.

The overall maximized reward resulting as the best scheduling decisions of all the BOTs under both MDP scheme and heuristic greedy algorithm is presented in Fig. 4.15. The Bar Chart in the center is the total reward of the overall VCC while taking into account the cost related to under utilization of VMs. While the dotted chart shows per

reward per VC without considering additional costs of unused VMs. We showed that the proposed MDP scheduling decisions outperform the greedy ones for every VC and for the total reward of the VCC system in both of the utilization and the reward that involve both the revenue and the cost of decisions with β_{vc}=1, β_{tc}=1.2 and γ_{vc}=1.

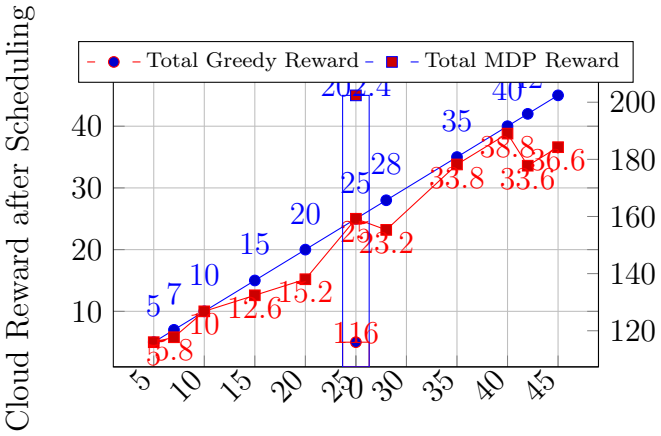

Figure 4.15. Overall VCC and per VC Reward.

The CUDA Occupancy calculator is used to find out the optimal block size for the execution of the algorithm on the Quadro K1100M GPU. The BSDI algorithm was assigned a 124 threads per block, which resulted in 124/192=64.5% occupancy per MP. As shown in Fig. 4.16, the parallel version iteration algorithm BSID performs considerably faster than the sequential algorithm in terms of the required time it takes converge for the optimal scheduling epochs. Performance is a key factor when solving MDPs as the size of the states increase quickly.

The parallel BSID algorithm achieves a speedup of almost 32% compared to the sequential value iteration algorithm to converge to the optimal decisions. The speedup can get even higher if we apply it to a higher number of VCC states the occupancy of threads running in parallel per every SM will have a higher usage.

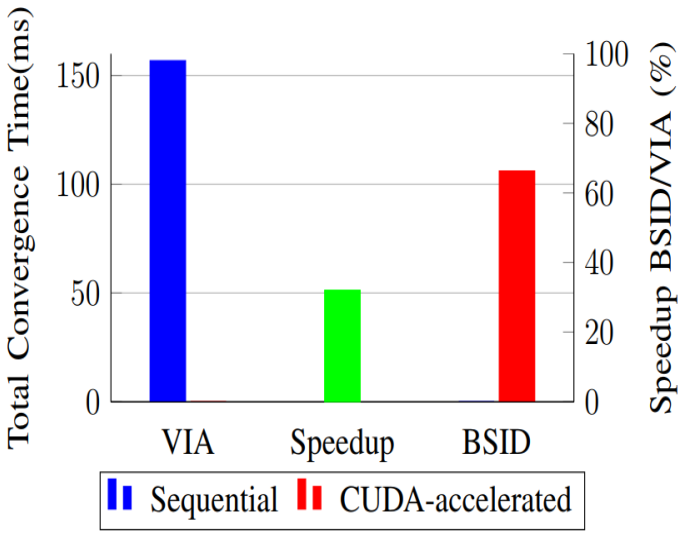

Figure 4.16. Sequential and CUDA-Accelerated Convergence Time and Speedup.

Chapter 5: Autonomous Vehicle Sensor Modalities

In this Chapter, we develop a multimodal surrounding recognition scheme for autonomous vehicles, capable of mapping corresponding recognized objects from camera, LIDAR, and V2V data sets. The proposed scheme exploits the fact that the spacial relationships between objects in all three data sets can be characterized by manifolds, both representing intersecting sets of these objects and exhibiting high neighborhood correlations. We thus employ a manifold alignment approach to learn the correspondences between similar objects within the different data sets and enrich them with the other objects not detected by each mode, thus accomplishing a more robust perception of its surroundings. We first tested the proposed approach for two specific scenes to gain preliminary insights and more rigorous analysis of the main factor affecting the alignment accuracy, such as the percentage of neighborhood correlation, overlapping objects, etc. We then applied the learned information in testing our proposed scheme in entire driving sequences. For many cases, 100% alignment accuracy was achieved, and alignment accuracy averages of 74%-92% and 50%-78% were obtained for cars and persons, respectively. Cases of mis-matches were further shown to occur only within the same type of objects and for objects that overlap or are in very close proximity to one-another, thus minimally effecting the possible decisions of the recognizing vehicle.

5.1 Preparing Data Sets for Alignment

In this section, we will illustrate the steps taken in order to prepare the manifolds of the different data sets from the multi-modes of vehicle sensing of its surroundings. Luckily, the raw V2V data usually comes with the required information for alignment, since each BSM clearly indicate the GPS position of its transmitting vehicle. Thus, the

adjacencies between vehicle objects from the V2V data can be easily computed and does not require any further recognition nor preparation. Thus, the next two subsections will focus on adapting the object recognition schemes from camera feeds and LIDAR scans to both extract the objects of interest to the alignment process, and determine their relative locations to one-another. This information will be then used to create manifolds for each of these data sets, representing the adjacency relations of their detected objects of interest, which will then enable us to feed them to the alignment process.

5.1.1 Adapting Recognition from Camera Feeds

In this section, we describe our approach of adapting the learning procedure from camera feeds, in order to prepare the camera data set that is suitable for our proposed alignment process. This adaptation consists of tailoring the darknet's CNN, developed in [101] to test the KITTI stereo images using the architecture illustrated in Fig. 5.1. To render both the two important object classes for the alignment process, namely "Cars" and "Persons", and the center of gravity of the object bounding box (or object pixels) in the 2D space. Inspired by this CNN design, we propose to exploit the feature of objects' anchor boxes, which predict the coordinates of the bounding boxes around recognized objects, to find their pixel adjacency directly from the fully connected layers that are developed on top of the convolutional network extractor (both illustrated in Fig. 5.1). We use a single CNN that processes the entire image pixels during both the training and test time, and predicts all the bounding boxes and their corresponding classes probabilities. We then apply a hard threshold on the probabilities of the object suspected to be either a "Car" or a "Person", beyond which these objects are deem to be so.

While parsing through the frames of any driving sequence, our tailored CNN stores the characteristics of each object in an organized way, including its class name, anchor box dimensions in pixels, and its calculated center of gravity in the 2D space. Moreover, counters are set to keep track of the total number of objects from each class per frame.

The extracted details of objects are stored separately from one frame to another both within the same driving sequence and across different sequences. To test the efficiency of our tailored CNN, we applied it to recognize the objects in Fig. 5.2 and Fig. 5.4, representing two original frames (that we will denote in the rest of the Dissertation as Frame (a) and Frame (b), respectively) from two different driving sequences from the KITTI Suite. Frame (a) and Frame (b) include different object counts per class either from image recognition or from labeled LIDAR object recognition. The model is being changed to support reading all the frames contained in both of the drive sequences 2011_09_28_*drive*_00016 and 2011_09_26_*drive*_0005 from the KITTI Suite. We opt to select from sequences recorded with RGB cameras instead of gray-scale, for clarity of work purposes. The results of applying our tailored CNN on Frame (a) and Frame (b) are illustrated in Fig. 5.3 and Fig. 5.5, respectively. We can clearly see from both figures that the objects are perfectly recognized and labeled with their proper classes. The center of gravity of each object is depicted by a red cross within each object's anchor box. The centers of gravity are computed according to the pixels belonging to each object and are expressed in pixels in the 2D space.

It is worth noting that some vehicles are not detected in both Fig. 5.3 and Fig. 5.5. The failure in detecting these vehicles occurred due to either the overlap between vehicles (as in the parked undetected vehicle on the right of street in Fig. 5.3) or the shaded zone they lie in (as in the case of the parked vehicle to the right of the leftmost recognized vehicle in Fig. 5.5) and thus the low variance, as perceived by the camera, between their pixel colours and those of the background. As aforementioned, the latter phenomenon was the cause of the fatal accident by the Tesla autonomous vehicle prototype. This calls for our proposed multi-modal sensing of the vehicle's 3D surrounding environment by adding LIDAR, and most importantly V2V (and also V2I and vehicle-to-pedestrian V2P) information to each vehicle's object recognition system. Indeed, the problem of these undetected objects will be resolved if the camera information is aligned with the re-

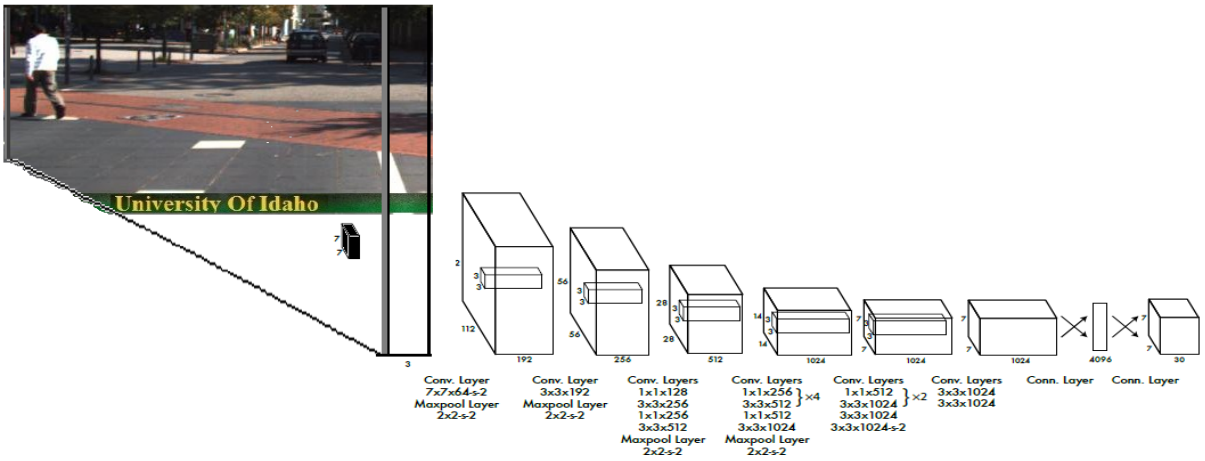

Figure 5.1. Darknet Convolutional Neural Network Architecture [101]

ceived V2V information from the other vehicles, thus reducing chances of having missing knowledge of the surrounding vehicles. In Fig. 5.6, we plot the 2D pixel-wise manifolds of the car and people objects detected by camera for both Frame (a) and Frame (b). These manifolds illustrate the adjacency relations between the centers of gravity of the detected objects in each of the frames, which represents approximate distances in terms of pixels between them in the 2D space. The neighborhood relations between the objects from the camera frames will be thus expressed in terms of pixels, and will be inserted with that format to the *neighborhood weight calculations* that will be introduced in Section 5.2.3. The obtained curves for both Frame (a) and Frame (b) in Fig. 5.6 represent camera data set manifold in the pixel domain.

It is important to note here that the representation of distant objects in the manner illustrated in Fig. 5.6 can give a meaningful value of distance between them, and can be easily correlated with distances from both LIDAR scans and V2V BSMs positions. However, due to the 2D nature of camera images, the distance representation between very close or overlapping objects in the image may be highly inaccurate and will not provide relevant information to the alignment process.

Figure 5.2. Original Frame (a) from RGB sequence drive 2011_09_26 drive_0005 #117.

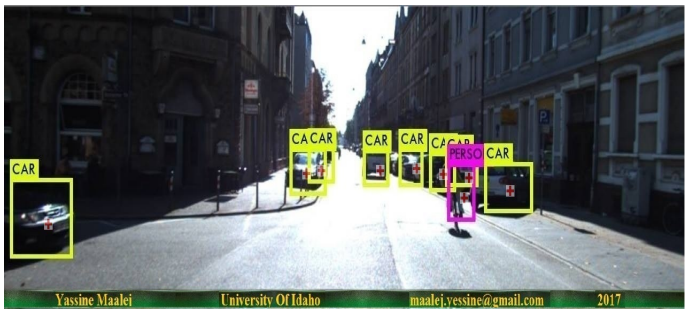

Figure 5.3. Detection of Frame (a) objects and their centers of gravity.

5.1.2 Adapting Recognition from LIDAR Point Clouds

This section illustrates the adaptation of object recognition from LIDAR point clouds to prepare a data set that is suitable for our proposed alignment process. To simplify this process, we are not considering the recognition of every object from the LIDAR point clouds, since a tremendous number of unknown, and most importantly un-mappable, objects can be detected as a set of neighbored point clouds. As in the previous section, we will thus restrict the recognition of the Vote3Deep 3D CNN, developed in [41], to identify only "Cars" and "Persons", since the remaining items do not represent major importance in the alignment process. The Vote3Deep 3D CNN uses feature-centric voting to detect persons and cars that are spatially sparse along many unoccupied regions, without the need to transform the 3D point cloud to a lower dimensional space.

The Velodyne LIDAR scans corresponding to both of Frame (a) and Frame (b) are

Figure 5.4. Original Frame (b) from RGB sequence drive 2011_09_28 drive_0016 #32.

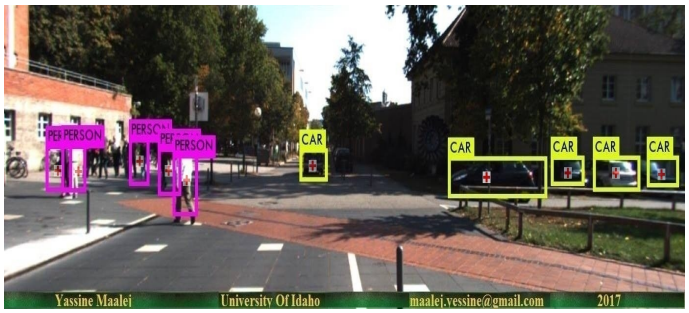

Figure 5.5. Detection of Frame (b) objects and their center of gravity.

plotted in Fig. 5.7 and in Fig. 5.8, respectively. The black centered area in both figures is the car that is equipped with the 360° Velodyne spinning laser scanner. The circled points in both figures represent the free space contour lines where no obstacles have been encountered (i.e., each circle represents points of equal distance from the vehicle with no objects in them). All cars and persons in both of Frame (a) and Frame (b) are represented by more dense dots (almost creating a 3D surface) due to the reflections of the laser beams from these objects. We note that the other black areas without circled points nor objects correspond to zones in which the LIDAR beams were blocked by obstacles. Consequently, the LIDAR scans cannot provide any knowledge of what is in these zones.

Fig. 5.9 depicts the 3D manifolds of the detected car and people objects from the LIDAR scans of both Frame (a) and Frame (b). The relative position of each object with respect to the Velodyne LIDAR Scanner position is represented by an *(x,y,z)* triplet in the 3D space. As in Fig. 5.6, the manifolds represent the adjacency relations between

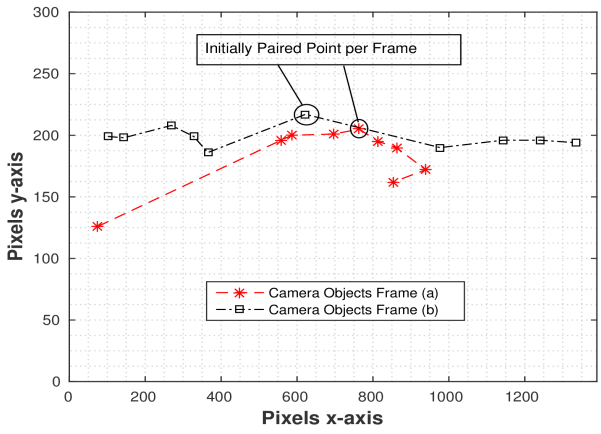

Figure 5.6. Pixel-wise manifolds of the recognized objects for both Frame (a) and Frame (b).

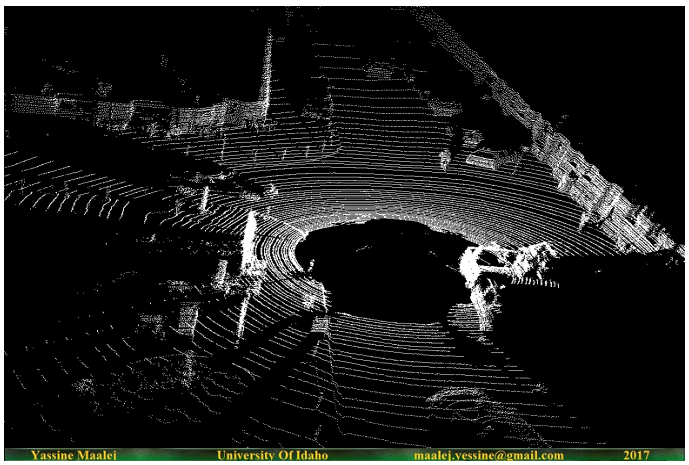

Figure 5.7. Point cloud scan corresponding to Frame (a).

the centers of gravity of the detected objects in each of the frames, which represents the approximate distances between them in the 3D space.

We note that the manifolds representing the detected objects from the LIDAR point clouds contain a larger number of persons and cars compared to those detected from the camera in Fig. 5.6. For example, the undetected shaded vehicle next to the leftmost detected vehicle in Fig. 5.5 by the camera was detected by the LIDAR point clouds as shown in Fig. 5.9. Moreover, the objects that are behind the camera were captured by the LIDAR spinning scans.

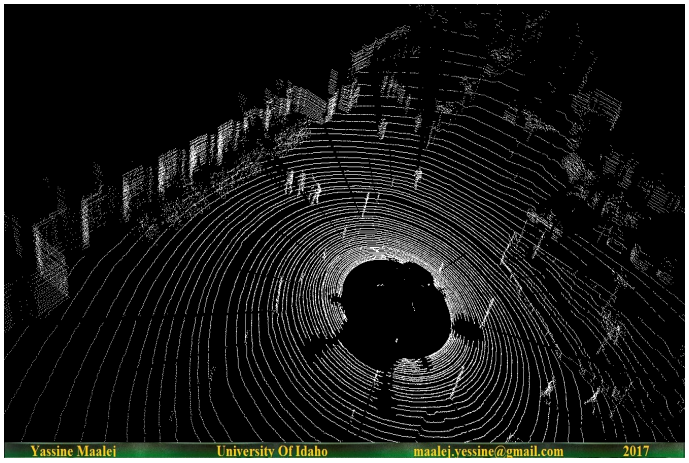

Figure 5.8. Point cloud scan corresponding to Frame (b).

5.1.3 V2V BSMs Generation from KITTI Benchmark Objects

The lack of time-stamped V2V BSMs data that corresponds to camera frames and LIDAR scans have encouraged us to dynamically generate the V2V related messages of the recognized objects from the LIDAR scans. We chose the LIDAR scans over the camera feeds to generate BSMs as usually BSMs can be received from larger distances that the range of cameras and thus should include a larger number of vehicles. Moreover, since we are using the camera as our reference alignment set (i.e., our methods aligns both the LIDAR and V2V data sets with camera data set), it is better generate the V2V BSMs from another set to have stronger discrepancies in the data sets and make the alignment more challenging. Note that, although they are generated from LIDAR data in our testing, the V2V BSM data sets and their corresponding manifolds will be significantly different from those of the LIDAR data sets, since the former ones include objects identified as "Car" only. Indeed, pedestrians do not typically generate BSMs, and thus objects identified as people in the LIDAR data set will not show up in the V2V BSM dataset.

For each identified vehicle by the LIDAR in every frame of the two aforementioned sequences, we generate a simple BSM only stating its position. The vehicles detected per-scan, whether moving or parked, are assumed to generate the V2V BSM beacons.

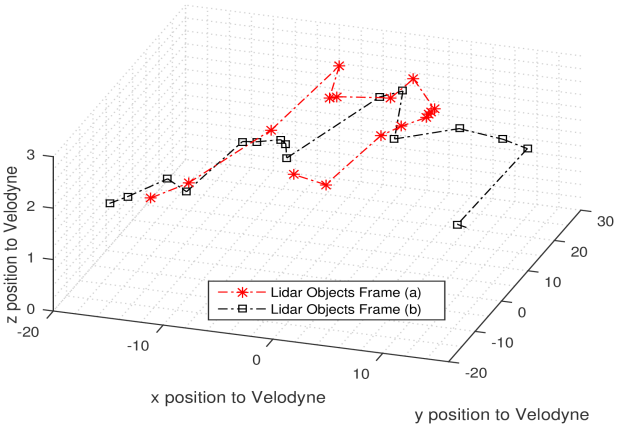

Figure 5.9. Manifolds of the detected objects from LIDAR point clouds corresponding to both Frame (a) and Frame (b).

It is important to note that, in practice, parked vehicles may or may not send beacons depending on the employed V2V standard. However, this fact will not affect the proposed alignment process, but will rather only increase or decrease the number of objects to be aligned. We employed the position *(x,y,z)* triplet of the object of the Velodyne LIDAR Scanner to emulate the GPS coordinate system *(Latitude, Longitude, Elevation)* that will be used in practical scenarios. Note that we did not include to our generate data set other typical BSM fields, such as messages count, temporary ID, brake system status, and acceleration, as they will not be used in the alignment process. In practical scenarios, the vehicle can easily extract the position information from the BSM for alignment purposes.

Fig. 5.10 depicts the manifolds of the generated V2V BSM data sets of Frame (a) and Frame (b), respectively. As expected, the shape of and the number of points in the manifolds in Fig. 5.10 are different from the ones formed by LIDAR objects in Fig. 5.9. This is due to the lack of objects identified as "Persons" from the former.

5.2 Proposed Alignment Approach

Having the manifolds of all vehicle sensing modes created, this section will illustrate the proposed approach to align these three manifolds, thus linking each object in each

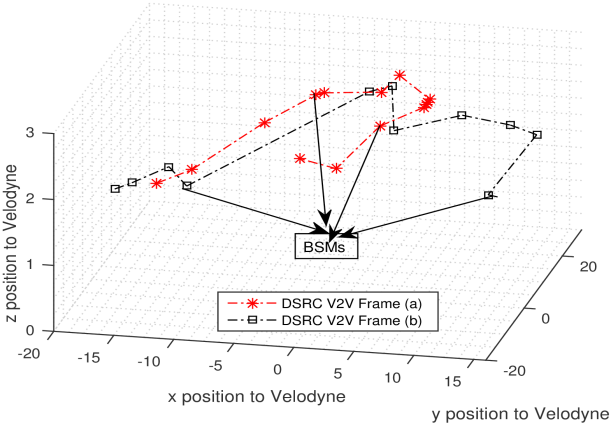

Figure 5.10. Adjacency of Recognized Objects from DSRC.

data set to its corresponding ones in the two others. Objects with no correspondences (e.g., objects detected in one data set but not the other) can then be added to the global knowledge of the vehicle, thus creating one enriched scene for the vehicle's surroundings.

Our proposed approach to align these three data sets is done by aligning both the LIDAR and V2V data sets individually with the camera data set. This choice is driven by the fact that the camera component is the most widely used component in autonomous vehicles, both at the research and prototyping level. The main two types of input data that the system takes are raw RGB frames from camera, LIDAR points cloud recorded in the driving sequences and V2V generated BSMs. Consequently, the alignment process is applied between camera, LIDAR and V2V data are principally the anchor boxes surrounding objects extracted from 2D Convnet, 3D position of objects retrieved after detection from 3D LIDAR and the V2V positions of cars. Once both data sets are aligned with the camera data sets, the correspondences between each two objects in them will be set naturally through their common corresponding object in the camera data set. Consequently, the rest of this section will focus on aligning only two data sets, one from the LIDAR or V2V data set, and the other being the camera.

The manifold alignment procedure between any two correlated data sets (i.e., having some correlation to one another in some space) is based on jointly embedding the objects of these data set in a lower dimensional space while both:

- Aligning points of initial correspondence between these data sets (i.e., points that are initially known to correspond to one another in the two data sets) in this lower dimensional space.
- Preserving the neighborhood correlation (i.e., the local structure) in each of them.

Once this embedding is done, each pair of points from the two data sets that have the closest proximity in this lower dimensional embedding are declared as corresponding to one-another. Clearly, our data sets of interest have clear correlation in the 3D space as a subset of each of them represent the exact same objects perceived by the vehicle through three different modes. They thus qualify to be aligned using manifold alignment. Given the above methodology, the proposed alignment process between our camera and LIDAR/V2V data sets will follow three main steps:

1. Neighborhood weight computation within each data sets (to preserve local structures).
2. Initial correspondence determination.
3. Semi-supervised alignment.

Each of these steps will be explained in details in the next three subsections.

5.2.1 Neighborhood Weight Computation

The neighborhood weights are a one dimensional representation of the distance/adjacency relations between each pair of objects in any one data set. A higher/lower weight value between two data points symbolizes their higher/lower proximity in their original space. Computing these weights are usually done through dimensionality reduction

techniques, such as locally linear embedding, heat kernels, etc. In our work, we select the locally linear embedding (LLE) technique [108] because it gives more importance in preserving the neighborhood correlation in the one-dimensional space.

To compute the neighborhood weights using LLE for any data set, the N data points having the closest Euclidian distance to each higher dimensional data point $\mathbf{t}^{(i)}$ are first identified as its neighbor set $\mathcal{N}(i)$. Let $\left[\mathbf{t}^{(\mathcal{N}(i,1))}, \ldots, \mathbf{t}^{(\mathcal{N}(i,N))}\right]$ be such set of points for data point $\mathbf{t}^{(i)}$. Given this set, the LLE thus computes the neighborhood weights of $\mathbf{t}^{(i)}$ using the following optimization problem, Eq. 5.1:

$$\arg\min_{W_{ij}} = \left\{ \left| \mathbf{t}^{(i)} - \sum_{j \in \mathcal{N}(i)} W_{ij} \mathbf{t}^{(\mathcal{N}(i,j))} \right|^2 \right\} \mathbf{s.t} \sum_{j \in \mathcal{N}(i)} W_{ij} = 1 \tag{5.1}$$

Clearly, a closer the point $\mathbf{t}^{(\mathcal{N}(i,j))}$ to $\mathbf{t}^{(i)}$ will have a higher weight W_{ij}. Points $\mathbf{t}^{(j)}$ that are not in $\mathcal{N}(i)$ will have $W_{ij} = 0$. The above optimization problem can be solved for each point $\mathbf{t}^{(i)}$ using a closed form solution as follows [108]. Defining the distance matrix $\mathbf{D}_i$ of point $\mathbf{t}^{(i)}$ as shown in Eq. 5.2:

$$\mathbf{D}_i = \begin{bmatrix} \mathbf{t}^{(i)} - \mathbf{t}^{(\mathcal{N}(i,1))} \\ \mathbf{t}^{(i)} - \mathbf{t}^{(\mathcal{N}(i,2))} \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{t}^{(i)} - \mathbf{t}^{(\mathcal{N}(i,N))} \end{bmatrix} \tag{5.2}$$

W_{ij} for all $\mathbf{t}^{(j)} \in \mathcal{N}(i)$ can be computed as in Eq. 5.3:

$$W_{ij} = \frac{\sum_{k=1}^{N} \left\{ (\mathbf{D}_i \mathbf{D}_i^T)^{-1} \right\}_{jk}}{\sum_{m=1}^{N} \sum_{n=1}^{N} \left\{ (\mathbf{D}_i \mathbf{D}_i^T)^{-1} \right\}_{mn}} \tag{5.3}$$

Note that $\left\{ \left(\mathbf{D}_i \mathbf{D}_i^T \right)^{-1} \right\}_{uv}$ is the element of the uth row and the vth column in the inverse of matrix $\mathbf{D}_i \mathbf{D}_i^T$. By repeating this procedure for all data points in each of the data sets, we obtained all required neighborhood weights for the alignment process.

5.2.2 Initial Correspondence Determinations

The second step of the alignment process consists of determining at least one point of correspondence between the camera and LIDAR/V2V data set. Some techniques were employed in the literature to estimate these initial correspondences based on the geometry of the data sets [97]- [80] . The main idea behind these techniques is to match some local geometries within the data sets, and declare points having a few highly similar local geometries as corresponding. In most cases, this requires a long search among all combinations of points to find high similarities and thus close to correct correspondences, which sometimes increases the complexity of the process.

Luckily, the considered data sets already possess some geometric references that can somewhat relate them to one-another. For instance, the front camera and LIDAR are usually aligned in the vehicle, and it is easy to roughly determine the LIDAR reflected beams (and thus its corresponding detected objects if any) from a certain direction/range corresponding to a certain direction/range of the camera picture (and thus its detected object). We can then use one of these objects from the same exact direction and distance from the vehicle to tag them as initial correspondences. For example, we can pick the initial point of correspondence between these two data sets to be the ones representing the one or few furthest point(s) captured by both the camera and LIDAR within the same direction(s) from the vehicle. If we apply this approach for Frame (b), we can align,

and estimate that the leftmost identified vehicle in Fig. 5.5 with the vehicle identified by the LIDAR beam reflected from that direction. This approach is also suitable for the alignment with V2V as we can estimate the distance from the camera and/or LIDAR to this farthest recognized object in a certain direction and match it to the most likely received GPS that matches it in distance and direction from the vehicle. Consequently, we will employ this approach to identify one or few point of correspondence across the three data sets, and employ them in the alignment process.

It is important to mention that, although some errors may occur in this initial alignment process, but so could happen in the previously proposed techniques [80]- [97] given the exact same three data sets. Most importantly, the error in our case will be in a slight range within the entire geometry of the data sets, and we can still get good alignment results for the other objects.

5.2.3 Semi-Supervised Alignment

As aforementioned, the semi-supervised alignment of two manifolds is done by jointly embedding their data points in a lower dimensional space while both aligning the initial points of correspondence and preserving the neighborhood correlation in each of them. The problem of aligning between the camera data set (that we will denote by $\mathcal{X}$) to the LIDAR/V2V data set (that we will tab by $\mathcal{Y}$) can be expressed as shown in Eq. 5.4:

$$\arg\min_{\mathrm{f,g}} \left\{ \lambda^x \sum_{i,j} [f_i - f_j]^2 W_{ij}^x + \lambda^y \sum_{i,j} [g_i - g_j]^2 W_{ij}^y + \mu \sum_{i \in \mathcal{P}} |f_i - g_i|^2 \right\} \tag{5.4}$$

where $f = [f_1, .., f_X]^{(T)}$ and $g = [g_1, .., g_Y]^{(T)}$ are vectors in $\mathbb{R}^X$ and $\mathbb{R}^Y$ of the X camera points and Y LIDAR/V2V points, respectively, $\mathcal{P}$ is the set of paired points between $\mathcal{X}$ and $\mathcal{Y}$, whereas W_{ij}^x and W_{ij}^y are the neighborhood weights between the ij pair of points within data sets $\mathcal{X}$ and $\mathcal{Y}$, respectively.

Clearly, this problem is a three-objective function optimization, with weighting factors λ^x, λ^y and μ. The first two terms aim to separately find the vectors $\mathbf{f}$ and $\mathbf{g}$ that are separately preserving the neighborhood structures of the points in $\mathcal{X}$ and $\mathcal{Y}$, respectively. Indeed, minimizing the first term will occur by attributing a smaller $f_i - f_j$ term (i.e., close values of f_i and f_j) for any $i-j$ pair of points in $\mathcal{X}$ with larger W^x_{ij}. Minimizing the second term plays the exact same role with the elements of vector $\mathbf{g}$ to preserve in it the neighborhood structure of $\mathcal{Y}$. The third term aims to equalize (i.e., align) the elements in $\mathbf{f}$ and $\mathbf{g}$ that correspond to paired points in $\mathcal{X}$ and $\mathcal{Y}$. Indeed, it penalizes discrepancies between f_i and g_i corresponding to each point $i \in \mathcal{P}$.

With some simple manipuation, Eq. 5.4 can be re-written as Eq. 5.5:

$$\arg\min_{\mathrm{f,g}} \left\{ \lambda^x \mathbf{f}^T L^x \mathbf{f} + \lambda^y \mathbf{g}^T L^y \mathbf{g} + \mu(\mathbf{f}-\mathbf{g})^T(\mathbf{f}-\mathbf{g}) \right\} \tag{5.5}$$

where $L^x = [L^x_{ij}]\ \forall\ i,\ j \in \mathcal{X}$, such that:

$$L^x_{ij} = \begin{cases} \sum_j W^x_{ij}, & i = j \\ -W^x_{ij} & j \in \mathcal{N}_i \\ 0 & \text{Otherwise} \end{cases} \tag{5.6}$$

while $L^y = [L^y_{ij}]\ \forall\ i, j \in \mathcal{Y}$ such that L^y_{ij} is defined by replacing each W^x_{ij} by W^y_{ij} in Eq. 5.6. In general, the problem in Eq. 5.5 is ill-defined. Nonetheless, imposing a hard constraint to make $f_i = g_i\ \forall\ i \in \mathcal{P}$ (i.e. as $\mu \to \infty$), the problem in Eq. 5.5 is be re-casted as an eigenvalue problem. Define vector $\mathbf{h}$ as:

$$\mathbf{h} = \begin{bmatrix} \mathbf{f}_{\mathcal{P}} = \mathbf{g}_{\mathcal{P}} \\ \mathbf{f}_{\mathcal{Q}^x} \\ \mathbf{g}_{\mathcal{Q}^y} \end{bmatrix} \tag{5.7}$$

where $\mathcal{Q}^x = \mathcal{X}/\mathcal{P}$ and $\mathcal{Q}^y = \mathcal{Y}/\mathcal{P}$. In other words, $\mathbf{h}$ is vector structured in a way that it begins with elements of $\mathbf{f}$ and $\mathbf{g}$ corresponding to the paired points in $\mathcal{P}$, followed by the remaining (i.e., unpaired) elements of $\mathbf{f}$, and ends with the remaining (i.e., unpaired) elements of $\mathbf{g}$.

Having this vector defined, and setting $\mu \to \infty$ in (5.5), we can re-write the problem as Eq. 5.8:

$$\arg\min_{\mathrm{h}} \left\{ \frac{\mathbf{h}^T L^z \mathbf{h}}{\mathbf{h}^T \mathbf{h}} \right\} \qquad \text{s.t.} \quad \mathbf{h}^T 1 = 0 \tag{5.8}$$

where:

$$L^z = \begin{bmatrix} \lambda^x L^x_{\mathcal{P}\mathcal{P}} + \lambda^y L^y_{\mathcal{P}\mathcal{P}} & \lambda^x L^x_{\mathcal{P}\mathcal{Q}^x} & \lambda^y L^y_{\mathcal{P}\mathcal{Q}^y} \\ \lambda^x L^x_{\mathcal{Q}^x\mathcal{P}} & \lambda^x L^x_{\mathcal{Q}^x\mathcal{Q}^x} & \mathbf{0} \\ \lambda^y L^y_{\mathcal{Q}^y\mathcal{P}} & \mathbf{0} & \lambda^y L^y_{\mathcal{Q}^y\mathcal{Q}^y} \end{bmatrix} \tag{5.9}$$

with $L^x_{\mathcal{I}\mathcal{J}}$ ($L^y_{\mathcal{I}\mathcal{J}}$) defined as the sub-matrix of $L^x(L^y)$ corresponding to the rows indexed by the elements in $\mathcal{I}$ and the columns indexed by the elements in $\mathcal{J}$. λ^x and λ^y are computed following the work in [80] as: $\lambda^x = X/(X+Y)$ and $\lambda^y = Y/(X+Y)$.

The solution of the problem in Eq. 5.8 is known to be the eigenvector $\mathbf{h}$ that corresponds to the smallest non-zero eigenvalue of L^z. Finding the optimal vector $\mathbf{h}^*$ naturally defines the optimal vectors $\mathbf{f}^*_{\mathcal{Q}^x}$ and $\mathbf{g}^*_{\mathcal{Q}^y}$. By finding the elements of closest distance from these two latter vectors, the corresponding unpaired points in $\mathcal{X}$ and $\mathcal{Y}$ are paired together. This concludes the alignment process.

Note that the vectors $\mathbf{f}^*_{\mathcal{Q}^x}$ and $\mathbf{g}^*_{\mathcal{Q}^y}$ may not (and most probably won't) be of the same dimension, and thus the remaining points in the longer vector should correspond to objects that were detected by one mode but not the other. These points can thus be added to the global vehicle understanding its surrounding while respecting its neighborhood correlations (i.e., weights) within its corresponding data set. When this alignment and object addition operation is repeated for between the camera-LIDAR and camera-V2V data set pairs, this results in an enriched construction of the vehicle surroundings that no

single mode of information (camera, LIDAR, or V2V) can attain individually.

5.3 Alignment Accuracy and Gain Testing

In this section, we aim to test both the accuracy of our proposed alignment scheme and its gain in enriching vehicles' knowledge about its surroundings using multi-modal inputs. In these tests, we will use the KITTI benchmark as our source of camera feeds and LIDAR scans of vehicle surroundings, due to its wide approval and adoption in the testing of various autonomous vehicle recognition and driving systems. The raw data recording of both of the driving sequences contain color stereo sequences recorded with a 0.5 Megapixels camera stored in png format, 3D Velodyne LIDAR point clouds stored as binary float matrix, 3D GPS/IMU data for location and timestamps information stored in text files.

The only problem with this benchmark is that it does not include a library of BSMs as it did not assume any V2V communications. We will first generate a V2V BSM streams for the vehicles in the entire sequences and perform the tests using these generated BSMs. The driving sequences 2011_09_28_*drive*_0016 and 2011_09_26_*drive*_0005 are recorded during daytime in non-rush hour around the campus of Karlsruhe Institute of Technology and metropolitan area of Karlsruhe in Germany, respectively. The former sequence has a total size of 0.7 GB, 192 frames of 1392*512 pixels each and contains 11 distinct cars and 9 distinct persons. while the latter sequence has a total size of 0.6 GB, 160 frames of 1392*512 pixels each and contains 12 distinct cars and 3 distinct persons. We will then illustrate the detailed alignment accuracy and gain results for Frame (a) and Frame (b). Finally, we present the aggregate alignment accuracy on the entire two aforementioned sequences.

In the remainder of this section, we define the alignment accuracy (in percentage) as the accuracy in mapping all the points from $\mathcal{X}$ (i.e., camera data set) to $\mathcal{Y}$ (LIDAR/V2V data sets). More formally, it is defined as the percentage of points in $\mathcal{X}$ mapped to wrong

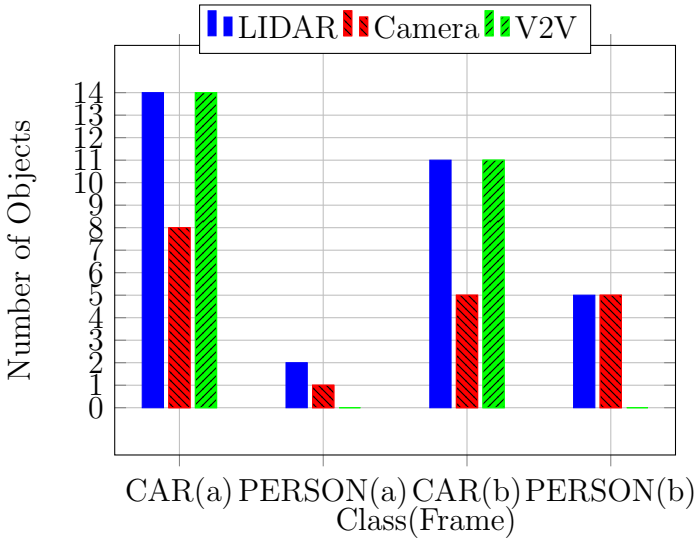

Figure 5.11. Number of Objects per Class for both Frame (a) and Frame (b).

(i.e., non-actually corresponding) points in $\mathcal{Y}$ normalized by the total number of points of $\mathcal{X}$ (i.e., the camera data set) We also define the alignment gain (in percentage) as the percentage of added objects from the alignment process (i.e., from mapping LIDAR/V2V detected objects to the camera detected objects), normalized to the original number of detected objects in the camera data set.

5.3.1 Alignment Results for Frame (a) and Frame (b)

To understand the accuracy and gain results of our alignment process on Frames (a) and (b), we first illustrate in Fig. 5.11 the number of detected objects per class in these two frames. We note that, for both frames, the LIDAR scans have detected more cars and persons than the camera feeds. Due to the method with which they are obtained, the V2V data set have the same number of cars as the LIDAR data set, but no persons as expected.

Fig. 5.12 and Fig. 5.13 represent the detailed camera-to-LIDAR and camera-to-V2V alignment results, respectively, for Frame (a).

Similarly, Fig. 5.14 and Fig. 5.15 represent the detailed camera-to-LIDAR and

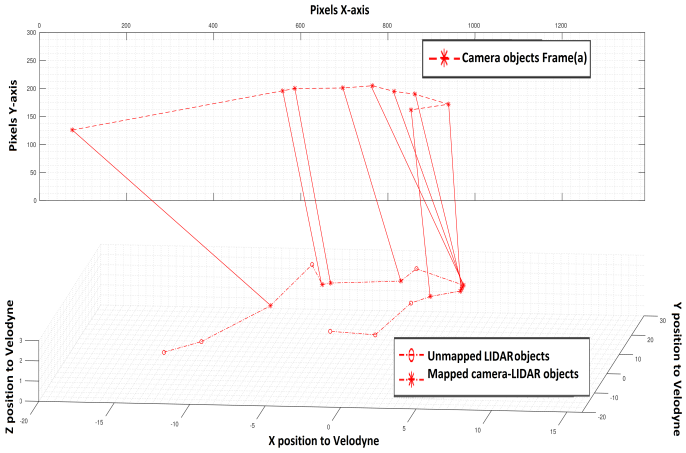

Figure 5.12. Camera-LIDAR Object Alignment of Frame (a).

camera-to-V2V alignment results, respectively, for Frame (b).

The top portion of all four figures illustrate the camera dat set points in 2D, where as the bottom figure illustrate the LIDAR or V2V data set points in 3D. The first result to report from these figure is that the alignment accuracy was 100% for all the four alignment tasks. In other words, all data points from the camera were mapped to the points of the exact same objects in the LIDAR and V2V data sets. The second observation is the presence of objects from the camera data set that were not aligned with points from the V2V data set for both Frame (a) and Frame (b), which are represented by the circled points in the top portions of Fig. 5.13 and Fig. 5.15, respectively. By comparing the positions of these points with the recognized objects of Frame (a) and Frame (b) in Fig. 5.3 and Fig. 5.5, respectively, it is easy to notice that all such unmapped points from the camera data set represent only pedestrians. This behavior is indeed expected as it is known that V2V data set does not include points representing pedestrians. We can note from the top portions of Fig. 5.12 and Fig. 5.14 that this case did not occur when aligning with the LIDAR data set, due to the ability of the LIDAR to detect "Person" objects, and their proper mapping to those detected by the camera. These facts also clarify the number of mapped points between camera and LIDAR exceed those mapped between the

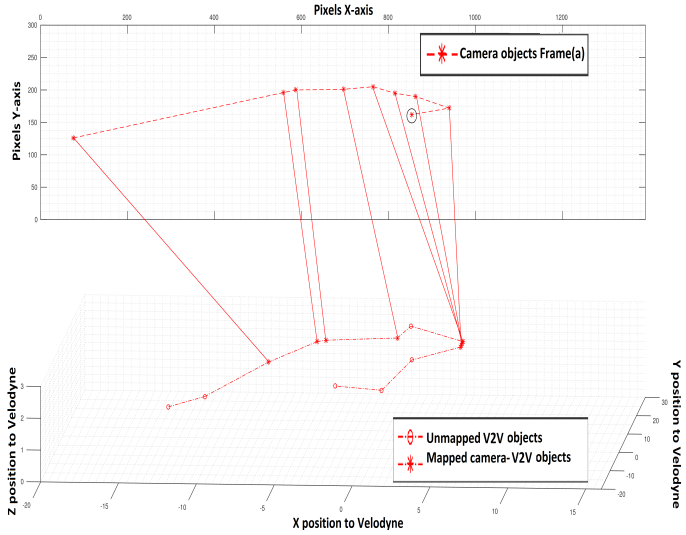

Figure 5.13. Camera-V2V Object Alignment of Frame (a).

camera and V2V objects.

Another important observation is the presence of unmapped data points in the bottom portions of all four figures, represented by different markers. These points consists of the objects detected by the LIDAR scans and V2V BSMs but were not detected by analyzing the camera feeds for both frames. Consequently, these points constitute the components contributing to the alignment gain as they enrich the vehicle recognition of its surrounding compared to the sole use of camera feeds (widely used in autonomous vehicle experiments and prototypes). The "Car", "Person", and overall alignment gains, obtained for both LIDAR and V2V alignment process with the camera information, are illustrated in Fig. 5.16. We can see that up to 120%, 50%, and 78% alignment gains can be obtained by in the car, person, and overall objects, respectively. This clearly show the merits of our multi-model surrounding recognition system. We can also notice the overall LIDAR gains are always equal or larger than the V2V gains in this test, because the V2V data sets represent a fraction of the LIDAR data set. However, in practical scenarios, V2V gains can jump significantly as V2V BSMs cover larger distances (around 250 meters) than those scannable by cameras and LIDARs, and thus can significantly enrich the knowledge

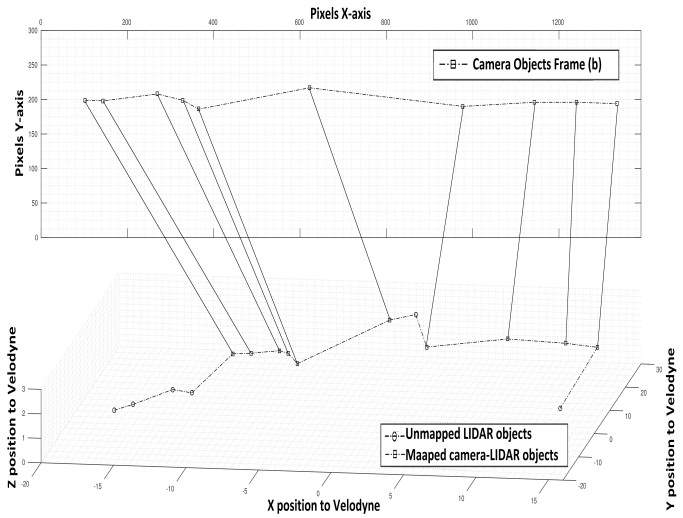

Figure 5.14. Camera-LIDAR Object Alignment of Frame (b).

about a much wider range of its surroundings.

5.4 Alignment Results over Entire Sequences

To extend our testing beyond two limited frames (i.e., only Frame (a) and Frame (b)), this section illustrates the alignment results for two driving sequences from the KITTI data set in terms of both accuracy and gain. In such more practical setting of an autonomous vehicle driving, we cannot know in advance the number of objects in each frame. Consequently, the selection of the percentage of neighbors used to create the weights of the graph are performed using the best values obtained from both of the two frames (a) and (b) as well as previous studies on manifold alignment (usually in the range between 30% to 40%).

For both driving sequences 2011_09_26_*drive*_0005 (consisting of 160 frames, the 117-th being Frame (a)) and 2011_09_28_*drive*_0016 (consisting of 192 frames, the 32nd being Frame (b)), we tested the alignment accuracy performance and gains. Since immediate subsequent frames do not significantly change in nature, performing alignment of such frames will not yield to any tangible new information, thus adding non-justified computa-

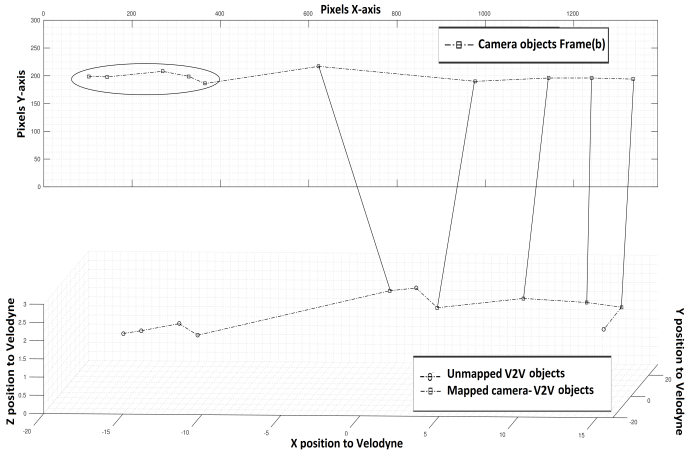

Figure 5.15. Camera-V2V Object Alignment of Frame (b).

tional burden on the system. Consequently, we consider a practical scenario by aligning the received multimodal data every 5 frames of each sequence drive in order to capture tangible dynamism of the objects without enduring excessive unnecessary computational loads. It is important to note that the number of identified objects as cars and vehicles vary from one frame to another, and from one sequence to the other, as presented in Table 5.1.

For each of these frames, we first prepared the camera and LIDAR data sets as explained in Section 5.1, and generated BSMs for all vehicles in each of the considered scenes as highlighted in Section 5.1.3. We then preformed the camera-LIDAR and camera-V2V alignment processes for each of these frames, using 33% (as identified from Frame (a)) and 40% (as identified from Frame (b)) of each point's neighbors in its weights computations for sequences 2011_09_26_*drive*_0005 and 2011_09_28_*drive*_0016, respectively. We recorded the alignment accuracy and gains for each of the aligned frames and finally averaged these results over the entire number of used frames in each of the sequences.

The per-frame alignment accuracy for both sequences are presented in Table 5.1, and the average alignment accuracy results are illustrated in Fig. 5.19. We can first notice that our proposed scheme achieved 100% of alignment accuracy per-frame in many frames

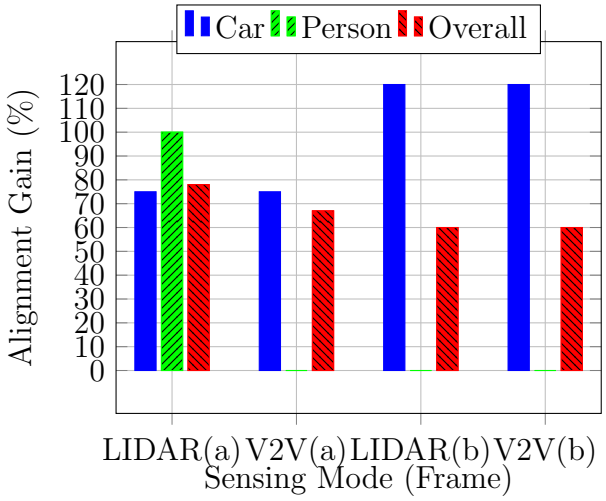

Figure 5.16. Alignment gains for Frame (a) and Frame (b).

of both of driving sequence. This result demonstrates the fact the neighboring distance in pixels is very meaningful and describes the objects' neighborhood as in the LIDAR and V2V data sets, especially when objects are sparse. Some alignment results were less accurate (50%, 66.66% and 75%) due to some instances of overlapping and very close objects (especially for persons). The resulting variations in accuracy levels is also strongly related to the number of objects, the selected percentage of neighbors as well as how the objects are dispatched in the 2D camera space, 3D LIDAR representation and V2V environment. Moreover, we notice a better achieved alignment accuracy achieved in the camera-V2V alignment compared to the camera-LIDAR, which can be explained both as in the previous section and the fact of not having human alignment in the V2V cases (which is one of the causes of lower alignment performance due to the object overlaps and higher proximity). Looking at Fig. 5.19, we can observe between 74% and 92% of average overall alignment accuracy for vehicles and 50% to 78% average accuracy for persons. These results are quite encouraging, especially on the vehicle side, as they exhibit high accuracy of alignment, thus yielding to a better understanding of the observed objects (especially vehicles) by each autonomous vehicle. Even for the cases of mis-alignment, they occur due to same object overlap or close proximity, which does not represent a sever

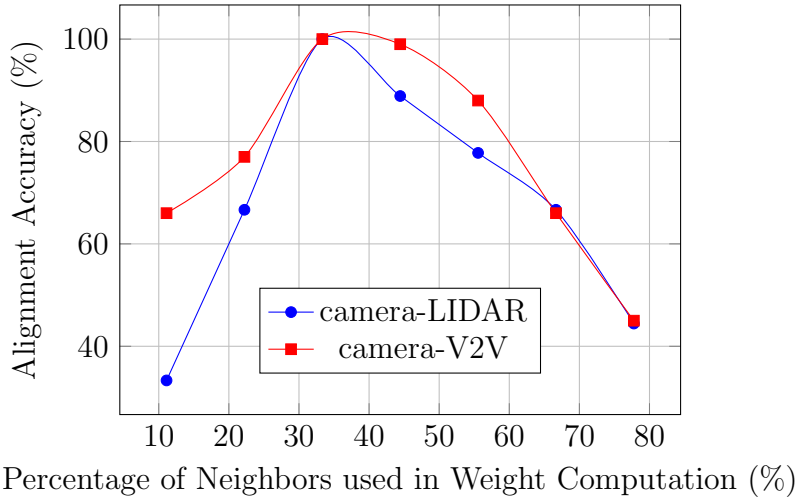

Figure 5.17. Effect of number of neighbors on the alignment accuracy for Frame (a).

drawback to the overall vehicle recongnition of its surroundings. Indeed, all mis-aligned objects are all from the same type and are occupying the same neighborhood within the vehicle surroundings, thus not significantly affecting its decisions. Table 5.3 illustrates the alignment gains obtained in each frame in both considered sequences and the average gains across all frames per are plotted in Fig. 5.20. The depicted results show significant gains (from 17% and up to 150%) in the knowledge of the vehicle using the proposed multimodal surrounding recognition approach. Again, the alignment gains of the objects of type persons in the V2V data set are null because pedestrian do not send BSMs.

We finally studied the sensitivity of the alignment accuracy in response to the change in the percentage of points selected for neighborhood weight calculations. Indeed, the used number of neighbors (N) used to compute these weights, and its percentage (i.e., ratio) with the respect to the entire data set size, can be crucial factors in the accuracy of the alignment. For Frame (a) and Frame (b) respectilvey, Fig. 5.17 and Fig. 5.18 depict the accuracy performance of both camera-LIDAR and camera-V2V alignments against the percentage of the number of neighbors in the neighborhood weight computation process, normalized by the total size of each of the data sets. Both figures clearly justify our initial intuition by showing the significant effect of this parameter on the alignment accuracy.

Table 5.1. Alignment Accuracy per Sequence Drive

ID	Sequence 09_26_0005				Sequence 09_28_0016			
	CAR		PERSON		CAR		PERSON	
	LIDAR	V2V	LIDAR	V2V	LIDAR	V2V	LIDAR	V2V
0	100	100	100	-	75	100	83.33	-
5	100	100	100	-	75	100	83.33	-
10	100	100	100	-	75	100	80	-
15	50	100	0	-	80	80	83.33	-
20	50	100	0	-	60	80	80	-
25	50	100	50	-	80	100	80	-
30	100	100	100	-	100	100	100	-
32	x	x	x	-	100	100	100	-
35	50	100	50	-	80	100	83.33	-
40	50	50	0	-	100	100	100	-
45	50	50	50	-	80	80	80	-
50	50	100	0	-	66.66	83.33	60	-
55	66.66	100	0	-	80	100	60	-
60	50	100	0	-	100	100	57	-
65	50	50	0	-	100	100	100	-
70	50	50	50	-	100	100	100	-
75	50	100	50	-	75	75	60	-
80	100	100	100	-	75	100	80	-
85	100	100	100	-	100	100	83.33	-
90	66.66	66.66	50	-	100	100	71.42	-
95	50	50	50	-	100	100	71.42	-
100	75	75	50	-	75	100	55.55	-
105	83.33	100	50	-	100	100	83.33	-
110	100	100	100	-	75	100	66.66	-
115	100	100	100	-	66.66	66.66	66.66	-
117	100	100	100	-	x	x	x	x
120	100	100	100	-	66.66	100	80	-
125	87.5	87.5	0	-	66.66	66.66	75	-
130	100	100	100	-	66.66	66.66	80	-
135	100	100	100	-	33.33	33.33	80	-
140	90	100	0	-	100	100	40	-
145	80	100	100	-	100	100	50	-
150	100	100	100	-	66.66	100	80	-
153	90	90	66.66	-	66.66	100	80	-

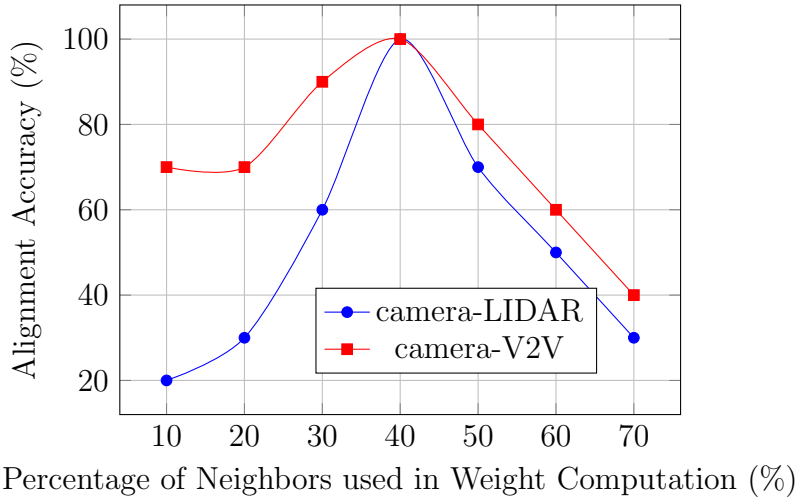

Figure 5.18. Effect of number of neighbors on the alignment accuracy for Frame (b).

We can also observe that having a mid-range percentage of the points in the neighborhood weight computation (from 30 to 40 %) results in much better accuracy compared to both lower (10 to 25 %) and higher ranges (50% and above). The degradation in the lower range can be explained by the effect of outliers (i.e., points seeming to be close, especially in the camera and LIDAR scans, while not being truly so) in weight computations when the chosen percentage of neighbors is small. When the percentage increases to the mid range, the number of considered neighbors in the computation becomes quite larger, thus diminishing the effect of the smaller percentage of outliers in misrepresenting neighborhood correlation. However, as this percentage grows beyond a certain limit, the algorithm will relate each point to a lot of points, some of which definitely not being in its true vicinity. Thus, the concept of neighbourhood dilutes which results in the exhibited performance degradation.

Another final notice about Fig. 5.17 and Fig. 5.18 is that the camera-V2V alignment process always results in a better performance. This can be interpreted by the fact that the V2V data set object is known to all vehicles. Consequently, it is easy in this case to reject any alignment between a point in the V2V data set with a point representing a person from the camera data set. In other words, we can easily restrict the alignment

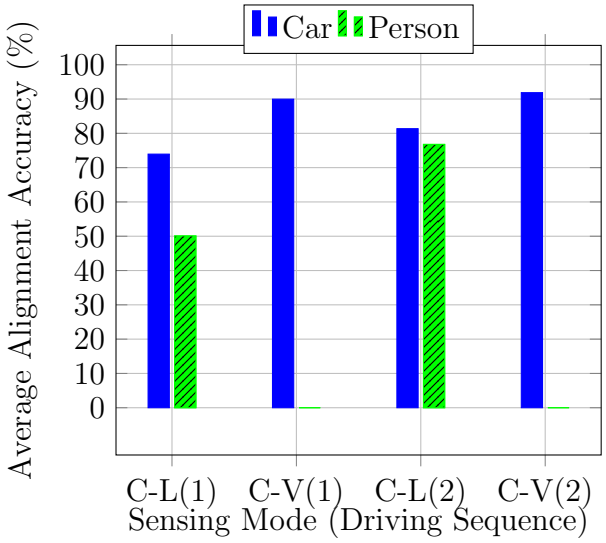

Figure 5.19. Average Alignment Accuracy of the driving Sequences (1): Sequence 09_26_0005 and (2): Sequence 09_28_0016.

between the V2V data set and only the objects identified as cars in the camera data set, which reduces the number of points to be aligned, removes all potential errors of aligning cars to persons (which can be the case in with LIDAR), and thus results in a better overall alignment performance.

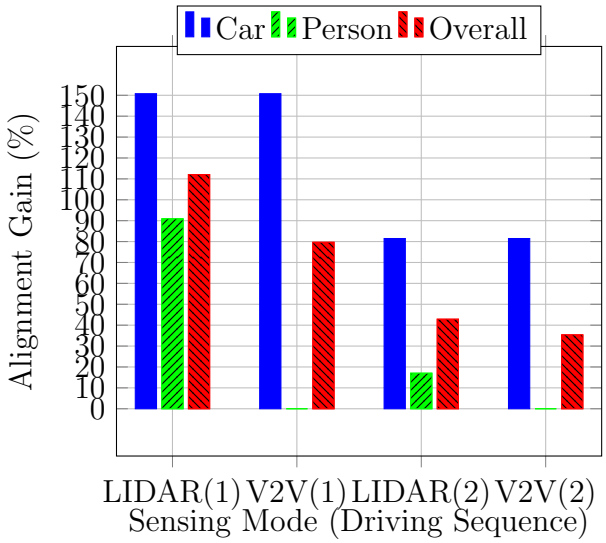

Figure 5.20. Average Alignment Gains of the driving Sequences (1): Sequence 09_26_0005 and (2): Sequence 09_28_0016.

Table I. NUMBER OF OBJECTS PER CLASS FOR BOTH OF THE DRIVING SEQUENCES

ID	Sequence 09_26_0005						Sequence 09_28_0016					
	Camera		LIDAR		V2V		Camera		LIDAR		V2V	
	CAR	PERSON	CAR	PERSON	CAR	PERSON	CAR	PERSON	CAR	PERSON	CAR	PERSON
0	1	2	4	6	4	-	4	6	9	7	9	-
5	1	2	4	5	4	-	4	6	9	7	9	-
10	1	1	4	5	4	-	4	5	11	5	11	-
15	2	1	5	4	5	-	5	6	11	5	11	-
20	2	1	4	4	4	-	5	5	11	6	11	-
25	2	2	4	3	4	-	5	5	11	5	11	-
30	2	2	3	3	3	-	5	5	10	6	10	-
32	x	x	x	x	x	x	5	5	10	6	10	-
35	2	2	4	3	4	-	5	6	10	8	10	-
40	2	1	4	3	4	-	4	6	10	9	10	-
45	2	2	4	3	4	-	5	5	11	8	11	-
50	2	1	5	3	5	-	6	5	11	8	11	-
55	3	1	5	2	5	-	5	5	10	7	10	-
60	2	1	5	2	5	-	4	7	10	6	10	-
65	2	1	6	2	6	-	4	5	10	6	10	-
70	2	2	6	2	6	-	4	4	10	6	10	-
75	2	2	6	2	6	-	4	5	8	5	8	-
80	1	2	6	2	6	-	4	5	8	5	8	-
85	1	2	6	2	6	-	3	6	8	5	8	-
90	3	2	7	2	7	-	4	7	8	6	8	-
95	2	2	11	2	11	-	3	7	5	6	5	-
100	4	2	10	2	10	-	4	9	4	6	4	-
105	6	2	12	2	12	-	4	6	4	5	4	-
110	7	1	13	2	13	-	4	6	4	5	4	-
115	8	1	14	2	14	-	3	6	4	5	4	-
117	8	1	14	2	14	-	x	x	x	x	x	x
120	7	1	14	2	14	-	3	5	3	5	3	-
125	8	1	14	2	14	-	3	4	3	6	3	-
130	10	1	14	2	14	-	3	5	3	6	3	-
135	9	1	13	2	13	-	3	5	3	6	3	-
140	10	1	12	2	12	-	3	5	4	6	4	-
145	10	1	13	2	13	-	2	4	4	5	4	-
150	10	3	11	2	11	-	3	5	3	5	3	-
153	10	3	12	2	12	-	3	5	3	6	3	-

Table III. ALIGNMENT GAINS FOR THE DRIVING SEQUENCES

ID	Sequence 09_26_0005						Sequence 09_28_0016					
	CAR		PERSON		OVERALL		CAR		PERSON		OVERALL	
	LIDAR	V2V	LIDAR	V2V	LIDAR	V2V	LIDAR	V2V	LIDAR	V2V	LIDAR	V2V
0	300	300	200	-	233.33	100	125	125	16.66	-	60	50
5	300	300	150	-	200	100	125	125	16.66	-	60	50
10	300	300	500	-	350	150	175	175	0	-	77.77	77.77
15	150	150	300	-	200	100	120	120	0	-	45.45	54.54
20	100	100	300	-	166.66	66.66	120	120	20	-	60	60
25	100	100	50	-	75	50	120	120	0	-	70	60
30	50	50	50	-	50	25	100	100	20	-	60	50
32	x	x	x	x	x	x	100	100	20		60	50
35	100	100	50	-	75	50	100	100	33.33	-	63.63	45.45
40	100	100	200	-	133	33.33	150	150	50	-	90	60
45	100	100	50	-	75	50	120	120	60	-	90	60
50	150	150	200	-	166.66	100	83.33	83.33	60	-	72.72	45.45
55	66.66	66.66	100	-	75	50	100	100	40	-	70	50
60	150	150	100	-	133.3	100	150	150	0	-	45.45	54.54
65	200	200	100	-	166.6	133.33	150	150	20	-	77.77	66.66
70	200	200	0	-	100	100	150	150	50	-	100	75
75	200	200	0	-	100	100	100	100	0	-	44.44	44.44
80	500	500	0	-	166.66	166.66	100	100	0	-	44.44	44.44
85	500	500	0	-	166.66	166.66	166.66	166.66	0	-	44.44	55.55
90	133.33	133.33	0	-	80	80	100	100	0	-	27.27	36.36
95	450	450	0	-	225	225	66.66	66.66	0	-	10	20
100	150	150	0	-	66.66	100	0	0	0	-	0	0
105	100	100	0	-	75	75	0	0	0	-	0	0
110	85.71	85.71	100	-	87.5	75	0	0	0	-	0	0
115	75	75	100	-	77.77	66.66	33.33	33.33	0	-	0	11.11
117	75	75	100	-	78	66.66	x	x	x	x	x	x
120	100	100	100	-	100	87.5	0	0	0	-	0	0
125	75	75	100	-	77.77	66.66	0	0	50	-	28.57	0
130	40	40	100	-	45.45	36.36	0	0	20	-	12.5	0
135	44.44	44.44	100	-	50	40	0	0	20	-	12.5	0
140	20	20	100	-	27.27	18.18	33.33	33.33	20	-	25	12.5
145	30	30	100	-	36.36	27.27	100	100	25	-	50	33.33
150	10	10	0	-	15.38	7.69	0	0	0	-	0	0
153	20	20	0	-	23.07	15.38	0	0	20	-	12.5	0

Chapter 6: Extended Kalman Filters Applied to the 3D CNN for the Raw LIDAR Scans

In this Chapter, a novel approach is presented to track recognized 3D vehicle point clouds from LiDAR scans. This technique is based on tracing the 3D anchor boxes of these vehicles, recognized though convolutional neural networks (CNNs). Exploiting the 3D CNNs detection of vehicles and persons, and the EKF two-steps process for prediction and update, the proposed scheme guarantees the awareness of moving detected objects and improves the perception of Autonomous Vehicles (AV). The proposed scheme reduces the usage of the expensive detection process of feeding the point cloud to CNN by tracking the 3D rectangular coordinates containing already detected objects from early Velodyne scans of the driving sequences. The testing of the proposed method on the well-known KITTI data set, featuring LiDAR scans of realistic vehicular environments. Results show that the merits of the proposed scheme in achieving high tracking accuracy.

6.1 Related Work for tracking

With the recent advances of spinning LIDARs, such as Velodyne [103], and solid state LIDARs [40], various solutions are being developed by researchers and companies to enhance the 3D awareness of moving objects, defined as the activity layer, surrounding the self-driving cars. The mobility tracking based on LIDAR scans is not the objective itself, but it represents a crucial component toward High Definition (HD) 3D maps for autonomous vehicles. One suggested technique is motion-based tracking [32], which uses motion cues without any prior information about the objects and operates by initially detecting multiple motions in the LIDAR scene and segmenting moving objects using a Bayesian approach. This kind of approach is useful for specific applications, such as object

avoidance and free derivable path detection, but does not offer rich information about the tracked objects. In addition, the authors in [16] presented a detection and tracking of moving objects (DATMO) method that operates at every time step by building a local 2.5D grid that uses the last sets of the LIDAR measurements. Then, all the created grids are combined with localization data that are successively integrated into a unique local environment called 2.5D map. Consequently, the 2.5D grid will be compared with the updated 2.5D map after each scan to find out the 2.5D motion grid. Finally, the 2.5D motion grid is processed in order to find object level representation in the scene and the detected objects that are moving are tracked by using data association and Kalman filters. Alternatively, the authors in [85] attempted to adapt Bayesian filtering to the generated occupancy grids in order to predict the dynamics of the objects in the grids and combined them with the developed Fast Clustering and Tracking Algorithm (FCTA) to detect and track moving objects throughout consecutive grids. The aforementioned techniques are based on the association of the displacement occurring from one representation of the LIDAR point cloud to another. This is done in order to detect and track changes that are abstractly referred to as moving objects without knowing neither their type, their first occurrence in the scan or any other details. We present a different approach to solving the LIDAR scene understanding and tracking, by detecting class of objects in early scans using 3D CNNs, then tracking the objects while the car is moving. Various LIDAR based perception approaches for objects detection and recognition from a point cloud are developed in the literature. The work in [34] developed a solution based on LIDAR data only to explore the vehicle surroundings. This solution involves three consecutive phases; segmentation, fragmentation detection and clustering. Vote3Deep is developed in [41] for fast point cloud object detection using 3D CNN, in order to keep the key power of LIDAR as distance and objects 3D shapes and depth detection applied to the KITTI Vision Benchmark Suite [48] that offers raw LIDAR and labeled objects from point cloud. More simplistic and secure scheme that does not rely on incoming cloud data is developed

in [28], where a Multi-View 3D networks (MV3D) is developed to guarantee accurate 3D object detection applied to autonomous driving scenarios. Similarly in [117], the 3D point cloud and 2D front view images are fused via a CNN with the usage of a Region Proposal Network (RPN) that is used in the network's multiple layers to generate proposal region objects in the front view.

The goal of this Chapter is to merge the key features of LIDAR in giving accurate distances, 3D deep learning for object detection and EKF powerful tracking algorithm. We first parse raw LIDAR scans and execute object recognition modules to prepare the resulting data for tracking. Our proposed tracking approach is run separately for each 3D detected object and uses its information to predict the state of the 3D box before the next LIDAR scan gets loaded. Then, we update the measurement by using the new observation of the point cloud and correct the belief about the previous predicted state of the box. The main Kalman filter prediction parameters include position and speed of the 3D surrounding box, while the key update parameters are the difference between the actual and predicted measurement, the process and measurement combined covariance, the Kalman gain after update that contains the weight adjustement to the current prediction, the state vector and uncertainty matrix of the box , etc.

6.2 Object Detection and Recognition in LIDAR Scans from KITTI Driving Sequences

This section illustrates the adaptation of object recognition from LIDAR point clouds from an early scan to prepare the set of detected objects to be tracked. To simplify this process, we are not considering the recognition of every object from the LIDAR point clouds, since a tremendous number of class objects can be detected. Thus, we will restrict the recognition of the Vote3Deep 3D CNN, developed in [41], to identify only "Cars" and "Persons", since the remaining items do not represent major importance in the tracking process. The Vote3Deep 3D CNN uses feature-centric voting to detect persons and cars

Figure 6.1. First Scan in Sequence Drive 2011_09_26 drive_0005 #1.

that are spatially sparse along many unoccupied regions, without the need to transform the 3D point cloud to a lower dimensional space.

We present in Fig. 6.1, Fig. 6.2 and Fig. 6.3 initial frames for both of driving sequences 2011_09_26$drive$_0005, 2011_09_28_$drive$_0016 and 2011_09_26_$drive$_0106, respectively.

The black centered area in both figures is the car that is equipped with the 360° Velodyne spinning laser scanner. The circled points in both figures represent the free space contour lines where no obstacles have been encountered (i.e., each circle represent points of equal distance from the vehicle with no objects in them) . All cars and persons are represented by more dense dots due to the reflections of the laser beams from these objects. We note that the other black areas without circled points nor objects correspond to zones in which the LIDAR beams were blocked by obstacles. Consequently, the LIDAR scans cannot provide any knowledge of what is in these zones, as presented in Fig. 6.1 and Fig. 6.2.

Figure 6.2. First Scan in Sequence Drive 2011_09_28 drive_0016 #1.

6.3 EKFs Nonlinear Adaptations of the KFs

6.3.1 KF Formulation and Parameter descriptions

The KF's widely known closed form solution to the problem of recursive optimal filtering is guaranteed by respecting the following three conditions:

- All the states and the measurements obey a linear model.
- Every prior distribution is considered to be Gaussian.
- The noise in measurements and in motions are Gaussian.

The Eauqtions representing the general motion of the tracklet and the measurement model are shown in Eq. 6.1:

$$\begin{aligned} x_k &= F_{k-1}x_{k-1} + v_{k-1}, \quad v_k \sim \mathcal{N}(0, Q_{k-1}) \\ y_k &= H_k x_k + w_k, \qquad\quad w_k \sim \mathcal{N}(0, R_k) \end{aligned} \tag{6.1}$$

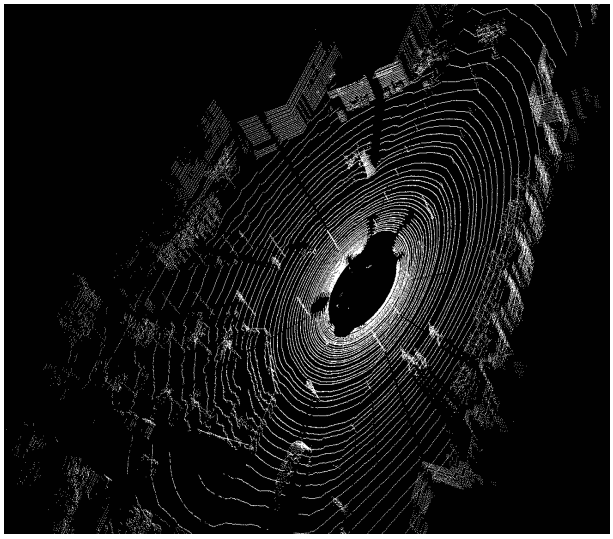

Figure 6.3. First Scan in Sequence Drive 2011_09_26 drive_0106 #1.

We only need the mean $\hat{x}_{k|k}$ and the covariance $P_{k|k}$ to represent the posterior density because of the Gaussian densities. Consequently, the key densities are formulated as follows:

$$p(x_{k-1}|y_{1:k-1}) \sim \mathcal{N}(\hat{x}_{k-1|k-1}, P_{k-1|k-1}) \tag{6.2}$$

$$p(x_k|y_{1:k-1}) \sim \mathcal{N}(\hat{x}_{k|k-1}, P_{k|k-1}) \tag{6.3}$$

$$p(x_k|y_{1:k}) \sim \mathcal{N}(\hat{x}_{k|k}, P_{k|k}) \tag{6.4}$$

The prediction phase described in KF is applied to find the results of the moments, and because of the linearity of the models then the prediction results in finding the covariance and the expected value of shifted and scaled linearily of Gaussian density as shown in Eq. 6.5.

$$\begin{aligned} \mathbb{E}[x_k|y_{1:k-1}] &\equiv \hat{x}_{k|k-1} = F_{k-1}\hat{x}_{k-1|k-1} \\ \mathrm{Cov}[x_k|y_{1:k-1}] &\equiv P_{k|k-1} = F_{k-1}P_{k-1|k-1}F_{k-1}^T + Q_k \end{aligned} \tag{6.5}$$

The update phase described in KF is decomposed in many parts that are formulated to resolve the expressions for the moments in the posterior density. Detailed equations are depicted as follows:

$$v_k = y_k - H_k \hat{x}_{k|k-1} \tag{6.6}$$

$$S_k = H_k P_{k|k-1} H_k^T + R_k \tag{6.7}$$

$$K_k = P_{k|k-1} H_k^T (S_k)^{-1} \tag{6.8}$$

The v_k part represents the difference between the predicted measurement of the 3D box and the actual measurement found in the next LIDAR frame. It is a zero-mean with a covariance equal to S_k that represents the expected measurement covariance. In addition, it is used to represent the region surrounding the predicted state where the actual measurements of the 3D box are more probable to be received from. The K_k part is the kalman gain that holds the measure of how much v_k needs to be counted in the posterior prediction phase.

Consequently, the posterior covariance and mean are presented as follows:

$$\begin{aligned} P_{k|k} &= P_{k|k-1} - K_k S_k K_k^T \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k v_k \end{aligned} \tag{6.9}$$

6.3.2 EKF non Linearity Extensions

EKF incorporates the nonlinear motion of the 3D tracklet with nonlinear measurement models from the KF libraries, that uses approximate linearization. Consequently, the motion and measurement models of the system:

$$\begin{aligned} x_{k|k-1} &= f_{k-1}(x_{k-1|k-1}) + q_{k-1} \\ y_k &= h_k(x_{k|k-1}) + r_k \end{aligned} \tag{6.10}$$

with nonlinear parts $f_{k-1}(\cdot)$ and $h_k(\cdot)$. We perform the linearization by first-order Taylor expansion for every function around the $\hat{x}_{k-1|k-1}$ and $\hat{x}_{k|k-1}$:

$$\begin{aligned} f_{k-1}(x_{k-1|k-1}) \approx f_{k-1}(\hat{x}_{k-1|k-1}) + f_{k-1}^{'} \\ (\hat{x}_{k-1|k-1})(x_{k-1|k-1} - \hat{x}_{k-1|k-1}) \end{aligned} \tag{6.11}$$

$$h_k(x_{k|k-1}) \approx h_k(\hat{x}_{k|k-1}) + h_k^{'}(\hat{x}_{k|k-1})(x_{k|k-1} - \hat{x}_{k|k-1}) \tag{6.12}$$

$$\frac{\partial f_{k-1}(\hat{x}_{k-1|k-1})}{\partial \hat{x}_{k-1|k-1}} \tag{6.13}$$

with the Jacobian matrices are $f_{k-1}^{'}(\hat{x}_{k-1|k-1})$ and $h_k^{'}(\hat{x}_{k|k-1})$. The derivation of linearizations for every model and presented the prediction and update parts can be expressed as in the following two sub-sections.

6.3.2.1 Prediction Steps

After the linearizations are derived to every model, the before predicted tracking mean and covariance are now expressed as:

$$\hat{x}_{k|k-1} = f_{k-1}(\hat{x}_{k-1|k-1}) \tag{6.14}$$

$$P_{k|k-1} = f_{k-1}^{'}(\hat{x}_{k-1|k-1})P_{k-1|k-1}f_{k-1}^{'}(\hat{x}_{k-1|k-1})^T + Q_{k-1} \tag{6.15}$$

6.3.2.2 Update Steps

After prediction boxes are calculated and compared with the real boxes, we can derive for every individual tracking the aposteriori motion noise v_k, expected measurement covariance S_k, Kalman Gain K_k, tracking mean $\hat{x}_{k|k}$ and covariance $P_{k-1|k-1}$.

$$v_k = y_k - h_k(\hat{x}_{k|k-1}) \tag{6.16}$$

$$S_k = h_k^{'}(\hat{x}_{k|k-1})P_{k|k-1}h_k^{'}(\hat{x}_{k|k-1})^T + R_k \tag{6.17}$$

$$K_k = P_{k|k-1}h_k^{'}(\hat{x}_{k|k-1})^T S_k^{-1} \tag{6.18}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k v_k \tag{6.19}$$

$$P_{k-1|k-1} = P_{k|k-1} - K_k S_k K_k^T \tag{6.20}$$

6.4 Proposed Tracking Approach of Multiple 3D Anchor Boxes

The extension of the Bayesian reasoning to multi 3D boxes is defined on RFS and requires many tools that are defined and developed by finite-set statistics (FISST) [79].

Every part in any given state RFS $x_{k-1} \in X_{k-1}$ can continue existing at a time step k with a defined probability $p_{S,k}$, otherwise it will disappear with the probability of $1 - p_{S,k}$. The parts that will continue to figure while transitioning to another state x_k will follow a transition function denoted by $f_{k|k-1}(x_k|x_{k-1})$. Consequently, the state RFS X_k will be predicted as $\{x_k\}$ among the elements $x_{k-1} \in X_{k-1}$ that continue existing, otherwise it is $\emptyset$. This transition $S_{k|k-1}(x_{k-1})$ is modeled in [17].

In addition to the 3D objects still detected from the previous scan, there will be new 3D detected objects that are created at k and defined as Γ_k. The multiple 3D target tracking with RFS X_k is described as:

$$X_k = \left[\bigcup_{x_{k-1} \in X_{k-1}} S_{k|k-1}(x_{k-1}) \right] \cup \Gamma_k \tag{6.21}$$

The measured values of RFS Z_k consist of the clutter's and target's actual observations. For a target $x_k \in X_k$, the probability of missing the detection of the target is $1 - p_{D,k}$ and the probability of successfully detecting the target is $p_{D,k}$. If the target is detected then the likelihood function of the measurement z_k is defined as $g_k(z_k|x_k)$ and equals to $\emptyset$ in the case of the target is not detected.

The transition function is $\Theta_k(x_k)$, the clutter observations is defined as K_k and the measurement RFS Z_k is the union those sets.

$$Z_k = \left[\bigcup_{x_k \in X_k} \Theta_k(x_k) \right] \cup K_k \tag{6.22}$$

The optimal Bayesian multi-target filter have a similar prediction and update steps of the single-target Bayesian recursion.

$$p_{k|k-1}(X_k|Z_{1:k-1}) = \int f_{k|k-1}(X_k|X_{k-1}) \\ p_{k-1}(X_{k-1}|Z_{1:k-1})\mu_s(dX_{k-1}) \tag{6.23}$$

$$p_k(X_k|Z_{1:k}) = \frac{g_k(Z_k|X_k)p_{k|k-1}(X_k|Z_{1:k-1})}{\int g_k(Z_k|X_k)p_{k|k-1}(X_k|Z_{1:k-1})\mu_s(dX_{k-1})} \tag{6.24}$$

Where μ_s is the reference measure for all the collection of subsets of X_k [17].

6.5 Captures of Tracking Results

Fig. 6.4, Fig. 6.5 and Fig. 6.6 represent the tracked screen shots that are recorded after 5 seconds of the intial frames in Fig. 6.1, Fig. 6.2 and Fig. 6.3, respectively. The 3D boxes in red represent tracked boxes from initial frames and the 3D boxes in grey represent the wrong prediction of the EKF.

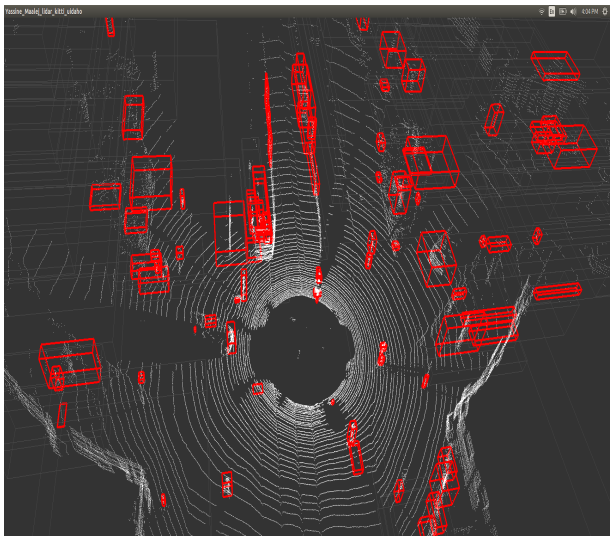

Figure 6.4. Tracking 2011_09_26 drive_0005 #1, EKF Predicted Box in Grey and Actual Box in Red.

The first result to report from these Figures is that there is a lot of objects that have been successfully tracked and the Kalman gain recovered from earlier tracked scenes depending on the nonlinear motion of boxes did successfully estimate the actual position.

In addition, some predicted states of the boxes have not coincided with the actual ones. Thus, we need to feed the area containing the object again to the 3D CNN and consequently may loose track.

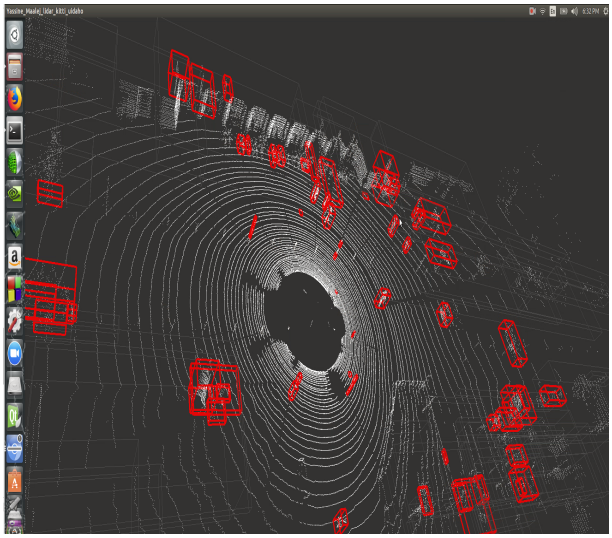

Figure 6.5. Tracking 2011_09_28 drive_0016 #1, EKF Predicted Box in Grey and Actual Box in Red.

In order to accurately discuss the numerical results of the accuracy of the tracking for the three driving sequences, we have to remove the ground and non valid sub-point cloud that is not inferred from the CNN. Then, we discuss the overall number and distances between the predicted and actual tracked boxes for every scan in every driving sequence.

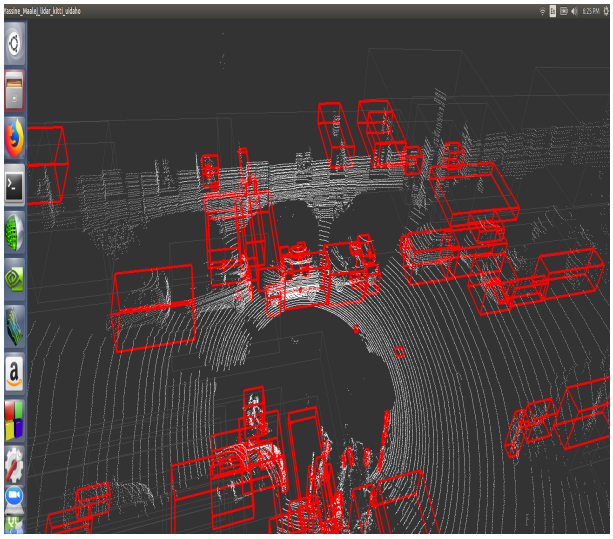

Figure 6.6. Tracking 2011_09_26 drive_0106 #1, EKF Predicted Box in Grey and Actual Box in Red.

In summary, we have developed in this Chapter a 3D LIDAR-based solution for autonomous vehicles, capable of tracking recognized objects from LIDAR. The proposed scheme employs the extended nonlinear tracking models in EKFs for the prediction and the update of the 3D detected box surrounding cars and persons. We presented a single object tracking formulation and extend it to include the Bayesian multi-target filter that runs recursively for each 3D box.

Chapter 7: Conclusion and Future Research Directions

The lack of a widely used and standardized wireless communication technology in VANETs is one the most challenging problems blocking the expansion of VCs. Existing state-of-the-art research on communications would satisfy the safety aspects required for V2V safety systems, but does not meet the need of autonomous driving connectivity. Moreover, many challenges and issues related to the vehicular cloud and vehicular network management with fast changing topologies have been presented. Key challenges to enable VC and AD systems to be coherent and reinforce the desired functionalites related to driving commands or the driving experience are also presented. We studied the various complex components that need to be built, integrated and fit together to guarantee an autonomous driving vehicle. The sensing, perception, decision and actuation layers come with many challenges to be deployed, tested and then put into real-world systems.

We present a novel Advanced Activity-Aware (AAA) scheme to enhance Multi-Channel Operations based on IEEE 1609.4 standard in MAC Protocol implemented in Wireless Access in Vehicular Environments (WAVE). The developed AAA scheme relies on the awareness of the vehicular safety load. It aims at dynamically locating an optimal setup to switch between Service Channel Interval (SCHI) and Control Channel Interval (CCHI) by decreasing every inactivity in the network. Our scheme is implemented using NS3 and maintains the default Synchronization Interval (SI), as defined by the standard in Vehicular Ad hoc Networks (VANETs). We evaluate the performance of our proposed scheme through real-time simulation of VC load and VANET setups using NS3. We evaluate a sequential and a parallel CUDA-accelerated Markov Decision Process (MDP) based scheme and a fast greedy heuristics algorithm to optimize the problem of vehicular task

placement with both IEEE 1609.4 and opportunistically available V2I of the AAA scheme. We derive the system reward of Vehicular Cloud Computing (VCC) by considering the overall utilization of virtualized resources of the distributed Vehicular Clouds (VCs) as well as the optimality of the solution of placing the vehicular Bag-of-Tasks (BOTs).

We propose a scheme to create joint pixel-to-point-cloud and pixel-to-V2V correspondences of objects in frames from the KITTI Vision Benchmark Suite. Using a semi-supervised manifold alignment, to achieve camera-LIDAR and camera-V2V mapping of their recognized objects. We present the alignment accuracy results over 2 different driving sequences and illustrate the additional acquired knowledge of objects from the various input modalities. We also study the effect of the number of neighbors employed in the alignment process on the alignment accuracy. With proper choice of parameters, the testing of our proposed scheme over two entire driving sequences exhibits 100% accuracy in the majority of cases, 74-92% and 50-72% average alignment accuracy for vehicles and pedestrians, and up to 150% additional object recognition of the testing vehicle's surrounding.

5G might be the complete communications technology that will ease transforming VCs and automation systems from a vision into a widely affordable reality. Consequently, the challenges facing 5G connections float on the surface as a disruptive factor for VCs exploitation. Similarly, knowing that self-driving vehicles are making Big Data even bigger, high speed connectivity like 5G would be the solution to tap into faster networks and offload to the cloud the several hundreds of mega bits created per second.

The security issues in both the communications and the automation systems would be definitely areas of study for the complete standardized systems when developed. Additionally, in the future VCs are supposed to manage the microdatacenters mounted in vehicles, communications base stations and the traditional VC architectures in the hope that they will revolutionize various systems to be taken advantage of in modern vehicles. The existing low level vehicles' automation has not led to fully self-driving complete sys-

tems yet. This is due to the many open unsolved problems or existing solutions that need more refinements and improvements. In other words, it is obvious that owning the most intelligent accurate HD maps will certainly own the future of self-driving vehicles, independently of the perception, planning and command technologies. This HD (virtual-world) map allows vehicles to accurately map themselves on well-known empty streets and add detected dynamic objects later.

Moreover, new sophisticated and cheaper sensors should be developed to make the business model of self-driving vhicles more attractive. For example, the spinning Velodyne LIDAR offers great 360 degrees visibility and accurate depth information, but comes at a high price tag and huge amount of data creation making processing complex. Recently announced solid state LIDARs by Quanergy [40] and Velodyne [103] are expected to hit the market at a few hundred US dollars in comparison to the several thousand US dollars for the spinning LIDARs.

Other problems facing Spinning-LIDARs along with a comparison with solid state LIDAR are presented in [113] including durability, precision and stability. Fail-safe real-time software used as components in the autonomous driving systems are far from being completely developed and validated. These pieces of software may include failing functionalities for:

- Objects detection, classification and tracking.
- Sensors fusion and vehicle mapping.
- Deep scene understanding and path planning.
- Control and steering systems.

A sophisticated deployable self-driving solution requires an effective combination of:

- High precision maps to position the car in relation to the surrounding objects in the scene.
- Depth and stereo sensor detection and fusion.

- Deep understanding of the scene with effective tracking.
- Accelerated sensor data processing and inference.
- Precise and quick steering commands system.

Self-driving vehicles are known as the mother of artificial intelligence systems and the one with most promising business models. The emerging idea of VCs is very promising too, aiming to improve the driving experience by using various types of infotainment applications. In addition, it should increase vehicular safety by using beaconing applications and supporting additional over the air data for self-driving systems.

The challenges that are involved in these layers are presented in this Chapter. We also presented an overview of the existing communication technologies, VANET MAC protocols, algorithms for VANET network coalition, reliability of safety and non-safety applications, Vehicular cloud selection based on cost and QoS, all used to bridge reliable communications, VC and AD systems.

References

[1] Full self-driving hardware on all cars. https://www.tesla.com/autopilot. Advanced Sensor Coverage, Processing Power Increased 40x, Enhanced Autopilot, Full Self-Driving Capability.

[2] Global connected car market (technology, connectivity solutions, product and services, applications and geography)-size, share, global trends, company profiles, demand, insights, analysis, research, report, opportunities, segmentation and forecast, 2013. www.marketresearchreports.com/allied-market-research/global-connected-car-market-technology-connectivity-solutions.

[3] Google keeps self-driving cars offline to hinder hackers. https://www.ft.com/content/8eff8fbe-d6f0-11e6-944b-e7eb37a6aa8e?mhq5j=e3.

[4] Here hd live map. https://here.com/en/products-services/products/here-hd-live-map. The most intelligent sensor for autonomous driving.

[5] Multimedia broadcast/multicast service (mbms); architecture and functional description (release 13). *Third Generation Partnership Project (3GPP), Technical Specification Group Services and System Aspects (TSG SA)*, 3GPP TS 23.246 V13.0.0.

[6] Technology: Cloud-to-car mapping system for driverless vehicles. http://www.archer-soft.com/en/blog/technology-cloud-car-mapping-system-driverless-vehicles. Posted May 08, 2017.

[7] Federal communications commission, fcc report and order std. *IEEE Wireless Communication*, pages FCC 99–305, October 1999.

[8] P. Dean A. *Neural Network Perception for Mobile Robot Guidance*. Kluwer Academic Publishers, Norwell, MA, USA, 1993.

[9] K. Abboud, H. Omar, and W. Zhuang. Interworking of dsrc and cellular network technologies for v2x communications: A survey. *IEEE Transactions on Vehicular Technology*, PP(99):1–1, 2016.

[10] Sam Abuelsamid. New cars could be required to 'talk' to each other as soon as 2020. `https://www.forbes.com/sites/samabuelsamid/2016/12/13/nhtsa-finally-issues-draft\-v2v-communications-rule-could-be-mandatory-from-2021/#696da21f7581`. Published December 13, 2016.

[11] Sam Abuelsamid. Toyota and university of michigan to expand world's largest real-world v2x communications test. `www.forbes.com/sites/samabuelsamid/2016/04/13/toyota-and-university-of-michigan-to-expand-worlds-largest-real`.

[12] F. Ahmad, M. Kazim, A. Adnane, and A. Awad. Vehicular cloud networks: Architecture, applications and security issues. In *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, pages 571–576, Dec 2015.

[13] L. Aminizadeh and S. Yousefi. Cost minimization scheduling for deadline constrained applications on vehicular cloud infrastructure. In *Computer and Knowledge Engineering (ICCKE), 2014 4th International eConference on*, pages 358–363, Oct 2014.

[14] G. Araniti, C. Campolo, M. Condoluci, A. Iera, and A. Molinaro. Lte for vehicular networking: a survey. *IEEE Communications Magazine*, 51(5):148–157, May 2013.

[15] M. Asgari and S. Yousefi. Traffic modeling of safety applications in vehicular networks. In *Information and Knowledge Technology (IKT), 2013 5th Conference on*, pages 389–393, May 2013.

[16] A. Asvadi, P. Peixoto, and U. Nunes. Detection and tracking of moving objects using 2.5d motion grids. In *2015 IEEE 18th International Conference on Intelligent Transportation Systems*, pages 788–793, Sept 2015.

[17] Wing-Kin Ma Ba-Ngu Vo. The Gaussian Mixture Probability Hypothesis Density Filter. *IEEE TRANSACTIONS SIGNAL PROCESSING*, 54(11):4091–4104, November 2006.

[18] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *CoRR*, abs/1511.00561, 2015.

[19] Stefano Basagni, Marco Conti, Silvia Giordano, and Ivan Stojmenovic. *The Next Paradigm Shift: From Vehicular Networks to Vehicular Clouds*, pages 645–700. Wiley-IEEE Press, 2013.

[20] D. Benhaddou and A. Al-Fuqaha. *Wireless Sensor and Mobile Ad-Hoc Networks: Vehicular and Space Applications*. Springer New York, 2015.

[21] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D. Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, Xin Zhang, Jake Zhao, and Karol Zieba. End to end learning for self-driving cars. *CoRR*, abs/1604.07316, 2016.

[22] Claus Brenner. Vehicle localization using landmarks obtained by a lidar mobile mapping system. *Int. Arch. Photogramm. Remote Sens*, 38:139–144, 2010.

[23] Anne-Muriel Brouet. With or without a driver, vehicles are able to cooperate. https://actu.epfl.ch/news/with-or-without-a-driver-vehicles-are-able-to-coop/. Poseted in Mediacom.

[24] Zhaowei Cai, Quanfu Fan, Rogério Schmidt Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. *CoRR*, abs/1607.07155, 2016.

[25] J. Calabuig, J. F. Monserrat, D. Gozalvez, and O. Klemp. Safety on the roads: Lte alternatives for sending its messages. *IEEE Vehicular Technology Magazine*, 9(4):61–70, Dec 2014.

[26] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 2722–2730, Dec 2015.

[27] J. Chen, B. Liu, L. Gui, F. Sun, and H. Zhou. Engineering link utilization in cellular offloading oriented vanets. In *2015 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2015.

[28] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-View 3D Object Detection Network for Autonomous Driving. *ArXiv e-prints*, November 2016.

[29] ZJ Chong, Baoxing Qin, Tirthankar Bandyopadhyay, Marcelo H Ang, Emilio Frazzoli, and Daniela Rus. Synthetic 2d lidar for precise vehicle localization in 3d urban environment. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1554–1559. IEEE, 2013.

[30] Ryan Robinson Gina Pingitore Craig A. Giffi, Joe Vitale. The race to autonomous driving. https://dupress.deloitte.com/dup-us-en/deloitte-review/

issue-20/winning-consumer-trust-future-of-automotive-technology.html. Published January 23, 2017.

[31] M. CÃ¡p, P. NovÃ¡k, M. SeleckÃ½, J. Faigl, and J. Vokffnek. Asynchronous decentralized prioritized planning for coordination in multi-robot system. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3822–3829, Nov 2013.

[32] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard. Motion-based detection and tracking in 3d lidar scans. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4508–4513, May 2016.

[33] Hoang T. Dinh, Chonho Lee, Dusit Niyato, and Ping Wang. A survey of mobile cloud computing: architecture, applications, and approaches. *Wireless Communications and Mobile Computing*, 13(18):1587–1611, 2013.

[34] R. DomÃnguez, E. Onieva, J. Alonso, J. Villagra, and C. GonzÃ¡lez. Lidar based perception solution for autonomous vehicles. In *2011 11th International Conference on Intelligent Systems Design and Applications*, pages 790–795, Nov 2011.

[35] Ziqian Dong, Ning Liu, and Roberto Rojas-Cessa. Greedy scheduling of tasks with time constraints for energy-efficient cloud-computing data centers. *Journal of Cloud Computing*, 4(1):1–14, 2015.

[36] NVIDIAÂ® DriveWorks. Nvidia drive developer program. https://developer.nvidia.com/driveworks. DriveWorks SDK Overview.

[37] D.R.Stephens, T.J.Timcho, R.A.Klein, and J.L.Schroeder. Vehicle-to-infrastructure (v2i) safety applications, concept of operations document. *Intelligent Transportation Systems, National Highway Traffic Safety Administration*, 8 March 2013.

[38] R. Duarte, R. Sendag, and F. J. Vetter. On the performance and energy-efficiency of multi-core simd cpus and cuda-enabled gpus. In *Workload Characterization (IISWC), 2013 IEEE International Symposium on*, pages 174–184, Sept 2013.

[39] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE robotics & automation magazine*, 13(2):99–110, 2006.

[40] Louay Eldada. The world's first affordable solid state lidar sensor. `http://quanergy.com/s3/`. Published in 2016 by Quanergy Systems, Inc.

[41] M. Engelcke, D. Rao, D. Zeng Wang, C. Hay Tong, and I. Posner. Vote3Deep: Fast Object Detection in 3D Point Clouds Using Efficient Convolutional Neural Networks. *ArXiv e-prints*, September 2016.

[42] Michael Essery. Self-driving cars an $87 billion opportunity in 2030, though none reach full autonomy. `http://www.luxresearchinc.com/news-and-events/press-releases/read/self-driving-cars-87-billion-opportunity-2030-though-none-reach`.

[43] Darrell Etherington. Bmw and mobileye to crowdsource real-time data. `techcrunch.com/2017/02/21/bmw-and-mobileye-to-crowdsource-real-time-data-for-self-driving`. February 21, 2017.

[44] N. Fanani, M. Ochs, H. Bradler, and R. Mester. Keypoint trajectory estimation using propagation based tracking. In *2016 IEEE Intelligent Vehicles Symposium (IV)*, pages 933–939, June 2016.

[45] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, Aug 2013.

[46] Clement Farabet, Camille Couprie, Laurent Najman, and Yann Lecun. Scene parsing with multiscale feature learning, purity trees, and optimal covers. In John Langford and Joelle Pineau, editors, *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, pages 575–582, New York, NY, USA, 2012. ACM.

[47] F.A.Zaid, F.Bai, S.Bai, C.Basnayake, B.Bellur, S.Brovold, G.Brown, L.Caminiti, D.Cunningham, H.Elzein, K.Hong, J.Ivan, D.Jiang, J.Kenney, H.Krishnan, J.Lovell, M.Maile, D.Masselink, E.McGlohon, P.Mudalige, Z.Popovic, V.Rai, J.Stinnett, L.Tellis, K.Tirey, and S.VanSickle. Vehicle safety communications applications (vsc-a) final report. *Intelligent Transportation Systems, National Highway Traffic Safety Administration*, September 2011.

[48] A Geiger, P Lenz, C Stiller, and R Urtasun. Vision meets robotics: The kitti dataset. *Int. J. Rob. Res.*, 32(11):1231–1237, September 2013.

[49] M. Gerla. Vehicular cloud computing. In *Ad Hoc Networking Workshop (Med-Hoc-Net), 2012 The 11th Annual Mediterranean*, pages 152–155, June 2012.

[50] P. Ghazizadeh, S. Olariu, A. G. Zadeh, and S. El-Tawab. Towards fault-tolerant job assignment in vehicular cloud. In *Services Computing (SCC), 2015 IEEE International Conference on*, pages 17–24, June 2015.

[51] S. Gould, R. Fulton, and D. Koller. Decomposing a scene into geometric and semantically consistent regions. In *2009 IEEE 12th International Conference on Computer Vision*, pages 1–8, Sept 2009.

[52] Ryan Grenoble. Self-driving uber blows through red light on first day in san francisco. `http://www.huffingtonpost.com/entry/self-driving-uber-san-fran-red-light-video_us_5851c9c5e4b0732b82fedb01`. Published December 14, 2016.

[53] K. A. Hafeez, A. Anpalagan, and L. Zhao. Optimizing the control channel interval of the dsrc for vehicular safety applications. *IEEE Transactions on Vehicular Technology*, 65(5):3377–3388, May 2016.

[54] K. A. Hafeez, L. Zhao, Z. Liao, and B. N. W. Ma. A new broadcast protocol for vehicular ad hoc networks safety applications. In *Global Telecommunications Conference (GLOBECOM 2010), 2010 IEEE*, pages 1–5, Dec 2010.

[55] Xuming He and Richard S. Zemel. Learning hybrid models for image annotation with partially labeled data. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, *Advances in Neural Information Processing Systems 21*, pages 625–632. 2009.

[56] Vishakh Hegde and Reza Zadeh. Fusionnet: 3d object classification using multiple data representations. *CoRR*, abs/1607.05695, 2016.

[57] HERE. Power meets possibilities. `https://here.com/en`. Companies rely on HERE for innovative and intelligent solutions.

[58] X. Hu, X. Li, E. C. H. Ngai, J. Zhao, V. C. M. Leung, and P. Nasiopoulos. Health drive: Mobile healthcare onboard vehicles to promote safe driving. In *System Sciences (HICSS), 2015 48th Hawaii International Conference on*, pages 3074–3083, Jan 2015.

[59] Dana Hull. The tesla advantage: 1.3 billion miles of data. `https://www.bloomberg.com/news/articles/2016-12-20/the-tesla-advantage-1-3-billion-miles-of-data`. Posted December 20, 2016.

[60] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue,

et al. An empirical evaluation of deep learning on highway driving. *arXiv preprint arXiv:1504.01716*, 2015.

[61] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. Design of 5.9 ghz dsrc-based vehicular safety communication. *IEEE Wireless Communications*, 13(5):36–43, October 2006.

[62] D. Jiang, V. Taliwal, A. Meier, W. Holfelder, and R. Herrtwich. Design of 5.9 ghz dsrc-based vehicular safety communication. *IEEE Wireless Communications*, 13(5):36–43, October 2006.

[63] M. P. Kumar and D. Koller. Efficiently selecting regions for scene understanding. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3217–3224, June 2010.

[64] Swarun Kumar, Shyamnath Gollakota, and Dina Katabi. A cloud-assisted design for autonomous driving. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 41–46. ACM, 2012.

[65] Abhijit Kundu, Vibhav Vineet, and Vladlen Koltun. Feature space optimization for semantic video segmentation. In *CVPR*, 2016.

[66] L. LadickÃ½, C. Russell, P. Kohli, and P. H. S. Torr. Associative hierarchical crfs for object class image segmentation. In *2009 IEEE 12th International Conference on Computer Vision*, pages 739–746, Sept 2009.

[67] R. Lasowski, C. Scheuermann, F. Gschwandtner, and C. Linnhoff-Popien. Evaluation of adjacent channel interference in single radio vehicular ad-hoc networks. In *2011 IEEE Consumer Communications and Networking Conference (CCNC)*, pages 267–271, Jan 2011.

[68] E. Lee, E. K. Lee, M. Gerla, and S. Y. Oh. Vehicular cloud networking: architecture and design principles. *IEEE Communications Magazine*, 52(2):148–155, February 2014.

[69] Jesse Levinson, Michael Montemerlo, and Sebastian Thrun. Map-based precision vehicle localization in urban environments. In *Robotics: Science and Systems*, volume 4, page 1. Citeseer, 2007.

[70] F. Li and Y. Wang. Routing in vehicular ad hoc networks: A survey. *IEEE Vehicular Technology Magazine*, 2(2):12–22, June 2007.

[71] M. Lott, M. Meincke, and R. Halfmann. A new approach to exploit multiple frequencies in dsrc. In *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, volume 3, pages 1539–1543 Vol.3, May 2004.

[72] G. Karagiannis M. van Eenennaam, A. van de Venis. Impact of ieee 1609.4 channel switching on the ieee 802.11p beaconing performance. In *Wireless Days (WD)*, pages 1–8, Dublin, 21-23 November 2012.

[73] X. Ma, J. Zhang, and T. Wu. Reliability analysis of one-hop safety-critical broadcast services in vanets. *IEEE Transactions on Vehicular Technology*, 60(8):3933–3946, Oct 2011.

[74] Y. Maalej, A. Abderrahim, M. Guizani, and B. Hamdaoui. Cuda-accelerated task scheduling in vehicular clouds with opportunistically available v2i. In *2017 IEEE International Conference on Communications (ICC)*, pages 1–6, May 2017.

[75] Y. Maalej, A. Abderrahim, M. Guizani, B. Hamdaoui, and E. Balti. Advanced activity-aware multi-channel operations1609.4 in vanets for vehicular clouds. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2016.

[76] Y. Maalej, A. Abderrahim, M. Guizani, B. Hamdaoui, and E. Balti. Advanced activity-aware multi-channel operations1609.4 in vanets for vehicular clouds. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2016.

[77] Y. Maalej, A. Abderrahim, M. Guizani, B. Hamdaoui, and E. Balti. Advanced activity-aware multi-channel operations1609.4 in vanets for vehicular clouds. In *2016 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6, Dec 2016.

[78] Y. Maalej, S. Sorour, A. Abdel-Rahim, and M. Guizani. Vanets meet autonomous vehicles: A multimodal 3d environment learning approach. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, Dec 2017.

[79] Ronald PS Mahler. Multitarget bayes filtering via first-order multitarget moments. *Aerospace and Electronic Systems, IEEE Transactions on*, 39(4):1152–1178, 2003.

[80] K. Majeed, S. Sorour, T. Y. Al-Naffouri, and S. Valaee. Indoor localization and radio map estimation using unsupervised manifold alignment with geometry perturbation. *IEEE Transactions on Mobile Computing*, 15(11):2794–2808, Nov 2016.

[81] T. Mangel. *Inter-Vehicle Communication at Intersections : An Evaluation of Ad-Hoc and Cellular Communication*. KIT Scientific Publ., 2014.

[82] T. Mangel, T. Kosch, and H. Hartenstein. A comparison of umts and lte for vehicular safety communication at intersections. In *Vehicular Networking Conference (VNC), 2010 IEEE*, pages 293–300, Dec 2010.

[83] AARIAN MARSHALL. With intelâs chips, google could at last deliver self-driving cars. `https://www.wired.com/story/waymo-and-intel-self-driving/`. Published September 19, 2017.

[84] P. Matzakos, J. HÃ□rri, B. Villeforceix, and C. Bonnet. An ipv6 architecture for cloud-to-vehicle smart mobility services over heterogeneous vehicular networks. In *2014 International Conference on Connected Vehicles and Expo (ICCVE)*, pages 767–772, Nov 2014.

[85] K. Mekhnacha, Y. Mao, D. Raulo, and C. Laugier. The x201c;fast clustering-tracking x201d; algorithm in the bayesian occupancy filter framework. In *2008 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems*, pages 238–245, Aug 2008.

[86] M. Menze and A. Geiger. Object scene flow for autonomous vehicles. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3061–3070, June 2015.

[87] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicles. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[88] M.Shulman and R.Deering. Vehicle safety communications in the united states. *United States Department of Transportation*, pages Rep. 07–0010, 2006.

[89] Daniel Munoz, J Andrew Bagnell, and Martial Hebert. Stacked hierarchical labeling. In *European Conference on Computer Vision*, pages 57–70. Springer, 2010.

[90] R. Mur-Artal, J. M. M. Montiel, and J. D. TardÃ^3*s*. $Orb - slam : A versatile and accurate monocular slam system. IEEE Transactions on Robotics, 31(5) : 1147 - -1163, Oct 2015.$

[91] H. Nakata, T. Inoue, M. Itami, and K. Itoh. A study of inter vehicle communication scheme allocating pn codes to the location on the road. In *Intelligent Transportation Systems, 2003. Proceedings. 2003 IEEE*, volume 2, pages 1527–1532 vol.2, Oct 2003.

[92] Vinod Namboodiri, Manish Agarwal, and Lixin Gao. A study on the feasibility of mobile gateways for vehicular ad-hoc networks. In *Proceedings of the 1st ACM International Workshop on Vehicular Ad Hoc Networks*, VANET '04, pages 66–75, New York, NY, USA, 2004. ACM.

[93] David Nistér, Oleg Naroditsky, and James Bergen. Visual odometry. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 1, pages I–I. Ieee, 2004.

[94] U. Ozguner, T. Acarman, and K. Redmill. *Autonomous Ground Vehicles*. Artech House ITS series. Artech House, 2011.

[95] A.S.K. Pathan. *Security of Self-Organizing Networks: MANET, WSN, WMN, VANET*. CRC Press, 2016.

[96] A. K. Paul, S. K. Addya, B. Sahoo, and A. K. Turuk. Application of greedy algorithms to virtual machine distribution across data centers. In *2014 Annual IEEE India Conference (INDICON)*, pages 1–6, Dec 2014.

[97] Y. Pei, F. Huang, F. Shi, and H. Zha. Unsupervised image matching based on manifold alignment. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(8):1658–1664, Aug 2012.

[98] T. Pire, T. Fischer, J. Civera, P. De CristÃ$^3 foris, and J. J. Berlles. Stereo parallel tracking and map
$-1378, Sept 2015$.

[99] Dean A. Pomerleau. Advances in neural information processing systems 1. chapter ALVINN: An Autonomous Land Vehicle in a Neural Network, pages 305–313. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1989.

[100] D. Jiang Q. Chen and L. Delgrossi. Ieee 1609.4 dsrc multi-channel operations and its implications on vehicle safety communications. In *Vehicular Technology Conference*, pages 1 – 8, Tokyo, Japan, 28-30 Oct. 2009.

[101] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[102] T. K. Refaat, B. Kantarci, and H. T. Mouftah. Dynamic virtual machine migration in a vehicular cloud. In *2014 IEEE Symposium on Computers and Communications (ISCC)*, volume Workshops, pages 1–6, June 2014.

[103] Philip E. Ross. Velodyne announces a solid-state lidar. `http://spectrum.ieee.org/cars-that-think/transportation/sensors/velodyne-announces-a-solidstate-lidar`. Posted 19 Apr 2017.

[104] S. Ruiz and B. HernÃ¡ndez. A parallel solver for markov decision process in crowd simulations. In *2015 Fourteenth Mexican International Conference on Artificial Intelligence (MICAI)*, pages 107–116, Oct 2015.

[105] S. Saini, H. Jin, R. Hood, D. Barker, P. Mehrotra, and R. Biswas. The impact of hyper-threading on processor resource utilization in production applications. In *2011 18th International Conference on High Performance Computing*, pages 1–10, Dec 2011.

[106] M. A. Salahuddin, A. Al-Fuqaha, M. Guizani, and S. Cherkaoui. Rsu cloud and its resource management in support of enhanced vehicular applications. In *2014 IEEE Globecom Workshops (GC Wkshps)*, pages 127–132, Dec 2014.

[107] J. Sanders and E. Kandrot. *CUDA by Example: An Introduction to General-Purpose GPU Programming, Portable Documents*. Pearson Education, 2010.

[108] Lawrence K Saul and Sam T Roweis. An introduction to locally linear embedding. *unpublished. Available at: http://www. cs. toronto. edu/˜ roweis/lle/publications. html*, 2000.

[109] R. K. Schmidt, R. Lasowski, T. Leinmiiller, C. Linnhoff-Popien, and G. SchÃ¤fer. An approach for selective beacon forwarding to improve cooperative awareness. In *Vehicular Networking Conference (VNC), 2010 IEEE*, pages 182–188, Dec 2010.

[110] DANNY SHAPIRO. Guardian angel: Why your next car may have an ai co-pilot. `https://blogs.nvidia.com/blog/2017/05/16/ai-co-pilot/`. Posted in May 16, 2017.

[111] Varuna De Silva, Jamie Roche, and Ahmet M. Kondoz. Fusion of lidar and camera sensor data for environment sensing in driverless vehicles. *CoRR*, abs/1710.06230, 2017.

[112] Ioana Sima. Iot and smart cars: Changing the world for the better. `http://www.digitalistmag.com/iot/2016/08/30/iot-smart-connected-cars-will-change-world-04422640`. Published August 30, 2016.

[113] Tom Simonite. Self-driving carsâ spinning-laser problem. `https://www.technologyreview.com/s/603885/autonomous-cars-lidar-sensors/`. Posted March 20, 2017 by MIT Technology Review.

[114] The Tesla Team. A tragic loss. `https://www.tesla.com/blog/tragic-loss`. Published June 30, 2016.

[115] The Tesla Team. A tragic loss. `https://www.tesla.com/blog/tragic-loss`. Published June 30, 2016.

[116] Joseph Tighe and Svetlana Lazebnik. Superparsing: Scalable nonparametric image parsing with superpixels. In *Proceedings of the 11th European Conference on Computer Vision: Part V*, ECCV'10, pages 352–365, Berlin, Heidelberg, 2010. Springer-Verlag.

[117] Masayoshi Tomizuka. 3d object detection based on lidar and camera fusion. `https://deepdrive.berkeley.edu/project/3d-object-detection-based-lidar-and-camera-fusion`.

[118] Steffen Urban and Stefan Hinz. Multicol-slam - A modular real-time multi-camera SLAM system. *CoRR*, abs/1610.07336, 2016.

[119] Christoph Vogel, Konrad Schindler, and Stefan Roth. 3d scene flow estimation with a piecewise rigid scene model. *International Journal of Computer Vision*, 115(1):1–28, 2015.

[120] Yi Wang, A. Ahmed, B. Krishnamachari, and K. Psounis. Ieee 802.11p performance evaluation and protocol enhancement. In *Vehicular Electronics and Safety, 2008. ICVES 2008. IEEE International Conference on*, pages 317–322, Sept 2008.

[121] B. C. Ward. Relaxing resource-sharing constraints for improved hardware management and schedulability. In *Real-Time Systems Symposium, 2015 IEEE*, pages 153–164, Dec 2015.

[122] Waymo. Waymo safety report on the road to fully self-driving. `https://storage.googleapis.com/sdc-prod/v1/safety-report/waymo-safety-report-2017-10.pdf`. Waymoâs mission is to bring self-driving technology to the world, making it safe and easy for people and things to move around.

[123] Md Whaiduzzaman, Mehdi Sookhak, Abdullah Gani, and Rajkumar Buyya. A survey on vehicular cloud computing. *J. Netw. Comput. Appl.*, 40:325–344, April 2014.

[124] Ryan W Wolcott and Ryan M Eustice. Visual localization within lidar maps for automated urban driving. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 176–183. IEEE, 2014.

[125] K. Xu, K. C. Wang, R. Amin, J. Martin, and R. Izard. A fast cloud-based network selection scheme using coalition formation games in vehicular networks. *IEEE Transactions on Vehicular Technology*, 64(11):5327–5339, Nov 2015.

[126] R. Yu, X. Huang, J. Kang, J. Ding, S. Maharjan, S. Gjessing, and Y. Zhang. Cooperative resource management in cloud-enabled vehicular networks. *IEEE Transactions on Industrial Electronics*, 62(12):7938–7951, Dec 2015.

[127] R. Yu, Y. Zhang, S. Gjessing, W. Xia, and K. Yang. Toward cloud-based vehicular networks with efficient resource management. *IEEE Network*, 27(5):48–55, September 2013.

[128] Ji Zhang, Michael Kaess, and Sanjiv Singh. Real-time depth enhanced monocular odometry. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 4973–4980. IEEE, 2014.

[129] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2. Citeseer, 2014.

[130] Ji Zhang and Sanjiv Singh. Visual-lidar odometry and mapping: Low-drift, robust, and fast. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, pages 2174–2181. IEEE, 2015.

[131] Lin Zhang, Yu Liu, Zi Wang, Jinjie Guo, Yiding Huo, Yiqun Yao, Chang Hu, and Yuting Sun. In *Communication Technology (ICCT), 2011.*

[132] R. Zhang, X. Cheng, L. Yang, X. Shen, and B. Jiao. A novel centralized tdma-based scheduling protocol for vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):411–416, Feb 2015.

[133] K. Zheng, H. Meng, P. Chatzimisios, L. Lei, and X. Shen. An smdp-based resource allocation in vehicular cloud computing systems. *IEEE Transactions on Industrial Electronics*, 62(12):7920–7928, Dec 2015.