UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Lidar-based scene understanding for autonomous driving using deep learning

Victor Vaquero Gomez

UNIVERSITAT POLITÈCNICA DE CATALUNYA

Doctoral Programme

AUTOMATIC CONTROL, ROBOTICS AND COMPUTER VISION

Ph.D. Thesis

LIDAR-BASED SCENE UNDERSTANDING FOR AUTONOMOUS DRIVING USING DEEP LEARNING

Victor Vaquero Gomez

Advisors:
Francesc Moreno-Noguer and Alberto Sanfeliu

Barcelona, January 2020

Lidar-Based Scene Understanding for Autonomous Driving Using Deep Learning

A thesis submitted to the Universitat Politècnica de Catalunya to obtain the degree of Doctor of Philosophy

Doctoral programme:
Automatic Control, Robotics and Computer Vision

This thesis was completed at:
Institut de Robòtica i Informàtica Industrial, CSIC-UPC

Thesis advisors:
Francesc Moreno-Noguer and Alberto Sanfeliu

A mi familia y amigos

To my family and friends

If you are the smartest person in the room,
you are in the wrong room.

- Anonymous

LIDAR-BASED SCENE UNDERSTANDING
FOR AUTONOMOUS DRIVING USING DEEP LEARNING

Victor Vaquero Gomez

Abstract

With over 1.35 million fatalities related to traffic accidents worldwide, autonomous driving was foreseen at the beginning of this century as a feasible solution to improve security in our roads. Nevertheless, it is meant to disrupt our transportation paradigm, allowing to reduce congestion, pollution, and costs, while increasing the accessibility, efficiency, and reliability of the transportation for both people and goods. Although some advances have gradually been transferred into commercial vehicles in the way of Advanced Driving Assistance Systems (ADAS) such as adaptive cruise control, blind spot detection or automatic parking, however, the technology is far from mature. A full understanding of the scene is necessary in order to allow the vehicles to be aware of their surroundings and the existing elements on the scene, as well as their respective motions, intentions and interactions.

In this PhD dissertation, we explore new approaches for understanding driving scenes from 3D LiDAR point clouds by using Deep Learning methods. To this end, in Part I we analyze the scene from a static perspective using independent frames to detect the neighboring vehicles. Next, in Part II we develop new ways for understanding the dynamics of the scene. Finally, in Part III we apply all the developed methods to accomplish higher level challenges such as segmenting moving obstacles while obtaining their rigid motion vector over the ground.

More specifically, in Chapter 2 we develop a 3D vehicle detection pipeline based on a multi-branch deep-learning architecture and propose a Front (FR-V) and a Bird's Eye view (BE-V) as 2D representations of the 3D point cloud to serve as input for training our models. Later on, in Chapter 3 we apply and further test this method on two real uses-cases, for pre-filtering moving obstacles while creating maps to better localize ourselves on subsequent days, as well as for vehicle tracking. From the dynamic perspective, in Chapter 4 we learn from the 3D point cloud a novel dynamic feature that resembles optical flow from RGB images. For that, we develop a new approach to leverage RGB optical flow as pseudo ground truth for training purposes but allowing the use of only 3D LiDAR data at inference time. Additionally, in Chapter 5 we explore the benefits of combining classification and regression learning problems to face the optical flow estimation task in a joint coarse-and-fine manner. Lastly, in Chapter 6 we gather the previous methods and demonstrate that with these independent tasks we can guide the learning of challenging higher-level problems such as segmentation and motion estimation of moving vehicles from our own moving perspective.

Keywords: Scene Understanding, Point Clouds, Deep Learning, Vehicle Detection, Optical Flow.

Resumen

Con más de 1,35 millones de muertes por accidentes de tráfico en el mundo, a principios de siglo se predijo que la conducción autónoma sería una solución viable para mejorar la seguridad en nuestras carreteras. Además la conducción autónoma está destinada a cambiar nuestros paradigmas de transporte, permitiendo reducir la congestión del tráfico, la contaminatión y el coste, a la vez que aumentando la accesibilidad, la eficiencia y confiabilidad del transporte tanto de personas como de mercancías. Aunque algunos avances, como el control de crucero adaptativo, la detección de puntos ciegos o el estacionamiento automático, se han transferido gradualmente a vehículos comerciales en la forma de los Sistemas Avanzados de Asistencia a la Conducción (ADAS), la tecnología aún no ha alcanzado el suficiente grado de madurez. Se necesita una comprensión completa de la escena para que los vehículos puedan entender el entorno, detectando los elementos presentes, así como su movimiento, intenciones e interacciones.

En la presente tesis doctoral, exploramos nuevos enfoques para comprender escenarios de conducción utilizando nubes de puntos en 3D capturadas con sensores LiDAR, para lo cual empleamos métodos de aprendizaje profundo. Con este fin, en la Parte I analizamos la escena desde una perspectiva estática para detectar vehículos. A continuación, en la Parte II, desarrollamos nuevas formas de entender las dinámicas del entorno. Finalmente, en la Parte III aplicamos los métodos previamente desarrollados para lograr desafíos de nivel superior, como segmentar obstáculos dinámicos a la vez que estimamos su vector de movimiento sobre el suelo.

Específicamente, en el Capítulo 2 detectamos vehículos en 3D creando una arquitectura de aprendizaje profundo de dos ramas y proponemos una vista frontal (FR-V) y una vista de pájaro (BE-V) como representaciones 2D de la nube de puntos 3D que sirven como entrada para entrenar nuestros modelos. Más adelante, en el Capítulo 3 aplicamos y probamos aún más este método en dos casos de uso reales, tanto para filtrar obstáculos en movimiento previamente a la creación de mapas sobre los que poder localizarnos mejor en los días posteriores, como para el seguimiento de vehículos. Desde la perspectiva dinámica, en el Capítulo 4 aprendemos de la nube de puntos en 3D una característica dinámica novedosa que se asemeja al flujo óptico sobre imágenes RGB. Para ello, desarrollamos un nuevo enfoque que aprovecha el flujo óptico RGB como pseudo muestras reales para entrenamiento, usando solo information 3D durante la inferencia. Además, en el Capítulo 5 exploramos los beneficios de combinar los aprendizajes de problemas de clasificación y regresión para la tarea de estimación de flujo óptico de manera conjunta. Por último, en el Capítulo 6 reunimos los métodos anteriores y demostramos que con estas tareas independientes podemos guiar el aprendizaje de problemas de más alto nivel, como la segmentación y estimación del movimiento de vehículos desde nuestra propia perspectiva.

Resum

Amb més d'1,35 milions de morts per accidents de trànsit al món, a principis de segle es va predir que la conducció autònoma es convertiria en una solució viable per millorar la seguretat a les nostres carreteres. D'altra banda, la conducció autònoma està destinada a canviar els paradigmes del transport, fent possible així reduir la densitat del trànsit, la contaminació i el cost, alhora que augmentant l'accessibilitat, l'eficiència i la confiança del transport tant de persones com de mercaderies. Encara que alguns avenços, com el control de creuer adaptatiu, la detecció de punts cecs o l'estacionament automàtic, s'han transferit gradualment a vehicles comercials en forma de Sistemes Avançats d'Assistència a la Conducció (ADAS), la tecnologia encara no ha arribat a aconseguir el grau suficient de maduresa. És necessària, doncs, una total comprensió de l'escena de manera que els vehicles puguin entendre l'entorn, detectant els elements presents, així com el seu moviment, intencions i interaccions.

A la present tesi doctoral, explorem nous enfocaments per tal de comprendre les diferents escenes de conducció utilitzant núvols de punts en 3D capturats amb sensors LiDAR, mitjançant l'ús de mètodes d'aprenentatge profund. Amb aquest objectiu, a la Part I analitzem l'escena des d'una perspectiva estàtica per a detectar vehicles. A continuació, a la Part II, desenvolupem noves formes d'entendre les dinàmiques de l'entorn. Finalment, a la Part III apliquem els mètodes prèviament desenvolupats per a aconseguir desafiaments d'un nivell superior, com, per exemple, segmentar obstacles dinàmics al mateix temps que estimem el seu vector de moviment respecte al terra.

Concretament, al Capítol 2 detectem vehicles en 3D creant una arquitectura d'aprenentatge profund amb dues branques, i proposem una vista frontal (FR-V) i una vista d'ocell (BE-V) com a representacions 2D del núvol de punts 3D que serveixen com a punt de partida per entrenar els nostres models. Més endavant, al Capítol 3 apliquem i provem de nou aquest mètode en dos casos d'ús reals, tant per filtrar obstacles en moviment prèviament a la creació de mapes en els quals poder localitzar-nos millor en dies posteriors, com per dur a terme el seguiment de vehicles. Des de la perspectiva dinàmica, al Capítol 4 aprenem una nova característica dinàmica del núvol de punts en 3D que s'assembla al flux òptic sobre imatges RGB. Per a fer-ho, desenvolupem un nou enfocament que aprofita el flux òptic RGB com pseudo mostres reals per a entrenament, utilitzant només informació 3D durant la inferència. Després, al Capítol 5 explorem els beneficis que s'obtenen de combinar els aprenentatges de problemes de classificació i regressió per la tasca d'estimació de flux òptic de manera conjunta. Finalment, al Capítol 6 posem en comú els mètodes anteriors i demostrem que mitjançant aquests processos independents podem abordar l'aprenentatge de problemes més complexos, com la segmentació i estimació del moviment de vehicles des de la nostra pròpia perspectiva.

Acknowledgements

I would like to express my deep gratitude to all the people that made this thesis possible. First of all, I would like to thank my advisors Alberto Sanfeliu and Francesc Moreno-Noguer for their guidance, support and help during these years.

Additionally, I would like to thank to all the colleagues and staff I had the pleasure to work with during these years in the Instituto de Robòtica i Informàtica Industrial (IRI), from which I did not only learn about science but also the most diverse topics, from beer brewing to prawn breeding. Special gratitude deserve my friends Juan, Julio, Noe and Karla, for all the positive lunch and coffee breaks, as well as all the colleagues that after our Master studies shared the research path with me. Also I want to mention my lab-mates from D19, always there for insightful conversations and technical advises, I would have never get here without you.

I am thankful to the colleagues from the CVC, specifically to Germán Ros for introducing me to the Deep side of the learning, as well as Gemma, Felipe, and the Carla team. Moreover I would like to thank to Stefan Milz and all the Valeo crew for giving me the opportunity of learning and develop part of my research there, as well as to Senthil Yogamani for making it possible and all his insights. Let me also express my gratitude to Adrien Gaidon, Rares Ambrus, Vitor Guizilini and all the TRI people for the incredible months in such a motivating, vibrant and enjoyable environment in Silicon Valley, it was a dream come true.

Para finalizar, quisiera dar las gracias a mi familia y amigos por su apoyo incondicional, en todos los sentidos. Han sabido soportarme y animarme en momentos duros, así como celebrar mis pequeños exitos con un orgullo contagioso que me ha hecho creer que este momento llegaría. Durante estos años además, he tenido la oportunidad de compartir mi tiempo con personas increibles que han pasado por mi vida, especialmente Pao, que me hizo ver la vida con otros ojos. Habéis aguantado pacientemente mi falta de horarios y pasión por este trabajo, creando con vuestro soporte un pilar fundamental de este logro.

Victor Vaquero
June, 2019

This work has been partially supported by the European projects:

FP7-SST-2013-RTD-1-605598 Cargo-ANTS: Cargo handling by Automated Next generation Transportation Systems for ports and terminals.

FLAG-ERA-JTC-2016 ROBOCOM++: Rethinking robotics for the robot companion of the future.

As well as by the Spanish Ministry of Education and Science under the research project:

DPI2016-78957-R ColRobTransp: Colaboración robots-humanos para el transporte de productos en zonas urbanas.

Contents

List of Figures

List of Tables

Nomenclature

$\mathcal{P} = \{q_1, \cdots, q_Q\}$	Point Cloud containing Q points
$q_k \in \mathbb{R}^4$	3D Point represented by Euclidean coordinates (x, y, z) + Reflectivity
$(x,\ y,\ z)$	Euclidean coordinates
$\mathbf{I}_{(\cdot)}$	LiDAR representation as input to a network. Front (FR). Bird's Eye (BE)
$\mathcal{F}_{(\cdot)}$	Model mapping an input to a desired output
$\mathbf{Y}_{(\cdot)}$	Ground Truth on the corresponding representation
$\hat{\mathbf{Y}}_{(\cdot)}$	Model prediction on the corresponding representation
$\theta_{(\cdot)}$	Model learnable parameters
$Id_{[\cdot']}(\cdot)$	Index function selecting corresponding probability according to class $[\cdot']$
$\omega(\cdot)$	Class weight to correct imbalanced classes
$\boldsymbol{\pi} = (x, y, z, \alpha)$	Bounding box pose: Euc. coords. and on-ground orientation
$\mathbf{d} = (w, l, h)$	Bounding box parameters, width, length, height
$(\phi,\ \theta,\ \rho)$	Spherical coordinates: azimuth, elevation and range
η	Confidence value of a generated cluster
$Q_{(FR)}$	Number of points in a cluster that the FR branch classified as vehicle
$Q_{(BE)}$	Number of points in a cluster that the BE branch classified as vehicle
ϵ	Bounding box fitting error
$\mathbf{S}_{BE} \in \mathbb{R}^{H' \times W'}$	Extended occupancy map in BE domain
Q_h	Number of points hitting a cell of the $\mathbf{S}_{BE}$ map
Q_f	Number of laser beams that pass over a cell of the $\mathbf{S}_{BE}$ map
z_g	Ground height
z_m	Lowest height value of any laser beam over a cell of the $\mathbf{S}_{BE}$ map
p_o	Probability of a cell in the $\mathbf{S}_{BE}$ of being occluded
p_h	Probability of a cell in the $\mathbf{S}_{BE}$ of being occupied
p_f	Probability of a cell in the $\mathbf{S}_{BE}$ of being free
C	Growing cost for a bounding box
ν	Confidence of a final bounding box
S	Final bounding box detection score for a detected vehicle
$\mathbf{Y}_{LiDAR}$	Ground Truth for LiDAR-Flow
$\hat{\mathbf{Y}}_{LiDAR}$	LiDAR-Flow estimation (at low resolution)
$\mathbf{Y}_{Dense}$	Dense Ground Truth for optical flow in the LiDAR overlapping area

$\hat{\mathbf{Y}}_{Dense}$	Dense optical flow hallucinated from LiDAR data
$\mathcal{G}_{\theta_{LiDAR}}$	Deep subnetwork for predicting LiDAR-Flow (low resolution)
$\mathcal{H}_{\theta_{Up}}$	Deep subnetwork for upscaling LiDAR-Flow to Image domain
$\mathcal{K}_{\theta_{End}}$	Deep subnetwork for refining dense hallucinated LiDAR-Flow
T	Transformation matrix for point cloud
$1st_r$	Frame number when the initial pose estimation occurs
$\#_r$	Number of frames at which an algorithm manages to relocate
β, $\beta_{\perp}$	Perpendicular motion orientations
$\mathbf{X}_t$	RGB image at time t
$\hat{\mathbf{Y}}^{\textit{class-X}}$	Predicted coarse flow for horizontal or vertical displacement
$\hat{\mathbf{Y}}^{\textit{reg}}$	Predicted fine flow from regression
m_r, M_r	Minimum and maximum flow bounds for classification
B_k	Optical flow classification bins
C_k	Classification bin centroid

Acronyms

AD	Autonomous Driving. 1, 5, 88
ADAS	Advanced Driving Assistance Systems. 1–3, 16
AV	Autonomous Vehicle. 1–4, 9
BB	Bounding Box. 16
BE-V	Bird's Eye View. 9
CaF	Coarse-and-Fine. 74
CNN	Convolutional Neural Network. 7–10, 15–19, 21, 23, 24, 58, 60–63, 73, 82, 87, 88, 98
CRF	Conditional Random Fields. 67
EDD	Effective Detection Distance. 37
EPE	End-Point-Error. 70, 81, 94, 95
FOV	Field of View. 22, 63, 65
ICP	Iterative Closest Point. 44, 45, 49
IoU	Intersection over Union. 34, 51
ITS	Intelligent Transportation System. 6
IV	Intelligent Vehicle. 6
MAE	Mean Absolute Error. 47
MDD	Maximum Detection Distance. 37
MSE	Mean Square Error. 80
SAE	Society of Automotive Engineers. 1
WCE	Weighted Cross Entropy. 77

1 Introduction

Autonomous Driving (AD) is foreseen as one of the most promising technologies of this century and is meant to disrupt our lives in the next decade. Indeed, it will transform the transportation paradigm as we know it today, providing a more efficient, reliable and affordable way of conveyance both for people and goods. As a direct consequence, new public transportation and shared services such as robo-taxis will be created, impacting on cities parking problems by having those vehicles running 24x7, as well as allowing commuters to take advantage of their travel time. However, benefits of these technologies are far deeper, as for example robust and reliable systems will decrease the 1.35 million fatalities per year caused by traffic accidents. In addition, the mobility options for elderly, disabled or people not fitted to drive will increase as well as traffic flow and fuel efficiency will improve, decreasing contamination in urban areas. Moreover, goods transportation and logistics will also be heavily altered, with the introduction of truck platoons and last mile urban deliveries as for example.

From the business point of view, the global market opportunity for Autonomous Vehicles (AVs) is established to be worth $\$54.23$ billion in 2019 and is predicted to scale to $\$800$ billion in 2035 and \$7 trillion by 2050 [1, 2]. With this perspective, it is not a surprise that research and development on this topic has gathered in the last years billions of dollars both in industry and academia, allowing AVs to steadily become a reality.

Preliminary research advances have been gradually transferred into commercial vehicles in the way of Advanced Driving Assistance Systems (ADAS). Assistance elements such as collision avoidance systems, automatic parking, lane guidance, driver awareness detection, adaptive cruise control or blind spot detection, are now rather integrated into the public psyche and no one disputes their importance or effectiveness. From their side, governments are also starting to legislate allowing to test and drive AVs on their roads (i.e. California since 2012).

Seen the fast development of vehicles autonomy the Society of Automotive Engineers (SAE) has proposed a taxonomy [3] to integrate in our roads these self-driving vehicles. It classifies

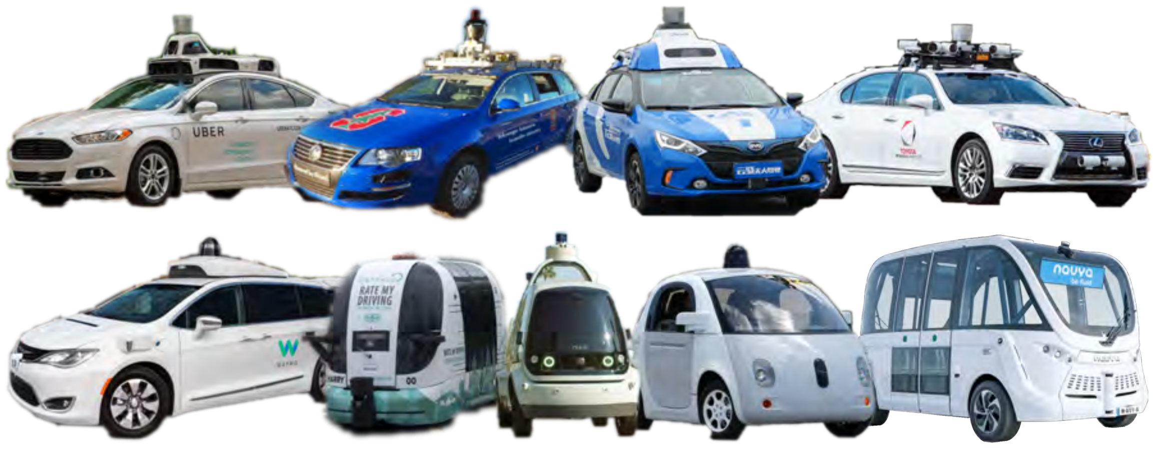


Figure 1.1: **Autonomous Vehicles (AVs) examples.** Top, some platforms employed to collect data and test driving algorithms. Bottom, some early developments of AVs providing services such as Robo-taxis, autonomous shuttle buses or last-mile autonomous delivery.

the AVs revolution in six levels diluting the human driver figure as we raise the automation level going from totally manual, where a human driver is required at all times, to full autonomy, where a vehicle operates by itself without a human driver. These levels are summarized as:

- **Level 0 – No Automation.** The driver fully controls the vehicle and the system only emit warnings but does not interfere at all. The driver must constantly monitor the drive.
- **Level 1 – Driver Assistance.** The driver controls the vehicle, but some ADAS such as Adaptive Cruise Control, Assisted Parking or Lane Departure Warning are perhaps present and are allowed to modify the speed or steering direction of the vehicle. Driver must be ready to resume full control immediately at any time.
- **Level 2 – Partial Automation.** The system may control the steering and acceleration of the vehicle in some defined cases such as Lane Keeping Assistance or Automated Parking. The driver must constantly monitor the drive and be ready to resume control immediately.
- **Level 3 – Conditional Automation.** The system may control the full vehicle if certain conditions are given for tasks such as Autonomous Parking, Highway driving or Stop & Go in traffic jams. The driver is still a necessity and must be ready to take control with notice, but is not required to constantly monitor the environment.
- **Level 4 – High Automation.** The vehicle can operate all driving functions autonomously in all but few conditions, such as severe weather. The driver may have the option to take control of the vehicle, although his/her attention is not required.

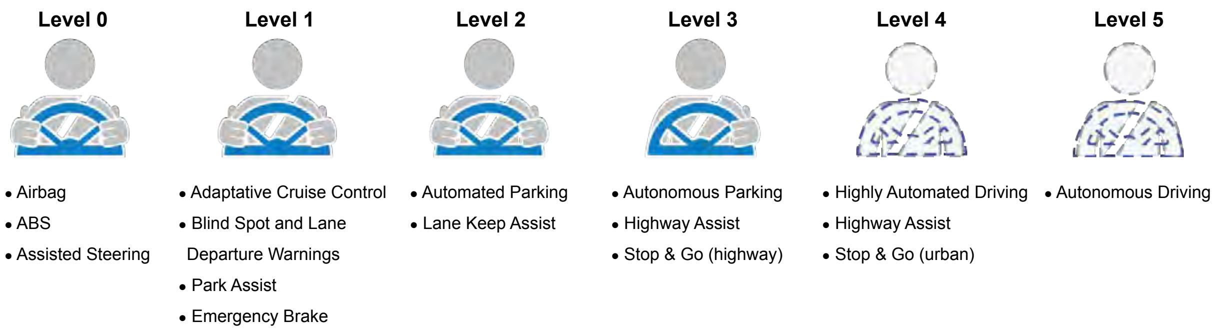


Figure 1.2: **SAE levels examples.** Some examples of driving assistance systems at the different SAE automation levels. As the level grows, the human driver need gets diluted.

- **Level 5 – Full Automation.** The vehicle is capable of performing all driving functions under all conditions without requiring human intervention. The driver becomes mostly a passenger although may have the option to take control of the vehicle.

In Fig. 1.2 we present some existing or projected ADAS examples for each of the aforementioned levels. At the moment, vehicles with SAE 2 and 3 levels of autonomy are already present on our roads. Moreover, traditional automakers have clarified their purpose of releasing fully automated level-5 agents in the near future. They have also forecasted that by progressively improving the technologies, there would be approximately eight million vehicles with SAE level 3, 4 or 5 capabilities sold by year 2025. However, providing AVs with the aforementioned capabilities presents several technological challenges, starting from the hardware with which the vehicles perceive the environment, continuing with the algorithms that analyze the complex live scene, and ending with the final control actions to execute.

In this thesis, we will focus on understanding both statically and dynamically the surrounding driving scene seen through the sensors of an autonomous vehicle. In other words, our objective is to discern the different elements of the scene and its motion characteristics by answering questions such as: where are the vehicles in there?, how are the dynamics of the scene evolving? and where are the different vehicles heading?

The knowledge of this dissertation intersects with the fields of robotics, computer vision and artificial intelligence. Along this introductory chapter, we first review in Section 1.1 how autonomous robots can perceive. We also introduce in Section 1.2 the computer vision subtopic known as scene understanding and why it is of vital importance for future urban robots and vehicles. Additionally, we show in Section 1.3 how deep learning techniques are greatly succeeding and advancing on both robot perception and scene understanding topics. Finally, we summarize the motivation of this thesis dissertation on Section 1.4, as well as respectively detail our objectives and contributions in Section 1.5 and Section 1.6.

1.1 Perceptive Sensors on Autonomous Vehicles

In general, AVs can be considered as robotic platforms that actively interact with the environment and move through it. Like any robot, these vehicles are endowed with three main components: I) a set of sensors allowing the perception of the exterior environment; II) hardware capable of processing the obtained information in real-time to analyze the situation; III) different actuators to perform the required control actions. The perfect symbiosis of these systems and components will provide the vehicles with different capabilities, therefore conditioning its grade of autonomy.

Yet, if we want autonomous vehicles to freely move and interact with our dynamic and challenging road and urban scenarios, we need to assure full security of all the scene interveners, both inside and outside the vehicle. In this way, self-driven vehicles should also be equipped with several sensors providing diverse and redundant information from the environment. Here we broadly revise the pros and cons of commonly used sensors on robotic platforms and autonomous vehicles, such as cameras, radars, lasers or ultrasounds:

- **Camera.** Optical RGB sensors are the most common source of information due to their versatility and reduced cost. The color images obtained with these sensors are analyzed with computer vision techniques tackling tasks such as object detection, segmentation, tracking, optical flow, etc. This field of research has experienced considerable advancements with the advent of deep learning technologies as will be shown later in Section 1.3.

  However, camera-based images get easily degraded due to external disturbances such as harsh weather conditions, e.g. heavy rain, fog, snow, etc. In addition, the same scene captured at different day times may look considerably different, for instance in daytime versus sunset or night. In the autonomous driving field, circumstances can be even harder as for example during night drives, where other vehicles' lights may dazzle the own camera-based perception systems spoiling the captured images.

- **Ultrasonic.** These sensors are used for distance measure attending to the principle of reflected sound waves. Nowadays, we already enjoy driving assistance applications in which these sensors take part, such as obstacle warning or parking distance measurement.

  Despite this success, their application to further distances is not feasible due to the lack of accuracy and robustness on their measurements. Sound waves propagation depend greatly on atmospheric conditions such as pressure and humidity from which minimal variations introduce long disturbances. This fact relegates ultrasonic family of sensors to assist in close range applications, precluding the obtainment of richer knowledge.

- **Radar.** Automotive radars are a family of sensors that emit radio signals and capture their echoes, relying on the Doppler effect to obtain sparse measurements about the dynamics of the scene. Although their outputs are usually noisy, radars provide very valuable information in autonomous driving environments because dynamic elements are commonly the ones throwing more danger on the ego-vehicle drive.

  Notwithstanding, these sensors use a short wavelength radio signal which, while enables them to take far measurements, does not allow the detection of small objects and introduce additional noise. Moreover, radar sensors do not provide robust information about the static structure of the close-by environment, also vital for AD applications.

- **LiDAR.** Light Detection and Ranging (LiDAR) sensors emit pulsed light beams (lasers) at high rate and measure the reflection time obtaining the distance to the collision, known as range. Additionally, some sensors can also measure the reflected energy, obtaining information about the reflectivity of the collision material. Initial developments were composed by a single laser rotating around itself, being able to generate accurate 2D maps of a slice of the directly seen environment in the shape of a point-cloud. Evolution of this technology now is able to stack several rotating lasers vertically to measure millions of 3D points, which generates denser three-dimensional maps of the surrounding scene.

  LiDAR sensors are robust to external weather and light conditions, and very reliable in terms of accuracy of the measurement. As a drawback, these sensors contain abundant moving parts, which leaves room for mechanical errors. Moreover, some inaccuracies can also exist due to the collision materials. For example black objects can absorb much of the emitted light maybe generating no-measurements and windows or translucent materials may produce several reflections.

Due to their robustness and the valuable information provided, in this thesis we explore the capacities of LiDAR sensors as the only input to understand the driving scene. For that we will use LiDAR-generated point clouds in combination with the latest deep learning techniques to recognize the different elements on the scene and analyze their dynamic behaviors and relationships.

1.2 Scene Understanding

In computer vision, scene understanding refers to the set of tasks aiming to accurately and comprehensively recognize and understand the complex visual world to obtain a human-like interpretation [4, 5]. However, fully comprehension of the real environment is one of the fundamental challenges that the robotics community is still struggling to solve. In this specific context, scene understanding tasks must create a helpful model of the environment from which a robot can infer the general rules, behaviors and relations between components in order to interact with it. In other words, scene understanding can be thought as the initial step from which the robot can start making smart decisions [6].

In traffic related situations, capturing and understanding the highly dynamic urban scenarios is of vital importance in order to take decisions and navigate efficiently and safely. Future level-3 and further Intelligent Vehicles (IVs) and Intelligent Transportation Systems (ITSs) therefore require precise information about the semantic and motion characteristics of the objects in the scene, enabling robust applications such as obstacle avoidance, vehicle tracking, navigation, localization or map refinement. Special attention should also be paid to moving elements in traffic situations, e.g. vehicles/pedestrians, and moreover if their motion paths are expected to cross the direction of other objects or the observer.

To fully understand a scene we need to incorporate as much information from the environment as we can. Sensor fusion, understood as the ability of merging information from different sources, could allow to obtain a broader perception of the environment. As shown in Section 1.1 the data provided by the usual sensors turns to be complementary, as for example cameras are able to provide textures during daylight while LiDARs capture the depth obtaining 3D interpretation even at night. In this way, a synergistic early fusion of both information sources can help on overcoming their independent drawbacks, e.g. color textures can be added to a 3D point cloud by fusing cameras and LiDAR.

However, with these strategies a single sensor failure may compromise the full perception system. Therefore redundant systems based on different individual sensors should be included to add robustness to the vehicle's environment interpretation. High level algorithms can later gather the processed information from different sensors and fuse it in a more abstract level, which is commonly called a late fusion strategy.

In this thesis we employ this second approach, as it also protects to the unlucky event of failure of any of the sensors. We want to exploit LiDAR information and develop complete scene understanding algorithms that can act as fully aware redundant systems on the vehicle or even can be used to fuse the obtained understanding at higher levels. Additionally, we explore the use of early sensor fusion strategies, incorporating information from other sources in the training steps of our algorithms, but eliminating its need at deployment time.

1.3 Deep Learning

Perception algorithms for robots and autonomous vehicles are living their golden age due to the advances in Deep Learning. Classical vision tasks are experiencing a great boost after in 2011 a method [7] based in Deep Neural Networks, and more specifically Deep Convolutional Neural Networks (CNNs) [8], won with a large margin the ImageNet [9] competition on image classification. The successful application of these deep techniques to solve further challenging perception problems such as image classification, object detection, semantic segmentation or optical flow prediction [10–17], has initiated a vast transition in the research community towards these data driven learning methods.

Two main factors are responsible for this CNN renaissance. On the one hand, new hardware advances (e.g. GPUs, TPUs) as well as developing tools and libraries are now able to provide much faster development and computation while enabling their use for a broader public. On the other hand, the availability of new labeled datasets and benchmarks freely available to the community provide vast sets of samples to train and test these deep data driven models as well as a fair testbed for comparing algorithms results, alleviating the time and money effort for individual people to capture and annotate their own samples. In this way, the confluence of these factors has allowed research advances in areas such as robotics, machine learning, and computer vision, endowing perception systems with an everyday-higher level of scene comprehension.

Fresh CNN-based approaches are day after day becoming the new state of the art, producing models which prediction accuracy was inconceivable few years ago. Yet, the success of these systems for each different task heavily rely on a smart design of three main elements: i) the representation of the problem, ii) the training method and iii) the network architecture. As a general practice, the task under study is represented as a set of classification or regression problems, depending on the nature of the task. For example, it is common to represent semantic segmentation [12, 15] as multiple classification problems over a finite and discrete set of categories, while motion related tasks —such as optical flow prediction [16,18]— can be represented as regression problems over a continuous 'flow space'. To ensure the correct behavior of the full designed system, the training method needs to reflect the chosen representation with an appropriate loss function. Typical examples of losses are cross entropy and mean squared error, associated to classification and regression problems respectively. The last element, the network architecture, needs to provide enough capacity for the approximation of the task and support the propagation of the gradients in order to make the training possible. The use of certain network designs, as for instance the architectures based on residual blocks has proven to yield a notorious improvement in speed and accuracy [13].

In this thesis, we will employ Deep Learning techniques over LiDAR generated 3D point clouds and expand the limits of actual deep CNNs models to the 3D domain. Therefore, we aim to cover a research gap on analyzing 3D point clouds to tackle several challenges such as vehicle detection and tracking, optical flow estimation or motion segmentation, while obtaining the benefits and successful results of these data-driven deep techniques.

1.4 Motivation

This PhD dissertation is motivated by the confluence of different factors. Nowadays, LiDAR sensors from robotic platforms are able to provide accurate and robust 3D measurements that current algorithms can struggle to process in real-time. Moreover, as we have previously seen, Deep Learning has become a de-facto game-changer disrupting the Computer Vision field, but its application to other domains further than RGB images still needs additional research. Within this context, our motivation is to push forward the state of the art on the use of deep learning models for processing 3D LiDAR-generated point clouds in order to perform scene understanding tasks.

In other words, the motivation of this PhD thesis is to cover new unexplored areas for processing LiDAR-based 3D point clouds to perform scene understanding in driving situations by employing Deep Learning approaches.

1.5 Objectives and Scope

The main objective of this PhD dissertation is to contribute to the state of the art and develop novel data-driven approaches to perform high level scene understanding tasks using as only input 3D point clouds, for which we intend to use Computer Vision and Deep Learning techniques. In other words, we aim to answer questions such as what is on the scene? how are the dynamics of the situation evolving? where are the moving elements going?

More specifically, we set the following objectives:

- To explore new 3D LiDAR point cloud representations to serve as input to train CNNs.
- To create novel deep convolutional neural networks to segment vehicles from 3D point clouds.
- To explore new deep learning architectures for obtaining dynamic features from the scene, such as for example optical flow estimation.
- To develop strategies for fusing information in order to solve complex scene understanding tasks, such as motion estimation and segmentation.

- To analyze the use of prior knowledge and auxiliary/pretext tasks such as vehicle segmentation or optical flow estimation to guide the learning solution of higher level scene understanding objectives.
- To examine cross-domain techniques to assist the training of deep algorithms, such as the use of RGB images for learning tasks that will only require 3D LiDAR inputs on inference.
- To use and advance deep learning frameworks, tools and datasets in order to train and validate the developed algorithms.

1.6 Contributions and Thesis Overview

As aforementioned, the main ambition of this PhD is to contribute to the development of AVs by proposing new alternative solutions for scene understanding tasks using Deep Learning techniques and LiDAR-based information. To this end, we devise three different parts in this dissertation.

In part I we focus on 'what' can be seen on the scene. We study and propose in Chapter 2 novel methods to fast and robustly segment and extract the surrounding vehicles from the 3D point clouds obtained by the LiDAR sensor of an autonomous car. Moreover, in Chapter 3 we apply and test the developed ideas on real uses-cases. Firstly, for building better maps with standard SLAM algorithms based on longer lasting features with improved invariance properties which allow better relocalization on subsequent days. Secondly, for performing 3D vehicle detection and tracking on driving situations by mixing novel deep methods with classic multi-hypothesis Kalman Filters for tracking. Our main contributions in this part are:

- We propose two different encodings for the 3D LiDAR point cloud to serve as inputs to standard CNNs. In this way we devise an equivalent 2D Front representation (FR-V)and a Bird's Eye View (BE-V) (i.e. a zenithal view).
- We develop and train novel CNN architectures to segment vehicles from the two proposed encodings and build a multi-view 3D detection framework that performs a late-fusion strategy on the segmented data to obtain and validate 3D bounding boxes [19–21].

In part II, we focus on analyzing the dynamics of the scene. We here aim to find comprehensive dynamic features of the scene resembling the optical flow typically used in RGB images, but only using LiDAR information. For that, in Chapter 4 we first investigate how to estimate an equivalent image-based optical flow but using as primary source of information only the LiDAR sensor. For that, we may exploit RGB images during training, but use only 3D point clouds at inference. Additionally, in Chapter 5 we explore the combination of classification and regression losses to solve a single task, and propose a joint Coarse-and-Fine (CaF) reasoning for predicting optical flow from RGB images. The main contributions of this part are:

- We develop a novel dynamic feature using only LiDAR information that is able to substitute standard RGB-based optical flow [22].
- We further propose a deep learning approach to tackle the optical flow task by jointly solving a classification and a regression problem [23].

Finally in part III we tackle higher complexity tasks such as finding the on-ground motion vectors of moving vehicles in the scene, following the next contributions:

- We propose a novel CNN architecture to segment the moving vehicles as well as obtain their on-ground motion vector.
- We exploit the use of prior knowledge in the way of pretext/side tasks with the intention to guide the learning problem towards the final complex objective [24].

1.7 Publications

In the course of realization of the work detailed on this dissertation, the author collaborated in the development of other scientific publications [25–28]. Despite the fact that they are slightly related to the work presented in this thesis, these have not been detailed into the actual document, as including them would have diverged the final scope. A full list of the scientific publications accepted during the realization of this thesis is presented next in Section 1.7. Further achievements such patents obtained, workshops organised with international colleagues and international research stays completed in the scope of this thesis are enclosed in Section 7.1.

1.7.1 International Conferences

- V. Vaquero, K. Fischer, F. Moreno-Noguer, A. Sanfeliu, and S. Milz, "Improving map re-localization with deep 'movable' objects segmentation on 3d LiDAR point clouds", in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.
- V. Vaquero, A. Sanfeliu, and F. Moreno-Noguer, "Deep LiDAR CNN to understand the dynamics of moving vehicles", in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

- V. Vaquero, A. Sanfeliu, and F. Moreno-Noguer, “Hallucinating dense optical flow from sparse LiDAR for autonomous vehicles”, in *IEEE International Conference on Pattern Recognition (ICPR)*, 2018.
- V. Vaquero, G. Ros, F. Moreno-Noguer, A. M. Lopez, and A. Sanfeliu, “Joint coarse-and-fine reasoning for deep optical flow”, in *IEEE International Conference on Image Processing (ICIP)*, 2017.
- V. Vaquero, I. del Pino, F. Moreno-Noguer, J. Solà, A. Sanfeliu, and J. Andrade-Cetto, “Deconvolutional networks for point-cloud vehicle detection and tracking in driving scenarios”, in *European Conference on Mobile Robotics (ECMR)*, 2017.
- I. del Pino, V. Vaquero, B. Masini, J. Solà, F. Moreno-Noguer, A. Sanfeliu, and J. Andrade-Cetto, “Low resolution LiDAR-based multi-object tracking for driving applications”, in *3rd Iberian Robotics Conference*, Adv. Intell. Syst. Comput., pp. 287–298, Springer, 2017.
- V. Vaquero, E. Repiso, A. Sanfeliu, J. Vissers, and M. Kwakkernaat, “Low cost, robust and real time system for detecting and tracking moving objects to automate cargo handling in port terminals”, in *2nd Iberian Robotics Conference*, Adv. Intell. Syst. Comput., pp. 491–502, Springer, 2015.

1.7.2 Scientific Journals

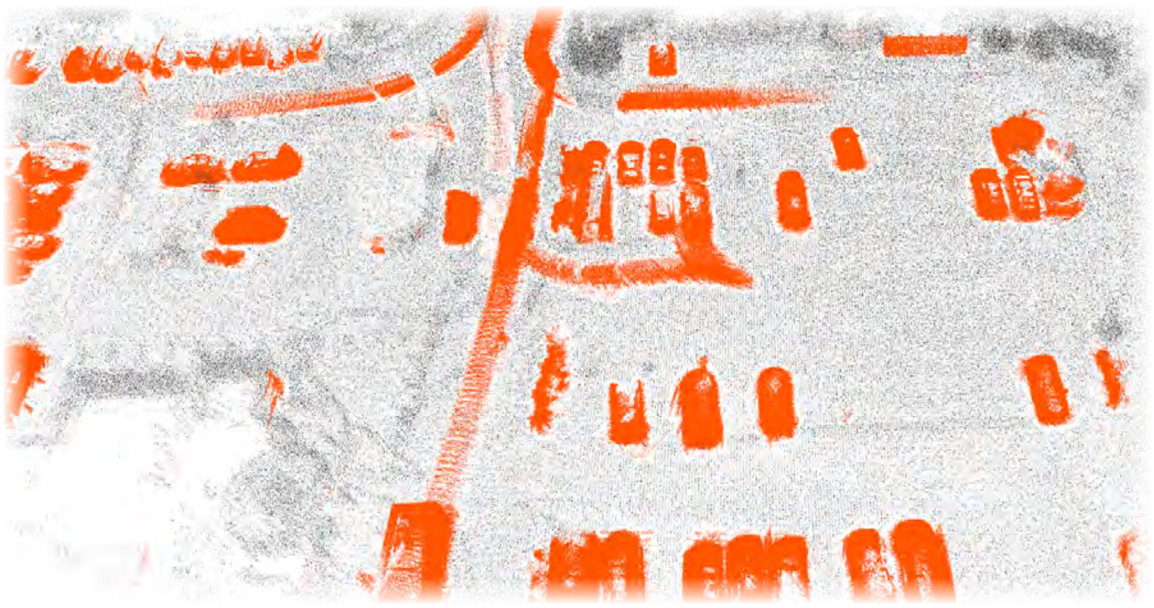
- V. Vaquero, I. del Pino, F. Moreno-Noguer, J. Solà, A. Sanfeliu, and J. Andrade- Cetto, “Dual-branch cnns for vehicle detection and tracking on LiDAR data”, under revision in *Transactions on Intelligent Transportation Systems*, 2019.
- V. Vaquero, E. Repiso, and A. Sanfeliu, “Robust and real-time detection and tracking of moving objects with minimum 2d LiDAR information to advance autonomous cargo handling in ports”, *Sensors*, vol. 19, no. 1, p. 107, 2019.

1.7.3 Workshop Publications

- H. Rashed, M. Ramzy, V. Vaquero, A. El Sallab, G. Sistu, and S. Yogamani, “Fusemodnet: Real-time camera and LiDAR based moving object detection for robust low-light autonomous driving,” in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019

Part I

What's around me?



Vehicle Segmentation over accumulated LiDAR pointclouds.

2 Vehicle Detection using LiDAR

In this first part of the thesis we tackle the LiDAR-based scene understanding challenge from a static perspective. This is, we focus our attention on extracting information from individual 3D LiDAR point clouds, such as where are the vehicles or obstacles around.

In this Chapter 2 we present a novel dual-branch vehicle detection system that works solely on 3D LiDAR information. For it, we devise a dual-view representation of the input 3D LiDAR point cloud. On the one hand, we generate a frontal projection of the point cloud featuring the range and reflectivity measurements, whereas on the other hand we introduce a bird's eye view projection with six different feature maps encoding various occupancy, reflectivity, and height parameters. These two representations of the input LiDAR measurements are fed into two independently trained CNNs, one for each of the views, and bounding boxes are extracted over the fused outputs.

Our system is thoroughly evaluated on the KITTI Detection benchmark, showing that our dual-branch detector consistently outperforms previous single-branch approaches, and directly competes with other state of the art LiDAR-based methods.

Later in Chapter 3, we present two real use-cases on which our segmentation approach is successfully applied. In this manner, in Section 3.1 we will employ the method described in this chapter to filter from each LiDAR input frame the objects that can move in the scene, to obtain longer standing maps with strong static features for further re-localization purposes [21]. Additionally, in Section 3.2 we show that outputs of our LiDAR-only 3D detection framework are sufficient to serve as observations for a tracking system based on a multi-hypothesis Kalman Filter, which is evaluated over the KITTI Tracking benchmark separately [19, 20].

2.1 Introduction

Although Autonomous Vehicle technologies are progressing rapidly, in part boosted by the new ADAS developments, there is still a long road until level-5 vehicles can drive freely in our cities. Real-world traffic situations raise very challenging scenarios that contain a great variety of elements like vehicles, cyclists, pedestrians, or even street furniture. The ability to detect frame-by-frame those elements is thus of vital importance and a core module to develop further systems like vehicle tracking, cruise control, collision avoidance or localization [19, 29–32].

As stated in Chapter 1, with the advent of deep learning technologies, image-based scene understanding has experienced a remarkable boost, providing in-vehicle perception systems with strong capacities on tasks such as object detection, classification or semantic segmentation [7, 11, 15]. Nevertheless, optical cameras may fail to correctly capture the environment under certain conditions, such as abrupt changes of illumination (entering a tunnel, light flashes, etc.) or harsh weather conditions (heavy rain, fog, snow, etc.). Additional sensors are thus required to fulfill the need of robustness on autonomous vehicles, whether for providing complementary scene information to the existing algorithms or to act as a full backup system.

LiDAR sensors are especially suitable for this purpose since they provide accurate spatial information while being robust to environmental conditions and their performance is independent to the scene illumination. Despite the remarkable results on camera images, the adoption of deep learning-based methods for LiDAR data is far from the level achieved on image processing tasks. Yet, recent works [19–21, 33–38] are pointing at deep learning techniques as powerful tools to extract information from point clouds, expanding their applicability beyond optical cameras. However, those approaches commonly use additional RGB data, complex structures with 3D or sparse convolutions, or end-to-end systems yielding bounding boxes directly without allowing the introduction of prior knowledge.

In this Chapter we present a novel segmentation and bounding box detection system for vehicles in driving scenarios based solely on 3D LiDAR data, as shown in Fig. 2.1. Our approach displays a dual-branch deep-LiDAR processing pipeline. Each branch analyses a different projection of the input 3D point clouds with a CNN and segments the existing vehicles as output. The resulting predictions are then projected back to the 3D Euclidean space to extract the surrounding vehicle bounding boxes. In a preliminary version of this work [19], we segmented vehicles from just a front projection of the LiDAR data employing a deconvolutional neural network and extracted its Bounding Boxes (BBs) using simple Euclidean clustering. In this Chapter we go beyond that initial work and heavily improve our system by introducing a new dual-view LiDAR pipeline along with several other novelties such as a recursive clustering using an adaptive threshold and a bounding box growing algorithm guided by contextual information.

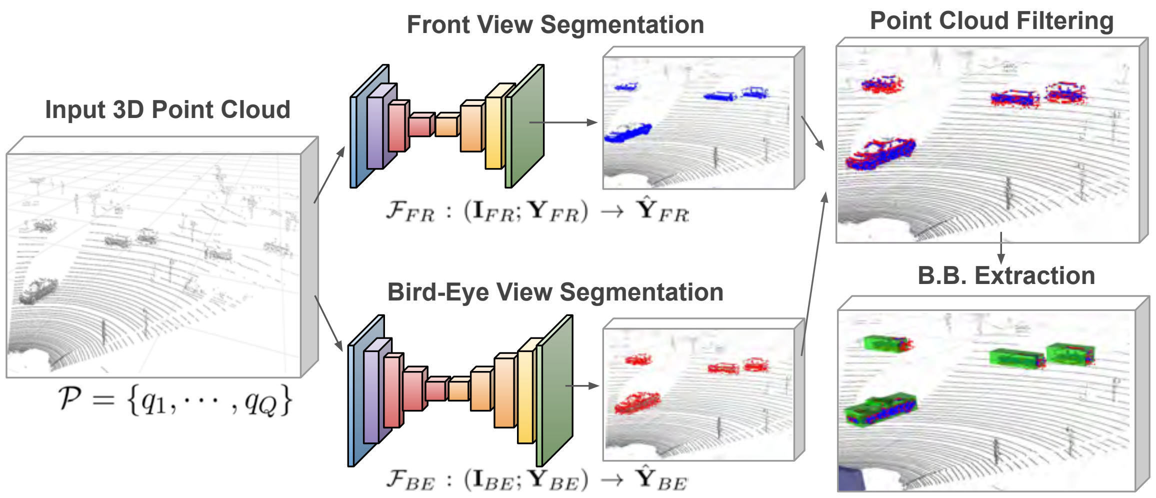

Input 3D Point Cloud
$\mathcal{P} = \{q_1, \cdots, q_Q\}$
Front View Segmentation
$\mathcal{F}_{FR} : (\mathbf{I}_{FR}; \mathbf{Y}_{FR}) \rightarrow \hat{\mathbf{Y}}_{FR}$
Bird-Eye View Segmentation
$\mathcal{F}_{BE} : (\mathbf{I}_{BE}; \mathbf{Y}_{BE}) \rightarrow \hat{\mathbf{Y}}_{BE}$
Point Cloud Filtering
B.B. Extraction

Figure 2.1: **LiDAR-based dual-branch vehicle segmentation and detection.** Using solely LiDAR information, our dual-view deep convolutional architecture is able to segment vehicles in real driving scenarios and extract their bounding boxes. The presented approach uses two different representations of the 3D input point cloud that are fed into two independent CNN-based branches for per-point vehicle classification. The outputs are then fused in a single 3D point cloud and recursively clustered. A novel bounding box growing method that relies on the use of contextual information generates trustworthy bounding boxes.

Extensive testing is performed in the KITTI Detection benchmark [6], quantitatively showing better performance in comparison to the state of the art. In addition, in Chapter 3 we will show the real applications of our approach to a pre-filtering approach for robust localization and a tracking system. Specifically, the contributions of this chapter are:

- We employ a dual-view deep architecture with specialized parallel branches segmenting vehicles from two different 3D point cloud representations. On one side a CNN architecture processes a front view projection. On the other, a novel deep architecture with fire modules [39] processes a featured bird's-eye view projection, which represents objects on their real on-ground position preserving its geometry and providing better size invariance.
- We develop a fusion strategy to retrieve accurate 3D bounding boxes from segmented vehicles which has three main features: a) a recursive Euclidean clustering with an adaptive threshold; b) a confidence metric to fuse detections from both branches exploiting the fact they produce uncorrelated false positives; and c) a method to recover full-size bounding boxes by expanding the small ones towards occluded regions, thus resolving possible orientation ambiguities.
- We extensively test our system in the KITTI Detection benchmark and present an ablation study testing our detection capacities at different distances. We comparare with other LiDAR-only works, showing better performance of how our dual-view deep-LiDAR method.

2.2 Related Work

Despite the great success of CNNs on image-based tasks such as object classification [7, 13], detection [11, 40, 41], or semantic segmentation [15, 42], the potential of these techniques has not yet been extensively deployed for analyzing 3D LiDAR point clouds. In this Section we review the most relevant approaches proposed to detect objects in such a sparse domain. We first do a small review of classical object detection algorithms in 3D point clouds, and then move forward to methods using deep learning technologies, and more specifically CNNs.

2.2.1 Classical Object Detection in LiDAR Point Clouds

There exists extensive literature about detecting objects in LiDAR-generated point clouds. Most common approaches, initially segment the point cloud using clustering algorithms to group points together and later classify them [25, 43–45]. These methods typically hold for both single (2D) and multi-layer (3D) LiDARs. For the latter, voting schemes are also used to vertically fuse single-layer clusters, obtaining part-based models of the objects [46, 47]. However, clustering over hundred of thousands of 3D points that modern LiDARs generate on each revolution can become very computationally expensive.

Including prior information can alleviate the clustering process. For example, in autonomous driving applications it is a common practice to firstly remove the ground-plane points [48, 49], as it is known that the scene elements are over the ground floor. Further methods try to reduce computational costs by voxelizing the 3D space. Connectivity graphs can be later built on top of the obtained voxels and exploited in posterior classification steps [50, 51]. More recent approaches directly scrutinize the 3D point cloud space with sliding window techniques. In this way, *Vote3D* [52] applies 3D sliding windows of different sizes and orientations over an encoded LiDAR space that contains diverse features such as an occupancy map, the measured intensity mean and its variance, and up to three other shape factors. The resulting candidate objects are then classified using Support Vector Machines and a voting scheme.

Once the clusters are obtained, these traditional approaches usually follow the computer vision pipelines for RGB images and extract hand-crafted features from the obtained groups of points. A detailed list and description of common 3D hand-crafted features can be found in [53], including spin images, shape models and other geometric statistics. Other useful features have also been obtained using classic learning procedures, such as in [54], where authors apply learning techniques over segmented regions of LiDAR scans reformatted into regularly sampled depth images.

2.2.2 Deep Learning for 3D LiDAR Object Detection

Initially, deep convolutional models were applied over 3D LiDAR point clouds as a way of learning useful features that can substitute hand-crafted ones, as in *Vote3D* [52].

Other works aim at obtaining an end-to-end deep learning solution. A straightforward approach can be to employ 3D convolutions, like it was done in [55] for 3D vehicle detection. However, this approach implies a high computational cost due to the additional dimension included on each convolutional filter to swap, as well as inefficient due to the sparse domain in which the filters are deployed. To deal with these problems, sparse convolutions were used on point clouds. In such a way, [56] extends the work of *Vote3D* [52] by replacing the support vector machine classifier with sparse 3D convolutions that act as voting weights for predicting the detection scores. Other works solve the sparsification problem by directly generating a structured space that subdivides the point cloud into voxels [37]. Then, the group of points within each voxel is transformed into a unified feature representation by a voxel feature encoding layer, to which standard convolutional filters are later applied.

Recent methods directly use the raw point cloud as input to deep architectures. PointNet [57] applies a set of transformations and multi-layer perceptrons obtaining features from each 3D point in a permutation-equivariant way which are later pooled to generate global point cloud features used for classification and segmentation tasks. The main drawback is that the method does not capture the local structures inherent to the points' metric space. Aiming to solve this issue, PointNet++ [58] proposes to recursively apply PointNet on nested areas of the input point cloud, learning local features with increasing contextual scales. The same ideas are shared in [36], which explores larger areas by extracting 3D frustums lifted from the corresponding bounding boxes predicted by a 2D CNN detector over RGB images. All these methods are often influenced by the point cloud density and contain ad-hoc steps which can greatly affect its performance and stability.

Nevertheless, nowadays the most commonly adopted procedure is to pre-process the 3D LiDAR point cloud to obtain equivalent 2D representations in which to apply the already well-known and optimized 2D CNN techniques and architectures. For instance, [34] projects the input point cloud to a front view representation encoding the range distance and height of each 3D point. They next train a fully convolutional network to predict the '*vehicleness*' confidence of each point and regress a 3D bounding box of the containing vehicle per point, which increase the computational load of the method. Similarly, in [19] we have segmented vehicles from a front view projection of the polar LiDAR coordinates that encodes range and reflectivity information.

Contrarily, BirdNet [59], TopNet [60] or RT3D [61] use only a bird's eye view projection (zenithal view) of the input LiDAR point clouds, encoding different features over each grid-cell.

Other approaches, such as [35], fuse different domains and combine RGB images and LiDAR front and bird's eye views. In their work, authors use only the bird's eye view to generate 3D bounding box proposals that are then deployed over the RGB images in a region-based fusion network, which does not fulfill the LiDAR-only requirement that we impose to our work.

Instead of using the bird's eye view just for bounding box proposals as [35], in this Chapter we devise a dual-branch LiDAR detector over a Front and a Bird's Eye views where both branches predict the probability of each of its input *pixels* to belong to a vehicle. Moreover, we perform a late fusion step of the segmented point clouds to extract vehicle bounding boxes in an efficient way thanks to the prior information given by probabilistic occupancy grid, which none of the previous works do. Our approach uses only LiDAR information and we do not make any use of RGB images during training nor at inference, obtaining a full accurate replacement system suitable for safety purposes in Autonomous Driving scenarios.

2.3 Approach

Our objective is to detect vehicles from just 3D LiDAR independent scans. Let us define an input point cloud as $\mathcal{P} = \{q_1, \cdots, q_Q\}$, where each point $q_k \in \mathbb{R}^4$ contains the point's Euclidean coordinates with respect to the sensor center of projection and a measured reflectivity value. We want the system to output a list of vehicle bounding boxes defined by their pose on the ground $\boldsymbol{\pi} = (x, y, z, \alpha)$, being x, y, z the Euclidean coordinates and α the on-ground rotation angle, as well as by their bounding box parameters $\mathbf{d} = (w, l, h)$. Our proposed solution to the problem includes two separated steps namely, vehicle segmentation on the 3D point cloud, and bounding box extraction. A general sketch of the developed approach is shown in Fig. 2.1.

We formulate the vehicle segmentation task as a per-point classification problem in which we want to obtain the probability of each 3D point to either belong or not to a vehicle: $p(l|q_k)$, where l represents the labels {*vehicle, not-vehicle*}. In order to accomplish this challenging problem, we define two different projections of the input 3D LiDAR point cloud $\mathcal{P}$: a front view $\mathbf{I}_{FR}$; and a bird's-eye view $\mathbf{I}_{BE}$, which are shown in Fig. 2.2. We therefore set a learning objective to model two functions $\mathcal{F}_{FR} : (\mathbf{I}_{FR}; \mathbf{Y}_{FR}) \rightarrow \hat{\mathbf{Y}}_{FR}$ and $\mathcal{F}_{BE} : (\mathbf{I}_{BE}; \mathbf{Y}_{BE}) \rightarrow \hat{\mathbf{Y}}_{BE}$, where $\mathbf{Y}_{FR}$ and $\mathbf{Y}_{BE}$ are the ground truth binary masks indicating whether or not each projected point belongs to a vehicle and $\hat{\mathbf{Y}}_{FR}$ and $\hat{\mathbf{Y}}_{BE}$ contain the estimated *vehicleness* probability map for the two projection planes. We will learn these functions $\mathcal{F}_{FR}$ and $\mathcal{F}_{BE}$ as two independent deep convolutional neural networks with learnable parameters θ_{FR} and θ_{BE} respectively. In such a way, once the network training is finished and the corresponding parameters are found, we can infer our predictions as $\hat{\mathbf{Y}}_{FR} = \mathcal{F}_{FR}(\mathbf{I}_{FR}, \theta_{FR})$ and $\hat{\mathbf{Y}}_{BE} = \mathcal{F}_{BE}(\mathbf{I}_{BE}, \theta_{BE})$.

We next reconstruct an annotated point cloud from the estimated probability maps and

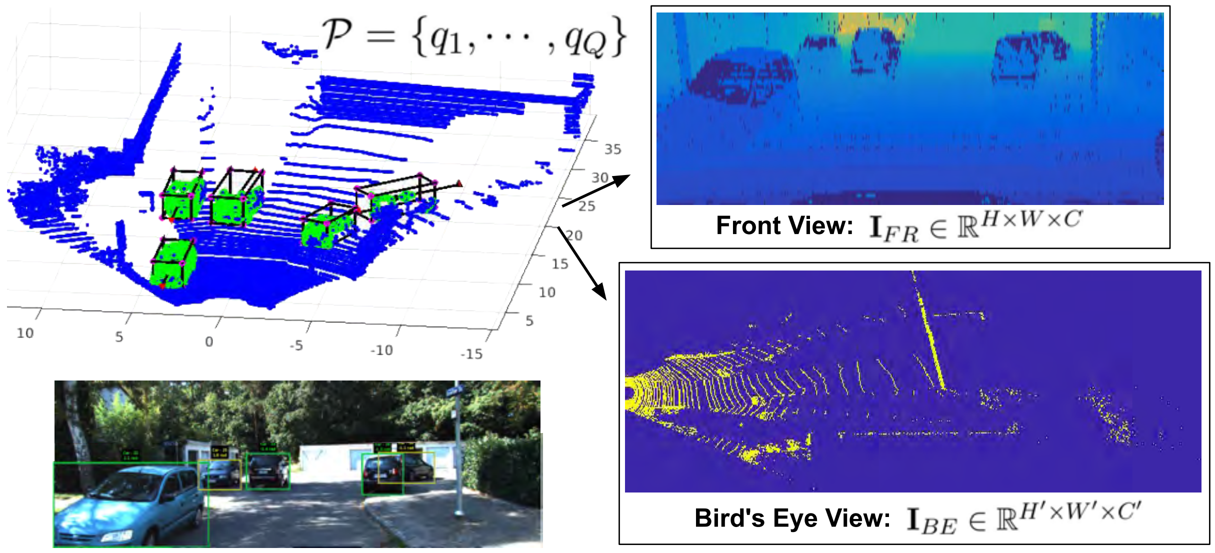

$\mathcal{P} = \{q_1, \cdots, q_Q\}$
Front View: $\mathbf{I}_{FR} \in \mathbb{R}^{H \times W \times C}$
Bird's Eye View: $\mathbf{I}_{BE} \in \mathbb{R}^{H' \times W' \times C'}$

Figure 2.2: **LiDAR representations for our Deep Neural Networks.** We project each captured 3D point cloud into two view planes to obtain structured inputs for our deep CNN detectors: a Front view, $\mathbf{I}_{FR}$ (top-right), and a Bird's-Eye view, $\mathbf{I}_{BE}$ (bottom-right). Ground truth for both views is generated by projecting back the image-based 3D bounding boxes over the LiDAR data and selecting the 3D points inside. RGB is shown here just for visualization purposes.

generate 3D bounding box proposals for each segmented vehicle via recursive clustering. A third probability map $\mathbf{S}_{BE}$ is created in the bird's eye view to encode the probability of each cell to either be occupied, free, or occluded, which is used in turn to grow the proposed boxes to standard vehicle sizes giving preference to occluded regions.

We next detail in Section 2.4 the Deep LiDAR-based vehicle detection phase, explaining the generation of both point cloud views employed and the developed deep architectures. Then, Section 2.5 focus on the bounding box extraction procedure. Finally, in Section 2.6 we show our detection results in the competitive KITTI Detection benchmark.

2.4 Vehicle Detection

2.4.1 3D Point Cloud Projections

One of the key reasons why convolutional networks are remarkably successful in computer vision tasks, is because standard RGB images are well structured and stable representations. To overcome common issues with the unordered and variable number of measurements existing in LiDAR data, we first project each point cloud scan into two different planes, a Front view, $\mathbf{I}_{FR}$, and a Bird's-Eye view, $\mathbf{I}_{BE}$, as shown in Fig. 2.2 and detailed in the following sections.

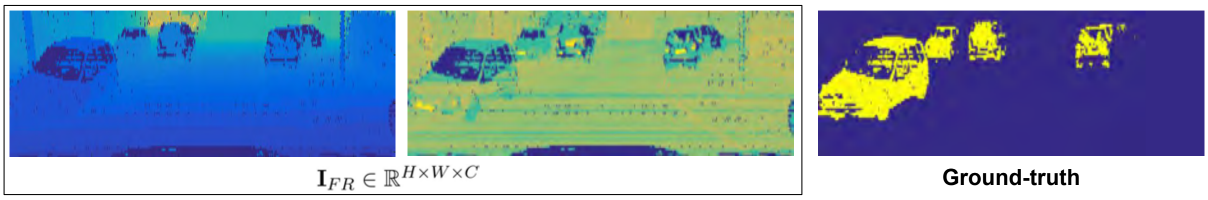


Figure 2.3: **LiDAR Front projection.** Our Front view projection, $\mathbf{I}_{FR}\ in\ \mathbb{R}^{H \times W \times C}$ encodes the LiDAR point cloud that overlaps the frontal RGB image, as it is the only annotated part. The subset of 3D points are transformed to polar coordinates and discretized according to the sensor geometry to obtain a final image of size $H = 64$, according to the each vertical laser on the KITTI Velodyne sensor, $W = 448$ according to each horizontal step on which a laser pulse is emitted and $C = 2$ channels encoding range and reflectivity channels for each 3D point.

Front-View Data Representation

Our front-view $\mathbf{I}_{FR}$ projection is obtained by modeling the Velodyne HDL-64 LiDAR geometry and arranging each 3D point from the point cloud $\mathcal{P}$ into a 2D array $\mathbf{I}_{FR} \in \mathbb{R}^{H \times W \times C}$. Initially, each point (x, y, z) is transformed to spherical coordinates (ϕ, θ, ρ). According to the sensor model, the elevation angle θ represents each of the $H = 64$ horizontal laser beams. In our sensor model, this array of lasers covers a vertical resolution from $-24,5$ to $+2$ degrees with variable resolution $\Delta\theta$ of $1/3$ degrees for the upper half of the sensor and $1/2$ degrees for the lower half. Even though the laser point cloud has a large horizontal field of view, we are forced to filter the point cloud in the range $\phi \in [-40.5, 40.5]$ degrees because the KITTI dataset has only annotated ground truth for those objects inside the camera Field of View (FOV). The resulting cropped point cloud is then discretized according to an azimuth step of $\Delta\phi = 0.18$ degrees, as specified by the sensor manufacturer, resulting in a map of width $W = 448$. The third dimension in our front view map represents $C = 2$ channels storing the sensor measured range ρ, and reflectivity r values. For multi-echoes, only the closer detection is considered and missing points in the projected area, e.g. points without collision or absorbed by dark areas, are tagged as invalid. Fig. 2.3 shows a sample of the created front view map, and its associated ground truth.

Bird's Eye-View Data Representation

The bird's eye (zenithal) view $\mathbf{I}_{BE}$ is obtained by cropping the original point cloud in the volume $(x \in [3, 63], y \in [-25, 25], z \in [-2.1, 10])$, which maps a 2D ground area of 60×50 meters in front of the LiDAR sensor. This design decision was chosen after carefully observing that roughly 95% of the KITTI ground truth annotated vehicles are within these margins. Inspired by [35], to build this data representation we project the cropped point cloud to the ground floor into a 2D cell grid with a resolution of 0.1 meters. Thus, we obtain a bird's eye view $\mathbf{I}_{BE} \in \mathbb{R}^{H' \times W' \times C'}$, where

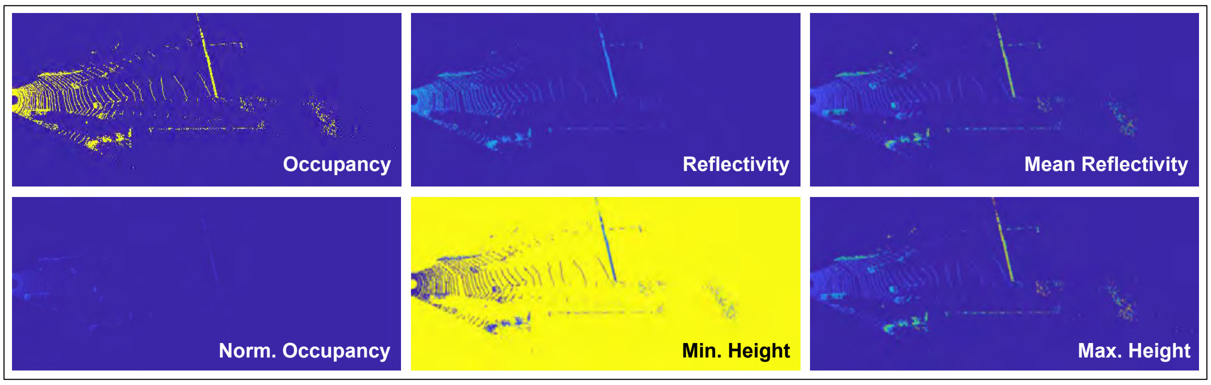


Figure 2.4: **LiDAR Bird's Eye view projection.** Our Bird's-Eye projection, $\mathbf{I}_{BE} \in \mathbb{R}^{H' \times W' \times C'}$, maps the points into a grid of size 10 centimeters and generates six channels encoding various occupancy, reflectivity, and height parameters. We restrict the an area to 60 meters in front of the vehicle and 25 meters to each side, according to the covered RGB area.

$H' = 600$, $W' = 500$, and $C' = 6$, accounting each cell for six different features: 1) a binary occupancy term with zero value if no points are projected onto that cell and one otherwise; 2) an absolute occupancy term, counting the number of points falling into that cell; 3) the mean reflectivity value of the set of points laying on the cell; and 4, 5, and 6) the mean, minimum and maximum height values calculated over the set of points projected onto the cell. We show a real sample of such six-dimensional feature map on Fig. 2.4.

Ground Truth Generation

We obtain ground truth vehicle masks for the front view $\mathbf{Y}_{FR}$ and the bird's eye view $\mathbf{Y}_{BE}$ by using the KITTI Tracking tracklets annotations, which provide real 3D-oriented bounding boxes in the camera frame. For that, we firstly transform these tracklets back to the LiDAR frame. Then, we annotate the 3D points that lay inside each vehicle bounding box labeling them accordingly, and apply a background label to the rest of the point cloud.

2.4.2 Network Architecture Design

To learn the set of parameters θ_{FR} and θ_{BE} that model the functions $\mathcal{F}_{FR}$ and $\mathcal{F}_{BE}$ to map the input point cloud projections to vehicle probability maps, we formulate the task as a deep learning per-point classification problem. Due to the fact that we manage here two classes, i.e., each *'pixel'* in the input projections $\mathbf{I}_{FR}$ and $\mathbf{I}_{BE}$ must be classified as either belonging or not to the vehicle class, our task can also be specifically considered as a binary semantic segmentation problem. With that purpose in mind, we set up a novel dual-branch scheme with two parallel CNNs, one devoted to the front view whereas the other to the bird's eye view. Results from both networks are later fused in a unique 3D point cloud to extract bounding boxes of each segmented entity, as can be observed in Fig. 2.1.

The learning problem is solved independently for both the front view and bird's eye view branches in a supervised manner with end to end back-propagation [62] guided by a class weighted cross entropy loss function [19], defined as:

$$\mathcal{L}^{WCE}(\mathbf{I}^n, \mathbf{Y}^n) = -\sum_{h,w,l}^{H,W,L} \omega(Y^n_{h,w}) Id_{[Y^n_{h,w}]} \log(\mathcal{F}(\mathbf{I}^n, \theta))_{h,w,l}, \tag{2.1}$$

where $\mathbf{I}^n, \mathbf{Y}^n$ are respectively the n-th training and ground truth maps and ω is a class imbalance weighting function computed from the training set statistics as the inverse ratio between the vehicle and background classes. As the output of the networks contains a channel with a probability predicted for each possible class encoded we use $Id_{[\cdot]}(\cdot)$ as an index function that selects only the predicted probability associated to the expected ground truth class. This pre-selection function makes the computation of the loss faster, accelerating the training.

For approximating the $\mathcal{F}_{FR}$ and $\mathcal{F}_{BE}$ functions, we introduce two contractive-expansive CNN architectures which allow for a good embedding of the vehicle features. In order to gather contextual information from the deeper layers and finer resolution features from the outer ones, we additionally introduce skip connections concatenating equivalent feature maps between both contractive and expansive parts of the network. These feature-flow shortcuts have already proved its effectiveness on other works [12, 15, 18]. In general lines, this technique improves the learning process by building better features but also by back-propagating gradients more directly from the deeper layers to the initial ones during training.

We further propose to solve the stated classification problems of each branch in a multi-scale manner by introducing intermediate loss functions at different resolutions on the networks. This technique guides the network faster to a correct solution by inserting valuable gradients at middle levels at the back-propagation step. Hence, for each branch we compute the final loss $\mathcal{L}_{Full}$ as

$$\mathcal{L}_{Full}(\mathbf{I}^n, \mathbf{Y}^n) = \sum_{m=1}^{M} \lambda_m \mathcal{L}^{WCE}(\mathbf{I}^n_m, \mathbf{Y}^n_m) \tag{2.2}$$

where m goes through the different resolutions in which the loss function is computed at this multi-scale architecture and λ_m is the regularization weight for the loss at such resolution. Next, we detail the specific architecture design choices performed for each input projection domain.

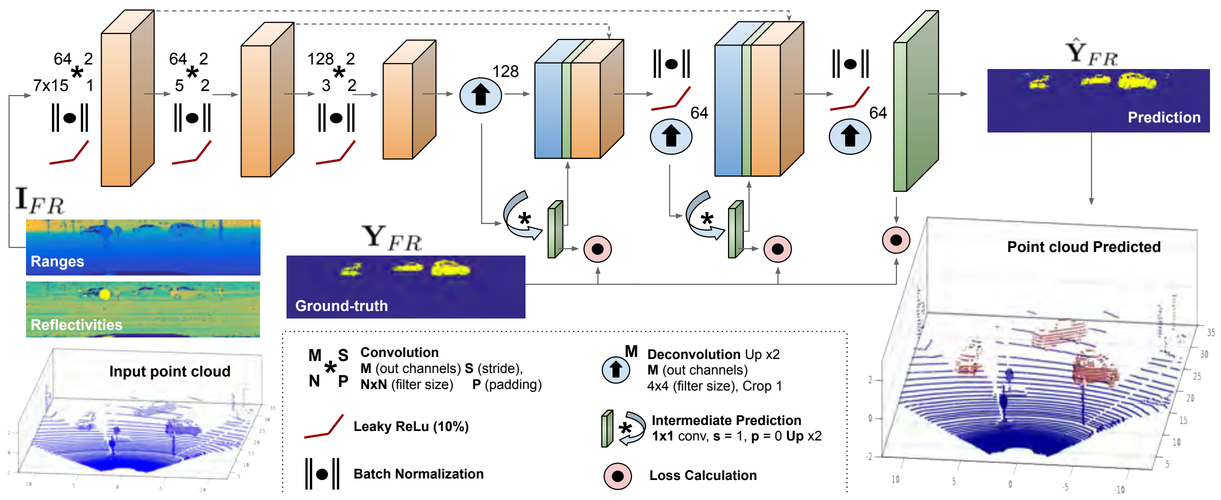


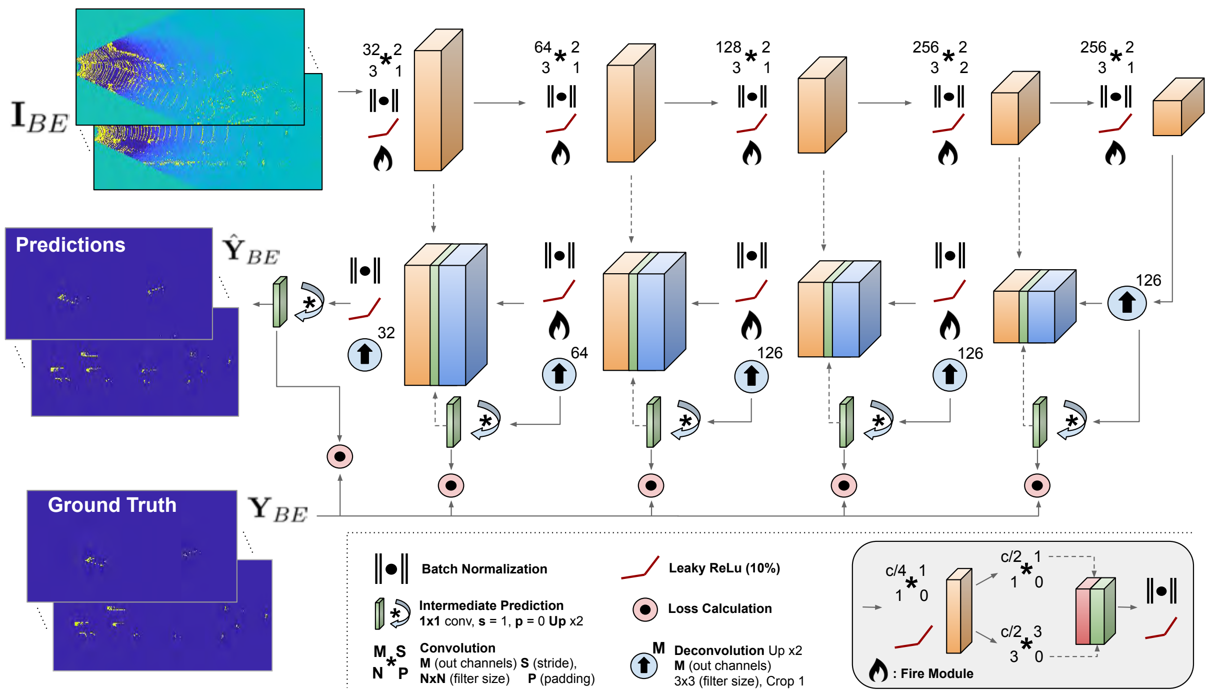
Figure 2.5: **Front View network.** Our front view network $\mathcal{F}_{FR}$ employs an encoder-decoder architecture with convolutional and deconvolutional blocks followed by batch normalization and ReLU non-linearities. The first three blocks generate rich features controlling, according to our vehicle detection objective, the size of the receptive fields and the feature maps generated. The next three deconvolutional blocks expand the information and concatenate feature maps from both parts of the network providing more gradient stability, which results in better learning. During training, three losses are computed at intermediate levels from low-resolution predictions, which are concatenated and propagated to obtain a more detailed final solution.

Front View Network

For the front view classification task, we employ the deconvolutional architecture shown in Fig. 2.5. Here we disclose some of the key insights of our design and architectural choices.

We carefully design the first layer of convolutional filters in order to adapt it to the proposed domain optimizing its performance. In this way, we impose an initial vertical vs horizontal stride ratio of 1:2, so that to obtain more tractable intermediate feature maps by reducing the size imbalance of $\mathbf{I}_{FR}$ (64 vs 448). Initial convolutional filter sizes are also designed according to the observed shape of the vehicles in this representation, so that to obtain a receptive field consistent with it. We set the size of these first convolutional filters to 7×15.

In this front representation of the LiDAR point cloud, the input $\mathbf{I}_{FR}$ has a small height ($H = 64$), which limits the number of convolutional blocks we can use in the encoder part of our network, as each of these reduces the resolution of the inputs by half. Thus, we decide to apply just three convolutional blocks, each one composed by convolution layers followed by Batch Normalization [63] and ReLU non-linearities. In this contractive stage, we obtain feature maps (tensors) of size $64 \times 224 \times 64$ after the first level, $32 \times 112 \times 64$ at the second level and $16 \times 56 \times 128$ at our inner step, which are displayed as orange volumes in Fig. 2.5.



$\mathbf{I}_{BE}$
Predictions
$\hat{\mathbf{Y}}_{BE}$
Ground Truth
$\mathbf{Y}_{BE}$
Batch Normalization
Leaky ReLu (10%)
Intermediate Prediction 1x1 conv, s = 1, p = 0 Up x2
Loss Calculation
Convolution M (out channels) S (stride), NxN (filter size) P (padding)
Deconvolution Up x2 M (out channels) 3x3 (filter size), Crop 1
: Fire Module

Figure 2.6: **Bird's Eye view network.** Our model $\mathcal{F}_{BE}$ is a refined encoder-decoder architecture. As the $\mathbf{I}_{BE}$ inputs are bigger (500×600) in this case, we apply five contractive and five expansive levels. To get richer features at each level, we insert customized fire modules that capture local and context information from the previous feature maps. These modules first reduce the number of feature channels and apply two parallel sets of convolutional filters on them to finally concatenate the results, obtaining local and context aware features. Intermediate losses are also computed in this network, merging partial predictions obtained at different resolutions.

On the expansive part of this front-view architecture we deploy three deconvolutional steps to recover the original input size 64×448. On each expansive step, we concatenate the corresponding tensors generated from the contractive part as shown in Fig. 2.5. Additionally, at each level we concatenate the low resolution predictions obtained while calculating the m multi-scale loss of Eq. 2.2, which results of great help to obtain refined predictions on upper levels.

Bird's Eye View Network

Having a cell resolution of 0.1 meters, a vehicle bounding box in $\mathbf{I}_{BE}$ occupies an average of just 800 cells from the existing 300000 on our 60×50 meters covered area. Although, the number of vehicles in each frame of the dataset varies greatly, averaging over the full training set we observed that only around the 10% of the existing cells are occupied by a vehicle. Detecting such small areas is still a challenging problem for deep neural networks, which made us to design a more complex specific architecture for this domain as shown in Fig. 2.6.

Ideally we would like our network to be accurate but small enough to process these bigger input frames fast in real time. To keep the number of network parameters θ_{BE} small while still providing high accuracy, we employ a customized version of the well established deep convolutional 'fire modules' [39, 64, 65] in our architecture. Our modified fire modules first apply a 1×1 convolution to reduce four times the number of feature maps from the input tensor, while maintaining its dimensions. Then two parallel branches apply convolutions with filter sizes of 1×1 and 3×3 obtaining different receptive fields. The results are then fused getting new robust features with local and context-aware information from the input feature maps, but using much less parameters. We insert fire modules after each resolution variation along the network pipeline, which accomplishes the strategy presented in [64] for downsampling late in the network, so that the convolution layers can retain large activation maps using fewer computations to process broad areas. A sketch of our customized fire modules is shown in the bottom-right area of Fig. 2.6.

2.5 Bounding Box Extraction

The above-mentioned architecture provides a powerful per-point vehicle class identification. In other words, we obtain realistic vehicle segmentations on each of the projections. Our next objective is to extract more abstract knowledge about the scene, obtaining 3D bounding boxes for the vehicles. By doing this, we are also able to obtain quantitative results over the KITTI Object Detection benchmark.

In this section we describe our proposed method to obtain bounding boxes from the deep learning-based vehicle segmentations. This method is summarized in four steps: 1) we fuse the vehicle segmentation results from the two deep neural network classifiers into a single annotated point cloud, 2) we employ a recursive clustering algorithm that uses an adaptive threshold to group points into individual vehicle hypotheses, 3) we extract initial bounding boxes from the resulting clusters, 4) we grow selected bounding boxes using contextual information around the cluster that assigns growing preference to occluded areas. We next show in more detail each of these steps.

2.5.1 3D Point Cloud Recovery from Both Branches

In this first step, we recover back a 3D segmented point cloud from each deep classifier. These can be appreciated in Fig. 2.1, as the blue and red point clouds for the front and bird's eye view branches respectively.

In the case of the front view network predictions $\hat{\mathbf{Y}}_{FR}$, we recover the 3D location of each classified vehicle point x, y and z, from the spherical coordinates represented by the front projections. We collect the azimuth ϕ and elevation θ angles respectively from the row and column indexes, and the range ρ from the value encoded in the representation. Notice that, since the spherical coordinates representation mimics the way that the sensor gathers the data, the original point cloud can be recovered from this projection with minimal distortion.

In the case of the bird's eye view, we recover Euclidean x and y coordinates from the centroid of the grid cells and generate 3 different point clouds attending to the bottom, middle and top heights encoded at each vehicle cell estimated in the map $\hat{\mathbf{Y}}_{BE}$.

Both recovered 3D point clouds are merged into a new cloud, preserving the originating view id on each point, as observed in the top-right area of Fig. 2.1. Given the use of a Softmax function in the final layer of the deep networks, it is safe to use a *'vehicleness'* probability threshold of 0.5, to select the cells to be considered as positively classified in the above-mentioned cases.

2.5.2 Recursive Euclidean Clustering

Due to the inherent sparsity of the LiDAR measurements, simple Euclidean clustering of the resulting segmented point clouds has the drawback of, depending on the chosen distance threshold one might end up over-clustering a single vehicle or on the contrary joining together several of them.

We propose a recursive clustering algorithm that gradually reduces the distance threshold for the clusters that exceed the maximum dimensions of a vehicle. This step ends when all the extracted clusters have a standard vehicle size or when the distance threshold reaches a minimum value, in which case, the cluster is discarded. Figs. 2.7a and 2.7b exemplify the importance of this step, as we can observe how a big cluster containing two vehicles is now partitioned. Once the final clusters are obtained, we apply an statistical outlier removal algorithm to assure that no outlier points will distort the shape of the object when fitting the final bounding box.

For each cluster, we also calculate a confidence measurement based on the fact that by definition both neural networks produce uncorrelated results, as each one receives its own input projection of the 3D point cloud. We can therefore assign a confidence value for each cluster by analyzing the ratio of contribution from each classifier on the final fused point cloud:

$$\eta = \frac{min\{Q_{BE}, Q_{FR}\}}{max\{Q_{BE}, Q_{FR}\}}, \quad \eta \in [0, 1] \tag{2.3}$$

where Q_{BE} and Q_{FR} indicate the number of points that the corresponding deep network branch identifies as vehicle in a particular cluster. For a fair comparison, Q_{FR} is obtained after re-projecting the front point cloud to the bird's eye view and the counted points are weighted by 3

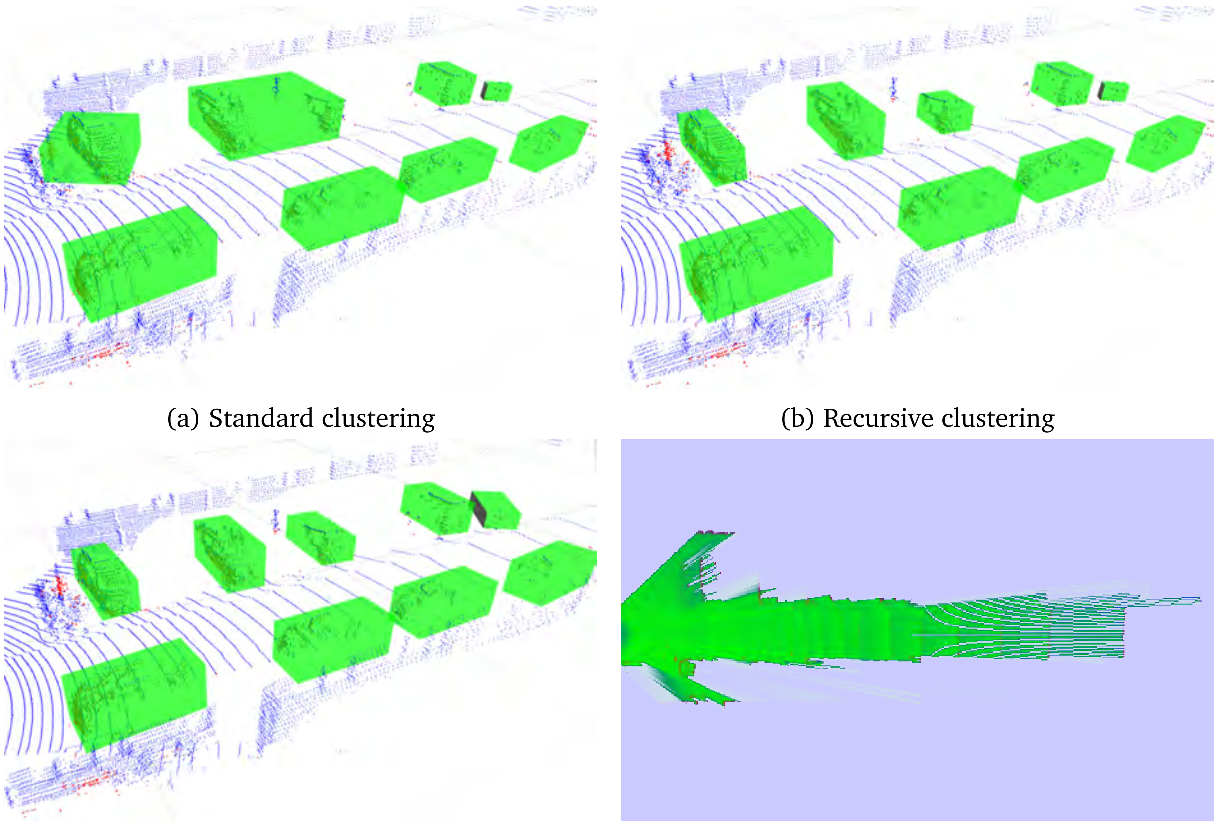

(a) Standard clustering
(b) Recursive clustering
(c) Bounding box growing
(d) Free space on probability map

Figure 2.7: **Bounding Box extraction method.** (a) Simple Euclidean clustering as used on [19]. (b) Improved results using the proposed recursive clustering with dynamic threshold applied on over-sized elements (notice the top vehicles with fitted boxes). (c) Bounding boxes grown on small elements up to standard vehicle sizes (notice the enlarged top-right boxes). (d) Obtained probability map of each cell for being free (green), occluded (blue) or occupied (red).

to compensate the three extracted points of each bird's eye cell. This η confidence measurement is high when a representative proportion of predictions from both classifiers exists in the same cluster.

2.5.3 Initial Bounding Box Extraction

To obtain quantitative results on the KITTI Object Detection benchmark, we extract initial bounding boxes for the segmented and clustered vehicles in the 3D point cloud, which will be later tightened to the vehicle shape.

The initial bounding box approximation is obtained by firstly projecting the 3D clustered points into the ground plane as shown in Fig. 2.8 b). We next localize the object perimeter by keeping for each azimuth angle the points with shorter range values, as can be seen in white

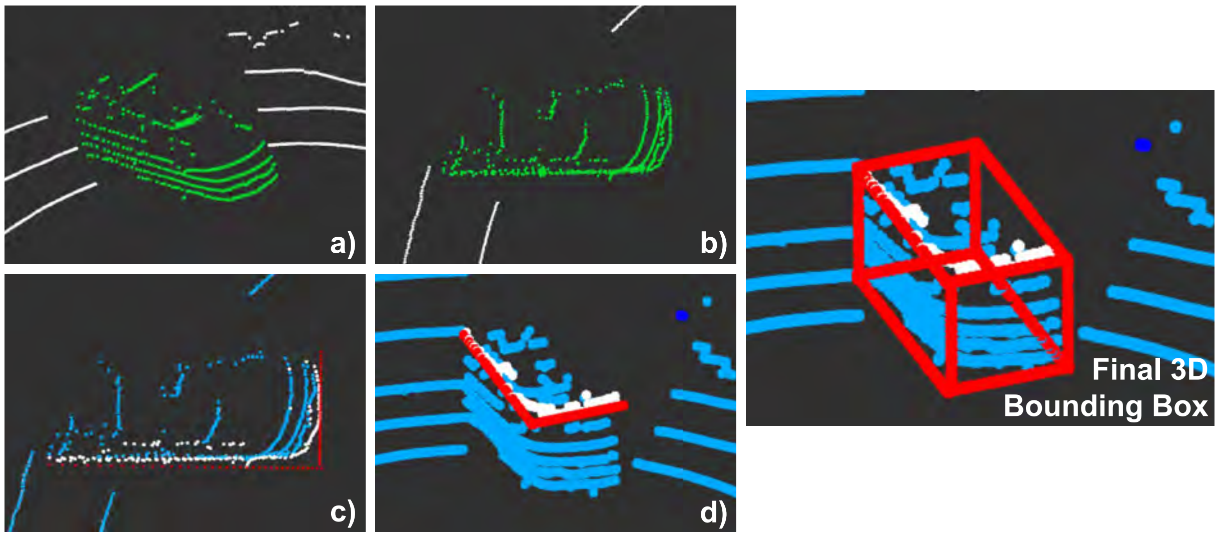


Figure 2.8: **Initial bounding box extraction pipeline.** a) shows the initial clustered vehicle, which is projected to the ground in b). We next extract in c) the perimeter (white points) and initial bounding box approximation in 2D (red points). Finally we the recover the 3D point cloud as shown in d) and obtain the full 3D bounding box.

color in Fig. 2.8 c). At this point, we store the cluster height and z coordinate enabling the 3D reconstruction to further compare our method against others. The best fitting bounding box for the ground-projected cluster, is obtained by performing an angular sweep of bounding boxes in the interval $[-\frac{\pi}{4}, \frac{\pi}{4}]$ over the cluster centroid. For each candidate we cast simulated 2D LiDAR rays to obtain the geometrically equivalent impact over the boxes, as shown in red color in Fig. 2.8 c). We therefore choose the bounding box with minimum fitting error ϵ, calculated as the mean square distance between the real points and the virtual ones. In the final step, we recover the 3D perspective as can be observed in Fig. 2.8 d) and we reconstruct the 3D bounding box with the help of the previously stored cluster height. The final result can be observed in the left-most image of Fig. 2.8.

2.5.4 Bounding Box Growing

We next seek to reshape this initial bounding box estimation to better enclose the vehicles. For this purpose, we design an efficient technique to grow vehicle bounding boxes by taking into account occlusion information.

This growing boxes algorithm is based on an extended occupancy map $\mathbf{S}_{BE} \in \mathbb{R}^{H' \times W'}$ in the bird's eye view domain, which registers the probabilities of a cell for being occupied, free, and occluded. In order to build it, we define some auxiliary variables, such as Q_h, which indicates the number of points in the fused point cloud that hit each cell. We also extract the variable Q_f

by counting for each cell the number of laser beams that pass above it, which is found by back-projecting each point in the point cloud back to the sensor center. For each cell, we additionally find z_m, which is the lowest height value of any laser beam passing through the cell or point living in it.

We compute the probability of a cell for being occluded as the ratio between the lowest observed point on that cell (z_m) to the ground and the expected vehicle height h:

$$p_o = (z_m - z_g)/h \;, \tag{2.4}$$

where z_g is the ground-floor height, known by calibration as the mounting point position of the the LiDAR sensor, or alternatively, the minimum point registered in a cell z_m given the case that this value is lower, so that accounting for slopes. The probabilities of a cell to be either occupied (p_h) or free (p_f) are computed from the extracted statistics as:

$$p_h = (1 - p_o) \cdot (Q_h/(Q_h + Q_f)) \;; \tag{2.5}$$

$$p_f = (1 - p_o) \cdot (Q_f/(Q_h + Q_f)) \;. \tag{2.6}$$

We want to guarantee that the vehicle bounding boxes do not grow along free areas, but do grow preferably along either occluded or occupied regions. To this end, we compute a bounding box cost C as the average p_f of all cells within the box. We start growing small boxes from the box corner closest to the sensor and iterate along the two facing sides of it. When the vehicle orientation is not well defined by the box dimensions we maintain two perpendicular box hypotheses and compute their respective costs C_a and C_b. The final growing process consists on expanding the box dimensions in small increments until its mean cost stops decreasing, until the maximum box dimensions are reached, or until a collision with another box is detected. We evaluate our confidence on each bounding box in terms of the relation between the costs of the individual hypotheses as follows

$$\nu_a = \frac{1}{2}(1 - C_a + C_b) \quad ; \quad \nu_b = \frac{1}{2}(1 - C_b + C_a) \;. \tag{2.7}$$

Once the growing process ends, we select the box with largest confidence value ν. We finally assign to each extracted bounding box a detection score S required to compute the average precision, which is obtained as:

$$S = \nu \cdot \eta \cdot (1 - \epsilon) \;, \tag{2.8}$$

where ϵ represents the bounding box fitting error defined in Section 2.5.3, η is the confidence value of the cluster as described in Section 2.5.2, and ν is the above-defined bounding box confidence.

2.6 Experiments

We evaluate the proposed system over the KITTI Vehicle Detection benchmark [6], showing results for three different configurations of our dual-branch deep learning classifier. These are: front view only, bird's eye view only and full approach using both classifiers. For each configuration we provide an ablation study and show the performance with and without including the bounding box growing algorithm. Additionally, we include a comparison to the published work [19] that uses only a front branch classifier, from which we have significantly improved the overall system performance as detailed along this chapter. We attribute this improvement to the confluence of two factors: a) the inclusion of the bird's eye view classifier and fusion approach that allow us to remarkably reduce the number of false positives and therefore boosting system performance. b) the full pipeline improvement including recursive clustering and bounding box growing, which decrease the number of under-and over-clustering situations, producing better and more accurate bounding boxes. Following, we detail our experimental environment:

Dataset. For learning purposes we use the training subset of the KITTI tracking dataset, which contains 21 sequences and a total of 8000 Velodyne HDL-64 annotated scans. For validation we use the first three driving sequences, that account for about 15% of the total vehicle-class points.

Network training. Both the front view and the bird's eye view deep neural networks are initialized with He's method [66]. We use Adam optimization [67] for training with the standard parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. In order to preserve the geometry properties of the driving scene we only augment the dataset by doing horizontal and vertical flips respectively in the front view and bird's eye view inputs with a 50% chance. No further augmentation is done. We train each network independently on a single Nvidia 1080Ti GPU for $400,000$ iterations with a batch size of 10. The learning rate is fixed to 10^{-3} during the first $150,000$ iterations after which, it is halved every $50,000$. We set the weighting factor ω to 25 for the front view and to 1000 for the bird's eye view classifier, according to the ratio between vehicle and background points on each representation. In both detector schemes, the multi-resolution loss regularizers λ_r are set to 1, assigning equal importance to each resolution.

Bounding Box Configuration. For the recursive clustering algorithm the initial Euclidean threshold is $1.0\ m$ and is gradually decreased in steps of $0.1\ m$ for over-sized clusters (more than $2.2\ m$ width or $5.0\ m$ length) up to a minimum value of $0.1\ m$. Clusters with less than 10 points or radius smaller than half a meter are discarded. Prior to computing the bounding box fit, a statistical outlier rejection algorithm is used to restrict possible false positive points.

Growing Bounding Box Parameters. Minimum width and length value for a grown bounding box are set to $1.6\ m$ and $3.4\ m$ respectively. Maximum length after the expansion is set to $3.8\ m$.

Table 2.1: Quantitative results for the Vehicle Detection Task - Validation Set (mAP %).

Grow.BB	Method	2D Front (mAP 2D)			2D BE-V (mAP BEV)			3D detection (mAP 3D)		
		Easy	Med.	Hard	Easy	Med.	Hard	Easy	Med.	Hard
✗	FR-V	20.13	18.69	17.63	42.31	29.67	30.35	02.32	02.37	01.67
	BE-V	26.26	24.53	23.08	64.86	49.02	41.48	07.68	08.51	07.68
	Fusion	51.28	42.99	39.07	65.15	50.80	42.52	12.22	13.19	11.45
✓	FR-V	47.41	40.67	40.44	69.37	60.41	52.24	13.44	13.38	12.87
	BE-V	52.35	49.22	45.00	77.75	60.97	61.33	15.84	18.61	19.67
	Fusion	72.11	61.41	54.59	79.73	62.29	62.37	22.45	25.36	24.74
✓	‘*Oracle*’ FR-V	54.08	44.54	45.09	79.96	71.26	62.61	28.52	27.97	28.33
	‘*Oracle*’ BE-V	51.26	42.47	43.14	89.33	70.98	71.16	27.39	26.97	24.24
	‘*Oracle*’ Fusion	70.85	58.37	52.06	89.10	71.48	71.51	42.11	40.40	35.57

2.6.1 Vehicle Detection Results

For analyzing the results, three variants of the classifier architecture are studied: a front view only (‘FR-V’); a bird’s eye view only (‘BE-V’); and the full fused architecture presented here (‘Fusion’). For each variant, we also evaluate the system with and without the growing bounding box module (‘Grow.BB’). Additionally, we include results of the vehicle detection task over bounding boxes obtained from a perfect per-point classification (‘*Oracle*’), which in fact acts as the upper bound for our learned segmentation problem.

Quantitative evaluations are done over three non-comparable domains: the front image view (‘2D Front’), performed on the bounding boxes projected over the KITTI RGB images; the zenithal view (‘2D BE-V’), which evaluates the boxes projected on the ground floor; and the ‘3D detections’, evaluating the full 3D boxes. For each domain, we follow the levels of difficulty defined in KITTI as: ‘Easy’, with fully visible vehicles having a minimum box height of 40 pixels in the image domain and a maximum truncation of 15%; ‘Moderate’, counting partly occluded vehicles with a min. box height of 25 pixels and a max. truncation of 30%; and ‘Hard’, including difficult to see vehicles with a min. box height of 25 pixels and a max. truncation of 50%;

In Table 2.1, we present a detailed evaluation of our vehicle detector over the proposed validation set. The measurement used is mean Average Precision (mAP) of the detections, also known as the area under the Precision/Recall curve. This mAP is calculated as the precision averaged for different recall values, considering positive a detection which intersection over union between the estimated bounding box and the ground truth is more than 0.7. At the light of the results, we can observe how the BE-V branch alone produces better results in comparison to the FR-V branch. This is mainly because the front view domain is noisier and the pixel classification task produces more false positives than in the BE-V, which has a more uniform distribution of the points in the scene. Furthermore, we can appreciate how our ‘Fusion’

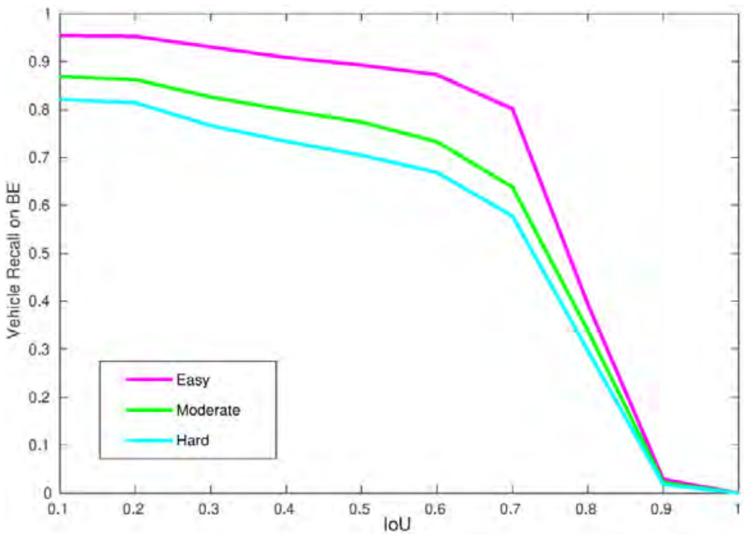


Figure 2.9: **Recall obtained by varying the Intersection over Union (IoU) threshold on steps of** 0.1**.** Setting an IoU of 0.5, we obtain approximately 80% of vehicle recall for moderate difficulty in the bird's eye view.

approach for combining both projections systematically boosts the results, obtaining consistent gains in all the three KITTI difficulty levels tested (i.e. 'Easy', 'Medium' and 'Hard') with respect to the stand-alone approaches. The inclusion of the growing bounding box module consistently helps on improving the results. This can be clearly observed in the 3D detection results, as the tight bounding boxes allow to gain around twice average precision.

After inspecting the small numerical difference of our Full system against the *'Oracle'* in Table 2.1, we can state that our approach successfully solves the learning problem for per-point classification. It is worth to remark that the KITTI evaluation on the '2D Front' domain is performed over the RGB images, and therefore projections of the resulting bounding boxes can introduce distortions that affect negatively the *'Oracle'* prediction. This explains the fact that in this evaluation our 'Fusion' system obtains slightly better results than the *'Oracle'* upper bound. It also states the vital importance of the inclusion of the BE-V branch and the fusion strategy, which helps on regularizing front projection errors as can be observed on the '2D BE-V' and '3D detection' evaluations. However, result differences between our fusion approach and the *'Oracle'* prediction on the 3D detection evaluation are more noticeable, suggesting a future work path.

The KITTI benchmark establishes as positive a detection with an Intersection over Union (IoU) of 0.7 over the ground truth. However, for autonomous driving environments, where a high recall is preferred, this can restrict the system performance. In this way, we additionally analyze the Average Precision (AP) of our method using different IoU thresholds in order to obtain better detection recall. As shown in Fig. 2.9, our detector is able to locate more than 80% of the moderate difficulty vehicles after setting an IoU of 0.5. To complete this experimentation,

Table 2.2: Car Detection Results on Validation Set with IoU 0.5 (AP).

Method	BEV detection (AP BEV)			3D detection (AP 3D)		
	Easy	Moder.	Hard	Easy	Moder.	Hard
Ours (FR-V)	81.29	80.62	71.85	46.17	47.38	43.01
Ours (BE-V)	90.61	81.25	72.30	55.92	52.92	52.90
Ours (FUSION)	90.31	81.42	72.40	71.73	62.19	54.97
MV(BV+FV) [35]	86.18	77.32	76.33	95.74	88.57	88.13
VeloFCN [34]	79.68	63.82	62.80	67.92	57.57	52.56
BirdNet [59]	90.43	71.45	71.34	88.92	67.56	68.59

Table 2.3: Vehicle Detection Task - Comparison over Online Testing Set (AP).

Method	2D Front (AP 2D)			2D BE-V (AP BEV)			3D detection (AP 3D)		
	Easy	Med.	Hard	Easy	Med.	Hard	Easy	Med.	Hard
RT3D [61]	49.96	39.71	41.47	54.68	42.10	44.05	23.49	21.27	19.81
BirdNet [59]	78.18	57.47	56.66	75.52	50.81	50.00	14.75	13.44	12.04
TopNet-HighRes [60]	59.77	48.87	43.15	67.53	53.71	46.54	15.29	12.58	12.25
Ours Fusion	59.32	48.76	43.19	45.12	37.10	32.88	15.16	14.49	12.94

in Table 2.2 we present quantitative results and compare our system with others in the state of the art providing this specific analysis on the available domains. Analyzing the results, we can observe that our method outperforms MV [35], VeloFCN [34] and BirdNet [59] in almost all the difficulties for the bird's eye view detection, and performs favorably with respect to VeloFCN for 3D detection. Notice here that the validation datasets of the related methods may differ, as there is not an established consensus subset for this on KITTI.

We further present quantitative results over the online KITTI Detection benchmark in Table 2.3, after verifying that there exist no correspondences between the Tracking training dataset used for learning and the Detection testing set. Some qualitative results from this evaluation are shown on Fig. 2.10. We can observe that our approach directly competes with other recent works based only LiDAR data. For the '2D Front' evaluation, we outperform [61] and are on par with [60]. However, we observe that there is still room for improvement on the '2D BE-V' evaluation, which may be a matter of increasing our cell resolution as [59] and [60] do. On the '3D detection' task it can be acknowledged that our method is well balanced due to the contributions of the dual-branch architecture. We obtain better results than [59] on the three difficulties, and than [60] on the moderate and hard ones. Comparing to [61] which directly focuses on predicting 3D bounding boxes, we obtain better results for the '2D Front' case only. On the other cases our performance is lower, which indicates a future avenue for improvement by additionally learning to solve the 3D bounding box prediction task along our pipeline.

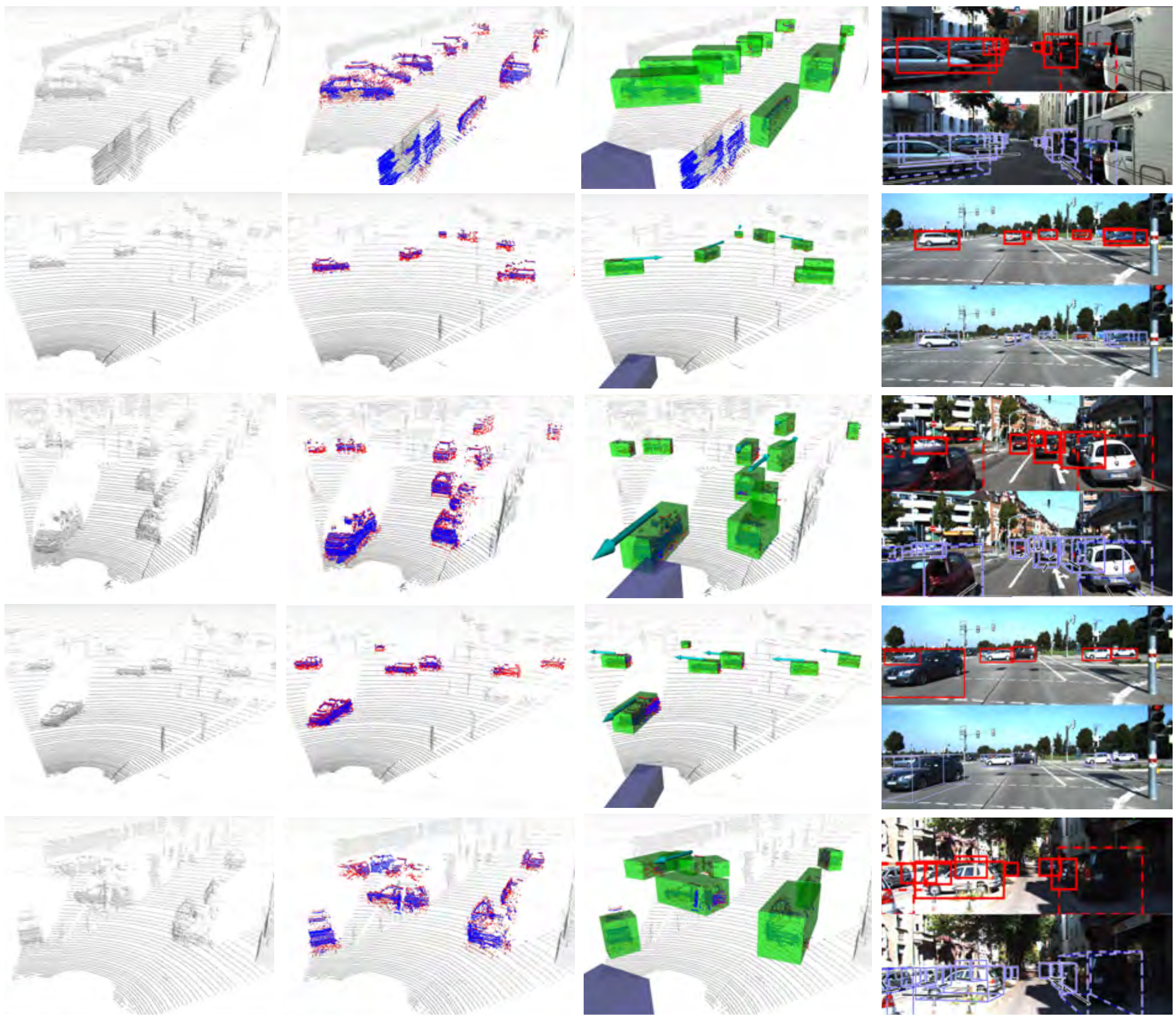


Figure 2.10: **Qualitative results of our system for vehicle detection.** All images are taken from the testing set of the KITTI benchmark, and were therefore not previously seen in the training step. In columns: 1) raw input point cloud; 2) fused output of the two deep-LiDAR classifiers where red and blue colored points represent segmentations for the BE-V and FR-V branches respectively; 3) bounding boxes extracted on the fused point cloud; and 4) bounding boxes projected over the RGB images, here just for these visualization purposes (dashed lines represent partially occluded vehicles). Notice that, despite the scarce information provided by the LiDAR sensor, our system is able to detect vehicles in complex urban environments, even when they are very close to each other (first row), or partially occluded (third and bottom row).

We perform a final experiment to gain insights about the limits on detection that our system can handle. Results are shown in Fig. 2.11, where we identify the number of ground truth vehicles in the validation dataset that are within any given distance to the sensor and total true positives identified by our system as a function of the distance to the sensor (continuous

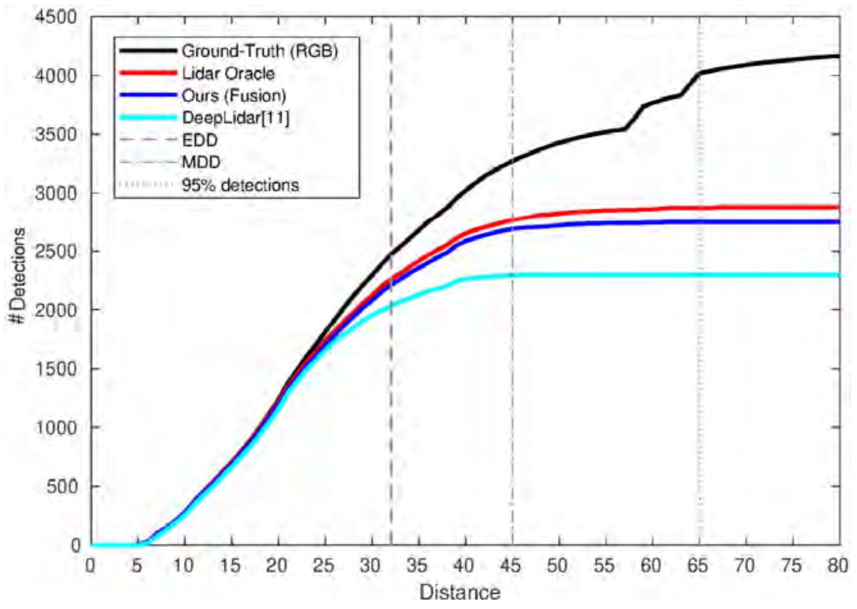


Figure 2.11: **Distance detection analysis.** Total number of true positive (TP) detections obtained at different maximum distances. We observe that: 1) Our system obtains almost human performance level in near field. 2) Comparing to [19], our new method boosts the performance to almost reach the *'Oracle'* level. 3) The bigger differences with the human annotators are produced by far vehicles, where the LiDAR point cloud becomes much more scarce.

lines in the plot). With this, we define two values for performance in terms of the distance: a) Effective Detection Distance (EDD), defined as the maximum distance at which the system recall $(TP/(TP+FN))$ is above 90%; b) Maximum Detection Distance (MDD), a less restrictive metric referring to the distance at which at least a third of the new appearing vehicles are correctly detected. To calculate it we set an incremental recall metric $(\Delta TP/(\Delta TP+\Delta FN))$ that computes the recall variation produced after increasing the distance threshold. The plot also shows the distance point at which 95% of the ground truth labels are visible.

Fig. 2.11 clearly shows that our system's performance heavily depends on the distance from the vehicle to the sensor, which is a direct effect of the LiDAR sparsity at further distances. We find our EDD at approximately 32 meters. Up to here the system performs close to the human level, indicated by the ground truth of annotated objects, represented with a black line in Fig. 2.11. Going further, at approximately 45 meters we find our MDD. Although at this point the number of False Negatives (FN) is increased, our system is still able to detect a significant part of the distant vehicles. We can clearly see here the benefits of our multi-view approach, as we almost match the results obtained with the *'Oracle'* detector. Going beyond the MDD we find at around 65 meters the point at which the 95% of the manual annotations have already appeared. However, it can be appreciated how the capabilities of our LiDAR only system are limited at these distances, in which laser observations become more sparse with the distance.

2.7 Conclusions and Future Work

In this chapter we presented a system to detect vehicles using only 3D LiDAR data. We initially devise two different projection of the input LiDAR point cloud, a frontal and a bird's eye one. We train two different deep neural networks to perform per-point classification tasks in order to segment vehicles respectively from each of the created projections. The generated point-wise predictions are then fused in a new 3D point cloud and a recursive clustering algorithm is applied. Bounding boxes are then grown using contextual information at vehicle level.

The system is thoroughly evaluated in the challenging KITTI Detection benchmark, and is compared against related approaches that only use front LiDAR information. The results show that the inclusion of our deep-classifiers and fusion method drastically increases the system performance, as we outperform or match other LiDAR-based approaches. These results confirm the hypothesis that the point-level false positives are to some extent uncorrelated between networks, and that the coincidence of different networks identifying the same cluster as a vehicle, constitutes a strong evidence of its real existence. In addition, the presented approach shows an outstanding performance in the near field ($\approx 32\ m$) on pair with the human based annotation, where the results of the fusion detector are comparable to the ones obtained with an ideal segmentation.

We leave for future research to explore the multi-class problem, detecting also pedestrians and cyclists, as well as to tackle the use of deep learning techniques for the bounding box extraction attending explicitly to context information. Other possible research paths is to investigate effective manners for end-to-end learn to solve the 3D bounding box detection task.

3 Use Cases of Single Frame LiDAR Detection

In this chapter, we demonstrate the usefulness of our dual-branch deep learning approach for detecting vehicles in driving environments using LiDAR information. In this way, we present two different real use cases, the first one related to the localization and mapping task [21], whereas the second one focused on tracking [20]. These tasks are essential components of autonomous vehicles and intelligent transportation systems, as they enable the accomplishment of higher level functions such as path planing, safety navigation or obstacle avoidance. Moreover, to precisely estimate both ours and others positions in a map also allows obtaining further environmental information such as traffic state/accidents, road closures/works, etc., which would, in turn, facilitate the eventual completion of the predefined mission. The same idea holds in the opposite direction, in which a correctly located vehicle may augment map meta-information with its current observations of the environment.

In Section 3.1 we prove that by filtering the segmented vehicle points and other potential movable objects from each LiDAR frame we are able to generate longer-lasting maps with common odometry and mapping algorithms, from which we can localize more accurately in subsequent days. We demonstrate this use case on a newly recorded dataset on a highly dynamic scenario such as a supermarket parking lot, enabling accurate localization over a standard map generated with a common SLAM method days before.

In Section 3.2, we show how the proposed method can produce high quality detections so as to be used on a vehicle tracking application obtaining promising results over the KITTI Tracking benchmark. Moreover, an alternative bird's eye view evaluation of the tracker performance is also introduced. This evaluation is more informative than one used on the KITTI website, based on the RGB image domain.

3.1 Improving Map Re-localization with Deep ‘Movable’ Objects Segmentation on 3D LiDAR Point Clouds

Nowadays, vehicle position can be easily obtained by different Global Navigation Satellite Systems (GNSS) such as GPS, Galileo, GLONASS, etc. Although these systems can provide good results, they have limited precision in urban scenarios with buildings and other elements that may block the satellite signals. Other accurate approaches like beacon-based methods exist, but require prior installation of external infrastructures and thus are not ready for general usage. Nevertheless, for autonomous vehicles it is important to additionally include localization systems based on their own perceptive sensors, such cameras or LiDARs.

Simultaneous Localization and Mapping (SLAM) has gained utmost attention within in-vehicle localization algorithms. However, in very dynamic and cluttered urban environments where vehicles and other elements are constantly moving or can potentially do, SLAM algorithms encounter difficulties on finding static and stable features that help the ego-localization process as well as would allow to re-use the generated map on subsequent days. Thus, in many applications where the goal is just to travel predefined routes in a known area, SLAM systems may introduce unnecessary and redundant computations creating a new map each time instead of just providing the desired localization within an existing map.

3.1.1 Objective

In this use-case we propose the creation of longer-lasting representations and 3D maps from 3D LiDAR scans, as sketched in Fig. 3.1. We use the deep learning-based architecture presented in Chapter 2 to find and segment out potential *‘movable’* objects from the scene. Our hypothesis is that by avoiding from the beginning the inclusion of possible outliers and known-dynamic elements, we allow the creation of better 3D maps with standard SLAM techniques, providing faster and more accurate re-localization and trajectory estimation on subsequent days.

We show the effectiveness of our approach in a very dynamic and cluttered scenario such as a supermarket parking lot, for which we build 3D maps with and without our segmentation applied and use them to assist our localization at different times and days. Results show that we are able to accurately re-localize over a filtered map reducing consistently trajectory errors an average of 35.1% with respect to a not filtered version and of 47.9% with respect to a standalone map created on the current session. Summarizing, the main contributions of this use-case are:

- We introduce a simple yet effective re-localization approach for odometry and mapping methods based on feature matching against a previously generated map, extending the life of those maps that otherwise will not last more than the current session.

- We train the deep convolutional dual-view architecture of Chapter 2 to segment possible moving elements from a driving scenario, such as vehicles, cyclists or pedestrians.
- We show that by eliminating from a driving scene the elements that can move while building a map, localization for subsequent days improves against the pre-built map improve.
- We perform real experiments in a parking lot recorded over several days and trajectories, obtaining consistent results that support our approach. We also demonstrate how our method is useful for building maps in a multi-agent manner or through different days.

3.1.2 Related Work

Localization and Mapping with LiDAR is a very active research topic in the robotics and automotive community. Early approaches used 2D laser data and ICP methods to correct the ego-motion distortion. When employing 3D LiDARs with further amounts of information, more sophisticated schemes need to be considered, like including other sensors' information such as IMU, wheel encoders or GPS/INS using for example extended Kalman filters [68, 69]. Other methods take motivation from visual SLAM algorithms [70] and use the laser reflectivity measures to create intensity images from which they extract and match distinctive features between frames in order to infer the ego-motion [71, 72]. However, in these algorithms based on matching visual or geometric features the localization and trajectory estimation is commonly recovered in a batch optimization post-process [73], which make them unsuitable for real time localization.

The introduction of the LOAM algorithm [74] resulted in a great advance in terms of accuracy and real-time performance, achieving top position on the KITTI Odometry benchmark. It proposes to divide the complex SLAM problem in two algorithms, where one estimates odometry at a high frequency but low fidelity whereas the other runs at lower frequencies for fine matching and registration of the point cloud. From this approach, incremental improvements have been recently proposed. LeGO-LOAM [75], deploys a lightweight, real-time pose estimation method that has into account the presence of the ground plane in its segmentation and optimization steps to obtain distinctive planar and edge features for the later optimization method. We will employ this approach to build our initial maps as it provides a real-time and accurate representation as well as is more robust using unprocessed raw LiDAR data.

Other contemporary works propose to infer the vehicle position by matching segments from the point cloud that may belong to partial or full objects as well as sections of larger structures [76]. In this way they obtain a good balance between local descriptors, which can suffer from ambiguity without context information, and global features, which are viewpoint dependent. This method extracts several features from the segmented clusters and tries to match those segments over the ones existing in a map for further localization purposes. Further incremental

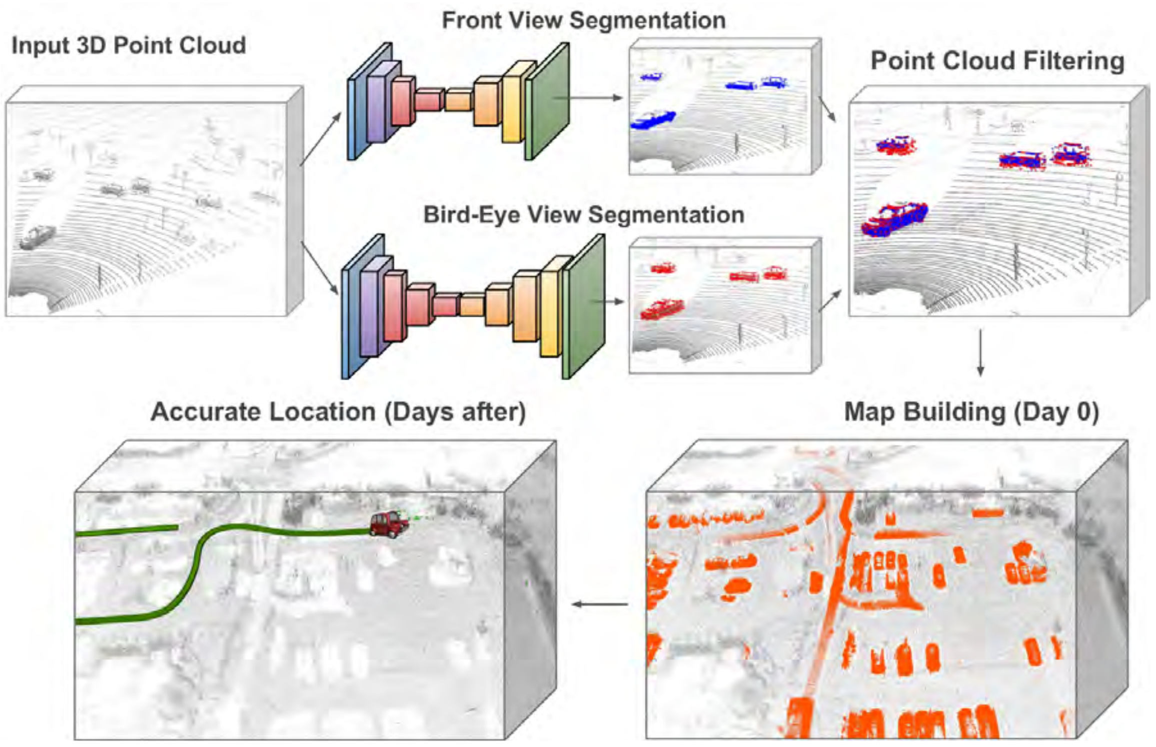


Figure 3.1: **Map re-localization approach segmenting movable objects.** We propose to segment *'movable'* objects from 3D LiDAR point clouds to build longer lasting maps that can assist on trajectory estimation and localization for subsequent days. For that we employ the dual-branch deep architecture from Chapter 2 and filter out the obtained segmentations, retaining mostly static elements on the scene. We are therefore able to accurately estimate our position and trajectory on subsequent days by additionally re-localizing on the map.

approaches employ a data driven feature encoder to extract compact and discriminative features from the segments [77]. These features can be used to create a compact map representation due to its high reconstruction capacities and for accurate location over an existing map, as long as it does not contain much movable elements.

Yet, the ability to create longer standing maps overcoming challenging issues such as the management of dynamic elements or allowing map scalability and updatability, is still a pending task for SLAM algorithms [78, 79]. Aiming to alleviate these challenges, we directly segment the input information with the deep learning pipeline proposed in Chapter 2 prior to build any map. Thus, we avoid from the beginning the inclusion of possible outliers and known-dynamic elements, so therefore do not give chances to further extract any feature from them. By doing so, we demonstrate in Section 3.1.4 that more accurate localizations are possible over a map created days before from a constantly changing environment.

3.1.3 Approach

We here detail our approach for accurate re-localization at different days over a map built using standard LeGO-LOAM algorithm in a cluttered scenario. Our system can also be employed to build a full map over several days or in a multi agent way as we will show also later.

- **Movable Objects Segmentation**

For this task we slightly modify the dataset created in Chapter 2. Our objective here is to classify each of the 3D LiDAR points as belonging to a *'movable'*, or *'non-movable'* class. As *'movable'* we consider all the KITTI [6] annotated classes, i.e. Vehicle, Van, Truck, Cyclist, Pedestrian, Tram and Misc. We follow the method of Chapter 2 to generate the ground truth for the front $\mathbf{Y}_{FR}$ and bird's eye $\mathbf{Y}_{BE}$ projections and use the same train, validation and test splits.

Filtering Movable Objects. After the new system is trained, we filter out the predicted *'movable'* points in the 3D input LiDAR point cloud. For that, we first transform the segmented points back in the 3D Euclidean space and cluster them. We discard clusters with less than 50 points as well as attending to the mean probability predicted, where we weight the contribution of the bird's eye samples to be twice as the front view ones [21]. Once we obtain the resulting clusters, we filter the original input LiDAR data by eliminating all points in a radio of 10 centimeters, so that restricting the possible adverse effect of projection errors. We do not need to perform any bounding box extraction in this use-case, as our main objective is just to filter the input point cloud in the faster way, without caring much about the tightness of boxes.

- **Vehicle Localization**

To locate in an existing map our system uses just LiDAR data. We can additionally use prior information from a GPS (available as standard on-vehicle equipment) which, although is imprecise and has a low refresh rate, can help us to estimate a coarse initial position that will be afterwards refined using our re-localization algorithm. In the rest of this section, we first show the creation of a ground truth map at day zero against which we aim to locate on the subsequent days. Then we detail our localization approach, which consist on obtaining a coarse initial guess followed by the final accurate localization.

Ground Truth Map Building. We build the initial ground truth map from LiDAR scans synchronized to a Differential GPS (DGPS) that are fed to the state of the art LeGO-LOAM algorithm [75]. Our aim is to accurately re-localize ourselves in this map on subsequent days. LeGO-LOAM extracts edge and surface features from the generated map by analyzing the local surface properties of certain areas in the point cloud. Edge features are extracted from rough local regions, whereas surface ones from smooth surfaces. In order to obtain more distinctive features from the map, the LeGO-LOAM algorithm does not account for features within a

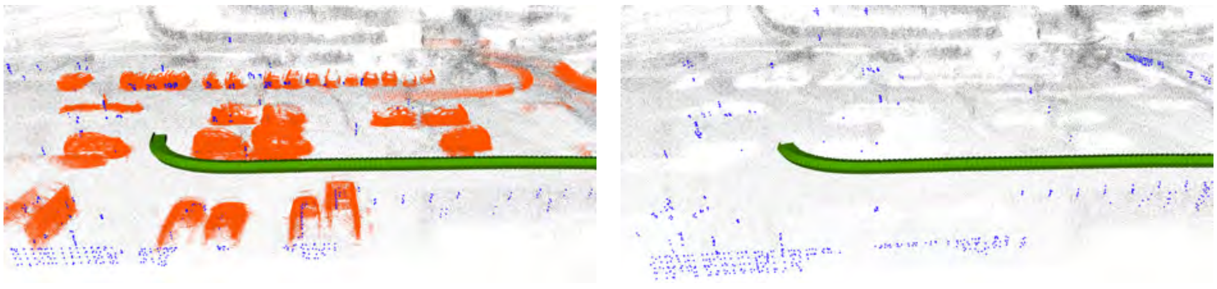

(a) Features extracted from full point cloud (b) Features extracted from filtered point cloud

Figure 3.2: **Comparison of extracted features (blue) on point clouds with and without filtering movable objects.** By providing an unfiltered point cloud the feature extraction mechanism selects a vast amount of points from dynamic objects, as seen on the top area vehicles. This can be compensated by filtering the movable objects with our deep-segmentation.

minimum distance from some of them considered as strong. However, this fact can suppose a big disadvantage; movable objects like vehicles having a very prominent surface structure are more likely to be chosen as strong features over other relevant static features of the scene.

By pre-filtering the raw point cloud we therefore enforce the selection of strong features just from distinctive static elements instead of from movable objects, thus allowing better inter-day re-localization. A comparison of the extracted features from the full and the filtered point cloud respectively can be observed in Fig. 3.2, where we can clearly appreciate how the blue features of the top image area vehicles, are not present in our filtered map. Additionally, we also remove the ground-floor features determined by LeGO-LOAM to obtain a more compact and distinctive representation of the features map. In this way, our resulting ground truth map (GT-Map) consists of LeGO-LOAM features extracted from each filtered frame along with the own frame transformation from the DGPS.

Initial Pose Estimation. To initially accelerate the current localization process over a previously existing map, we use the standard on-vehicle GPS. As GPS solely provides coarse information about the position, we additionally need to estimate the initial orientation over the ground truth map. For that, we extract a subset of the map around the GPS coordinates and perform feature matching from our current observed frame using Iterative Closest Point (ICP). Feature points used for the current frame are extracted and selected in the same way as described in the map building process. To optimize the ICP step, we firstly perform a coarse matching prediction by applying different rotations of the current frame features on steps of 45 degrees. For each rotation, we get a fitting score, which describes the remaining sum of squared differences from the feature points of the current frame to their corresponding nearest neighbors in the ground truth map. Then, the initial orientation guess is selected as the one with a fitting score lower than 0.4.

Finally, we express the transformation to our initial pose estimation as $T_{init} = T_{GPS} \cdot T_{ICP} \cdot T_{Rot}$; all parametrized as homogeneous transformation matrices where T_{GPS} would be initial rough position estimate given by the commercial GPS, T_{Rot} is the best fitting initial rotation found for the ICP, and T_{ICP} refers to the final refined transformation obtained by the ICP algorithm that best optimizes the feature matching. Notice here that T_{GPS} and T_{Rot} are just used to speed up the matching process, and that any other prior coarse pose estimation could be employed.

Continuous Re-Localization. Once the initial pose estimation is performed, we again use LeGO-LOAM to continuously calculate further pose transformations based on our segmented LiDAR scans. At the same time, we perform re-localization, trying to match our current position against the pre-existing generated map (GT-Map). For these continuous re-localization steps, we follow a similar process than above and employ the extracted features from the current scan with ICP to correct possible drifts caused by the current trajectory obtained with the LeGO-LOAM algorithm.

In comparison to the initial pose estimation, these re-localization transformations are simpler to calculate, as they only depend on the current estimated position and the correction given by the ICP over the GT-Map. Therefore, $T_{reloc} = T_{ICP} \cdot T_c$, where T_c stands for the current pose estimated. The threshold for the ICP fitting score in the course of this re-localization step is lowered to 0.3 in order to estimate the transformation more robustly.

3.1.4 Experiments

We show through several experiments the effectiveness of the proposed application for the deep-LiDAR segmentation from Chapter 2. In order to validate our proposal, we have recorded 7 different sequences of a cluttered and dynamic urban environment, i.e. a supermarket parking lot, during diverse days and hours. We first detail the data acquisition process of the new localization dataset. Next, we show our re-localization capacities in this highly unsteady scenario using as GT-Map a map built with LeGO-LOAM over a scene at a different day. Our hypothesis is that this GT-Map would be useless without our approach, as it will not last more than the session for which it was created. Finally, we show an additional application of our use-case to build a map through different days, which can also be extrapolated to a multi-agent map building task.

- **Data Acquisition**

This new dataset was captured and recorded for our purposes with a Test used by Valeo, a worldwide automotive supplier, which is equipped with multiple sensors. The relevant hardware concerning our experiments is a Velodyne LiDAR HDL-64E S3, a differential GPS by IMAR and the serial production car GPS. The 7 sequences were recorded on a parking lot of a local shopping mall in Kronach (Germany) at different days and times to ensure diverse constellation of the parking vehicles. Sequence 1 was recorded early in the morning aiming to obtain an almost empty parking lot. Sequence 2 and 3 were recorded at different hours on another day with the area being slightly crowded. Sequence 4 to 7 were recorded on different hours of another day with a very crowded parking lot. Our dataset therefore captures diverse constitution of the parking lot at three different days.

- **Re-localization in Dynamic Environments**

To validate our method we built two GT-Maps as described in 3.1.3 using sequences 1 (GT-Map 1) and 3 (GT-Map 2). In this regard sequences 2 to 7 are used for re-localization over GT-Map 1 and sequences 1 and 4 to 7 over GT-Map 2, ensuring inter-day experiments with different environmental constitution. Moreover, for each sequence we build two GT-Map variants, a *'Full'* one including all objects, and a *'Filtered'* one with removed *movable* elements.

In our experimentation, we apply three different methods:

- **'LeGO-LOAM':** in which subsequent frames are processed in an unfiltered map solely based on the pose estimation calculated by LeGO-LOAM.

- **'Reloc. Full':** in which we perform in parallel pose estimation by LeGO-LOAM and re-localization over an unfiltered map using the full unfiltered current frames.

- **'Ours (Reloc. Filtered)':** in which we perform in parallel pose estimation by LeGO-LOAM and re-localization over a pre-filtered map using the filtered current frames.

For each method, performances in consideration of localization in a dynamic pre-build map are validated based on three different metrics:

- **First Re-localization** ($1st_r$): defined as the frame number when the initial pose estimation happened, as detailed in Section 3.1.3.

- **Number of Re-localizations** ($\#_r$): number of frames at which the particular algorithm was able to relocalize, as shown at the end of Section 3.1.3.

- **Mean Absolute Error** (MAE): in meters, the averaged absolute error of the estimated positions over the whole sequence compared to ground truth poses of the DGPS.

Table 3.1 shows the quantitative re-localization performances of the considered methods and sequences applied on the respective maps. Additionally, we show in Fig. 3.3 a detail of the absolute error obtained with the three algorithms for re-localization along sequence 3 using the GT-Map from sequence 1 (almost empty parking map) in comparison to the ground truth data.

At the light of the results we can observe that filtering *'movable'* objects from point clouds greatly improves the performance of re-localization in dynamic environments. Compared to

Table 3.1: Re-Localization comparing LeGO-LOAM, Reloc. Full and Ours (Reloc. Filtered)

GT-Map	Curr. Seq	LeGO-LOAM		Reloc. Full			Ours (Reloc. Filtered)			MAE Improvements (%)	
		$1st_r$	MAE (m)	$\#_r$	$1st_r$	MAE (m)	$\#_r$	$1st_r$	MAE (m)	To Reloc.Full	To LeGO-LOAM
1	2	3	1.73	7	3	1.29	288	8	0.84	34.42 %	51.07 %
	3	2	2.93	20	2	2.38	257	2	0.99	58.30 %	66.13 %
	4	197	2.09	0	275	6.38	108	8	1.08	83.05 %	48.23 %
	5	273	3.65	0	274	3.74	78	233	1.98	47.03 %	45.76 %
	6	567	5.94	0	279	4.14	5	281	3.91	5.56 %	34.25 %
	7	33	3.59	0	27	4.05	49	26	1.01	75.05 %	71.84 %
2	1	2	2.80	503	2	0.74	548	2	0.70	5.02 %	75.03 %
	4	67	2.26	14	66	1.88	101	56	1.73	7.91 %	23.40 %
	5	273	2.78	54	289	2.73	112	269	2.12	22.32 %	24.32 %
	6	168	4.73	9	168	4.61	26	165	2.39	48.29 %	49.51 %
	7	1	1.93	53	1	1.20	135	1	1.21	-0.79 %	37.37 %

the unfiltered re-localization ('Reloc. Full', in Table 3.1) we reduced the Mean Absolute Error (MAE) value a 16.5% using Map 1 and up to 50.57% using Map 2, averaging over the respective sequences. These results are more remarkable using Map 2 as base due to the existence of more dynamic elements in this sequence than in Map 1, which was recorded with the parking lot almost empty. Additionally our approach consistently scores higher in number of localized frames and mostly gets faster initial localization compared to the unfiltered approaches.

Since GPS data is used solely for localization until the initial pose estimation is obtained, there are bigger MAE values when this initial localization takes place late. The consequences of this effect can be observed in Table 3.1 when re-localizing at sequences 5 or 6 over both maps. Observe how over these sequences the first re-localization occurs rather late, and therefore the MAE error is higher. This impact is mainly caused by partially nonexistent sequence and map overlap so therefore re-localization cannot be applied. The opposite effect can also be noticed in sequence 7 applied over Map 2, where unfiltered re-localization ranks higher in MAE than our approach. In this occasion, similar constellations of vehicles were present in the parking, so for the two baseline algorithms the first re-localization was performed fast.

Another factor affecting the results are partially erroneous re-localizations. These essentially happen in curves where a slight deviation in the orientation estimation has a huge impact on the subsequent trajectory calculation. Assuming no further re-localizations occur after a wrong one, there will be a huge drift in the ensuing poses which is reflected in the results of the unfiltered re-localization whose MAE is partially exceeding the ones of the standard LOAM approach. Comparing to the results of our approach this perturbation can be mostly eliminated, since in the filtered environment, more distinctive static features are selected to guarantee more robust pose estimations.

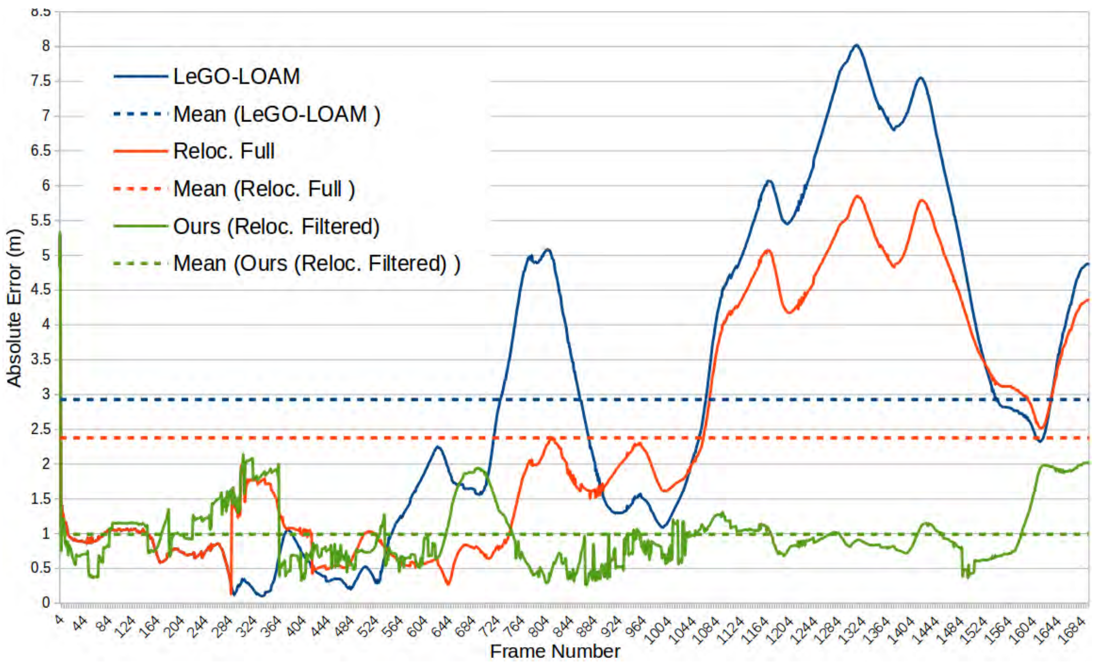
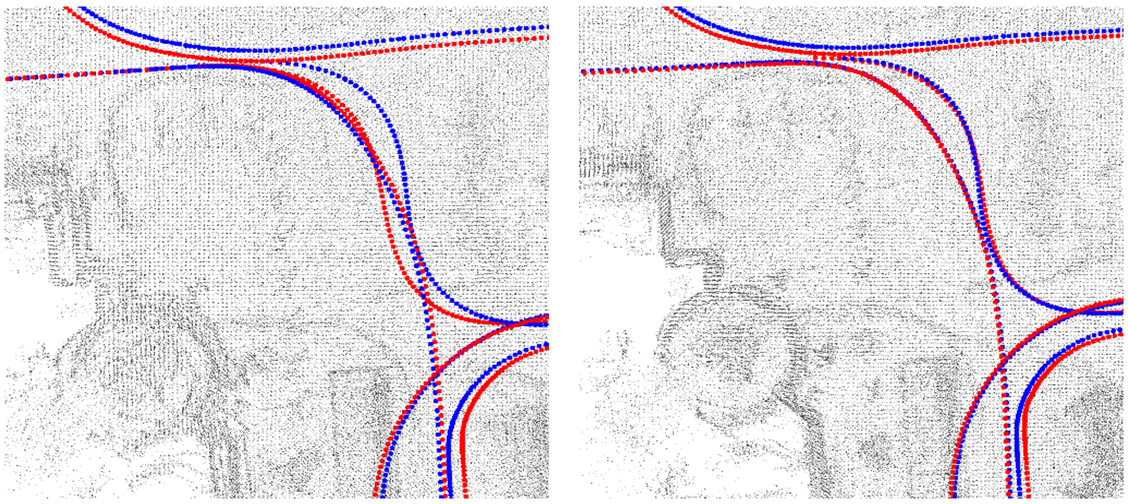


Figure 3.3: **Error progression.** Progression of the absolute error while re-localizing ourselves during Sequence 3 in GT-Map 1 as base for the three considered methods in comparison to the ground truth data. Each corresponding Mean Absolute Error (MAE) is shown with dotted lines.

- **Multi-Day Map Extension**

Apart from experiments on re-localization in dynamic environments we can prove the adaptability of the proposed algorithm to the application of a mapping process during several days. Here we are able to show that filtering movable objects from the processed data drastically improves the ability to build correspondences between maps from different days and consequently, the quality of the final map. In our experiments we choose sections of sequences from three different days which are partially overlapping to compose a final map of the entire parking lot.

Starting with a segment of sequence 1 we are building the map solely using the LeGO-LOAM algorithm with loop closure to accomplish a detailed mapping result. Next we are processing a section of sequence 3 and do re-localization in the previous built map based on the extracted features. In contrast to the previous experiments where we were re-localizing at every frame independently, here we are using the found correspondences over the previous map to do a graph optimization based on the inbuilt loop closure method of LeGO-LOAM in order to continuously update and refine the complete map. The previous step is repeated with a section of sequence 7 applied on the currently created map, so the final outcome is a merged map built upon sequences of three different days. This step could be further repeated to build larger maps, although we show here the proof of concept.



(a) Drift using 'Reloc. Full' (b) Results with 'Reloc. Filtered'

Figure 3.4: **Qualitative results.** Comparing 'Reloc. Full' (a) and 'Reloc. Filtered' (b) on the multi-day map assembling task. We display in red the estimated trajectories along the days and in blue the ground truth. For 'Reloc. Full' the map shows blurry and doubled contents caused by drifts due to incorrect feature matching, which are clearly compensated by our approach.

In this context we can prove the strength of our approach which, by filtering out movable objects, is able to establish more robust connections to previous days maps and more often, therefore obtaining a cleaner and more accurate final representation. A comparison of the quality of the resulting maps, with and without including movable objects can be observed in Fig. 3.4. Looking at the unfiltered map more blurry regions and doubled elements can be observed which are on the other hand compensated in our solution.

To get a quantitative measurement of these experiments we compare the composed trajectories of the map extension to the corresponding ground truth data. Here we use ICP to align the individual trajectories to the ground truth. In these experiments it is more important to obtain correct transformations in between the respective partial sequences rather than achieving a minimum pose-to-pose distance at every time step from the start, as was done in the previous experiment. Therefore, as a validation metric we use the fitness score, defined as the remaining sum of squared distances after aligning the full composed trajectories to the ground truth. In this regard, we observe that by filtering out movable objects we are able to decrease the trajectory fitting score from 0.26 to 0.13, which represents an improvement of 50% compared to the unfiltered approach.

3.2 Vehicle Tracking

In this use-case, our objective is to track through time the bounding boxes generated as specified on Chapter 2 using a number of multi-hypothesis extended Kalman filters, one per each tracked vehicle.

For this tracking task, we based ourselves in the previous work presented in [19], where we used a 2D multi-hypothesis extended Kalman filter approach. However, we include additional improvements to the track state. For each cluster obtained in Section 2.5.2 we make use of the defined confidence term η specified in Eq. 2.3 to decide whether new tracks should be initialized or not. Additionally, we use the orientation confidence term ν specified in Eq. 2.7 to initialize the vehicle track orientation when no evidence of motion exist. In this way, the final bounding box extraction process of Section 2.5.4 generates an initial observation for the track containing: the box pose $\boldsymbol{\pi} = (x, y, z, \theta)$; its dimensions $\mathbf{d} = (w, l, h)$; its fitting error ϵ defined in Section 2.5.3; its detection confidence η of the primal cluster as defined in Section 2.5.2; and its orientation confidence term ν defined in Section 2.5.4.

Detection-to-track association follows standard statistical tests using Mahalanobis distance. If a detection has a large confidence value η and is not associated to an existing track, it will indicate that a new track hypothesis can be initiated. On the contrary, track observations with low cluster confidence values are only used to track existing hypotheses.

When a vehicle track is initialized, one hypothesis for each possible perpendicular motion orientation β and $\beta_\perp$ are maintained in the filter. Both tracks are kept through time but at each step, only the one with the largest bounding box confidence value ν associated will be admitted as correct. After a while, the inherent and external motion observation will help the tracker to disambiguate which of both track hypotheses to follow. In this manner, the bounding box confidence ν helps only at the beginning of each track, when we have no evidence of present or past velocity.

3.2.1 Vehicle Tracking Results

We analyze the performance of the proposed tracking system with an ablation study with and without the deep-LiDAR detections from Chapter 2. Table 3.2 shows the tracking results obtained over the same validation dataset as in Chapter 2. We again include results using our deep-classifier with three detection modalities, i.e. front view ('FR-V'), bird's eye view ('BE-V') and 'Fusion'. We also add the *'Oracle'* comparison, which simulates perfect vehicle per-point classification. This comparison helps us to analyze the effect of the proposed deep architecture performance in that of the tracker. In this way, obtaining tracking results close to those got

Table 3.2: Vehicle Tracking Task - Validation Results on Front and Bird's Eye Domains (%)

Evaluation		Ours			Oracle		
		FR-V	BE-V	Fusion	FR-V	BE-V	Fusion
FR	MOTA	15.4	33.6	36.7	38.5	38.2	43.3
	Recall	66.6	65.7	66.7	68.6	67.2	69.0
	Prec.	65.9	76.7	78.7	80.1	79.9	82.5
	MT	47.3	52.7	49.0	50.0	46.3	52.7
	PT	37.3	30.0	34.5	37.2	40.0	33.6
	ML	15.5	17.3	16.3	12.7	13.6	13.6
BE	MOTA	-14.5	12.0	13.8	29.2	28.3	29.5
	Recall	56.4	57.7	58.3	64.3	62.8	62.8
	Prec.	50.2	62.5	63.7	73.4	72.3	73.2
	MT	36.4	35.4	37.3	45.4	41.8	45.5
	PT	40.9	40.9	40.9	37.3	40.0	37.3
	ML	22.7	23.6	21.8	17.2	18.1	17.2

when using the *'Oracle'* segmentation, is an indicator of the resilience of the tracking module to the possible errors of our deep segmentation. We present success rates (as percentages) for the 'Mostly Tracked' (MT), 'Partially Tracked' (PT), and 'Mostly Lost' (ML) tracking performance metrics. In addition, we show the precision and recall values, and the commonly used Multi-Object Tracking Accuracy (MOTA) metric [80].

The KITTI Tracking evaluation is also done on the front image plane ('FR' rows in the Table 3.2). We consider this as a drawback when evaluating the performance of LiDAR-only systems, since this projection may introduce distortions and not represent the real geometry of the scene. We therefore include a new tracking evaluation in the bird's eye domain ('BE' rows in Table 3.2) using the IoU of the ground truth and tracked boxes directly on the $X - Y$ plane. Analyzing the results, we can observe how our dual-branch deep architecture produces similar results in all tracking metrics than when using the *'Oracle'* segmentation. The improved performance of the presented detection approach with respect to [19] (represented by the 'FR-V' columns) is of great importance for the tracker, as it reduces the number of false positives that are fed to it. Furthermore, this allows us to reduce the cluster size thresholds obtaining also less false negative detections.

In Table 3.3, we additionally evaluate the tracking performance when using our deep models for point cloud segmentation over the online KITTI Testing benchmark, which analyze the 2D bounding boxes tracks in the RGB image plane. As using only LiDAR information makes it harder to fairly compare our results with other results that use images, we compare the obtained metrics with the work previously developed in [19] obtaining insights of the performance improvements

Table 3.3: Vehicle Tracking Task - Testing Set Evaluated over the Image Plane (%).

Testing	Prev. [19]	Ours		
	FR-V	FR-V	BE-V	Fusion
MOTA	15.5	21.9	36.3	39.7
Recall	55.4	62.3	62.7	63.6
Prec.	63.8	69.0	78.9	83.0
MT	18.5	26.3	32.3	29.5
PT	52.2	59.3	52.0	54.6
ML	29.4	14.3	15.7	15.8

obtained in the final tracking task with the introduction of the multi-branch architecture of Chapter 2.

From Table 3.3, we can extract several conclusions. Firstly, comparing both ‘FR-V’ methods we see how the additional improvements introduced in Chapter 2 boost the final tracking performance in all the metrics. Apart for consistently improving the independent branches results, our presented approach drastically beats the previous method [19] in all the metrics. We can also note that the number of false positives has been reduced, which is reflected in the precision improvement that evolves from 63.8% to 83.0%. Moreover, we remarkably increase the recall without producing undesired effects in the precision metric, which is directly translated into a more than double MOTA improvement.

3.3 Conclusions

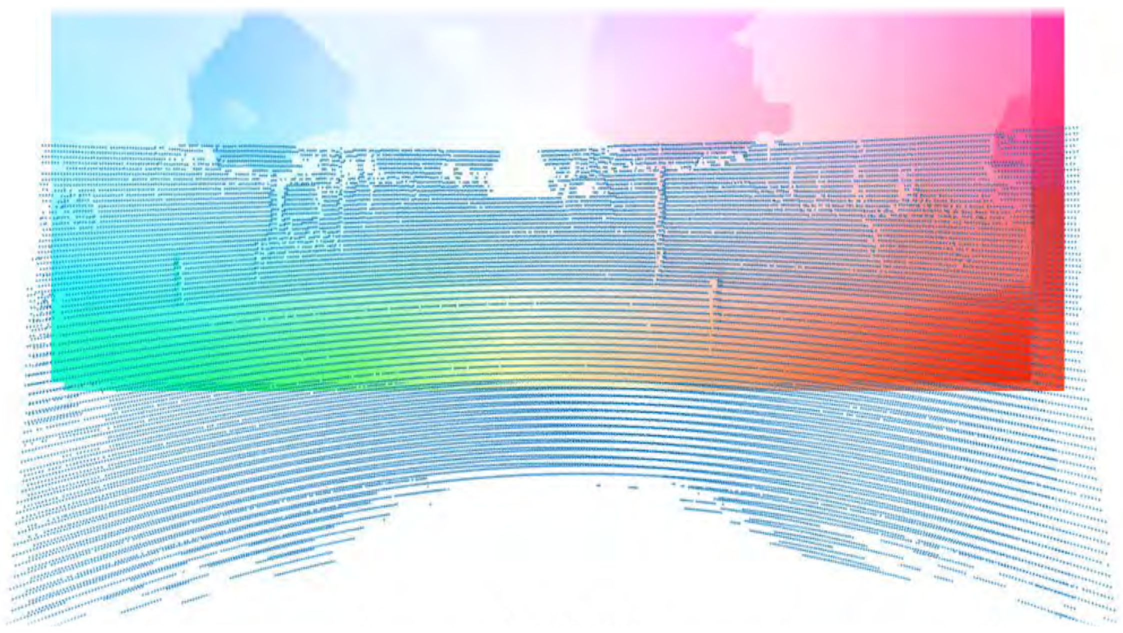
In this Chapter we have explored two different applications of our dual branch deep-LiDAR segmentation approach to achieve further and and more complex tasks. Specifically, in Section 3.1 we have detailed a full approach to filter LiDAR scans in order to build better maps that enable accurate re-localization over highly dynamic environments on subsequent days, whereas in Section 3.2 we demonstrate its use for tracking vehicles along time from the obtained bounding boxes.

In the first use-case we proposed a robust LiDAR-based re-localization algorithm for highly dynamic environments. By filtering possible movable objects based on the dual-view deep architecture presented on Chapter 2 we achieved a more robust, distinctive and static representation of the current environment which is used for further processing tasks such as path planning, map updating or re-localization. We proved that by eliminating from the source the movable objects, the re-localization accuracy inside a pre-built map can be increased by an average percentage of 35.1% compared to same re-localization performed over the full point clouds, and by 47.9% compared to a state-of-the-art LiDAR odometry and mapping algorithm. Furthermore we demonstrated the adaptability of our approach by applying it to a multi-day map building task, where the final map error is halved.

In the second use-case, we directly employ the bounding boxes extracted in Chapter 2 as initial observations for a multi-hypothesis extended Kalman filter tracker. We thoroughly evaluate the system performance in the KITTI Tracking benchmark. Results show that we obtain similar tracking scores employing our detection pipeline than the ones obtained when using an ideal detector based on the point-level ground truth used to train the networks. Moreover, when comparing to previous works, we remarkably increase the tracking recall without producing undesired effects in the precision metric, which is directly translated into a more than double MOTA improvement.

Part II

What are the scene's dynamics?


LiDAR projection over estimated Optical Flow to obtain pseudo LiDAR-flow ground truth.

4
Hallucinating Dense Optical Flow from LiDAR

In the second part of this dissertation, we focus on analyzing the scene dynamics. More specifically, we explore new ways to obtain optical flow, as it is the most common motion feature. For that purpose, we present a novel dynamic feature obtained from LiDAR point clouds which resembles RGB optical flow. Our final objective is to generate compatible optical flow from a different domain, and therefore the term *hallucinating*. In this manner, in the unlucky event of a camera failure or even at night conditions, our hallucinated optical flow could substitute standard RGB-based flow used by other on-vehicle algorithms. Then, in Chapter 5 we seek novel methods for combining both classification and regression tasks to solve a single problem, and propose a Joint Coarse-and-Fine reasoning for predicting optical flow from RGB images.

4.1 Introduction

In the previous chapters, we focused on analyzing the scene from a static point of view. However, the observation of the dynamism of a scene requires, at least, two frames captured within a time difference. In this chapter, we jump from the previous static frame analysis of the scene, to the interpretation of the environment dynamics by inspecting pairs consecutive frames.

On of the most expanded motion features used in Computer Vision and Robotics is optical flow. Determining optical flow consist on finding dense fields of displacement vectors which, assigned to certain pixel positions over an initial image, point to where that specific pixels will be found in the next image. In other words, it aims at finding correspondences between two consecutive input images. This correspondences are usually estimated by matching pixels or image feature representations but, up to date, it is yet a long-standing and crucial problem in computer vision. Finding corresponding pixels or extracted features is not often an easy task and many challenges may arise, such as untextured areas, outliers, occlusions, large displacements or illumination changes.

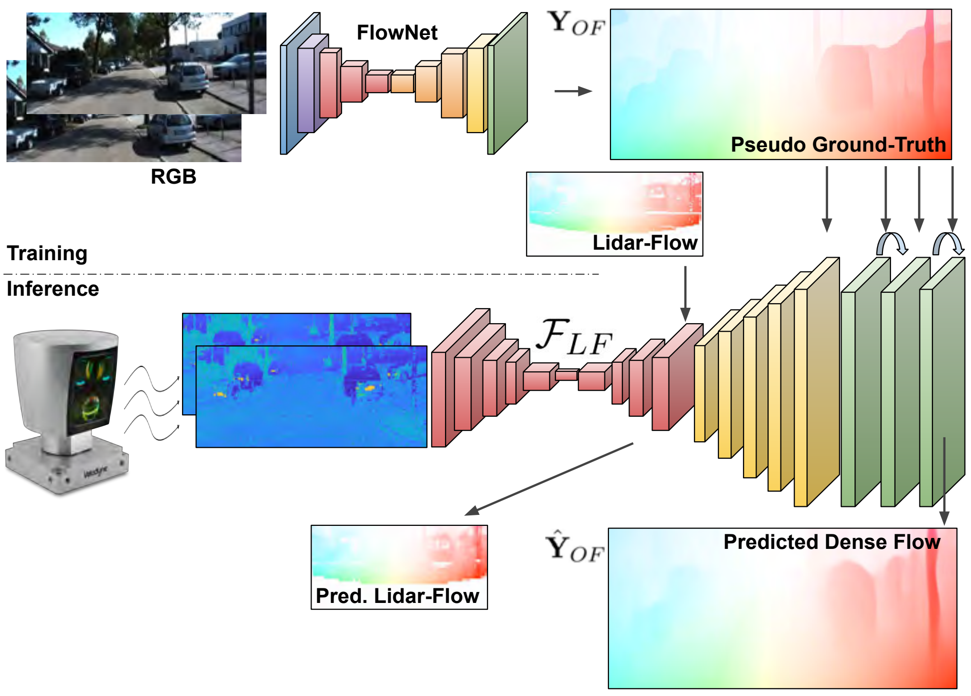


Figure 4.1: **Dense optical flow from sparse LiDAR.** We introduce a deep architecture that given two consecutive low-resolution and sparse LiDAR scans, produces a high-resolution and dense optical flow, equivalent to one that would be computed from images. Our approach, therefore, can replace RGB cameras when the quality images is poor due to e.g. adverse weather conditions. Notice that the RGB images shown in the top-left are only considered to generate the pseudo ground-truth used during training. Inference is done from only LiDAR scans.

Optical flow has gained great importance as a source of information for a wide range of computer vision and robotic tasks such as motion segmentation [81, 82], plane extraction [83], 3D reconstruction [84], object tracking [85] or even other diverse fields like video encoding [86]. In the autonomous driving field, optical flow estimation gains even more importance as it has become a decisive mid-level feature employed by higher-level tasks such as time-to-collision or object trajectory estimation. It is therefore relevant to investigate new alternative systems that are able to provide an substitute of this motion feature in the event of a camera failure.

In this chapter we propose a novel optical flow estimation approach based on CNNs which, using only sparse LiDAR information as input, is able to estimate in real-time dense and high resolution optical flow that is directly compatible with any camera-based estimated flow. In order to guide the network from the low-resolution and scarce LiDAR input to the final dense output we propose a three-block architecture. It introduces intermediate learning objectives at different resolutions in both LiDAR and image domains as well as refines the obtained prediction increasing the sharpness of the final solution. A sketch of our proposal can be seen in Fig. 4.1.

One of the main challenges that we face for this task is the lack of training data with pairs of LiDAR measurements and corresponding image-based optical flow. For training image-to-optical flow data-driven models, this issue has been commonly addressed with synthetically generated datasets [18, 87, 88]. However, common virtual datasets that provide optical flow ground-truth do not include LiDAR information nor provide a way of properly simulate both ranges and reflectivity measurements, and thus are not suitable for our purposes. On the other hand, real driving datasets such as the KITTI Dataset [6] may contain true LiDAR scans but not enough corresponding optical flow ground-truth samples as for training our desired deep models.

To circumvent this lack of training data, we elected a subset of the KITTI dataset, which contains RGB images with its corresponding LiDAR scans. We estimate a high-resolution optical flow from image pairs using the well established FlowNet2 [16], which will act as our pseudo ground-truth. This way, we build a LiDAR-to-optical flow dataset with around 20K samples.

We provide a quantitative evaluation on the LiDAR-available subset of the 'KITTI Flow 2015' benchmark showing that our approach is on par with other image based regressors and even close to FlowNet2. It demonstrates that it is possible to train models that may use image-derived information for training, but only need LiDAR inputs on inference. We also perform a qualitative evaluation on KITTI which show that, despite feeding the deep architecture with low-dimensional and sparse LiDAR measurements, we are able to predict high-resolution optical flow maps which are visually appealing.

To summarize, our contributions are:

- We generate an optical flow analogous to the one generated with camera images but using a much less informative yet robust to adverse weather conditions, LiDAR information only.
- We demonstrate that using ground truth derived from the image domain we can train data driven models without further needing RGB data on inference. This opens the door to further cross-modality training for algorithms deployed on hard to annotate domains.
- We propose a specialized convolutional architecture which handles first the problem in the LiDAR domain, then moves forward to the image-optical flow space and finally refine the obtained estimations in an end-to-end guided manner.
- We create a LiDAR-to-image flow dataset which does neither exist in the literature.

The rest of this Chapter is organized as follows. In Section 4.2 we show related works for both dynamic features and optical flow estimation topics. Next, Section 4.3 shows our approach to train a deep architecture in order to hallucinate optical flow from just LiDAR information. Experimental results are detailed in Section 4.4, where we show quantitatively and qualitatively that our new representation resembles RGB optical flow. We additionally demonstrate the effectiveness of this dynamic feature later in Chapter 6, in which we segment and predict the motion of vehicles in the scene. Conclusions and future work are presented in Section 4.5.

4.2 Related Work

Estimating accurate optical flow of RGB images from a real world scene is a recurrent challenge in computer vision since many years ago. By definition, it is an ill-posed problem, due to the fact that most of the 3D structural information gets lost over the capturing process of the camera. In this way, at pixel or patch level many issues arise in realistic dynamic environments, such as occlusions, 3D rotations, motion discontinuities, untextured areas, illumination changes, and large displacements. Additional difficulties emerge even for disambiguating the motion of the objects and the background from the camera perspective view.

The initial formulation for optical flow estimation was proposed by Horn and Schunck in 1981 as a variational approach [89]. It proposes to minimize an objective function with a data term enforcing brightness constancy and an spatial term to model the expected motion fields over the image. The formulation of these variational methods allows for the straightforward introduction of additional priors and different type of ad hoc blocks. These help to constrain the problem, accounting for key aspects such as for example combining local and global features [90], accounting for large displacements [91, 92], attending to edge motion [93], or focusing on robust patch matching [94, 95]. Further research has focused mainly in alleviating the drawbacks of the proposed methods, introducing specific improvements, [96–99]. For a more extensive report on classical optical flow methods, the reader is referred to [100].

The evolution of approaches trying to improve flow accuracy brought to the addition of object semantics and layered information [14, 101–104] to the problem formulation. Eventually the will of using semantic information and scene context to improve the flow estimation led to face the task with a learning approach. As we will see in the next Section, these learning approaches are mostly based on deep learning techniques and more specifically CNNs [8], which have shown its capacity to learn more useful features than traditional hand-made variational approaches, providing also robustness against rotation, translation and illumination changes.

4.2.1 Deep Learning for Optical Flow

Although Deep Learning penetrated with force contributing to a great number of computer vision tasks in an end-to-end fashion, for optical flow estimation deep models were initially applied just to improve different parts of the standard pipeline.

Primary approaches focused on descriptor matching schemes using deep neural networks to create and match descriptors between images patches or pixels. For example, PatchBatch [95] proposed a pipeline in which a CNN extracts patch descriptors which are then employed for matching via the PatchMatch [105] algorithm. These methods were also applied to stereo matching [106], or more specifically tailored for the optical flow problem with large displacements [107]. However, these descriptor matching schemes were mostly based on rigid descriptors, creating implicit rigid motion hypothesis which do not fit for fast neither large motions. To solve it, the *DeepMatching* [108] algorithm was proposed which, inspired by deep convolutional approaches, devised a hierarchical and multi-layer correlational architecture designed for matching even non rigid displacements.

Further methods, such as *EpicFlow* [93], grow over this deep matching algorithm focusing their attention on preserving and interpolating edge correspondences, which lead to better handling of occlusions and motion boundaries even for large displacements. Other approaches focus on the sparse-to-dense interpolation phase of optical flow pipelines, like Interponet [109], which propose a new data-driven algorithm based on a fully convolutional network.

The application of Deep Learning techniques to build end-to-end supervised optical-flow systems was expected to get higher accuracy and speed but was not immediate, basically due to the difficulty of obtaining a sufficiently large training set. The first totally CNN-based optical flow approach was introduced in FlowNet [18], where authors show that it is feasible to reach real-time state-of-the-art solutions training a CNN architecture end-to-end on synthetic data. FlowNet proposed two different networks for estimating optical flow without requiring any handcrafted method for aggregation, matching or interpolation. On the one hand, a simple end-to-end regressor architecture based on convolutional layers with an input of six channels from the two RGB images. On the other hand, a different CNN for each image input, which outputs are then merged by a new *correlation* layer. This layer helps the network in the pixel matching process by performing multiplicative patch comparisons between the two feature maps obtained. This correlation operation is identical to a convolution one, but instead of convolving data with a learned filter it directly convolves two data inputs, so no weights are learned.

These state of the art CNNs approaches for inferring optical flow (e.g. FlowNet), resort to synthetic training datasets with complete and dense ground-truth flow [18, 87, 88]. While the rendered images look very realistic, these datasets are not annotated with range nor reflectivity information provided by real laser sensors, so our objective is limited in this sense. Nonetheless, very new autonomous driving simulators and open virtual worlds such as Carla [110] or Synthia [111] are developing wider capacities. Some very recent works started to use simulated LiDAR-ranges information from them as well as video games [39, 112, 113] and even learned reflectivity distributions of real scenes [65] which can later be applied over virtual worlds gathered data. Yet, during this thesis these novel tools were not available and therefore we prioritized the use of real LiDAR information.

Apart from the use of synthetic data, the FlowNet [18] disruptive approach builds upon the recent success of deconvolutional [114, 115] blocks to solve dense pixel-wise prediction problems, such as semantic segmentation [15, 116–118] and super resolution [119]. Our work also has some connection with the super-resolution literature [119] in the sense that we aim to obtain a bigger and dense output than the input. Nevertheless, note that in super-resolution works, both input and output sources belong to the same domain of data. Here, besides having to handle the difference between the input and output domains and resolution, we need to resolve the additional task of estimating the flow.

Since FlowNet, other CNN architectures have been proposed for estimating optical flow. In this way, [120] uses a combination of traditional pyramids and convolutional networks providing features at different resolutions. *FlowNet2* [16], which is one of the main references, presents a scheme of stacked CNNs trained separately and with carefully chosen sample-learning schedules for large and small displacements. Later on, LiteFlowNet [121] presented a lightened and modernized version with a specialized architecture including regularizations, feature warping and feature-driven local convolutions, outperforming its predecessor while drastically reducing the number of parameters from $160M$ to $5M$ and improving the speed 1.36 times.

Contemporary methods ranking on the upper position in the KITTI flow 2015 benchmark [122], focus both on increasing time and performance by using combinations of wrapping techniques and pyramids [123, 124], or even hierarchically approximating the distribution of pixel correspondences in the dataset [125]. At the time of writing this thesis dissertation, the best performing model in this benchmark [126] tackles the problem jointly with 3D scene flow estimation. Similarly than the previous layered approaches, it bases on the hypothesis that in the autonomous driving scenario the motion of the scene can be obtained by estimating the 3D motion of each actor. Therefore, they propose a deep structured model that exploits optical flow, stereo, and instance segmentation as visual cues to build an energy minimization problem which is solved by unrolling a Gaussian-Newton solver efficiently on GPUs.

To solve the lack of optical flow ground-truth, other recent works tackle the problem in an unsupervised manner. Some of these methods [127, 128] directly replace the supervised loss by new ones that rely on the classical brightness and photometric constancy minimization even adding motion smoothness terms. Other works make use of more elaborated unsupervised losses taking advantage of warping techniques. For example, [129] wraps optical flow in a bidirectional manner and includes in its loss a census transform to compensate for additive and multiplicative illumination and gamma changes on the images. Although these methods alleviate the need of extensive annotated flow ground-truth or the use of virtual environments, their actual performance is lower and usually need to be precisely fine-tuned to provide results close to supervised methods.

4.3 Hallucinating Dense Optical Flow

In this Section we detail our deep architecture to hallucinate dense high resolution optical flow in the image domain using as input only sparse and low resolution LiDAR information. Our network solution bridges the gap between LiDAR and camera domains, so that when the camera images are spoiled (e.g. at night sequences or due to heavy fog), we can still provide an accurate optical flow to directly substitute the degenerated image-based prediction in any vehicle algorithm.

4.3.1 Problem Statement

Let us define an end-to-end CNN to predict dense optical flow in the image domain as $\mathcal{F}_{LF} : (\mathbf{I}_t, \mathbf{I}_{t+n};\ \mathbf{Y}_{OF}) \rightarrow \hat{\mathbf{Y}}_{OF}$ where $\mathcal{F}_{LF}$ is the desired LiDAR-Flow (LF) deep architecture that hallucinates a dense optical flow $\hat{\mathbf{Y}}_{OF} \in \mathbb{R}^{M \times N \times 2}$ from two n-time separated LiDAR frames $\mathbf{I}_t$ and $\mathbf{I}_{t+n} \in \mathbb{R}^{H \times W \times 2}$ obtained as specified in Chapter 2 Section 2.4.1, and $\mathbf{Y}_{OF} \in \mathbb{R}^{M \times N \times 2}$ represents the image domain optical flow used as pseudo-ground truth.

Our problem states two main challenges. On one hand, LiDAR and image FOV are not totally overlapping. On the other hand, the resolution of the input LiDAR scans $H \times W$ is generally much smaller than the $M \times N$ size of the RGB images on which we seek to hallucinate the optical flow. A naive end-to-end deep LiDAR-to-Flow model $\mathcal{F}_{LF}$ would consist of processing the $H \times W$ input by stacking layers of convolutions and deconvolutions or up-sampling [114] until obtain the desired $M \times N$ output size in the image FOV. However, in the first entry of Table 4.1, we show that a simple model like that naive approach is not capable of capturing correctly the motion of the scene.

We therefore devise a more elaborated architecture consisting of three main blocks, as shown in Fig. 4.3. The first one estimates the motion in the sparse LiDAR domain using a specific architecture resembling FlowNet [18], and it is trained with a ground-truth LiDAR optical flow. The second leads the transformation from LiDAR-to-image domain performing also an output up-sampling while guiding the learning towards the prediction of the final optical flow in the image domain. Finally, a refinement step is implemented to produce more accurate, dense, and visually appealing predictions.

In the following sections we first describe the process to create our training data, including the input LiDAR frames and their associated image-based optical flows. We then describe each one of the building blocks of the proposed architecture.

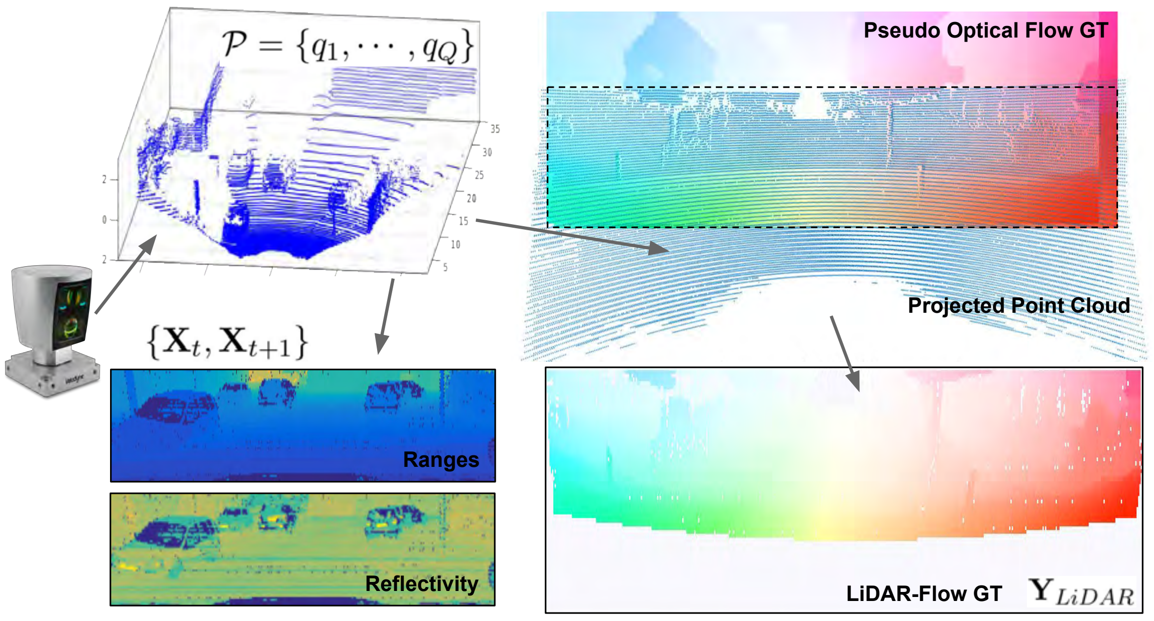

$\mathcal{P} = \{q_1, \cdots, q_Q\}$
$\{\mathbf{X}_t, \mathbf{X}_{t+1}\}$
Ranges
Reflectivity
Pseudo Optical Flow GT
Projected Point Cloud
LiDAR-Flow GT $\mathbf{Y}_{LiDAR}$

Figure 4.2: **LiDAR-to-optical flow dataset.** Given a 3D point cloud from a laser scan (top-left) we create our input tensors with the range and reflectivity information (bottom-left). Dense Optical Flow in the image domain $\mathbf{Y}_{Dense}$ used as pseudo ground-truth is created by cropping the overlapping areas between the image-based flow and the projected point cloud (dashed rectangle). The LiDAR-Flow $\mathbf{Y}_{LiDAR}$ ground-truth built to guide the domain transformation is obtained by getting the flow measurements of the LiDAR projection over the RGB optical flow.

4.3.2 Input/Output Data

To learn the parameters of the proposed deep architecture we need the following training data: i) LiDAR data aligned with an RGB camera, i.e. we need to know the mapping from the 3D range measurements to the image plane on which we aim to densely hallucinate the optical flow; ii) corresponding optical flow ground-truth annotated in the image domain. Both these types of data are by themselves scarce, and there exists no driving related dataset containing enough samples of both of them put in correspondence.

For our first requirement, we consider the KITTI Tracking dataset [6, 130], which specifically provides measurements from a Velodyne HDL-64 LiDAR sensor as well as grant the camera-LiDAR calibrations per frame.

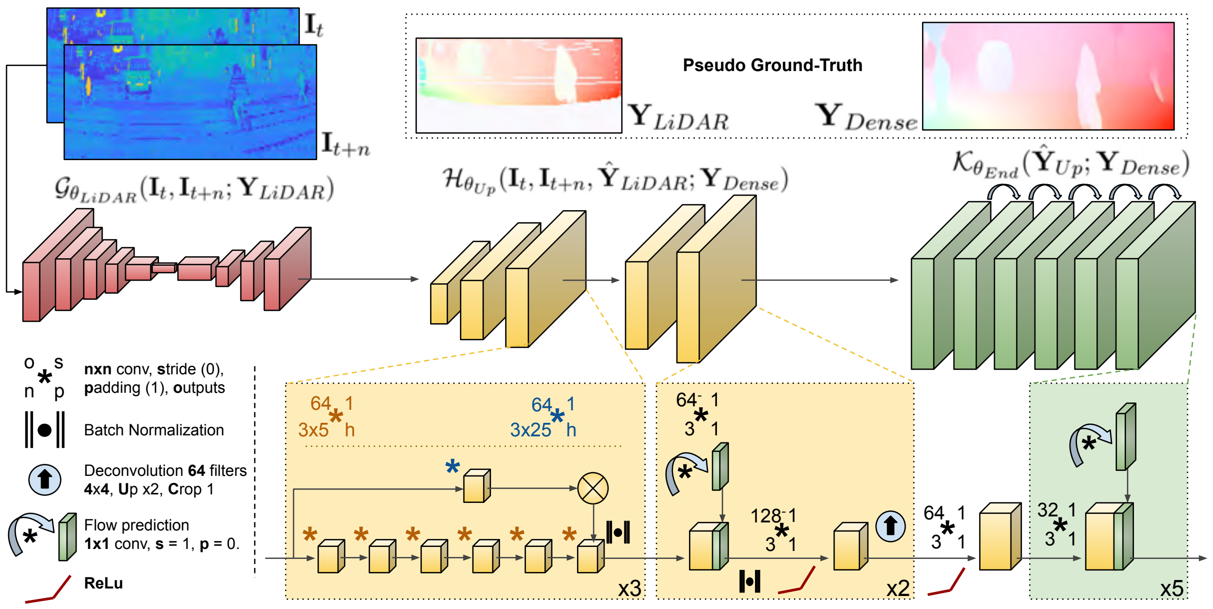
The second requirement is harder to achieve. The KITTI 'Optical Flow 2015' challenge contains around 200 samples of sparsely annotated optical flow with the corresponding RGB images which neither is enough for training purposes nor LiDAR information is provided. However, on other KITTI challenges we can find an associated RGB image per each LiDAR scan. We therefore exploit this correspondence and employ the RGB images from the 'Tracking' challenge to compute a pseudo ground-truth for optical flow using the well established method FlowNet2 [16]. Due to the specific KITTI vehicle's setup, the vertical FOV of the Velodyne LiDAR sensor does not cover the full corresponding RGB image height M. This can be clearly appreciated on the top-right area of Fig. 4.2, where we have projected the LiDAR measurements over the image-based FlowNet2 optical flow predictions. We therefore perform a cropping operation over this RGB-based optical flow ($\mathbf{Y}_{OF}$) eliminating the top area that is not covered by the LiDAR vertical FOV as can be appreciated in Fig. 4.2. Let us denote by $\mathbf{Y}_{Dense} \in \mathbb{R}^{M' \times N' \times 2}$ this subset of the optical flow in the image domain that we will use as pseudo ground-truth for training our LiDAR-to-Flow architecture. At this point, we have already built a training set consisting of input LiDAR frames $\{\mathbf{I}_t, \mathbf{I}_{t+n}\}$ and its corresponding FOV image-based optical flow ground-truth $\mathbf{Y}_{Dense}$, which would be enough to train the commented naive end-to-end regressor as shown in the first entry of Table 4.1. However, by building intermediate objectives using domain specific losses and training data, we obtain far better results.

In the following Sections, we detail the previously stated three main blocks of our LiDAR-to-Flow approach, to be: 1) LiDAR-Flow prediction 4.3.3, 2) LiDAR-to-Image domain transformation 4.3.4, and 3) hallucinated flow refinement 4.3.5.

4.3.3 LiDAR-Flow

With this first block, we aim to predict what we call *LiDAR-Flow*, which is the low resolution equivalent optical flow in the LiDAR domain from two consecutive LiDAR point clouds $\mathbf{I}_t$ and $\mathbf{I}_{t+n}$. For this task, we created an additional ground truth $\mathbf{Y}_{LiDAR} \in \mathbb{R}^{H \times W \times 2}$, which is computed by projecting the LiDAR point cloud $\mathcal{P}$ onto the dense RGB image flow $\mathbf{Y}_{Dense}$. We then keep the corresponding flow values from each overlapping pixels, building the desired $\mathbf{Y}_{LiDAR}$. Again, due to specific vehicle's setup, several layers of the LiDAR sensor do not collide with any flow value (mostly those layers pointing towards the floor) so by design we do not assign any motion to these points, as can be seen in Fig. 4.2 with the white areas. Since the input LiDAR frames are low resolution, noisy and scattered, so will be our $\mathbf{Y}_{LiDAR}$.

We train a network $\hat{\mathbf{Y}}_{LiDAR} = \mathcal{G}_{\theta_{LiDAR}}(\mathbf{I}_t, \mathbf{I}_{t+n}; \mathbf{Y}_{LiDAR})$ in order to learn this low dimensional flow, being $\hat{\mathbf{Y}}_{LiDAR}$ the predicted LiDAR-Flow and θ_{LiDAR} the trainable parameters of the network $\mathcal{G}_{\theta_{LiDAR}}$. A sketch of the network is shown in red color in Fig. 4.3. Although this network follows a similar contractive-expansive architecture as FlowNet [18], we include some important modifications. We perform 5 contraction levels, creating feature maps of up to $1/32$ of our initial LiDAR-input resolution. We expand back the generated feature maps up to the initial resolution predicting therefore $\hat{\mathbf{Y}}_{LiDAR} \in \mathbb{R}^{N \times M \times 2}$. During this expansion process, we predict intermediate LiDAR-Flows at $4x28$, $8x56$, $16x112$ and $32x224$ resolutions following a multi-resolution pyramid schema which greatly helps on the estimation accuracy.



$\mathbf{I}_t$
$\mathbf{I}_{t+n}$
Pseudo Ground-Truth
$\mathbf{Y}_{LiDAR}$
$\mathbf{Y}_{Dense}$
$\mathcal{G}_{\theta_{LiDAR}}(\mathbf{I}_t, \mathbf{I}_{t+n}; \mathbf{Y}_{LiDAR})$
$\mathcal{H}_{\theta_{Up}}(\mathbf{I}_t, \mathbf{I}_{t+n}, \hat{\mathbf{Y}}_{LiDAR}; \mathbf{Y}_{Dense})$
$\mathcal{K}_{\theta_{End}}(\hat{\mathbf{Y}}_{Up}; \mathbf{Y}_{Dense})$
nxn conv, stride (0), padding (1), outputs
Batch Normalization
Deconvolution 64 filters 4x4, Up x2, Crop 1
Flow prediction 1x1 conv, s = 1, p = 0.
ReLu
x3
x2
x5

Figure 4.3: **LiDAR to dense optical-flow architecture.** The proposed network is made of three main blocks sequentially connected which resolve the problem in different stages: 1) Estimation of the LiDAR-flow in low resolution (red layers); 2) Low-to-high resolution flow transformation and LiDAR-to-image domain change (yellow layers); 3) Final flow refinement (green layers).

The final LiDAR-Flow prediction at resolution of $64x448$ is concatenated to both LiDAR input frames and handed over to the next LiDAR-to-Image domain transformation step to perform the flow hallucination. We also include in the same way shortcuts between the contractive and expansive levels in order to create richer features for the final predictions. Filter sizes, steps and padding hyperparameters are the same as those used in FlowNet except some exceptions needed to match our resolutions.

4.3.4 LiDAR to Image Domain Transformation

The second major block of our architecture is in charge of performing the domain transformation, bringing the low resolution LiDAR-Flow to the high resolution RGB image domain. Specifically, this convolutional module receives as input the LiDAR-Flow $\hat{\mathbf{Y}}_{LiDAR} \in \mathbb{R}^{H \times W \times 2}$ predictions along with the two input LiDAR frames and produces as output an upscaled image-centered optical flow prediction learned from $\mathbf{Y}_{Dense}$. We formally describe this block as $\hat{\mathbf{Y}}_{Up} = \mathcal{H}_{\theta_{Up}}(\mathbf{I}_t, \mathbf{I}_{t+n}, \hat{\mathbf{Y}}_{LiDAR}; \mathbf{Y}_{Dense})$, where $\mathcal{H}_{\theta_{Up}}$ represents the model with learned parameters θ_{Up}, and $\hat{\mathbf{Y}}_{Up} \in \mathbb{R}^{M' \times N' \times 2}$ is the output predicted flow in the image domain as seen in Fig. 4.4.

In order to actively guide this domain transformation process, we devise an architecture with two sub-blocks, which can be seen in yellow in Fig. 4.3. The first sub-block consists on a set of multi-scale filters in two convolutional branches, providing context knowledge to the network. In one branch we produce high level features by applying 6 consecutive convolutional layers (plus batch normalization and leaky ReLU non-linearity) with small 3×7 filters and without any lateral padding (which allows the feature maps to grow horizontally for matching the desired output resolution). In the other branch, lower frequency features are generated with a convolutional layer using wider 3×25 filters and outputting the same feature map resolution. Finally, the features of both branches are concatenated. As can be seen in Fig. 4.3, this branched expansion process is replicated three times, with the distinction that in the second and third rounds the high frequency branch use respectively three convolutional layers with filters of size $3x9$ and two layers with $3x13$ filters. This design decision was proved to provide a good trade-off between network size and performance. The final spatial resolution of the feature maps from this transition sub-block is $M'/8 \times N'/8$ and no flow prediction is performed at this point.

The second sub-block raises the resolution of the feature map to the final size $M'/2 \times N'/2$ in the image domain, performing iteratively a refinement of the hallucinated flow in the image-domain already using as ground truth $\mathbf{Y}_{Dense}$ while increasing the resolution. This full process is repeated twice until the desired final flow resolution is obtained. Notice here that we upsampled until $M'/2 \times N'/2$ to boost the system speed, as in our experiments, including a third block does not produce better results than just bilinearly interpolating the final prediction.

4.3.5 Hallucinated Optical Flow Refinement

The flow estimated in the previous step tends to be over-smoothed. Algorithms predicting dense images in which the contours are important (e.g. semantic segmentation) commonly perform refinement steps to produce more accurate outputs. Conditional Random Fields (CRF) is one of the preferred methods for this purpose, and has recently been approached by using Recurrent Neural Networks [18,39,131]. The procedure can be roughly seen as an iterative process over a previous solution predicted. We design a similar iterative convolutional approach for refining the hallucinated optical flow, as is sketched in green in Fig. 4.3, so that avoiding the computational burden of CRFs and obtaining a fully end-to-end trainable architecture.

We formally denote this final refinement step as $\hat{\mathbf{Y}}_{End} = \mathcal{K}_{\theta_{End}}(\hat{\mathbf{Y}}_{Up}; \mathbf{Y}_{Dense})$. It works by consecutively predicting an optical flow which is concatenated to the input feature maps used to generate it. This composed tensor serves as input for the next iteration that generates again new feature maps and a new optical flow prediction, but this time with a better knowledge of the desired output. This process is repeated 5 times and allows obtaining sharper contours as well as smoother flows.

4.4 Experiments

Here we show the training procedures, experiments performed and results obtained on our approach for obtaining a suitable optical flow at image resolution from just LiDAR information. Some qualitative results are shown on Fig 4.4. There we show 4 different samples presenting in the first column the two input LiDAR-ranges, in the second column the ground truth (odd rows) and predicted (even rows) LiDAR-Flow, and correspondingly in the third column the dense ground truth and our final predicted dense flow from the LiDAR input.

4.4.1 Train, Test and Validation Sets

In our experiments we use the KITTI Tracking benchmark [6] to build a new LiDAR-to-Flow dataset. It contains $19,045$ sample pairs for both RGB and LiDAR measurements from a Velodyne HDL-64 sensor grouped in 50 different sequences. As pseudo ground-truth for training our architectures, we use the clipped optical flow $\mathbf{Y}_{Dense}$ predicted by FlowNet2 [16] from the RGB images. We build the LiDAR-flow ground truth $\mathbf{Y}_{LiDAR}$ from the associated LiDAR measurements processed as described in Section 4.3.3. Notice that our approach only uses RGB images to create the pseudo ground-truth optical flow for training, but none of them are used at all during inference, fulfilling our objective of creating a system that works solely with LiDAR information.

To provide a quantitative evaluation for our models we need real ground-truth to compare against. For this, we use the training set of the 'KITTI flow 2015' benchmark, which is composed of 200 pairs of RGB images. However, as no LiDAR information is provided on these samples, we performed a match search between the RGB images of the 'KITTI flow' and the 'Tracking' benchmarks. By doing this, we were able to annotate 90 pairs of Velodyne scans (from the Tracking benchmark) with their associated real optical flow in the image domain (from the Optical Flow 2015 challenge), which compose our *test* set with real $\mathbf{Y}_{Test}$ ground truth. We split the remaining 18,955 Velodyne frame pairs into two subsets, creating a *train* set of 17,500 samples and a *validation* set of 1455 samples, each of those containing non-overlapping sequences.

LiDAR and flow resolutions. In our experiments, input LiDAR frames $\mathbf{I}$ have a size of $H = 64; W = 448$, which is the same as the intermediate LiDAR-Flow ground truth $\mathbf{Y}_{LiDAR}$. Both the final output $\hat{\mathbf{Y}}_{Dense}$ and its corresponding ground truth $\mathbf{Y}_{Dense}$, have a resolution of $M' = 256; N' = 1224$. This increase of the resolution poses the main difficulty for our method, which needs to predict dense optical flow from a scarce source with more than ten times less input data.

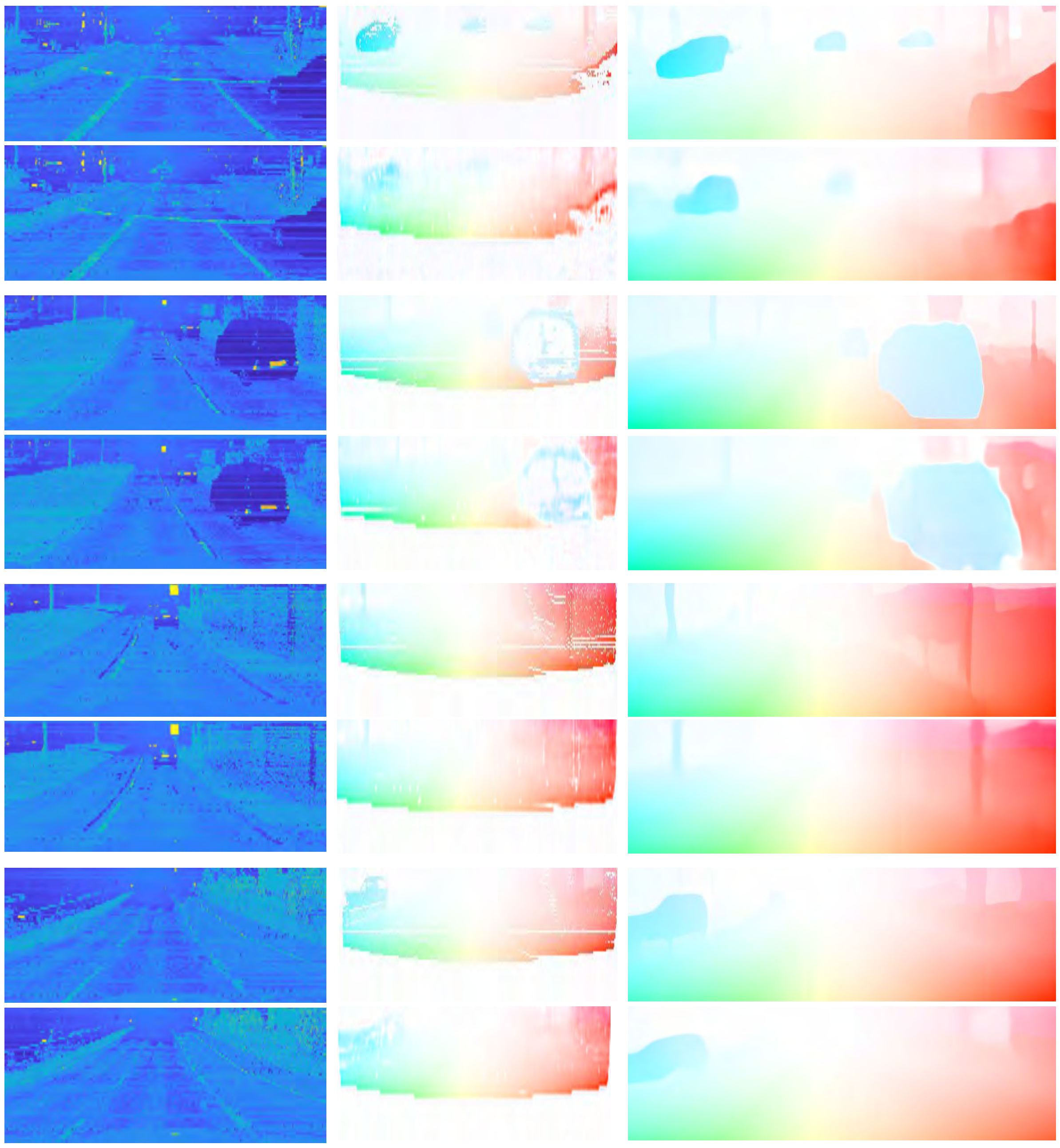


Figure 4.4: **Qualitative results of our system.** All images are taken during inference from our validation LiDAR-image-flow set, so none of them were previously seen during training. We show four different example scenes, representing the following: Column 1: LiDAR inputs $\mathbf{I}_t$ and $\mathbf{I}_{t+1}$; Column 2-odd rows: LiDAR-Flow pseudo ground-truth bY_{LiDAR}; Column 2-even rows: LiDAR-Flow prediction $\hat{\mathbf{Y}}_{LiDAR}$; Column 3-odd rows: dense pseudo ground-truth, bY_{Dense}; Column 3-even rows: final predicted dense optical flow $\hat{\mathbf{Y}}_{End}$. A video with the Full sequences can be found here: https://youtu.be/94vQUwCZLxQ

4.4.2 Implementation Details

Our modules are trained in an end-to-end manner, in the way that the output of each one becomes the input to the next. For that we use the MatConvNet [132] Deep Learning Framework. All the weights are initialized with He's method [66] and Adam optimization [67] is used with standard parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Training is carried out on a single NVIDIA 1080Ti GPU throughout 400000 iterations, each iteration using a batch of 10 Velodyne input pairs of consecutive frames, i.e. $\mathbf{I}_t$ and $\mathbf{I}_{t+1}$, sampled randomly over the training dataset. Data augmentation is performed over the LiDAR inputs only by flipping the paired frames horizontally with a 50% chance, so that preserving the strong geometric properties of the laser measurements and the natural motion of the scene. Learning rate is set to 10^{-3} for the first 150000 iterations, and halved each 60000 iterations.

Definition of the loss. Our approach performs an end-to-end regression, for which the learning loss is measured at up to twelve places $\sum_{i=1}^{12} \lambda_i \mathcal{L}_i$: five of them in the *LiDAR-Flow* module described in Section 4.3.3, two more in the up-sampling step shown in Section 4.3.4 and the last five at full resolution in the final refinement step detailed in Section 4.3.5. All these losses compute the ℓ_2-norm of the difference between the predicted optical flow and the corresponding pseudo ground-truth of the Training set. The multi-loss regularizers λ are set to 1 to avoid further fine-tuning, therefore assigning equal importance to each domain and resolution.

4.4.3 Ablation Study

We summarize in Table 4.1 the ablation study that analyze the contributions of our blocks LiDAR-Flow, Domain-Transformation and Refinement to the final quantitative results. We also trained a naive approach with a similar architecture to the one used in our LiDAR-Flow subnetwork, but extending the output resolution by stacking extra convolutional blocks. As quantitative measurements, we follow the 'KITTI Flow 2015' benchmark [133, 134] guidelines obtaining the Percentage of outlier pixels. A pixel is considered to be correctly estimated if the End-Point-Error (EPE) calculated as the averaged Euclidean distance between the prediction $\hat{\mathbf{Y}}_{Test}$ and the real ground-truth $\mathbf{Y}_{Test}$ is $<$ 3px or $<$ 5% of its true value. These measurements are averaged over background regions only, over foreground regions only (mostly accounting for the 'movable' objects of the scene), and over all ground truth pixels, which respectively are denoted in Table 4.1 as 'Fl-BG', 'Fl-FG' and 'All'. The 'Noc' and 'Occ' values refer to the evaluation performed over the 'Non-Occluded' regions only and over 'All' the regions respectively, as specified by the KITTI benchmark. In addition, we include the EPE obtained against our Validation set, which gives us an idea of how close we are to our upper bound of FlowNet2.

At the light of the results shown in Table 4.1 (the lower values, the best), it is clear that our objective of guiding the learning from the low-resolution LiDAR domain to the higher RGB-

Modules				Fl-BG	Fl-FG	Fl-ALL	EPE
LiDAR-Flow	Dom.Transf.	Refinement					
✗	✗	✗	Noc	56.74	82.75	61.24	14.19
			Occ	58.11	83.14	62.04	
✓	✗	✗	Noc	23.20	57.67	29.15	6.78
			Occ	24.80	58.56	30.09	
✓	✗	✓	Noc	20.09	54.02	25.95	5.49
			Occ	22.26	54.51	27.31	
✓	✓	✓	Noc	18.65	51.56	24.33	5.16
			Occ	20.88	52.58	25.84	
FlowNet2 [16] (*)			Noc	7.24	5.6	6.94	-
			Occ	10.75	8.75	10.41	
InterpoNet [109] (*)			Noc	11.67	22.09	13.56	-
			Occ	22.15	26.03	22.80	
EpicFlow [93] (*)			Noc	15.00	24.34	16.69	-
			Occ	25.81	28.69	26.29	

Table 4.1: **Quantitative evaluation and comparison with RGB optical-flow methods.** Flow for *background* (Fl-BG), *Foreground* (Fl-FG) and *All* (Fl-ALL) is measured for both *non-occluded* (Noc) and *full* (Occ) points, as in KITTI Flow. The End-Point-Error (EPE) is measured against the pseudo ground-truth computed using FlowNet2. (*) indicates that for these methods the test set is slightly different from ours, as we could not obtain the corresponding LiDAR frames needed for our method from all the KITTI 'flow' Test set RGB images. Although indicative, these results show that our LiDAR-based approach is on par with other well-known optical flow algorithms that rely on higher resolution and quality input images.

optical flow domain is much better fulfilled with our modular approach. We certainly can appreciate how just the introduction of the LiDAR-Flow intermediate feature reduces approximately the errors to half of the ones obtained by a similar naive model just focused on predicting the final result. This is a key cue that we later develop in Chapter 6, where we exploit the inclusion of pretext tasks in order to solve higher level problems. Moreover, the successive addition of the Domain Transformation and Refinement modules, keep pushing further the performance of our system, making it comparable to other image-based optical flow methods.

We can conclude that our approach performs very close to other state of the art methods which use high-resolution images as input. Although we suffer from larger errors for the foreground predictions ('Fl-FG'), our overall results are on par with the ones obtained by e.g. EpicFlow [93], which is one of the first approaches exploiting deep architectures. This appreciation shows there is quite room for improvement on our optical flow predicted for the Foreground objects, which is the weakest point of our method. We leave this topic open for further research, as is shown in Section 4.5. It is also noticable the robustness to occlusions of our method, as the difference between results for both 'Noc' and 'Occ' is less significant than in other methods, and even better or very similar to InterpoNet [109] and EpicFlow [93].

4.5 Conclusions and Future Work

In this chapter we have presented an approach to regress high resolution image-like optical flow from low resolution and sparse LiDAR measurements. For this purpose, we have designed a deep network architecture made of several blocks that incrementally solve the problem, first estimating a low resolution LiDAR-flow, and then increasing the resolution of the partial solution to that on the image domain as well as refining the final predictions. For training our network we composed a new dataset of corresponding LiDAR scans and high-resolution image flow predictions that we use as pseudo ground-truth for training. The results show that the flows estimated by our architecture are competitive with those computed by methods that rely on high-resolution input images. There is still room for improvement in order to get more accurate flow predictions that we left for future work. For example, the addition of better refinement steps to improve results on foreground objects or even the inclusion of additional tasks such as semantic segmentation to focus the attention on those elements standing out.

5
Joint Coarse-and-Fine Reasoning for Deep Optical Flow

Following the general practices in Machine Learning problems, in the previous chapters we have faced the task under study as a set of classification or regression problems depending on the nature of the task. In this way, we segmented vehicles in Chapter 2 by imposing a per-pixel classification task and predicted a new optical-flow feature from LiDAR data by stating a regression one.

In this Chapter, we explore the capabilities of using both classification and regression in order to solve a single problem, and propose a Joint Coarse-and-Fine reasoning that we exemplify for estimating optical flow from RGB images.

5.1 Introduction

New CNN architectures and training procedures are day after day becoming the new state of the art, producing models which prediction accuracy was inconceivable few years ago. These models commonly approach the task under study by stating either a regression or a classification problem, addressing in this way all kind of perception related problems, such as image classification, object detection or optical flow.

For example, it is common to represent semantic segmentation [12, 15, 19, 20, 42, 117, 118, 135, 136] as multiple classification problems over a finite and discrete set of categories, in the same manner as we did for segmenting vehicles in Chapter 2. On the other hand, motion related tasks, such as optical flow prediction [16,18,22,88,120,121,123,124,127–129,137], are usually represented as regression problems over a continuous 'flow space'. Typical examples of losses associated to classification and regression problems are respectively Cross-Entropy and Mean Squared Error.

Unlike existing methods that address the learning task casting it to a classification or a regression problem depending on its nature, in this Chapter we propose an alternative representation.

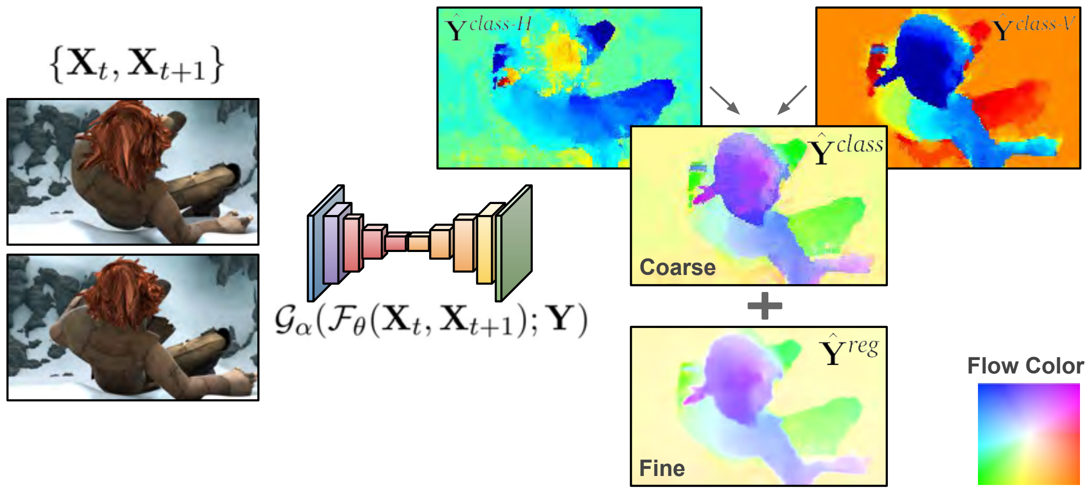

$\{\mathbf{X}_t, \mathbf{X}_{t+1}\}$
$\mathcal{G}_\alpha(\mathcal{F}_\theta(\mathbf{X}_t, \mathbf{X}_{t+1}); \mathbf{Y})$
$\hat{\mathbf{Y}}^{class\text{-}H}$
$\hat{\mathbf{Y}}^{class\text{-}V}$
$\hat{\mathbf{Y}}^{class}$
Coarse
+
$\hat{\mathbf{Y}}^{reg}$
Fine
Flow Color

Figure 5.1: **Joint Coarse-and-Fine Reasoning.** We approach dense per-pixel regression problems with a joint Coarse-and-Fine method and apply it to the challenging problem of optical flow estimation. Our method produces a coarse result ($\hat{\mathbf{Y}}^{class}$) that estimates separately the horizontal ($\hat{\mathbf{Y}}^{class\text{-}H}$) and vertical ($\hat{\mathbf{Y}}^{class\text{-}V}$) pixel motions stated as a per-pixel classification problem. On top of that, we perform a fine flow ($\hat{\mathbf{Y}}^{reg}$) estimation obtained by regression. The explicit combination of both coarse and fine solutions produces our final optical flow.

In our novel approach, we simultaneously state a classification and a regression problem that jointly solve the final task (exemplified here as optical flow prediction). We are therefore able to take advantage of their respective benefits, i.e., i) obtaining a simplified coarse solution via classification, which helps the training to converge quicker and ii) distilling the fine details of the coarse predictions via a regression solution that generates a more detailed estimation.

The classification component obtains general coarse information that is important to focus the search around the solution space, while the regression component carries the fine details needed to produce an accurate prediction. We therefore call this a Coarse-and-Fine (CaF) reasoning and apply it to the context of optical flow due to its challenging nature, where a real value needs to be predicted for each pixel of an image that may follow any kind of large or small motion. The general idea of our approach is represented Fig. 5.1.

Our conjecture is that this representation is more suitable that the existing ones as it is able to tackle both small and large displacements, as well as helps to reach better solutions faster. To enforce this joint representation we propose one simple but effective loss function that linearly combines the classification and the regression costs. We also show how to fully integrate this representation in any network architecture by introducing a new layer that expresses the final prediction as the addition of a refinement real component on top of a coarse discrete approximation. We demonstrate the benefits of our proposal in state-of-the-art optical flow datasets.

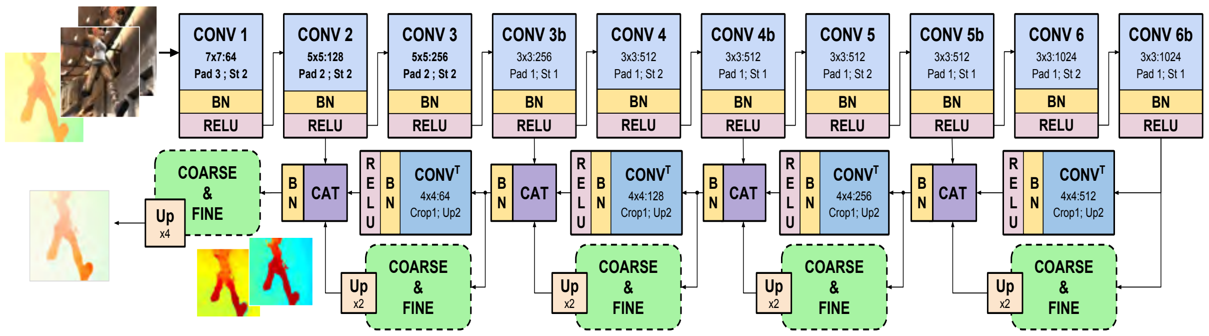


Figure 5.2: **Coarse-and-Fine deep architecture for estimating RGB optical flow.** We employ an encoder-decoder architecture similar to FlowNet [18], composed by a contractive and an expansive part. First, a set of Convolutions, Batch Normalization and ReLU layers (CONV, BN and RELU) are interleaved to obtain higher-level hierarchical representations while contracting the input images. The final dense prediction is generated by deconvolution layers (CONV^T), and guided to an optimal solution by concatenating (CAT blocks) also the outputs from the previous feature maps and the partial Coarse-and-Fine (CaF) module solutions obtained. We perform this concatenations at five different resolutions of the expansive area to obtain finer details.

5.2 Coarse-and-Fine Formulation

Deep Learning-based optical flow, as well as many other dense pixel-wise generation tasks, is commonly formulated as a regression problem in order to predict a solution that is intended to capture fine details.

In this work we show that it is also convenient and accurate to jointly represent a coarse classification component, which contains a generic and discrete approximation to the solution, and a fine regression component, which provides a fine and continuous refinement. Fig. 5.2 shows a general draft of our approach where both coarse and fine predictions can be appreciated. The introduction of an explicit discrete classification term draws inspiration from semantic segmentation methods, which exhibit fast convergence rates. In our case, this component helps on accelerating the training by quickly centering the search space around a coarsely correct solution. Moreover, the effect obtained is that the fine regressor will not need to care distinctly about large or small motions as other solutions do, and just provide results within its own bin space.

Here we describe the key concepts and ingredients used to fully exploit this joint Coarse-and-Fine representation, including two different network topologies with their respective training methods and associated loss functions.

5.2.1 Estimating coarse information as an auxiliary task

Let us here define a basic architecture devoted to estimating RGB-based optical flow as a combination of two blocks, $\mathcal{F}_\theta(\cdot)$ and $\mathcal{G}_\alpha(\cdot)$. Given a pair of RGB image inputs $\{\mathbf{X}_t, \mathbf{X}_{t+1}\} \in \mathbb{R}^{H\times W\times 3}$, an initial stage of the network computes features $\mathcal{F}_\theta(\mathbf{X}_t, \mathbf{X}_{t+1})$ according to the network model θ. Then, we can devise a second stage which, under supervision of a real ground-truth $\mathbf{Y}$, transforms these features into pixel-wise optical flow predictions as $\mathcal{G}_\alpha(\mathcal{F}_\theta(\mathbf{X}_t, \mathbf{X}_{t+1}); \mathbf{Y}) = \hat{\mathbf{Y}} \in \mathbb{R}^{H\times W\times 2}$, where $\hat{\mathbf{Y}}$ is the predicted flow solution that shares the same resolution as the ground-truth.

In FlowNet [18], for example, we could interpret $\mathcal{F}_\theta$ as the set of convolutions and deconvolutions along the network that produce features, and $\mathcal{G}_\alpha$ as the intermediate set of convolutions that transform at certain points the extracted representations into the final and intermediate optical flow predictions. In that example, during training a regression loss function is used to find a suitable model of parameters θ and α just using fine-grained information.

In our approach, we adapt the FlowNet architecture to use it as our $\mathcal{F}_\theta$, but account two different $\mathcal{G}_\alpha$ prediction functions, that will compose our Coarse-and-Fine components. A simple but effective way to account for these Coarse-and-Fine components is to branch $\mathcal{G}_\alpha$ into $\mathcal{G}_{class} = \hat{\mathbf{Y}}^{class}$ and $\mathcal{G}_{reg} = \hat{\mathbf{Y}}^{reg}$. Here, $\hat{\mathbf{Y}}^{class}$ stands for a *coarse* classification prediction over a small set of flow categories and $\hat{\mathbf{Y}}^{reg}$ stands for a fine-grained *regression* estimation to act over the coarse-centered solution space.

More specifically, we define $\mathcal{G}_{reg}$ as a clean 3×3 convolution kernel, which maps the input features to a 2-channel output flow prediction. On the other hand the classification part required more effort. We define $\mathcal{G}_{class}$ as a 3×3 convolution that maps the input features to K categories, followed by a soft-max operator that produce a probability for each category. We refer to this 'Coarse-and-Fine' approach as CaF.

The K categories are defined by projecting optical flow within the $[m_r, M_r]$ range, which bounds are empirically selected according to typical minimum (m_r) and maximum (M_r) flow values for this problem. Then this range is divided into K bins B_k such that:

$$B_k = \begin{cases} (-\infty, C_1 + \delta/2), & \text{if} \quad k = 1 \\ [C_k - \delta/2, C_k + \delta/2), & \text{if} \quad 1 < k < K \\ [C_K - \delta/2, +\infty), & \text{if} \quad k = K \end{cases}, \tag{5.1}$$

where $C_k = m_r + \delta(k-1),\ k \in \{1, \dots, K\}$ are the centroids of the bin categories. Notice that outbound motions are saturated and codified on the outer classes. Through this procedure the classification ground truth $\mathbf{Y}^{class}$ is also generated from the real ground truth $\mathbf{Y}^{reg}$, which is employed in the regression task.

During training θ and α parameters are adjusted via standard end-to-end back-propagation, guided by the following coarse-and-fine loss function:

$$\begin{aligned}\mathcal{L}_{CaF}(\mathcal{G}(\mathcal{F}(\mathbf{X}_t, \mathbf{X}_{t+1})); \mathbf{Y}) &= \mathcal{L}_{CaF}(\hat{\mathbf{Y}}; \mathbf{Y}) \\ &= \mathcal{L}_{coarse}(\hat{\mathbf{Y}}^{\,class}, \mathbf{Y}^{\,class}) + \lambda \mathcal{L}_{fine}(\hat{\mathbf{Y}}^{\,reg}, \mathbf{Y}^{reg}).\end{aligned} \tag{5.2}$$

In our approach, $\mathcal{L}_{fine}$ is a standard regression loss in the form of ℓ_2-norm. For $\mathcal{L}_{coarse}$, we use a slightly modified multi-class Weighted Cross Entropy (WCE) loss [23], similar than the one used for the binary case classification in Chapter 2, such that:

$$\begin{aligned}\mathcal{L}^{WCE}(\mathcal{G}_{class}(\mathcal{F}(\mathbf{X}_t, \mathbf{X}_{t+1})), \mathbf{Y}^{class}) &= \mathcal{L}^{WCE}(\hat{\mathbf{Y}}^{\,class}, \mathbf{Y}^{class}) \\ &= -\sum_{h,w,k}^{H,W,K} \omega(\mathbf{Y}^{class}_{h,w}) Id_{[\mathbf{Y}^{class}_{h,w}]}(\log(\hat{\mathbf{Y}}^{class}_{h,w,k})),\end{aligned} \tag{5.3}$$

where $Id_{[\cdot']}(\cdot)$ is an index function that acts as a selector for the probability associated to the expected ground truth class bin, forcing that only predicted probabilities of correct real classes add to the loss. After observing the training set statistics we regularize the contribution of each class to the loss with $\omega(\cdot)$, which is a key factor to prevent the bias introduced by class imbalance due to some predominant vector flows. For each class, $\omega(k)$ is calculated proportionally to the inverse of the frequency of appearance of the k-th class in the dataset.

In practice, and without loss of generality, for the specific optical flow problem we further sub-divide $\mathcal{G}_{class}$ into its horizontal and vertical optical flow terms $\hat{\mathbf{Y}}^{class\text{-}H}$, $\hat{\mathbf{Y}}^{class\text{-}V}$ to simplify the representation of the problem. In other works, each sub-branch aggregates the flow motion in bins over one spatial dimension, either horizontal or vertical. The combination of both predictions therefore allow our coarse estimator to situate the new pixel over a 2D grid.

5.2.2 Simple Joint Coarse-and-Fine

The simpler operating mode of our CaF module, takes advantage of the training procedure by combining the classification and regression losses as show in Eq. 5.2. However, this naive approach does only actually use the regression output of the module, and therefore the coarse and fine predictions are not explicitly combined. In this way, the coarse component is used just as an auxiliary internal task to provide additional guidance and speed up to the training process.

Despite its simplicity, this method serves us to test and validate the importance of accounting for both coarse and fine components while training. In Table 5.1 we can see the obtained results of this approach under the name 'CaF'. A more detailed interpretation is given later in Section 5.3.1.

5.2.3 Explicit Joint Coarse-and-Fine

We propose a refinement of the simple CaF approach shown previously that explicitly represents the optical flow estimation by adding the output of the regressor to the classifier component. In this case, the regressor does not encode the whole optical flow, but just the fine details over the coarse solution. In other words, the classifier situates the solution in a coarse manner over one flow solution subspace, and the regressor provides the motion corrections (refinement) from this grid centroid, even with capacity to correct the initial coarse proposed solution. This process, that we call 'CaF-Full' in Table 5.1, is depicted by Fig. 5.3. This full representation has the advantage of reducing the search space of the fine component to a bounded area around zero, which makes the training convergence faster and leads to more accurate models, as it will be shown in Section 5.3).

In practice, the combination of the three components, i.e., $\hat{\mathbf{Y}}^{\,class\text{-}H}$, $\hat{\mathbf{Y}}^{\,class\text{-}V}$ and $\hat{\mathbf{Y}}^{\,reg}$ requires to map the discrete classification solutions back to a real value. For this reason, we introduce the 'DeCLASS' blocks that can be seen in Fig. 5.3, which output the flow centroid associated to a given class. A sample visualization of these DeCLASS blocks is shown in the top area of Fig. 5.1, with the names H and V for the Horizontal and Vertical partial solutions respectively. Afterwards both components are concatenated creating the final coarse estimation, to which the regression output is added.

5.3 Experiments

In this Section we show the benefits of using our joint Coarse-and-Fine reasoning scheme. For that, we incorporate both of our operational modules, 'CaF' and 'CaF-Full' to the well-established FlowNet architecture, which predict optical flow just using regression losses along its pipeline and evaluate the performance in terms of optical flow end-point-error (EPE). As additional baselines for comparison reasons, we also report results for a classification-only and regression-only predictions by modifying our CaF modules. Experiments are summarized in Table 5.1, where we show that our proposal decreases EPE by up to a 15%.

All the presented architectures, including basic FlowNet are trained from scratch under the exact same conditions. In this way we can certainly assure and measure the real performance boost that our approach produce. We use for training the FlyingChairs dataset [18], adopting the same splits than the original paper and a batch size of 8 pairs of RGB images as input. We perform slight data augmentation by mirroring upside-down and left-to-right the images with a 50% chance each. Additionally, we test our models on the MPI-Sintel [87] dataset without any further fine-tuning, which contains in total 1064 training and 564 test frames. All models are

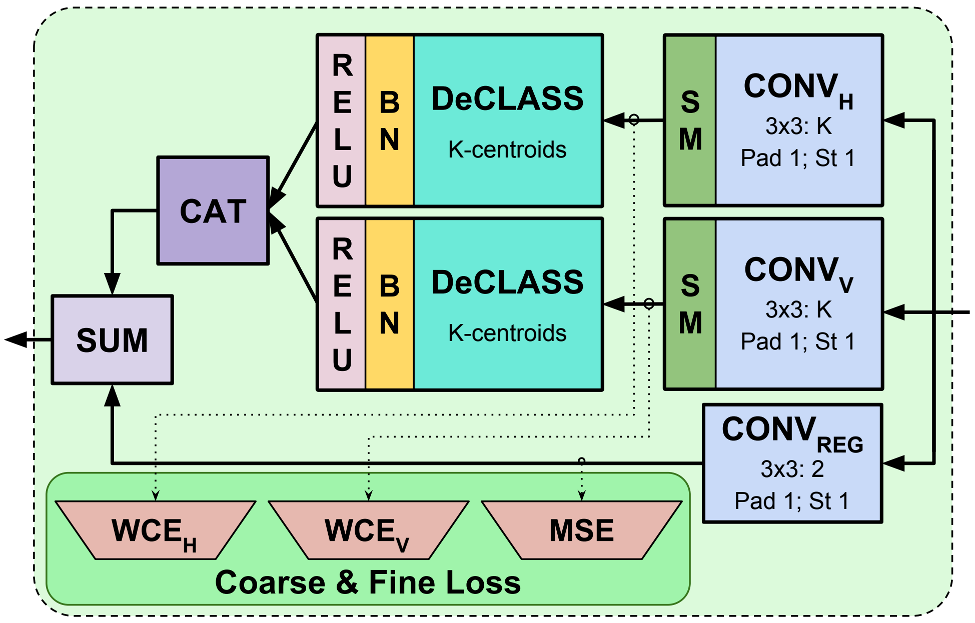


Figure 5.3: **Coarse-and-Fine module.** Each Coarse-and-Fine module solves one regression and two classification per-pixel problems. As inputs, the module receives the previous feature maps, and outputs more elaborated ones. Our CaF architecture is totally configurable by activating and deactivating internal modules and losses, achieving the proposed architectures for regression and classification baselines reflected in Table 5.1 and detailed in Section 5.3.1. For classification tasks, Softmax outputs (SM) are declassified obtaining the coarse solution. For regression, the corresponding convolutional block (CONV$_{\text{REG}}$) output is added at the end. Moreover, by combining the activation of the losses and internal modules, we can produce the two CaF flavors, 'CaF-c' and 'CaF-Full', detailed in Section 5.3.2 and seen in Table 5.1.

implemented in MatConvNet, initialized following He's method [66] and trained using Adam with the standard parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The training process is performed on a single NVIDIA K40 GPU for 600000 epochs. The learning rate is fixed to 10^{-3} during the first 300000 epochs and successively halved each 100000 epochs.

Following the FlowNet [18] architecture, we measure the network loss at 5 different resolution points on the expansive part as seen in Fig. 5.2. Yet, contrary to their approach we weight all these losses equally, so that focusing on the performance of the proposed joint approach more than on hyper-parameter tuning. For the coarse prediction, we bound the continuous flow space between -40 and 40 (parameters m_r and M_r respectively), and discretize the resulting sub-domain to create our K bin classes.

Three different experiments are presented here, attending to the number of classes created and which indirectly relates to the size of the pixel flow bins δ of Eq. 5.1. The bigger the number of classes configured for the coarse problem, the smaller the bin size for the pixel motions represented and therefore more concentrated around zero would be the regression solution. We choose to test our classification task with 5, 21 and 41 classes, each one representing flow bins with ranges of 20, 10 and 2 respectively.

5.3.1 Baselines

We next present the different baselines that we use for comparison purposes. Notice that all of them do have the CaF modules shown in Fig. 5.3 integrated on their architectures, but for each of the baselines we deactivate the corresponding elements that differentiate them. Notice that we use the same exact training conditions to keep a fair comparison.

Regression Baseline. Our regression baseline consists on a Batch-Normalized FlowNet [18] architecture trained from scratch under the previously defined conditions. This regression baseline is trained by deactivating the contribution of the classification modules to the final loss function as well as to the final output. This is done by setting to zero the λ regularizers of Eq. 5.2 on the classification modules and by deactivating the upper part of Fig. 5.3, in charge of the classification task.

The reported results observed under the 'Regression' entry in Table 5.1 are fairly close to the ones of the original FlowNet paper, although we used less data augmentation and avoided the hyper-parameter tuning in order to create a simple reproducible test environment in which to test our approach without bells and whistles. Notwithstanding, as will be shown later, the increase in performance of our joint approach is evident, as the training procedure is rigorous and fixed for all the methods.

Classification baseline. In addition to the regression baseline, we present a classification-only architecture, which results are reported in Table 5.1, tagged as 'Class-Kc', for $K = \{5, 21, 41\}$ classes. This classification baseline is trained by isolating the regression contribution to the network output, this is, deactivating the SUM block in Fig. 5.3 as well as the Mean Square Error (MSE) error of the Coarse-and-Fine loss during training, so that only the coarse classification components are used.

As observed in Table 5.1, using only classification produces worse results than the regression approach. However, it can be appreciated how the results get consistently better by increasing the number of classes. This effect stems from the fact that, as the class bins get smaller, the coarse solution obtained by pointing to their respective centroids gets more accurate. As a direct conclusion, we could state that the network capacity to perform accurate classifications is more than enough and could improve by increasing the number of classes. Yet, observing the modest performance increase between the model with 41 classes with respect to the one with 21 classes (double classes), we can state that there exist a complexity vs performance trade-off on the problem solution, for which we do not recommend the inclusion of more than 41 classes.

	Flying Chairs Validation	Sintel Train		Sintel Test	
		Clean	Final	Clean	Final
Regression	3.78 (100)	6.93 (100)	7.66 (100)	9.98 (100)	10.72 (100)
Class-5c	6.99 (184.7)	9.66 (139.4)	10.20 (133.1)	13.11 (131.3)	13.54 (126.3)
Class-21c	4.06 (107.3)	7.91 (114.1)	8.50 (110.9)	10.70 (107.1)	11.34 (105.8)
Class-41c	3.81 (100.7)	7.69 (110.87)	8.38 (109.3)	10.66 (106.7)	11.53 (107.5)
CaF-5c	3.55 (93.8)	6.85 (98.8)	7.54 (98.5)	9.98 (99.9)	10.69 (99.7)
CaF-21c	3.44 (90.9)	6.76 (97.5)	7.43 (96.9)	9.88 (98.9)	10.53 (98.2)
CaF-41c	3.47 (91.7)	6.75 (97.4)	7.39 (96.4)	9.77 (97.8)	10.48 (97.7)
CaF-Full-5c	3.25 (85.8)	6.85 (98.84)	7.72 (100.7)	9.74 (97.5)	10.51 (98.1)
CaF-Full-21c	3.23 (85.3)	6.75 (97.34)	7.59 (99.0)	9.57 (95.8)	10.28 (95.9)
CaF-Full-41c	3.18 (84.0)	6.51 (93.84)	7.28 (95.0)	9.42 (94.3)	10.18 (95.0)

Table 5.1: Evaluation of the End-Point-Error for the presented Coarse-and-Fine configurations. Values in parenthesis show the percentage difference with respect to the regression baseline based on FlowNet. Suffixes Kc indicate the number of classes used during training on Flying Chairs. Results over the Sintel Training and Test sets are also presented, showing the generalization of our method on unseen datasets.

5.3.2 Joint Coarse-and-Fine performance

We finally report experiments for the two flavors of our proposal, i.e., i) 'CaF', which is the regression baseline trained with the proposed coarse-and-fine loss function –turning the DeClass modules of Fig. 5.3 off, but keeping its measured errors on–, and ii) our 'CaF-Full', where the coarse-and-fine refinement modules are fully active, and therefore explicitly creating the estimated solution in that way.

According to the results, we can observe how significant is the performance boost produced by our approach in the trained networks. As it can be appreciated in the 'CaF' rows of Table 5.1 (rows 5–7), the addition of the combined loss function consistently decreases the EPE with respect to the basic regression solution. Moreover, by introducing our full Coarse-and-Fine architecture, 'CaF-Full' (rows 8–10), the performance is boosted up to a 15% in the Flying chairs validation set.

Regarding the number of classes of the coarse prediction the conclusion withdrawn on the previous Section sustains, as we observe a trend in the full architecture for the error decreasing when the number of classes is increased. This is more clear for the 'CaF-Full' models in terms of percentage with respect to the regression solution; having smaller class bins allows the fine prediction to recover misclassified pixels easier.

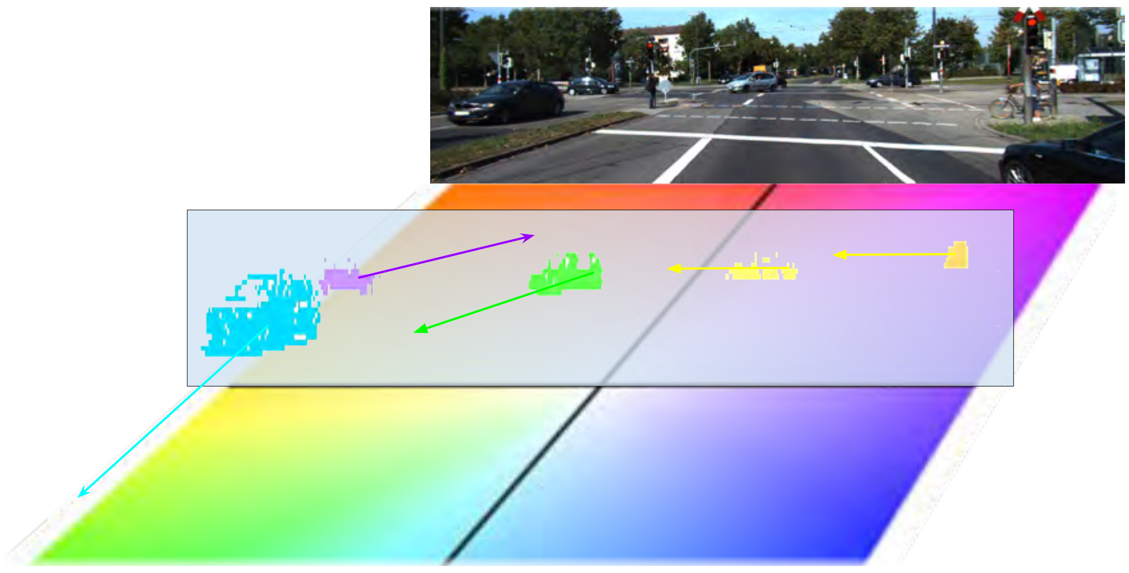
We further evaluate the generalization capacities of our approach by testing the models trained on FlyingChairs over unseen data from the MPI-Sintel [87] dataset without any further fine-tuning, as shown in the 'Sintel Train' and 'Sintel Test' columns of Table 5.1. Two versions of the dataset are tested, the 'Clean' with less visual effects, and the 'Final', which includes the full rendering with all effects such as blur due to camera depth of field, motion and atmospheric effects. Having into account the difficulty of not having seen this dataset at all during training, the observed results do confirm once more the benefits of our joint Coarse-and-Fine approach, as the same conclusions can be systematically obtained for both 'Sintel Train' and 'Sintel Test' splits.

5.4 Conclusions and Future Work

This chapter presented the benefits of using a joint Coarse-and-Fine representation for dense pixel-wise estimation task, such as optical flow, by casting the task to a joint classification and regression problem. Our novel representation has proven to speed up training convergence and to increase model accuracy when compared against CNN-based state-of-the-art methods and other baselines. We have experimentally demonstrated that this joint representation achieves its maximum potential by exploiting a new type of architecture, which expresses its prediction as the addition of a refinement real component to a coarse discrete approximation. Next steps could be focused on study the impact that complementary sources of information have in models accuracy and how to efficiently combine those sources.

Part III

How are Vehicles Moving?



High Level Dynamics: Motion estimation and segmentation using LiDAR data.

6 Understanding the Dynamics of Moving Vehicles

In this chapter we propose a novel solution to understand the dynamics of moving vehicles from the scene using only LiDAR information. In other words, we aim at obtaining the on-ground motion vector of each vehicle that is moving around us. For this purpose, we devise a deep-convolutional architecture fed with pairs of consecutive LiDAR scans. However, this is little input information for the complexity of the stated task, which additionally requires to disambiguate the proprio-motion of the 'observer' vehicle from that of the 'observed' ones. In order to ease the process, we assist the learning problem, at training time, by introducing a series of so-called 'pretext' tasks, which we define as prior solutions to simpler problems related to the final aimed result. In this way, for our motion estimation and segmentation problem, we include prior knowledge such as semantic *vehicleness* information as presented in Chapter 2, and motion information like the novel LiDAR-Flow feature shown in Chapter 4. We obtain promising results and show that by introducing these simpler pretext tasks, we are able to correctly guide the learning towards finding solutions for more complex problems. Additionally, we confirm that including image-based information only during training, allows improving the inference results of the network at test time, even when image data is no longer used.

6.1 Introduction

Capturing and understanding the dynamics of a scene is a paramount ingredient for multiple Autonomous Driving applications such as obstacle avoidance, map localization and refinement or vehicle tracking. In order to efficiently and safely navigate in our unconstrained and highly changing urban environments, Autonomous Vehicles require precise information about the semantic and motion characteristics of the objects of its surroundings. Special attention should be paid to moving elements such as vehicles, pedestrians or cyclists, mainly if their motion paths are expected to cross the direction of other objects or the own observer.

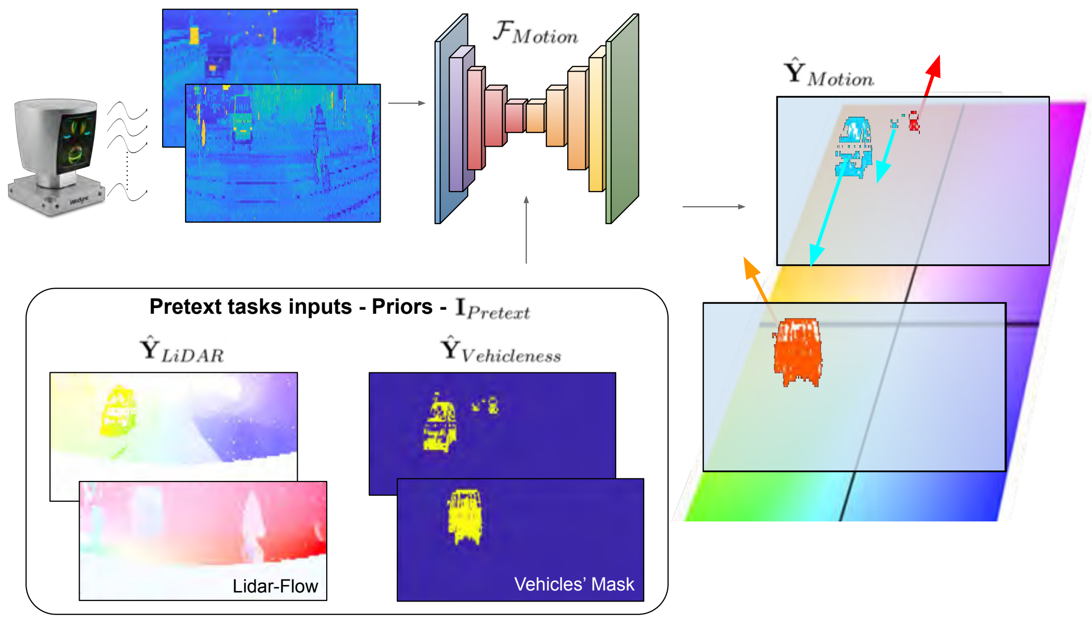

$\mathcal{F}_{Motion}$
$\hat{\mathbf{Y}}_{Motion}$
Pretext tasks inputs - Priors - $\mathbf{I}_{Pretext}$
$\hat{\mathbf{Y}}_{LiDAR}$
$\hat{\mathbf{Y}}_{Vehicleness}$
Lidar-Flow
Vehicles' Mask

Figure 6.1: **Vehicle motion vector estimation using LiDAR information.** We present a deep learning approach that, using only LiDAR information, is able to estimate the ground-plane motion vector of the surrounding vehicles. In order to guide the learning process we introduce prior semantic and pixel-wise motion information obtained from solving simpler pretext tasks, as well as odometry measurements.

This challenging task of estimating the dynamics of moving objects requires both from advanced acquisition devices and interpretation algorithms. Detecting independent dynamic objects reliably from a moving platform (ego vehicle) is by itself an arduous task. The propriomotion of the own 'observer' vehicle needs to be disambiguated from the actual motion of the rest of the 'observed' objects in the scene, which introduces additional difficulties.

In this chapter we detail a novel approach to infer the motion vector of dynamic vehicles over the ground plane using only LiDAR information. Given two consecutive LiDAR scans acquired from a moving vehicle, our objective is to detect the movement of the other vehicles in the scene which have an actual motion with respect to a 'ground' fixed reference frame. We tackle this challenging task by designing a novel Deep Learning framework which during inference is only fed with LiDAR data, although for training we consider a series of pretext tasks to guide the problem solution that can potentially exploit other data sources. Specifically, we introduce the *LiDAR-Flow* feature detailed in Chapter 4 that is learned by combining LiDAR and standard image-based optical flow, as well as a semantic *vehicleness* information as shown in Chapter 2. Apart from these pretext tasks –or priors–, we introduce knowledge about the ego motion by providing odometry measurements as inputs too.

A sketch of our developed approach is shown in Figure 6.1, where two different scenes are presented along with the corresponding pretext tasks that introduce the prior knowledge. The final output shows the predicted motion vectors for each scene, encoded locally for each vehicle according to the color pattern represented in the ground, as shown on the left-most area of the image. An ablation study with several combinations of the aforementioned pretext tasks shows that the use of the LiDAR-Flow feature produces very promising results towards achieving the overall goal of understanding the motion of dynamic objects from LiDAR data only. To summarize, the contributions of this chapter are:

- We design a new deep framework that, using only LiDAR information, is able to segment moving vehicles and predict their on-ground motion vector. For that we also propose a new dataset.
- We show how by employing prior information in the way of solving simpler pretext tasks such as vehicle segmentation or LiDAR-Flow helps on learning more complex problems.
- We perform an ablation study getting insights of which kind of prior informations benefits more towards achieving an accurate final result.
- We corroborate the hypothesis stated on Chapter 4 about the usefulness of training tasks in a cross-domain manner that do only employ one source of information on inference. In this way, we show how our LiDAR-Flow feature that employs data derived from the camera domain during training but only LiDAR information at inference, gains relevant importance.

The rest of this Chapter is organized as follows. In Section 6.2 we present related work on motion segmentation and prediction. Next, Section 6.3 details the proposed approach and the different pretext tasks included, showing how by exploiting this extra knowledge we can introduce prior information about the semantics and the dynamics to guide the complex learning problem. Finally, Section 6.4 shows our experimentation and an ablation study including different side assisting tasks on the solution.

6.2 Related Work

The crucial task of distinguishing whether or not an object is moving disjointly from the ego motion remains a challenge, despite the great advances introduced by deep learning technologies on perception problems. Some recent articles analyzing the motion of the scene from RGB images, which is also an ambitious problem recently tackled using CNNs, share ideas with the approach that we present in this Chapter. In [138], the authors train a convolutional network on synthetic data that taking as input the optical flow between two consecutive images, is able to mask independently moving objects. In our approach we go a step further and not only distinguish moving objects from static ones, but also estimate their motion vector on the ground plane reference. Other methods try to disentangle own and real objects' motions by inverting the problem. For instance [139] shows that a CNN trained to predict odometry of the ego vehicle, compares favorably to standard class-label networks on further trained tasks like scene and object recognition. This suggests that it is possible to exploit ego odometry priors to guide a CNN to disambiguate our movement from the free scene motion, which we do in Section 6.3.2.

The aforementioned works, though, are not focused on AD applications. On this setting, classic approaches segment the objects motion by minimizing geometrically-grounded energy functions. In this way, [122] assumes that outdoor scenes can decompose into a small number of independent rigid motions and jointly estimate them by optimizing a discrete-continuous CRF. Aside, [140] estimates the 3D dynamic points in the scene through a vanishing point analysis of 2D optical-flow vectors. Then, a three-term energy function is minimized in order to segment the scene into different motions.

LiDAR-based approaches to solve the vehicle motion segmentation task, have been lead by clustering methods, either motion- or model-based. The former ones, e.g. [141], estimate point motion features using RANSAC or similar methods, which are then clustered to help on reasoning at object level. Model-based approaches, e.g. [19], cluster vehicle points and retrieve those moving by matching them through frames, for example using a tracking algorithm.

As already shown in previous chapters, deep learning techniques are recently being also applied to the vehicle detection task over LiDAR information [19, 34, 37, 55, 57–59]. However, none of these approaches is able to estimate the movement of the vehicles in an end-to-end manner without further post-processing the output as we propose. The closest work is [142] which makes use of RigidFlow [143] to classify each point as *'non-movable'*, *'movable'*, and *'dynamic'*. In this Chapter we go a step further, and not only classify the dynamic elements of the scene, but also predict their motion vector.

Our approach also draws inspiration from progressive neural networks [144] in that we aim to help the network to fulfill a complex problem by solving a set of simpler intermediate 'pretext', auxiliary or side tasks. Our main difference is that we directly use the solutions to those side tasks as prior information providing extra knowledge as input to the final learning problem, so therefore we prefer the term 'pretext' here. For instance, in the problem of visual optical flow, [104] and [14] use semantic segmentation pretext tasks to improve the flow estimated for foreground objects. Similarly, during training, we also feed the network with prior knowledge about segmented vehicles on the point cloud information, but we include other additional pretext tasks such as our LiDAR-Flow as well as odometry information.

6.3 Deep LiDAR-based Motion Estimation

We next describe the proposed deep learning framework to estimate the actual motion of vehicles in a scene independently from the ego movement using only LiDAR information. We aim at finding the mapping function $\mathcal{F}_{Motion} : (\mathbf{I}_t,\ \mathbf{I}_{t+n},\ \mathbf{I}_{Pretext};\ \mathbf{Y}_{Motion}) \rightarrow \hat{\mathbf{Y}}_{Motion}$. We set $\mathcal{F}_{Motion}$ to be a Fully Convolutional Network that receives as main input front-projected LiDAR information (detailed in 2.4.1) from two different but temporary close frames of the scene $\mathbf{I}^t$, $\mathbf{I}^{t+n}$. As output $\hat{\mathbf{Y}}_{Motion}$, we want to obtain a motion mask where each pixel encodes the on-ground motion of the belonging object, this is, what is the direction of motion of that object over the ground floor. Additionally, we propose to guide the learning of this complex task by including prior information in the way of solutions to 'pretext' tasks such as vehicle segmentation from Chapter 2 and the LiDAR-flow representation obtained in Chapter 4. Thus, $\mathbf{I}_{Pretext} \in \{\hat{\mathbf{Y}}_{Vehicleness}, \hat{\mathbf{Y}}_{LiDAR}\}$. The learning process is carried out in a supervised way from the $\mathbf{Y}_{Motion}$ ground-truth, detailed in Section 6.3.1.

Following, we first introduce the dataset built from the 'KITTI Tracking' benchmark that has been specifically created to be used as ground truth in our supervised problem in Section 6.3.1. Since LiDAR information by itself is not enough to solve the proposed complex mission, in Section 6.3.2 we consider exploiting pretext tasks to introduce prior knowledge about the semantics and the dynamics of the scene to the main framework, which is defined in Section 6.3.3.

6.3.1 LiDAR Motion Dataset

In order to train our deep model, in this Section we define the input information ($\mathbf{I}_t$) and the ground truth ($\mathbf{Y}_{Motion}$) of the desired output ($\hat{\mathbf{Y}}_{Motion}$) to compare the learned estimations.

The simpler input data we use consists on the concatenation of two consecutive LiDAR front-projections featuring the ranges and reflectivity measured, as the one shown in Chapter 2. In this way, our main inputs $\mathbf{I}^t$, $\mathbf{I}^{t+n} \in \mathbb{R}^{64\times448\times2}$, are both generated as specified in 2.4.1 for $\mathbf{I}_{FR}$. For easier comprehension, we here omit the $(\cdot)_{FR}$ subindex.

To build the ground truth that represents the per-pixel on-ground motion of the elements of the scene, we make use of the annotated 3D bounding boxes of the 'KITTI Tracking' dataset [6], which provides diversity of motion samples. As vehicles still move on the ground plane, we express its motion as a 2D vector over the Z/X plane, with Z being our forward ego direction and X its transversal one. Thus, for each time t we define our ego-vehicle position as $O_t \in \mathbb{R}^2$, i.e the 'observer' position. Considering there are K vehicles moving in the scene at each moment, we define each vehicle centroid (C) seen from the 'observer' reference frame as ${}^{O_t}C_{t,k} \in \mathbb{R}^2$ where $k = 1 \ldots K$. For a clearer notation, we show here a use case with just one free moving vehicle, omitting therefore the k index from now on. As both the 'observer' and the other vehicle

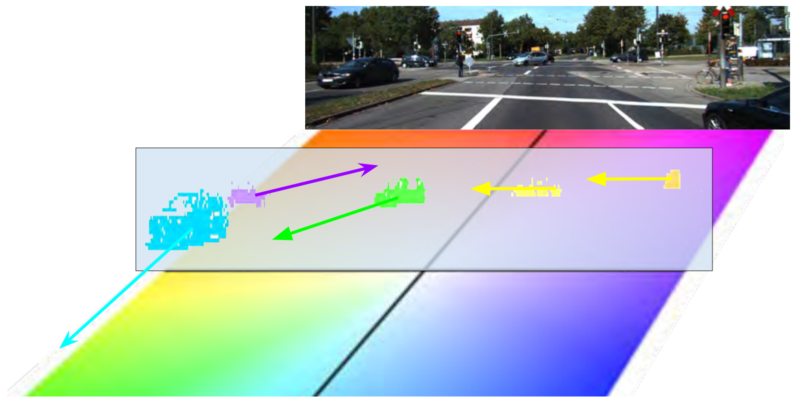


Figure 6.2: **Motion dataset representation**. LiDAR-based vehicle motion representation of our motion dataset. The output we seek to learn represents the motion vectors of the moving vehicles, colored here with the shown pattern attending to the motion angle and magnitude.

are moving, we will see each time the free vehicle centroid ${}^{O_{t+n}}C_{t+n}$ from a different position O_{t+n}. Therefore, in order to get the real displacement of the object in the interval $t \rightarrow t+n$ we transform this last observation to our previous reference frame O_t, obtaining ${}^{O_t}C_{t+n}$.

Let us denote our own frame displacement as ${}^{O_t}T_t^{t+n}$, which can roughly be estimated by the differential ego-odometry measurements. Then, the vehicle transformed position is simply:

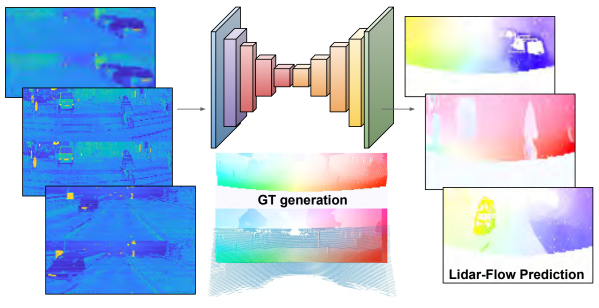
$$
{}^{O_t}C_{t+n} = ({}^{O_t}T_t^{t+n})^{-1} \cdot {}^{O_{t+n}}C_{t+n} \ , \tag{6.1}
$$

and the on-ground motion vector of the free moving vehicle in the analyzed interval can be calculated just as ${}^{O_t}C_{t+n} - {}^{O_t}C_t$.

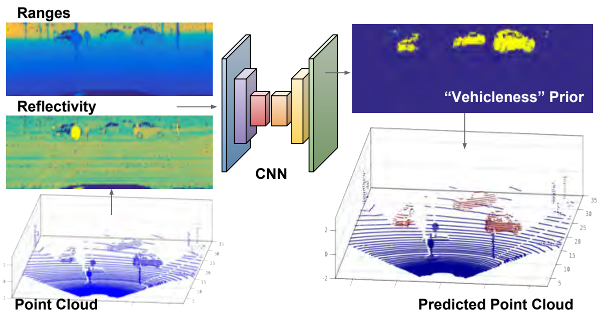
Notice that the ground truth needs to be calculated in a temporal sliding window manner using the LiDAR scans from frames t and $t+n$ and therefore, different results will be obtained depending on the time step n. The bigger this time step is, the longer will be the motion vector, but it will be harder to obtain matches between vehicles.

In addition, some drift can happen as accumulation of errors from i) the KITTI bounding box annotations, ii) noise in the odometry measurements and iii) the transformation computations. These can make some static vehicles to be tagged as slightly dynamic. We therefore filtered the obtained moving vehicles setting as dynamic only the ones which displacement is larger than a threshold depending on the time interval n, and consequently, directly related to the minimum velocity from which we consider a movement. This threshold was experimentally set to 10Km/h.

Finally, we encode each vehicle motion vector according to its angle and magnitude according to the color-code used to represent optical flow. Figure 6.2 shows a frame sample of the described ground truth, with the color motion representation shown on the ground. In this way, we can see how the yellow vehicles represent that are moving towards our right, or the blue one is moving towards us. RGB image is shown just for visualization purposes.



(a) LiDAR-Flow prior, $\hat{\mathbf{Y}}_{LiDAR}$



(b) Semantic prior, $\hat{\mathbf{Y}}_{Vehicleness}$

Figure 6.3: **Pretext tasks used to guide the final learning.** a) LiDAR-Flow prior, obtained by processing pairs of frames LiDAR frames; b) Semantic 'vehicleness' prior, obtained by processing single frames through our deep-LiDAR vehicle detector.

6.3.2 Pretext Tasks

We guide the network learning towards the correct solution introducing prior knowledge obtained by solving other pretext tasks. This idea of solving increasing complexity tasks draws similarities from progressive networks [144]. With the same motivation, we introduce three kinds of additional information: a) a LiDAR-Flow motion prior to guide the network for finding matches between the two LiDAR inputs; b) a 'vehicleness' segmentation of the LiDAR input offering semantic concepts to help with focusing on the vehicles in the scene; c) the ego motion information based on the displacement given by the odometry measurements.

The motion prior $\hat{\mathbf{Y}}_{LiDAR}$ for matching inputs is given by stating a LiDAR-Flow feature as can be seen in Figure 6.3a. It is obtained in a similar manner than in Chapter 4, with the main difference that here the task is to predict this feature only in the LiDAR domain, without going further to hallucinate the dense optical flow similar to the one in the image domain. As for what refers to the process of dataset creation and network architecture for this LiDAR-Flow feature, it draws full similarities to what was shown in Section 4.3.

Semantic priors $\hat{\mathbf{Y}}_{Vehicleness}$ about the vehicles of the scene are introduced via the per-pixel classification problem shown in [19]. It is a simpler version than the one presented in Chapter 2 in the way that this one only uses the Front view LiDAR inputs ($\mathbf{I}_{FR}$) and network ($\mathcal{F}_{FR}$) of the architecture. An example of these predictions is shown in Figure 6.3b.

Finally, we introduce further information about the ego-motion in the interval $t \rightarrow t+n$. For this, we create a 3 channel matrix with the same size as the 2D LiDAR inputs where each 'pixel' triplet takes the values for the forward (Z) and transversal (X) ego-displacement as well as the rotation over the Y axis in the interval.

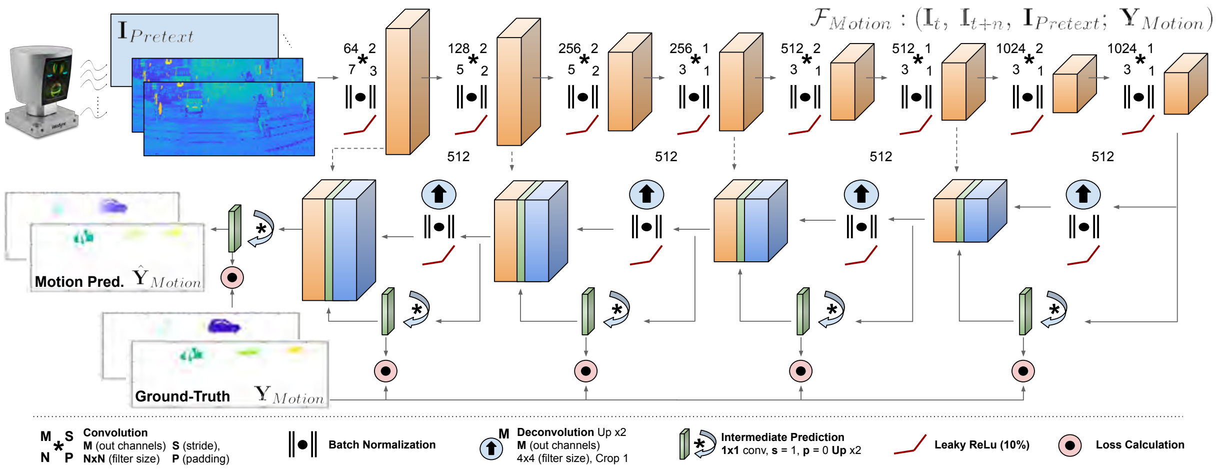

$\mathcal{F}_{Motion} : (\mathbf{I}_t,\ \mathbf{I}_{t+n},\ \mathbf{I}_{Pretext};\ \mathbf{Y}_{Motion})$
$\mathbf{I}_{Pretext}$
64 2 7 3
128 2 5 2
256 2 5 2
256 1 3 1
512 2 3 1
512 1 3 1
1024 2 3 1
1024 1 3 1
512
512
512
512
Motion Pred. $\hat{\mathbf{Y}}_{Motion}$
Ground-Truth $\mathbf{Y}_{Motion}$
Convolution M (out channels) S (stride), NxN (filter size) P (padding)
Batch Normalization
Deconvolution Up x2 M (out channels) 4x4 (filter size), Crop 1
Intermediate Prediction 1x1 conv, s = 1, p = 0 Up x2
Leaky ReLu (10%)
Loss Calculation

Figure 6.4: **Deep-Motion estimation architecture.** Deep Network architecture used to predict the motion vector of moving vehicles from just two LiDAR scans.

6.3.3 Deep-LiDAR Motion Network

We employed a Fully Convolutional Network architecture for estimating the rigid motion of each vehicle over the ground floor, which can be seen in Fig. 6.4. It draws inspiration from the encoder-decoder FlowNet [18] architecture, which is designed to solve a similar regression problem. However, we introduced some changes to further exploit the geometrical nature of our LiDAR information.

Standard FlowNet output size is a fourth of the input and bi-linearly interpolated in a subsequent step. This low resolution output is not feasible for our approach as our input is already very sparse and contains only few groups of LiDAR points belonging to moving vehicles thus, mid resolution outputs may not account for far vehicles seen by only a few points. To solve this issue, we introduce new deconvolutional layers in the decoder part of our architecture along with corresponding batch normalization (BN) and non-linearity (ReLU).

We additionally eliminate inner convolution and deconvolution blocks of FlowNet, for which the generated feature maps would reach a resolution of $1/64$ with our input sizes. Note that our LiDAR input data has per-se low resolution (64×448), and performing such an aggressive resolution reduction has shown to result in missing targets.

Similarly than FlowNet [18] our architecture performs feature concatenations between equally sized feature maps from the contractive and the expansive parts, which produce richer representations and allows better gradient flow. In addition, we also use a multi-resolution loss schema by obtaining predictions at different resolutions which are up-sampled and concatenated to the immediate upper feature maps. This guides the final estimation from early steps and allows the back-propagation of simpler gradients to the upper parts of the network.

6.4 Experiments

This section provides a thorough analysis of the performance of our motion-estimation deep framework. We detail our training procedure, the input combinations of LiDAR information and pretext tasks and finally show our results for the proposed objective. A video with the final results can be found in `https://youtu.be/9jn0A_AwX_I`.

6.4.1 Training Details

For training the presented architectures from both the main framework and the pretext tasks, we set n to 1, so that measuring the movement of vehicles between two time consecutive frames. All the networks are trained from zero, initializing them with the He's method [66] and using Adam optimization with the standard parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$.

The 'KITTI Tracking benchmark' contains a large number of frames with static vehicles, which results in a reduction of the number of samples from which we can learn motion patterns. We therefore use a distilled KITTI LiDAR-Motion dataset which contains 4953 frames with moving vehicles, and 3047 that either contain static vehicles or do not contain any vehicle at all. To balance the batch sampling and avoid a biased learning, we take for each batch 8 frames containing movement and 2 that do not. For training all models we left for validation the sequences 1, 2 and 3 of our distilled KITTI Training dataset, which results in 472 samples with motion and 354 without. As our training samples represent driving scenes, we perform data augmentation providing only horizontal flips with a 50% chance, in order to preserve the strong geometric LiDAR properties from the front point cloud projection.

The training process is performed on a single NVIDIA 1080 Ti GPU for $400,000$ iterations with a batch size of 10 Velodyne scan pairs per iteration. The learning rate is fixed to 10^{-3} during the first $150,000$ iterations, after which, it is halved each $60,000$ iterations. As learning loss for this problem, we use the Euclidean distance between the ground-truth and the estimated motion vectors for each pixel. We set all the intermediate calculated losses to equally contribute for the learning purposes.

6.4.2 Input Data Management for Prior Information

Depending on the data and pretext information introduced in our main network, different models have been trained; following, we provide a brief description.

Our basic approach takes as input a tensor of size $64 \times 448 \times 4$ which stacks the 2D LiDAR projected frames from instants t and $t+1$, so therefore $\mathbf{I}^t$, $\mathbf{I}^{t+1}$. Each projected frame contains values of the ranges and reflectivity measurements, as described in Section 2.4.1 of this Thesis.

For obtaining motion priors, the LiDAR data is processed through our featured LiDAR-Flow network, similar to the first module described in Section 4.3. It produces as output a two channel flow map where each pair *(u,v)* represents the RGB equivalent motion vector on the virtual camera alike plane as shown in Figure 6.3a. When incorporating this motion prior to the main network, the LiDAR-Flow map is concatenated with the basic input to build a new input tensor containing 6 depth channels.

In order to add semantic prior knowledge, we separately process both LiDAR input frames through our learned vehicle segmentation network [19], similar to the Front-branch described in Section 2.4.2. The obtained outputs indicate the predicted probability of each pixel to belong to a vehicle, so what we called 'vehicleness' information. When employed, this data is further concatenated with the raw LiDAR input plus the LiDAR-Flow maps, yielding a tensor with a depth of 8 channels.

Finally, for introducing the odometry information as well, three more channels are concatenated to the stacked input, resulting in a tensor of depth 11.

6.4.3 Results

Table 6.1 shows a quantitative analysis of our approach where we use the End-Point-Error (EPE) as metric. We compare the performance of our framework against the results produced by two baseline models, i.e., *error@zero* and *error@mean*. The *error@zero* baseline assumes a *zero* regression estimation, so that sets all the predictions to zero as if no solution were given. The *error@mean* baseline measures the End-Point-Error (EPE) from a mean-motion solution, calculated as the mean dataset motion.

Notice also that in our dataset only a few LiDAR points per frame fall into moving vehicles, so therefore measuring the predicted error over the full image may not give us a correct notion about the accuracy of the answer. In this way errors generated by false negatives (i.e. points that are dynamic but considered as static without assigning them a motion vector) and false positives (i.e. points that are static but considered as dynamic assigning them a motion vector), would get weakened over the full image. To account for this fact, we additionally measure EPE over the real dynamic points only. Both measurements are indicated in Table 6.1 columns 'Full' and 'Dynamic-Only'. All the values presented in Table 6.1 are calculated at test time over the validation set only, which during the learning phase has never been used for training neither on the main network nor on the pretext tasks. Recall that during testing, all the final networks are evaluated only using LiDAR Data.

We named all combinations of input information a D for *Data*, F referring to the use of *LiDAR-Flow* prior, S for *Vehicle Segmentation* prior and O for *Odometry*. When combining different inputs, we express it with the $\&$ symbol between names; In this way, for example a

Table 6.1: Evaluation of our framework using several combinations of LiDAR and pretext samples. Errors are measured in pixels, as the end-point-error. As notation we use D for *Data*, F referring to the use of *Lidar-Flow* prior, S for *Vehicle Segmentation* prior and O for *Odometry*. $GT.F$ refers to the Ground truth optical flow introduced as prior, but extracted from RGB images.

	Full	Dynamic-Only
error@zero	0.0287	1.3365
error@mean	0.4486	1.5459
D	0.0369	1.2181
GT.F	0.0330	1.0558
GT.F & D & S	0.0234	0.8282
Pred.F	0.0352	1.1570
Pred.F & D	0.0326	1.1736
Pred.F & D & S	0.0302	1.0360
Pred.F & D & S & O	0.0276	0.9951

model named D & F & S has been trained using as input the standard LiDAR *Data*, plus the priors *Lidar-Flow* and *Vehicle Segmentation*. To account for the strength of introducing optical flow as motion feature for guiding the network, we also tested training with only the *LiDAR-Flow ground truth* ($GT.F$ rows in the table) as well as with a combination of flow ground-truth, semantics and LiDAR data. Both experiments show favorable results as can be appreciated by lower EPE error values in Table 6.1, being the second one the most remarkable.

At the light of the results several conclusions can be extracted. First, we can appreciate the use of only LiDAR information is not enough for solving the problem of estimating the motion vector of freely moving vehicles in the ground, as we can see how the measured dynamic error is close to the error at zero baseline (error@zero).

We trained our initial model using as input the ground truth LiDAR-Flow and obtained slightly better results. Even better results were obtained combining this LiDAR-Flow feature with the other LiDAR input and segmentation priors, as seen in the 5th row. However, this input is obtained from the RGB-based optical flow, which does not accomplish our *solo-lidar* goal.

For the rest of experiments we trained our network using the learned prior LiDAR-flow feature ('Pred.F' rows in the table), therefore eliminating any dependence on camera images. When comparing 'GT.F' and 'Pred.F' rows, we can clearly appreciate that our learned LiDAR-Flow feature only adds a 6% of error, which is a small cost for being able to not rely at all on camera information at inference time. As expected, it introduces some noise but it still allows us to get better results than using only LiDAR information, which suggest that flow notion is truly important in order to solve the major task. Furthermore, it can be appreciated that the consecutive addition of prior information consistently lowers the obtained error.

6.5 Conclusions

In this Chapter we have addressed the problem of understanding the dynamics of moving vehicles using only LiDAR information acquired by a vehicle which is also moving. Disambiguating the proprio-motion from other vehicles' motion poses a very challenging problem, which we have tackled using Deep Neural Networks and pretext tasks. The main contribution of the Chapter was to show how the inclusion of simpler problem solutions helps towards accomplishing more complex tasks. Moreover, we re-affirm the hypothesis stated on Chapter 4 and demonstrate that while at testing time the proposed Deep model is only fed with LiDAR scans, its performance can be highly boosted by exploiting other prior image information during training. We have introduced a series of pretext tasks for this purpose, including semantics about the 'vehicleness' and an optical flow texture built from both image and LiDAR data. The results we have reported are very promising and demonstrate that a network can be guided towards solving a higher level problem by introducing side pretext and simpler tasks. Moreover, we can conclude that exploiting image information only during training really helps the LiDAR-based deep architecture. For future work, we left the further exploitation of introducing other image-based priors during training, such as the semantic information of all object categories in the scene and dense depth obtained from images.

7 Conclussions and Future Work

In this PhD dissertation we explored new approaches for solving scene understanding tasks in driving situations using Deep Learning methods that take as inputs only 3D LiDAR point clouds.

To this end, in part I of this document we have focused on analyzing the driving scene from a static perspective, this is, using only independent frames.

In Chapter 2 we proposed a 2D Front (FR-V) and a Bird's Eye view (BE-V) representations of the 3D point cloud, to serve as input for our convolutional models. We devised a dual-branch deep architecture that performs per-point classification on each of the input representations in order to segment vehicles from the scene. As the last step, we performed a late-fusion strategy over the model predictions and extracted the final 3D bounding boxes of the surrounding vehicles. We thoroughly evaluated our results on the KITTI Detection benchmark and showed that the inclusion of the dual branch architecture increased the system performance, as we outperformed or match other LiDAR-based approaches. In addition, the presented approach showed results comparable to the human based annotations in the near field at up to 32 meters.

In Chapter 3 we applied and further tested the previous dual-branch trained models on real uses-cases. Specifically, in the first case we proposed to pre-filter the *movable* objects from the input point cloud in order to obtain longer lasting features on map building tasks, thus enabling accurate re-localization over highly dynamic environments on subsequent days. We built a new dataset by recording a parking lot over different days and hours, and demonstrated that with our segmentation approach we were able to drastically increase the re-localization accuracy. Furthermore we demonstrated that we could build a much better map in several sessions by applying our *movable* objects filtering, obtaining a 50% more accurate map than using full point clouds. Finally, in the second use-case, we employed the extracted vehicles' 3D bounding boxes as initial observations for a multi-hypothesis extended Kalman filter tracker. We evaluated this task in the KITTI Tracking benchmark and observed that we could obtain similar tracking scores when employing our detection pipeline than the ones obtained when using an ideal detector based on the point-level ground truth created to train the networks.

The outcome of part I was the publication of two articles at international conferences [19,21] and one journal [20] which, at the moment of writing this document, is under revision.

For future research on the topics of this part, we left the exploration of multi-class problem segmentation over point clouds, detecting also pedestrians and cyclists. Further research could also focus on the use of deep learning techniques for end-to-end 3D bounding box extraction attending explicitly to context information.

In part II, we went a step further from single frame analysis and focused on understanding the dynamism of the scene. In this regard, in Chapter 4 we learned, from the 3D point cloud, a novel dynamic feature which resembles the optical flow extracted from RGB images. For that, we designed a deep architecture that solved the problem in an incremental manner. It firstly estimated a low sized equivalent optical flow in the LiDAR domain, which resolution was later increased up to that on the image domain and later refined in a final step. We developed new techniques that exploited RGB-derived optical flow as pseudo ground truth for training our initial low resolution LiDAR-Flow estimation training, but used at inference time only 3D LiDAR information. The results showed that the flow vectors estimated just from the LiDAR data were competitive with those computed by methods relying on high-resolution input images.

In Chapter 5 we explored the benefits of combining classification and regression learning problems to face the RGB optical flow estimation task in a joint coarse-and-fine manner. Our approach expressed the final optical flow estimation as the addition of a refinement component to a coarse discrete approximation, which proved to speed up training convergence and to increase model accuracy when compared against CNN-based state-of-the-art methods.

Outcomes from this part were two publications in international conferences [22, 23].

On the topic of generating motion features from LiDAR point clouds, there is still room for improvement in order to get more accurate optical flow predictions, which we left for future work. For example, it would be interesting to research on better refinement steps to improve results on foreground objects or even the inclusion of additional tasks such as semantic segmentation to focus the attention on those elements standing out. Additionally, it would be worth to compare at night driving sequences the behavior of our hallucinated LiDAR-Flow against an standard RGB optical flow algorithm.

Finally, in part III we gathered all the learned methods and tackled higher complexity tasks such as segmenting and estimating the motion of moving vehicles from our own moving perspective, again using only LiDAR information. We showed how the inclusion of simpler task solutions developed in the previous chapters, such as the 'vehicleness' semantics and an LiDAR-flow dynamics, helped towards accomplishing the higher level task of finding the on-ground motion vectors of elements in the scene. In this way we built a convolutional architecture for this task and reported promising results that demonstrated our guided learning hypothesis.

As outcome of this part, the publication [24] was presented in an international conference.

For future work, we proposed to introduce other image-based priors during training, as well as the explorations of other higher level tasks that could be approached by the proposed method. Although we have experimented with lower resolution sensors providing less 3D points in the point cloud [26], we consider it is a worth to explore future line of research, as those sensors are cheaper and commonly find in smaller autonomous robots.

7.1 Further Achievements

Besides the full list of publications presented in Section 1.7, further achievements have been accomplished during the period of this PhD, such as research stays, patents submitted and workshops organized, which are detailed in the following sections.

7.1.1 Research Stays

During this PhD, the author has done two industrial research internships working in related topics within two autonomous driving companies:

- Valeo, Schalter und Sensoren gmbh. (Germany)
  Hummendorfer Str. 74, 96317, Kronach.
  4 months, from April 2018 to August 2018.

- Toyota Research Institute. (United States)
  4440 El Camino Real, 94022, Los Altos, California.
  6 months, from June 2018 to December 2018.

7.1.2 Patents

The following patents, in which the author figures as inventor have been submitted by Valeo:

- “Deep LiDAR Semantic and Motion Segmentation to Improve Odometry”.
  Victor Vaquero, Kai Fischer and Dr. Stefan Milz.
  Entry date: 26.06.2018

- “Occlusion filter for depth maps.”
  Christian Witt, Martin Simon, Varun Ravi-Kumar, Dr. Stefan Milz and Victor Vaquero.
  Entry date: 22.05.2018

Three more invention disclosures are being written as a result of the research stay on Toyota Research Institute.

7.1.3 Workshops Organization

- V. Vaquero, R. Kiran, and S. Yogamani.
  "3D-DLAD: 3D data using Deep Learning for Autonomous Driving"
  https://sites.google.com/view/3d-dlad-iv2019/.
  IEEE Intelligent Vehicles Symposium (IV)
  June 9, 2019, Paris (France)

- V. Vaquero, R. Kiran and S. Yogamani.
  "DLAD-BP: Deep Learning for Autonomous Driving: Beyond Perception"
  https://sites.google.com/view/dlad-bp-itsc2019/
  IEEE International Conference on Intelligent Transportation Systems (ITSC)
  October 27, 2019, Auckland (New Zeland)

- V. Vaquero, R. Kiran and S. Yogamani.
  "3D-DLAD: 2nd Workshop on Deep Learning for Automated Driving"
  https://sites.google.com/view/3d-dlad-v2-iv2020/home
  IEEE Intelligent Vehicles Symposium (IV)
  June 23, 2020, Las Vegas (USA)

7.1.4 Talks and Seminars

- "Deep LiDAR Flow and Motion Segmentation for Autonomous Driving"
  Centre for Autonomous Systems, the University of Technology Sydney (UTS)
  May 31, 2018, Sydney (Australia)

- "Deconvolutional Networks for Point-Cloud Vehicle Detection and Tracking in Driving Scenarios"
  Institut de Robotica i Informatica Industrial, IRI (CSIC-UPC)
  October 5, 2017, Barcelona (Spain)

- "Low cost, robust and real time system for detecting and tracking moving objects to automate cargo handling in port terminals"
  Institut de Robotica i Informatica Industrial, IRI (CSIC-UPC)
  October 5, 2017, Barcelona (Spain)

- "Real Time People Detection Combining Appearance and Depth Image Spaces using Boosted Random Ferns"
  Institut de Robotica i Informatica Industrial, IRI (CSIC-UPC)
  November 16, 2015, Barcelona (Spain)

Bibliography

[1] R. Lanctot, "Accelerating the future: The economic impact of the emerging passenger economy," *Strategy analytics*, vol. 5, 2017.

[2] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, 2015.

[3] SAE International Surface Vehicle Recommended Practice, "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles," SAE Standard J3016, Rev. Jun. 2018.

[4] L. Li, R. Socher, and L. Fei-Fei, "Towards total scene understanding: Classification, annotation and segmentation in an automatic framework," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2036–2043, June 2009.

[5] L. Fei-Fei, A. Iyer, C. Koch, and P. Perona, "What do we perceive in a glance of a real-world scene?," *Journal of vision*, vol. 7, no. 1, pp. 10–10, 2007.

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

[7] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2012.

[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[10] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[11] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, 2017.

[12] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[13] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[14] M. Bai, W. Luo, K. Kundu, and R. Urtasun, “Exploiting semantic information and deep matching for optical flow,” in *European Conference on Computer Vision (ECCV)*, 2016.

[15] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, pp. 640–651, April 2017.

[16] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox, “Flownet 2.0: Evolution of optical flow estimation with deep networks,” in *Proc. CVPR*, 2017.

[17] A. Behl, O. H. Jafari, S. K. Mustikovela, H. A. Alhaija, C. Rother, and A. Geiger, “Bounding boxes, segmentations and object coordinates: How important is recognition for 3d scene flow estimation in autonomous driving scenarios,” in *IEEE International Conference on Computer Vision (ICCV)*, 2017.

[18] A. Dosovitskiy, P. Fischer, E. Ilg, P. Hausser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, “Flownet: Learning optical flow with convolutional networks,” in *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[19] V. Vaquero, I. del Pino, F. Moreno-Noguer, J. Solà, A. Sanfeliu, and J. Andrade-Cetto, “Deconvolutional networks for point-cloud vehicle detection and tracking in driving scenarios,” in *European Conference on Mobile Robotics (ECMR)*, 2017.

[20] V. Vaquero, I. del Pino, F. Moreno-Noguer, J. Solà, A. Sanfeliu, and J. Andrade-Cetto, “Dual-branch cnns for vehicle detection and tracking on lidar data,” *Submitted: Transactions on Intelligent Transportation Systems*, 2019.

[21] V. Vaquero, K. Fischer, F. Moreno-Noguer, A. Sanfeliu, and S. Milz, “Improving map re-localization with deep ‘movable’ objects segmentation on 3d lidar point clouds,” in *IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019.

[22] V. Vaquero, A. Sanfeliu, and F. Moreno-Noguer, “Hallucinating dense optical flow from sparse lidar for autonomous vehicles,” in *IEEE International Conference on Pattern Recognition (ICPR)*, 2018.

[23] V. Vaquero, G. Ros, F. Moreno-Noguer, A. M. Lopez, and A. Sanfeliu, "Joint coarse-and-fine reasoning for deep optical flow," in *IEEE International Conference on Image Processing (ICIP)*, 2017.

[24] V. Vaquero, A. Sanfeliu, and F. Moreno-Noguer, "Deep lidar cnn to understand the dynamics of moving vehicles," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[25] V. Vaquero, E. Repiso, A. Sanfeliu, J. Vissers, and M. Kwakkernaat, "Low cost, robust and real time system for detecting and tracking moving objects to automate cargo handling in port terminals," in *2nd Iberian Robotics Conference*, vol. 418 of *Adv. Intell. Syst. Comput.*, pp. 491–502, Springer, 2015.

[26] I. del Pino, V. Vaquero, B. Masini, J. Solà, F. Moreno-Noguer, A. Sanfeliu, and J. Andrade-Cetto, "Low resolution lidar-based multi-object tracking for driving applications," in *Robot 2017: Third Iberian Robotics Conference, Vol 694 of Advances in Intelligent Systems and Computing*, pp. 287–298, Springer, 2017.

[27] V. Vaquero, E. Repiso, and A. Sanfeliu, "Robust and real-time detection and tracking of moving objects with minimum 2d lidar information to advance autonomous cargo handling in ports," *Sensors*, vol. 19, no. 1, p. 107, 2019.

[28] H. Rashed, M. Ramzy, V. Vaquero, A. El Sallab, G. Sistu, and S. Yogamani, "Fusemodnet: Real-time camera and lidar based moving object detection for robust low-light autonomous driving," in *The IEEE International Conference on Computer Vision (ICCV) Workshops*, 2019.

[29] X. Ji, G. Zhang, X. Chen, and Q. Guo, "Multi-perspective tracking for intelligent vehicle," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 2, pp. 518–529, 2018.

[30] M. Y. Abualhoul, P. Merdrignac, O. Shagdar, and F. Nashashibi, "Study and evaluation of laser-based perception and light communication for a platoon of autonomous vehicles," in *IEEE Conference on Intelligent Transportation Systems*, pp. 1798–1804, 2016.

[31] J. Larson, K. Y. Liang, and K. H. Johansson, "A distributed framework for coordinated heavy-duty vehicle platooning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 419–429, 2015.

[32] R. Kohlhaas, T. Schamm, D. Lenk, and J. M. Zollner, "Towards driving autonomously: Autonomous cruise control in urban environments," in *IEEE Intelligent Vehicles Symposium*, pp. 109–114, 2013.

[33] C. You, C. Wen, C. Wang, J. Li, and A. Habib, "Joint 2-D-3-D traffic sign landmark data set for geo-localization using mobile laser scanning data," *IEEE Transactions on Intelligent Transportation Systems*, 2019. In Press.

[34] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," in *Robotics: Science and Systems XII*, Robotics: Science and Systems Foundation, 2016.

[35] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *30th IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6526–6534, 2017.

[36] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum PointNets for 3D object detection from RGB-D data," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 918–927, 2018.

[37] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4490–4499, 2018.

[38] Y. Yu, H. Guan, and Z. Ji, "Automated detection of urban road manhole covers using mobile laser scanning data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3258–3269, 2015.

[39] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3d lidar point cloud," in *IEEE International Conference on Robotics and Automation*, pp. 1887–1893, 2018.

[40] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 6517–6525, 2017.

[41] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar, "Focal loss for dense object detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. In press.

[42] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2018.

[43] K. O. Arras, Ó. M. Mozos, and W. Burgard, "Using boosted features for the detection of people in 2D range data," in *IEEE International Conference on Robotics and Automation*, pp. 3402–3407, 2007.

[44] B. Douillard, J. Underwood, N. Kuntz, V. Vlaskine, A. Quadros, P. Morton, and A. Frenkel, "On the segmentation of 3D lidar point clouds," in *IEEE International Conference on Robotics and Automation*, pp. 2798–2805, 2011.

[45] C. Mertz, L. E. Navarro-Serment, R. MacLachlan, P. Rybski, A. Steinfeld, A. Suppé, C. Urmson, N. Vandapel, M. Hebert, C. Thorpe, D. Duggins, and J. Gowdy, "Moving object detection with laser scanners," *Journal of Field Robotics*, vol. 30, no. 1, pp. 17–43, 2013.

[46] O. M. Mozos, R. Kurazume, and T. Hasegawa, "Multi-part people detection using 2D range data," *Intl. Journal of Social Robotics*, vol. 2, no. 1, pp. 31–40, 2010.

[47] L. Spinello, K. O. Arras, R. Triebel, and R. Siegwart, "A layered approach to people detection in 3D range data," in *AAAI Conf. Artif. Intell.*, pp. 1625–1630, 2010.

[48] A. Petrovskaya and S. Thrun, "Model based vehicle detection and tracking for autonomous urban driving," *Autonomous Robots*, vol. 26, no. 2-3, pp. 123–139, 2009.

[49] A. Teichman, J. Levinson, and S. Thrun, "Towards 3D object recognition via classification of arbitrary object tracks," in *IEEE International Conference on Robotics and Automation*, pp. 4034–4041, 2011.

[50] D. Z. Wang, I. Posner, and P. Newman, "What could move? Finding cars, pedestrians and bicyclists in 3D laser data," in *IEEE International Conference on Robotics and Automation*, pp. 4038–4044, 2012.

[51] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmentation - supervoxels for point clouds," in *IEEE CS Conference on Computer Vision and Pattern Recognition*, pp. 2027–2034, 2013.

[52] D. Zeng Wang and I. Posner, "Voting for voting in online point cloud object detection," in *Robotics: Science and Systems XI*, Robotics: Science and Systems Foundation, 2015.

[53] J. Behley, V. Steinhage, and A. B. Cremers, "Performance of histogram descriptors for the classification of 3D laser range data in urban environments," in *IEEE International Conference on Robotics and Automation*, pp. 4391–4398, 2012.

[54] M. De Deuge, A. Quadros, C. Hung, and B. Douillard, "Unsupervised feature learning for classification of outdoor 3D scans," in *Australasian Conference on Robotics and Automation*, 2013.

[55] B. Li, “3D fully convolutional network for vehicle detection in point cloud,” in *IEEE International Conference on Intelligent Robots and Systems*, pp. 1513–1518, 2017.

[56] M. Engelcke, D. Rao, D. Z. Wang, C. H. Tong, and I. Posner, “Vote3deep: Fast object detection in 3d point clouds using efficient convolutional neural networks,” in *IEEE International Conference on Robotics and Automation (ICRA)*, 2017.

[57] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, “PointNet: Deep learning on point sets for 3D classification and segmentation,” in *30th IEEE Conference on Computer Vision and Pattern Recognition*, pp. 77–85, 2017.

[58] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in *31st Conference on Neural Information Processing Systems*, pp. 5099–5108, 2017.

[59] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. García, and A. De La Escalera, “BirdNet: A 3D object detection framework from LiDAR information,” in *IEEE Conference on Intelligent Transportation Systems, Proceedings*, pp. 3517–3523, 2018.

[60] S. Wirges, T. Fischer, C. Stiller, and J. B. Frias, “Object detection and classification in occupancy grid maps using deep convolutional networks,” in *IEEE Conference on Intelligent Transportation Systems, Proceedings*, pp. 3530–3535, 2018.

[61] N. Sun, Y. Zeng, Y. Hu, Y. Han, J. Ye, X. Li, and S. Liu, “RT3D: Real-time 3-D vehicle detection in LiDAR point cloud for autonomous driving,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3434–3440, 2018.

[62] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, “Backpropagation applied to handwritten zip code recognition,” *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[63] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *International Conference on Machine Learning*, 2015.

[64] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer, “SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5mb model size,” *arXiv preprint arXiv:1602.07360*, 2016.

[65] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, “Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud,” in *ICRA*, 2019.

[66] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proc. ICCV*, 2015.

[67] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[68] F. Moosmann and C. Stiller, "Velodyne slam," in *IEEE Intelligent Vehicles Symposium (IV)*, 2011.

[69] D. Wang, H. Liang, T. Mei, H. Zhu, J. Fu, and X. Tao, "Lidar scan matching ekf-slam using the differential model of vehicle motion," in *IEEE Intelligent Vehicles Symposium (IV)*, 2013.

[70] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: part i," *IEEE robotics & automation magazine*, vol. 13, no. 2, pp. 99–110, 2006.

[71] H. Dong and T. D. Barfoot, "Lighting-invariant visual odometry using lidar intensity imagery and pose interpolation," in *Field and Service Robotics*, pp. 327–342, Springer, 2014.

[72] S. Anderson and T. D. Barfoot, "Ransac for motion-distorted 3d visual sensors," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2093–2099, 2013.

[73] M. Bosse and R. Zlot, "Continuous 3d scan-matching with a spinning 2d laser," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4312–4319, 2009.

[74] J. Zhang and S. Singh, "Loam: Lidar odometry and mapping in real-time.," in *Robotics: Science and Systems*, vol. 2, p. 9, 2014.

[75] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4758–4765, IEEE, 2018.

[76] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based place recognition in 3d point clouds," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5266–5272, IEEE, 2017.

[77] R. Dubé, A. Cramariuc, D. Dugas, J. Nieto, R. Siegwart, and C. Cadena, "SegMap: 3d segment mapping using data-driven descriptors," in *Robotics: Science and Systems (RSS)*, 2018.

[78] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[79] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, pp. 194–220, 2017.

[80] B. Keni and S. Rainer, "Evaluating multiple object tracking performance: the CLEAR MOT metrics," *EURASIP Journal on Image and Video Processing*, vol. 2008, p. 246309, 2008.

[81] T. Meier and K. N. Ngan, "Automatic segmentation of moving objects for video object plane generation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 8, no. 5, pp. 525–538, 1998.

[82] G. R. Bradski and J. W. Davis, "Motion segmentation and pose recognition with motion history gradients," *Machine Vision and Applications*, vol. 13, no. 3, pp. 174–184, 2002.

[83] M. Zucchelli, J. Santos-Victor, and H. I. Christensen, "Multiple plane segmentation using optical flow.," in *BMVC*, vol. 2, pp. 313–322, 2002.

[84] E. Trulls, A. Sanfeliu, and F. Moreno-Noguer, "Spatiotemporal descriptor for wide-baseline stereo reconstruction of non-rigid and ambiguous scenes," in *Proc. ECCV*, 2012.

[85] T. Dang, C. Hoffmann, and C. Stiller, "Fusing optical flow and stereo disparity for object tracking," in *Proc. ITS*, 2002.

[86] R. Krishnamurthy, P. Moulin, and J. Woods, "Optical flow techniques applied to video coding," in *Proc. ICIP*, 1995.

[87] D. J. Butler, J. Wulff, G. B. Stanley, and M. J. Black, "A naturalistic open source movie for optical flow evaluation," in *Proc. ECCV*, 2012.

[88] N.Mayer, E.Ilg, P.Häusser, P.Fischer, D.Cremers, A.Dosovitskiy, and T.Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proc. CVPR*, 2016.

[89] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial intelligence*, vol. 17, no. 1-3, pp. 185–203, 1981.

[90] A. Bruhn, J. Weickert, and C. Schnörr, “Lucas/kanade meets horn/schunck: Combining local and global optic flow methods,” *International Journal of Computer Vision*, vol. 61, no. 3, pp. 211–231, 2005.

[91] T. Brox and J. Malik, “Large displacement optical flow: Descriptor matching in variational motion estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2011.

[92] F. Steinbrücker, T. Pock, and D. Cremers, “Large displacement optical flow computation without warping,” in *IEEE International Conference on Computer Vision - ICCV*, 2009.

[93] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, “Epicflow: Edge-preserving interpolation of correspondences for optical flow,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1164–1172, 2015.

[94] C. Bailer, B. Taetz, and D. Stricker, “Flow fields: Dense correspondence fields for highly accurate large displacement optical flow estimation,” in *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[95] D. Gadot and L. Wolf, “Patchbatch: A batch augmented loss for optical flow,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4236–4245, 2016.

[96] A. Bruhn and J. Weickert, “Towards ultimate motion estimation: combining highest accuracy with real-time performance,” in *IEEE International Conference on Computer Vision (ICCV)*, 2005.

[97] A. Wedel, D. Cremers, T. Pock, and H. Bischof, “Structure - and motion - adaptive regularization for high accuracy optic flow,” in *IEEE International Conference on Computer Vision (ICCV)*, 2009.

[98] D. Sun, S. Roth, and M. J. Black, “Secrets of optical flow estimation and their principles,” in *Computer Vision and Pattern Recognition (CVPR)*, 2010.

[99] L. Xu, J. Jia, and Y. Matsushita, “Motion detail preserving optical flow estimation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence - PAMI*, 2012.

[100] D. Sun, S. Roth, and M. Black, “A quantitative analysis of current practices in optical flow estimation and the principles behind them,” *International Journal of Computer Vision*, vol. 106, pp. 115–137, 2014.

[101] S. Hsu, P. Anandan, and S. Peleg, "Accurate computation of optical flow by using layered motion representations," in *Proc. ICPR*, 1994.

[102] F. Güney and A. Geiger, "Deep discrete flow," in *Proc. ACCV*, 2016.

[103] E. Trulls, I. Kokkinos, A. Sanfeliu, and F. Moreno-Noguer, "Dense segmentation-aware descriptors," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[104] L. Sevilla-Lara, D. Sun, V. Jampani, and M. J. Black, "Optical flow with semantic segmentation and localized layers," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[105] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman, "Patchmatch: A randomized correspondence algorithm for structural image editing," in *ACM Transactions on Graphics (ToG)*, vol. 28,3, p. 24, ACM, 2009.

[106] J. Zbontar and Y. LeCun, "Computing the stereo matching cost with a convolutional neural network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1592–1599, 2015.

[107] P. Weinzaepfel, J. Revaud, Z. Harchaoui, and C. Schmid, "Deepflow: Large displacement optical flow with deep matching," in *The IEEE International Conference on Computer Vision (ICCV)*, 2013.

[108] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid, "Deepmatching: Hierarchical deformable dense matching," *International Journal of Computer Vision*, vol. 120, no. 3, pp. 300–323, 2016.

[109] S. Zweig and L. Wolf, "Interponet, a brain inspired neural network for optical flow dense interpolation," in *Proc. CVPR*, 2017.

[110] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

[111] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The synthia dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3234–3243, 2016.

[112] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, “Squeezedet: Unified, small, low power fully convolutional neural networks for real-time object detection for autonomous driving,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, July 2017.

[113] X. Yue, B. Wu, S. A. Seshia, K. Keutzer, and A. L. Sangiovanni-Vincentelli, “A lidar point cloud generator: from a virtual world to autonomous driving,” in *ICMR*, pp. 458–464, ACM, 2018.

[114] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, “Deconvolutional networks,” in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[115] M. D. Zeiler and R. Fergus, “Visualizing and understanding convolutional networks,” in *European Conference on Computer Vision (ECCV)*, 2014.

[116] M. D. Zeiler, G. W. Taylor, and R. Fergus, “Adaptive deconvolutional networks for mid and high level feature learning,” in *International Conference on Computer Vision (ICCV)*, 2011.

[117] H. Noh, S. Hong, and B. Han, “Learning deconvolution network for semantic segmentation,” in *The IEEE International Conference on Computer Vision (ICCV)*, 2015.

[118] G. Ros, S. Stent, P. F. Alcantarilla, and T. Watanabe, “Training constrained deconvolutional networks for road scene semantic segmentation,” *arXiv preprint arXiv:1604.01545*, 2016.

[119] C. Dong, C. C. Loy, K. He, and X. Tang, “Image super-resolution using deep convolutional networks,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 2, pp. 295–307, 2016.

[120] A. Ranjan and M. J. Black, “Optical flow estimation using a spatial pyramid network,” in *Proc. CVPR*, 2017.

[121] T.-W. Hui, X. Tang, and C. Change Loy, “Liteflownet: A lightweight convolutional neural network for optical flow estimation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8981–8989, 2018.

[122] M. Menze and A. Geiger, “Object scene flow for autonomous vehicles,” in *Proc. CVPR*, 2015.

[123] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, “Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8934–8943, 2018.

[124] D. Sun, X. Yang, M.-Y. Liu, and J. Kautz, "Models matter, so does training: An empirical study of cnns for optical flow estimation," *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2019. to appear.

[125] Z. Yin, T. Darrell, and F. Yu, "Hierarchical discrete distribution decomposition for match density estimation," in *CVPR*, 2019.

[126] W.-C. Ma, S. Wang, R. Hu, Y. Xiong, and R. Urtasun, "Deep rigid instance scene flow," in *CVPR*, 2019.

[127] J. Y. Jason, A. W. Harley, and K. G. Derpanis, "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness," in *Proc. ECCV Workshops*, 2016.

[128] Z. Ren, J. Yan, B. Ni, B. Liu, X. Yang, and H. Zha, "Unsupervised deep learning for optical flow estimation," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[129] S. Meister, J. Hur, and S. Roth, "Unflow: Unsupervised learning of optical flow with a bidirectional census loss," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[130] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research*, 2013.

[131] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr, "Conditional random fields as recurrent neural networks," in *Proc. CVPR*, 2015.

[132] A. Vedaldi and K. Lenc, "Matconvnet – convolutional neural networks for matlab," in *Proc. ACM Multimedia*, 2015.

[133] M. Menze, C. Heipke, and A. Geiger, "Object scene flow," *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 2018.

[134] M. Menze, C. Heipke, and A. Geiger, "Joint 3d estimation of vehicles and scene flow," in *ISPRS Workshop on Image Sequence Analysis (ISA)*, 2015.

[135] J. Dai, K. He, and J. Sun, "Instance-aware semantic segmentation via multi-task network cascades," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3150–3158, 2016.

[136] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1925–1934, 2017.

[137] T. Schuster, L. Wolf, and D. Gadot, “Optical flow requires multiple strategies (but only one network),” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4950–4959, 2017.

[138] P. Tokmakov, K. Alahari, and C. Schmid, “Learning motion patterns in videos,” in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[139] P. Agrawal, J. Carreira, and J. Malik, “Learning to see by moving,” in *IEEE International Conference on Computer Vision (ICCV)*, 2015.

[140] J.-Y. Kao, D. Tian, H. Mansour, A. Vetro, and A. Ortega, “Moving object segmentation using depth and optical flow in car driving sequences,” in *IEEE International Conference on Image Processing (ICIP)*, 2016.

[141] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, “Motion-based detection and tracking in 3d lidar scans,” in *IEEE/RSJ International Conference on Robotics and Automation (ICRA)*, 2016.

[142] A. Dewan, G. L. Oliveira, and W. Burgard, “Deep semantic classification for 3d lidar data,” *IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2017.

[143] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, “Rigid scene flow for 3d lidar scans,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2016.

[144] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, “Progressive neural networks,” *arXiv preprint arXiv:1606.04671*, 2016.