MULTIMODAL PERCEPTION FOR AUTONOMOUS DRIVING

JORGE BELTRÁN DE LA CITA

A dissertation submitted in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in

Electrical Engineering, Electronics and Automation

Universidad Carlos III de Madrid

Advisors:

Arturo de la Escalera Hueso
Fernando García Fernández

Tutor:

Arturo de la Escalera Hueso

January, 2022

This thesis is distributed under license Creative Commons
Attribution-NonCommercial-NoDerivs 3.0 Spain

If you wish to make an apple pie from scratch,
you must first invent the universe.

— Carl Sagan

ACKNOWLEDGEMENTS

I hope the reader understands my decision to write these words in Spanish, my native language.

Ha llegado el momento de escribir estas líneas, finalizando así una etapa inolvidable de mi vida, tanto en el plano académico como en el personal. Más de cinco años estresantes, cansados, difíciles... pero que me han permitido aprender muchísimas cosas, visitar lugares maravillosos y conocer gente increíble, algunos de los cuáles considero ahora buenos amigos.

En primer lugar, he de agradecer el apoyo y la confianza de mis directores de tesis, Arturo y Fernando, durante todo este proceso. A pesar de los contratiempos hemos llegado hasta aquí, y eso en parte es gracias a vosotros. También me gustaría mencionar a Jose María y David, y a todos los compañeros que han pasado por el laboratorio a lo largo de estos años, los *bestias* y no tan bestias, que han llenado de grandes momentos el día a día en la universidad. En especial, me gustaría dar las gracias a Ángel e Irene, por estar ahí cuando más se ha necesitado.

En segundo lugar, quisiera nombrar a un grupo de personas con las que he compartido humedales, karaokes, y tardes memorables: Carlos, Noe, Irene y Nacho. En el caso de Carlos, agradecimiento doble, por ser parte fundamental de esta tesis y por las noches de paper y chupitos.

No puedo evitar mencionar a mi ~~amigo~~ hermano Javi, inmejorable compañero de aventuras y un pilar básico en mi vida. Gracias también a ti, Espe, por estos últimos años.

Quisiera teminar agradeciendo de corazón a las personas que más ilusión les hace el final de esta tesis: a mis padres. Gracias por vuestro apoyo incondicional y por sostenerme en los momentos más complicados. Sin vosotros esto no habría sido posible.

Jorge Beltrán
Leganés, enero de 2022

PUBLISHED AND SUBMITTED CONTENT

Some ideas, figures, and tables used in this thesis have appeared previously in the following publications:

Journal articles

- J. Beltrán, C. Guindel, A. De la Escalera, and F. García, "Automatic Extrinsic Calibration Method for LiDAR and Camera Sensor Setups," *IEEE Transactions on Intelligent Transportation Systems*, 2022, [Under review]. arXiv: 2101.04431

  Partially included in the thesis: Chapters 2 and 3. The inclusion in the thesis of material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

- A. Barrera, J. Beltrán, C. Guindel, J. A. Iglesias, and F. García, "BirdNet+: Two-Stage 3D Object Detection in LiDAR Through a Sparsity-Invariant Bird's Eye View," *IEEE Access*, vol. 9, pp. 160 299–160 316, 2021. DOI: 10.1109/ACCESS.2021.3131389

  Partially included in the thesis: Chapters 2 and 4. The inclusion in the thesis of material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

- V. Milanes, D. Gonzalez, F. Navas, I. Mahtout, A. Armand, C. Zinoune, A. Ramaswamy, F. Bekka, N. Molina, E. Battesti, *et al.*, "The tornado project: An automated driving demonstration in peri-urban and rural areas," *IEEE Intelligent Transportation Systems Magazine*, pp. 2–18, 2021, ISSN: 1941-1197. DOI: 10.1109/MITS.2021.3068067

  Partially included in the thesis: Chapter 5. The inclusion in the thesis of material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

Conference articles

- J. Beltrán, C. Guindel, I. Cortés, A. Barrera, A. Astudillo, J. Urdiales, M. Álvarez, F. Bekka, V. Milanés, and F. García, "Towards autonomous driving: A multi-modal 360° perception proposal," in *Proc. of the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2020, pp. 3295–3300. DOI: 10.1109/ITSC45102.2020.9294494

Partially included in the thesis: Chapter 5. The inclusion in the thesis of material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

- A. Barrera, C. Guindel, J. Beltrán, and F. García, "Birdnet+: End-to-end 3d object detection in lidar bird's eye view," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2020, pp. 2985–2990. DOI: 10.1109/ITSC45102.2020.9294293

  Partially included in the thesis: Chapter 4. The inclusion in the thesis of material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

- J. Beltrán, I. Cortés, A. Barrera, J. Urdiales, C. Guindel, F. García, and A. de la Escalera, "A method for synthetic lidar generation to create annotated datasets for autonomous vehicles perception," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 1091–1096. DOI: 10.1109/ITSC.2019.8917176

  Partially included in the thesis: Chapter 5. The inclusion in the thesis of material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

- 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 *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3517–3523. DOI: 10.1109/ITSC.2018.8569311

  Partially included in the thesis: Chapter 4. The inclusion in the thesis of material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

- C. Guindel, J. Beltrán, D. Martín, and F. García, "Automatic extrinsic calibration for lidar-stereo vehicle sensor setups," in *Proc. IEEE International Conference on Intelligent Transportation Systems*

(ITSC), 2017, pp. 674–679. DOI: 10.1109/ITSC.2017.8317829

Partially included in the thesis: Chapter 3. The inclusion in the thesis of material from this source is specified in a footnote to each chapter where an inclusion occurs. The material from this source included in this thesis is not singled out with typographic means and references.

OTHER RESEARCH MERITS

Some ideas, figures, and tables used in this thesis have appeared previously in the following publications:

ADDITIONAL PUBLISHED CONTENT

Journal articles

- L. C. L. Bianco, J. Beltran, G. F. López, F. Garcia, and A. Al-Kaff, "Joint semantic segmentation of road objects and lanes using convolutional neural networks," *Robotics and Autonomous Systems*, vol. 133, p. 103 623, 2020. DOI: 10.1016/J.ROBOT.2020.103623

Conference articles

- A. Astudillo, N. Molina, I. Cortés, I. Mahtout, D. González, J. Beltrán, C. Guindel, A. Barrera, M. Álvarez, C. Zinoune, V. Milanés, and F. García, "Visibility-aware adaptative speed planner for human-like navigation in roundabouts," in *Proc. of the 2021 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2021, pp. 885–890. DOI: 10.1109/ITSC48978.2021.9564451
- A. Barrera, J. Beltrán, C. Guindel, J. A. Iglesias, and F. García, "Cycle and semantic consistent adversarial domain adaptation for reducing simulation-to-real domain shift in lidar bird's eye view," in *Proc. of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 3081–3086. DOI: 10.1109/ITSC48978.2021.9564553
- I. Cortés, J. Beltrán, A. de la Escalera, and F. García, "Sianms: Non-maximum suppression with siamese networks for multi-camera 3d object detection," in *2020 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2020, pp. 933–938. DOI: 10.1109/IV47402.2020.9304685
- G. N. Doval, A. Al-Kaff, J. Beltrán, F. G. Fernández, and G. F. López, "Traffic sign detection and 3d localization via deep convolutional neural networks and stereo vision," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 1411–1416. DOI: 10.1109/ITSC.2019.8916958

- L. C. L. Bianco, A. Al-Kaff, J. Beltrán, F. G. Fernández, and G. F. López, “Joint instance segmentation of obstacles and lanes using convolutional neural networks,” in *Iberian Robotics conference*, Springer, 2019, pp. 229–241. DOI: 10.1007/978-3-030-35990-4_19

- J. Beltrán, C. Jaraquemada, B. Musleh, A. de la Escalera, and J. M. Armingol, “Dense semantic stereo labelling architecture for in-campus navigation.,” in *VISIGRAPP (5: VISAPP)*, 2017, pp. 266–273. DOI: 10.5220/0006131602660273

- P. Marın-Plaza, J. Beltrán, A. Hussein, B. Musleh, D. Martın, A. de la Escalera, and J. M. Armingol, “Stereo vision-based local occupancy grid map for autonomous navigation in ros,” in *11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications VISIGRAPP*, vol. 2016, 2016, pp. 701–706. DOI: 10.5220/0005787007010706

- B. Musleh, J. Beltrán, C. Jaraquemada, M. J. Gómez-Silva, N. Hernández, and J. M. Armingol, “Autocalibración de parámetros extrínsecos de sistemas estéreo para aplicaciones de tráfico,” in *XXXVII Jornadas de Automática*, 2016, pp. 675–682

RESEARCH VISITS

- Research visit to the Toronto Robotics and Artificial Intelligence Laboratory (TRAILab) group at the University of Toronto, Canada, under the supervision of Associate Professor Steve L. Waslander, from 01/04/2019 to 31/07/2019.

SUPERVISED ACADEMIC WORKS

Bachelor Theses

- I.M. Carpio, "Desarrollo de herramienta para etiquetado denso de escenas (Ground Truth)", Bachelor’s Thesis (TFG), Bachelor’s Degree in Industrial Electronics and Automation Engineering, Universidad Carlos III de Madrid, Oct. 2017.

- A. Elghobashy, "Scene understanding visualization for automated vehicles using Unity3D", Bachelor’s Thesis, Bachelor’s Degree in Computer Science and Engineering, German University in Cairo, Jun. 2018.

- C. Melero, "Implementación de funcionalidades para sistema de detección, clasificación y seguimiento de vehículos y peatones basado en información LiDAR", Bachelor’s Thesis (TFG), Bachelor's Degree in Industrial Electronics and Automation Engineering, Universidad Carlos III de Madrid, Jul. 2018.

- D. Martínez, "Desarrollo de interfaz para gestión y visualización de algoritmos de asistencia a la conducción en un vehículo inteligente", Bachelor's Thesis (TFG), Bachelor's Degree in Industrial Electronics and Automation Engineering, Universidad Carlos III de Madrid, Jul. 2018.
- B. Romero, "Desarrollo de software de etiquetado multimodal avanzado", Bachelor's Thesis (TFG), Bachelor's Degree in Industrial Electronics and Automation Engineering, Universidad Carlos III de Madrid, Oct. 2018.
- P. Muñoz, "Desarrollo de herramienta para generación de bases de datos de nubes de puntos anotadas para vehículos autónomos a partir de modelos sintéticos", Bachelor's Thesis (TFG), Bachelor's Degree in Industrial Electronics and Automation Engineering, Universidad Carlos III de Madrid, Oct. 2018.
- R. Romero, "Estudio sobre detección y clasificación de semáforos en imagen mediante técnicas de deep learning", Bachelor's Thesis (TFG), Bachelor's Degree in Industrial Technologies Engineering, Universidad Carlos III de Madrid, Jun. 2020.
- C. Piedrahita, "Detección de vehículos para monitorización del tráfico en tiempo real usando deep learning", Bachelor's Thesis (TFG), Bachelor's Degree in Industrial Electronics and Automation Engineering, Universidad Carlos III de Madrid, Jul. 2020.

ABSTRACT

Autonomous driving is set to play an important role among intelligent transportation systems in the coming decades. The advantages of its large-scale implementation –reduced accidents, shorter commuting times, or higher fuel efficiency– have made its development a priority for academia and industry. However, there is still a long way to go to achieve full self-driving vehicles, capable of dealing with any scenario without human intervention. To this end, advances in control, navigation and, especially, environment perception technologies are yet required. In particular, the detection of other road users that may interfere with the vehicle's trajectory is a key element, since it allows to model the current traffic situation and, thus, to make decisions accordingly.

The objective of this thesis is to provide solutions to some of the main challenges of on-board perception systems, such as extrinsic calibration of sensors, object detection, and deployment on real platforms. First, a calibration method for obtaining the relative transformation between pairs of sensors is introduced, eliminating the complex manual adjustment of these parameters. The algorithm makes use of an original calibration pattern and supports LiDARs, and monocular and stereo cameras. Second, different deep learning models for 3D object detection using LiDAR data in its bird's eye view projection are presented. Through a novel encoding, the use of architectures tailored to image detection is proposed to process the 3D information of point clouds in real time. Furthermore, the effectiveness of using this projection together with image features is analyzed. Finally, a method to mitigate the accuracy drop of LiDAR-based detection networks when deployed in ad-hoc configurations is introduced. For this purpose, the simulation of virtual signals mimicking the specifications of the desired real device is used to generate new annotated datasets that can be used to train the models.

The performance of the proposed methods is evaluated against other existing alternatives using reference benchmarks in the field of computer vision (KITTI and nuScenes) and through experiments in open traffic with an automated vehicle. The results obtained demonstrate the relevance of the presented work and its suitability for commercial use.

KEYWORDS: 3D object detection; computer vision; multi-modal fusion; convolutional neural networks; autonomous driving

RESUMEN

La conducción autónoma está llamada a jugar un papel importante en los sistemas inteligentes de transporte de las próximas décadas. Las ventajas de su implementación a larga escala –disminución de accidentes, reducción del tiempo de trayecto, u optimización del consumo– han convertido su desarrollo en una prioridad para la academia y la industria. Sin embargo, todavía hay un largo camino por delante hasta alcanzar una automatización total, capaz de enfrentarse a cualquier escenario sin intervención humana. Para ello, aún se requieren avances en las tecnologías de control, navegación y, especialmente, percepción del entorno. Concretamente, la detección de otros usuarios de la carretera que puedan interferir en la trayectoria del vehículo es una pieza fundamental para conseguirlo, puesto que permite modelar el estado actual del tráfico y tomar decisiones en consecuencia.

El objetivo de esta tesis es aportar soluciones a algunos de los principales retos de los sistemas de percepción embarcados, como la calibración extrínseca de los sensores, la detección de objetos, y su despliegue en plataformas reales. En primer lugar, se introduce un método para la obtención de la transformación relativa entre pares de sensores, eliminando el complejo ajuste manual de estos parámetros. El algoritmo hace uso de un patrón de calibración propio y da soporte a cámaras monoculares, estéreo, y LiDAR. En segundo lugar, se presentan diferentes modelos de aprendizaje profundo para la detección de objectos en 3D utilizando datos de escáneres LiDAR en su proyección en vista de pájaro. A través de una nueva codificación, se propone la utilización de arquitecturas de detección en imagen para procesar en tiempo real la información tridimensional de las nubes de puntos. Además, se analiza la efectividad del uso de esta proyección junto con características procedentes de imágenes. Por último, se introduce un método para mitigar la pérdida de precisión de las redes de detección basadas en LiDAR cuando son desplegadas en configuraciones ad-hoc. Para ello, se plantea la simulación de señales virtuales con las características del modelo real que se quiere utilizar, generando así nuevos conjuntos anotados para entrenar los modelos.

El rendimiento de los métodos propuestos es evaluado frente a otras alternativas existentes haciendo uso de bases de datos de referencia en el campo de la visión por computador (KITTI y nuScenes), y mediante experimentos en tráfico abierto empleando un vehículo automatizado. Los resultados obtenidos demuestran la relevancia de los trabajos presentados y su viabilidad para un uso comercial.

PALABRAS CLAVE: detección de objetos 3D; visión por computador; fusión multimodal; redes convolucionales; conducción autónoma

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

AAA	American Automobile Association
ABS	Anti-lock Braking System
ADAS	Advanced Driver Assistance Systems
AI	Artificial Intelligence
AP	Average Precision
BEV	Bird's Eye View
CNN	Convolutional Neural Network
DA	Domain Adaptation
DNN	Deep Neural Network
EV	Electric Vehicle
FC	Fully-Connected
FOV	Field of View
FPN	Feature Pyramid Network
GDP	Gross Domestic Product
GPS	Global Positioning System
GPU	Graphic Processing Unit
HFOV	Horizontal Field of View
HOG	Histogram of Oriented Gradients
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
IoU	Intersection over Union
NMS	Non-Maximum Suppression
NN	Neural Network
ODD	Operational Design Domain
R-CNN	Region-based Convolutional Neural Network
ReLU	Rectified Linear Unit

ROI	Region of Interest
ROS	Robot Operating System
RPN	Region Proposal Network
RV	Range View
SAE	Society of Automotive Engineers
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TP	True Positive
VRU	Vulnerable Road User
UAV	Unmanned Aerial Vehicle
WHO	World Health Organization
WRI	World Resources Institute

Part I

PROBLEM STATEMENT AND LITERATURE REVIEW

1 INTRODUCTION

The invention of the automobile at the end of the nineteenth century was a revolution for the mobility of human beings: longer and longer distances could be covered in less time, so people became more connected than ever before. The comfort and ease of use of the new motorized vehicles soon led to high demand for private cars. Since then, the number of units in use has increased steadily decade after decade, to become the most popular means of transport.

The extended use of cars brought many benefits to our lives, such as greater independence, more travel opportunities, or shorter commuting times. However, these advantages came at a non-negligible cost: as driving spread so did the number of injuries and deaths caused by road accidents.

To reduce this problem, major efforts are made on three levels. First, manufacturers evolve the design of vehicles over time so that they are safer for both the occupants and vulnerable road users (i. e., pedestrians and cyclists). Second, governments make heavy investments to upgrade road infrastructure. Last, strict driving regulations are approved to educate drivers' behavior and limit risk situations.

Nevertheless, there is still a key element in most traffic accidents that is not addressed by any of these measures: the human factor. Whenever a human being is the centerpiece of the control of the vehicle, the absence of casualties cannot be guaranteed, no matter how safe the car is or the road it is driving on.

Fortunately, the first Advanced Driver Assistance Systems (ADAS) began to be introduced in the mid-twentieth century, with the aim to assist the driver while driving. This trend, started with the adoption of the Anti-lock Braking System (ABS), was progressively developed through the integration of new technologies able to identify objects in blind spots, detect unintentional lane changes, or maintain a certain speed without becoming too close to the preceding vehicle, for instance. Nowadays, ADAS are critical components of our vehicles' safety systems. In fact, the European Union has recently made mandatory the implementation of ten of these technologies in all cars manufactured from July 2022 onwards.

Lately, the rapid development of sensors, artificial intelligence, and Graphic Processing Units (GPUs) have opened up the possibility to further reduce human intervention in the driving task. The promising results of Deep Neural Networks (DNNs) for scene understanding, planning, and control suggest that fully autonomous vehicles will be a reality in the near future.

The research presented in this document aims to push forward the field of multi-modal 3D object detection for automated vehicles, paying attention to the different stages in the perception pipeline. Thus, the tasks of sensor calibration, object classification and location, and deployment into real vehicles are studied. Detailed experimentation is provided to support the analysis of the proposed methods.

1.1 ROAD TRANSPORTATION CHALLENGES

Though modern vehicles mount a myriad of technologies to ease driving and prevent traffic accidents, road transportation systems still have some major challenges ahead.

1.1.1 *Traffic accidents*

According to World Health Organization (WHO), road fatalities rank in the top ten of world's most common causes of death [66], as shown in Figure 1.1. This tragedy is even more dramatic considering people of 5 to 29 years old, where it holds the first place. Moreover, an additional 20 to 50 million people are injured in traffic accidents every single year. From an economic perspective, studies estimate that the impact of road crashes accounts for around 3% of the global Gross Domestic Product (GDP).

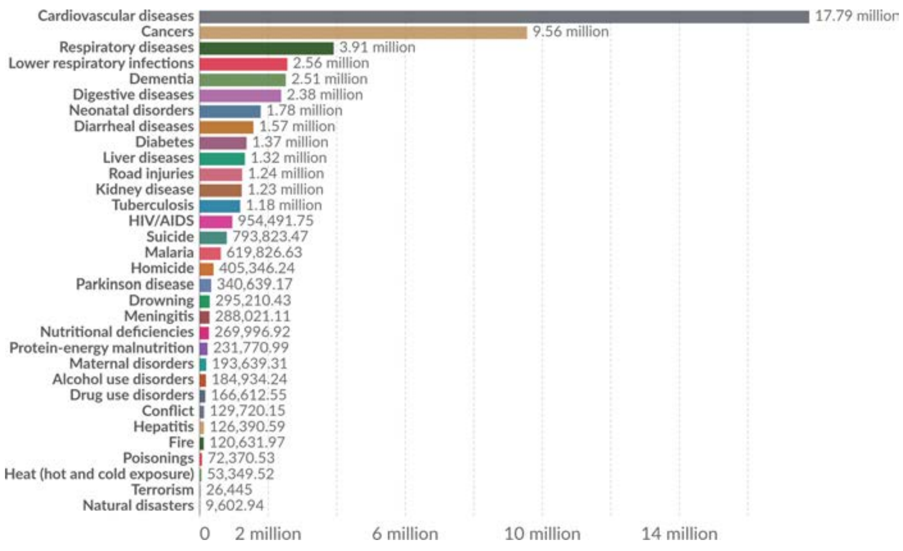


Figure 1.1: Worldwide deaths by cause in 2017 [147] (license CC BY 4.0)

Despite deaths caused by road crashes affect all countries worldwide, its impact is not proportionally distributed with the number of vehicles in use per world region. As can be observed in Figure 1.2, low- and medium-income areas have greater death rates than more developed countries, although the volume of registered cars is much lower. This situation may be explained by the differences in the quality and maintenance of the road infrastructure, the lifespan of vehicles, and the enforcement of traffic regulations [67].

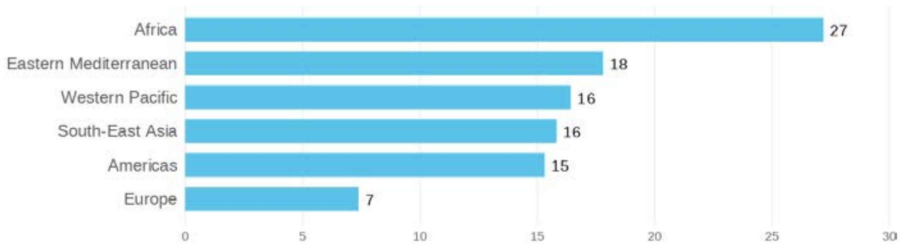


Figure 1.2: Death rate per 100K population by WHO region [44]

Recent studies on European road safety [45] show that among all traffic participants, cars are the kind of vehicle mostly involved in road crashes, participating in almost 50% of all fatal accidents (see Figure 1.3). However, data collected by the WHO indicates that Vulnerable Road Users (VRUs), such as pedestrians and cyclists, stand for the majority of deaths. This uneven distribution of fatalities makes it necessary to focus on improving car designs to reduce the damage to non-occupants in the event of a collision, as well as to identify the human factors associated to the higher involvement of four-wheeled vehicles in traffic accidents so that they can be prevented.

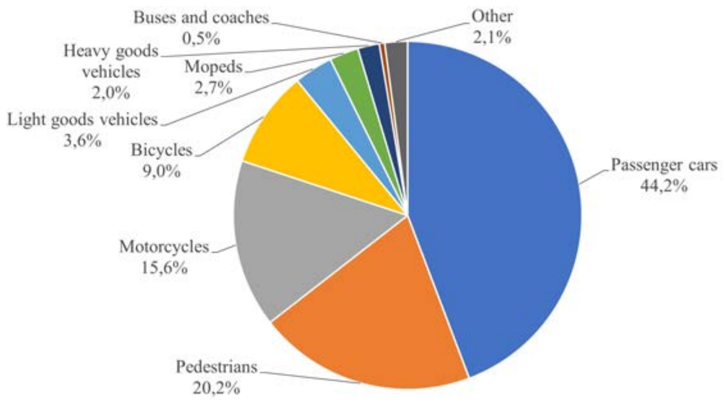


Figure 1.3: Road accident fatalities by type of user [45]

In this regard, the national report on road safety in Spain draws that 90% of car accidents are linked to driver errors [128]. Concretely, although the causes of most road accidents cannot be easily explained by a single reason, a set of concurrent human factors have been identified to stand out over the rest: driver distraction, speeding, and alcohol consumption are present in at least the 20-30% of fatal crashes, as shown in Figure 1.4. Even though weather and road conditions may also play a major role, the fact that the driver action is so often involved brings hope that autonomous driving will drastically reduce the number of traffic accidents and, thus, road fatalities.

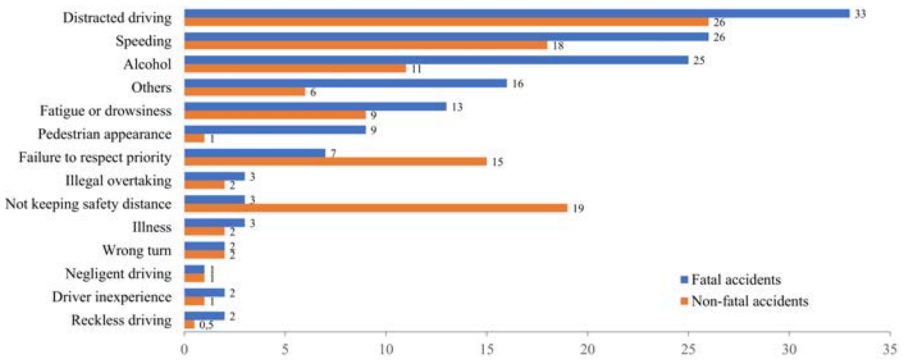


Figure 1.4: Presence (%) of concurrent human factors in road accidents in Spain in 2019 [128]

1.1.2 *Climate footprint*

The second major challenge of road transportation systems is related to climate. The WHO estimates that 7 million peoples die every year of respiratory diseases associated with air pollution [82]. Besides, 90% of the world population lives exposed to air whose quality is below the health standards.

The contribution of road transport to these numbers is not negligible at all. According to the World Resources Institute (WRI), vehicles are responsible for 11.9% of total greenhouse gas emissions, more than twice as much as all other means of transport combined [20]. Despite the magnitude, this amount only accounts for exhaust gases and does not consider those derived from the manufacturing process. A recent report from the ICCT[1] puts the number of deaths by tailpipe emissions from on-road vehicles at around 245.000 in 2015 [3], with attributable health damages of 625 billion US$.

In view of this information, it is understandable that there is a growing interest in promoting the development of Electric Vehicles (EVs), which will have zero emissions while driving. However, if the transition to EVs is not coupled with a decarbonization of the energy sector, air pollution will not be reduced but relocated to non-urban areas with power plants.

1.1.3 *Road networks congestion and traveling times*

An average commuter in the USA spends 15 days a year in their vehicle, 54 hours of which are stuck in a traffic jam [1]. However, the impact of road congestion goes far beyond traveling times. A study

1 International Council on Clean Transportation

from the Texas Transportation Institute [170] on mobility in the USA indicates that time spent in congested networks caused a total waste of 3.5 billion gallons of fuel just in 2019. For every commuter, being stuck in traffic meant 22 extra gallons (83 liters) of consumption and an annual average loss of 1.170$. Considering countrywide information, the total cost amounted to 20 billion dollars. Besides, the additional use of fuel derived from stationary or slow-moving traffic also has climate consequences. In the same year, an excess of 36 million tons of greenhouse gas emissions was produced.

Although poor road infrastructure is heavily associated with jams [48], most causes are linked to human actions. As shown in Figure 1.5, traffic accidents are responsible of 25% of total congestion. Furthermore, some other factors also point to people's limitations as being ultimately responsible. For instance, the weather does not often produce delays through its direct environmental action, but because it affects drivers' behavior. Similarly, a bad schedule of traffic lights signaling or road works may generate slow flows of vehicles.

Hence, even though road networks congestion constitutes a major challenge of road transportation systems which requires more research and investment at the infrastructure level, the automation of cars and the integration of technologies like vehicle-to-infrastructure communication may alleviate the problem and mitigate its consequences.

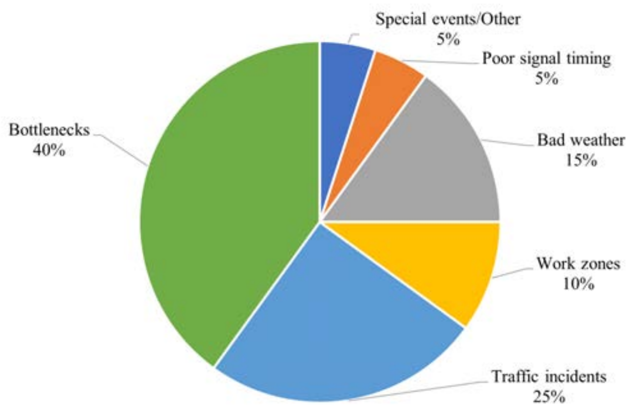


Figure 1.5: Causes of road networks congestion in the USA [48]

1.2 AUTOMATED DRIVING

The seek for increasing automation started with an emerging necessity to improve the safety of on-road vehicles to protect both the occupants and other road participants. In the 1950s, the first assistive systems were developed to ease some tasks of the driving operation and continue evolving over time to achieve the actual ADAS, which are usually found in commercial cars. In the last decade, the idea of fully autonomous vehicles seems no longer unattainable in near future thanks to the progress made in the field of machine learning, which allow processing the vast amount of data captured by modern sensors to obtain an increasingly accurate scene understanding. This accurate perception should permit to automate the navigation control and, thus, overcome the consequences of drivers' limitations.

1.2.1 *ADAS*

Since the first cars started to hit the roads, the protection of drivers, passengers, and people outside has been a major concern for society as a whole. Passive systems such as crumple zones or laminated windshields were soon mounted in commercial units to reduce injuries in case of an accident. More advanced features like seatbelts and airbags were later built to further improve the safety of vehicles occupants. The growing demand for these systems, along with the development of the necessary technology, naturally led to the creation of active systems tailored to prevent the crash from happening instead of reducing its consequences [16].

These active systems, also known as Advanced Driver Assistance Systems, are usually composed of a combination of hardware and software components and aim to aid the driver in different tasks, including parking, collision avoidance, or lane changes. Concretely, some of the most remarkable ADAS are:

Anti-lock Braking System (ABS): invented in 1978, ABS was built to prevent the locking of the wheels during braking, so the traction with the surface is not lost, and the driver can keep control of the vehicle.

Navigation system: early versions came to the market around 1990 with the aim to free the driver from routing tasks so that the efforts could be focused on the actual driving operation. Global Positioning System (GPS) sensors were used for localization.

Parking Assistance System (PAS): in the mid-1990s, ultrasonic sensors were mounted into commercial vehicles with the purpose of aiding drivers while parking. The system was tailored to guide the process by emitting a sound when the distance to walls or objects close to the parking lot becomes to short.

Adaptive Cruise Control (ACC): the development of the electronic brake and drive control, together with the reduction of the price of RADAR technology, led to the development of ACC in 1999. This ADAS allowed to partially automate some driving functions by keeping a selected speed while maintaining a safe distance to the next vehicle.

Lane Departure Warning (LDW) system: cars were equipped with real-time cameras in order to detect the lane markings and warn the driver in case of an involuntary change was about to happen. The first versions of the LDW appeared in the early 2000s.

Collision Avoidance System (CAS): introduced in 2010, these systems monitor the speed of the own vehicle and the preceding one and warn the driver to brake in case the distance is about to be reduced significantly so that the crash can be prevented. A combination of inexpensive LiDARs (close range) and RADAR (long-range) sensors are typically used.

As can be observed, the complexity of tasks these assistance systems are able to help with has notably increased over the years, endowing the automobiles with a higher degree of understanding of the traffic situation. These features have not only allowed increasing the safety of cars but are also key technologies for the development of future autonomous vehicles.

1.2.2 *Self-driving cars*

The advent of self-driving cars seems the natural evolution from ADAS as soon as more and more mature technologies become available. Following the advances in science, the degree of human intervention while driving may be gradually reduced until the need for a person behind the wheel is eventually eliminated. To provide a common taxonomy for this process, the former Society of Automotive Engineers (SAE), now SAE International, defines a set of six autonomy levels [83], summarized in Table 1.1. This classification is based on four criteria:

Dynamic driving task (DDT): all of the real-time operational and tactical functions required to operate a vehicle in on-road traffic, including lateral and longitudinal control, maneuvering, or environment monitoring.

Object and Event Detection and Response (OEDR): subtask of the DDT in charge of detecting the objects and events in the environment and executing the necessary response actions to perform the DDT (i. e.. lane change, overtaking).

DDT-fallback: the response after a system failure while executing the DDT or after exiting the Operational Design Domain (ODD). It may perform the DDT (user fallback) or achieve a minimal risk condition (user or system fallback).

Operational Design Domain (ODD): the set of conditions under a driving automation system is designed to operate. It may refer to limitations such as road kind, weather conditions, congestion level, etc.

Table 1.1: Summary of the different automation levels defined by SAE International [83]

Lvl.	Name	Drive	Monitor	Fallback	ODD
0	No Automation	Human	Human	Human	n/a
1	Driver Assistance	Human & System	Human	Human	Limited
2	Partial Automation	System	Human	Human	Limited
3	Conditional Automation	System	System	Human	Limited
4	High Automation	System	System	System	Limited
5	Full Automation	System	System	System	Unlimited

Technologies belonging to some of the automation levels defined by SAE can be found in present commercial vehicles. Lane-keeping assistance systems featured in many mid-height range units are usually part of the first level, as they apply lateral corrections to the vehicle to prevent a lane change, but the throttle and braking are performed by the driver. An evolution of this technology, namely Highway Driving Assist (HDA), adds the capability to longitudinally control the vehicle, thus being a case of Partial Automation. Other sound solutions such as Tesla Autopilot and Cadillac Super Cruise also qualify as Level 2. Recently, Japan authorities certified the Honda Legend as a conditional automated car thanks to its Traffic Jam Pilot system, being the first in the world to achieve it. This model is able to control brakes, acceleration, and steering to drive centered on a lane during a traffic jam while monitoring the surroundings. Human attention is not required throughout the process, but the system relies on the driver in case of failure. Although no other Level 3 vehicles are available in the market, other actors have directly targeted the next tier: high automation. This is the case of projects such as Waymo, Aurora, and nuTonomy robo-taxis, which are testing their technology by offering autonomous rides in predefined limited areas (geofencing). In all cases, an on-board safety driver is yet required to take control in case of a risky situation arises. Though the results of these prototypes are promising, there is still a long road ahead towards achieving full automation.

1.3 ON-BOARD PERCEPTION

The automation of cars entails the creation of a set of different technologies geared towards removing the need for a human behind the wheel. Consequently, all tasks traditionally carried out by drivers have to be replaced, including positioning, navigation, control, or environment monitoring. Although some of these issues are already solved, mimicking human-like perception is still a daunting challenge, as it requires not only an outstanding accuracy but also the adaptability to work in a wide range of driving situations such as extreme weather, the presence of unknown objects in the roadway, or unpredictable road conditions.

None of the advances on the automation level of on-road vehicles would have been possible without the recent evolution of sensors and perception algorithms. In the last decade, the appearance of efficient and reliable devices such as modern LiDARs, able to complement the information gathered by cameras with accurate 3D measurements of the surroundings, has laid the groundwork for the development of precise detection methods that enable safe navigation.

1.3.1 *Sensors*

The requirements for a fully autonomous vehicle (Level 5) entail the ability to drive in any given situation. This includes that the system should be capable of dealing with different weathers and illumination conditions, as well as supporting distinct road topologies without human intervention. Consequently, the selected sensor setups are usually composed of a set of heterogeneous devices whose joint use can provide coverage for all the potential scenarios. The combination of several modalities allows not only to switch between the sensors if the information of any of them becomes unavailable but to fuse their data in order to build more robust perception algorithms whenever is possible.

Even though every manufacturer has a different perception proposal and, thus, there is not a standard sensor setup to be equipped in an automated vehicle, most of them rely on the same kinds of sensors:

CAMERAS are passive sensors able to project the 3D world information into a 2D plane. According to the electromagnetic spectrum where they operate, cameras can be classified as visible (400-780nm) or infrared (IR) (780nm-1mm). Despite the lack of the third dimension, monocular devices capture rich cues about the visual appearance of objects. The ability to gather shape, color (or temperature), and texture information is sufficient for tasks such as the detection and classification of objects on the road or the identification of traffic signs. Other advantages such as their high resolution, real-time performance, and low cost make them a reasonable choice for perception in automated vehicles.

On the other hand, stereo cameras are composed of two lenses pointing to the same scene and mimic human vision to estimate the depth of objects in the scene by finding correspondences between the pair of images. These systems inherit the benefits of their monocular counterpart and are able to obtain information about the third coordinate. The accuracy of range measurements decreases with distance and is not comparable to the one provided by active sensors.

Regardless of the type, their operation is heavily affected by adverse weather conditions. As for illumination changes, although color cameras suffer in extreme bright or dark conditions, they are usually preferred to IR cameras as visual characteristics are generally more significant than those derived from thermal differences.

LiDAR is the acronym of LIght Detection And Ranging, and names a set of active sensors capable of measuring distances to objects by computing the round-trip time of the emitted light beams, typically belonging to the near-infrared spectrum (NIR). A rotating mirror is used to change the horizontal angle from where an array of laser pulses are triggered, usually spanning for 360°. The readings from a full revolution generate a 3D point cloud representing the geometry of the scene. The number of layers of commercial devices often ranges from 16 to 128, offering different vertical distributions.

While laser scanners are able to provide both accurate 3D shape and reflectivity information of the surroundings and have an operating distance up to 200m in the absence of occlusions, they also present some drawbacks. First, the captured point cloud gets warped when driving at high speeds, as the beams emitted at different horizontal angles do not share the same origin due to vehicle displacement. Moreover, objects composed of specular surfaces are difficult to be detected as the laser signals are deflected. Last, the vertical sparsity of the cloud leaves parts of the scene uncovered where small objects may be overlooked.

In the last years, research is focused on building solid-state LiDARs, which get rid of the rotating mirror and capture all readings at once, thus solving the cloud deformation problem of actual devices when mounted on moving platforms. Besides, unlike the preceding models, their shape does not compromise the vehicles' design.

RADAR stands for Radio Detection And Ranging and works similarly to LiDAR devices: a transmitter produces electromagnetic (EM) waves that are reflected whenever they collide with an obstacle. The returning waves are captured by the receiver, and the position of the target is estimated based on the direction of the reflected signal and its time of flight. In addition, through the *Dopler* effect, the speed of the detected target is also retrieved. Thanks to the use of EM technology, these active sensors can work in all weather, as the waves are mostly unaffected by rain, fog, or snow phenomena. On the contrary, radars are very sensitive to the geometry and reflectivity of objects, which may lead to false positive and false negative detections. Besides, their coarse horizontal resolution can occasionally hinder the identification of close objects at large distances [119].

ULTRASOUND sensors are the third kind of active range sensors usually mounted on vehicles. Their operating principle is identical to the one of the LiDAR, but ultrasonic sound waves are used instead. Due to signal attenuation and dispersion, the distance measurements are limited to a very close range. Nonetheless, the low cost of these sensors makes them an inexpensive solution for non-critical slow-speed maneuvers such as automatic parking.

A summary of the characteristics of the aforementioned sensors from a task suitability perspective on perception for autonomous driving is provided in Figure 1.6.

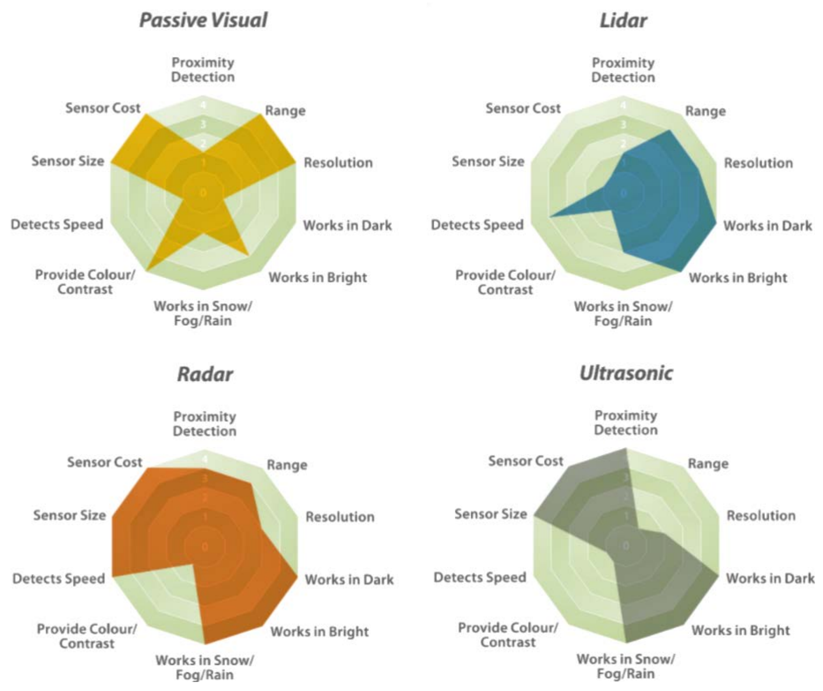


Figure 1.6: Features of exteroceptive automotive sensors. Figure from [180]

1.3.2 *Perception systems*

The advances on the capabilities of machine perception in the automotive sector are not only related to the development of better sensors. The evolution in hardware components such as GPUs, which are key for efficient image analysis, and the emergence of public annotated datasets have triggered an unprecedented leap forward in the development of more complex neural networks models able to increase the vehicle's awareness of the traffic situation.

Nonetheless, for self-driving cars to become a reality, on-board perception systems need to provide reliable information in real-time to enable safe and precise navigation and control. To this end, traffic signaling has to be properly identified, and detection and tracking of the rest of the traffic participants are required. Besides, accurate positioning of the vehicle on the road is mandatory.

Following this view, a set of separate tasks are usually defined to compose the final perception pipeline. However, end-to-end approaches where an artificial intelligence model takes all sensors information as input and provides the control commands as output are also being explored [21]. Even though the latter has shown some potential, this approach presents a significant disadvantage: it lacks explainability. Whenever a failure or unexpected behavior occurs, there is no possibility to blame any specific component, thus being harder to understand and correct the system limitation.

For this reason, the most accepted line of work adheres to the first methodology, having different dedicated processes whose partial outcomes are monitored. Some of the key components of an autonomous vehicle's perception pipeline are:

Localization lets the vehicle know its position and orientation in the world. This module typically relies on GPS and IMUs information, which offers sufficient precision in most navigation use cases. However, certain scenarios, such as urban canyons where the signal gets degraded by reduced satellite visibility, require the use of other localization techniques using, for instance, semantic road features captured by vision sensors. On-board perception sensors are also used for better positioning when High-Definition (HD) maps of the area are available.

Object detection deals with both the localization and classification of elements in the scene. It is usually responsible for providing an accurate 3D characterization of the objects around, which may range from other road users to traffic signaling and road marking. Parameters describing detections usually include category, position, orientation, and size.

Tracking stage is in charge of endowing previously detected objects, also known as agents, with a consistent identifier along time, as well as computing its velocity and most probable trajectory.

Although all these tasks are key to reach full automation, object detection stands as the most critical stage. On the one hand, the variability of obstacles to be found imposes high generalization requirements to any detection model. On the other hand, other tasks of the pipeline heavily rely on its output. For instance, HD maps should be updated to reflect persistent changes on elements of the road infrastructure identified at the object detection stage, and the tracking of agents cannot be carried out with an unstable performance of the detection process.

1.4 OBJECTIVES

This thesis falls in the field of perception systems for automated driving, focused on multi-modal object detection using information from LiDAR and camera devices. To this end, a holistic approach was followed, with the goal of producing significant contributions along the entire perception pipeline, from the sensor association stage to the deployment of algorithms on real platforms, with a particular interest in the development of DNNs frameworks for 3D object detection.

For that purpose, the following objectives are set:

1. To provide a practical and efficient solution for the problem of estimating the extrinsic parameters between common sensor setups used for on-board perception, as an accurate calibration is an essential requisite before fusing data from different devices.

2. To propose innovative representations of LiDAR data capable of mitigating the domain gap of detection algorithms caused by the differences in the characteristics of laser scanners while preserving salient features that enable the identification of multiple object categories.

3. To study the use of image-based network architectures to perform 3D Object detection using LiDAR information so that strict real-time and precision requirements of the automotive industry can be fulfilled.

4. To explore new LiDAR-camera fusion strategies to maximize the benefit from the joint use of geometrical and appearance features captured by these complementary modalities.

5. To address some challenges related to the performance of existing object detection approaches when used on real domains with significant differences from those found on well-known detection datasets.

6. To deploy and evaluate these advances in open-traffic scenarios, providing a deep understanding of their performance and usability in real use cases.

1.5 OUTLINE

This thesis is structured in three parts, which in turn are divided into different chapters. The content of each chapter is summarized below:

PART I: PROBLEM STATEMENT

- CHAPTER 1 has provided an introduction to the context in which this work is framed, stating the motivation of the research and including the set of specific objectives to be achieved throughout the thesis.

- Chapter 2 presents a detailed literature review of the most recent findings on perception for autonomous driving. The chapter comprises most popular driving datasets, methods for the extrinsic calibration of sensor setups, and comprehensive analysis of state-of-the-art deep learning approaches to both 2D and 3D object detection.

Part II: Proposed methods

- Chapter 3 addresses the extrinsic calibration problem between sensors commonly used for perception in automotive applications. An original method to automatically compute the relative position between camera and LiDAR devices is presented.
- Chapter 4 is devoted to discussing the suitability of the LiDAR BEV projection for 3D object detection. Different detection frameworks are proposed to exploit the features of a novel cell encoding, including single and multi-modal architectures. Comprehensive experimental results are provided to assess their respective performance.
- Chapter 5 tackles some of the domain adaptation issues of LiDAR-based 3D detection pipelines. An approach to generate derived training samples from existing annotated datasets is introduced, so available benchmarks can be used to deploy models on vehicles using different laser devices. Experimental analysis both in public datasets and a research platform in open traffic is presented.

Part III: Concluding remarks

- Chapter 6 includes the general conclusions of the work described in this document and poses different lines of research which may be explored to further extend the findings of this thesis.

2 RELATED WORKS

Perception systems have always played a key role in the path towards self-driving cars. Through the use of exteroceptive sensors, many driver assistance systems were developed to endow vehicles with a certain automation level by augmenting the degree of understanding of the traffic scene. The arrival of more advanced sensors, together with the increased computational capabilities of modern hardware, opened the door for building more complex algorithms able to exploit the new amount of information.

Artificial Intelligence (AI) models, which learn from data, soon benefited from these advances. Among the different machine learning categories, supervised methods have historically shown the most suitable for machine vision problems. These approaches, which are taught to establish a relationship between the inputs and the desired outputs, require a significant number of labeled samples to perform well over unknown data. In particular, the training of DNNs demands a vast amount of examples when the complexity of the true function is high, which is the case for automotive applications.

In this chapter, a survey of the different topics related to the matter of this thesis is presented. First, a revision of the major milestones in the history of self-driving technologies is given, with a focus on the evolution of on-board machine vision. Second, the most relevant public datasets which have been key to the development of perception stacks for autonomous cars are introduced. Then, a review of the latest works on automatic extrinsic calibration is provided. Finally, the state of the art in DNNs applied to 2D and 3D object detection is included.

2.1 AUTONOMOUS DRIVING

The dream of autonomous driving dates back to the beginning of the twentieth century. In 1925, the first driverless experiment took place. A vehicle with no driver at the wheel was teleoperated using radio signals sent from another car. Since then, many others pursued the idea of removing the human intervention while driving, most of them using ad-hoc infrastructure elements to guide vehicles. Nonetheless, it took six more decades for researchers to start building prototypes capable of acting over the vehicle control based on a real-time perception of the environment.

This chapter includes content from [8]

In 1989, Pomerleau et al. [138] presented a three-layer neural network able to generate control commands by using a synchronized pair of camera and rangefinder called ALVINN[1]. Their test platform, Navlab, was able to follow a road under certain field conditions. In 1995 at the same place, Carnegie Mellon University and Assistware Technology joined efforts to develop RALPH[2] [139], a perception method able to determine the road curvature and compute the lateral offset with respect to the lane center using a single image stream. A year later, their vehicle platform was able to drive from Washington DC to San Diego using this system with minimal human intervention.

In parallel, the *Programme for a European traffic of highest efficiency and unprecedented safety* (PROMETHEUS) was launched in 1987. The main agenda was focused on boosting the development of software and hardware to endow vehicles with intelligence and, hence, increase their automation. The project lasted for eight years and led to numerous advances. For instance, in 1994 Dickmanns et al. [40] presented an object and road detection and tracking algorithm to assist with the autonomous operation of a passenger car Mercedes 500 SEL. In the final demonstration, the car, equipped with four video cameras, was able to drive more than a thousand miles in a round-trip from Munich, Germany, to Copenhagen, Denmark, with an average intervention distance of 9km. Moved by this unprecedented success, Franke et al. [50] extended the driving domain to urban scenarios shortly after.

Albeit self-driving technologies improved significantly with these projects, the real take-off in this field occurred when the Defense Advanced Research Projects Agency (DARPA) of the US Department of Defense called for a set of three races where the competing cars had to operate autonomously. The $1M prize for the winner of the first competition, where vehicles were required to drive off-road more than 200km from California to Nevada, attracted the attention of the main American Universities and the automotive sector. In this first edition, no prize was given as all robots crashed early in the course, and the longest driven distance was less than 12km. It was assessed that the presented perception stacks were not mature enough to accurately detect the road and objects for a safe navigation.

The year after, the second DARPA Grand Challenge was held and, this time, the vehicles of five participating teams managed to complete the route. The Standford University ranked first, followed by two entries of Carnegie Mellon University (CMU). The winning robot Stanley [172, 173] mounted a myriad of sensors including five laser scanners, two radars, and a monocular color camera to build occupancy maps defining the road free of obstacles ahead of the vehicle. A similar setup was used by CMU vehicles, though the use of a stereo

1 ALVINN: Autonomous Land Vehicle In a Neural Network
2 RALPH: Rapidly Adapting Lateral Position Handler

camera provided dense depth and color information in the driving direction. A special mention is required for the sensor configuration of Team TerraMax [22, 130], which finished in the fifth position. Their vehicle made use of a four-layer LiDAR device, drawing the path for upcoming perception pipelines.

After the success of the second race, DARPA announced the last of the series for 2007: the Urban Challenge. In this edition, teams were asked to complete the 96km course in less than 6 hours. Apart from following the route, vehicles needed to deal with other road participants, including non-automated cars, drive through intersections, avoid obstacles, and obey traffic regulations. Boss vehicle [177, 178], developed by the CMU and General Motors, won the competition. The primary sensor of the robot was a Velodyne HDL-64, a 64-layer LiDAR providing dense range measurements in a 360° HFOV, which enables the detection and tracking of vehicles at long distances. A picture of the equipped robot can be observed in Figure 2.1. Although a set of 16 additional sensors were used to complement this device, no cameras were mounted. The lack of visual information was compensated with the use of a precise world model. As dynamic rules like traffic lights were not used, the benefits from the use of images could be disregarded to alleviate the overall processing time.

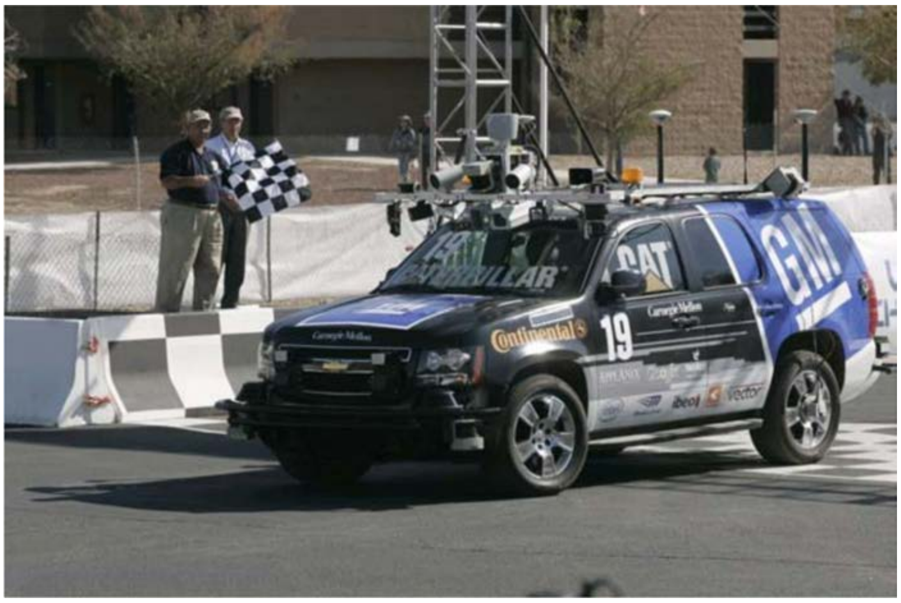


Figure 2.1: View of the sensor configuration of the Boss vehicle at the DARPA Urban Challenge in 2007 [178] © 2009 Springer

The impact of the DARPA races pushed forward the development of autonomous vehicle technologies, and many projects flourished in the subsequent years. In 2009, the Google car project was launched after hiring some of the talented researchers who participated in the Grand Challenges. A year later, Vislab's vehicle [17] was able to autonomously drive from Parma, Italy, to Shanghai, China, by following a leading car operated by a human. Due to the magnitude of the trip (over 13,000km), no world model was used. The robot was able to perform road and object detection making use of several LiDAR devices and vision systems. This same laboratory set a new milestone just four years later when they presented a real-time perception system capable of dealing with unrestricted urban scenarios in open traffic under the PROUD project [24]. The sensing devices used by the BRAiVE prototype are displayed in Figure 2.2.

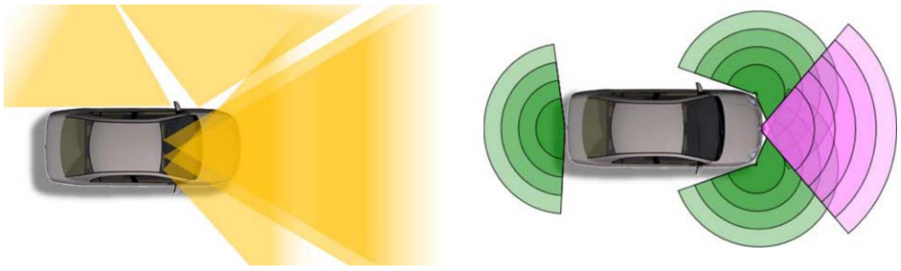


Figure 2.2: Sensor setup of the BRAiVE platform for the PROUD test[24], composed by cameras (in yellow), single-line (in green) and four-layer (in purple) LiDAR devices. © 2015 IEEE

In 2011, Team AnnieWAY [59] from the Karlsruhe Institut of Technology (KIT), Germany, won the Grand Cooperative Driving Challenge [99]. In this competition, both the performance of a longitudinal controller and vehicle-to-vehicle (V2V) communication was assessed in a platooning setup. The robots were required to follow the preceding vehicle at a safety distance by exchanging their positions and velocities. The winning team, led by Geiger, was later responsible for releasing the first benchmark tailored to self-driving perception [61]. Two years after, KIT perception technology, founded on their experience in DARPA Challenges, was embedded in a Mercedes Benz S-Class to perform a 103km long test in open traffic [212]. The perception sensors mounted in this vehicle are shown in Figure 2.3.

Driven by the momentum of research achievements, a set of commercial solutions have emerged in the last years. In 2014, Tesla released the first version of Autopilot[3], a Level 2 automated driving solution equipped in their manufactured vehicles. This system is an evolving software that aims to reach Level 5 automation using inexpensive sensing units (e. g., cameras and ultrasonic sensors) so that the self-driving technology is affordable for the customers. Other competitors such as Waymo[4] (former Google's car project) rely on complementary modalities such as LiDAR, radar, and vision to guarantee a robust perception performance. Waymo's robotaxis have been operating successfully in geofenced areas in the USA since 2017. More recently, many car manufacturers have partnered with other technol-

3 https://www.tesla.com/autopilot
4 https://waymo.com/waymo-driver/

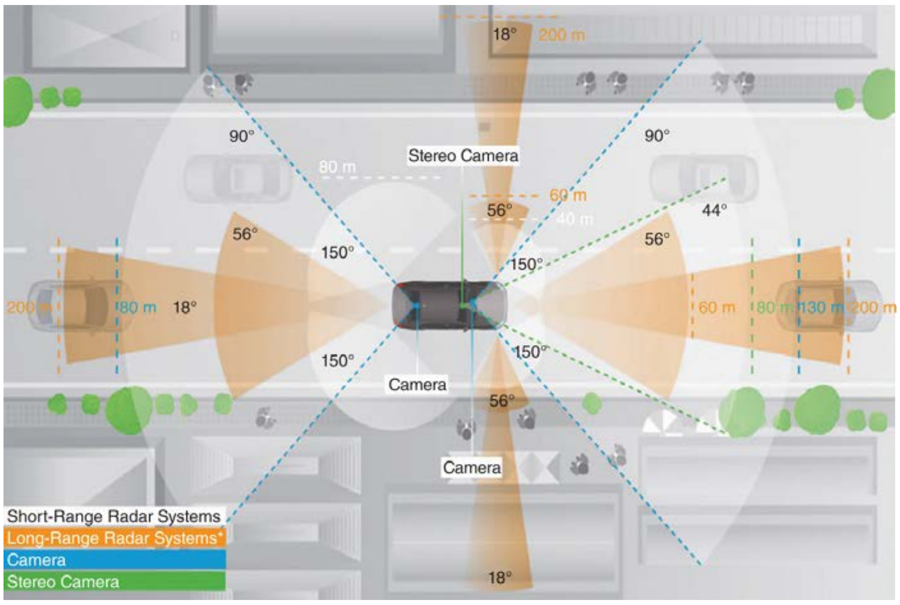


Figure 2.3: On-board sensors of the experimental vehicle used at the Bertha Benz historical route test [212] © 2014 IEEE

ogy companies to boost the development of their own autonomous driving stack. This is the case of Daimler and Bosch, which started a car-sharing pilot in San Diego, California, in 2019.

2.2 DATASETS

Advances in machine vision for autonomous cars are tightly related to the progress of Artificial Intelligence (AI) and, concretely, supervised methods. These kinds of algorithms aim to learn a model from a set of examples that is able to perform well when used in unseen inputs. As a consequence, a significant amount of labeled data is usually required for training.

The first major datasets designed for perception tasks such as object detection were focused on computer vision. Collections such as Caltech101 [49] or PASCAL[46] were composed of thousands of images and served as benchmarks for early detection algorithms. However, the number of categories and the amount of annotated samples were still short to obtain truly generic object detection models.

In this regard, ImageNet [38] database was released in 2009. The set aimed to provide an annotated large-scale collection of images organized in a semantic hierarchy mimicking WordNet structure [122]. To this end, a semi-automatic process was followed. First, image search engines of the time were used to gather all pictures on the internet for each of the semantic categories. Then, a manual filtering step was performed to discard wrong samples and balance the number of images per class. When the dataset was published, it contained more than 3M images organized in 5,247 synsets. A tiny sample of the whole dataset can be observed in Figure 2.4. Some years later, Microsoft built the COCO dataset [111] to support the research on instance segmentation of everyday objects. Both the scale and the challenges associated with these datasets turned them into fundamental pieces of the object detection publications over the subsequent decade.

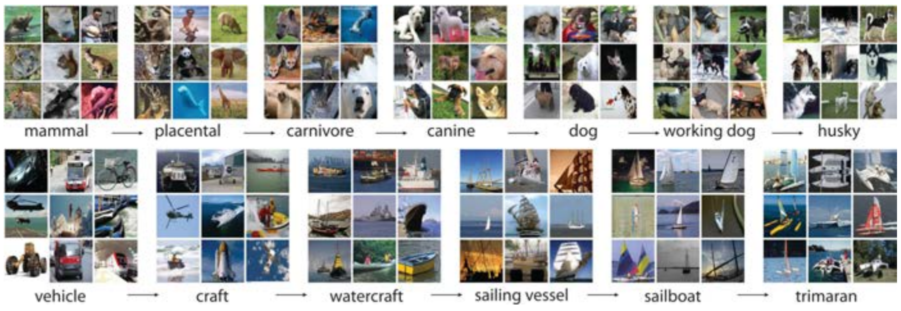


Figure 2.4: Sample images from two subtrees of the ImageNet dataset [38]. Top and bottom rows represent the mammal and vehicle branches, respectively © 2009 IEEE

Regarding the automotive field, the inflection point came in 2012 with the creation of the KITTI Vision Benchmark Suite [60, 61] as a result of the collaboration between Stiller's laboratory (KIT) and the Toyota Technological Institute at Chicago. Unlike pre-existing datasets for on-board perception [25, 89], KITTI Benchmark included annotations for common traffic objects using a calibrated multimodal sensor configuration composed by a stereo camera pointing in the forward direction and a Velodyne HLD-64 LiDAR mounted on the roof to cover the full horizontal field of view. Labels were provided not only for 2D and 3D detection but also for common robot vision tasks such as stereo matching, optical flow, visual odometry, SLAM, and 3D tracking. A GPS/IMU receiver was also equipped to obtain the ground truth for motion-related challenges. Figure 2.5 shows the recording platform and some of the provided annotations. Last but not least, an evaluation server with public rankings was released to compare the different submitted methods against common standard metrics. The availability of such an annotated dataset permitted the scientific community to work on the field without the need for expensive sensors or a robotic platform, thus pushing forward the state of the art in machine vision for self-driving applications.

After this milestone, other datasets were soon published [152]. In 2016, Cityscapes [33] was created to provide annotations for a problem not addressed by KITTI: semantic segmentation. This collection provided pixel-level classification ground truth of 25k images captured in urban scenarios recorded in 50 different cities and covering a wider set of object categories and city traffic scenes than any previous image segmentation dataset. Also following an image-only approach, BDD100K [202] was presented in 2018 by Berkeley University. Labels

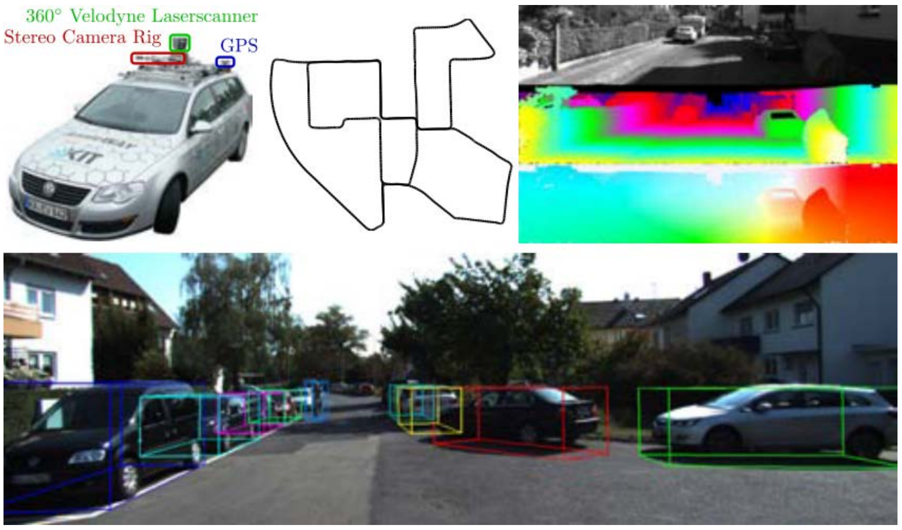


Figure 2.5: Recording vehicle and ground truth samples for trajectory, stereo, optical flow and 3D object detection of the KITTI Benchmark [61] © 2012 IEEE

for 1M frames divided into a hundred thousand scenes covering a set of 10 different computer vision challenges were provided. In this case, the main purpose was to promote the development of multi-task models.

More recently, a set of larger datasets have emerged driven by the demand for bulky amounts of annotated data to train new DNNs models of higher complexity and capabilities. In 2019, nuScenes [26] was created. To this end, a vehicle equipped with six surrounding cameras and a top LiDAR to cover 360° with both modalities was used. Several radar devices and GNSS positioning were also included in the sensor configuration (see Figure 2.6). Annotations are provided for 3D objects in the 400k frames that compose the collection. A detailed HD map containing information of 11 semantic classes was also labeled. The sequences were recorded in Boston, USA, and Singapore, in order to allow models to generalize to the particularities of dissimilar domains. Moreover, scenes cover a range of illumination and weather conditions, including night and rain. In the same year, other companies with broad experience in the field also opened to the public a subset of the training data they use to teach the perception models embedded in their commercial robo-taxis [90, 164].

Albeit these efforts have expanded the horizon of current methods, the incessant need for new labels to train supervised algorithms and the high costs of the manual annotation process have led researchers to seek less constrained sources of data: simulated environments. In this regard, numerous projects have taken advantage of modern graphic engines to build realistic simulators to generate unlimited ground truth for training and testing perception and control algorithms.

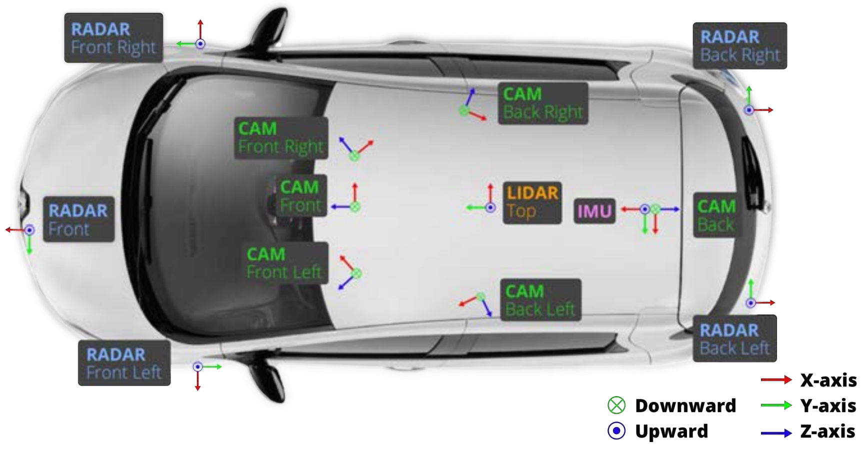


Figure 2.6: Sensor setup of the recording vehicle used in nuScenes [26] © 2019 IEEE

Richter et al. [146] explored the idea of reusing photo-realistic worlds of already developed videogames to generate automatic annotations to be used for training. To this end, the rendering pipeline of Grand Theft Auto 5 (GTA5) game was hacked. In another work [55], a synthetic clone of some sequences from the KITTI dataset was presented. Unlike virtual worlds without a real sibling, the scenarios of this work were created in a semi-automatic process, taking advantage of annotation provided by the original dataset. Thus, a manual refinement step was sufficient to obtain realistic results. The main shortcoming is that the amount of scenes is limited to those defined in the real benchmark. On the contrary, a wider variety of lighting and weather conditions can be simulated to enhance existing perception methods.

Alternatively, ad-hoc simulators have also been developed. Ros et al. published the SYNTHIA dataset [149] in 2016, containing semantic annotations for more than 60k images. Newer versions have extended the collection by including depth information and instance labels. One year later, Carla simulator was open-sourced [42]. This platform includes a set of predefined urban scenarios and sensor configurations, although they can be easily extended. Besides, realistic illumination and weather changes can be set. By giving full control of the software, this work allowed researchers to generate an unrestricted amount of synthetic training data. Shortly after, Microsoft released AirSim [154], a dedicated engine to close the gap between virtual and real data by offering high-fidelity visual appearance and realistic physics simulations. Although it was initially designed for testing algorithms in Unmanned Aerial Vehicles (UAVs), self-driving vehicles were soon included. Sample frames from the mentioned simulation solutions are shown in Figure 2.7.

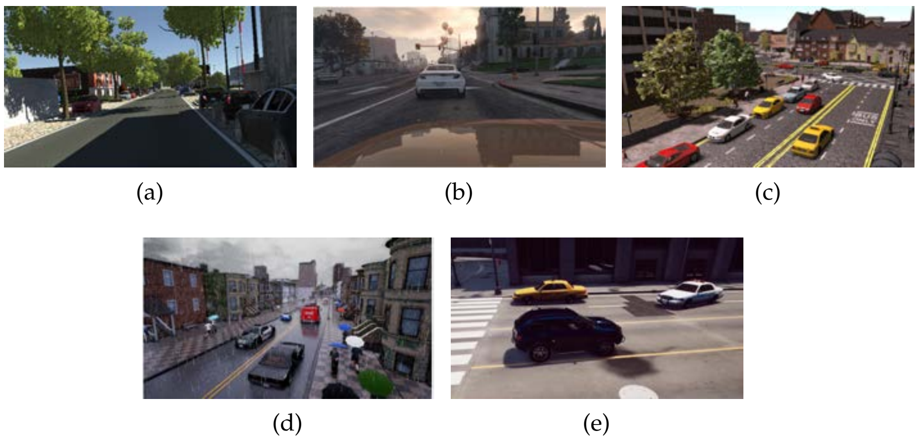


Figure 2.7: Snapshots of different simulation environments: (a) Virtual Kitti [55] © 2016 IEEE; (b) GTA5 [146] © 2016 Springer; (c) SYNTHIA [149] © 2016 IEEE (d) Carla [42] © 2017 CoRL; (e) AirSim [154] © 2018

2.3 SENSOR CALIBRATION

Multi-sensor perception systems in robotics and automotive platforms usually presume an a priori knowledge of their positioning. A good estimate of these coordinates allows to find correspondences between sensors' data and, thus, combine their information to build more robust algorithms. However, measuring the exact position between sensors is not a trivial task, as the optic center of sensor devices is not generally accessible. Thus, in order to compute the transformation matrix defining their relative pose, a calibration process is needed.

Motivated by the difficulty to manually adjust the extrinsic parameters of a pair of sensors and the common miscalibrations derived by the operation of mobile robots like cars, many research works have focused their attention on automating the calibration process. Although most efforts have tried to address the camera-LiDAR problem, the emergence of more complex configurations that make use of several vision and range devices have opened new challenges.

Sensors calibration is commonly seen as a preliminary stage to be performed before the operation of the robot starts. As a result, this process takes place in controlled environments and makes use of adhoc scenarios. Even though initial approaches required the intervention of a human to manually annotate the LiDAR-camera correspondences for the final transformation to be obtained [151, 168], the trend soon shifted into fully automatic methods.

Most extrinsic calibration algorithms use unambiguous fiducial targets specifically designed to provide key features in the different sensor data modalities. To this end, rectangular [105], triangular [37] or polygonal boards [136] are common choices, as planar targets are easily identifiable using both range or color information.

To further enhance the calibration of sensor pairs where cameras are involved, planar boards are frequently endowed with visual motives which offer higher quality features than those derived from shape detection in the image space. QR markers [39] and, specially, checkerboards [182] are often used. The use of several calibration artifacts (see Figure 2.8) in a single scene [62] or in multiple frames [208] has also been explored and typically leads to better results.

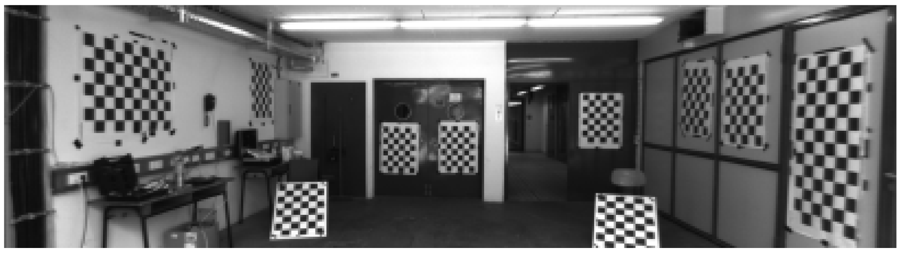


Figure 2.8: Calibration setup for Geiger et al. [62] © 2012 IEEE

Similarly, other approaches have opted for customized calibration patterns to make them more distinguishable in either domain. Velas et al. [181] proposed an approach enabling the estimation of the extrinsic parameters using a single point of view, based on the detection of circular features on a planar board. In a parallel work [211], the use of a checkerboard with holes at the tiles' center was presented.

Alternatively, some methods have successfully used 3D objects like spheres[137] or boxes [140] as calibration targets.

A second set of works aim to perform the calibration process without the use of any calibration artifacts. Unlike the former group, they take advantage of the elements available in the scene to establish the required correspondences to compute the relative pose.

Similar to target-based methods, most approaches rely on extracting features within the overlapping field of view of the sensors. The registration of linear characteristics computed in both modalities have been used, both in outdoor [163] and indoor scenarios [124], achieving acceptable results (see Figure 2.9). Other works, more tailored to automotive applications, have benefited from the objects commonly found in traffic environments to perform extrinsic parameters estimation, such as the ground plane ahead the vehicle [148].

Pandey et al. [131, 132] present the calibration problem as the maximization of mutual information between LiDAR and camera intensity values. Likewise, Castorena et al. [27] focused on optimizing the edge alignment between an interpolated dense depth map and image intensity features using simulated annealing.

In another fashion, some approaches dispose of the feature extraction stage and try to solve the extrinsic calibration issue as the correspondence between the trajectories of the sensors in a moving plat-

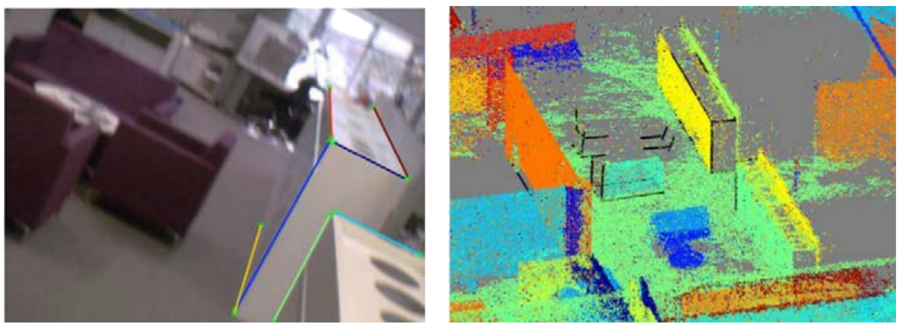


Figure 2.9: Line extraction on the image and LiDAR modalities in a typical calibration scene for [124] © 2013 IEEE

form. Thus, the robot requires to compute the odometry for each of the sensors separately. The relative pose is then estimated by solving the transformation between sensors that allow aligning both trajectories. Although less precise due to the accumulated error from the computation of the ego-motion stage, these kinds of algorithms have proved suitable for sensor pairs with overlapping [135] and non-overlapping field of views [2, 169]

In the last years, the potential shown by Convolutional Neural Networks (CNNs) in perception tasks has also been applied to address the calibration issue. RegNet [153] made use of a deep convolutional neural network to perform a targetless end-to-end calibration process, where the extrinsic parameters are inferred from a single frame and continuously adjusted online during the operation. Although precise, this method suffers from generalization problems as it has to be trained on a previously calibrated configuration. On the other hand, CalibNet [85] followed a self-supervised approach that iteratively re-aligns the LiDAR cloud so that photometric and geometric errors are minimized.

Pushed by the increasing use of multi-LiDAR setups in the perception stack of automated vehicles, some research works have proposed specific methods for range-to-range calibration. First approaches made use of a third sensor to compute the relative pose of the desired pair of laser scanners [56] [74]. Moreover, a coarse seed pose was needed to guide the process and ensure a proper final transformation. More recently, Jiao et al. [86] presented a more convenient method that gets rid of the additional device and is solely based on three linearly independent planar surfaces.

Although the literature on sensor calibration methods is extensive, the evaluation of their results remains an open issue. The absence of real ground truth, which requires a precise relative pose as a reference that cannot be obtained in practice, has led to the use of custom schemes, which are difficult to extend to other domains and eventually based on inaccurate manual annotations. In this regard, Levinson and Thrun [102] presented a method to detect miscalibrations through the variations in an objective function computed from the discontinuities in the scene. A more recent work tried to address the assessment problem by measuring the mean line reprojection error of ad-hoc planar targets edges [123].

2.4 OBJECT DETECTION IN IMAGES

Although visual reasoning is an effortless task for humans, computers struggle to obtain knowledge from images. As a result, computer vision techniques need to be applied to exploit the pixels' information and infer valuable outputs, such as the detection and classification of objects.

2.4.1 *Historical background*

Classical methods for object detection usually consist of three separate steps: feature computation, candidate generation, and classification. First, the image is processed to extract features considering local information of pixels. Second, a set of Regions of Interest (ROIs) of several shapes and locations are proposed by means of sliding windows or more advanced approaches like Selective Search [175]. Last, a classification method is applied over these candidate regions. This latter stage is usually based on the use of machine learning algorithms that take hand-crafted features as inputs. As these features need to be designed by humans, they are limited in their representation capabilities.

Several local filters based on differences in shape or pixel intensities have been developed for object detection purposes. Haar-like features [184], Histogram of Oriented Gradients (HOG) [36] or scale-invariante feature transforms (SIFT) [115] are a common choice for the detection of cars [77, 134, 162] and pedestrians [155, 183, 210]. For the classification part, distinct discriminative models are used, such as Support Vector Machines (SVMs) [34] or AdaBoost [52]. Although the results provided by any classification architecture differ for the same input set, Benenson et al. [15] discuss that the accuracy of classical models is mainly limited by the number and variety of the input features.

In this regard, Neural Networks (NNs) stand as an alternative approach capable of addressing the drawbacks of hand-crafted features, as they are able to jointly learn features from the input and perform the classification task in an end-to-end fashion. This way, the training procedure is not only dedicated to giving a higher weight to the features that enhance discrimination between object categories, but also to learning the best representation that maximizes the desired objective function. Although researchers encountered difficulties to apply NNs for computer vision problems in the 1980s [101], novel

DNNs approaches to image processing showed promising results at the beginning of the twenty-first century [75].

2.4.2 *Convolutional neural networks*

Despite regular neural networks can be applied to computer vision problems, the all-to-all mapping of neurons between layers makes them impractical for data structures containing thousands or millions of inputs, like image pixels. To tackle this issue, convolutional layers were introduced in 1980 by Fukushima et al. [53] based on the findings of [81], which demonstrated that the human visual cortex was made of individual neurons that get excited by small regions of the visual space, called receptive fields.

A convolution is a filter function that takes the pixel values in the vicinity of a given position in the image and computes a weighted sum of them, followed by a bias offset. The receptive field or kernel size of the convolution function is usually small, but it extends through the depth of the input volume. In the forward pass, an activation map is generated by computing the dot product of the kernel matrix at all coordinates of the input. As a network layer, both the weights of the convolution and the bias are the learnable parameters.

When compared to regular perceptrons, convolutional layers have a set of advantages for image processing. On the one hand, CNNs exploit local connectivity of adjacent neurons, following the principles of visual receptive fields. As layers get stacked, the model is capable of learning more global features, as overlapping kernels incorporate information from connected units. On the other hand, the use of the same operation across the whole input volume enables uniform responses for a given input feature vector regardless of its position, thus providing translation invariance. Moreover, the number of model parameters remains unchanged for any input size.

In addition to convolutional layers, CNNs are usually composed of other types of operations required to build complex objective functions and generate the desired output volume.

To perform image classification, Fukushima et al. [53] proposed a downsampling layer that averages the activations from units inside the kernel. Applying this layer to non-contiguous positions generates a map of reduced size. This operation allows to decrease the memory footprint and lower the computational requirements of the model, as subsequent convolutions are fed with smaller input volumes. Although the method worked well for Japanese character recognition, the averaging operation hampered the identification of sharp cues. To address this drawback, subsequent works switched to the use of max-pooling, as this layer outputs the maximum value of a sub-region, preserving salient features.

To be able to approximate the model to arbitrary complex objective functions, non-linearities are required. To this end, activation layers are applied to the output of linear operations like convolutions. Rectified Linear Units (ReLUs), for instance, performs an element-wise clipping-to-zero to remove the negative values from activation maps. Despite being the most popular activation layer in CNNs, other non-linear functions like *sigmoid* or *tanh* can be applied. A special mention is needed with the *Softmax* function, which is usually used as the last layer in multinomial logistic regression problems, as it normalizes the output logits to a probability distribution of the predicted classes, so they all add up to 1.

Regular artificial neurons are also part of some convolutional neural models. Once the feature map has been significantly reduced by a sequence of convolutional and downsampling operations, Fully-Connecteds (FCs) layers can be used to leverage all activation units from the input volume regardless their spatial positioning. When used, FCs layers are usually appended at the end of the pipeline to predict the final classification scores, just before the *Softmax* normalization.

Even though CNNs have been successfully applied to a variety of computer vision problems, the scientific community become aware of their true potential with the emergence of DNNs, fostered by the extended usage of GPUs for training [129, 159]. The acceleration of the training process allowed to feed networks with unprecedented volumes of data, so the depth of the models started to increase to be able to learn discriminative features to classify among a growing number of categories. Benchmarks like ImageNet [38] favored this trend and led to the creation of popular CNNs architectures nowadays:

ALEXNET: Krizhevsky et al. [94] won the ImageNet ILSVRC challenge in 2012 with a slightly deeper architecture than LeNet. Contrary to its predecessor, AlexNet uses more than one convolution between ReLU activation layers.

ZFNET: the model created by Zeiler and Fergus [203] introduces minor changes to AlexNet, most of them related to parameter tuning. Besides, the filter and stride size of the first layers are reduced to enhance the quality of the corresponding feature maps. This network won the 2013 ImageNet Challenge.

GOOGLENET: the main contribution of the ILSVRC 2014 winner [166] was the *Inception module*, which allowed to significantly reduce the number of parameters of the model (less than 10% of the number in AlexNet). With the use of 1x1 convolutions, the width and depth of the network could be increased while decreasing the computation bottleneck. It is one of the early adopters of Batch Normalization [84].

VGGNET: this name defines a set of CNNs[161] sharing a common structure but with different depths. It follows a similar approach to AlexNet, although kernels are reduced to 3x3 convolutions with a higher number of filters. The most popular model is the VGG16, although there exist other variants with a distinct number of stacked layers.

RESNET: He et al. presented the Residual Neural Networks [73] at ILSVRC 2015. This architecture introduced the *skip connections*, which allows building deeper models while preventing the associated degradation problem. As the network depth increases, the accuracy saturates before it starts going down as more layers get stacked. The *residual block*, shown in Figure 2.10, establishes a relationship between the layer input and the output through the identity function, forcing the convolutional operation to learn a simple residual. This way, if an excessive number of layers were added, it would be easier for the model to learn to produce a zero residual than the identity function.

XCEPTION: this model [32] is an adaptation of the inception model. However, Xception relies on the use of depth-wise separable convolutions, which is equivalent to set all *Inception* cross-channel convolutions to 1x1, and its spatial correlations to 3x3 kernel.

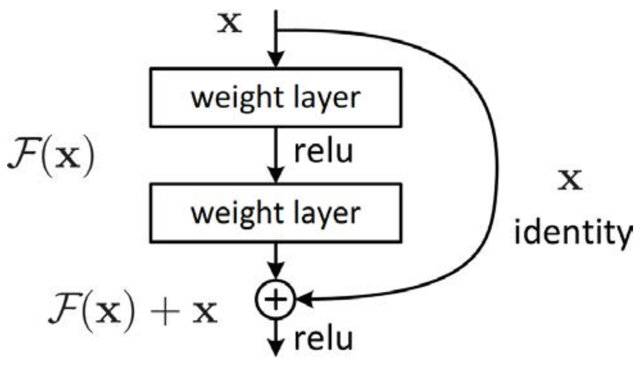

x
weight layer
$\mathcal{F}(\mathbf{x})$
relu
weight layer
x
identity
$\mathcal{F}(\mathbf{x}) + \mathbf{x}$
relu

Figure 2.10: ResNet block diagram [73] © 2016 IEEE

2.4.3 *CNNs for object detection*

With the advent of DNNs, representation learning models soon outperformed classical approaches on vision classification problems. However, the localization of objects inside images still exceeded the capabilities of these works. In this regard, Girshick et al., inspired by classical pipelines, presented the Region-based Convolutional Neural Network (R-CNN) [64, 65], a CNN which uses ROIs of an image to perform classification (see Figure 2.11). The generation of image patches used as input was made through Selective Search [175], and all candidate regions were warped to a fixed-size vector to compute CNN features. After a set of convolutions, the resulting features are fed to dedicated per-category binary SVM classifiers. Due to the high amount of candidate regions, the algorithm was very computationally expensive.

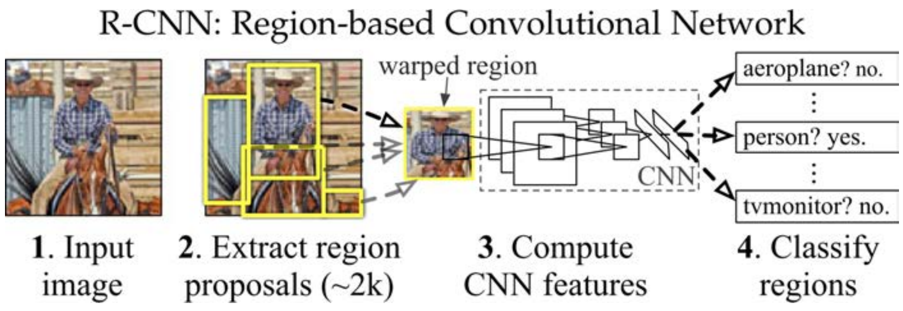
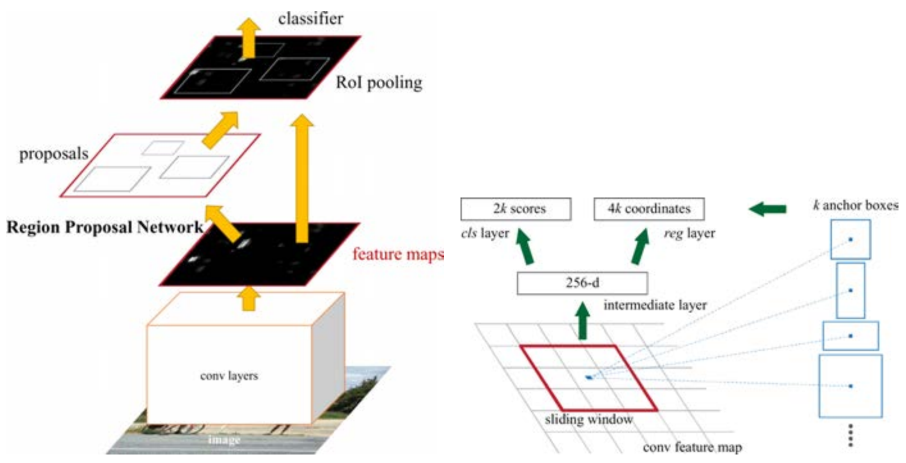


Figure 2.11: R-CNN detection architecture [64] © 2014 IEEE

To tackle this shortcoming, a major evolution of the method was released soon after [63]. Fast R-CNN introduces ROI pooling, a key concept that allows to feed the whole image at once through the convolutional layer of the model and extract the candidate regions directly over the features map without stopping the gradient computation. This improvement simplified the training process and dramatically reduced the computation time of the algorithm.

Two years later, Ren et al. [145] introduced the final iteration of the architecture: Faster R-CNN. This work presented the concept of Region Proposal Networks (RPNs), a neural network branch aimed to perform a positive-negative classification of a set of predefined boxes, as well as a box refinement. The patches are classified with an objectness score, which is used to filter out ROIs belonging to the image background. Positive samples are then applied to extract candidate regions by means of the ROI pooling operation. Using the proposed Region Proposal Network (RPN), the external candidate generator is no longer needed. Selected ROIs are then fed to the model heads as in previous versions of the framework. For Faster R-CNN to be trained in an end-to-end fashion, a multi-task loss able to optimize both the Region Proposal Network (RPN) and the classification head is used. Figure 2.12 illustrates the different components of the approach.

In order to mitigate the performance bottleneck of Region Proposal Networks (RPNs), single-stage detectors have attracted considerable attention. Redmon et al. [143] propose to learn both the class probabilities and the bounding box parameters using the feature map resulting from the latest layer in a CNN based on GoogleLeNet [165]. Dispensing with the region proposal branch reduces the computational time, making it more suitable for real-time applications. An overview of the framework is shown in Figure 2.13. On the other hand, Single Shot Multibox Detector (SSD) [112] commits to a fully convolutional pipeline where the class and box inference also leverages intermediate feature maps. In order to cope with the high foreground-background classification imbalance while training, a hard negative mining strategy is followed. The detection at multiple scales and the use of default boxes, similar to anchors in [145], outperforms



(a) Overview of Faster R-CNN (b) RPN

Figure 2.12: Faster R-CNN detection architecture [145] © 2017 IEEE

YOLO's accuracy. However, the joint classification of positive objects and background examples still prevent comparable performance to two-stage detectors. To overcome this limitation, Lin et al. [110] introduced the *Focal Loss*, which forces to focus the learning on hard examples and reduces the contribution of the vast number of negative examples. When published, RetinaNet [110] outperformed any existing two-stage framework while maintaining the speed of other one-stage detectors. Although several new approaches have been presented ever since [47, 100, 144], the performance of modern two-stage detector prevails.

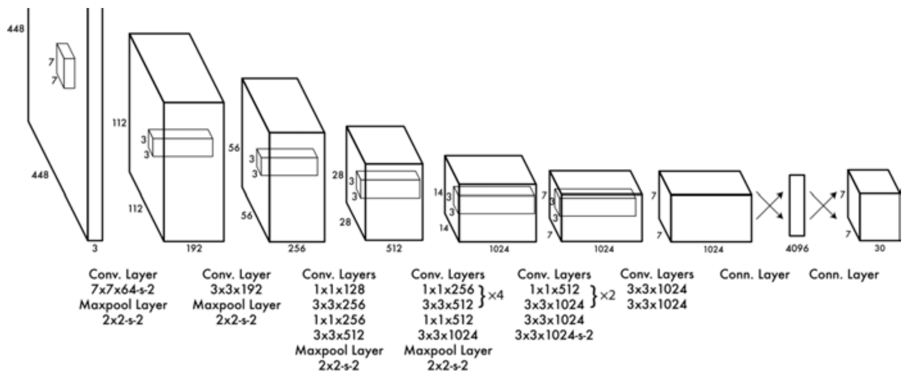

448
448
3
192
256
512
1024
1024
1024
4096
30
Conv. Layer 7x7x64-s-2 Maxpool Layer 2x2-s-2
Conv. Layer 3x3x192 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x128 3x3x256 1x1x256 3x3x512 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x256 3x3x512 ×4 1x1x512 3x3x1024 Maxpool Layer 2x2-s-2
Conv. Layers 1x1x512 3x3x1024 ×2 3x3x1024 3x3x1024-s-2
Conv. Layers 3x3x1024 3x3x1024
Conn. Layer
Conn. Layer

Figure 2.13: YOLO detection architecture [143] © 2016 IEEE

2.5 3D OBJECT DETECTION

While detection of objects in the image space is sufficient for many computer vision applications, the perception systems for autonomous driving aim to infer a spatial understanding to enable safe control and navigation tasks. Among the different sensor configurations typically found in automated vehicles, those based on cameras and LiDAR devices have gathered the attention of most research studies due to the low-cost and precise geometry information, respectively. As a result, the single or combined use of these devices has led to a myriad of frameworks for the 3D object detection problem.

2.5.1 *Image-based detection*

A first group of approaches makes use of appearance information provided by camera sensors to perform 3D object detection. Like those described in Section 2.4, these methods benefit from both the structureness and the rich, dense features of image data. However, the lack of explicit spatial information becomes a major challenge when the objective is the inference of accurate 3D bounding boxes for the objects in the scene.

To address this shortcoming, different lines of work are being explored. On the one hand, the detection of keypoints defining the object geometry to later find a 3D correspondence with annotated CAD models has proved to be effective in end-to-end models [6, 28]. Figure 2.14 illustrates an overview of the method in [6].

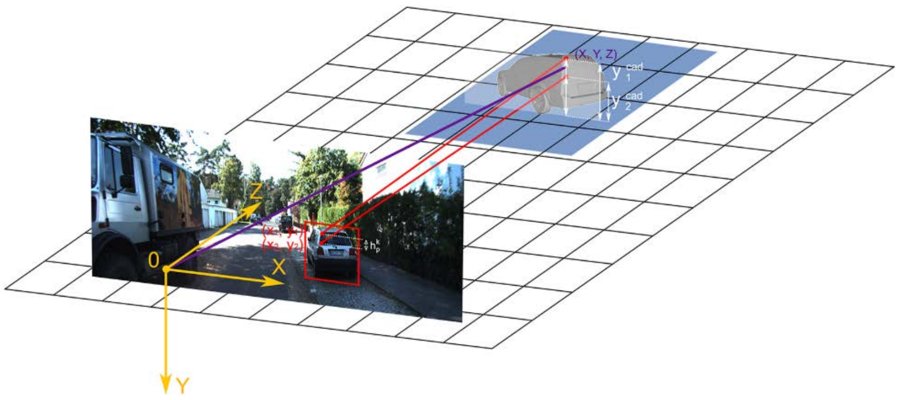


Figure 2.14: Association between detected keypoints in the image space and a CAD model used for 3D box estimation in [6] (license CC BY-NC-ND 4.0))

Similarly, other approaches advocate for simplifying the task and rely solely on the estimation 2D keypoints of the target 3D bounding box before the final regression. Liu et al. [113] uses the image projection of the center of 3D anchors, while Li et al. [104] also includes the box vertices in the keypoint detection stage.

On the other hand, some studies focus on exploiting the implicit consistency between the 2D and 3D boxes in the image view. In this regard, Gahler et al. [54] proposes to split the common 2D detection problem into two subtasks: front and side detection. These two ROIs compose a *MergeBox* with a pair of visible surfaces of every object, which are then used to infer the final regression from compatible 3D anchors. Likewise, [103] performs a joint estimation of the 2D box and the object heading to populate an oriented 3D box which is further refined by the network. Naiden et al. [127] follows a comparable framework, but the spatial box is computed from a least-square minimization instead of from orientation cues. Other methods add complementary losses to improve the detection of overlapping objects [31].

Finally, other works prefer to tackle the absence of range data before performing the 3D detections. In [23], depth-aware convolutions are used in combination with regular kernels. However, the explicit estimation of the disparity map, which contains pixel-wise depth information, is usually the preferred solution [41, 142, 193]. Among this latter fashion, some methods take advantage of the estimated depth to build a pseudo lidar cloud and apply point-based inference models [117, 189], as shown in Figure 2.15. Ku et al. [96] suggest the delimitation of the depth estimation problem to object candidates so the model can optimize the output to regions of interest. Also based on depth computation, a different set of approaches stack the range information as an additional channel in the image and perform the 3D regression with variations of popular 2D detectors [41, 116].

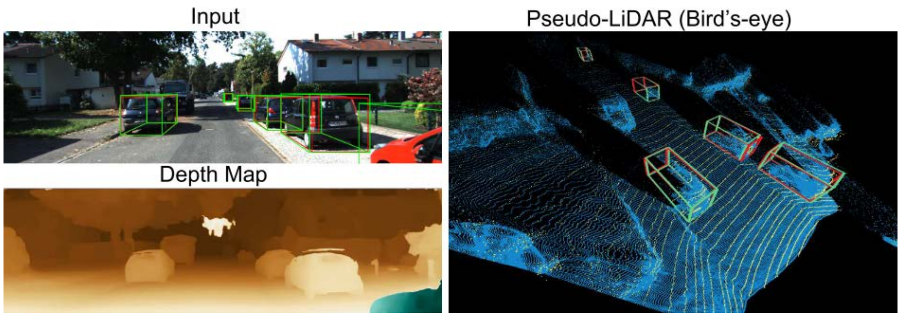


Figure 2.15: Depth estimation and pseudo-lidar point cloud infered from an RGB image in [189] © 2019 IEEE

A quantitative snapshot of the current state-of-the-art methods for monocular 3D detection can be found in Table 2.1. Among the different approaches to 3D detection from images, the most representative methods have been selected for comparison.

2.5.2 *LiDAR-based detection*

The task of object detection using LiDAR sensors exploits the spatial representation provided by modern laser scanners [150]. These devices are able to faithfully capture both the 3D shape and reflectiveness of the objects in the environment, usually spanning the whole horizontal field of view. Although this information is sufficient to identify and classify the different road participants, the characteristics of the collected data pose a challenge when deploying such al-

Table 2.1: BEV and 3D object detection performance of selected monocular methods on the testing set of the KITTI Benchmark. Results are given for the *Car* category and *Moderate* difficulty

Method	Key Concept	AP BEV (%)	AP 3D (%)
RTM3D [104]	Keypoints	14.20	10.34
MonoPair [31]	KP + Pair distance	14.83	9.99
AM3D [117]	Image depth	17.32	10.74
MonoPSR [96]	Object depth	12.58	7.25
M3D-RPN [23]	Depth as channel	13.67	9.71

gorithms for real-time applications. Thus, its lack of structure and sparsity has led to the creation of different lines of research based on distinct representations of LiDAR information.

The first group of works uses the LiDAR input as-is in order to obtain better features based on the fine-grained geometry and intensity values provided by the captured point clouds. Although some approaches utilize the information from the raw cloud through all the layers to produce the final 3D detection boxes [157], most approaches rely on a previous downsizing of the cloud to reduce the computational load. For instance, Point-RCNN [156] reduces the size of the LiDAR cloud by performing a background-foreground classification of the points before feeding the positive set into the final 3D box estimation networks. Other proposals, such as STD [199], lay between raw and voxel-based pipelines using a point-wise feature extraction over the whole LiDAR set followed by a discretization step that significantly reduces the inference time.

The second subset of networks takes a voxelized version of the LiDAR data as input. This kind of representation reduces the information volume and guarantees a regular structure that enables its processing using 3D convolutions. Space is divided into a volumetric grid where each 3D cell, also known as voxel, stores features computed from the points lying inside. Many approaches combine voxel feature encoders with region proposal networks, either using a single [196, 209] or multiple voxel scales [97] as inputs. Similarly, Part-A2 [157] follows a two-stage approach, where, firstly, both intra-object parts and point-wise semantic segmentation are estimated and fed to a part-aggregation step that produces the final detections. Other works predict the final 3D boxes following single-stage [114] or anchor-free [188] schemes.

Finally, a third approach aims to further compact the input by making use of bidimensional representations of the LiDAR cloud. To this aim, the laser data is projected into pseudo-images, decreasing the processing time and enabling the use of 2D object detection networks.

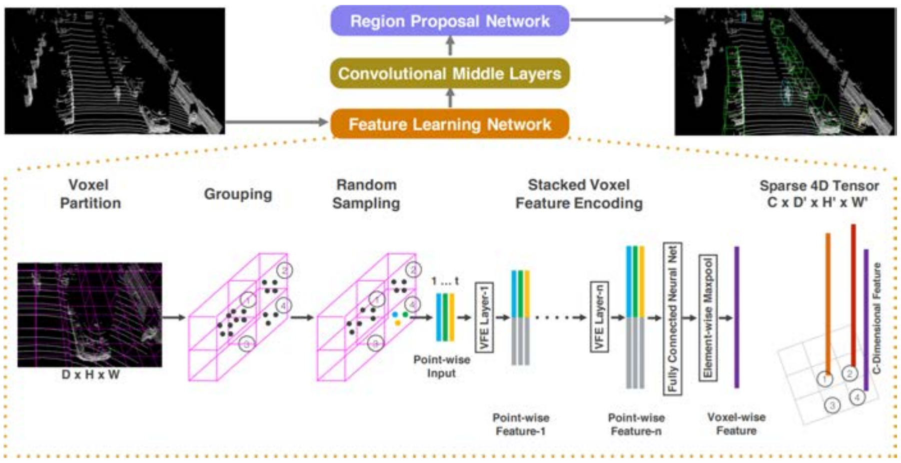


Figure 2.16: Overview of the VoxelNet architecture [209] © 2018 IEEE

Among this group of frameworks, those taking the BEV projection as input are the most popular. Some works use hand-crafted BEV images to feed single-stage [160, 197], or two-stage object detectors [192]. Each cell in the 2D input typically includes information on the intensity, height, and distribution of the points lying inside. In MODet [206], the representation of the BEV is simplified to a binary occupancy grid. Alternatively, PointPillars [98] introduced a learned BEV encoding produced by a PointNet network able to compute features directly from the original 3D points contained in the cell.

Although less frequently, other papers make use of the Range View (RV) projection of the LiDAR to perform end-to-end object detection. LaserNet [120] is able to predict per-point class and bounding box distributions that are then clustered to obtain the final BEV detections. Recently, RangeRCNN [108] presented a novel approach able to learn cues in the range projection of the LiDAR cloud and transfer them to the BEV representation to produce the final 3D box estimation.

Table 2.2 includes the results of some selected frameworks to allow the reader to understand the current state of the art in the field of LiDAR-based object detection.

Table 2.2: BEV and 3D object detection performance of selected LiDAR-based methods on the testing set of the KITTI Benchmark. Results are given for the *Car* category and *Moderate* difficulty

Method	Key Concept	AP BEV (%)	AP 3D (%)
Voxel-FPN [97]	3D Grid	87.21	76.70
CenterNet [188]	Anchor-Free	88.29	77.62
PointPillars [98]	Learnt BEV	86.56	74.31
RangeRCNN [108]	Range image	88.40	81.33

2.5.3 *Fusion-based detection*

In order to cope with the limitations of the camera and LiDAR modalities, a third research line aims for the joint use of data from both sensors in a fusion perception framework. The main motivation is the possibility to combine complementary information sources to enhance the learned representation of the objects and increase the robustness of the model among different detection ranges or weather and illumination conditions. Nevertheless, how these modalities are fused effectively is yet a matter of research.

Some approaches have opted for a sequential pipeline, where the output of image processing facilitates object detection in the point cloud. In this regard, 2D ROIs have been used to reduce the search space before applying 3D detectors [141, 158, 190, 204]. On the other hand, semantic segmentation in the image space is also a common choice. Distinct methods make use of pixel-wise classification via 3D-2D projection, either to filter background points [198] or as an additional component for every point in the LiDAR cloud [29, 185]. Likewise, FusionPainting [195] addresses the joint use of 2D and 3D semantic cues. Alternatively, Yin et al. [200] proposes the generation of 3D virtual points to reduce the sparsity of objects clouds by using the information from 2D instance segmentation masks.

On the contrary, another set of works performs LiDAR-camera fusion at the feature level. To this end, both the image and the laser sweep are fed into a network, where encoded characteristics are combined at certain layers so that the model is able to learn to best exploit the statistics coming from both sensor modalities. Although initial approaches took advantage of several projection views of the LiDAR cloud as input [30] (see Figure 2.17), most approaches solely rely of the BEV and camera information by explicitly performing manual 3D-2D feature mapping [80, 95, 106, 107, 187] or in a learnt manner [201].

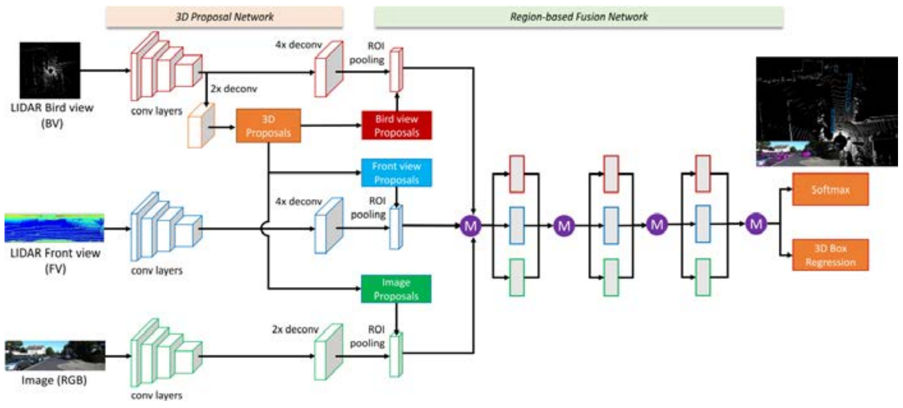


Figure 2.17: MV3D fusion network diagram [30] © 2017 IEEE

Finally, late fusion schemes have also been explored in the literature. For this purpose, separate detections in the image and LiDAR space are combined either as a validation step for low-resolution lasers canners [57], or to enhance the results of a 3D LiDAR detector by finetuning the classification scores [133].

Table 2.3: BEV and 3D object detection performance of selected fusion-based methods on the testing set of the KITTI Benchmark. Results are given for the *Car* category and *Moderate* difficulty

Method	Key Concept	AP BEV (%)	AP 3D (%)
F-PointNet[141]	2D Det. + Frustum	84.67	69.79
MV3D [30]	RV-BEV-Image	78.93	63.63
PointPainting [185]	2D Seg. + 3D Det.	88.11	71.70
MMF [106]	Dense Fusion	88.21	77.43
3D-CVF [201]	Learnt Feat. Map	89.56	80.05
CLOCS [133]	Late fusion	89.48	82.28

2.6 CONCLUSION

This chapter has presented a survey on some of the research lines related to the scope of this thesis. A historical revision of autonomous driving technologies has been drawn, with special attention to the evolution of their sensor setups and machine vision techniques. Additionally, some of the most recent works in the field of multimodal perception have been introduced, identifying the current trends and the open questions.

Based on the state-of-the-art review, the focus of this thesis is directed towards generating new solutions to help solve some of the challenges faced by these technologies. Namely, a set of approaches are proposed in the areas of sensor calibration, 3D object detection, and deployment in real platforms.

Part II

PROPOSED METHODS

3 SENSORS CALIBRATION

Autonomous driving relies on accurate information about the environment to make proper decisions concerning the trajectory of the vehicle. High-level inference modules receive these data from the perception systems, which must be therefore endowed with an exceptional degree of robustness under different circumstances such as illumination and weather.

Consequently, the design of perception systems intended for onboard automotive applications is currently geared towards topologies with several complementary sensory modalities. Vision systems are frequent in close-to-market vehicle setups [51] due to their ease of integration and their ability to provide appearance information. Stereo-vision systems, which use a pair of cameras separated by a fixed distance to get depth information about the environment, stand out as a cost-effective solution able to provide additional dense 3D information to model the surroundings of the vehicle.

On the other hand, the remarkable development of 3D laser scanning technology has enabled its widespread use in both research and industrial driving applications in recent years. Unlike vision systems, LiDAR range measurements are accurate and, frequently, provide information in a full 360° field of view. Configurations made of more than one LiDAR device are becoming more and more popular since they allow gathering high-resolution data using compact setups.

Due to the particular features of these sensory technologies, they are suitable to be part of the same perception system, providing complementary information. To best exploit multi-sensor topologies, data from the different devices must be appropriately combined. In the most usual setup, sensors have overlapping fields of view (as in Figure 3.1), and the advantages conferred by their joint use come from the ability to make correspondences between both data representations. This is the case, for example, with popular multi-modal 3D detectors [30, 141]. These methods assume that the relative pose between the sensors, given by their extrinsic parameters, has been obtained beforehand and is available during operation. Due to the field of application, an extraordinary accuracy in the calibration is required so that it is still valid for data association at long distances.

Nevertheless, current calibration methods still do not provide a comprehensive response to the need to estimate the extrinsic parameters of certain sensor setups, such as the ones found in autonomous

This chapter includes content from [69] and [14]

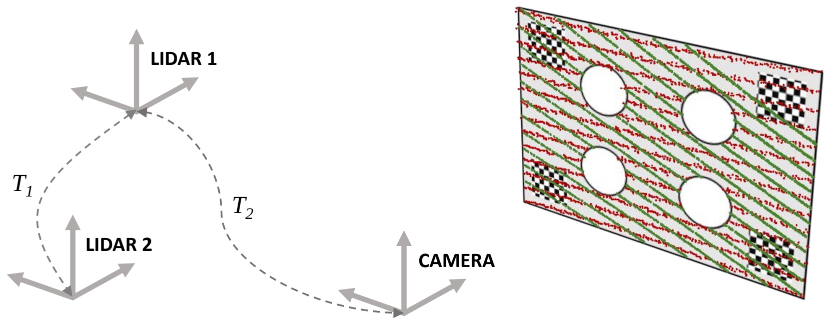


Figure 3.1: Sample calibration scenario for an arbitrary setup with a camera and two LiDAR scanners, where the calibration target is placed in the overlapping field of view of the involved sensors.

driving. They are either excessively focused on specific configurations, lacking generalization ability, require burdensome ad-hoc environments, or have not been sufficiently validated due to the unavailability of objective assessment methods.

In this chapter, we present an original self-calibration method tailored to automotive sensor setups composed of vision devices and multi-layer LiDAR scanners. The approach makes use of a novel fiducial calibration target that allows the unambiguous extraction of robust reference points in each of the supported modalities. At a second stage, the optimal transform relating a pair of sensors is obtained through the registration of the detected 3D key points.

Along with the calibration method, we also introduce a novel framework for the assessment of extrinsic calibration algorithms based on a simulation environment. This approach provides a perfect ground truth of the transform between sensors in space and establishes a fair benchmark for comparing calibration methods through metrics that truly represent the accuracy of the final estimation. Besides, it allows testing virtually unlimited sensor devices and relative poses to guarantee the generality of the results.

An extensive set of experiments using the proposed evaluation benchmark shows that the accuracy of the calibration estimate exceeds other approaches in the literature. Tests on real sensors corroborate the results obtained in the simulation environment, confirming the adequacy of the method for self-driving applications.

The implementation of the method has been made publicly available to promote reproducibility and provide researchers with a convenient tool to face the usual problem of extrinsic calibration in an easy and effective way. The software makes use of open source libraries and is published as a package in the popular ROS framework[1]. The synthetic test suite used for the experimentation has also been released[2].

1 http://wiki.ros.org/velo2cam_calibration
2 https://github.com/beltransen/velo2cam_gazebo

3.1 AUTOMATIC EXTRINSIC CALIBRATION

The proposed calibration solution estimates the rigid-body transformation that defines the relative pose between a pair of sensors. Each of these sensors can be a monocular camera, a stereo camera, or a multi-layer LiDAR scanner, in any possible combination. Rangefinders with a lower resolution (e. g., 16-layer devices) are also supported.

The transformation between the pair of sensors can be defined by a vector of six parameters $\theta = (t_x, t_y, t_z, r_x, r_y, r_z)$, which describe the position and rotation of one of the devices in the reference frame attached to the other one. Rotations around the axes (r_x, r_y, r_z) are usually referred to as roll, pitch, and yaw angles.

Parameters in θ unambiguously define a matrix $\mathbf{T}$ that can be used to transform a 3D point between the two coordinate systems. For instance, in a LiDAR-monocular setup, a point $\mathbf{p}_M$ in monocular coordinates, $\{M\}$, can be transformed into LiDAR space, $\{L\}$, by means of $\mathbf{p}_L = \mathbf{T}_{LM}\mathbf{p}_M$ once the transformation matrix $\mathbf{T}_{LM}$ is built. Note that, in that particular case, the parameters θ_{LM}, used to obtain $\mathbf{T}_{LM}$, express the pose of $\{M\}$ with respect to $\{L\}$.

With the proposed approach, the transformation is obtained automatically from data retrieved by the sensors to be calibrated. A custom-made planar target is used to provide features that are detected and paired between both data representations. As noticeable in the two different embodiments shown in Figure 3.2, this calibration pattern is endowed with geometrical and visual characteristics that enable the estimation of keypoints in LiDAR, stereo, and monocular modalities. On the one hand, four circular holes are used to take advantage of geometrical discontinuities in LiDAR and stereo point clouds. On the other hand, four ArUco markers [58] are placed near the corners so that 3D information can be inferred from monocular images.

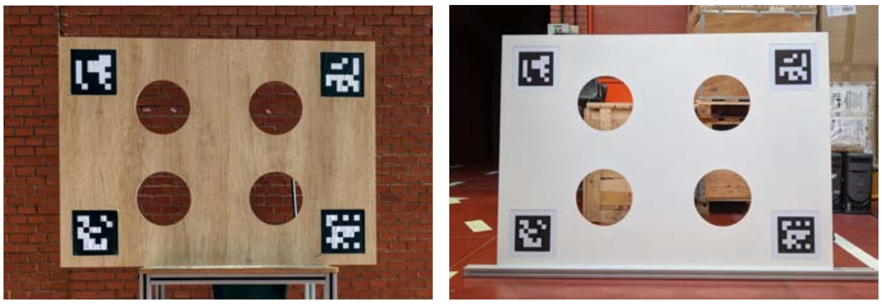

(a) (b)

Figure 3.2: Two different embodiments of the custom calibration pattern made with a CNC machine

The method does not impose severe limits on the relative pose between the devices and is therefore suitable for sensor setups where the magnitudes of the translation and rotation parameters are substantial. Only two reasonable constraints are required. First of all, there has to be an overlapping area between the sensors' field of view, where the calibration target is to be placed. Secondly, the holes in the pattern must be well visible in the data retrieved by the sensors; in particular, whenever range data is involved in the calibration, each circle must be represented by at least three points. In the case of multi-layer LiDAR sensors, this means that at least two scan planes intersect with each of the circles. Moreover, the parameters intrinsic to each device are assumed known.

The procedure is designed to be performed in a static environment. Although the method can provide a quick estimate of the extrinsic parameters with just one pose of the target, it is possible to increase the accuracy and robustness of the results by accumulating several positions, as will be shown later.

The proposed calibration algorithm, illustrated in Figure 3.3, is divided into two different stages: the first one involves the segmentation of the calibration target and the localization of the reference points in each of the sensors' coordinate systems; afterward, the second one performs the computation of the transformation parameters that enable the registration of the reference points.

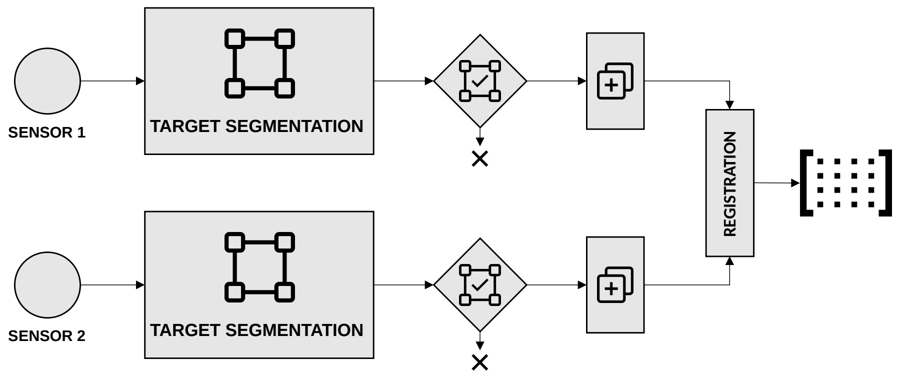


Figure 3.3: Overview of the different stages of the presented method: target segmentation, geometric consistency check, point aggregation, and sensor registration

3.1.1 *Target segmentation*

This first stage aims to localize the calibration target in each sensor's data. Consequently, the measurements at this step are relative to the local coordinate system of the corresponding sensor. As the features used to localize the pattern are different for each modality, three different variants of the procedure are proposed here, one per sensor type. In all cases, the output of this stage is a set of four 3D points representing the center of the holes in the target, in local coordinates. These points will be later used to find correspondences between the different data sources.

Although the processing of LiDAR and stereo data has some differences, especially at the beginning of the segmentation stage, both share a common trunk once the useful range data is represented in a 3D point cloud structure. The monocular alternative is substantially different as it relies on the ArUco markers instead.

The procedure described in this section is intended to be applied to every data frame provided by the corresponding sensor. Data from all sensors are processed in parallel, so they do not have to share a common trigger nor have identical refresh rates, as long as the scene is static.

3.1.1.1 *LiDAR data preprocessing*

Data from a LiDAR scanner is assumed to be represented as a 3D point cloud, $\mathcal{P}_0^L$, with measurements distributed into different layers, as typical in mechanical devices based on rotating mirrors. Each point in the cloud is defined as $p_i = (x, y, z, i)$, being the three first values its spatial coordinates, and i the reflectivity or *intensity*. In the proposed method, the point's reflectivity is dismissed. A sample laser point cloud can be observed in Figure 3.4.

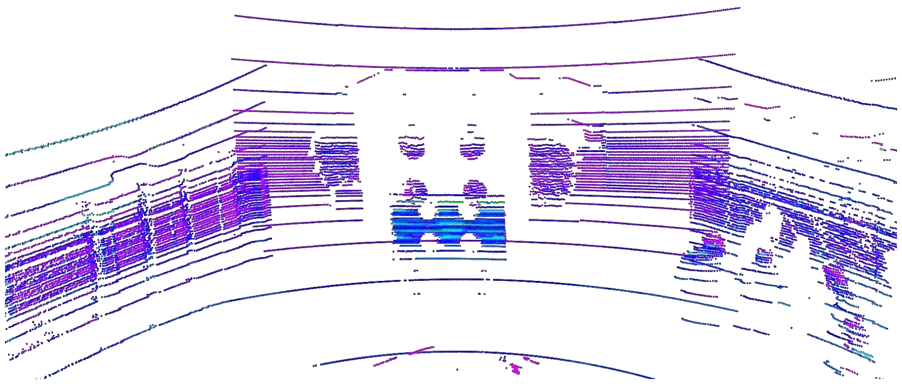


Figure 3.4: LiDAR point cloud of a sample calibration scenario

Before feeding the data to the segmentation stage, pass-through filters are applied in the three cartesian coordinates to remove points outside the area where the target is to be placed, avoiding spurious detections that could slow down the processing. The limits of the pass-through filters must be set according to the location and size of the sensors' overlapping area. The resulting cloud, $\mathcal{P}_1^L$, must represent both the calibration target and the points behind it, visible from the LiDAR through the holes.

As a first step towards segmenting the holes in the pattern, the points representing the edges of the target must be extracted. For the

LiDAR modality, we follow the method in [102] to find depth discontinuities. Each point in the cloud, $\mathbf{p}_i \in \mathcal{P}_1^L$, is assigned a magnitude representing the depth gradient with respect to their neighbors:

$$p_{i,\Delta} = \max(p_{i-1,r} - p_{i,r}, p_{i+1,r} - p_{i,r}, 0) \tag{3.1}$$

Where $p_{i,r}$ is the range measurement given by the sensor for the point $\mathbf{p}_i$ (i. e., the spherical radius coordinate), and $\mathbf{p}_{i-1}$ and $\mathbf{p}_{i+1}$ are the points adjacent to $\mathbf{p}_i$ in the same scan plane. Then, we filter out all points $\mathbf{p}_i$ with a discontinuity value $p_{i,\Delta} < \delta_{\text{discont},L}$, resulting in $\mathcal{P}_2^L$. Note that this procedure assumes that measures from rays passing through the holes exist, so they must collide with some solid located behind the target within the measurement range of the LiDAR.

3.1.1.2 *Stereo data preprocessing*

When one of the sensors to be calibrated is a stereo-vision system, data processing starts by converting the raw image pair into a disparity map using a stereo matching algorithm. For this map to be obtained, the correspondences between the projection of every 3D point in the scene into the image planes of both cameras have to be computed. In most common stereo rigs, these two planes are coincident, and the images can be rectified so that their epipolar lines are aligned with their horizontal axes (i. e.for each point in the left camera, its counterpart in the right image is located in the same vertical coordinate). Hence, for a 3D point P, being $p_1 = (u_1, v_1)$ and $p_2 = (u_2, v_2)$ the image coordinates of its projection in the left and right images, respectively, and $v_1 = v_2$, its disparity value can be calculated as $d = u_2 - u_1$.

In our experiments, the Semi-Global Block Matching (SGBM) variant of [76] is used, which we found reasonably accurate for disparity estimation. Note that, when this modality is involved, the calibration target is expected to have some texture (e. g., wood grain) so that the stereo correspondence problem can be successfully solved. However, we found that the intensity differences caused by the pattern borders themselves are generally sufficient.

After the disparity map has been computed, a point cloud $\mathcal{P}_0^S$ can be built. Since the system is assumed canonical and the baseline between cameras known, the image coordinates and the disparity value of every pixel can be used to obtain the corresponding 3D points using the pinhole model. Figure 3.5 shows the disparity map and 3D point cloud of a pair of images captured by a stereo rig.

Similar to the LiDAR branch, pass-through filters are applied to $\mathcal{P}_0^S$ to limit the search space. However, for the stereo modality, the extraction of the points representing the target edges in the filtered cloud, $\mathcal{P}_1^S$, relies on the appearance information provided by one of the images of the stereo pair. Concretely, a Sobel filter is applied over the image, and then, all points in $\mathcal{P}_1^S$ that map to pixels with a low value

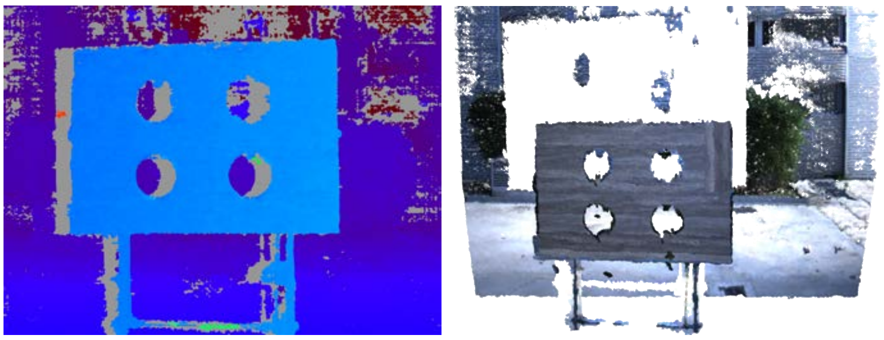

(a) (b)

Figure 3.5: Disparity map (a) and 3D point cloud (b) of a sample calibration scenario captured with a stereo camera

in the Sobel image (smaller than $\tau_{\text{sobel,S}}$) are filtered out, producing $\mathcal{P}_2^S$. In this way, edge segmentation is less affected by inaccuracies in border localization, which are frequent in stereo matching.

3.1.1.3 *Range data*

The steps followed to segment the pattern holes in the preprocessed point clouds are common for both the LiDAR and stereo modalities. as can be seen in Figure 3.6. The intended outcome is an estimate of the 3D location of the centers in sensor coordinates.

Plane Segmentation: First of all, RANSAC is applied to $\mathcal{P}_1$ (the cloud resulting from the pass-through filters, either $\mathcal{P}_1^L$ or $\mathcal{P}_1^S$), which provides a plane model π representing the calibration target. To ensure the model's accuracy, we use a tight RANSAC threshold δ_{plane}, which neutralizes all the points representing extraneous objects and impose that the plane must be roughly vertical in sensor coordinates, with a tolerance α_{plane}. If it is impossible to find a plane that fits the data, the current frame is discarded.

Afterwards, the plane model π is employed in $\mathcal{P}_2$ (i. e., the cloud representing the edges of the pattern) to remove all the points not belonging to the plane. A threshold of δ_{inliers} is considered for the inliers. Consequently, the new cloud $\mathcal{P}_3$ contains only points representing the edges of the calibration target; that is, the outer borders and the holes.

Transformation to 2D Space: As all the remaining points belong to the same plane, dimensionality reduction is performed at this point. This is implemented by transforming $\mathcal{P}_3$ so that the XY-plane coincides with π and projecting all the 3D points onto π. Points in the resulting $\mathcal{P}_4$ cloud are, therefore, in 2D space.

Circle Segmentation: Next, a model of the pattern holes present in $\mathcal{P}_4$ is extracted through 2D circle segmentation. This step is performed iteratively in a process that seeks out the most supported circle and removes its inliers before starting the search for the next one. Iterations continue until the remaining points are not enough to describe a circle. If at least four circles have been found, the procedure moves forward; otherwise, the current frame is not considered. Inliers are required to be below a threshold of δ_{circle} from the model, and only circles within a radius tolerance of δ_{radius} are considered.

The points found in the circle segmentation procedure are checked for geometric consistency with the dimensions of the pattern. To that end, the centers are grouped in sets of four, and the dimensions of the rectangle that they form (diagonal, height, width, and perimeter) are compared with the theoretical ones, with a tolerance $\delta_{\text{consistency}}$ expressed as a percentage of deviation from the expected values. Presumably, only one set of centers will fulfill these restrictions; if either none or more than one sets pass the check, the frame is discarded. This step is intended to prune out spurious detections that may occur due to confusion with other elements in the scene.

Once the holes are correctly identified, their centers are converted back from the 2D space defined by π to the 3D space in sensor coordinates, forming the cloud $\mathcal{P}_p$. Note that $\mathcal{P}_p$ must contain exactly four points.

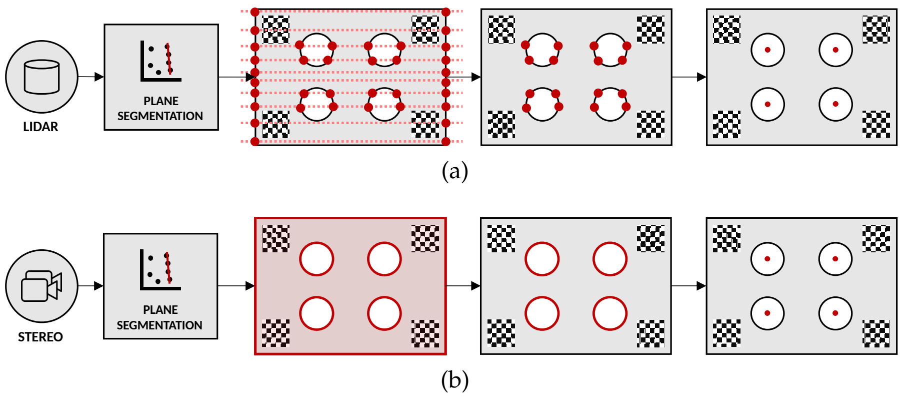


Figure 3.6: Target segmentation stages in the LiDAR (a) and stereo (b) range modalities: plane segmentation, target and circles detection, and keypoints extraction

3.1.1.4 *Monocular data*

If the sensor to be calibrated is a monocular camera, the extraction of the reference points requires the detection of ArUco markers, which provide the cues necessary to retrieve the geometry of the target.

ArUco markers are synthetic square markers made of a black border and an inner binary matrix designed to allow its unequivocal identification [58]. In our calibration target, four ArUco markers are used, one on each corner; due to this location, they do not affect either target or hole detection by other modalities.

As both the intrinsic parameters of the camera and the marker dimensions are known, it is possible to retrieve the 3D pose of each marker with respect to the camera through the resolution of a classic perspective-n-point (PnP) problem. In our implementation, we handle our four-marker setup as an *ArUco board*, which allows estimating the pose of the calibration target accurately by using all the markers jointly. An iterative Levenberg-Marquardt optimization is carried out to find the board pose that minimizes the reprojection error [167], using the average pose of the four individual markers as an initial guess. As a result, the 3D position of the center of the board is obtained, along with its orientation in space.

To generate a set of four points equivalent to the $\mathcal{P}_p$ clouds obtained from range data, we extract the points representing the center of the reference holes by taking advantage of the fact that their relative positions in the calibration target are known. These points constitute the resulting cloud $\mathcal{P}_p^M$.

Figure 3.7 shows the different steps for the estimation of the target keypoints in the image space.

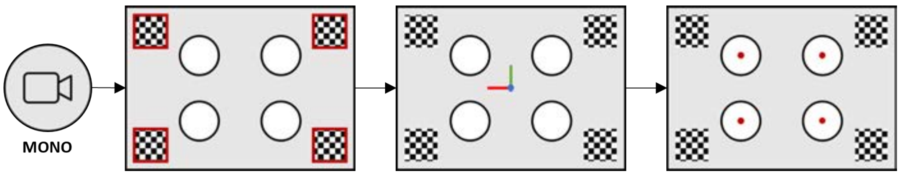


Figure 3.7: Target segmentation stages in the image data: ArUco detection, estimation of the target's 3D pose, and keypoints extraction

3.1.2 *Point aggregation*

At the end of the segmentation stage, two clouds $\mathcal{P}_p$ must have been generated, one per sensor involved in the calibration. Each represents the 3D location of the reference points (the centers of the target holes) for a single static scene in the coordinate frame attached to the respective sensor.

These data would be enough to find the transform representing the relative pose of the sensors. However, different sources of noise inherent to the method (e. g., sensor noise and non-deterministic procedures such as RANSAC) can affect the accuracy of the result. To increase the robustness of the algorithm, we augment the information available by repeatedly applying the segmentation step and accumulating the results in two different ways.

3.1.2.1 *Accumulation over several frames*

Since it is usually feasible to maintain the calibration scene static for a certain period, we accumulate the points that compose $\mathcal{P}_p$ over N data frames to generate $\mathcal{P}'_p$ and then perform Euclidean clustering on this cumulative cloud. If more than four clusters are found, data is considered unreliable and not used for registration; otherwise, cluster centroids, stored in the resulting cloud $\mathcal{P}_c$, are employed as a consolidated estimate of the centers' locations. The clustering parameters, namely cluster tolerance δ_{cluster}, minimum cluster size $N_{\text{cluster,min}}$, and maximum cluster size $N_{\text{cluster,max}}$, depend on the number of iterations taken into account.

According to the experimental results shown later, $N = 30$ is usually adopted, which offers satisfactory results in a limited timeframe. Naturally, the time necessary to complete the procedure depends on the sensor's framerate but is rarely longer than a few seconds.

3.1.2.2 *Accumulation over several target poses*

As will be shown later, the method can deliver an estimated calibration with a single target position. However, it is possible to increase the accuracy of the estimation by considering more than four reference points. If the segmentation procedure is repeated for M different poses of the calibration target with respect to the sensors, the $\mathcal{P}_c$ clouds obtained with each pose are accumulated in a $\mathcal{P}'_c$ cloud where $4 \times M$ reference points are available to perform the registration stage.

If the poses of the target are selected so that the resulting reference points are not coplanar and cover a wide range of distances from the sensors, the additional constraints provided by the new poses solve possible ambiguities and improve the overall quality of the final calibration.

3.1.3 *Registration*

As a result of the segmentation stage, two clouds $\mathcal{P}'_c$, one per sensor, are obtained. They contain the estimated 3D location of the centers of the circles expressed in sensor coordinates; that is, with respect to a frame attached to the sensor.

The goal of the registration step is to find the optimal parameters $\hat{\theta}$ so that when the resulting transformation $\hat{\mathbf{T}}$ is applied, it results in the best alignment (i. e., minimum distance) between the reference points obtained from both sensors. Note that the approach has been designed to handle only two sources at a time so that the problem can be viewed as a multi-objective optimization with $4 \times M$ objective functions.

Before that, the registration procedure needs that every point in one of the $\mathcal{P}'_c$ clouds is correctly paired with its homologous in the other cloud; that is, pairs of points representing the same reference points in both clouds must be associated.

3.1.3.1 *Point association*

A point association procedure has been developed to avoid assuming that reference points in both $\mathcal{P}_c$ clouds have the same ordering in their respective coordinate frames. Note that this condition would not be fulfilled when calibrating a front-facing 360° LiDAR and a rear-looking camera, for instance.

Therefore, we convert the four centers in each $\mathcal{P}_c$ to spherical coordinates and only presume that the point that appears highest in the cloud, that is, the one with the lowest inclination angle, belongs to the upper row of the calibration target (i. e., either the top-left or the top-right circle).

Distances from this point to the other three determine the correct ordering. In that way, each point can be associated with the circle in the calibration target that it represents: top-left (tl), top-right (tr), bottom-left (bl), and bottom-right (br). The procedure is repeated for each of the M poses of the calibration target, so that each point $\mathbf{p}_i$ in $\mathcal{P}'_c$ is provided with labels $p_{i,a}$ and $p_{i,m}$ containing the hole in the pattern and the pose to which it corresponds, respectively:

$$p_{i,a} \in \{tl, tr, bl, br\} \tag{3.2}$$

$$p_{i,m} \in \{1, \dots, M\} \tag{3.3}$$

3.1.3.2 *Solution*

Later, the two resulting clouds, obtained from two arbitrary modalities X and Y and denoted here by $\mathcal{P}'^X_c$ and $\mathcal{P}'^Y_c$, undergo a Umeyama registration procedure [176], responsible for finding the rigid transformation that minimizes the distance between their corresponding points. That is, assuming that the points in each cloud, $\mathbf{p}_i^X \in \mathcal{P}'^X_c$ and $\mathbf{p}_i^Y \in \mathcal{P}'^Y_c$, are ordered so that, $\forall i$:

$$p^X_{i,a} = p^Y_{i,a} \wedge p^X_{i,m} = p^Y_{i,m} \tag{3.4}$$

Then, the desired transformation matrix $\hat{\mathbf{T}}_{XY}$ is the one that minimizes the least-squares error criterion given by:

$$\frac{1}{4 \cdot M} \sum_{i=1}^{4 \cdot M} \|\mathbf{p}_i^X - \mathbf{T}_{XY}\mathbf{p}_i^Y\|^2 \tag{3.5}$$

This optimization problem is solved through singular value decomposition (SVD) and provides a closed-form solution from which the set of parameters expressing the relative position between both sensors, $\hat{\theta}_{XY}$, can be straightforwardly retrieved. Conveniently, the Umeyama method handles singular situations where all the points are coplanar, as is the case when a single pattern position ($M = 1$) is used, thus avoiding misjudging them as reflections.

3.1.4 *Experimental results*

The validation of the proposed approach has been addressed from two different perspectives. First, tests on a realistic synthetic test suite have been performed to retrieve plentiful quantitative data with respect to perfect ground truth. Second, the method has also been applied in a real environment to prove its validity in real use cases.

All the experiments were carried out without user intervention, except for the setup of the scenario and the tuning of the pass-through filters mentioned in Section 3.1.1, which must be coarsely adapted to the location of the calibration pattern. The rest of the parameters were set to a fixed value for all the experiments, as reported in Table 3.1. Unless otherwise stated, reference points are accumulated over 30 frames ($N = 30$); however, it should be noted that every frame delivered by the sensors counts toward this limit, regardless of whether a four-point solution has been extracted from it. Conversely, only successful frames (N') are taken into account for the cluster size limits.

Table 3.1: Setting of constant parameters in the method

Parameter	Description
Preprocessing (edge segmentation)	
$\delta_{\text{discont},L} = 10\,\text{cm}$	Distance threshold (LiDAR)
$\tau_{\text{sobel},S} = 128$	Sobel intensity threshold (stereo)
Plane segmentation	
$\delta_{\text{plane}} = 10\,\text{cm}$	Distance threshold
$\alpha_{\text{plane}} = 0.55\,\text{rad}$	Angular tolerance
$\delta_{\text{inliers}} = 10\,\text{cm}$	Distance threshold for outlier removal
Circle segmentation	
$\delta_{\text{circle},L} = 5\,\text{cm}$	Distance threshold (LiDAR)
$\delta_{\text{circle},S} = 1\,\text{cm}$	Distance threshold (stereo)
$\delta_{\text{radius}} = 1\,\text{cm}$	Radius tolerance (stereo)
$\delta_{\text{consistency}} = 6\,\text{cm}$	Geometry consistency tolerance
Clustering	
$N_{\text{cluster,min}} = \frac{1}{2}N'$	Minimum cluster size
$N_{\text{cluster,max}} = N'$	Maximum cluster size
$\delta_{\text{cluster}} = 5\,\text{cm}$	Cluster tolerance

3.1.4.1 *Synthetic test environment*

As stated before, the quantitative assessment of the set of extrinsic parameters relating two sensors in space is a nontrivial issue, as it is impossible, in practice, to obtain exact ground truth. Most works dealing with extrinsic calibration in the literature use manual annotations [62] or other approximations such as scene discontinuities [102].

In order to provide a comprehensive set of data describing the performance of the presented method, we propose a novel evaluation approach based on a synthetic test suite, where the exact-ground truth of the relative transformation between sensors is available. The open-source Gazebo simulator [93] was used, and the operation modes of the three sensor modalities considered in this work (i. e., LiDAR, and stereo and monocular cameras) were faithfully replicated, taking into account the specifications of real devices in terms of field of view, resolution, and accuracy. Table 3.2 shows the set of devices used in the experiments.

Table 3.2: Sensor models used in the synthetic environment

Device	Modality	Resolution[a]	HFOV
FLIR Bumblebee XB3	Stereo	1280×960	43°
Velodyne VLP-16	LiDAR	16 layers, 0.2°	360°
Velodyne HDL-32	LiDAR	32 layers, 0.2°	360°
Velodyne HDL-64	LiDAR	64 layers, 0.2°	360°
FLIR Blackfly S 31S4C-C	Monocular	2048×1536	85°

[a] Image resolution, for cameras, and number of channels and horizontal (azimuth) angular resolution, for LiDAR scanners.

Remarkably, the different LiDAR devices employed in the experiments are fairly representative of the diversity of laser scanners available in the market regarding the number of scanning layers and their distribution, thus enabling the assessment of the adaptability of the reference point extraction approach.

A model of the fiducial calibration target was also created by mimicking the appearance of the actual wooden embodiment shown in Figure 3.2a. In the experiments, the target was placed with a wall behind so that LiDAR beams going through the circular holes reach a surface, generating the necessary gradient between foreground and background points.

Gaussian noise $\epsilon \sim \mathcal{N}(0, (K\sigma_0)^2)$ was applied to the sensors' captured data, with $\sigma_0 = 0.007$ and $\sigma_0 = 0.008\,\mathrm{m}$ for the pixel intensities (expressed in a range from 0 to 1) and the LiDAR distances, respectively. The noise factor K allows simulating ideal, noise-free environments ($K = 0$), realistic environments ($K = 1$), and noisy environments ($K = 2$). $K = 1$ is used by default.

Despite the eventual domain gap, experiments in this controlled setup enable systematic analysis and provide valuable insight into the method that will be otherwise unfeasible. Experimentation in the synthetic suite can be divided into three different focus points: reference point extraction, calibration with a single target position, and multi-pose calibration.

3.1.4.2 *Single-sensor experiments*

The first set of tests is aimed to analyze the accuracy in the extraction of the reference points from the four circular openings in the calibration target. Four different relative positions between sensor and calibration pattern, combining translations and rotations, were considered. Table 3.3 shows the position of the calibration pattern in sensor coordinates for each of these configurations, assuming that axes are defined as customary in LiDAR devices; i. e., x pointing forward, y to the left, and z upward. Besides, translation is denoted by (t_x, t_y, t_z), whereas (r_x, r_y, r_z) represent roll, pitch, and yaw rotations (in radians).

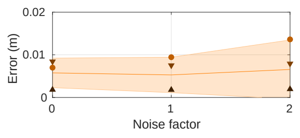
Table 3.3: Relative sensor-target poses for reference point extraction assessment

	Translation (m)				Rotation (rad)		
Cfg.	t_x	t_y	t_z	$\|\mathbf{t}\|$	r_x	r_y	r_z
P1	2.00	0.00	−0.50	2.06	0.0	0.0	0.0
P2	3.63	−0.50	−0.28	3.67	0.8	0.0	0.0
P3	5.38	−0.10	−0.50	5.41	0.0	−0.2	0.0
P4	6.50	−1.39	−1.43	6.80	0.0	0.0	−0.4

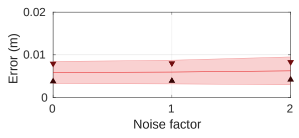
These setups were purposely chosen to investigate the limits of the reference point extraction branches. In fact, the method was unable to provide results in some extreme configurations; concretely, with the VLP-16 LiDAR in P3 and P4, the HDL-32 LiDAR in P4, and the stereo camera in P4 as well. In the case of the LiDAR scanners, their limited resolution made it impossible to find the circles at far distances, whereas the stereo was affected by the substantial degradation in depth estimation that this modality suffers as the distance increases. In typical use cases, it should be possible to avoid these situations by restricting the pattern locations to a reasonable range of distances with respect to the sensors.

The reference point localization performance was measured by determining the distance between the estimation provided by the approach and the ground-truth position of the center of the corresponding circle. The assignment was unambiguous in all cases and could be straightforwardly performed based on distance. Results were aggregated over three iterations for each pose and modality to account for the effect of the stochastic processes in the pipeline (e. g., RANSAC segmentations).

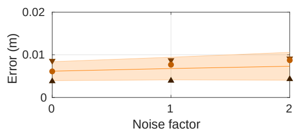
Firstly, Figure 3.8 analyzes the effect of noise in the reference points location error. The results show that the procedure is highly robust to noise in all the modalities, given that the impact is limited to an increase in the standard deviation of the error in noisy situations ($K = 2$). In all cases, the error is well below $1\,\text{cm}$ for the P1 and P2 configurations (upwards and downwards triangle markers in the graph), whereas P3 (circle markers) and, especially, P4 (square markers) involve a significant increase across all the noise levels. This fact is particularly noticeable for the monocular modality (please note the different scale in the y-axis), where the accuracy in the detection of the ArUco markers proves to be much more sensitive to the size of their projections onto the image than to the pixel-wise noise.



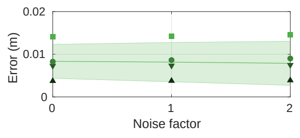
(a) Stereo



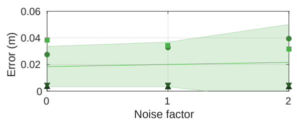
(b) VLP-16



(c) HDL-32



(d) HDL-64



(e) Monocular

Figure 3.8: Euclidean error in single-frame reference point localization vs. noise level (K) for each tested modality. The mean is depicted as a solid line, whereas the shaded area represents the standard deviation. Mean errors for each pose are depicted as individual markers (P1: upwards triangle, P2: downwards triangle, P3: circle, P4: square)

Focusing on the realistic noise setup ($K = 1$), Figure 3.9 shows the single-frame estimation error in each of the four configurations, further highlighting the relative position between sensor and calibration pattern as a significant factor. Apart from the most challenging configurations, the reference point localization proves accurate and precise across all the modalities, with LiDAR scanners exhibiting high robustness even in P3 and P4. As mentioned before, monocular struggles with these configurations but shows an excellent performance in P1 and P2.

The effect of the point aggregation and clustering strategy introduced in Section 3.1.2.1 is investigated in Table 3.4, where the root-

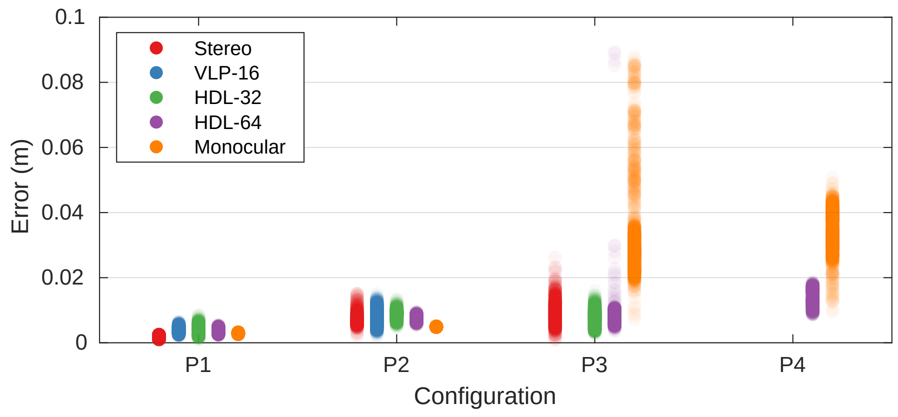


Figure 3.9: Euclidean error in single-frame reference point localization, for each tested configuration and modality, with realistic noise

mean-square error (RMSE) of single-frame estimations and cluster centroids at 30 iterations are compared under realistic noise conditions. The cluster centroid proves to be a consistently better representation of the reference points than the single-frame estimation in all cases, achieving a more remarkable improvement in situations with high dispersion; e. g., stereo in P3 (25.22% error reduction) or HDL-64 also in P3 (19.02% error reduction).

Table 3.4: RMSE (mm) in reference point location using a single-shot estimation (S) and the cluster centroid at $N = 30$ (C)

	P1		P2		P3		P4	
	S	C	S	C	S	C	S	C
Stereo	1.84	1.83	7.82	6.83	10.11	7.56	-	-
VLP-16	3.98	3.87	8.39	8.27	-	-	-	-
HDL-32	4.12	3.98	8.82	8.61	8.02	7.41	-	-
HDL-64	3.81	3.74	7.38	7.29	9.99	8.09	14.43	14.28
Mono	2.82	2.80	4.92	4.91	35.78	35.58	34.70	33.87

Once again, the results suggest that the accuracy in reference point extraction is primarily impacted by the relative pose of the calibration target and, to a lesser extent, by the sensor modality. In contrast, the density of LiDAR data seems to have little influence on the results, although minor differences in the way laser beams interact with the target depending on the layer distribution produce a few counterintuitive results.

3.1.4.3 *Single-pose experiments*

Next, the full calibration pipeline will be evaluated considering only a single target position; that is, for $M = 1$. To that end, four combinations representative of real automotive sensor setups were analyzed:

A. HDL-32/HDL-64 (LiDAR/LiDAR)

B. Monocular/HDL-64 (camera/LiDAR)

C. Monocular/monocular (camera/camera)

D. Stereo/HDL-32 (camera/LiDAR)

Setups A and C embody situations where several devices of the same modality are included in the same sensor setup to enhance the field of view or the resolution of the captured data, whereas setups B and D exemplify setups aimed at camera-LiDAR sensor fusion. Both situations are frequently found in the onboard perception literature, even jointly on the same platform, e. g., [26].

For each setup, the three different relative positions between sensors reported in Table 3.5 were considered. They were picked as a challenging set of configurations involving a wide range of translations and rotations. Representative pictures of these poses in the synthetic test suite are depicted in Figure 3.10. As in the previous case, three different iterations were considered in the results for each possibility. In all cases, the calibration pattern was placed arbitrarily in a location suitable for both sensors. Like in the per-sensor analysis, different distances to the target are used to further study its effect on final calibration.

Table 3.5: Transformation parameters of the different calibration scenarios

Cfg.	t_x (m)	t_y (m)	t_z (m)	ψ (rad)	θ (rad)	ϕ (rad)
P1	−0.300	0.200	−0.200	0.300	−0.100	0.200
P2	−0.128	0.418	−0.314	−0.103	−0.299	0.110
P3	−0.433	0.845	1.108	−0.672	0.258	0.075

The analysis is now focused on the final calibration result. Therefore, following [62], results are given in terms of the linear (e_t) and angular (e_r) errors between the estimated rigid-body transformation and the ground truth:

$$e_t = \|\hat{\mathbf{t}} - \mathbf{t}\| \tag{3.6}$$

$$e_r = \angle(\hat{\mathbf{R}}^{-1}\mathbf{R}) \tag{3.7}$$

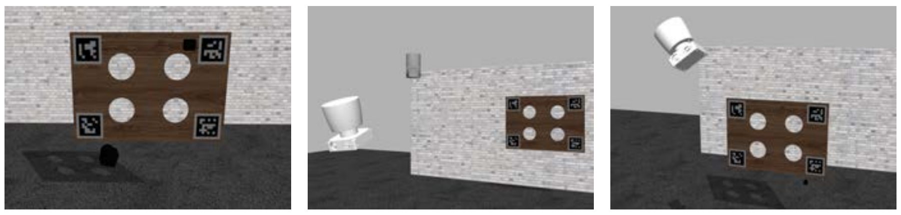
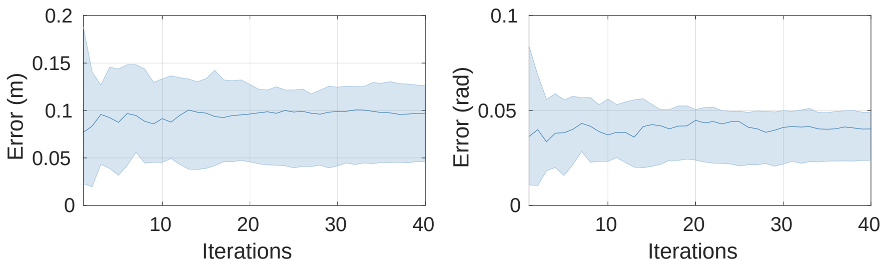

(a) (b) (c)

Figure 3.10: Sensor setups for the single-pose experiments in the synthetic environment: P1 (a), P2 (b), and P3 (c)

Where $\mathbf{t}$ is the translation vector, $\mathbf{t} = (t_x, t_y, t_z)$, and $\mathbf{R}$ the 3×3 rotation matrix, representing the r_x, r_y, and r_z rotations; both elements compose the transformation matrix:

$$\mathbf{T} = \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{3.8}$$

In the first place, the effect of the number of data frames used for reference point extraction, N, was studied. Figure 3.11 aggregates the error for every setup and configuration when the calibration procedure is stopped at a point in the $N = [1, 40]$ interval. The results suggest that the method can provide a reliable estimation of the extrinsic parameters in a wide range of values of N, even with very few iterations. Nevertheless, $N = 30$ offers a fair accuracy-time tradeoff where outliers are extremely rare.



(a) (b)

Figure 3.11: Linear (a) and angular (b) calibration errors vs. number of iterations considered for clustering (N). The solid line represents the median and the shaded area, the interquartile range (IQR)

Table 3.6 shows the linear (e_t) and angular (e_r) calibration errors sorted by sensor setup and configuration for $N = 30$. Monocular/-monocular calibration (setup C) shows excellent accuracy and precision, in line with the reference point extraction results, featuring errors up to 100 times smaller than the rest of the setups. On the contrary, the stereo/HDL-32 (setup D) presents higher errors, likely due to the difficulties found by the stereo matching procedure to provide an accurate depth estimation at the distance where the pattern was placed in the experiments. Despite this, we observed that the implementation of the ArUco detector in use (OpenCV *aruco* module) was considerably more sensitive to light conditions than the stereo matching approach, so the method based on the stereo modality might still be useful in some instances. Overall, the results are reasonably accurate, even though the single-target situation poses a very challenging case for registration due to the coplanarity of the reference points, which can eventually become a source of ambiguity.

Table 3.6: Mean (and standard deviation) of linear (e_t) and angular (e_r) calibration errors for different setups using a single target pose ($M = 1$)

Set.	Error	P1	P2	P3
A	e_t (cm)	8.94 (1.49)	17.39 (2.13)	11.95 (1.56)
	e_r (10^{-2} rad)	4.36 (0.72)	3.91 (0.48)	5.80 (0.78)
B	e_t (cm)	10.34 (0.53)	4.31 (0.29)	9.68 (0.22)
	e_r (10^{-2} rad)	5.08 (0.26)	2.23 (0.13)	4.74 (0.12)
C	e_t (cm)	0.17 (0.01)	0.08 (0.00)	0.16 (0.00)
	e_r (10^{-2} rad)	0.03 (0.01)	0.04 (0.00)	0.04 (0.00)
D	e_t (cm)	9.62 (1.12)	47.02 (1.49)	31.60 (2.95)
	e_r (10^{-2} rad)	2.85 (0.34)	14.87 (0.47)	8.75 (0.84)

Table 3.7 shows a comparison of the proposed approach with two single-pose LiDAR-camera calibration methods in the literature: the one by Geiger et al. [62], which estimates both the intrinsic and extrinsic parameters of the sensors with only one shot, and the one proposed by Velas et al. [181], which makes use of a calibration pattern very similar to ours. For a fair comparison, all the methods were fed with sensor data from the synthetic test suite. The sensor setup was composed of the stereo camera and the HDL-64 LiDAR introduced in Table 3.2. We consider the two available options for reference point extraction in visual data: stereo and monocular, the latter employing the left image of the stereo rig as input. The errors were averaged over the same three poses used in the previous experiments.

According to these results, the stereo and mono alternatives yield similar accuracy, significantly outperforming the other methods. Particularly noteworthy is the substantial improvement in angular error brought about by our approach, which stands out as the only one suitable for data fusion at far distances. These results prove that the baseline method, requiring a single pose of the calibration pattern

Table 3.7: Mean (and standard deviation) of linear (e_t) and angular (e_r) calibration errors using a single target pose ($M = 1$)

Method	e_t (m)	e_r (rad)
Geiger et al. [62]	0.93 (0.36)	1.30 (1.35)
Velas et al. [181]	0.99 (1.17)	0.35 (0.37)
Ours (Stereo-LiDAR)	0.12 (0.09)	0.04 (0.03)
Ours (Monocular-LiDAR)	0.12 (0.12)	0.04 (0.03)

($M = 1$), works acceptably and provides a solid foundation for the full version with $M > 1$.

3.1.4.4 *Multi-pose experiments*

The last set of experiments focuses on the aggregation strategy presented in Section 3.1.2.2, where the registration procedure is performed on $M \times 4$ points coming from M different calibration target positions. The sensor setups are identical to those used in the single-pose tests, but only the first configuration (P1) has been selected. For every sensor pair, the calibration pattern was moved along five different poses within a range of 5×5 m in front of the devices, up to 6 m in depth. To avoid the eventual bias introduced by the order in which these positions are used, results are obtained through three different iterations in which the sorting is changed.

The evolution of the linear and angular calibration errors with M follows an almost-exponential decay for all the tested setups, as shown in Figure 3.12 (please note the logarithmic scale). Only by introducing an additional target pose, an average reduction of 61.2% (linear) / 68.15% (angular) can be achieved. Increasing the number of poses is positively beneficial up to $M = 3$; higher values lead to mixed effects ranging from almost neutral to slightly positive. Nevertheless, when five poses are employed, the average errors drop by 85.42% (linear) / 87.01% (angular). The largest decreases correspond to the HDL-32/HDL-64 setup, where the reduction is around 97% for both kinds of errors, yielding a final calibration with a deviation of 6.5 mm and 0.002 rad from the ground truth.

The proposed approach has been compared with the state-of-the-art method recently introduced by Zhou et al. [208], aimed at LiDAR-camera calibration using one or several views of a checkerboard. To that end, we used the implementation included in the MATLAB Lidar Toolbox [171]. Tests were performed with the monocular/HDL-64 sensor setup, using $M = 2$ and $M = 3$ poses of the respective calibration patterns. Mean calibration errors by both methods are shown in Table 3.8.

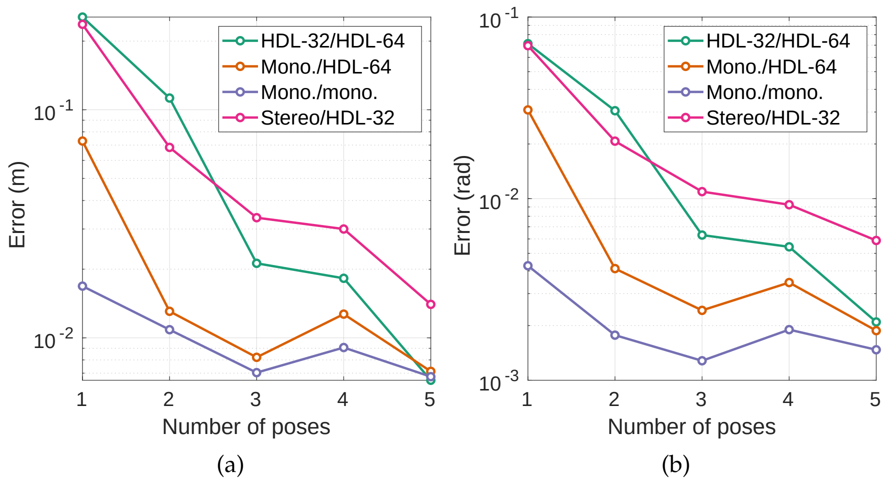


Figure 3.12: RMSE of the linear (a) and angular (b) calibration errors (m and rad) vs. number of calibration poses (M) for four sensor setups

Table 3.8: Mean of linear (e_t) and angular (e_r) calibration errors using several target poses ($M > 1$)

	$M = 2$		$M = 3$	
Method	e_t (cm)	e_r (10^{-2} rad)	e_t (cm)	e_r (10^{-2} rad)
Zhou et al. [208]	1.51	0.63	1.08	0.50
Ours	1.15	0.39	0.82	0.24

As apparent from the results, the performance of both approaches is comparable, although our solution achieves consistent improvements that even exceed 50% for the angular error when $M = 3$. These results confirm the effectiveness of the aggregation of reference points across different target locations, providing a calibration solution that features subcentimeter accuracy.

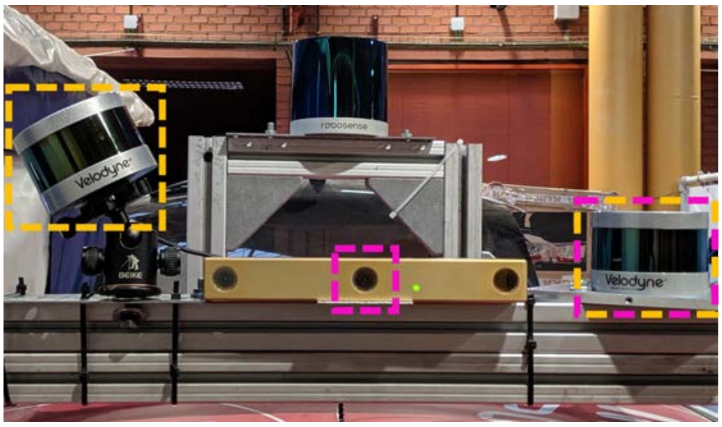
3.1.4.5 *Real test environment*

The set of experiments presented in the previous section offers a systematic and exact analysis of the performance of the proposed calibration method. Nevertheless, experiments in a real use case were also carried out to validate the applicability of the approach, assessing its adequacy to meet the requirements of the intended application.

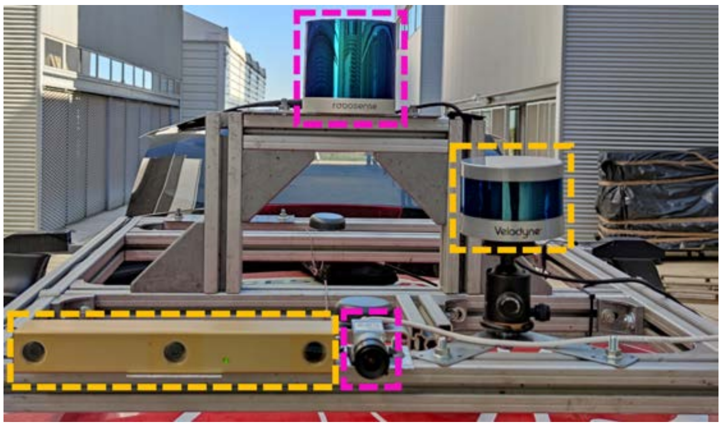
The CNC manufactured calibration targets shown in Figure 3.2 were employed in the process. We performed two rounds of experiments using different sensor stacks to test the multiple capabilities of the approach adequately. Both configurations were mounted on an experimental vehicle's roof rack.

For the first round, depicted in Figure 3.13a, two Velodyne VLP-16 LiDARs and a Bumblebee XB3 camera were mounted in a rig, with rotations emulating the ones that can be found in vehicle setups. In this step, we performed two different calibration procedures: monocular/LiDAR, involving one of the cameras of the stereo system and one of the LiDAR scanners, and LiDAR/LiDAR, between the two VLP-16 devices.

In the second round, we used the topology shown in Figure 3.13b, with the Bumblebee XB3 stereo camera, a Basler acA2040-35gc camera with a 90° HFOV lens, a Robosense RS-LiDAR-32, and a Velodyne VLP-16 Hi-Res LiDAR. Here, we tested three different calibration alternatives: stereo/LiDAR, linking the XB3 and the VLP-16 Hi-Res scanner, monocular/LiDAR, this time with the wide-angle Basler camera and the RS-LiDAR-32, and monocular/monocular, between two of the cameras of the stereo system.



(a)



(b)

Figure 3.13: The two sensor setups used in the real experiments. Calibrated pairs of devices are framed with the same color

The sensors used in these experiments have very different features from each other; thus, the VLP-16 Hi-Res LiDAR has a tighter layer distribution than the regular VLP-16, whereas the RS-LiDAR-32 has twice as many scan planes, but they are irregularly spread, with much higher density around the central area. All the devices pose their own challenges for calibration, as the set of locations where the four circles of the calibration pattern are fully visible is much more limited than, for example, with the Velodyne HDL-64. As for the cameras, the narrow field of view exhibited by the XB3's cameras (43°) contrasts with the wide angle of the Basler. Overall, the number and variety of sensors and combinations used in the experiments ensure the generality of the results. As with the synthetic experiments, points were extracted from the accumulation of $N = 30$ frames, and $M = 5$ target poses were used. The rest of the parameters remained unchanged from Table 3.1.

Ground truth of the relative position between sensors was not available, but some illustrative statistics about the performance of the calibration procedure with real sensors are presented below. On the one hand, Figure 3.14 shows the dispersion of the estimated reference points across different poses of the calibration pattern, each represented by a point. Data from the five separate calibration procedures are included. The black line represents the mean, the dark shadow spans the standard deviation, and the light shadow covers 1.96 times the standard error of the mean.

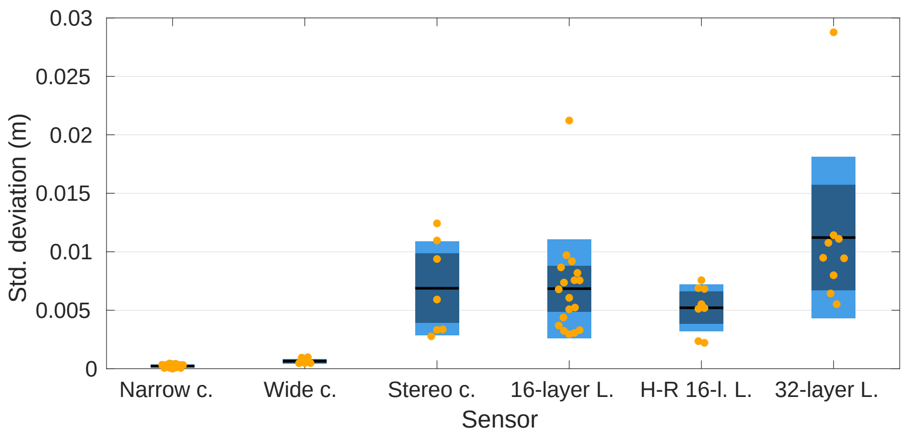


Figure 3.14: Dispersion in the localization of the reference points in real experiments for the different camera (c.) and LiDAR (L.) devices

The results confirm that the dispersion in the LiDAR and stereo modalities is significantly higher than the one exhibited by its monocular counterpart, as suggested by the tests in the synthetic environment. However, the deviation is still small enough to enable higher accuracy in registration. It is possible to observe the presence of outliers corresponding to some particular poses of the calibration pattern; however, they do not raise relevant issues for the multi-pose calibration as they are well mitigated by the rest of the poses.

On the other hand, Figure 3.15 shows the difference, measured in linear and angular errors, of the calibrations performed with $M \in [1, 4]$ versus the final result with $M = 5$. The results validate the conclusion drawn in the previous section: using several pattern poses ($M > 1$) causes significant changes in the calibration result up to 3 poses, where it plateaus.

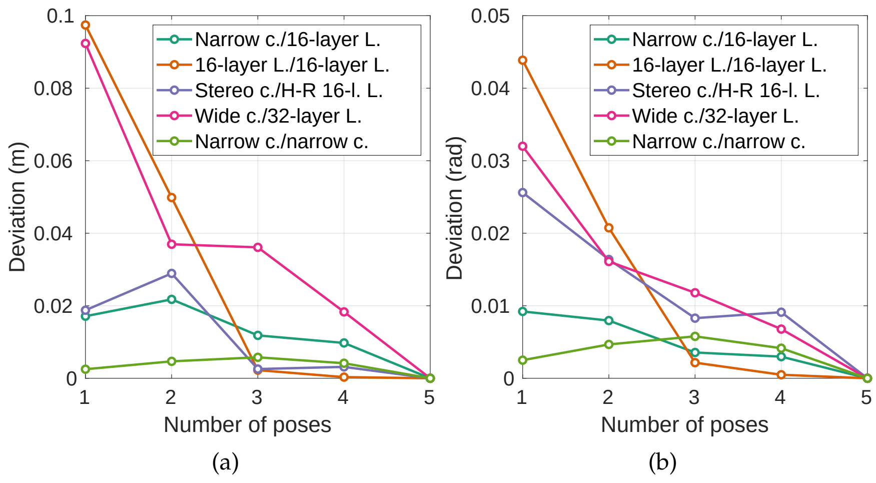


Figure 3.15: Linear and angular deviation from the final calibration result at $M = 5$ in real experiments

In the particular case of the narrow-angle camera/narrow-angle camera calibration, the result can be compared with the baseline provided by the manufacturer for the rectified stereo pair, yielding an average error across coordinates of 2.73 mm.

Finally, qualitative results are presented through the projection of LiDAR point clouds onto the image plane for different sensor setups. Figure 3.16 depicts various examples of traffic scenarios captured by a sensor stack composed of a narrow-angle monocular camera and two 16-layer LiDARs, calibrated following separated camera-LiDAR and LiDAR-LiDAR procedures. On the other hand, the performance of the method for a sensor configuration with a stereo camera, a wide-angle monocular camera, a high-resolution 16-layer LiDAR, and a 32-layer LiDAR is illustrated in Figure 3.17. This time, the pairwise calibration has been performed using the stereo rig with the 16-layer LiDAR, and the monocular camera with the 32-layer scanner. For all samples, a zoomed view of specific regions is provided so that the details can be well perceived.

As shown, the use of the extrinsic parameters extracted by the proposed approach enables a perfect registration between both data modalities. The quality of the sensors' alignment is especially noticeable when vertical thin objects are represented in the scene (i. e., lamp poles or trees), as can be seen in the close-up views. Moreover, it is remarkable how well the method performs even at far distances from the vehicle, considering that the calibration target is usually located within a limited distance to the sensors so that the pattern can be successfully segmented in all modalities. The outstanding accuracy at long ranges can be explained by the use of multiple poses, which further increases the precision obtained through a single-pose procedure, as depicted in Figure 3.15.

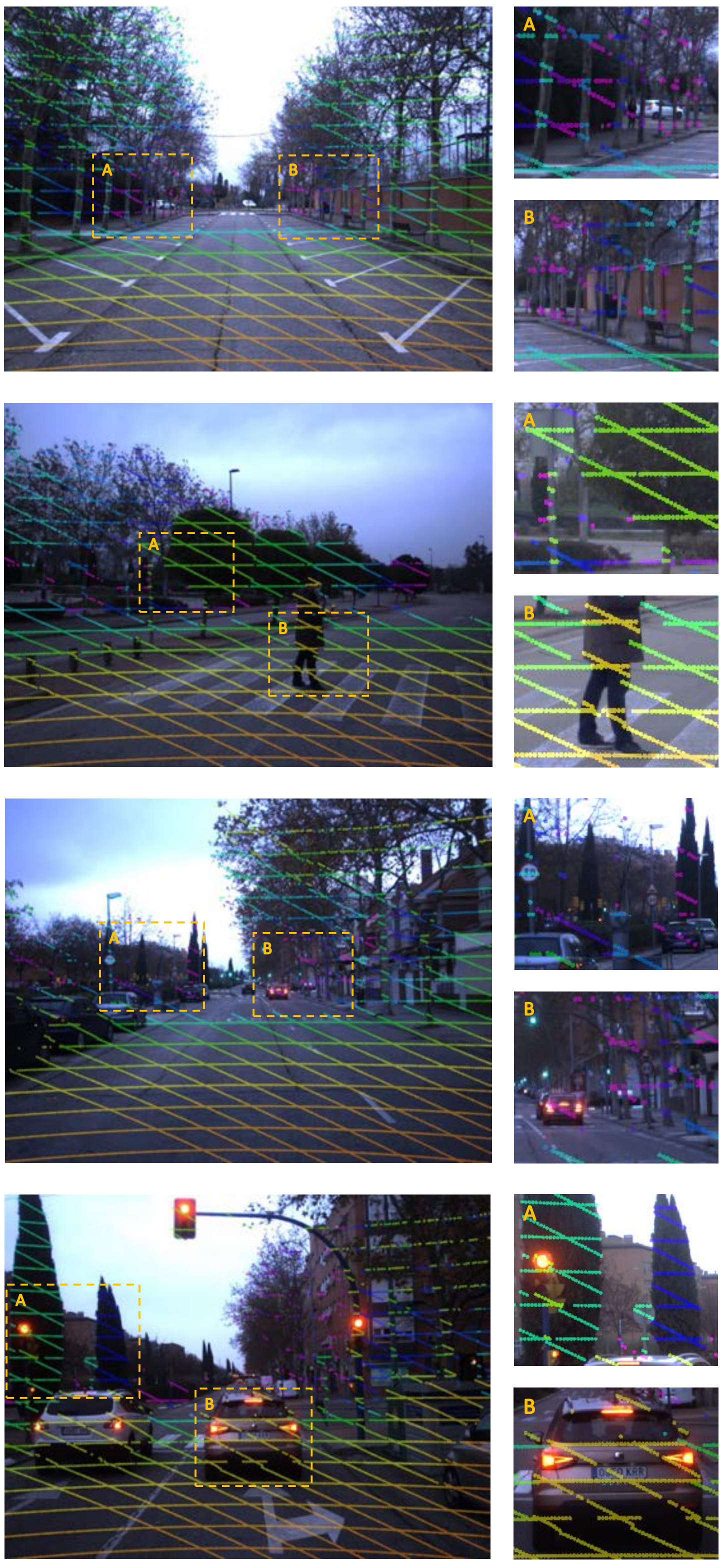
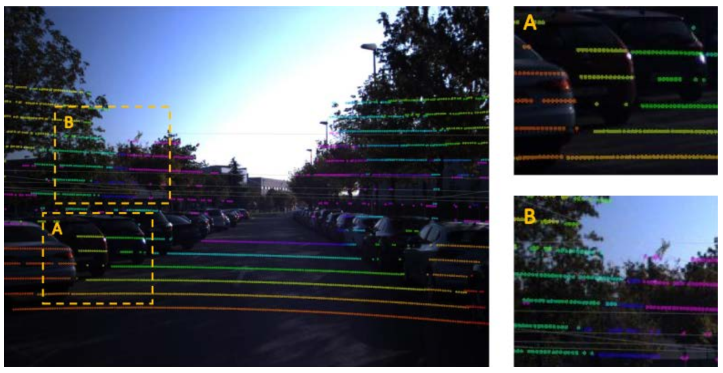
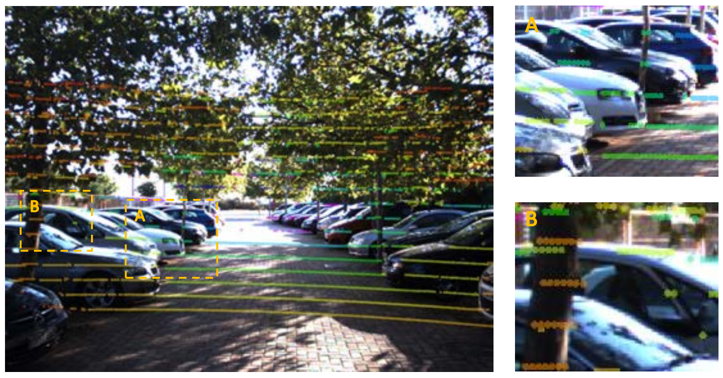


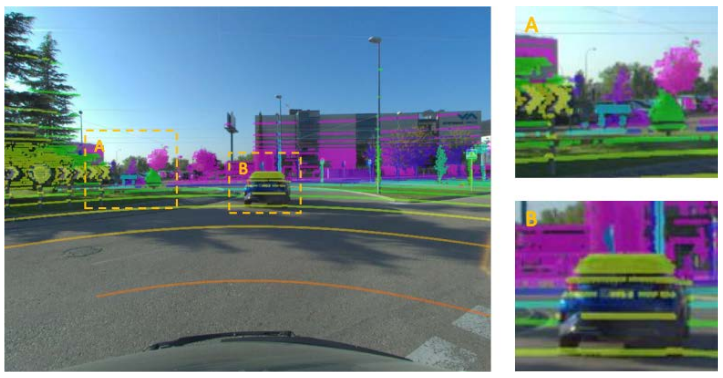
Figure 3.16: Calibration results in different traffic scenarios for a sensor setup composed of a narrow monocular camera and two 16-layer LiDARs. The LiDAR point clouds have been projected onto the image using the extrinsic parameters estimated through camera-LiDAR and LiDAR-LiDAR calibrations



(a)



(b)



(c)

Figure 3.17: Calibration results in different traffic scenarios for two sensor configurations: (a-b) a stereo camera with a Hi-Res 16-layer LiDAR; (c) a wide-angle camera with a 32-layer LiDAR. The LiDAR point clouds have been projected onto the image using the estimated extrinsic parameters.

3.2 CONCLUSION

In this chapter, we have presented an approach for the automatic estimation of the extrinsic parameters representing the relative pose of any pair of sensors involving LiDARs, monocular or stereo cameras, of the same or different modalities. Unlike the existing works, the simplicity of the calibration scenarios and the characteristics provided by the proposed target allow obtaining accurate results for most sensing setups featured by autonomous vehicles. Moreover, minimal user intervention is required during the process. Thanks to the design of the framework, the algorithm should be easily extendable to other sensors commonly found in vehicles (e. g., radars), as long as the fiducial target is endowed with elements that allow its unequivocal characterization in the 3D space using the new data representation.

Additionally, we have introduced an advanced simulation suite that copes with the traditional imprecision at performance assessment and provides exact ground truth that enables a reliable evaluation of extrinsic calibration methods.

Results obtained from the conducted experiments demonstrate that the algorithm presented in this chapter notably outperforms existing approaches. Tests on real data confirm the accuracy obtained in the simulation environment. The utilization of the proposed solution for the calibration of the multi-modal sensor configuration used for the perception system of an autonomous vehicle, described in Chapter 5, further validates its suitability for automotive applications.

Although there is still a road ahead, this proposal provides a practical approach to solve a common problem for the scientific community working in this field, bringing autonomous driving and robotics solutions closer to their final deployment. In this regard, the calibration software has been open-sourced[3], gathering remarkable attention among both academia and industry practitioners around the world.

3 As of December 2021, the repository of the project `https://github.com/beltransen/velo2cam_calibration` features 350 stars and 115 forks.

4 3D OBJECT DETECTION USING LIDAR'S BEV

The precise classification and localization of other traffic participants is a task of paramount importance to increase the level of automation of vehicles, as it provides the decision-making modules with the necessary information to perform the right maneuvers. Although historically, most efforts in the field of object detection were dedicated to the image space, the 2D characterization of elements in the surroundings is not sufficient to perform safe navigation.

Even though some approaches have made use of the pinhole model to infer depth from the signal captured by monocular camera devices, the results are not accurate enough for automotive applications. On the other hand, the degradation with the distance of the depth estimation computed through stereo pairs prevents its usage for medium and high-speed use cases.

To this effect, the incorporation of range sensors into the perception pipeline is key to finding the spatial positioning of the road users around. In this regard, LiDAR scanners have shown superior performance over radar devices, as they provide more precise and consistent measurements. Indeed, thanks to the increment of the resolution of modern multi-layer LiDARs in the last decade, they have become part of the sensor configuration of most autonomous prototypes under development. By using a single device of this kind, vehicles can capture dense geometrical information of the environment in 360° .

As a consequence, the detection and classification of objects in the scene may not only rely on image cues but can also benefit from laser features, traditionally relegated to object localization purposes. Although this brings new opportunities to foster new advances in the 3D object detection field, it also poses some challenges that need to be addressed.

On the one hand, detection frameworks that take LiDAR information as input need to handle the bulkiness and sparsity of the point clouds in real-time, which is a non-trivial task. On the other hand, new fusion strategies need to be developed to effectively combine features from complementary sensors like cameras and LiDARs, so that previous single-modality object detection approaches can be outperformed both in terms of accuracy and robustness.

In this chapter, the LiDAR's BEV projection is proposed as an effective trade-off between having a detailed representation of the scene geometry and offering a relatively simple data structure that can be

This chapter includes content from [12], [9] and [8] processed efficiently. The convenience of its use for on-board 3D detection is studied in a twofold approach. First, a two-stage object detection pipeline based on RGB-oriented architectures is presented. Experiments show that our network is able to provide state-of-the-art results while working at almost 10 hHz. Second, different fusion strategies are explored to exploit the joint use of image and BEV features in 3D object detection frameworks. To this end, a single-stage model is proposed so that real-time performance is guaranteed both when the network is used on its own or as a part of a more complex perception solution.

4.1 TWO-STAGE 3D OBJECT DETECTION

The application of machine learning techniques to the information gathered by high-resolution multi-layer LiDARs has led to promising results in the field of 3D object detection in traffic scenarios. These methods are able to endow vehicles with a reasonably accurate perception under a wide range of scene conditions, providing greater robustness than camera devices in some situations where their reliability is compromised, such as night driving. However, dealing with the sparse nature of laser scans requires the engineering of innovative approaches suitable for online operation.

In this regard, several research lines are being explored, either considering the point cloud as-is or through a simplified representation. Using the raw cloud enables to preserve all features and avoid any loss of information, although the uneven layout of the points in space forces to build new model architectures to encode meaningful descriptors from unordered data, which often are computationally expensive. To ease the design of inference networks and deal with sparsity, different alternatives have emerged. Voxel-based approaches discretize the LiDAR points into spatial cells, losing some information along the way. However, the resulting format is an ordered grid with a regular distribution of data much easier to process. Similarly, LiDAR projections dispense of some of the original information to produce 2D representations, which significantly enhance the efficiency during inference.

Here, we propose BirdNet, a framework for 3D detection and classification based on a novel LiDAR's BEV encoding containing distance-invariant features that enable the identification of multiple object categories using a single frame as input. Through the use of the BEV projection, deep learning models originally intended for image inputs can be applied, thus leveraging the detection architectures from a more mature field. The pipeline may be divided into input preparation, inference network, and 3D estimation, whose details can be found in the following sections.

4.1.1 *Bird eye's view generation*

The first stage of the pipeline is in charge of converting the LiDAR raw point cloud, where each point $P = \{x, y, z, i\}$ contains the Cartesian coordinates in meters and the intensity value from the reflected laser beam, into a bidimensional structure known as BEV. This grid map stores the LiDAR readings as seen from an orthographic top view, where each cell represents a square pillar lying in a theoretical ground plane. At each position, information of the LiDAR points falling inside the pillar is encoded. The main advantage of this format over the raw laser signal is that, as a 2D pseudo-image, the BEV projection can be fed to CNN detectors. Besides, it offers other interesting characteristics for automotive applications:

- The impact caused by the dimensionality reduction at the vertical axis is low, as it is presumable that the traffic participants which are likely to interact with the ego-vehicle lay on the same plane.
- Since all road actors move on the ground plane, there are no occlusions between objects in this view.
- The scale of the elements in the scene is preserved at all distances, so prior knowledge of average object dimensions can be used to ease the detection phase.

The BEV is a matrix of $M \times N$ square cells of size s^2, so that it extends for a total area of $sM \times sN$. Each cell can encode a variable number of features C, thus composing a final 3D tensor with dimensions of $M \times N \times C$. Regarding the encoding, we propose the use of three different channels, storing information about the height, intensity, and density of the cloud, whose details will explained later. As for the dimensions, they should be selected in accordance with the specifications of the LiDAR device in use and the application requirements, taking into account that the accuracy of the detection degrades as the sparsity of the cloud increases. Therefore, both the number and vertical distribution of the laser scans, the desired detection distance, and the time requirements need to be considered. In our experiments in the KITTI dataset, regions of $70 \times 35m$ and $45 \times 50m$ have been used. For the cell size, we empiricall yfound that a resolution of $s = 5cm$ allows for the detection of both large and small objects (i. e.cars, pedestrians, and cyclists). The use of a higher resolution barely improves the results while having a severe impact in the inference time, whereas greater cell sizes drop the performance for less bulky road users.

The three encoded features, which mimic the RBG channels of a color image, are shown in Figure 4.1. They describe the maximum height (H_{max}), mean intensity (I), and density (D) of the points falling inside the corresponding pillar of the grid. These features aim to provide an invariant representation of objects regardless of their distance to the LiDAR device. Even though the intensity and height readings are roughly homogeneous in the whole operating area of the range scanner, the number of points in a cell is directly related to its position in the space. Hence, a novel normalized density channel representing the number of laser readings divided by the maximum possible is proposed instead.

Before computing each of the image channels, upper and lower boundaries in the vertical axis are set to filter out points below the theoretical ground plane or above a maximum height H_{top}, since objects of interest are unlikely to be located in such regions. The minimum height is set to the $-Z_{lidar}$, being Z_{lidar} the height of the LiDAR sensor over the ground. The value of H_{top} corresponds to an offset of $3\,m$ above this plane. The magnitudes of all features are scaled to the 0–255 range.

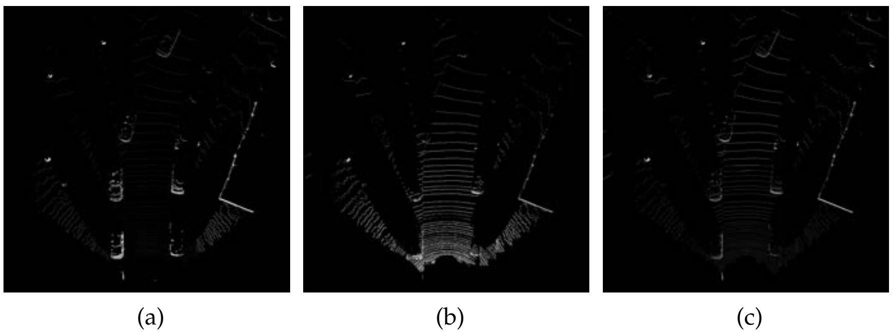


Figure 4.1: Baseline BEV feature encoding: (**a**) Maximum height. (**b**) Mean intensity. (**c**) Normalized density. Contrast corrected for visualization [8] © IEEE 2021

In order to build the density channel, the maximum number of points at each cell needs to be known. To this end, the properties of both the multi-layer LiDAR device and the pillars of the BEV are considered. From the sensor perspective, the number and vertical distribution (elevation angles) of the set of scan layers, L, and their horizontal angular (azimuth) resolution, $\Delta\theta$ determine the number of beams traversing through a region in the space. As for the grid cells, the maximum number of possible points is affected by their size and position. Hence, the normalized density feature is defined as:

$$D(x, y) = \frac{N_{points}(x, y)}{N_{max}(x, y, L, \Delta\theta)}, \tag{4.1}$$

where $D(x, y)$ is the density value at the position (x, y), $N_{points}(x, y)$ the number of points measured in that cell, and $N_{max}(x, y, L, \Delta\theta)$, the maximum number of points.

The function $N_{max}(x, y, L, \Delta\theta)$ can be obtained by geometric derivation. Let us limit the analysis to the 0–H_{top} range with respect to the theoretical ground plane considered for the BEV image's computation. The maximum number of points from a particular LiDAR layer that can be contained in a BEV $s \times s$ cell is given by the case in which a solid cuboid is placed at that location, spanning the whole volume represented by the cell ($s \times s \times H_{top}$). The problem then boils down to finding the number of points that would fall in that hypothetical cuboid.

The analysis will be performed on a per-layer basis so that the final value will be obtained as the sum of the contributions of each layer, $N_{max,l}$:

$$N_{max}(x, y, L, \Delta\theta) = \sum_{l \in L} N_{max,l}(x, y, l, \Delta\theta). \tag{4.2}$$

When taking into account the paths of the laser beams, each layer can be seen as a cone (without the base) whose vertex is in the LiDAR rotation axis. By definition, the planes limiting the represented volume (i.e., planes at $z = 0$ and $z = H_{top}$) are aligned with the LiDAR scanner. Therefore, the intersection of each of the cones with these planes is a circle.

A top view of the situation is shown in Figure 4.2, where the intersection of the LiDAR layers with the upper and lower limits are depicted in grey/black, and the cell for which N_{max} is being computed is outlined in blue as the $ABCD$ square. As shown, three different cases can be distinguished, depending on the relative position of the square representing the cell with respect to the circle generated by the intersection of the LiDAR layer with the vertical limits:

a) The square is completely outside the circle.

b) The circle cuts the square at two points, P_0 and P_n.

c) The square is completely inside the circle.

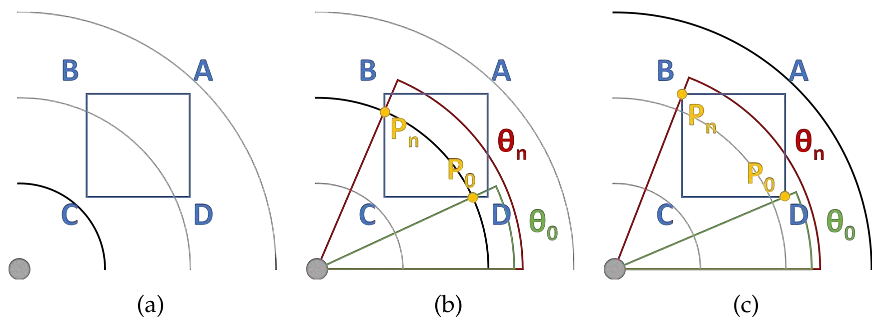


Figure 4.2: Horizontal cross-section of possible LiDAR-cell intersection scenarios: (a) No intersection. (b) LiDAR ring intersects with the upper/lower plane of the 3D pillar. (c) The cell is fully intersected by the laser plane [8] © IEEE 2021

In case a), no point from that layer can fall in the cell under consideration, so $N_{max,l} = 0$. In the other two cases, LiDAR beams from the analyzed layer would indeed collide with the $s \times s \times H_{top}$ cuboid. Points in the cell would be those in the interval between θ_0 and θ_n, each representing the azimuth angle of one of the border points (P_0 and P_n). Then, $N_{max,l}$ can be straightforwardly computed given the horizontal resolution of the LiDAR scanner, $\Delta\theta$:

$$N_{max,l} = \left\lceil \frac{|\theta_n - \theta_0|}{\Delta\theta} \right\rceil . \tag{4.3}$$

The calculation of θ_0 and θ_n from P_0 and P_n is trivial. Let $P_{i,x}$ and $P_{i,y}$ denote the x and y coordinates of point P_i, for $i \in \{0, n\}$; then

$$\theta_i = \arctan\left(\frac{P_{i,x}}{P_{i,y}}\right) . \tag{4.4}$$

Therefore, the focus is on the localization of points P_0 and P_n. In case c), P_0 and P_n coincide with the B and D vertices of the square representing the cell, as shown in Figure 4.2c. As the grid is arbitrarily created, these positions are known. However, in case b), P_0 and P_n are in intermediate positions in the BC and CD sides of the square. The computation of their positions involves solving the system of equations posed by the equations of the circle and the lines where the BC and CD segments lie. Let us denote V_y and V_y the x and y coordinates of one of the vertices of the square, V. Thus, point P_i (either $i = 0$ or $i = n$) must satisfy

$$\begin{cases} P_{i,x}^2 + P_{i,y}^2 = d^2 \\ P_{i,y} - C_y = \dfrac{V_y - C_y}{V_x - C_x}(P_{i,x} - C_x) \end{cases} \tag{4.5}$$

Here, d denotes the radius of the circle, and V is one of the vertices connected with C; that is, either $V = D$ (if $i = 0$) or $V = B$ (if $i = n$). Among the two solutions obtained, only one would be feasible at the current location.

This is also the way to determine the case corresponding to each cell (x, y) among the three discussed above: in a), the points of intersection will be closer to the origin than the square, and in c), they will be further away.

This approach deals with the differences in data density across the measurement range inherent to the LiDAR modality. Hence, the resulting BEV image fits better with the parameter sharing paradigm of convolutional layers, which assumes that features are invariant to location.

Despite calculating the cell-wise number of maximum points may be computationally expensive for large BEV dimensions, the process can be performed in an offline manner prior to the start of the operation, as the density normalization map is the same for all frames.

4.1.2 *Inference framework*

To perform object detection on the presented BEV images, we adopt the Faster R-CNN meta-architecture [145], as it offers excellent performance for objects of different dimensions. This two-stage approach consists of three main components: the CNN feature encoder or *backbone*, the Region Proposal Network (RPN), and the desired detection heads. The backbone processes the input through a set of convolutional operations and generates a feature map, which is then shared by the region proposal generation branch and the final estimation layers. Although the architecture was designed to process RGB inputs, it can be fed with arbitrary 2D structures such as LiDAR projections.

4.1.2.1 *Feature encoder*

For the feature encoder, any convolutional feature extractor intended for image detection may be selected depending on the requirements of the specific application. In our experiments, we have used both the VGG-16 [161], and Resnet-50 [73] architectures, as they offer an excellent trade-off between accuracy and computation speed. Regardless of the backbone of choice, the scale of the objects at a given feature map level needs to be considered so that small instances such as pedestrians and cyclists are not represented by too few pixels, which may hamper their detection. To this effect, the fourth pooling layer of the VGG-16 has been removed so that the last convolutional layer is 8 times smaller than the input tensor, instead of the original reduction of 16. In the ResNet case, its Feature Pyramid Network (FPN) version has been used, as it provides detections at downsampling resolutions of 4, 8, and 16, improving even further the performance for less bulky traffic participants.

4.1.2.2 *Proposals generation*

Once the features maps have been computed through the encoder layers, the RPN can take them as input to produce the set of proposals that will be forwarded to the heads of the model. For every element in the input tensor, this CNN tiny subnetwork is responsible for the estimation of an objectness score, which will be used to discard background cells where no objects of interest are located, and an array of residuals to refine the dimensions of a set of predefined axis-aligned candidate boxes k, known as *anchors*. Thus, for each position in the feature map, an output of $2k$ and $4k$ is obtained for anchors classification and regression, respectively. The sizes of the anchors are selected through a statistical analysis of the geometry of the projection of road users on the BEV. For efficiency reasons, only three scales, with box areas of 16^2, 48^2, and 80^2 pixels, and three ratios, of $1:1$, $1:2$, and $2:1$, are employed. After a Non-Maximum Suppression (NMS) procedure to remove highly overlapping proposals, the remaining boxes are sorted by their probability score of being an object, and the top 300 are selected. The refined anchor encoded by the residuals is then used to extract the object candidates from the backbone's feature map through a pooling process. Here, we adopt the *ROIAlign* method introduced in [72] to increase the localization accuracy of the ROI pool operation of the vanilla architecture.

4.1.2.3 *Detection heads*

The network estimation of the final detections includes the object category and the box parameters. Namely, the heads of the proposed model produce a box encoding described its center (x, y), its size (l, w), given by the dimensions of the minimum axis-aligned rectangle enclosing the object-oriented bounding box, and the heading angle θ, as shown in Figure 4.3a.

To infer the desired parameters, three sibling branches are employed. First, a classification head that replicates the configuration of the original Faster R-CNN architecture is used. In our case, the last FC layer generates an output of $N_cls + 1$ logits, being the first N_cls elements the probability of the candidate to belong to each of the object categories of interest, and the last one an additional score to allow the identification of background proposals classified as false positives at the RPN stage. A softmax operation is used for the normalization of the scores. Unlike other approaches in the literature, our framework performs simultaneous detection and classification of all relevant classes using a single model. Hence, we set $N_cls = 3$ to match the object types of the KITTI dataset, used for evaluation.

The second head is dedicated to the inference of the size and location of the bounding box. The regression targets of the four estimated magnitudes are defined as:

$$\begin{aligned} \Delta x &= \lambda_x \cdot \frac{x - x_p}{w_p} & \Delta w &= \lambda_w \cdot \frac{\ln(w)}{w_p} \\ \Delta y &= \lambda_y \cdot \frac{y - y_p}{l_p} & \Delta l &= \lambda_l \cdot \frac{\ln(l)}{l_p} \end{aligned} \tag{4.6}$$

being x,y,w,l the real position and dimension values, x_p,y_p,w_p,l_p the anchor parameters refined by the RPN, and $\lambda_i; i \in \{x, y, l, w\}$ the target-wise weights to scale the residuals so that they roughly have unit variance.

Finally, to fully describe the position of a detected object, its orientation has to be known. To do so, the Faster R-CNN has been endowed with a third branch which formulates the heading estimation as a classification problem, where the yaw angle is discretized into N_{bin} angle bins of equal amplitude, following the approach presented in [70]. The final orientation is obtained as a weighted average of the centers

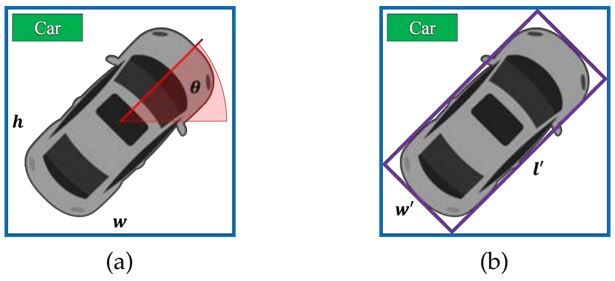


Figure 4.3: 2D detection refinement process. (a) Network output: class, axis-aligned detection, and yaw estimation; (b) 2D refined box [12] © IEEE 2018

of the predicted angle bin and its highest-scoring neighbor. The probabilities of the selected bins obtained after a softmax normalization of the output logits are used as weights. To ease the learning, bins are defined so that the most common orientations(i. e., forward/rear, and left/right) are represented unambiguously.

Both the size regression and the angle classification are performed on a per-category basis, as the refinement of the anchor candidates is done in the RPN.

4.1.2.4 *Multi-task training*

To train the proposed framework, a multi-task loss is used to optimize the model parameters for the different targets to be estimated: generation of proposals, classification, box size residuals, and objects orientation:

$$\mathcal{L} = \mathcal{L}_{p,cls} + \mathcal{L}_{p,bbox} + \mathcal{L}_{cls} + \mathcal{L}_{bbox} + \mathcal{L}_{\theta} \tag{4.7}$$

The first two components, $\mathcal{L}_{p,cls}$ and $\mathcal{L}_{p,bbox}$, represent the objectness classification and box regression of the proposals in the RPN, respectively. while the other terms account for the loss of the output of the three network heads.

All the elements composing the loss function follows the approach of the original Faster R-CNN detection framework. Therefore, the classification targets (i. e., $\mathcal{L}_{p,cls}$, $\mathcal{L}_{cls}$, and $\mathcal{L}_{\theta}$) are optimized via cross-entropy losses:

$$\mathcal{L}_{cls} = -\frac{1}{N}\sum_{i=1}^{N} y_i^* \cdot \log(\hat{y}_i), \tag{4.8}$$

where $\mathbf{y}^*$ stands for the one-hot ground-truth vector and $\hat{\mathbf{y}}$ for the network's estimation after softmax. Although they all share the same objective function, the different classification components differ from each other in the number of categories. A foreground-background binary classification is trained through $\mathcal{L}_{p,cls}$, while $\mathcal{L}_{cls}$ and $\mathcal{L}_{\theta}$ are multinomial losses with N_{cls} and N_{bin} possible labels, respectively. In all cases, the loss is normalized over the number of training samples used in each iteration at the corresponding stage, N.

On the other hand, residuals outputs make use of the mean absolute error loss ($\mathcal{L}_1$):

$$\mathcal{L}_{reg} = \frac{1}{N}\sum_{i=1}^{N} |\hat{y}_i - y_i^*|, \tag{4.9}$$

where y_i^* denotes the ground-truth value and $\hat{y}_i$ for the predicted residual. Again, the loss is normalized over the number of training samples.

In addition, two different strategies are applied to reduce the impact of the high class imbalance in the KITTI dataset. On the one hand, the final object categories are optimized as a weighted multinomial logistic loss where the underrepresented classes have a higher contribution to the final loss. On the other hand, training BEV samples are augmented by performing random horizontal flipping to increase the variability of the objects and enhance the model's generalization capabilities.

To accelerate the convergence of the objective function, network parameters are initialized with a model pre-trained on ImageNet [71]. Experiments suggest that this approach is still valid, although the learned weights were optimized to extract features on inputs of very different nature, i. e., RGB images. For the new layers, Xavier initialization is used [68].

The association between anchors and object labels is based on their Intersection over Union (IoU), following the approach in [145]. For the optimization of the proposals and final box parameters losses, only the error corresponding to the ground-truth class is considered. Besides, a NMS operation is carried out to remove redundant detections at inference time.

4.1.3 *3D box estimation*

Although the detections in the BEV are usually enough for safe navigation, the estimation of 3D boxes is also a topic of interest in the field of perception for automotive applications. Knowing the vertical position and height of the objects may be of help when trying to combine outputs coming from different frameworks in the same vehicle or from multiple agents in the context of cooperative detection.

Two different approaches have been studied. In the first version of the BirdNet framework, see Figure 4.4, a final stage was responsible for processing the model output to generate the oriented 3D boxes. In a second iteration, this step has been removed, and both the z coordinate and the object height h are estimated by the network together with the rest of the box parameters, as shown in Figure 4.5.

4.1.3.1 *Hand-crafted estimation of the 3D parameters*

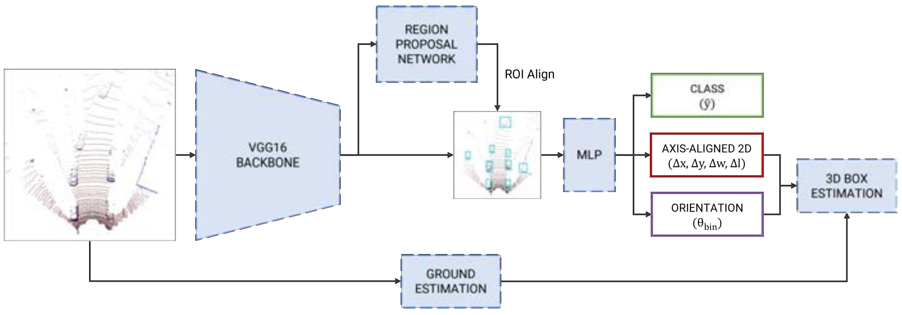


Figure 4.4: BirdNet object detection framework. The three outputs of the network are: class (green), 2D bounding box (red) and yaw angle (purple). For the final 3D box estimation, the axis-aligned detection, the orientation, and the ground estimation are used

Computing an oriented 3D box from the aforementioned predicted parameters entails a twofold task. On the one hand, the estimated axis-aligned detection and the yaw angle need to be processed to output the minimum oriented box enclosing the object in the BEV perspective, as shown in Figure 4.3. On the other hand, the box parameters in the third axis have to be calculated.

To obtain the object-aligned 2D boxes, prior information of the average physical dimensions of each of the categories is used. Concretely, a constant width w' is used for each object type, whose value is selected based on a statistical analysis of the annotated instances of the class in the dataset. In our experiments, w' is set to $1.8\,m$ for *cars* and $0.6\,m$ for the pedestrian and cyclist detections. Having the width fixed, the two length candidates are computed following (Equation 4.10) and (Equation 4.11), as the combination of w' and the predicted heading will unlikely provide a unique solution that perfectly fits the axis-aligned rectangle. Then, their corresponding oriented bounding boxes are obtained, and the choice of the final length l' is given by the box that maximizes the IoU with the non-rotated enclosing detection.

$$l_w = \left| \frac{h_{bbox} - \left|\cos(\theta + \frac{\pi}{2}) \cdot w_{fixed}\right|}{\cos\theta} \right| \tag{4.10}$$

$$l_h = \left| \frac{w_{bbox} - \left|\sin(\theta + \frac{\pi}{2}) \cdot w_{fixed}\right|}{\sin\theta} \right| \tag{4.11}$$

For the final 3D boxes to be fully encoded, the detections need to be endowed with information on their height and vertical position.

To this end, the raw point cloud is used to compute a coarse ground plane estimation, which is later used to get the bottommost z coordinate of the obstacles. A grid is created for the XY plane of the LiDAR point cloud, where each position stores the minimum height of the points falling inside. The size of the square cells should be big enough to guarantee that they cannot be fully covered by an object of interest so that laser beams can hit the ground. To remove the noise caused by outliers, a median blur is applied to the resulting grid. Afterward, the height of the object h' can be straightforwardly computed by subtracting the minimum height, taken from the ground grid cells below the detection, to the maximum height value inside the rotated BEV box found at the corresponding image channel. The vertical coordinate of the 3D object center $C = (x, y, z)$ will be then given by adding half the object height to the z of the ground plane.

4.1.3.2 *Learned estimation of the 3D parameters*

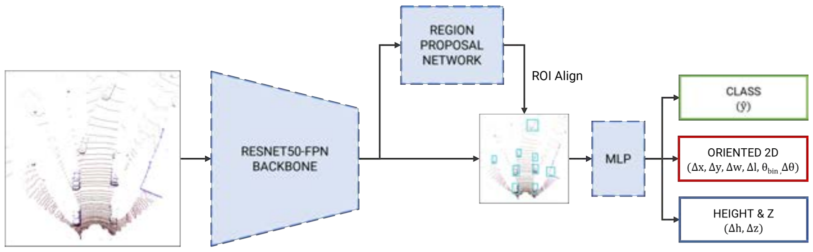

REGION PROPOSAL NETWORK
ROI Align
RESNET50-FPN BACKBONE
MLP
CLASS ($\hat{y}$)
ORIENTED 2D (Δx, Δy, Δw, Δl, θ_{bin}, $\Delta\theta$)
HEIGHT & Z (Δh, Δz)

Figure 4.5: BirdNet+ object detection framework. The three outputs of the end-to-end network are: class (green), oriented 2D bounding box (red) and object heigh and z position (blue)

As a logical evolution of the presented work, the framework has been extended to an end-to-end model capable of predicting 3D oriented boxes from the proposed BEV projection, known as BirdNet+. To this effect, two additional residuals are learned using the bounding box regression head. The parameters for encoding the vertical information of the boxes, i. e., Δz and Δh, follow the same fashion as the ones used for object detection in the XY plane.

Furthermore, a few other changes have been made. On the one hand, the estimated length and width of the object now correspond to the final desired dimensions, although the RPN still generate axis-aligned candidates. On the other hand, a hybrid approach has been adopted to improve the yaw angle prediction, so the estimation of the heading is encoded as an N_{bin} classification plus a residual estimation, which enables a greater precision than the method used in the preceding pipeline.

Naturally, the global loss has been updated to include the optimization of the new regression targets, which are carried out through $\mathcal{L}_1$ losses like in the first version of the network.

As will be shown in the results, the introduced changes enhance the object detection performance both in the 3D and the BEV domains.

4.1.4 *Experimental results*

A comprehensive set of experiments are carried out to assess the validity of the proposed framework using the well-known KITTI Object Detection Benchmark [61]. Thus, the object categories under evaluation are *Car*, *Pedestrian* and *Cyclist*.

In this section, we present a twofold analysis of the BirdNet pipeline. On the one hand, a series of ablation studies are reported to quantify the effect of changes in the input features, the network design, and the training initialization strategy over the performance of the approach. To this end, the training set of the KITTI dataset is used, since 3D annotations of objects in the scene are publicly available. The samples are divided into training and validation splits as in [30]. On the other hand, an evaluation of the overall performance of our solution is provided through a comparison with state-of-the-art LiDAR-based detectors in the official testing set.

The results detailed below follow the KITTI benchmark official metrics for the tasks of BEV and 3D object detection, where average precision (AP) is computed for each category of interest (c):

$$AP_c = \frac{1}{40} \sum_{r \in R} p_{\text{interp}}(r), \tag{4.12}$$

which accounts for the average value, over 40 recall values $r \in R$, of the interpolated precision p_{interp}, derived from the precision p as

$$p_{\text{interp}}(r) = \max_{\tilde{r} \geqslant r} p(\tilde{r}), \tag{4.13}$$

For a predicted detection to be counted as positive, a certain overlap with the corresponding object labels is required. The IoU thresholds used for both the BEV and 3D detection tasks are 70%, 50%, and 50%, for the three aforementioned categories, respectively.

To comply with the KITTI annotation policy, detections are computed for objects within the field of view of the camera. Consequently, the BEV images only contain information of LiDAR measurements in front of the vehicle. Besides, points falling outside a span of 110° centered in the forward direction are removed from the projection. For the ablation studies, we cover a region of 70×35m, whereas for the overall performance evaluation a 45×50m BEV is used. As for the grid resolution, cells of 0.05 m size are employed.

A specific hyperparameters configuration is set for each of the tested feature encoders:

- VGG: the model is trained for 150k iterations using a batch size of 1. The learning rate for the Stochastic Gradient Descent (SGD) optimizer is initialized at 10^{-3} and decays by a factor of 10 every 50k iterations.

- ResNet: the model is trained for 40k iterations with a batch size of 4. SGD is again used to optimize the objective function, here with a constant learning rate of 0.01.

For the overall evaluation of the networks in the KITTI testing set, where the whole training samples are used, the number of total iterations and the learning rate decay steps are doubled. In all cases, the number of proposals in the RPN is fixed to 300.

4.1.4.1 *Ablation studies*

Given that the BEV projection transforms LiDAR data into a pseudo-image structure which can be fed into popular 2D object detection architectures, it is interesting to investigate whether the proposed model can benefit from the common practice of initializing RGB-based CNN networks using weights of a pre-trained model. As usual, we take parameters optimized on the ImageNet dataset. Table 4.1 shows the results of BirdNet using the VGG-16 backbone . As can be seen, the pre-trained model provides a significantly higher performance despite the differences between features of both domains, indicating the convenience of adopting the fine-tuning approach over training the network from scratch.

Table 4.1: BEV and 3D detection performance (mAP %) using different weight initialization strategies, with $N_b = 8$ [12] © IEEE 2018

initial weights	mAP 3D (%)			mAP BEV (%)		
	Easy	Moder.	Hard	Easy	Moder.	Hard
ImageNet	22.92	18.02	16.92	54.46	41.61	40.57
gaussian	19.76	15.04	14.75	41.89	30.77	29.92

On the other hand, the effect of variations to some network's hyper-parameters is shown in Table 4.2. Different alternatives are studied:

1. preserving or removing the *pool4* operation of the backbone.
2. filtering or not the points belonging to the ground.
3. two possible resolutions for the orientation binning (N_{bin}).

Removing the fourth pooling layer in the VGG backbone reduces the downsampling, so the scale of the shared feature map used for both region proposal and the final estimation is doubled. As shown in the results, it is beneficial for the overall performance, making a remarkable difference in the detection of the smallest objects.

Table 4.2: BEV detection performance (AP %) on the validation set for different variants [12] © IEEE 2018

Class	*pool4*	ground	N_{bin}	BEV Detection (AP)		
				Easy	Mod.	Hard
Car	✗	✓	8	72.32	54.09	54.50
	✗	✓	16	**73.73**	**54.84**	**56.06**
	✓	✓	8	70.29	49.84	54.52
	✓	✗	8	66.63	48.52	47.98
	✗	✗	16	70.19	52.36	52.53
	✗	✗	8	69.80	52.56	48.44
Ped	✗	✓	8	43.62	**39.48**	**36.63**
	✗	✓	16	**44.21**	39.13	35.67
	✓	✓	8	25.01	23.23	21.84
	✓	✗	8	24.59	23.07	22.25
	✗	✗	16	41.73	37.17	34.81
	✗	✗	8	36.19	32.97	31.39
Cyc	✗	✓	8	47.44	31.26	30.57
	✗	✓	16	**50.45**	**33.07**	**31.15**
	✓	✓	8	41.87	27.49	25.79
	✓	✗	8	37.60	23.55	22.62
	✗	✗	16	41.59	26.94	26.21
	✗	✗	8	45.23	29.32	26.89

Regarding the ground plane, two configurations taking the raw and the filtered clouds as input have been investigated to understand whether the points belonging to the ground provide or not useful features for the detection task. For their removal, a height map algorithm was used, where the maximum difference in height between the points inside a cell is computed, and those falling on cells whose values lay below a certain threshold are discarded. As apparent, the disposal of these points hurts the performance of the method, which may be explained by the fact that although these cells do not contain information of the objects of interest, they give a meaningful context that can be leveraged by convolutions receptive fields.

Lastly, the experiments regarding the optimal slicing policy for the heading estimation throw that an output of $N_{bin} = 16$ achieves slightly better accuracy than using half the number of angle categories. The minor differences may be caused by the trade-off between the resolution and the number of samples for each orientation bin.

Henceforward, in view of these findings, all results referring to the vanilla BirdNet model correspond to a configuration without the *pool4* layer, 16 yaw categories, and fed with a BEV projection made from the whole LiDAR point cloud.

Table 4.3: BEV detection performance (AP %) on the validation set for different inputs [12] © IEEE 2018

Class	I	D	H	BEV Detection (AP)		
				Easy	Mod.	Hard
Car	✓	✗	✗	55.04	41.16	38.56
	✗	✓	✗	70.94	53.00	53.30
	✗	✗	✓	69.80	52.90	53.69
	✓	✓	✓	**72.32**	**54.09**	**54.50**
Ped	✓	✗	✗	36.25	30.43	28.37
	✗	✓	✗	38.21	32.72	29.58
	✗	✗	✓	38.37	34.04	32.37
	✓	✓	✓	**43.62**	**39.48**	**36.63**
Cyc	✓	✗	✗	33.09	22.83	21.79
	✗	✓	✗	43.77	28.62	26.99
	✗	✗	✓	**48.06**	31.21	30.40
	✓	✓	✓	47.44	**31.26**	**30.57**

Finally, the novel cell encoding introduced in Section 4.1.1 is analyzed. To this end, four different networks are compared: three of them being trained using images with only one of the proposed channels, and a fourth one using the whole three-layer BEV projection. As can be observed on Table 4.3, the intensity features provide the least meaningful cues for object detection, which might be expected due to many factors affecting reflectance measurements [88]. Besides, both the normalized density and the maximum height channels provide similar numbers, going well beyond those obtained using intensity information. Finally, the results produced when using the three-channel input image exhibit greater AP for all categories, proving their complementarity and, as a consequence, showing the positive effect of aggregating them.

4.1.4.2 *Performance evaluation*

A comparison of both versions of the proposed framework on the KITTI validation set is provided in Table 4.4. Hereon, an input BEV of 45×50m is used in the experiments since we found these dimensions are best suited to the annotated area in front of the vehicle. As can be seen, the numbers suggest that the modifications introduced in BirdNet+ have a significant impact on the final results, boosting the accuracy in both BEV and 3D detection tasks. The effect of the backbone change is made evident by the gap in the performance on the top view, as the prediction at multiple scales has led to an increase of about 20 points in all difficulty levels for the *Pedestrian* and *Pedestrian* classes. Furthermore, learning the box parameters of the vertical axis has proven a more convenient approach than its manual calculation. Thus, the degradation of the results from 2D to 3D is much lower in the end-to-end configuration, being of special interest the differences in the detection of vehicles.

Table 4.4: BEV and 3D detection performance (AP %) of BirdNet on the KITTI validation set

Class	Method	AP 3D (%)			AP BEV (%)		
		Easy	Mod.	Hard	Easy	Mod.	Hard
Car	BirdNet	50.27	37.07	36.94	87.69	63.57	63.37
	BirdNet+	81.26	68.71	66.74	91.90	82.99	82.58
Ped.	BirdNet	40.87	35.26	31.85	49.10	42.84	39.15
	BirdNet+	61.07	54.01	49.01	70.65	64.07	58.50
Cyc.	BirdNet	51.76	31.81	29.77	55.14	34.20	31.96
	BirdNet+	70.11	50.23	47.59	71.64	52.25	49.50

Even though the results from the vanilla BirdNet model look appreciably lower, the demanding requirements of the official KITTI metrics prevent the numbers from capturing the real potential of the method that will be qualitatively shown later on. This holds particularly true in the detection of the *Car* category, where the required 0.7 threshold for the overlapping between the predicted and the ground-truth boxes sets as false positives many of the detections. By studying the recall at different IoU thresholds (see Figure 4.6), we can observe that our method is able to locate more almost all vehicles in the *Easy* difficulty, and more than 70% of vehicles in the *Moderate* and *Hard*. Similarly, our method is capable of detecting pedestrians very efficiently at lower IoU. On the contrary, it has some problems at *Cyclist* detection. BirdNet+ data is also plotted to enrich the understanding of the differences between both versions.

Based on these findings, a comparison detection with the state-of-the-art LiDAR-based approaches in the literature at the date of publication of the BirdNet framework is shown in Table 4.5. In this case, an IoU threshold of 0.5 is applied, as it offers sufficient accuracy for *Car* detection in automotive applications. Results have been extracted

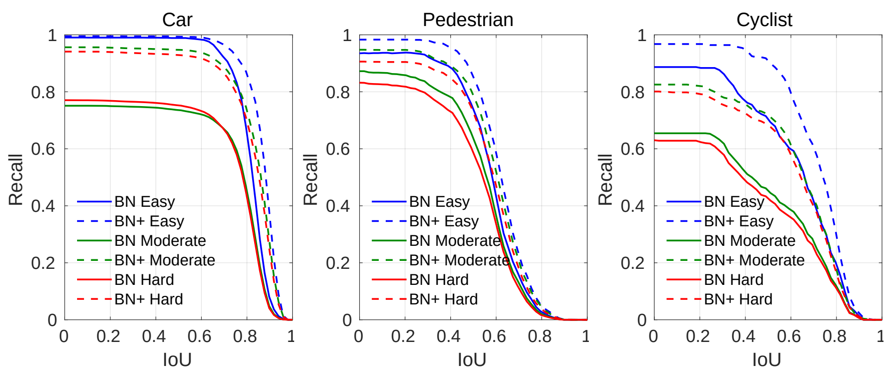


Figure 4.6: BEV recall at different IoU thresholds for both versions of the BirdNet framework

from the respective original works. Again, BirdNet+ numbers are included to provide information about the whole picture.

Table 4.5: BEV and 3D detection performance (AP %) for the *Car* category of different LiDAR-based approaches on the KITTI validation set with IoU 0.5 (%). LiDAR data can be used as BEV, Range View (RV), or as-is (RAW)

Method	Data	AP 3D (%)			AP BEV (%)			T
		Easy	Mod.	Hard	Easy	Mod.	Hard	(s)
MV3D	BEV+RV	95.74	88.57	88.13	86.18	77.32	76.33	0.24
VeloFCN	RV	67.92	57.57	52.56	79.68	63.82	62.80	1
F-PC_CNN*	RAW	87.16	87.38	79.40	90.36	88.46	84.75	0.5
BirdNet	BEV	95.52	70.36	72.21	95.65	72.71	74.76	0.11
BirdNet+	BEV	95.34	89.37	88.96	95.39	91.28	89.29	0.11

* Uses 2D image detections to filter the cloud.

As shown in the table, our pipeline outperforms VeloFCN by a large margin and provides comparable results to the other two methods, though being slightly lower for the *Moderate* and *Hard* difficulties. It is noteworthy to mention that our solution is the only one producing multi-class predictions, being the other models only designed for vehicle detection. Moreover, MV3D makes use of both the BEV and RV projections, and F-PC_CNN fuses LiDAR and RGB information. Besides, BirdNet is by far the framework with the fastest execution time, nearly operating at 10 hHz, which is often considered the minimum frequency for real-time perception applications.

Finally, the evaluation of the method on the official KITTI benchmark testing set is presented in Table 4.6. Results of comparable methods are also reported. Here, the two models have been trained with the samples of the full training set, used as train/val subsets in the previous experiments.

Table 4.6: BEV and 3D detection performance (AP %) of different approaches using LiDAR as input on the KITTI testing set. LiDAR data can be used as BEV, RV or voxelized (VX). In addition, some of the methods make use of RGB images (I) [8] © IEEE 2021

Cls	Method	Data	AP 3D (%)			AP BEV (%)			T
			Easy	Mod.	Hard	Easy	Mod.	Hard	(ms)
Car	MODet	BEV	-	-	-	90.80	87.56	82.69	50
	PIXOR++	BEV	-	-	-	93.28	86.01	80.11	35
	AVOD-FPN*	I+BEV	83.07	71.76	65.73	90.99	84.82	79.62	100
	MV3D (L)	BEV+RV	68.35	54.54	49.16	86.49	78.98	72.23	240
	C-YOLO	I+VX	55.93	47.34	42.60	77.24	68.96	64.95	60
	TopNet-Ret.	BEV	-	-	-	80.16	68.16	63.43	52
	BirdNet	BEV	40.99	27.26	25.32	84.17	59.83	57.35	110
	BirdNet+	BEV	76.15	64.04	59.79	87.43	81.85	75.36	115
Pedestrian	AVOD-FPN*	I+BEV	50.46	42.27	39.04	58.49	50.32	46.98	100
	C-YOLO	I+VX	17.60	13.96	12.70	21.42	18.26	17.06	60
	TopNet-Ret.	BEV	-	-	-	18.04	14.57	12.48	52
	BirdNet	BEV	22.04	17.08	15.82	28.20	23.06	21.65	110
	BirdNet+	BEV	41.55	35.06	32.93	48.9	42.87	40.59	115
Cyclist	AVOD-FPN*	I+BEV	63.76	50.55	44.93	69.39	57.12	51.09	100
	TopNet-Ret.	BEV	-	-	-	47.48	36.83	33.58	52
	C-YOLO	I+VX	24.27	18.53	17.31	32.00	25.43	22.88	60
	BirdNet	BEV	43.98	30.25	27.21	58.64	41.56	36.94	110
	BirdNet+	BEV	65.67	53.84	49.06	70.84	59.58	54.2	115

* AVOD makes use of two separate models: one for *Car* and another for *Pedestrian* and *Cyclist* detection.

As can be seen, BirdNet+ outperforms other LiDAR-only approaches in the literature and yields a detection performance close to top-performing fusion pipelines, all at a framerate around 10 FPS. Regarding BirdNet original framework, its detection accuracy for vehicles falls some points behind the one achieved by other *Car*-only BEV methods. On the contrary, the results on the other categories are comparable to the rest of multi-class approaches, regardless of whether they are fed solely with LiDAR inputs or also with RGB data.

Figure 4.7 depicts the results of both BirdNet and BirdNet+ models on several sample frames from the KITTI testing set. As can be observed, the end-to-end framework improves the results of its predecessor in the detection of small objects, reducing the number of false positives and providing more adjusted boxes. Regarding vehicle identification, both networks perform similarly in the short and medium range, while BN+ is able to provide detections at larger distances. Moreover, the height and vertical position are generally better in the second generation of the pipeline.

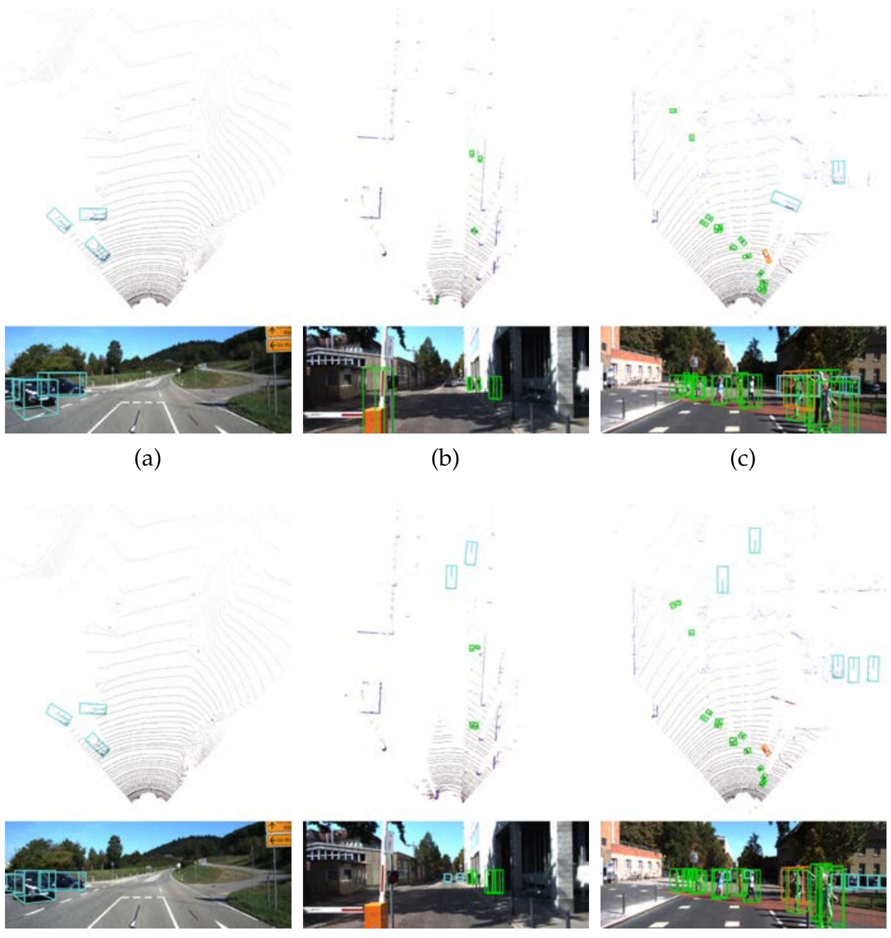


Figure 4.7: Results on the KITTI testing set. BirdNet detections are shown on the top row, while BirdNet+ results are in the bottom row

4.2 SINGLE-STAGE 3D OBJECT DETECTION

To further investigate the suitability of the LiDAR's BEV for 3D object detection purposes, its applicability to fusion paradigms has been studied. In this case, a single-stage approach is proposed so that it can comply with the tight computational and time requirements of on-board processing, regardless of the selected fusion strategy. The baseline architecture, which uses only LiDAR data as input, works at a frame rate of around 50 Hz while providing a superior accuracy than the one of the vanilla BirdNet framework.

In the following sections, the different image-LiDAR fusion schemes that have been evaluated are introduced, and the details of each model configuration are provided. Finally, the experimental results on the KITTI benchmark are shown to provide a fair comparison among the presented solutions.

4.2.1 *Fusion arquitectures*

Combining the information captured by complementary sensors such as LiDARs and cameras may be beneficial for dealing with complex tasks such as 3D object detection in traffic scenarios, where a myriad of objects of uncountable geometries and appearances can interfere with the trajectory of the vehicle. However, although establishing correspondences between their data is roughly trivial if the extrinsic transformation between devices is available, extracting knowledge from their joint use is yet a matter of research.

The kind of input formats and strategies used when fusing image and LiDAR information determines how well the framework is able to exploit features from both modalities. Regarding the format, we propose the use of the LiDAR scans in their BEV projection, as it has proven suitable for accurate 3D object detection. As for the fusion schemes, several alternatives have been explored:

EARLY FUSION. Raw features from the images and the raw LiDAR point clouds are combined before the network is fed. To do so, the laser beams are projected onto the image plane, and the pixel values are appended as new channels to each point in the cloud. Thus, the points of the resulting cloud store information about their 3D position, their reflectivity, and their corresponding image features. Two different setups have been tested, using the RGB color and grayscale, respectively.

SEQUENTIAL FUSION. The second type of strategy aims to take advantage of two steps, where the outputs of image-based detectors are used to enhance the input of the LiDAR model. This line of work draws from the early multi-modal methods, which applied a *detect-and-locate* approach using images and low-resolution laser data, respectively. In our experiments, we employ the output of a 2D instance detector to modify the encoded features of the projected LiDAR points falling inside detected objects. Namely, a Mask R-CNN model has been used [72]. Several variations have been considered:

- Using the vanilla BEV encoding plus the addition of separate per-category channels, either using a binary mask or object-wise classification score.
- Filtering the cells not belonging to objects detected in the image space, leaving just the original LiDAR encoding or including the mentioned additional features.

Feature-level Fusion. This strategy consists of processing the data from both modalities in separate backbones within the network and fusing their information at one or several feature map scales before the final estimation heads are fed. Although this scheme has been studied by some RPN approaches in the literature, no single-stage architectures are known. Here, we propose a geometrical coherent fusion scheme, where information from both streams is fused through efficient indexing based on 3D-2D correspondences. Two different merging operations are evaluated: addition and concatenation.

Despite the rapid evolution of monocular depth estimation networks, the 3D object detection accuracy of methods that solely rely on RGB information as input is not yet comparable to their LiDAR counterparts. For this reason, *late fusion* schemes, where the outputs of two separate detectors are combined, are excluded from the comparison.

Some samples of the aforementioned new input channels are shown in Figure 4.8. As can be seen in subfigures 4.8c and 4.8b, certain cells behind the objects also contain image-based features. Most of these artifacts are generated when establishing correspondences between the two modalities, as some LiDAR beams that fall in regions occluded from the camera perspective are projected within the contours of the detected instances. In addition, any inaccuracies in the output of the image detector may also cause this effect. This misprojection affects as well the channels storing color information.

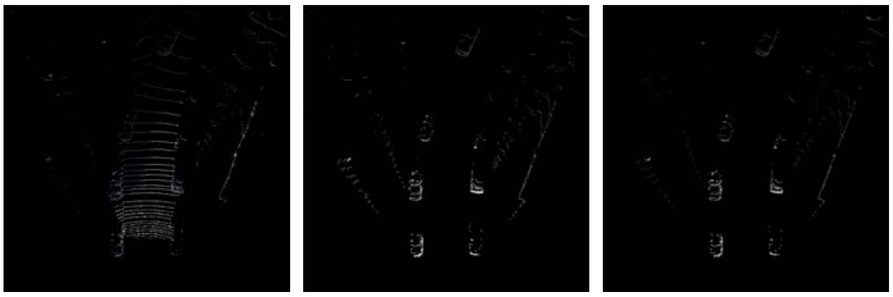

(a) (b) (c)

Figure 4.8: Three sample channels that incorporates image information to BEV: (a) RGB. (b) Object classification scores scaled to 0-255. (c) Maximum height of cells belonging to objects detected in 2D. Contrast corrected for visualization

4.2.2 *Inference framework*

In order to have a baseline network capable of operating in real-time regardless of the fusion scheme followed to incorporate information from images, a single-stage convolutional detector is proposed. These kinds of architectures dispose of the auxiliary proposal generation branch to reduce the computational load, as the time required to perform the pooling operation of RPNs and the subsequent per-candidate prediction often hinder the applicability of two-stage pipelines to onboard perception tasks. Although BirdNet has shown outstanding results working at around 10 Hz, its use for online 360° detection will entail having a coarser cell resolution, leading to a performance drop.

For this reason, the proposed architecture directly estimates the class and 3D boxes of objects from the feature map extracted by the backbone through the use of three detection heads. Figure 4.9 shows an overview of the baseline model. As can be seen, several ResNet blocks are used as encoder. The final configuration is composed of the common *stem* layer (i. e., a 7x7 convolution and a max pooling) followed by two groups of 3 and 4 normal convolutions, respectively. A comparison with alternative backbones is provided in Section 4.2.3.1.

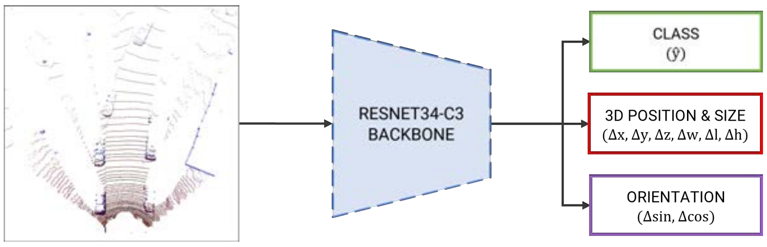

RESNET34-C3
BACKBONE
CLASS
(φ)
3D POSITION & SIZE
($\Delta x, \Delta y, \Delta z, \Delta w, \Delta l, \Delta h$)
ORIENTATION
($\Delta \sin, \Delta \cos$)

Figure 4.9: Proposed single-stage 3D detection framework. The three outputs of the network are: class (green), 3D bounding box (red) and yaw angle (purple)

4.2.2.1 *Feature-level fusion*

Most existing approaches that combine LiDAR and image information within the layers of the network are based on two-stage architectures. The main reason for this kind of design is that it allows to perform the fusion operation at the object level [30, 95]. Separate streams process the different modalities through the backbone layers and, at some point, the projections of the 3D proposals in each tensor are used to extract and fuse the anchor's features from each data source. Although these frameworks provide acceptable improvements over their LiDAR-only counterparts, the use of the RPN branch compromises their throughput speed. Regarding single-stage approaches, attempts have been made to teach the network how to transfer the features from the image space to the BEV [107]. Despite its promising results for vehicles, this method fails to generalize well for less represented classes, as it requires a huge amount of training samples.

In order to evaluate the feature-level fusion scheme in our framework for multi-class detection, we present a new approach to merging the information from two modalities by exploiting geometrical correspondences between the modalities without the need for region pooling operations. The operation lies on the nature of single-stage detectors, which leverage the receptive field of consecutive CNN layers to build rich pixel-wise features, so simple prediction heads can infer the final outputs at each position. Following this principle, we propose to perform the fusion through an efficient indexing layer that extracts the feature vectors of each anchor candidate by projecting the coordinates of each 3D box center onto the corresponding encoded map. Once both tensors with the anchor features are obtained, their information can be merged. A diagram of the proposed scheme is shown in Figure 4.10. In our experiments, two fusion operations are explored: addition and concatenation. To avoid the slow anchor-wise processing of R-CNN-like methods, the indexing operation arranges the candidates' information in the top-view representation, preserving the BEV layout and enabling further spatial reasoning.

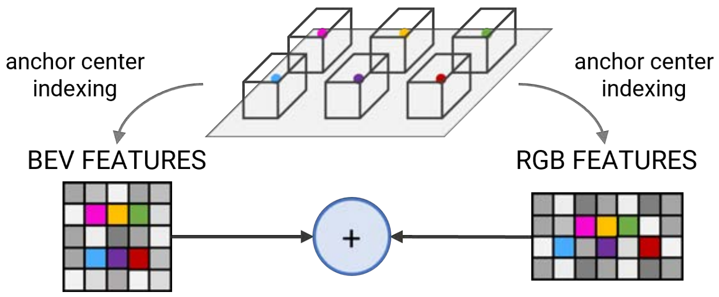


Figure 4.10: Proposed fusion scheme: the centers of the 3D anchors are projected onto the feature maps encoded from BEV and image inputs, so the anchor-wise features can be fused

4.2.2.2 *Detection heads*

The proposed framework provides 3D detections encoded by nine parameters: one for the category, three for the 3D position of the object center (x, y, and z), another three for the dimensions of the enclosing cuboid (length l, width w, and height h), and two additional ones for its orientation on the road plane, or yaw angle ($\sin(\theta)$ and $\cos(\theta)$). Detections are given in the LiDAR coordinate system, and both the location and dimensions units are expressed in meters (m) within the network.

Three sibling branches are employed for the prediction of the objects' parameters: a classification head to determine the object category, and two regression heads to estimate the box position and size, and the heading, respectively. All branches are composed of four convolutional layers of kernel size 3 that are fed with the shared tensor from the output of the last layer in the feature encoder.

Regarding the 3D box characterization, the position (x, y, z) and dimensions (w, l, h) of the objects are encoded as residuals with respect to the reference proposals. Hence, the targets are defined as:

$$\begin{aligned} \Delta x &= \lambda_x \cdot \frac{x - x_p}{w_p} & \Delta w &= \lambda_w \cdot \frac{\ln(w)}{w_p} \\ \Delta y &= \lambda_y \cdot \frac{y - y_p}{l_p} & \Delta l &= \lambda_l \cdot \frac{\ln(l)}{l_p} \\ \Delta z &= \lambda_z \cdot \frac{z - z_p}{h_p} & \Delta h &= \lambda_h \cdot \frac{\ln(h)}{h_p} \end{aligned} \tag{4.14}$$

where λ_i is used to scale the residuals to approximately have unit variance, and i_p is the dimension of the box parameter i of the anchor box.

Since the nature of angles artificially amplifies the errors between similar orientations (e. g., π and $-\pi$), a residual approach is followed to estimate the heading of the objects, instead of inferring the absolute magnitude. Concretely, the angle prediction branch generate two regression targets, encoded as:

$$\begin{aligned} \Delta\sin &= \sin(\theta - \theta_p) \\ \Delta\cos &= \cos(\theta - \theta_p) \end{aligned} \tag{4.15}$$

where θ_p stands for the orientation of the anchor, which in our experiments is set to 0.

Using these two values, the yaw angle (θ) can be unambiguously computed:

$$\theta = \theta_p + \arctan\left(\frac{\Delta\sin}{\Delta\cos}\right) \tag{4.16}$$

4.2.2.3 *Multi-task learning*

To optimize the proposed network for the classification of objects, the estimation of the 3D box size and location, and the regression of the yaw angle, a weighted multi-objective loss function made of three components is used:

$$\mathcal{L} = \omega_{cls} \cdot \mathcal{L}_{cls} + \omega_{box} \cdot \mathcal{L}_{box} + \omega_{\theta} \cdot \mathcal{L}_{\theta} \tag{4.17}$$

In our experiments, we found that $\omega_{cls} = 1$, $\omega_{box} = 1$, and $\omega_{\theta} = 4$ yield optimal results.

Due to the particularities of single-stage approaches, where there exists a huge imbalance between anchors corresponding to objects and background, the prediction of categories cannot be trained through a conventional cross-entropy loss. Instead, we make use of a focal loss [110], which helps focus on the examples that are hardest to learn (i. e., those belonging to objects) and reduces the contribution of predominant classes (i. e., background):

$$\mathcal{L}_{cls} = -\alpha(1 - p_t)^{\gamma} \log(p_t) \tag{4.18}$$

where p_t is defined as:

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise} \end{cases} \tag{4.19}$$

being $p \in [0, 1]$ the estimated probability for the class y, and y equal 0 for all categories except for the ground-truth label, in a one-hot encoding fashion.

To cope with the significant differences among the amount of samples of each category, the weighting parameter has been experimentally set as $\alpha \in \{0.75, 0.99, 0.99, 0.1\}$ for *Car*, *Pedestrian*, *Cyclist* and *Background*, respectively.

Regarding the residuals optimization, the *smooth* $\mathcal{L}_1$ loss have been used:

$$\mathcal{L}_{reg} = \begin{cases} 0.5(\hat{y} - y)^2 & \text{if } |\hat{y} - y| < 1 \\ |\hat{y} - y| - 0.5 & \text{otherwise} \end{cases} \tag{4.20}$$

where y is the ground-truth value, and $\hat{y}$ the output of the network.

For the computation of the regression losses, only the errors of the ground-truth labels are considered. Besides, a per-category weight α_{reg} is applied to the residual targets so that the impact of the class imbalance is mitigated.

In this respect, the matching between anchor boxes and detections aims to equalize the number of positive proposals of each category. To this end, the dimensions of the most frequent objects (i. e., cars) are shrunk before the anchors whose center falls inside the ground-truth box in the BEV projection are marked as positive. In parallel, those having the center inside the object boundaries but outside the positive downscaled region are ignored during training.

4.2.3 *Experimental results*

Following the evaluation approach introduced in Section 4.1.4, the KITTI benchmark has been used to assess the performance of the proposed single-stage network. Hence, the accuracy in the detection of *Car*, *Pedestrian*, and *Cyclist* objects that are visible in the vehicle's front camera is measured using the Average Precision (AP) metric.

Once again, a set of ablation studies using the training-validation splits from [30] were carried out to tune the final model configuration. In this case, these experiments address both the baseline architecture and the different fusion schemes. Finally, a comparison to other recent works on the official testing set is provided.

Guided by the findings from the BirdNet framework, all results make use of a 45×50m BEV with a cell resolution of $0.05\,\text{m}^2$. Besides, the model has been trained for 80 epochs with a batch size of

4. For optimizing the losses, SGD has been used with a weight decay of 0.0005 and a momentum of 0.9. An initial learning of 0.0004 is set, which decays by a factor of 10 at epochs 50 and 75. Horizontal flipping is randomly applied to increase the number of training samples.

4.2.3.1 *Ablation studies*

Mimicking the feature extractor in BirdNet+, a ResNet encoder is used. In order to identify the configuration that maximizes the model performance, a set of backbone settings have been tested, with a different number of layers and depths. Table 4.7 shows the average results of all categories in both the BEV and 3D spaces.

Table 4.7: BEV and 3D detection performance (AP %) of different backbones

Backbone	mAP 3D (%)			mAP BEV (%)		
	Easy	Moder.	Hard	Easy	Moder.	Hard
Resnet18-C3	46.47	35.43	32.12	59.07	47.71	44.52
Resnet18-C4	27.74	21.92	19.80	39.45	33.42	31.02
Resnet34-C3	**50.13**	**38.88**	**35.29**	**62.08**	**51.31**	**47.70**
Resnet34-C4	28.55	22.52	20.49	40.73	34.73	32.22
Resnet50-C3	45.78	35.63	32.71	59.66	48.35	45.20
Resnet50-C4	30.68	25.04	23.04	45.34	38.15	35.65

On the one hand, the use of features after the third ResNet block yields better results than *C4*, as the downsampling in the latter layer hurts de detection of *Pedestrian* and *Cyclist*. On the other hand, the model using Resnet34 offers better performance at this scale. The gap with the other two encoders is due to its greater size at *C4* level. In the case of ResNet18 configuration, the fewer number of layers may be insufficient to learn the desired targets with precision. As for ResNet50, the *bottleneck* module reduces the parameters per block, which hurts its performance when pruned at early stages. Henceforward, the reported results make use of features from *C3* from a ResNet34 encoder.

Before studying the distinct multi-modal fusion schemes, the performance of the proposed LiDAR-based single-stage detector (SS) is compared to the two versions of the BirdNet framework (BN and BN+) in Table 4.8.

As can be seen, the non-RPN model is able to outperform the original BirdNet pipeline in both *Car* and *Pedestrian* detection. In the *Cyclist* category, *BN* offers better accuracy, as the focal loss is unable to completely cope with the high class imbalance. As for BirdNet+, its numbers are still better in BEV and 3D, being especially notable the gap in the detection of small objects. This difference agrees with the

Table 4.8: BEV and 3D detection performance (AP %) of the different LiDAR-based proposals on the KITTI validation set

Class	Method	AP 3D (%)			AP BEV (%)			FPS
		Easy	Mod.	Hard	Easy	Mod.	Hard	
Car	BN	50.27	37.07	36.94	87.69	63.57	63.37	9.1
	BN+	81.26	68.71	66.74	91.90	82.99	82.58	8.7
	SS	62.78	50.74	45.53	85.04	75.75	71.34	47.6
Ped.	BN	40.87	35.26	31.85	49.10	42.84	39.15	9.1
	BN+	61.07	54.01	49.01	70.65	64.07	58.50	8.7
	SS	43.65	38.25	34.53	52.81	47.10	43.14	47.6
Cyc.	BN	51.76	31.81	29.77	55.14	34.20	31.96	9.1
	BN+	70.11	50.23	47.59	71.64	52.25	49.50	8.7
	SS	43.96	27.64	25.82	48.39	31.09	28.61	47.6

results found in the literature in other fields such as 2D object detection in images [87]. On the contrary, the recall at different IoU thresholds shown in Figure 4.11 demonstrates that the strict overlapping required for the detection of vehicles dilutes the real performance on these kinds of objects.

Regarding the throughput of all models, the single-stage architecture operates at a remarkable 47.6 Hz, providing a 5× greater frame rate than their two-stage counterparts.

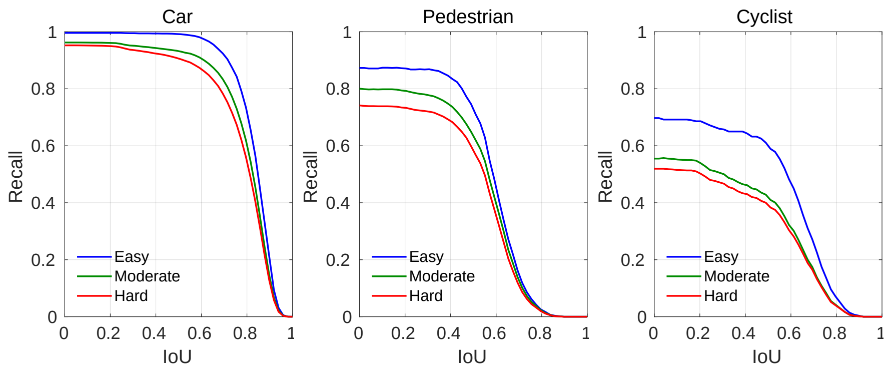


Figure 4.11: BEV recall at different IoU thresholds for both versions of the BirdNet framework

Once the baseline model has proven valid for the 3D object detection using BEV tensors as input, the suitability of different image-LiDAR fusion strategies can be evaluated. Namely, nine configurations have been explored:

- Two *early fusion* schemes using the original cell encoding plus additional gray (E1) or RGB (E2) channels.
- Two *sequential fusion* schemes using the original cell encoding plus class-wise binary (S1) or classification score (S2) features obtained through correspondences with the outcome of the 2D object detector.
- Three *sequential fusion* schemes using the original (S3) and the two encodings described above (S4 and S5, respectively), but where features of cells not belonging to objects are set to zero.
- Two *feature-level fusion* schemes using the concatenation (F1) and the addition (F2) operations to combine features from both sensor modalities.

Their results are depicted in Table 4.9, where the numbers for the LiDAR-only baseline (B) are included as a reference.

As apparent from the table, the different configurations have unlike impact on the vanilla model. In the early fusion approaches, although the use of image information may help in the detection of four or two-wheeled road users, it causes a slight downfall in the overall performance. This might be explained by the fact that raw color information, either in its gray or RGB representation, cannot be properly exploited in a top-view representation, as the aggregation of cues from all points in a cell blurs salient features and reduces the differences between the objects and the background. Moreover, valuable geometrical information provided by images cannot be leveraged.

Regarding the sequential fusion approaches, two different outcomes are observed. On the one hand, when the image data is used to complement the original BEV encoding, it yields a notable improvement over the baseline in all categories and difficulties, with a higher effect in the top-view metrics. These new channels endow the input tensor with valuable information about regions of interest where the probability of having an object is high. Moreover, the predicted category is also encoded, easing the learning of the classification head. It is noteworthy that the configuration using the confidence score performs better than the binary maps, suggesting this way that the model is able to correct some false positives coming from the RGB modality. On the other hand, when the 2D detections are used to narrow down the search space by filtering out background points (S3, S4, and S5), the 3D accuracy of the model dramatically drops. As only cells belonging to objects are preserved, important information around the objects is lost, leaving the detector without the necessary context to determine the vertical coordinate of the box center. Among the different configurations, their mAP is consistent with their non-filtering counterparts.

Finally, the feature-level schemes drop the overall results by a few points, especially when doing the fusion through concatenation. Although the addition configuration shows some potential for Cyclist detection, its precision on the other categories falls behind the baseline. Though this behavior is coincident with those experienced by other works in the literature with respect to the *Car* class [30, 95], it is not the expected outcome for the small objects. This might be explained by the inability of the indexing operation used to establish correspondences among the feature maps of the two modalities to mimic the ROI pooling of variable-size anchors found in RPN. The use of features from multiple scales may address this limitation.

4.2.3.2 *Performance assessment*

After the baseline architecture and the best fusion strategy have been identified through the ablation studies, the overall performance of the single-stage detector can be assessed in the KITTI testing set. Table 4.10 shows a comparison with other existing approaches. As usual, both models have been trained using all samples from the official training split.

As can be seen, the baseline model offers competitive performance, being superior to other multi-class approaches solely using BEV inputs (i. e., BirdNet and TopNet). On the contrary, car-only LiDAR detectors benefit from their specialization, providing a better accuracy in this category.

Regarding the sequential fusion approach, it confirms the trend shown in the validation split, boosting the baseline's precision in the detection of small objects. Besides, its results are comparable to MV3D and F-PC_CNN, while operating at a significantly superior rate. As for Complex-YOLO, our solution presents much higher numbers. On the other hand, AVOD, ContFuse, and F-PointNets fusion schemes are able to exploit image features in a very effective way, outperforming our proposal by a large margin. However, the comparison with the first two methods is not totally fair, as AVOD makes use of separate models for vehicles and small objects, while ContFuse disregards *Pedestrian* and *Cyclist* categories. With respect to F-PointNets pipeline, the use of raw cloud data in the box inference stage allows filtering background information since points in frustum do contain all the necessary information. As a result, the search space is significantly narrowed down, and the possibility of getting false positive detections is diminished.

The outcome of both the proposed LiDAR baseline and the best fusion configuration on some scenes of the KITTI test set is shown in Figure 4.12. Regarding vehicles, both frameworks provide excellent performance as observed in the BEV, where all objects inside the field of view are properly located. Despite the inherent difficulties of single-stage models with small objects, it can be observed that most

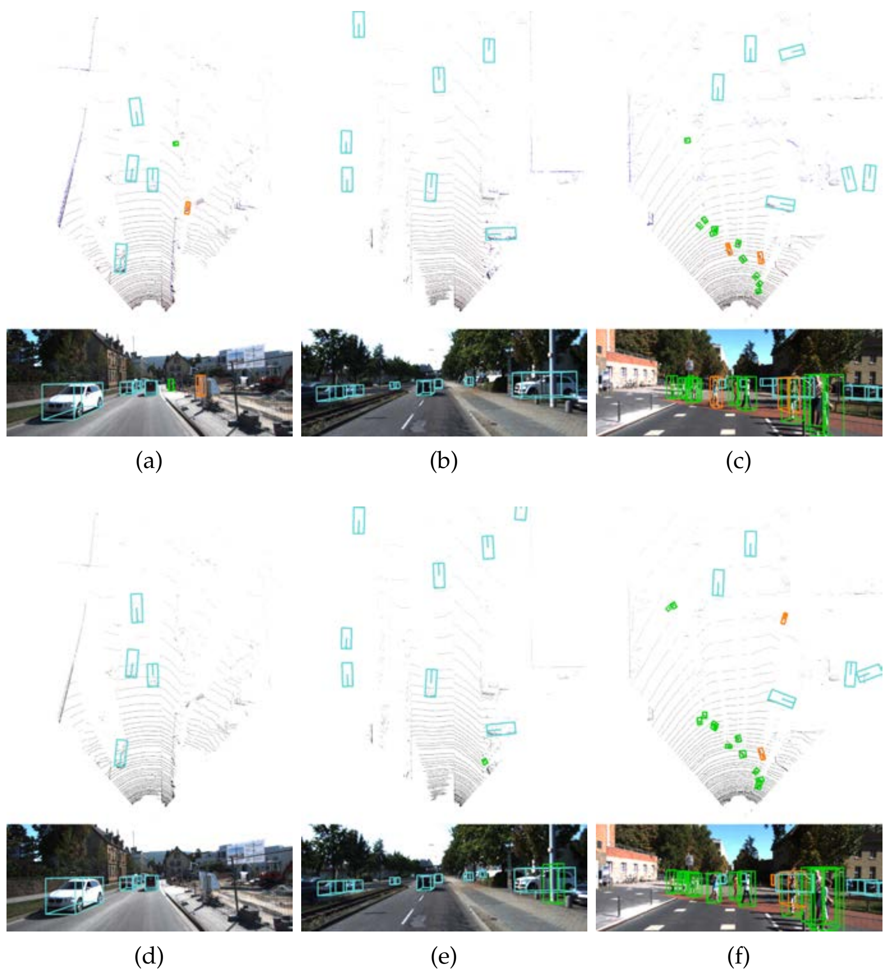


Figure 4.12: Results on the KITTI testing set. Detections of the LiDAR-based model are shown on the top row, while those produced by the selected fusion scheme are displayed on the bottom row.

Pedestrian and *Cyclist* instances in the close and mid-range distances are detected, though their 3D box characterization is not always perfectly adjusted.

The main advantage given by the use of image information is evidenced when comparing Figure 4.12a and 4.12d, or Figure 4.12c and 4.12f. In the former, the fusion model exploits the additional channels to better classify spurious detections as background. In the latter, this data helps in the discrimination between similar categories. Although quantitative results proved that images cues are generally beneficial, they occasionally lead to false positives when, for instance, the 2D detector produces a misprediction with high confidence, as in Figure 4.12e.

4.3 CONCLUSION

In this chapter, the task of on-board 3D object detection for automated driving through the use of LiDAR's BEV projection as input has been addressed. To this end, a novel cell encoding containing an ad-hoc normalized density channel able to provide distance-invariant features has been introduced. Contrary to existing approaches, the proposed projection is sufficient to infer fully characterized 3D boxes without needing auxiliary data sources.

To demostrate its suitability for 3D detection, two different frameworks have been presented. On the one hand, the BirdNet pipeline makes use of a Faster R-CNN architecture to estimate the class and oriented boxes of vehicles, pedestrians, and cyclists in the scene. At the date of publication, the vanilla version yielded state-of-the-art results in the KITTI BEV benchmark among methods using the top-view projection. In its second iteration (i. e., BirdNet+), the accuracy of the model was widely enhanced by extending the learned encoding of the boxes to the third dimension and improving the feature backbone, again outperforming most existing comparable works.

On the other hand, a single-stage approach has been proposed to explore different image-BEV fusion strategies that can be used in real-time automotive applications. The baseline configuration, which solely takes LiDAR data as input, outperforms some recent two-stage detectors in the literature while operating nearly at 50 Hz. Despite the lack of RPN, the network is able to provide fairly accurate detections of small road participants at near distances. When used in conjunction with image information, the model presents assorted behaviors, depending on the selected fusion paradigm. Among the tested schemes, sequential fusion is the only configuration that consistently boosts performance. Combining the raw features in an early fashion has shown ineffective in the BEV space. Regarding the fusion at the feature-level, the proposed strategy has been unable to exploit image cues. However, further investigation may be required. Some ideas to enhance its capabilities will be drawn in Chapter 6.

Table 4.9: BEV and 3D detection performance (AP %) of the different fusion configurations on the KITTI validation set

Cls	Cfg	AP 3D (%)			AP BEV (%)			T
		Easy	Mod.	Hard	Easy	Mod.	Hard	(ms)
Car	B	62.78	50.74	45.53	85.04	75.75	71.34	21
	E1	**63.57**	**51.93**	**47.90**	87.37	**76.42**	**73.48**	22
	E2	62.02	49.62	45.65	86.37	73.51	70.50	23
	S1	62.26	51.02	46.75	87.71	75.60	70.98	77
	S2	62.42	51.12	46.76	**87.85**	76.07	72.92	77
	S3	52.58	42.74	38.89	86.98	74.68	68.63	75
	S4	56.71	47.44	44.19	86.09	74.66	70.49	77
	S5	60.84	47.89	42.75	87.49	75.03	70.40	77
	F1	57.52	45.38	42.03	84.48	72.54	68.21	32
	F2	57.51	45.16	41.90	84.71	72.22	68.13	30
Ped	B	43.65	38.25	34.53	52.81	47.10	43.14	21
	E1	39.51	35.81	32.41	49.56	45.77	42.21	22
	E2	37.18	33.02	29.95	47.17	41.81	38.40	23
	S1	42.43	36.90	33.69	**56.63**	50.52	46.89	77
	S2	**43.70**	**39.29**	**35.91**	56.51	**51.22**	**47.30**	77
	S3	32.45	27.42	24.28	49.44	42.46	37.67	75
	S4	33.35	27.86	24.81	50.67	42.62	38.67	77
	S5	31.22	26.21	23.31	49.36	41.60	37.70	77
	F1	40.43	36.93	33.31	50.62	46.96	43.19	32
	F2	39.73	36.22	33.00	49.84	44.88	41.48	30
Cyc	B	43.96	27.64	25.82	48.39	31.09	28.61	21
	E1	44.22	27.42	25.92	49.71	31.41	29.42	22
	E2	40.58	25.05	23.42	43.06	26.74	25.33	23
	S1	42.42	26.89	25.44	51.34	33.78	32.50	77
	S2	**45.45**	**29.23**	**27.65**	**52.62**	**35.20**	**33.25**	77
	S3	29.63	18.78	17.68	38.93	24.21	22.67	75
	S4	36.28	24.22	22.88	43.58	29.60	26.98	77
	S5	38.60	24.94	23.63	48.21	31.22	29.26	77
	F1	41.47	25.58	24.19	47.91	30.04	28.42	32
	F2	45.44	28.67	26.73	51.07	32.87	29.99	30

Table 4.10: BEV and 3D detection performance (AP %) of different approaches using LiDAR data on the KITTI testing set. LiDAR data can be used in different formats: raw, BEV, RV or voxelized (VX). Some of the methods make use of RGB images (I).

Cls	Method	Data	AP 3D (%)			AP BEV (%)			T
			Easy	Mod.	Hard	Easy	Mod.	Hard	(ms)
Car	F-PC_CNN	I+RAW	60.06	48.07	45.22	83.77	75.26	70.17	500
	MV3D	I+BEV+RV	68.35	54.54	49.16	86.49	78.98	72.23	240
	AVOD-FPN*	I+BEV	83.07	71.76	65.73	90.99	84.82	79.62	100
	F-PointNets	I+RAW	81.20	70.39	62.19	88.70	84.00	75.33	170
	C-YOLO	I+VX	55.93	47.34	42.60	77.24	68.96	64.95	60
	ContFuse	I+BEV	82.54	66.22	64.04	88.81	85.83	77.33	60
	Ours (S2)	I+BEV	67.49	50.72	44.43	85.08	74.10	68.64	77
	MODet	BEV	-	-	-	90.80	87.56	82.69	50
	PIXOR++	BEV	-	-	-	93.28	86.01	80.11	35
	TopNet-Ret.	BEV	-	-	-	80.16	68.16	63.43	52
	BirdNet	BEV	40.99	27.26	25.32	84.17	59.83	57.35	110
	BirdNet+	BEV	76.15	64.04	59.79	87.43	81.85	75.36	115
	Ours (B)	BEV	65.38	48.21	43.01	85.31	74.11	68.64	21
Pedestrian	AVOD-FPN*	I+BEV	50.46	42.27	39.04	58.49	50.32	46.98	100
	F-PointNets	I+RAW	51.21	44.89	40.23	58.09	50.22	47.20	170
	C-YOLO	I+VX	17.60	13.96	12.70	21.42	18.26	17.06	60
	Ours (S2)	I+BEV	27.21	21.62	19.87	35.52	28.61	26.86	77
	TopNet-Ret.	BEV	-	-	-	18.04	14.57	12.48	52
	BirdNet	BEV	22.04	17.08	15.82	28.20	23.06	21.65	110
	BirdNet+	BEV	41.55	35.06	32.93	48.9	42.87	40.59	115
	Ours (B)	BEV	24.69	20.38	18.28	31.71	26.08	24.60	21
Cyclist	AVOD-FPN*	I+BEV	63.76	50.55	44.93	69.39	57.12	51.09	100
	F-PointNets	I+RAW	71.96	56.77	50.39	75.38	61.96	54.68	170
	C-YOLO	I+VX	24.27	18.53	17.31	32.00	25.43	22.88	60
	Ours (S2)	I+BEV	49.15	35.47	31.89	58.72	43.86	39.90	77
	TopNet-Ret.	BEV	-	-	-	47.48	36.83	33.58	52
	BirdNet	BEV	43.98	30.25	27.21	58.64	41.56	36.94	110
	BirdNet+	BEV	65.67	53.84	49.06	70.84	59.58	54.2	115
	Ours (B)	BEV	41.24	30.51	27.77	48.98	38.07	33.88	21

* AVOD makes use of two separate models: one for *Car* and another for *Pedestrian* and *Cyclist* detection.

5 DEPLOYMENT IN REAL PLATFORMS

The breakthrough of *deep learning* and the arrival of large-scale annotated datasets for the automotive sector pushed the scientific community towards the development of new perception algorithms. The use of standard benchmarks enabled a fair comparison of the published works and fostered the research on new machine vision models to outperform existing approaches.

Despite the rapid evolution in the field, the performance of current perception frameworks still suffers if the operational domain is not similar to the one in which they have been trained: when the input data is significantly different between the training and testing stages, the accuracy drops. In the field of scene understanding, the domain gap phenomenom may occur when driving on very distinct scenarios, e.g. cities of unlike countries, under diverse weather conditions, or when information captured by unknown devices is used.

In all cases, the effect is produced by the variations in the signals of the elements around, caused by either real appearance changes or by the disparities between sensors specifications.

To address these limitations, modern datasets like nuScenes [26] include recordings from several countries and climate conditions. This increases the generalization capabilities of the networks and partially alleviates the problem. However, trained models struggle when being fed with data from custom sensor configurations, hampering its deployment on real vehicles.

In this regard, some solutions have been developed to reduce the performance drop issue, under the name of Domain Adaptation (DA). Their goal is to learn domain-invariant features so that the model provides a similar accuracy in both the source (train) and target (test) domains. Although these methods have been successfully applied to tasks like 2D detection [91] or semantic segmentation [205] in images, its application to LiDAR is still limited to certain projections [191].

In this chapter, a proposal tailored to close the domain gap for LiDAR object detection pipelines is presented. Contrary to existing DA approaches, the method aims to generate training data in the target domain using annotations of existing datasets. A set of comprehensive experiments on the KITTI Benchmark [61] supports its suitability for the task. Additionally, a complete perception framework trained following this approach is embedded in a self-driving vehicle. Results of the tests in open traffic further validate the solution.

This chapter includes content from [10], [11], and [121]

5.1 SYNTHETIC LIDAR CLOUD GENERATION FOR 3D PERCEPTION

The unavailability of a sufficient number of multi-modal datasets for the training and evaluation of object detection frameworks and the sensitivity of current perception approaches to changes in the sensor configuration may decelerate their deployment on automated vehicles, which usually mount ad-hoc sensor sets designed to meet the requirements of specific use cases. Significant differences in the extrinsic calibration between sensors or in the LiDAR characteristics produce substantial changes in the perceived representation of the objects, as shown in Figure 5.1. As a consequence, these differences often yield to a degradation of the models' performance, making them unsuitable for demanding applications such as autonomous driving [4].

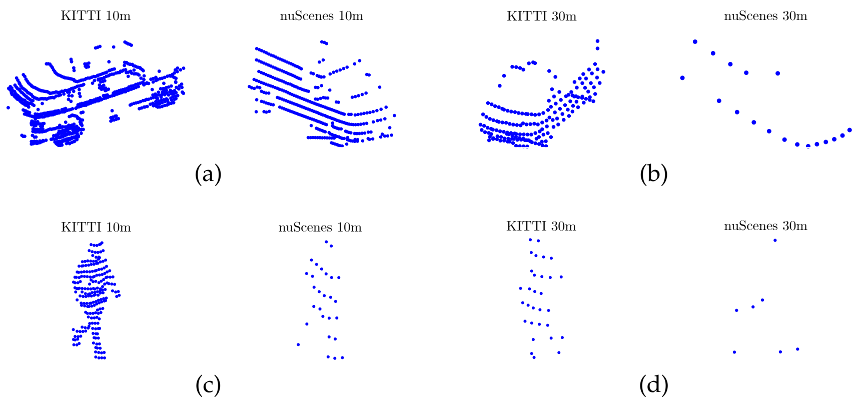


Figure 5.1: Representations of several objects in the scene at multiple distances as captured by KITTI and nuScenes LiDAR devices

To tackle this problem, we propose a method to automate the generation of new annotated datasets for 3D object detection using the information from existing ones. Concretely, the presented approach is geared towards building a 3D mesh representation of the driving scenario using the readings from subsequent LiDAR sweeps, so that a synthetic point cloud of the scene can be simulated for any possible rangefinder model. This way, both the inner specifications of the device and its relative position to the car coordinates can be chosen to recreate available 3D detection benchmarks as if they were captured using a different LiDAR sensor, eliminating the need for recording and labeling ad-hoc datasets whenever a novel sensor configuration is built. As the sensitivity to light reflections of distinct LiDAR devices does not behaves uniformly, this work focuses on the generation of realistic spatial coordinates for the points in the cloud, disregarding intensity values.

Hence, the presented approach is divided into three stages. First, dynamic objects are isolated from the static parts of the scene so that multiple frames can be aggregated properly. Second, a mesh of triangles is approximated to a 3D point cloud resulting from the accumulation of a sequence of LiDAR clouds received within a given time frame. Lastly, a virtual LiDAR device is simulated by ray tracing the laser beams defined by its internal parameters, e. g., layers distribution and horizontal resolution. Even though the formulation of the solution allows for single-frame operation, the joint use of multiple LiDAR clouds has proved to enhance the synthetic output.

5.1.1 *Multi-frame alignment*

Due to the sparsity of the data captured by rangefinders, there are many details of the scene geometry that cannot be represented by the point cloud of a single frame. Besides, oclussions produced by non-static elements such as vehicles and other road participants lead to regions with no laser information. As a consequence, the LiDAR-based 3D reconstruction of the environment benefits from the use of multiple frames captured from a moving platform, as temporary occluded areas might be visible when seen from different viewpoints and the density of the cloud increases as the number of aggregated cloud does.

In order to accumulate frames over time, all point clouds have to be transformed to a common coordinate system so that they can be successfully aligned. However, as some of the elements composing the driving scenario are dynamic, the sweeps aggregation is not an straightforward process, as moving objects will introduce artifacts into the final cloud. To this end, a pre-processing stage is needed so points belonging to the static and non-static parts of the scene, hereon also referred as background and foreground respectively, can be handled separately.

Before any frame can be accumulated, background and foreground clouds have to be computed. To obtain the former, data belonging to dynamic objects has to be filtered out from raw LiDAR data. To this end, the parameters of the 3D object labels $\{\mathbf{t}, \mathbf{s}, \theta\}$ –being $\mathbf{t}$ the translation vector (t_x, t_y, t_z), $\mathbf{s}$ the dimensions (s_x, s_y, s_z), and θ the obstacle heading– are used to select and remove the foreground points.

Once the background points have been isolated for each single LiDAR scan, multiple frames can be added to form an aggregated dense cloud describing the whole scene. For the alignment to be performed, the relative position between LiDAR readings needs to be known. This can be achieved by means of an odometry method, where translation and rotation increments are computed using the information of on-board sensors such as cameras or LiDARs, or by using global positioning provided by GPSs. Although both approaches provide the required relative poses between frames, the latter is usually preferred as it does not suffer from cumulative errors (drift). Fortunately enough, most object detection benchmarks for automotive perception includes a per-frame ground truth of the location and orientation of the vehicle in the world.

Hence, to align multiple sweeps, each frame is transformed from its local coordinates system $\{L\}$ to a shared coordinate system $\{S\}$, following:

$$\mathcal{C}_i^S = (R|t)\ \mathcal{C}_i^L \tag{5.1}$$

being $\mathcal{C}_i^L$ the background point cloud at a certain instant i, $\mathcal{C}_i^S$ the points transformed into the shared axis, and R and t the rotation and translation matrix from the $\{L\}$ to $\{S\}$.

On the other hand, points belonging to dynamic objects at each frame can be combined for the sake of shape completion. In the same way as with its background counterpart, this process reduces sparsity and allows to obtain a more detailed representation of their geometry. Unlike the static cloud, the foreground points cannot be handled as a whole, as the alignment of data belonging to different objects is given by distinct geometrical transformations, i. e., their relative pose between frames differs from one object to another, depending on their motion. Consequently, the aggregation of every object cloud is performed separately. It is noteworthy to mention that changes in the shape of objects are not considered, yielding to some artifacts when accumulating points of deformable road users like pedestrians, as will be discussed in Section 5.1.4.

As for the background alignment, the position and orientation of the foreground elements need to be known for every frame. This information is given by their corresponding 3D annotations, which have been previously used to remove non-static LiDAR readings. Moreover, a unique identifier for the same object over time is required so that clouds of different obstacles are not added together. In this regard, most datasets provide this information. Otherwise, a tracking algorithm similar to the one described in Section 5.2.2.3 can be used to associate labels of the same object at different frames, where labeled boxes can be seen as the output of a single-frame detector.

After extracting all clouds of the same object, they are aligned in a common coordinate system. For simplicity, the selected axis corresponds to the object's coordinates system $\{O\}$. Working in the object local coordinates eases the process, as the alignment of each cloud to this shared axis is given by the inverse transformation of the given annotated 3D bounding box so that axis origin matches the center of the label, and the object's rotation is set to zero. Thus, every point is transformed following:

$$\mathcal{P}_i^O\ =\ (R|t)^{-1}\ \mathcal{P}_i^L \tag{5.2}$$

being $\mathcal{P}_i^L$ and $\mathcal{P}_i^O$ the points of the object at a frame i in local and object axes, respectively, and $R|t$ the transformation between the coordinates systems.

Finally, the dense clouds of every foreground element can be moved back to their original position at a given frame $\{L\}$ by applying the traslation and rotation described by their corresponding label:

$$\mathcal{P}_i^L = (R|t)\ \mathcal{P}_i^O \tag{5.3}$$

5.1.2 *Reconstruction of the scene*

The second phase of the pipeline is tailored to build a 3D mesh that fits the resulting cloud from the motion-aware aggregation of multiple LiDAR frames described in the previous section. This reconstruction aims to serve as a continuous surface able to model the geometry of the traffic scene based on a dense yet sparse cumulative cloud so that synthetic LiDAR signal can be simulated. Henceforward, the described steps are applied to both the object and background point sets independently.

The proposed algorithm is based on [78], who models the shape of the ground surface by fitting the cells of an irregular grid to the points lying inside from a top-view perspective. Due to the nature of LiDAR data, our method operates in spherical coordinates. As a result, every point of the cloud in the Cartesian space $P_c = (x, y, z)$ has to be transformed into spherical coordinates $P_s = (\theta, \phi, r)$ following:

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2} \\ \theta &= \arctan\left(y/x\right) \\ \phi &= \arcsin\left(z/r\right) \end{aligned} \tag{5.4}$$

where θ stands for the horizontal angle of the beam, ϕ for the vertical inclination and r for the range.

The presented meshing approach is divided into two sequential stages. First, a coarse approximation of the scene geometry is performed using an even distribution of triangles over the field of view. Then, a fine-grained mesh is derived to increase the fitting capabilities of the coarse grid to the shape of the environment.

After the transformation between coordinates systems is performed, a rectangular grid with a cell resolution of $\Delta\theta \times \Delta\phi$ is used to divide the spherical space into square tiles. Figure 5.2 shows an intuition of the process in the Cartesian space, as it might be a more familiar representation for the reader. Once the patches are defined, each cell gets assigned a range value r equal to the average range of the points falling inside. The resulting depth map is composed of tiles whose coordinates are contiguous along θ and ϕ, but unlinked in the range axis. As the goal of the method is to build a connected mesh, for every vertex $V = (\theta, \phi)$, which may be a corner of up to four neighbor tiles, its r coordinate is set to the average depth values of the surrounding cells. By modifying the range of the vertices, the patches are divided into two triangles, and the grid becomes connected again. To avoid the creation of surfaces for regions of the scene where no LiDAR information is available, triangles containing no points are removed.

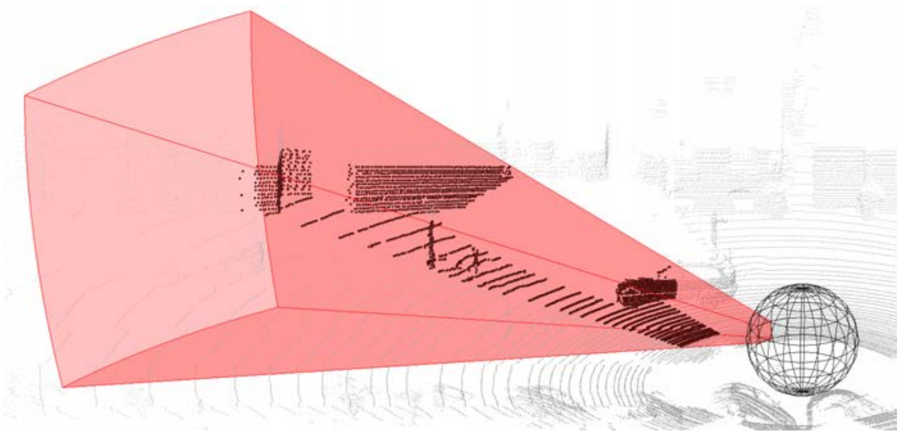


Figure 5.2: Overview of the spherical grid generation process as seen in Cartesian axes. The red volume represents the area covered by a square patch. Points falling inside the highlighted tile are displayed in dark gray [10] © IEEE 2019

The outcome of this first stage is a coarse mesh composed by regular triangular faces of equal size in the $\theta\phi$ plane, whose ability to adjust to the point cloud geometry mainly depends on the dimension of its initial cells. However, picking a resolution is not a trivial task, as any selected value sets a tradeoff between obtaining a more faithful representation of the scene geometry and reducing the amount of *holes* in the mesh, e.g. areas containing no points.

To address this shortcoming, the structure of the resulting mesh is converted into a multi-resolution grid, where every vertex may have between four and eight triangle neighbors, except those located in the borders. In so doing, the former regular distribution of cells becomes a more flexible organization of triangular cells which can be recursively divided to adapt the local resolution of the grid at a given region of the scene where LiDAR information presents high variance. Thus, the adaptability of the mesh can be increased wherever it is needed without generating unwanted holes.

Even though the higher resolution, the greater the fitting capabilities of the mesh, the number of subdivisions of a triangle is limited by the resolution of its neighbors so that the four or eight connectivity of the vertices is preserved. To this end, the procedure is performed in a breadth-first fashion, evaluating the candidacy of every cell at a given resolution level to be subdivided. This way, the grid remains in a valid state at every step of the refinement process.

For a triangle to be considered a candidate, the error between the face plane and the LiDAR points falling inside the patch must be greater than a threshold δ. The fitting error is computed by means of the bidirectional Hausdorff distance between the triangle vertices $\mathcal{V}_i$ and its corresponding point cloud set $\mathcal{P}_i$:

$$d_H(\mathcal{V}, \mathcal{P}) = \max(\widetilde{d}_H(\mathcal{V}, \mathcal{P}), \widetilde{d}_H(\mathcal{P}, \mathcal{V})) \tag{5.5}$$

where the one-sided Hausdorff distance is defined as:

$$\widetilde{d}_H(A, B) = \max_{a \in A} \min_{b \in B} \|a - b\| \tag{5.6}$$

A triangle may be divided by its altitude segment, creating two sibling faces. Whenever a candidate is divided, so does the triangle that meets its hypotenuse (or bottom neighbor), if any. For a *split* operation to be performed, two conditions have to be fulfilled. First, the triangle to be split and its bottom neighbor must have equal size. Second, the newly created children faces must contain at least one point, except if any of the resulting triangles would not have a bottom neighbor. In this case, the empty child is removed. If both requirements are met, the candidate is divided. The refinement process stops when for every triangle in the mesh $d_H(V_i, P_i) < \delta$, or no triangle can be split. Figure 5.3 shows a feasible configuration for a refined mesh.

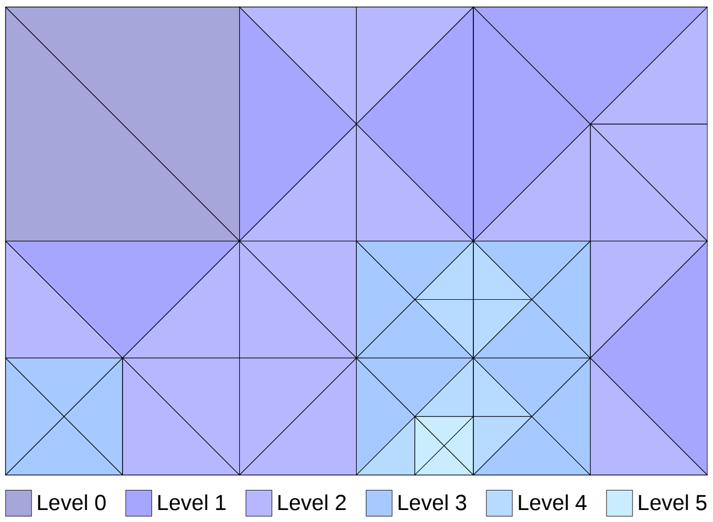


Figure 5.3: Sample of the multi-resolution grid used for mesh approximation in a valid state [10] © IEEE 2019

After the *split* operations are performed, the range coordinate of the newly created vertices must be recalculated. To this end, the approach previously followed for the coarse meshing procedure is conducted so that the final value is computed as the average of the ranges of the neighbor triangle faces. After this step, the multi-resolution mesh becomes connected again.

Once the vertices of the triangles composing the mesh are obtained, they are converted back into Cartesian space by computing the (x, y, z) coordinates of each vertex in the cloud reversing the Cartesian-to-spherical transformation performed before.

Since the reconstruction of the background and foreground meshes are handled independently, grids of unlike resolutions can be used. Moreover, selecting different fitting thresholds δ for static and non-static objects is also possible so that different levels of refinement are applied to each of the meshes. These parameters may be tuned to, for instance, increase the details of the objects' surfaces while keeping a low number of holes in background surfaces such as the road or buildings. Figure 5.4 shows the computed meshes a sample frame. Lastly, background and foreground meshes are merged to reproduce the original layout of the geometry of the environment, as shown in Figure 5.4f.

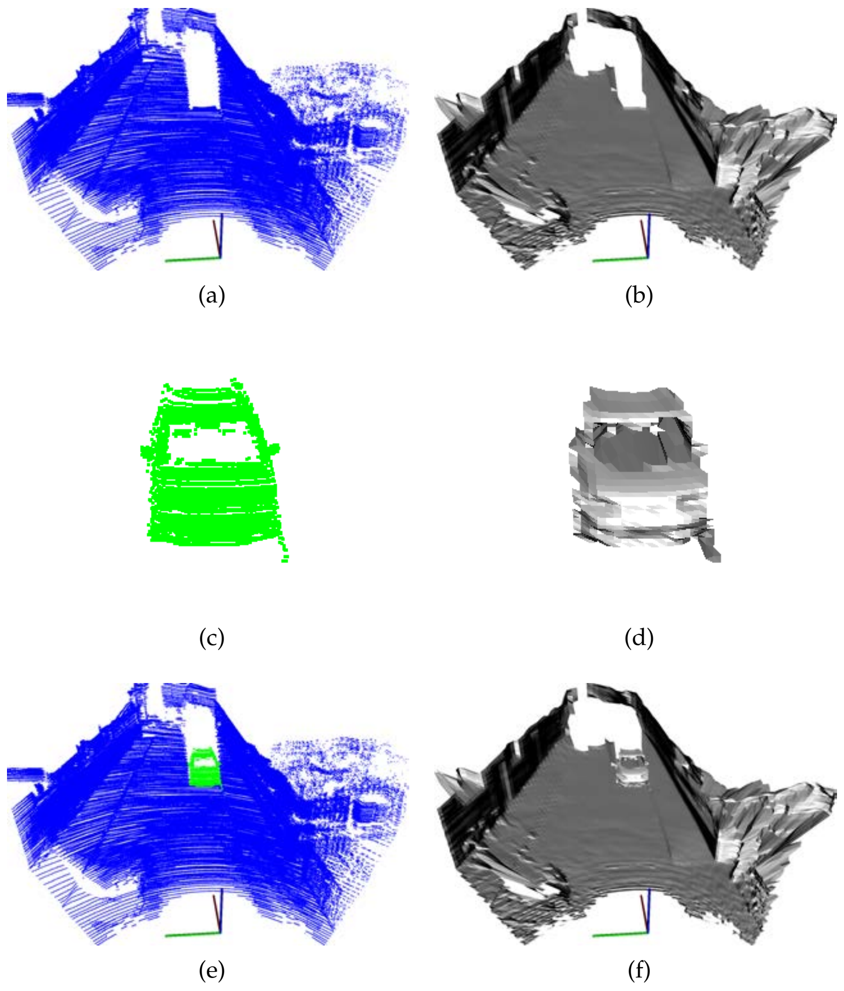


Figure 5.4: Results of the 3D reconstruction pipeline: (a) background points; (b) background mesh; (c) foreground points; (d) foreground mesh; (e) all points; (f) final mesh [10] © IEEE 2019

To provide the reader with a side-by-side comparison of the outcome of the described stages, the coarse and fine-grained meshes of two sample objects are shown in Figure 5.5. The surfaces are represented through the edges of the triangles so that the details can be better observed.

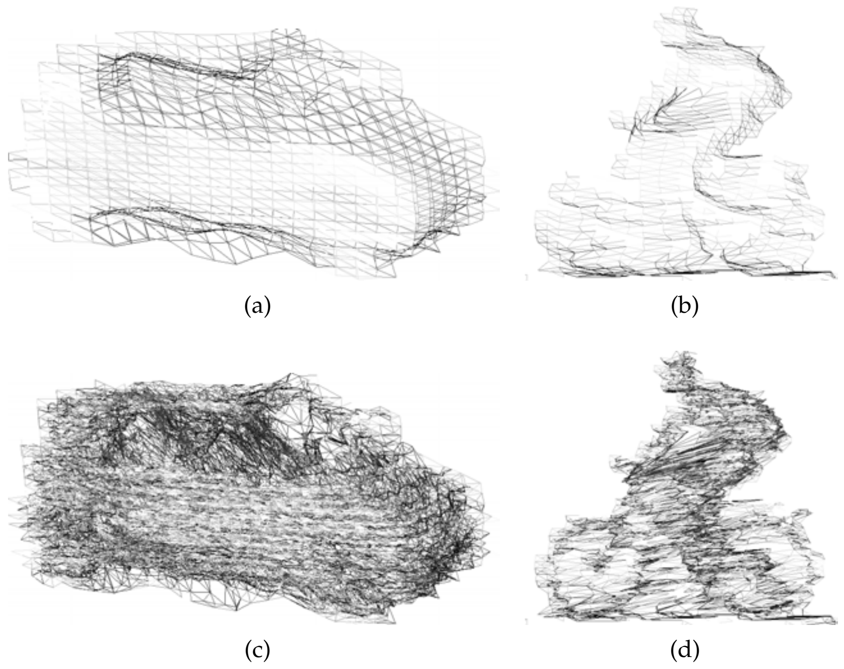


Figure 5.5: Reconstructed meshes of two different objects at the end of the coarse phase (a, b) and after the multi-resolution refinement (c, d) [10] © IEEE 2019

5.1.3 *LiDAR simulation*

The third and last phase of the proposed approach deals with the generation of the synthetic LiDAR scans as if a real device was operating in the scene reconstructed in the previous stage. By simulating a virtual device, a LiDAR point cloud can be computed for every frame from the annotated dataset, thus being possible to create new datasets for any custom LiDAR configuration.

In order to perform the simulation, both the pose and the characteristics of the virtual device have to be defined. Thus, the resulting point cloud depends on two variables: $\xi = (x, y, z, \phi, \theta, \psi)$, which defines the translation along the x, y and z axis and the rotation around x (roll), y (pitch) and z (yaw) of the origin of the desired simulated scanner; and the intrinsic parameters of the sensing unit $L = \{(\Phi, \theta)\}$ where $\Phi = \{\phi_1, \phi_2, ..., \phi_N\}$ being ϕ_i the vertical angle of ring i, N the number of layers of the device, and θ the horizontal resolution or azimuth.

Regarding the simulation procedure, the behavior of real LiDAR devices is mimicked by conducting a ray tracing algorithm. Therefore, a laser beam is triggered from the origin of the sensor following a rotation around the Z axis -using the azimuth as the step size- for all inclination angles described by the distribution of the device rings. For every possible combination of (ϕ_i, θ), the intersection between the corresponding ray and the triangle faces composing the reconstructed mesh is computed. Since the calculation of the intersection between a line and a plane for so many beams at each sweep is computationally demanding, the well-known Möller-Trumbore [125] algorithm is used, as it allows to compute these intersection points without solving the triangle's plane equation.

Though the separate reconstruction of background and foreground meshes brings the possibility of having occluded surfaces once the final scene is composed, the ray tracing algorithm is designed to compute a single intersection point for each laser beam. Thus, although a ray might go through more than one triangle of the mesh, only the collision with the closest face is returned. Beams not colliding with the mesh within a maximum distance are ignored and do not contribute to the generated point cloud.

Despite the inherent measurement inaccuracies derived from the approximation of a mesh surface to an aggregated LiDAR point cloud, an additional noise signal is added to the virtual laser readings in order to increase the realism of the generated data. To this end, a random error following a normal distribution with a mean of zero, based on the specifications of the real device is applied to each point of the cloud.

Figure 5.6 includes the outcome of several executions of the proposed pipeline over the same scene for different LiDAR models. As can be observed, the alignment of just 30 consecutive frames captured when the recording platform is moving is enough to obtain a dense representation of the scene suitable for the aforementioned meshing algorithm. Once the 3D mesh is computed, any LiDAR device can be simulated to generate training and testing data for a given custom sensor configuration.

5.1.4 *Experimental results*

To evaluate the proposed synthetic LiDAR generation approach, a comprehensive quantitative analysis of the resulting clouds is conducted.

The process is performed in a twofold manner. On the one hand, the similarity of the simulated data to their real counterparts is assessed by comparing both clouds with an annotated ground truth originally geared towards benchmarking depth prediction algorithms.

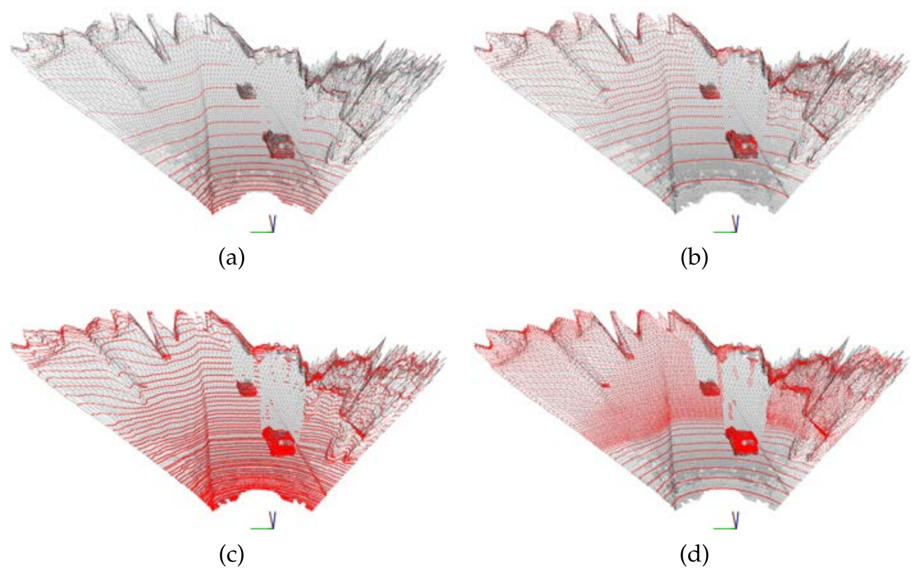


Figure 5.6: Simulation of different LiDAR sensors for the same 3D reconstructed mesh (in gray). (a) Velodyne HDL32E; (b) Velodyne VLP32C; (c) Velodyne HDL64E; (d) Velodyne VLS128 [10] © IEEE 2019

On the other hand, the performance of three different state-of-the-art 3D detection frameworks when trained using the generated laser scans is compared to the results they provide using the original LiDAR information as input. All the selected methods take point clouds as input, although the laser data is processed in different ways. In the work of Qi et al. [141], the cloud is used in its raw format, although 2D detections are used to isolate the frustum candidate regions. In SECOND [196], the LiDAR data is discretized into a spatial grid made of voxels before 3D convolutions are applied. Finally, a fusion pipeline is also evaluated [95], where a BEV projection is created before feeding the network with both laser and image information.

5.1.4.1 *Evaluation of the reconstruction quality*

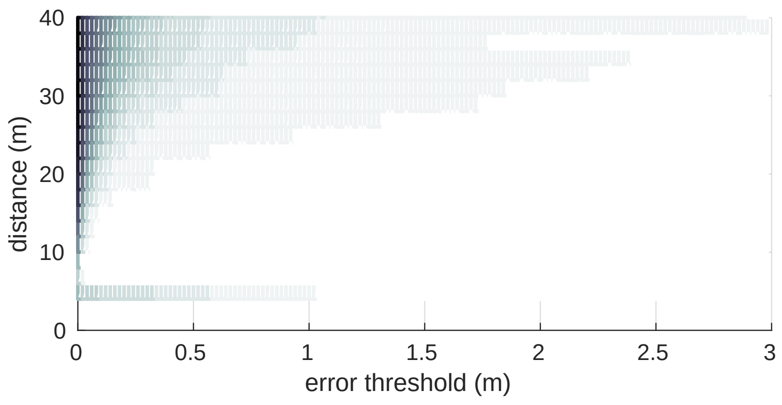
The first type of evaluation aims to determine the quality of the reconstructed mesh by comparing the depth of the simulated LiDAR points to a label depth map. To this end, the KITTI Depth Prediction Evaluation Benchmark [174] is used. This dataset provides depth annotations in the image space so that the precision of LiDAR clouds can be assessed by projecting the points to the camera frame and measuring their corresponding depth error. After image projection, the percentage of correct pixels can be computed by determining the number of points where $\left|D_i^c - D_i^{gt}\right| < \epsilon$ fulfills, being D_i^c the value of the z coordinate of point i in camera coordinates, D_i^{gt} its corresponding ground truth depth, and ϵ the error threshold.

To establish a fair comparison, the intrinsic parameters of the virtual scanner are set to match those of the LiDAR device used in the reference dataset. Therefore, the laser model used to capture both the real and synthetic clouds is a Velodyne HDL64E. Statistics are computed over a random selection of 2.000 frames from the benchmark. For every frame, a window size of 30 sweeps is aligned -including data from both the preceding and subsequent LiDAR sweeps- to enhance the output of the 3D reconstruction phase. Regarding the meshes configuration, a spherical grid with a cell resolution of $\Delta\theta = 1$, $\Delta\phi = 2$ and a fitting threshold of $\delta = 0.1$ is used for the background elements. The size of the tiles for the coarse foreground mesh remains equal, although the adaptation capabilities of the reconstruction are increased by setting $\delta = 0.01$. These parameters have been empirically obtained and may be adjusted for other applications or LiDAR devices with significant differences.

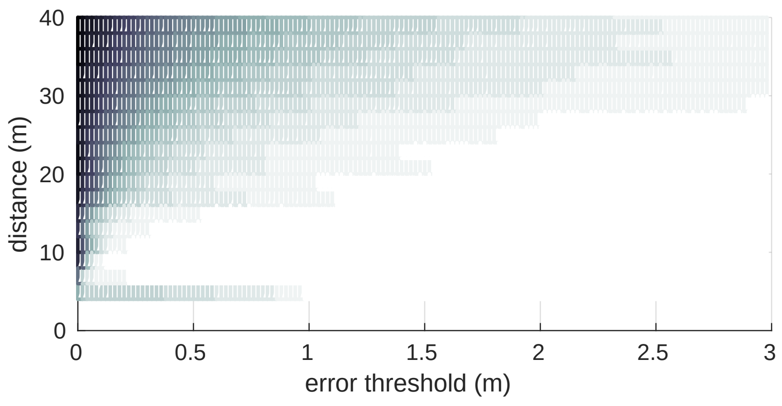
Figure 5.7 depicts the depth error of both the original LiDAR cloud and the synthetic signal as the percentage of points located at different distances whose range error is lower than a certain threshold. As can be observed, the real laser data presents differences when compared to the manually annotated ground truth depth map, being particularly noteworthy those of points falling beyond 20 meters distance. Regarding the simulated cloud, it obtains comparable metrics for near distances, where the error falls under the specifications of the real device margins. These remarkable results validate the suitability of the proposed meshing algorithm, even though the scene geometry is approximate from partial data aggregated from a few laser sweeps. On the other hand, as the range of the points increases, the quality of the signal degrades proportionally. This situation is explained by two different factors that directly determine the number of points available at any range: the number of scans used during the aggregation step and the traveling distance of the vehicle within the corresponding time frame. For longer distances, the number of frames considered at the meshing stage becomes a key factor, as the density of the cloud significantly increases as the ego-vehicle moves forward. Hence, for larger window sizes, the sparsity of the aggregated LiDAR cloud decreases and, as a result, a finer resolution can be computed for distant scene regions, reducing the reconstruction error.

5.1.4.2 *Evaluation through 3D object detectors in KITTI*

The second evaluation criteria focus on assessing the validity of the simulated LiDAR datasets when used to train different 3D object detection frameworks. Concretely, a comparison of the performance of the aforementioned networks is made between models trained with real and virtual data. As in the evaluation of the reconstruction quality, both the specifications of the simulated device and its positioning



(a)



(b)

Figure 5.7: Correct points (%) for increasing error thresholds at different distances to the reference sensor. The scale of the colormap ranges from 0% (black) to 100% (white). (a) Original Velodyne HDL64E. (b) Simulated Velodyne HDL64E [10] © IEEE 2019

in the ego-vehicle mirror those of the original rangefinder. Moreover, to guarantee a fair analysis, both models are fed with the original laser scans at the validation stage. In so doing, insights on the degree of similarity between real and synthetic sweeps can be obtained.

The well-known KITTI Object Benchmark is used to conduct the described experiments. Results of the different frameworks for BEV can be found in Table 5.1, whereas those for 3D are depicted in Table 5.2. As in the experiments presented in Chapter 4, the Average Precision (AP) metric is used to determine the ability of the methods to detect cars, pedestrians, and cyclists, using Intersection over Union (IoU) thresholds of 0.7, 0.5, and 0.5, respectively. Likewise, the performance at the three standard difficulty levels is shown.

As can be seen, the performance of all selected methods for the detection of vehicles in the BEV is comparable when using either real or simulated LiDAR clouds as training data. In the 3D evaluation, the Frustum network suffers more than the other two models, as the differences between clouds have a greater impact on methods

Table 5.1: KITTI BEV object detection results on the validation set for models trained with real and synthetic inputs [10] © IEEE 2019

Class	Method	AP BEV (real)			AP BEV (synthetic)		
		Easy	Moder.	Hard	Easy	Moder.	Hard
Car	Frustum	86.93	81.27	72.91	82.27	72.57	63.98
	AVOD	89.03	79.84	79.41	75.36	78.28	77.79
	SECOND	89.83	86.93	79.42	89.47	79.07	78.38
Ped.	Frustum	70,21	60.79	53.10	64.42	55.77	49.38
	AVOD	53.47	48.44	42.50	36.21	32.62	30.65
	SECOND	62.33	59.33	53.21	65.20	56.42	54.61
Cyc.	Frustum	78.66	57.51	53.69	64.63	48.32	45.79
	AVOD	64.58	40.41	39.86	44.51	29.94	26.89
	SECOND	80.58	62.23	60.80	67.64	53.77	48.54

using the LiDAR information in its raw format. Regarding the 3D pedestrian detection, both AVOD and Frustum experience a significant drop in precision when synthetic LiDAR is used at the training stage. Conversely, SECOND shows similar results for both models as the voxelization of the cloud makes the framework less sensitive to small changes in the input point cloud. In the BEV projection, the performance gap between the models is dramatically lower. For the cyclist class, the same trend is observed for all three approaches when trained using clouds from virtual devices: the metrics are lower than when using the original data. However, it is noteworthy to mention that the results for the cyclist class are also the worst among models trained using real LiDAR scans. The results for both pedestrian and cyclists categories can be explained by the higher variance in their point cloud representation, mainly due to their different appearance over time, as the multi-frame alignment stage does not take into account the movements of the person nor the variations in the bicycle shape caused by the turns of the handlebars or changes in the position of the pedals.

Although the accuracy naturally drops when using simulated laser scans for training, some of the analyzed models still offer satisfactory object detection results, especially in the BEV projection. In fact, the quality of the generated point clouds allows achieving alike performance in object classes with fewer variations in their geometry, such as vehicles. Moreover, the results obtained using SECOND prove the utility of the approach for the creation of LiDAR datasets to successfully train precise voxel-based 3D networks for the detection of different categories of road participants.

Table 5.2: KITTI 3D object detection results on the validation set for models trained with real and synthetic inputs [10] © IEEE 2019

Class	Method	AP 3D (real)			AP 3D (synthetic)		
		Easy	Moder.	Hard	Easy	Moder.	Hard
Car	Frustum	81.96	68.76	60.84	69.39	49.99	42.57
	AVOD	77.17	68.20	67.73	75.36	64.99	63.68
	SECOND	87.66	77.57	75.61	84.95	67.14	60.02
Ped.	Frustum	63.07	54.59	47.41	52.46	44.69	39.25
	AVOD	44.99	44.73	39.26	32.24	29.08	26.96
	SECOND	60.01	52.72	50.20	61.87	53.67	46.76
Cyc.	Frustum	74.98	54.19	49.84	57.98	42.43	40.34
	AVOD	63.00	39.57	38.72	40.57	28.67	25.93
	SECOND	79.01	60.29	55.11	65.99	52.10	46.94

5.1.4.3 *Evaluation of the cross-domain performance*

To further investigate the adequacy of the proposed method, experiments have been conducted to provide a quantitative analysis of the domain adaptation capabilities provided by the generated training datasets. To this end, the data from KITTI [61] is used at the training stage, while the nuScenes 3D object detection benchmark [26] is set as the target domain, and therefore employed for the evaluation. As nuScenes' annotations are available, a fair comparison can be made between models trained with different LiDAR inputs. It is noteworthy to mention that the specification of the LiDARs from KITTI and nuScenes domains differ significantly, as they were recorded using a 64-layer Velodyne HDL64E and 32-layer Velodyne HDL32E, respectively. Furthermore, the position of the sensors in the car, as well as the location of the scenes, also vary from one to another, increasing the training-testing gap even more.

The assessment is performed through the Frustum PointNet v1 framework [141], as it has shown to be the most sensitive against changes in the input LiDAR data. Two models have been trained using distinct training datasets. For the first one, KITTI raw clouds are used. For the second model, virtual scans as measured from a Velodyne HDL32E are generated using the reconstruction of KITTI scenes. In the latter case, the original position of the LiDAR in KITTI is modified to match the exact sensor configuration in nuScenes. Additionally, a third model is trained using the *train* split from nuScenes dataset so that it can be used as a reference.

The official nuScenes evaluation metrics are used to measure the accuracy of the models for the task of 3D object detection. On the one hand, the AP is computed for all classes using the distance between the centers of the detection and ground truth objects as the association condition, instead of the traditional IoU. The AP is calculated by integrating the precision vs. recall curve for values greater than 0.1. The per-class final value is the result of averaging over four different distance thresholds $\{0.5, 1, 2, 4\}$. On the other hand, a set of metrics are evaluated for the True Positive (TP) detections:

- Average Translation Error (ATE): Euclidean distance between centers in the XY plane, in meters.
- Average Scale Error (ASE): calculated as $1 - IoU$ after aligning centers and orientation.
- Average Orientation Error (AOE): smallest yaw angle difference between prediction and ground-truth boxes, in radians.

Although the benchmark includes two other metrics for the TP boxes, i. e., Average Velocity Error (AVE) and Average Attribute Error (AAE), we have disregarded them in our experiments since they are not applicable to the task under evaluation.

All models are trained for 80 epochs with a batch size of 32 objects. Adam [92] optimizer is used, and the rest of the hyperparameters are set as follows: a learning rate of 0.001, a momentum of 0.9, and a decay rate of 0.5 with a step of 15 epochs. Frustum candidates are generated using detections from a Mask R-CNN model pre-trained on ImageNet and COCO, and finetuned on Cityscapes.

To be able to train and validate a model using different datasets, an agreement between the labeled categories is required. In our particular case, only the KITTI classes *Car*, *Pedestrian* and *Cyclist* (*Bicycle* in nuScenes) have been considered, since there are no straight correspondences among the rest of annotated types in the two benchmarks. Their respective results in the nuScenes *validation* set are shown in Table 5.3, Table 5.4, and Table 5.5.

Table 5.3: F-PointNet [141] 3D *Car* detection results in the nuScenes benchmark [26] using different training datasets

Training dataset	AP	ATE	ASE	AOE
Original nuScenes	0.507	0.358	0.172	0.462
Original KITTI	0.341	0.498	0.316	0.756
KITTI Virtual HDL32E	0.419	0.473	0.329	0.809

As can be observed, there is a large gap between the performance of models trained using raw KITTI and nuScenes clouds, which confirms the magnitude of the domain adaptation problem. On the other

Table 5.4: F-PointNet [141] 3D *Pedestrian* detection results in the nuScenes benchmark [26] using different training datasets

Training dataset	AP	ATE	ASE	AOE
Original nuScenes	0.574	0.265	0.291	1.031
Original KITTI	0.425	0.347	0.334	1.296
KITTI Virtual HDL32E	0.502	0.352	0.320	1.302

Table 5.5: F-PointNet [141] 3D *Cyclist* detection results in the nuScenes benchmark [26] using different training datasets

Training dataset	AP	ATE	ASE	AOE
Original nuScenes	0.200	0.413	0.507	1.630
Original KITTI	0.048	0.513	0.456	1.299
KITTI Virtual HDL32E	0.107	0.516	0.451	1.455

hand, the AP of the network trained using the simulated point clouds consistently exceeds the one of the KITTI baseline in all evaluated categories. Concretely, the use of our proposed training dataset leads to improvements of around 23%, 18% and 123% in the 3D detection of *Car*, *Pedestrian* and *Cyclist* objects, respectively. Here we can see that despite the difficulties to accumulate clouds from subsequent frames for non-rigid objects noticed in the previous experiments, the model trained using data from the virtual LiDAR device clearly outperforms its original counterpart.

Differences in the TP metrics are not significant, with slight changes in the average translation, size, and orientation errors. This suggests that there is an underlying limitation in the method performance that can be explained by the variations in the geometry, poses, and quantity of the annotated objects among datasets. The results provided by the model trained on nuScenes also support this hypothesis.

Although the selected 3D detection framework is heavily affected by minor alterations of the input, since it processes LiDAR data in its raw format, the conducted experiments demonstrate that the presented pipeline for laser scans simulation is an effective approach for reducing the gap when there exist major differences between the training and deployment domains.

5.2 MULTI-MODAL 3D DETECTION IN OPEN TRAFFIC

In order to obtain on-field insights about the suitability of the synthetic LiDAR datasets created through the method introduced in the previous section, a 3D object detection network has been trained to serve as the main perception stack of a self-driving car prototype.

The vehicle used in the tests is a Renault ZOE owned by the Renault SAS Research Department, and has been instrumented with a custom sensor configuration to navigate autonomously in slow and medium-speed roads. Concretely, the ODD involves daylight open traffic driving under good weather conditions in periurban areas around Guyancourt, Île-de-France, a commune located in the south-western suburbs of Paris, France. Moreover, the driving maneuvers are limited to those required for path following, e. g., turns, roundabouts, or yields, with the exception of overpassing. Traffic regulations are obeyed with the help of an HD annotated map. Thus, the operational domain is geographically restricted by map availability.

Henceforward, the details of the sensor setup, the processing units, and the complete object detection and tracking pipeline are presented. Different experiments in both benchmarks and open traffic validate the proposed solution.

5.2.1 *Sensor configuration*

Even though the requirements of the use cases may allow for simpler configurations, the selected sensors aim to provide enough flexibility so that the perception system of the research platform can be scaled to solve even more challenging tasks to widen the scope of the ODD in the near future. Namely, it is composed of a set of cameras and a laser scanner.

To be able to perform the necessary maneuvers in a safe manner, the proposed sensor setup covers the 360° with both modalities. All the sensors are mounted on a rack on the roof of the vehicle. The LiDAR scanner stands in a central position, while five cameras are evenly distributed around it to cover the full Horizontal Field of View (HFOV) around the vehicle with some overlap between images, as shown in Figure 5.8. In particular, the following sensors are employed:

- Five CMOS cameras equipped with an 85°-HFOV lens (FLIR Blackfly S 31S4C-C9).

- A 32-layer LiDAR scanner featuring a minimum vertical resolution of 0.33° and a range of 200 m (Velodyne VLP32C).

Sensors have been selected to ensure optimal performance in the short-to-medium range. The image resolution provided by the cameras' sensors is high enough to enable the use of pixel binning (by a factor of 2× in both directions) to increase the sensitivity to light. Besides, a GPS/INS receiver with RTK corrections is equipped to have accurate positioning and heading information of the ego-vehicle.

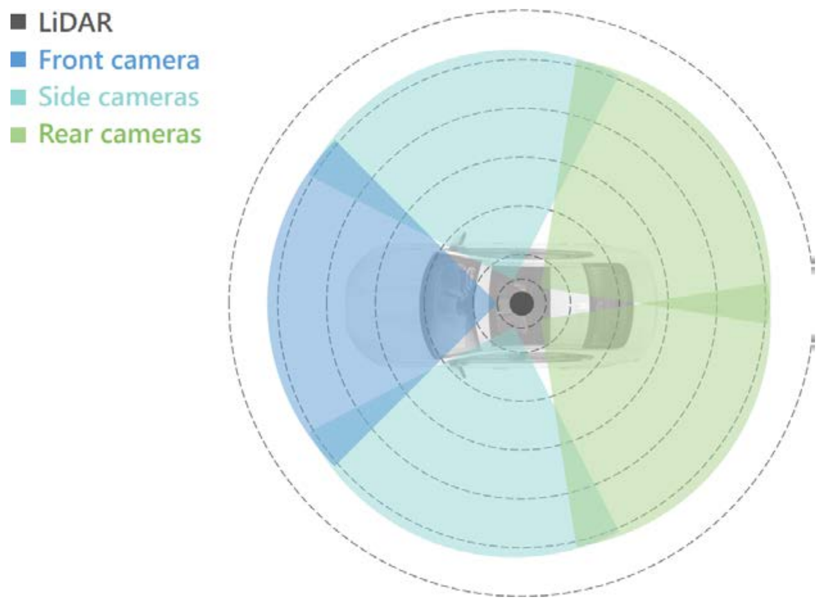


Figure 5.8: Top-view of the sensor setup of the vehicle [121] © IEEE 2021

The selected configuration has been chosen to take advantage of both the representation capabilities of the cameras, which provide appearance information, and the geometrical accuracy of the LiDAR scanner, which delivers range and intensity measurements, through a sensor fusion approach that exploits the correspondences between modalities. To do so, an accurate synchronization and calibration between sensors are required. Otherwise, the data captured by each of them would describe different moments in time, making it impossible to compute a faithful estimation of the current environment state.

For that purpose, two separate requirements have to be fulfilled. On the one hand, all the sensors have to be fired at the same instant. In this regard, an ad-hoc device has been created to emit a signal that triggers all cameras at once. The LiDAR unit is not included as it captures information in a continuous way due to its rotating mechanical design. On the other hand, all sensors need to share a common clock, so when they gather new data, their timestamp can be used to easily associate the five images and the laser scan belonging to the same frame. To this end, the IEEE-1588 Precision Time Protocol (PTP) is used, where the GPS is set as the master clock and the rest of the sensors and the computer act as slave devices.

At the calibration stage, the intrinsic parameters of the cameras are computed using the popular checkerboard-based approach in [207]. In order to estimate the relative position between the sensors, the automatic extrinsic calibration method introduced in Chapter 3 is used in a pairwise fashion between the LiDAR and each of the surrounding cameras.

Regarding the rest of the hardware configuration, data processing is carried out on a unit with four NVIDIA Tesla V100 GPUs, with

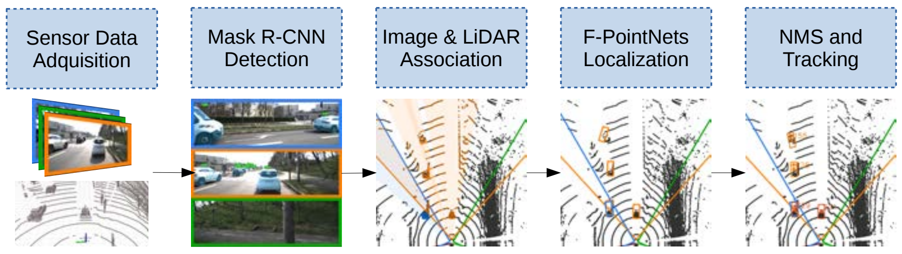


Figure 5.9: Overview of the proposed perception stack [11] © IEEE 2020

32 GB VRAM and 15.67 TFLOPS FP32 each. Additionally, the computer features 40 CPUs, 256 GB RAM, and several SSD disks for storage.

5.2.2 *Perception pipeline*

The proposed perception stack is composed of three separate stages: 2D detection, 3D estimation, and tracking. Since the inference of the final 3D boxes relies on the well-known Frustum-PointNet framework [141], a previous step is required to feed the network with the necessary image object detections. Once the object instances have been properly located and classified, the tracking module provides consistency over time. By tracking the dynamic agents across frames, the impact of instant misdetections in the preceding stage can be mitigated. The combination of these three components enables accurate and robust identification of the different road participants surrounding the vehicle. An overview of the whole pipeline is shown in Figure 5.9.

To control the correct funcioning of the system, a supervisor program monitors the status of every module to detect critical failures while in operation so that the safety driver can be notified to switch to manual driving. The different software components have been implemented using the Robot Operating System (ROS)[1] libraries, which provides an off-the-shelf inter-process communication protocol, among other convenient features for the development of real-time robot applications. A detailed description of each of the three aforementioned stages is provided in the following sections.

5.2.2.1 *Image-based object detection*

Since the dense and rich appearance information provided by camera sensors generally offers superior performance over other sensor modalities for object detection purposes, the proposed solution is built upon a robust 2D detection network. Namely, Mask R-CNN [72] is responsible for the first stage of the pipeline. This two-stage method is an evolution of the well-known Faster R-CNN meta-architecture described in Section 2.4.3, where an additional inference branch has

1 https://www.ros.org/

been added for estimating a per-object instance segmentation mask. Figure 5.10 shows an overview of the selected 2D detector.

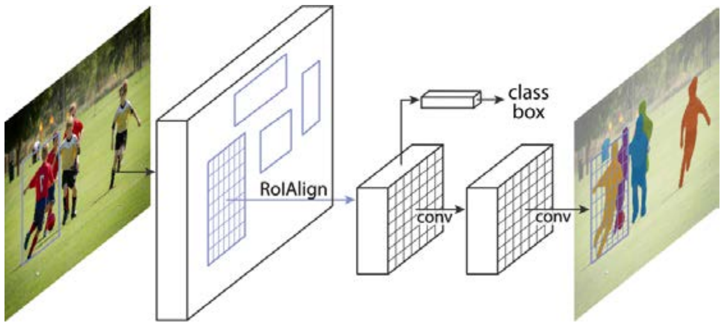


Figure 5.10: Mask R-CNN detection framework [72] © IEEE 2017

The choice for a 2D detector based on Faster R-CNN relies on the higher capabilities of two-stage frameworks for the detection of small objects over existing single-stage approaches [79], which is a key requirement of driving applications in order to be able to perform maneuvers well in advance. Despite the greater processing times associated with the usage of the RPN, modern GPUs such as the ones equipped in the on-board computer are able to provide throughputs suitable for online operation.

The improvement of Mask R-CNN over Faster R-CNN is twofold. On the one hand, the model is able to estimate a semantic mask for each of the detected objects. On the other hand, the joint multi-task training not only does not negatively hamper the original object detection branches but benefits its performance instead while providing an extra pixel-wise classification. From the computational point of view, endowing Faster R-CNN with the mask estimation head does not have a significant impact on the overall processing time, as features are shared among all branches.

Following the Faster R-CNN paradigm, the inference of the network is divided into two different stages. First, a feature encoder or *backbone* made of convolutional layers is used. The outcome of the encoder is then fed to the RPN, which generates a set of candidate regions where an object is likely to be found. These ROIs are then used to pool the previously extracted feature map so that potential objects are fed into the network heads. In the second stage, the three different branches process the instance crops to infer an estimate of the class, final bounding box, and semantic mask. The model is trained using a multi-task loss that takes into account the contribution of the different outputs of the RPN and the three per-object heads.

In the proposed on-board solution, an instance of Mask R-CNN is run to process the data captured by each of the five surrounding cameras. Before inference is performed, original images are cropped so that uninteresting areas belonging to the sky or the ego-vehicle are removed. Thus, the input size is reduced, and so it does the required execution time. The selected backbone network is the ResNet-50 [73], due to its excellent accuracy-speed tradeoff. Moreover, to further increase the performance of the detection of small objects, the information from feature maps at different scale levels is combined at both the region proposal branch and the final estimation heads following a FPNs approach [109].

Regarding the training, multiple datasets are employed. The model is pre-trained in a two-step fashion using ImageNet [94] and COCO [111], images, one after another. The use of large-scale vision benchmarks enhances the learned convolutional kernels and provides greater generalization. Later, the model is fine-tuned to perform well on the desired tasks of object detection and instance segmentation. For this purpose, the Cityscapes dataset [33] is used.

Since the output of the Mask R-CNN network is geared towards the estimation of candidate regions for the subsequent 3D object detector, which is trained using KITTI benchmark [61], the categories from Cityscapes are adapted to match those in the next stage. To this end, instances of *bicycle* are combined to their closer *rider* to compose a KITTI-like *Cyclist* label. Thus, the model is trained to detect *Car*, *Pedestrian*, and *Cyclist* objects.

5.2.2.2 *LiDAR-based 3D box estimation*

To fill the gap between the detecions in the image plane provided by Mask R-CNN and the desired 3D bounding boxes, the spatial information from the LiDAR modality is employed through the use of the Frustrum PointNet framework [141]. This model aims to leverage the use of both visual and LiDAR data in a sequential fusion approach. On the one hand, the whole 3D detection process is based on the result of the image inference, which offers excellent performance in detecting road users. On the other hand, it also exploits the highly accurate and precise geometric information provided by laser scanners in those tasks in which it is most appropriate. Besides, the method achieves outstanding results, among the best in renowned benchmarks such as KITTI, while using very lightweight models for inference. In the proposed perception stack, the F-PointNet v1 model is used, as the improvements introduced by the second version were proved not significant in this context.

The outcome of the 2D object detector is used to create regions of interest in the laser point cloud, which are then processed to estimate the final cuboids. In this regard, the association between LiDAR and camera data is computed by projecting the laser points onto the image plane of the corresponding surrounding camera. For this purpose, the relative pose between the sensors is considered. Afterward, the region of the cloud whose projection falls inside the 2D coordinates of each object candidate, known as *frustum*, is extracted. Although the search space of these *frustums* is limited, the estimation of the accurate 3D box parameters is far from being a trivial task since the LiDAR cloud only represents the visible geometry of the object and may include outlier points belonging to other elements of the scene.

Afterward, the isolated point cloud of every object is used as input of the F-PointNet model. The network is responsible for providing an accurate estimate of the location, size, and orientation of obstacles. To do so, this architecture is made of three successive stages, as seen in Figure 5.11. First, a PointNet module tailored to perform 3D a point-wise binary segmentation of the *frustum* cloud is used to filter out points not belonging to the object (i.e., outliers). This way, the noise is removed, and the next parts of the pipeline are fed solely with points of interest. The second stage, known as T-Net is in charge of providing an initial estimate of the center of the object so that points can be transformed to this new axis before the final box inference. In doing so, the variance between the signals of different objects is reduced, facilitating the learning process of the 3D cuboid parameters. Finally, an amodal PointNet is used to infer the 3D object size and coordinates.

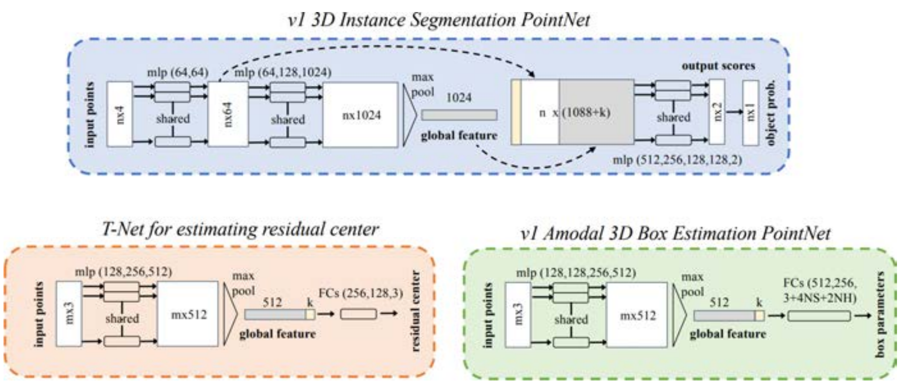


Figure 5.11: Frustum PointNet v1 pipeline [141] © IEEE 2018

To train the model, the KITTI 3D Object Benchmark [61] is used. Due to the significant differences between the layer distributions of the LiDAR device in the instrumented vehicle and the sensor in KITTI, a derived ad-hoc dataset generated through the method proposed in Section 5.1 is used. Therefore, the F-PointNet framework is trained with synthetic clouds simulated with a virtual device sharing the configuration of both the intrinsic parameters and the relative pose with the one in the sensor setup of our vehicle, using the reconstructed scenes from the german benchmark.

For deployment, we mimic the approach followed for the image modality, so the 3D detections are computed separately for each of the cameras. Furthermore, the cloud projection required for the LiDAR-camera association step is accelerated through a GPU-based implementation since the number of points of the captured laser scan is bulky enough to compromise the real-time performance of the solution. In the same way, a custom implementation based on NVIDIA TensorRT[2] is used for the five Frustum PointNet instances.

5.2.2.3 *Object tracking*

Once the object detection stage is finished, a set of 3D cuboids representing the objects in each of the cameras' Field of View (FOV) is obtained. For the tracking module to properly estimate objects trajectories in 360°, these detections have to be combined in a common coordinate system. To this effect, the LiDAR axis is used. Although the transformation from the camera to the laser coordinates is performed through a simple matrix product using the calibrated extrinsic parameters, the process of fusing the detected objects is not straightforward. Since images from consecutive cameras share an overlapping area to prevent blind spots, multiple detections of the same object may be estimated. To remove duplicate instances, a NMS procedure is applied on the BEV (i.e., ignoring the height coordinate). For efficiency reasons, a class-aware axis-aligned approach is followed.

After expressing all detected objects in the LiDAR frame, they can be fed into the tracking stage. This module is responsible for providing consistency over time for the instant detections computed at each frame in the previous stages of the pipeline. In particular, this task is in charge of endowing every object with a unique identifier stable accross frames, as well as predicting the position of objects when they are temporarily occluded. The selected tracking algorithm for this perception solution is the square root version of the Unscented Kalman Filter (UKF) [186]. The use of this variant of the UKF brings extra stability to the filter since it always ensures a positive-definite covariance matrix, avoiding numerical errors [179].

In order to adapt the prediction to the behavior of every kind of agent, the different types of obstacles are modeled by per-category cinematic equations. The state variables leverage the information estimated at the object detection stage and include both the 3D position and the orientation of the obstacles. Besides, the motion of the ego-vehicle is compensated using the location and heading data of the GPS/INS receiver so that the prediction of the previously tracked agents and the new observations belong to the coordinate system of the current timestep.

Finally, to deal with the association between detected objects over time, the Hungarian algorithm is used. The cost matrix for the pairwise association of existing agents and input detections is computed by means of the Mahalanobis distance, taking into account the uncertainty estimated by the Kalman filter. When a tracking object cannot be associated with any new detection at some timestep, it remains in an invisible state and continues being tracked in the background. This

2 https://developer.nvidia.com/tensorrt

provides temporary consistency in the event of misdetections. Concretely, the tracking management system makes use of a per-agent score, which increases every time the object is associated to a new detection, and decreases otherwise. If the score falls below a certain threshold, the agent is deleted.

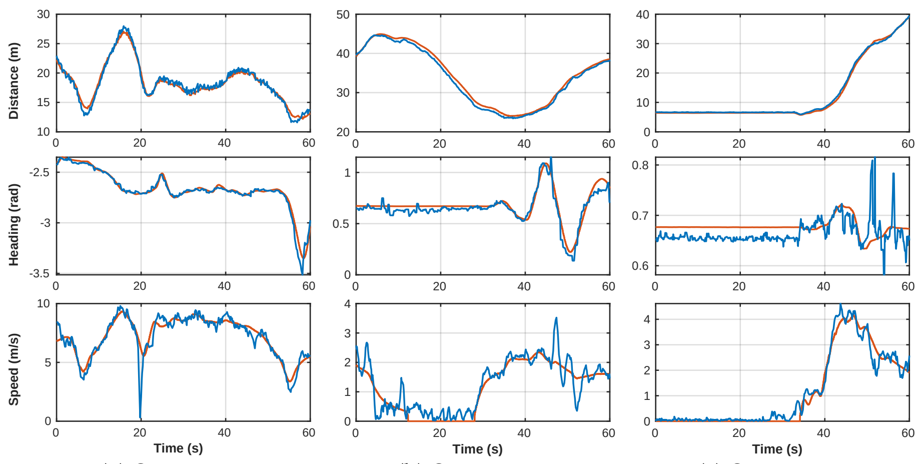
5.2.3 *Experimental results*

To evaluate the performance of the proposed perception stack, the solution has been deployed on the research vehicle so that it can be validated in the traffic situations included in the ODD. Different tests have been conducted, where the car has been driven on a manifold of urban and peri-urban scenarios sharing the road with other traffic participants. Despite the fact that all the stages of the pipeline rely on well-proven approaches, a systematic analysis is required to assess the suitability of the synthetic LiDAR dataset used for the training of the models to reduce the domain gap between benchmarks and real scenarios. Since there is no ground truth for the operational domain, the results of the full system are validated through both controlled experiments and qualitative observations.

On the one hand, the accuracy of the method is assessed by means of an auxiliary vehicle driving in the surroundings. The car, which features a GPS/INS unit with RTK corrections, identical to the one mounted on the ego-vehicle sensor rack, is used to collect information of its position, orientation and speed so that it can be used as the reference signal for the evaluation of our approach. Hence, provided that the dimensions of the instrumented vehicle are known, an automatic 3D label of the object can be obtained. Although the tests are conducted in open traffic and thus involve other road agents, the analysis is only presented for the reference vehicle, as the annotations of other participants are not available. Figure 5.12 shows the results of three typical use cases where a correct performance of the perception system is of paramount importance: an urban roundabout with moving traffic, a lane change during a traffic jam, and a pedestrian crossing.

As can be seen, the system is able to detect and track the reference vehicle with high accuracy, providing minimal error in the distance and heading estimation. Remarkably, the performance of the perception pipeline remains stable beyond $40\,\mathrm{m}$. Regarding the speed, although the predicted output is generally noisier, the error in the estimation remains reasonably low. It is noteworthy that the filter is robust against the inaccuracy of GPS measurements in urban areas, which might affect the ego-vehicle movement compensation stage.

Table 5.6 shows the average estimation errors of the pipeline on the above sequences. The metrics include the mean distance (on the ground plane), mean heading, and mean speed errors per sequence.



(a) Sequence 1 (b) Sequence 2 (c) Sequence 3

Figure 5.12: Comparison between the predicted distance, heading and speed of the instrumented vehicle (in blue), and the reference signal (orange) on three testing sequences [11] © IEEE 2020

The results observed on the different tested scenarios prove the high reliability and accuracy of the approach and demonstrate its proficiency in consistently tracking agents over time.

Table 5.6: Error metrics for the test sequences in Figure 5.12 [11] © IEEE 2020

	Seq. 1	Seq. 2	Seq. 3
Mean distance error (m)	0.48	0.71	0.30
Mean heading error (rad)	0.02	0.04	0.02
Mean speed error (m/s)	0.42	0.27	0.17

Figure 5.13 and Figure 5.14 depict some examples of the performance of our detection solution in different traffic situations. As evidenced by these results, the 360° capabilities of the system make it suitable for the identification of dynamic obstacles even in challenging situations such as intersections and roundabouts. Most roads users are correctly detected and provided with a representative 3D box. Some difficulties are found at the overlapping areas between consecutive cameras, where near objects appear truncated in both images leading to spurious 3D estimates.

On another note, whenever a software solution is deployed to solve a problem with real-time requirements like perception tasks for automated driving, it is of paramount importance to measure both the runtime and the frame rate of the system outputs to assess its suitability to the specific application.

As mentioned before, due to the nature of the F-PointNet framework, the 2D and 3D detection of the objects of each camera is computed separately. This situation allows distributing the ROS processes

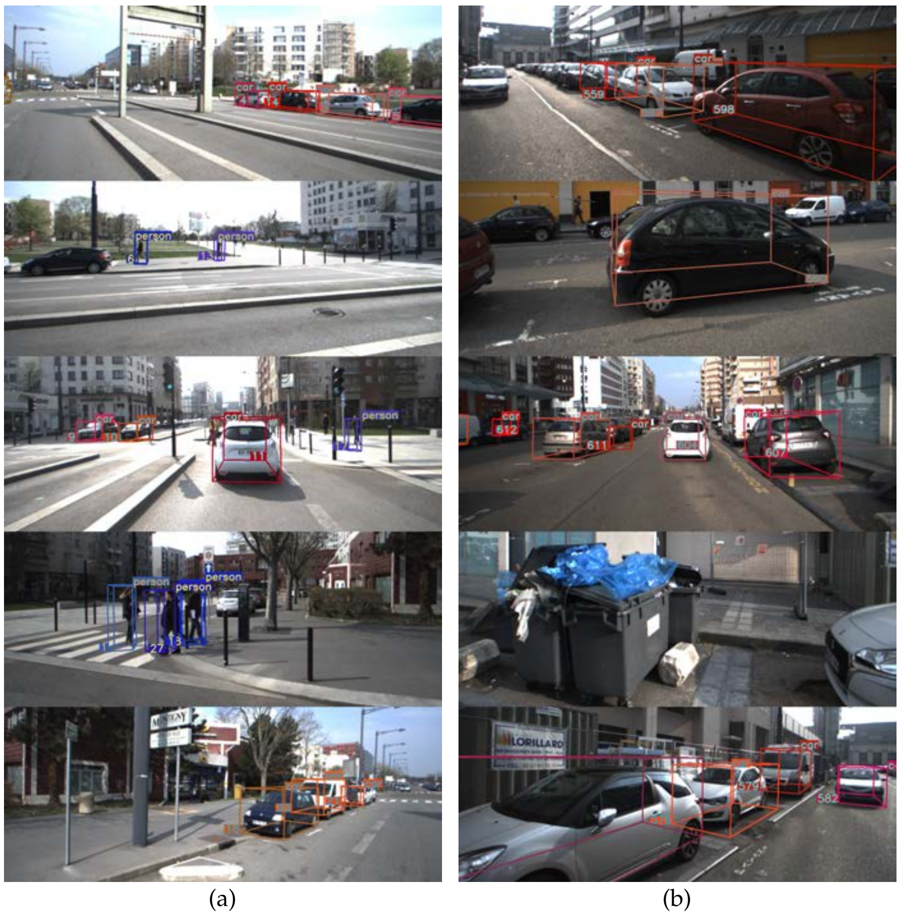


Figure 5.13: Qualitative results of the proposed system on some typical traffic scenarios. From top to bottom: 3D detections in rear-left, front-left, front, front-right, and rear-right cameras, and Bird's Eye View representation [11] © IEEE 2020

that compose the perception stack to maximize the usage of the processing resources available in the on-board computer. In particular, the five detection pipeline instances are scheduled to run in the different GPUs of the vehicle as detailed in Table 5.7. As the number of CUDA devices is lower than the number of instances, the processing of rear cameras is performed in the same GPU.

Taking into account the above distribution, the execution time for each of the parallel object detection processes can be analyzed. Figure 5.15 shows the average runtime for each of the stages in the pipeline during the tests. As can be observed, there exist uneven processing times for the distinct devices. This may be caused by several factors: the differences in the input resolution, whether a GPU is shared or not, the existence of other processes running in the same CPU, and the number of objects appearing in the camera FOV. The latter variable has a direct impact in the whole perception stage, as part of the execution times of both Mask R-CNN and Frustum Point-

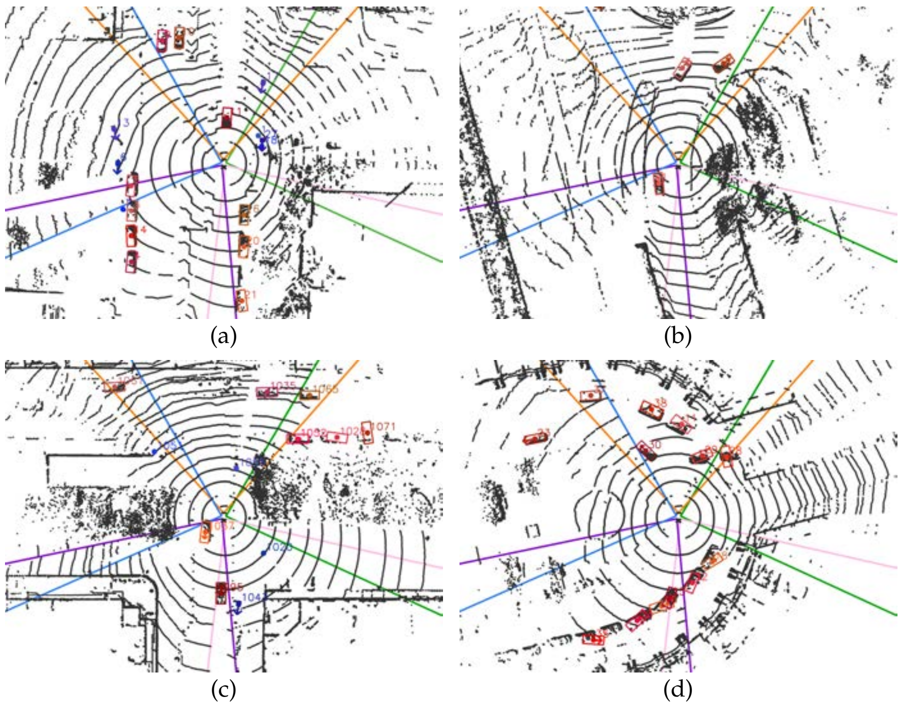


Figure 5.14: Qualitative results of the proposed system on some typical traffic scenarios. From top to bottom: 3D detections in rear-left, front-left, front, front-right, and rear-right cameras, and Bird's Eye View representation [11] © IEEE 2020

Net models depend on the scene's object density. Because of that, this factor may be the most decisive together with the sharing of GPU resources.

Despite the runtime differences, the real per-frame delay corresponds to the slowest detection instance, as 3D objects from all cameras are synchronized before the NMS stage takes place. Although this leads to some idle time in the fastest processes, it allows the tracking module to receive the detection measurements of a certain frame at once, achieving a consistent state over time.

Looking at Figure 5.15, one might argue that the proposed solution does not meet the requisites of real-time operation, typically set at 10 Hz, as the per-frame execution time lasts for around 150 ms. However, thanks to the modular implementation of the proposed solution, the different steps of the pipeline can be run in parallel so that the image detection node can process the next frame while the rest of the stages finish. Hence, the system frame rate is not bounded by the perception software, but limited to the sensor acquisition frequency.

5.3 CONCLUSION

This chapter has presented a detailed description and discussion of this thesis' proposal towards closing the domain gap of LiDAR-based

Table 5.7: Perception pipeline processes distribution

ID	Camera	Resolution	GPU
0	Front	1024x410	0
1	Left	1024x500	1
2	Right	1024x500	2
3	Rear-left	1024x520	3
4	Rear-right	1024x520	3

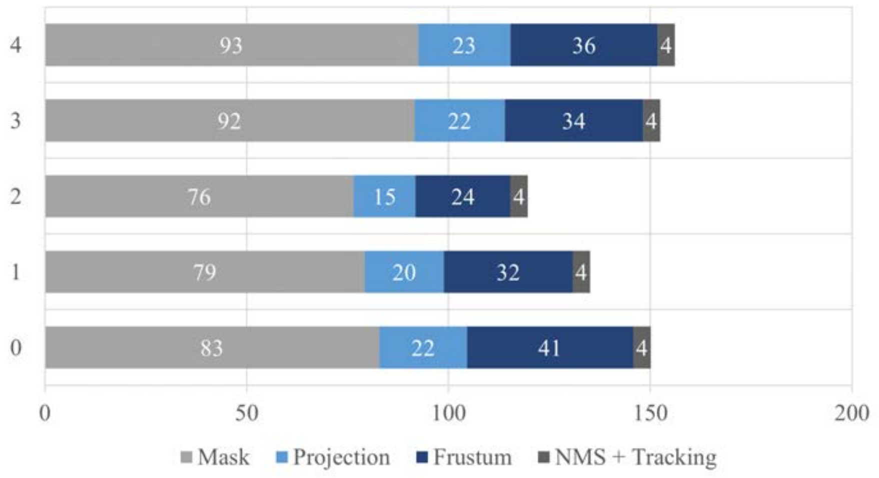


Figure 5.15: Processing times of the different stages in the pipeline, divided by camera device

detection models when trained with information from public datasets and deployed on vehicles mounting distinct sensor setups. Contrary to existing DA works, our algorithm focuses on the creation of new datasets as captured from any LiDAR device through a reconstruction of the scene using information existing benchmarks.

The experimental analysis of the proposed method for synthetic point clouds generation proves the utility of the approach to reduce the performance drop of 3D object detectors caused by the often significant differences between the training and testing domains. Conducted tests demonstrate that the solution is a leap forward on the deployment of LiDAR-based perception networks into real vehicles by allowing the usage of custom sensor configurations, which were previously limited to those used in available datasets. Moreover, the presented pipeline opens the door to the standardization of existing 3D detection benchmarks to any LiDAR sensor and may facilitate the adoption of upcoming devices in the incipient and fast-changing market of laser scanners.

The field tests further assess the validity of the method, as the current perception pipeline enables safe real-time navigation in open traffic during almost the whole duration of the rides in the operational design domain. However, from a safety perspective, a failback driver is still required to deal with corner cases.

In order to remove the need of human supervision, two lines of work have to be addressed. On the one hand, additional road participant categories should be considered while training, as certain objects in the traffic scenarios, e. g., trucks or scooters, are still unhandled, thus leading to risky situations where a switch to manual driving is mandatory. On the other hand, redundant single-modality detection frameworks relying on LiDAR or camera information should be embedded to improve the robustness of the system. Besides, to comply with the goal of having low processing times without increasing the hardware requirements, the software implementation of all stages, and particularly those related to inference networks, should be optimized.

Part III

CONCLUDING REMARKS

6 CONCLUSION AND FUTURE WORKS

This chapter contains a summary of the contributions made in this thesis. Besides, some lines of research are suggested to respond to the challenges arising from the presented work.

6.1 CONCLUSION

The widespread use of automated cars may become a reality when their driving capabilities overtake those of human operators. For this to happen, major advances are still required in localization, perception, or control technologies. Among all, scene understanding stands out as the key player in the pipeline, since a proper awareness of the traffic situation is essential for safe interactions with the road infrastructure and other participants.

In this thesis, we have focused our attention on the 3D object detection problem, proposing innovative solutions to some of the challenges found at different stages of the perception process. A significant amount of approaches has been presented to push forward the state-of-the-art in a wide variety of tasks, including multi-modal data association, object detection and classification, and deployment. The main achievements are summarized below:

- An original extrinsic calibration method to automate the calculation of the relative pose of a pair of sensors made of monocular cameras, stereo rigs, or LiDAR devices has been presented. Contrary to existing approaches, usually focused on high-resolution scanners, our work supports very diverse sensor configurations, including those mounting LiDAR units with a reduced number of layers, as long as at least two rings intersect each of the target's holes. Along the process, the human intervention is almost limited to the positioning of the calibration target inside the shared FOV of the sensors involved. The solution outperforms all previous works using calibration markers and offers better generalization capabilities than DNN-based alternatives, which require annotated training samples for each specific setup.

- We have introduced an evaluation software for calibration algorithms, which traditionally lack a formal validation procedure due to the unavailability of ground-truth measurements. The benchmark, based on a simulated environment, is able to accurately recreate the signal of an unlimited number of sensor models and custom configurations, and provides the perfect relative transformation between them so that it can be used as a reference.

- A novel encoding for the LiDAR's BEV projection has been proposed. This representation captures the key features of the original point cloud despite its discretization into a 2D structure. The image channels encode sparsity-invariant features, robust against differences among scanner specifications, enabling precise object characterization regardless the device resolution and operating range. Unlike other approaches in the literature, our top-view representation contains sufficient information to estimate 3D cuboids for objects of distinct categories.

- The adequacy of CNN architectures to exploit LiDAR features for the task of 3D object detection have been demonstrated by means of the BirdNet framework. This new pipeline leverages the feature encoders and RPNs tailored for image detectors to classify and estimate the bounding box of objects of multiple categories using the BEV as input in a single forward pass. The vanilla version, which requires a post-processing stage to compute the height and z position of the objects, outperformed all prior BEV-based approaches, although they were only focused on the detection of vehicles. With the second iteration of the method, the use of pyramid networks boosted the accuracy of the identification of small objects leading to unprecedented results for works solely based on the LiDAR's top-view projection. Besides, the learned estimation of the box parameters in the vertical axis has proven beneficial for the overall performance of the network.

- A single-stage architecture able to provide multi-class 3D detection using the information from BEV images in an end-to-end fashion has been presented. Despite the use of a light backbone and the removal of slow proposal generation branches, the network outperforms other non-RPN approaches and yields results comparable or superior to some existing two-stage frameworks. Moreover, its efficient design enables a throughput rate of nearly 50 Hz, making it an ideal solution for deployment in resource-constrained embedded computers.

- We have proposed the first single-stage network that leverages explicit LiDAR-image correspondences in the BEV space for multi-class object detection. A manifold of multi-modal fusion strategies has been evaluated, including early or sequential configurations. Among the different alternatives, a novel scheme has been introduced to exploit the joint use of both data modalities at the feature-level. The new layer aims to mimic the functionality of the ROI pooling operation of RPN at a much lower computational cost. Although the tested approaches have delivered uneven results, the comprehensive experimentation has laid a solid basis for further investigation.

- A method has been introduced to address the DA issues experienced by LiDAR-based detectors when deployed on sensor configurations significantly different from those in which they have been trained. As opposed to the popular trend which focuses on teaching the models to learn domain-invariant features, our proposal is devoted to the adaptation of currently available datasets to LiDAR units of different specifications. Through the use of the newly generated point clouds, the detection frameworks can be fine-tuned to exploit the features of any given device, allowing the network to optimize its weights to the target sensor. Besides, since the proposed solution produces a complete 3D point cloud, its usage can boost the performance of any framework based on LiDAR information, including raw-based pipelines, for which the DNN domain adaptation methods have no applicability yet.
- The performance of some of the contributions of this thesis has been assessed in complex real driving scenarios, where a moving research platform endowed with a custom sensor multimodal configuration spanning the whole horizontal FOV has operated autonomously in open traffic. The deployed perception stack proved its suitability to infer accurate 3D positioning of other road users to the control modules, enabling the navigation of the vehicle in a variety of traffic situations.

These contributions have made possible the accomplishment of the objectives set for this thesis and have led to the publication of 3 journals and 5 conference articles. In addition, 9 other papers related to complementary on-board perception tasks have been issued.

6.2 FUTURE WORK

Although the different works presented in this thesis have led to notable advances in the field of 3D object detection, results showed that there is still room for improvement. In the following lines, some ideas are drawn towards addressing some of the open questions and limitations of the proposed approaches:

- Extending the extrinsic calibration software with an outlier rejection scheme might be useful to discard spurious samples obtained in the reference point extraction procedure. At this point, accurate modeling of the sensor noise could be convenient, which will also enable adapting the parameter settings to each particular device. On the other hand, the proposed method has been designed to determine a fixed set of extrinsic parameters before the perception system is deployed; however, sensor setups mounted in movable platforms, such as autonomous vehicles, can suffer miscalibrations during regular operation. The use of the proposed method would require the ability to detect these situations early, prompting the user to perform a recalibration when necessary.

- To further enhance the representation capabilities of the proposed BEV, the inclusion of additional channels that complement the current encoding may be studied, such as occupancy information or context data that leverages the features of points in the vicinities of the cell.

- The accuracy of BirdNet framework, like other BEV-based methods, still falls behind other approaches that make use of raw LiDAR information as input. To close the performance gap with point cloud networks, the use of the original 3D points in the box regression phase may be explored. For efficiency reasons, the estimation of the object candidates should remain in the top-view projection.

- The uneven performance among the different classes in the single-stage detector may be addressed by means of appending heads on top of feature maps of multiple resolutions. Moreover, different augmentation techniques could help reduce the network bias, such as random shifts and rotations, noise addition, or artificially incrementing the number of instances of less represented categories [194].

- Different alternatives may be explored to boost the effectiveness of the evaluated fusion schemes. On the one hand, dividing the cell pillars into a set of vertical slices may improve the accuracy of early and sequential strategies, as an increased resolution in the z axis will reduce the height of each cell, thus mitigating the inaccuracies caused by color blurring and misprojections. On the other hand, the feature-level fusion operation may take advantage of both a multi-scale indexing procedure and an auxiliary loss to train the image stream as a 2d object detector. Through the former, different receptive fields will provide flexibility to detect objects of distinct sizes, while the latter will make all positions in the RGB stream contribute to the final loss, forcing the image backbone to encode meaningful feature maps.

- The high quality of the simulated LiDAR scans may be improved by the incorporation of a reflectivity channel. To this end, an interpolation of the intensity values of the nearest beams could be used. Additionally, an image-based pose estimation network may be used to refine the accumulation of clouds belonging to deformable objects like pedestrians or cyclists. By identifying the position of the body parts, the corresponding LiDAR readings can be rearranged and aligned to the object position in the target frame, eliminating the often noisy surface modeling of the current approach. Besides, in view of the rapid evolution of monocular depth prediction networks, the scene reconstruction stage might be fed with image-based point clouds so the proposed solution can be used to add the LiDAR modality to RGB datasets.

In the next decades, scientific advances in the field of scene understanding will set the pace for the development of autonomous driving systems, where research on artificial intelligence will play a crucial role in achieving a beyond-human performance. We hope that this thesis represents a step forward in the creation of future self-driving cars perception technologies.

BIBLIOGRAPHY

[1] AAA Foundation for Traffic Safety, *New american driving survey: Updated methodology and results from july 2019 to june 2020*. [Online]. Available: https://aaafoundation.org/wp-content/uploads/2021/04/New-American-Driving-Survey-Report-April-2021-1.pdf (visited on 11/11/2021).

[2] K. M. Ahmad Yousef, B. J. Mohd, K. Al-Widyan, and T. Hayajneh, "Extrinsic calibration of camera and 2d laser sensors without overlap," *Sensors*, vol. 17, no. 10, p. 2346, 2017.

[3] S. Anenberg, J. Miller, D. Henze, and R. Minjares, *A global snapshot of the air pollution-related health impacts of transportation sector emissions in 2010 and 2015*, The International Council on Clean Transportation. [Online]. Available: https://theicct.org/publications/health-impacts-transport-emissions-2010-2015 (visited on 11/08/2021).

[4] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A Survey on 3D Object Detection Methods for Autonomous Driving Applications," *IEEE Transactions on Intelligent Transportation Systems (In press)*, 2019.

[5] A. Astudillo, N. Molina, I. Cortés, *et al.*, "Visibility-aware adaptative speed planner for human-like navigation in roundabouts," in *Proc. of the 2021 IEEE International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2021, pp. 885–890. DOI: 10.1109/ITSC48978.2021.9564451.

[6] I. Barabanau, A. Artemov, E. Burnaev, and V. Murashkin, "Monocular 3d object detection via geometric reasoning on keypoints," pp. 652–659, 2020.

[7] A. Barrera, J. Beltrán, C. Guindel, J. A. Iglesias, and F. García, "Cycle and semantic consistent adversarial domain adaptation for reducing simulation-to-real domain shift in lidar bird's eye view," in *Proc. of the 2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, 2021, pp. 3081–3086. DOI: 10.1109/ITSC48978.2021.9564553.

[8] A. Barrera, J. Beltrán, C. Guindel, J. A. Iglesias, and F. García, "BirdNet+: Two-Stage 3D Object Detection in LiDAR Through a Sparsity-Invariant Bird's Eye View," *IEEE Access*, vol. 9, pp. 160 299–160 316, 2021. DOI: 10.1109/ACCESS.2021.3131389.

[9] A. Barrera, C. Guindel, J. Beltrán, and F. García, “Birdnet+: End-to-end 3d object detection in lidar bird’s eye view,” in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2020, pp. 2985–2990. DOI: 10.1109/ITSC45102.2020.9294293.

[10] J. Beltrán, I. Cortés, A. Barrera, J. Urdiales, C. Guindel, F. García, and A. de la Escalera, “A method for synthetic lidar generation to create annotated datasets for autonomous vehicles perception,” in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 1091–1096. DOI: 10.1109/ITSC.2019.8917176.

[11] J. Beltrán, C. Guindel, I. Cortés, A. Barrera, A. Astudillo, J. Urdiales, M. Álvarez, F. Bekka, V. Milanés, and F. García, “Towards autonomous driving: A multi-modal 360° perception proposal,” in *Proc. of the 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2020, pp. 3295–3300. DOI: 10.1109/ITSC45102.2020.9294494.

[12] 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 *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3517–3523. DOI: 10.1109/ITSC.2018.8569311.

[13] J. Beltrán, C. Jaraquemada, B. Musleh, A. de la Escalera, and J. M. Armingol, “Dense semantic stereo labelling architecture for in-campus navigation.,” in *VISIGRAPP (5: VISAPP)*, 2017, pp. 266–273. DOI: 10.5220/0006131602660273.

[14] J. Beltrán, C. Guindel, A. De la Escalera, and F. García, “Automatic Extrinsic Calibration Method for LiDAR and Camera Sensor Setups,” *IEEE Transactions on Intelligent Transportation Systems*, 2022, [Under review]. arXiv: 2101.04431.

[15] R. Benenson, M. Omran, J. Hosang, and B. Schiele, “Ten years of pedestrian detection, what have we learned?” In *European Conference on Computer Vision*, Springer, 2014, pp. 613–627.

[16] K. Bengler, K. Dietmayer, B. Farber, M. Maurer, C. Stiller, and H. Winner, “Three decades of driver assistance systems: Review and future perspectives,” *IEEE Intelligent transportation systems magazine*, vol. 6, no. 4, pp. 6–22, 2014.

[17] M. Bertozzi, L. Bombini, A. Broggi, M. Buzzoni, E. Cardarelli, S. Cattani, P. Cerri, A. Coati, S. Debattisti, A. Falzoni, *et al.*, “Viac: An out of ordinary experiment,” in *2011 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2011, pp. 175–180.

[18] L. C. L. Bianco, A. Al-Kaff, J. Beltrán, F. G. Fernández, and G. F. López, "Joint instance segmentation of obstacles and lanes using convolutional neural networks," in *Iberian Robotics conference*, Springer, 2019, pp. 229–241. DOI: 10.1007/978-3-030-35990-4_19.

[19] L. C. L. Bianco, J. Beltran, G. F. López, F. Garcia, and A. Al-Kaff, "Joint semantic segmentation of road objects and lanes using convolutional neural networks," *Robotics and Autonomous Systems*, vol. 133, p. 103 623, 2020. DOI: 10.1016/J.ROBOT.2020.103623.

[20] S. Boehm, K. Lebling, K. Levin, H. Fekete, J. Jaeger, R. Waite, A. Nilsson, J. Thwaites, R. Wilson, A. Geiges, *et al.*, *State of climate action 2021: Systems transformations required to limit global warming to 1.5°c*, World Resources Institute, 2021.

[21] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," 2016. arXiv: 1604.07316.

[22] D. Braid, A. Broggi, and G. Schmiedel, "The terramax autonomous vehicle concludes the 2005 darpa grand challenge," in *2006 IEEE Intelligent Vehicles Symposium*, IEEE, 2006, pp. 534–539.

[23] G. Brazil and X. Liu, "M3d-rpn: Monocular 3d region proposal network for object detection," in *Proc. of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9287–9296.

[24] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, D. Molinari, M. Panciroli, and A. Prioletti, "Proud—public road urban driverless-car test," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 6, pp. 3508–3519, 2015.

[25] G. J. Brostow, J. Fauqueur, and R. Cipolla, "Semantic object classes in video: A high-definition ground truth database," *Pattern Recognition Letters*, vol. 30, no. 2, pp. 88–97, 2009.

[26] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 11 621–11 631.

[27] J. Castorena, U. S. Kamilov, and P. T. Boufounos, "Autocalibration of lidar and optical cameras via edge alignment," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2016, pp. 2862–2866.

[28] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teuliere, and T. Chateau, “Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2040–2049.

[29] C. Chen, L. Z. Fragonara, and A. Tsourdos, “Roifusion: 3d object detection from lidar and vision,” *IEEE Access*, vol. 9, pp. 51 710–51 721, 2021.

[30] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, “Multi-view 3d object detection network for autonomous driving,” in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[31] Y. Chen, L. Tai, K. Sun, and M. Li, “Monopair: Monocular 3d object detection using pairwise spatial relationships,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 12 093–12 102.

[32] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 1251–1258.

[33] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele, “The cityscapes dataset for semantic urban scene understanding,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 3213–3223.

[34] C. Cortes and V. Vapnik, “Support-vector networks,” *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[35] I. Cortés, J. Beltrán, A. de la Escalera, and F. García, “Sianms: Non-maximum suppression with siamese networks for multi-camera 3d object detection,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2020, pp. 933–938. DOI: 10.1109/IV47402.2020.9304685.

[36] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 1, 2005, pp. 886–893.

[37] S. Debattisti, L. Mazzei, and M. Panciroli, “Automated extrinsic laser and camera inter-calibration using triangular targets,” in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, 2013, pp. 696–701, ISBN: 9781467327558.

[38] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 248–255.

[39] A. Dhall, K. Chelani, V. Radhakrishnan, and K. Krishna, "LiDAR-camera calibration using 3D-3D point correspondences," 2017. arXiv: 1705.09785.

[40] E. D. Dickmanns, R. Behringer, D. Dickmanns, T. Hildebrandt, M. Maurer, F. Thomanek, and J. Schiehlen, "The seeing passenger car'vamors-p'," in *Proc. of the Intelligent Vehicles' 94 Symposium*, IEEE, 1994, pp. 68–73.

[41] M. Ding, Y. Huo, H. Yi, Z. Wang, J. Shi, Z. Lu, and P. Luo, "Learning depth-guided convolutions for monocular 3d object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 1000–1001.

[42] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An Open Urban Driving Simulator," in *Proc. Conference on Robot Learning (CoRL)*, 2017, pp. 1–16.

[43] G. N. Doval, A. Al-Kaff, J. Beltrán, F. G. Fernández, and G. F. López, "Traffic sign detection and 3d localization via deep convolutional neural networks and stereo vision," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2019, pp. 1411–1416. DOI: 10.1109/ITSC.2019.8916958.

[44] *Estimated road traffic death rate (per 100,00 population)*, World Health Organization, 2018. [Online]. Available: https://www.who.int/data/gho/data/themes/topics/sdg-target-3_6-road-traffic-injuries (visited on 12/15/2021).

[45] Eurostat, *Road accident fatalities, eu. statistics by type of vehicle*. [Online]. Available: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=Road_accident_fatalities_-_statistics_by_type_of_vehicle (visited on 11/21/2021).

[46] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[47] A. Farhadi and J. Redmon, "Yolov3: An incremental improvement," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Springer Berlin/Heidelberg, Germany, 2018, pp. 1804–2767.

[48] Federal Highway Administration, *Traffic congestion and reliability: Trends and advanced strategies for congestion mitigation*. [Online]. Available: https://ops.fhwa.dot.gov/congestion_report/congestion_report_05.pdf (visited on 11/09/2021).

[49] L. Fei-Fei, R. Fergus, and P. Perona, “Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories,” in *2004 conference on computer vision and pattern recognition workshop*, IEEE, 2004, pp. 178–178.

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

[51] U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M. Enzweiler, F. Stein, and R. G. Herrtwich, “Making Bertha see,” in *Proc. IEEE International Conference on Computer Vision (ICCV) Workshops*, 2013, pp. 214–221, ISBN: 9781479930227.

[52] Y. Freund and R. E. Schapire, “A decision-theoretic generalization of on-line learning and an application to boosting,” *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[53] K. Fukushima, “Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position,” *Biological Cybernetics*, vol. 36, no. 4, pp. 193–202, 1980.

[54] N. Gählert, M. Mayer, L. Schneider, U. Franke, and J. Denzler, “Mb-net: Mergeboxes for real-time 3d vehicles detection,” in *2018 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2018, pp. 2117–2124.

[55] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig, “Virtual Worlds as Proxy for Multi-Object Tracking Analysis,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4340–4349.

[56] C. Gao and J. R. Spletzer, “On-line calibration of multiple lidars on a mobile vehicle platform,” in *2010 IEEE International Conference on Robotics and Automation*, IEEE, 2010, pp. 279–284.

[57] F. García, A. de la Escalera, and J. M. Armingol, “Novel method for vehicle and pedestrian detection based on information fusion,” in *International Technology Robotics Applications*, Springer, 2014, pp. 79–88.

[58] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez, “Automatic generation and detection of highly reliable fiducial markers under occlusion,” *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014.

[59] A. Geiger, M. Lauer, F. Moosmann, B. Ranft, H. Rapp, C. Stiller, and J. Ziegler, "Team annieway's entry to the 2011 grand cooperative driving challenge," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1008–1017, 2012.

[60] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013, ISSN: 0278-3649.

[61] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the KITTI Vision Benchmark Suite," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3354–3361.

[62] A. Geiger, F. Moosmann, Ö. Car, and B. Schuster, "Automatic camera and range sensor calibration using a single shot," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 3936–3943, ISBN: 978-1-4673-1405-3.

[63] R. Girshick, "Fast R-CNN," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[64] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[65] ——, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142–158, 2015.

[66] Global Health Observatory (GHO), *Global burden of disease study 2017 (gbd 2017)*, World Health Organization. [Online]. Available: `http://ghdx.healthdata.org/gbd-results-tool` (visited on 11/08/2021).

[67] "Global status report on road safety 2018," World Health Organization, 2018.

[68] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proc. International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy, 2010, pp. 249–256.

[69] C. Guindel, J. Beltrán, D. Martín, and F. García, "Automatic extrinsic calibration for lidar-stereo vehicle sensor setups," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 674–679. DOI: `10.1109/ITSC.2017.8317829`.

[70] C. Guindel, D. Martin, and J. M. Armingol, "Joint object detection and viewpoint estimation using CNN features," in *Proc. IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 2017, pp. 145–150.

[71] K. He, R. Girshick, and P. Dollar, "Rethinking imagenet pre-training," in *Proc. of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[72] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.

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

[74] M. He, H. Zhao, F. Davoine, J. Cui, and H. Zha, "Pairwise lidar calibration using multi-type 3d geometric features in natural scene," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 1828–1835.

[75] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[76] H. Hirschmüller, "Stereo processing by semiglobal matching and mutual information," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 30, no. 2, pp. 328–341, 2008.

[77] V.-D. Hoang, L. Kurnianggoro, K.-H. Jo, *et al.*, "Scalable histogram of oriented gradients for multi-size car detection," in *2014 10th France-Japan/8th Europe-Asia Congress on Mecatronics (MECATRONICS2014-Tokyo)*, IEEE, 2014, pp. 228–231.

[78] C. Hoppe and S. Krömker, "Adaptive meshing and detail-reduction of 3d-point clouds from laser scans," *International archives of photogrammetry, remote sensing and spatial information sciences*, vol. 38, p. 5, 2009.

[79] J. Huang, V. Rathod, C. Sun, *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3296–3305.

[80] T. Huang, Z. Liu, X. Chen, and X. Bai, "Epnet: Enhancing point features with image semantics for 3d object detection," in *European Conference on Computer Vision*, Springer, 2020, pp. 35–52.

[81] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *The Journal of physiology*, vol. 148, no. 3, pp. 574–591, 1959.

[82] Institute for Health Metrics and Evaluation (IHME), *Public health and environment dataset*, World Health Organization, 2018. [Online]. Available: https://www.who.int/data/gho/data/themes/public-health-and-environment/GHO/public-health-and-environment (visited on 11/08/2021).

[83] S. International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *SAE*, 2021.

[84] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, PMLR, 2015, pp. 448–456.

[85] G. Iyer, R. K. Ram, J. K. Murthy, and K. M. Krishna, "CalibNet: Self-supervised extrinsic calibration using 3D spatial transformer networks," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1110–1117.

[86] J. Jiao, Q. Liao, Y. Zhu, T. Liu, Y. Yu, R. Fan, L. Wang, and M. Liu, "A novel dual-lidar calibration algorithm using planar surfaces," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, 2019, pp. 1499–1504.

[87] L. Jiao, F. Zhang, F. Liu, S. Yang, L. Li, Z. Feng, and R. Qu, "A survey of deep learning-based object detection," *IEEE access*, vol. 7, pp. 128 837–128 868, 2019.

[88] A. G. Kashani, M. J. Olsen, C. E. Parrish, and N. Wilson, "A review of lidar radiometric processing: From ad hoc intensity correction to rigorous radiometric calibration," *Sensors*, vol. 15, no. 11, pp. 28 099–28 128, 2015.

[89] C. G. Keller, M. Enzweiler, and D. M. Gavrila, "A new benchmark for stereo-based pedestrian detection," in *2011 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2011, pp. 691–696.

[90] R. Kesten, M. Usman, J. Houston, *et al.*, *Lyft level 5 av dataset 2019*, urlhttps://level5.lyft.com/dataset/, 2019.

[91] S. Kim, J. Choi, T. Kim, and C. Kim, "Self-training and adversarial background regularization for unsupervised domain adaptive one-stage object detection," in *Proc. of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6092–6101.

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

[93] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2004, pp. 2149–2154.

[94] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, pp. 1097–1105, 2012.

[95] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. Waslander, "Joint 3D proposal generation and object detection from view aggregation," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 5750–5757.

[96] J. Ku, A. D. Pon, and S. L. Waslander, "Monocular 3d object detection leveraging accurate proposals and shape reconstruction," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 11 867–11 876.

[97] H. Kuang, B. Wang, J. An, M. Zhang, and Z. Zhang, "Voxel-FPN: Multi-scale voxel feature aggregation for 3D object detection from LIDAR point clouds," *Sensors*, vol. 20, no. 3, p. 704, 2020.

[98] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "PointPillars: Fast encoders for object detection from point clouds," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 697–12 705.

[99] M. Lauer, "Grand cooperative driving challenge 2011 [its events]," *IEEE Intelligent Transportation Systems Magazine*, vol. 3, no. 3, pp. 38–40, 2011.

[100] H. Law and J. Deng, "Cornernet: Detecting objects as paired keypoints," in *Proc. of the European conference on computer vision (ECCV)*, 2018, pp. 734–750.

[101] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[102] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers," in *Robotics: Science and Systems*, 2013, ISBN: 9789810739379.

[103] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang, "Gs3d: An efficient 3d object detection framework for autonomous driving," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 1019–1028.

[104] P. Li, H. Zhao, P. Liu, and F. Cao, "Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16*, Springer, 2020, pp. 644–660.

[105] Y. Li, Y. Ruichek, and C. Cappelle, "3D triangulation based extrinsic calibration between a stereo vision system and a LIDAR," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2011, pp. 797–802, ISBN: 9781457721984.

[106] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 7345–7353.

[107] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Proc. of the European Conference on Computer Vision (ECCV)*, 2018, pp. 641–656.

[108] Z. Liang, M. Zhang, Z. Zhang, X. Zhao, and S. Pu, "RangeRCNN: Towards fast and accurate 3D object detection with range image representation," 2020. arXiv: 2009.00206.

[109] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature Pyramid Networks for object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2117–2125.

[110] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[111] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*, Springer, 2014, pp. 740–755.

[112] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*, Springer, 2016, pp. 21–37.

[113] Z. Liu, Z. Wu, and R. Tóth, "Smoke: Single-stage monocular 3d object detection via keypoint estimation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020, pp. 996–997.

[114] Z. Liu, X. Zhao, T. Huang, R. Hu, Y. Zhou, and X. Bai, "TANet: Robust 3D object detection from point clouds with triple attention," in *Proc. AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 11 677–11 684.

[115] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[116] S. Luo, H. Dai, L. Shao, and Y. Ding, "M3dssd: Monocular 3d single stage object detector," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021, pp. 6145–6154.

[117] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan, "Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving," in *Proc. of the IEEE International Conference on Computer Vision*, 2019, pp. 6851–6860.

[118] P. Marın-Plaza, J. Beltrán, A. Hussein, B. Musleh, D. Martın, A. de la Escalera, and J. M. Armingol, "Stereo vision-based local occupancy grid map for autonomous navigation in ros," in *11th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications VISIGRAPP*, vol. 2016, 2016, pp. 701–706. DOI: 10.5220/0005787007010706.

[119] E. Marti, M. A. de Miguel, F. Garcia, and J. Perez, "A review of sensor technologies for perception in automated driving," *IEEE Intelligent Transportation Systems Magazine*, vol. 11, no. 4, pp. 94–108, 2019.

[120] G. P. Meyer, A. Laddha, E. Kee, C. Vallespi-Gonzalez, and C. K. Wellington, "LaserNet: An efficient probabilistic 3D object detector for autonomous driving," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 677–12 686.

[121] V. Milanes, D. Gonzalez, F. Navas, I. Mahtout, A. Armand, C. Zinoune, A. Ramaswamy, F. Bekka, N. Molina, E. Battesti, *et al.*, "The tornado project: An automated driving demonstration in peri-urban and rural areas," *IEEE Intelligent Transportation Systems Magazine*, pp. 2–18, 2021, ISSN: 1941-1197. DOI: 10.1109/MITS.2021.3068067.

[122] G. A. Miller, "Wordnet: A lexical database for english," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[123] S. Mishra, P. R. Osteen, G. Pandey, and S. Saripalli, "Experimental evaluation of 3d-lidar camera extrinsic calibration," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 9020–9026.

[124] P. Moghadam, M. Bosse, and R. Zlot, "Line-based extrinsic calibration of range and image sensors," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2013, pp. 3685–3691, ISBN: 9781467356411.

[125] T. Möller and B. Trumbore, "Fast, minimum storage ray/triangle intersection," in *ACM SIGGRAPH 2005 Courses*, ACM, 2005, p. 7.

[126] B. Musleh, J. Beltrán, C. Jaraquemada, M. J. Gómez-Silva, N. Hernández, and J. M. Armingol, "Autocalibración de parámetros extrínsecos de sistemas estéreo para aplicaciones de tráfico," in *XXXVII Jornadas de Automática*, 2016, pp. 675–682.

[127] A. Naiden, V. Paunescu, G. Kim, B. Jeon, and M. Leordeanu, "Shift r-cnn: Deep monocular 3d object detection with closed-form geometric constraints," in *2019 IEEE International Conference on Image Processing (ICIP)*, IEEE, 2019, pp. 61–65.

[128] Observatorio Nacional de Seguridad Vial, "Las principales cifras de siniestralidad vial 2019," 2020.

[129] K.-S. Oh and K. Jung, "Gpu implementation of neural networks," *Pattern Recognition*, vol. 37, no. 6, pp. 1311–1314, 2004.

[130] U. Ozguner, K. A. Redmill, and A. Broggi, "Team terramax and the darpa grand challenge: A general overview," in *IEEE Intelligent Vehicles Symposium*, IEEE, 2004, pp. 232–237.

[131] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information," in *Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[132] G. Pandey, J. R. McBride, S. Savarese, and R. M. Eustice, "Automatic extrinsic calibration of vision and lidar by maximizing mutual information," *Journal of Field Robotics*, vol. 32, no. 5, pp. 696–722, 2015.

[133] S. Pang, D. Morris, and H. Radha, "Clocs: Camera-lidar object candidates fusion for 3d object detection," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2020, pp. 10 386–10 393.

[134] C. P. Papageorgiou and T. Poggio, "A trainable object detection system: Car detection in static images," 1999.

[135] C. Park, P. Moghadam, S. Kim, S. Sridharan, and C. Fookes, "Spatiotemporal camera-lidar calibration: A targetless and structureless approach," *IEEE Robotics Automation Letters*, vol. 5, no. 2, pp. 1556–1563, 2020.

[136] Y. Park, S. Yun, C. S. Won, K. Cho, K. Um, and S. Sim, "Calibration between color camera and 3D LIDAR instruments with a polygonal planar board," *Sensors*, vol. 14, no. 3, pp. 5333–5353, 2014, ISSN: 14248220.

[137] M. Pereira, D. Silva, V. Santos, and P. Dias, "Self calibration of multiple LIDARs and cameras on autonomous vehicles," *Robotics Autonomous Systems*, vol. 83, pp. 326–337, 2016, ISSN: 09218890.

[138] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," Carnegie-Mellon University, Tech. Rep., 1989.

[139] D. Pomerleau, "Ralph: Rapidly adapting lateral position handler," in *Proc. of the Intelligent Vehicles' 95. Symposium*, IEEE, 1995, pp. 506–511.

[140] Z. Pusztai and L. Hajder, "Accurate calibration of LiDAR-camera systems using ordinary boxes," in *IEEE International Conference on Computer Vision (ICCV) Workshops*, 2017, pp. 394–402.

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

[142] Z. Qin, J. Wang, and Y. Lu, "Monogrnet: A geometric reasoning network for monocular 3d object localization," in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 33, 2019, pp. 8851–8858.

[143] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[144] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7263–7271.

[145] 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.

[146] S. R. Richter, V. Vineet, S. Roth, and V. Koltun, "Playing for data: Ground truth from computer games," in *European conference on computer vision*, Springer, 2016, pp. 102–118.

[147] H. Ritchie and M. Roser, *Causes of death*, Our World in Data, 2018. [Online]. Available: https://ourworldindata.org/causes-of-death.

[148] C. H. Rodríguez Garavito, A. Ponz, F. García, D. Martín, A. de la Escalera, and J. M. Armingol, "Automatic laser and camera extrinsic calibration for data fusion using road plane," in *IEEE International Conference on Information Fusion (FUSION)*, 2014.

[149] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. Lopez, "The SYNTHIA Dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[150] S. Royo and M. Ballesta-Garcia, "An overview of lidar imaging systems for autonomous vehicles," *Applied Sciences*, vol. 9, no. 19, p. 4093, 2019.

[151] D. Scaramuzza, A. Harati, and R. Siegwart, "Extrinsic self calibration of a camera and a 3D laser range finder from natural scenes," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2007, pp. 4164–4169, ISBN: 1424409128.

[152] T. Scharwächter, M. Enzweiler, U. Franke, and S. Roth, "Efficient multi-cue scene segmentation," in *German Conference on Pattern Recognition*, Springer, 2013, pp. 435–445.

[153] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "RegNet: Multimodal sensor registration using deep neural networks," in *Proc. IEEE Intelligent Vehicles Symp. (IV)*, 2017, pp. 1803–1810.

[154] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*, Springer, 2018, pp. 621–635.

[155] A. Shashua, Y. Gdalyahu, and G. Hayun, "Pedestrian detection for driving assistance systems: Single-frame classification and system level performance," in *IEEE Intelligent Vehicles Symposium, 2004*, IEEE, 2004, pp. 1–6.

[156] S. Shi, X. Wang, and H. Li, "PointRCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 770–779.

[157] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3D object detection from point cloud with part-aware and part-aggregation network," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. In Press, 2020.

[158] K. Shin, Y. P. Kwon, and M. Tomizuka, "Roarnet: A robust 3d object detection based on region approximation refinement," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 2510–2515.

[159] P. Y. Simard, D. Steinkrau, and I. Buck, "Using gpus for machine learning algorithms," in *Proc.. Eighth International Conference on Document Analysis and Recognition*, IEEE Computer Society, 2005, pp. 1115–1119.

[160] M. Simon, S. Milz, K. Amende, and H.-M. Gross, "Complex-YOLO: An Euler-region-proposal for real-time 3D object detection on point clouds," in *European Conference on Computer Vision (ECCV) Workshops*, 2018, pp. 197–209.

[161] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. of the International Conference on Learning Representations (ICLR)*, 2015.

[162] T. T. Son and S. Mita, "Car detection using multi-feature selection for varying poses," in *2009 IEEE intelligent vehicles symposium*, IEEE, 2009, pp. 507–512.

[163] I. Stamos, L. Liu, C. Chen, G. Wolberg, G. Yu, and S. Zokai, "Integrating automated range registration with multiview geometry for the photorealistic modeling of large-scale scenes," *International Journal of Computer Vision*, vol. 78, no. 2, pp. 237–260, 2008.

[164] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, *et al.*, "Scalability in perception for autonomous driving: An open dataset benchmark," 2019. arXiv: 1912.04838.

[165] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[166] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2818–2826.

[167] R. Szeliski, *Computer vision: algorithms and applications*, 1st. New York, New York, USA: Springer, 2010, ISBN: 1848829353.

[168] L. Tamas, R. Frohlich, and Z. Kato, "Relative pose estimation and fusion of omnidirectional and lidar cameras," in *Computer Vision - ECCV 2014 Workshops*, 2014, pp. 640–651.

[169] Z. Taylor and J. Nieto, "Motion-based calibration of multimodal sensor extrinsics and timing offset estimation," *IEEE Transactions on Robotics*, vol. 32, no. 5, pp. 1215–1229, 2016.

[170] Texas A&M Transportation Institute, *2021 urban mobility report*. [Online]. Available: https://static.tti.tamu.edu/tti.tamu.edu/documents/mobility-report-2021.pdf (visited on 11/09/2021).

[171] The Mathworks, Inc. (). MATLAB Documentation: Lidar-camera calibration, [Online]. Available: https://www.mathworks.com/help/lidar/lidarcameracalibration.html (visited on 12/30/2020).

[172] S. Thrun, "Winning the darpa grand challenge," in *European Conference on Machine Learning*, Springer, 2006, pp. 4–4.

[173] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, *et al.*, "Stanley: The robot that won the darpa grand challenge," *Journal of field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[174] J. Uhrig, N. Schneider, L. Schneider, U. Franke, T. Brox, and A. Geiger, "Sparsity invariant cnns," in *International Conference on 3D Vision (3DV)*, 2017.

[175] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[176] S. Umeyama, "Least-squares estimation of transformation parameters between two point patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 4, pp. 376–380, 1991.

[177] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[178] ——, "Autonomous driving in urban environments: Boss and the urban challenge," in *The DARPA urban challenge*, Springer, 2009, pp. 1–59.

[179] R. Van der Merwe and E. A. Wan, "The square-root unscented Kalman filter for state and parameter-estimation," in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001, pp. 3461–3464.

[180] J. Vargas, S. Alsweiss, O. Toker, R. Razdan, and J. Santos, "An overview of autonomous vehicles sensors and their vulnerability to weather conditions," *Sensors*, vol. 21, no. 16, p. 5397, 2021.

[181] M. Velas, M. Spanel, Z. Materna, and A. Herout, "Calibration of RGB camera with Velodyne LiDAR," in *Comm. Papers Proc. International Conference on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2014, pp. 135–144, ISBN: 978-80-86943-71-8.

[182] S. Verma, J. S. Berrio, S. Worrall, and E. Nebot, "Automatic extrinsic calibration between a camera and a 3D lidar using 3D point and plane correspondences," in *Proc. IEEE Intelligent Transportation Systems Conference (ITSC)*, 2019, pp. 3906–3912.

[183] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," *International Journal of Computer Vision*, vol. 63, no. 2, pp. 153–161, 2005.

[184] P. Viola and M. Jones, “Rapid object detection using a boosted cascade of simple features,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 1, 2001, pp. 511–518.

[185] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, “Pointpainting: Sequential fusion for 3d object detection,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 4604–4612.

[186] E. A. Wan and R. Van Der Merwe, “The unscented Kalman filter for nonlinear estimation,” in *Proc. of the IEEE Adaptive Systems for Signal Processing, Communications, and Control Symposium*, 2000, pp. 153–158.

[187] C.-H. Wang, H.-W. Chen, and L.-C. Fu, “Vpfnet: Voxel-pixel fusion network for multi-class 3d object detection,” 2021. arXiv: 2111.00966.

[188] G. Wang, B. Tian, Y. Ai, T. Xu, L. Chen, and D. Cao, “CenterNet3D: An anchor free object detector for autonomous driving,” *IEEE Transactions on Intelligence Transportation Systems*, In Press.

[189] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger, “Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 8445–8453.

[190] Z. Wang and K. Jia, “Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection,” in *IROS*, IEEE, 2019.

[191] S. Wirges, S. Ding, and C. Stiller, “Single-stage object detection from top-view grid maps on custom sensor setups,” in *2020 IEEE Intelligent Vehicles Symposium (IV)*, 2020, pp. 939–944.

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

[193] B. Xu and Z. Chen, “Multi-level fusion based 3d object detection from monocular images,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 2345–2353.

[194] J. Xu, R. Zhang, J. Dou, Y. Zhu, J. Sun, and S. Pu, “Rpvnet: A deep and efficient range-point-voxel fusion network for lidar point cloud segmentation,” in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2021, pp. 16 024–16 033.

[195] S. Xu, D. Zhou, J. Fang, J. Yin, Z. Bin, and L. Zhang, "Fusionpainting: Multimodal fusion with adaptive attention for 3d object detection," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, IEEE, 2021, pp. 3047–3054.

[196] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[197] B. Yang, W. Luo, and R. Urtasun, "PIXOR: Real-time 3D object detection from point clouds," in *Proc. of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7652–7660.

[198] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "IPOD: Intensive point-based object detector for point cloud," 2018. arXiv: 1812.05276.

[199] ——, "STD: Sparse-to-dense 3d object detector for point cloud," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 1951–1960.

[200] T. Yin, X. Zhou, and P. Krähenbühl, "Multimodal virtual point 3d detection," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[201] J. H. Yoo, Y. Kim, J. Kim, and J. W. Choi, "3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16*, Springer, 2020, pp. 720–736.

[202] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell, "Bdd100k: A diverse driving dataset for heterogeneous multitask learning," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 2636–2645.

[203] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*, Springer, 2014, pp. 818–833.

[204] H. Zhang, D. Yang, E. Yurtsever, K. A. Redmill, and Ü. Özgüner, "Faraway-Frustum: Dealing with lidar sparsity for 3D object detection using fusion," pp. 2646–2652, 2021.

[205] Y. Zhang, P. David, and B. Gong, "Curriculum domain adaptation for semantic segmentation of urban scenes," in *Proc. of the IEEE international conference on computer vision*, 2017, pp. 2020–2030.

[206] Y. Zhang, Z. Xiang, C. Qiao, and S. Chen, "Accurate and real-time object detection based on Bird's Eye View on 3D point clouds," in *Proc. International Conference on 3D Vision (3DV)*, 2019, pp. 214–221.

[207] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[208] L. Zhou, Z. Li, and M. Kaess, "Automatic extrinsic calibration of a camera and a 3D LiDAR using line and plane correspondences," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 5562–5569.

[209] Y. Zhou and O. Tuzel, "VoxelNet: End-to-end learning for point cloud based 3D object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4490–4499.

[210] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, IEEE, vol. 2, 2006, pp. 1491–1498.

[211] Y. Zhuang, F. Yan, and H. Hu, "Automatic extrinsic self-calibration for fusing data from monocular vision and 3-D laser scanner," *IEEE Transactions on Instrumentation and Measurement*, vol. 63, no. 7, pp. 1874–1876, 2014.

[212] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Dang, U. Franke, N. Appenrodt, C. G. Keller, *et al.*, "Making bertha drive—an autonomous journey on a historic route," *IEEE Intelligent transportation systems magazine*, vol. 6, no. 2, pp. 8–20, 2014.