

CONVOLUTIONAL NEURAL NETWORKS FOR JOINT OBJECT
DETECTION AND POSE ESTIMATION IN TRAFFIC SCENES

by

CARLOS GUINDEL GÓMEZ

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:

José María Armingol Moreno
David Martín Gómez

Tutor:

José María Armingol Moreno

December 2019

This thesis is distributed under license [Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Spain](https://creativecommons.org/licenses/by-nc-nd/3.0/es/).



By far the greatest danger of AI
is that people conclude too early
that they understand it.

— Eliezer Yudkowsky

ACKNOWLEDGMENTS / AGRADECIMIENTOS

I beg the English-speaking readers to understand my decision to write these lines in my native language, Spanish. Before that, however, I would like to thank Prof. Stiller and the other members of the MRT team for the four months that I spent there. Also, thanks to the members of the scientific community who made this thesis possible by publicly releasing databases and source code.

Por fin toca escribir estas líneas, y eso significa que una etapa muy importante de mi vida está a punto de cerrarse. Es mucho lo que me llevo de ella, y lo más importante no forma parte del plano académico.

El origen de todo puede fijarse en aquel lejano 2011, cuando dos asustadizos proyectantes entran en el 1.3.B16 de la mano de Basam. Parte de la *culpa* de esto es suya, aunque he de reconocer tanto él como Dani fueron fundamentales para entender muchas cosas.

Durante estos años, he visto pasar varias generaciones de compañeros por el laboratorio. Desde los que estaban entonces hasta los actuales *chavalitos* ha pasado mucha gente estupenda, imposible de citar en su totalidad. A todos ellos quiero agradecerles el haber contribuido a que el día a día fuera un placer, más que un trabajo.

Espero que nadie se ofenda si hago una mención especial a un grupo de tres con los que me lo he pasado muy bien estos años: Piñi Jorge, María y Noelia, compañeros de alegrías y desgracias, y también grandes amigos. Ah, por cierto, el agradecimiento a Jorge va por partida doble, porque tengo que reconocer que sus siempre brillantes ideas son una parte importante de esta tesis.

No puedo dejar de mencionar a mis directores de tesis, José María y David, con los que he aprendido durante estos años lecciones muy valiosas para la vida; ni tampoco a Arturo y, especialmente, Nando, que bien podría aparecer, también, en la portada de este documento.

Tampoco me olvido de la gente que conocí en la estancia, especialmente los otros dos miembros del *Spanisches Team*, Carlos y Edu, que me ayudaron mucho durante esos meses. *Ein herzliches Dankeschön!*

Hay otro grupo que merece una mención especial: los que siempre me han apoyado pero han sufrido los daños colaterales de la tesis. Hablo de mi familia, mis padres y mi hermana, que por fin me van a ver terminar, y mi otro *hermano*, Dani, que es otro distinto al que era cuando empezó esto, pero sigue y seguirá ahí. Gracias a todos.

Finalmente... ¿se ha percatado el lector del detalle de que en el segundo párrafo se menciona a una segunda persona? Nunca le gustó esto de la tesis, pero es el alfa y el omega de ella. Gracias por todo, Irene. De corazón. Ahora, vamos a por la nueva etapa que comienza.

*Carlos Guindel
Leganés, septiembre de 2019*

PUBLISHED AND SUBMITTED CONTENT

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

PUBLISHED CONTENT

Journal articles

- C. Guindel, D. Martín, and J. M. Armingol, “Fast joint object detection and viewpoint estimation for traffic scene understanding,” *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 4, pp. 74–86, 2018, ISSN: 1939-1390. DOI: [10.1109/MITS.2018.2867526](https://doi.org/10.1109/MITS.2018.2867526) [111].

Partially included in the thesis: Chapters 4 and 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.

- C. Guindel, D. Martín, and J. M. Armingol, “Traffic scene awareness for intelligent vehicles using convnets and stereo vision,” *Robotics and Autonomous Systems*, vol. 112, pp. 109–122, 2019, ISSN: 0921-8890. DOI: [10.1016/j.robot.2018.11.010](https://doi.org/10.1016/j.robot.2018.11.010) [114].

Partially included in the thesis: Chapters 3, 4, 5 and 6. 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

- C. H. Rodríguez-Garavito, C. Guindel, and J. M. Armingol, “Sistema de asistencia a la conducción para detección y clasificación de carriles,” in *Actas de las XXXVI Jornadas de Automática*, Bilbao, Spain, 2015, pp. 26–31, ISBN: 978-84-15914-12-9 [219].

Partially included in the thesis: Annex A. 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, D. Martín, and J. M. Armingol, “Joint object detection and viewpoint estimation using cnn features,” in *Proceedings of the 2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 2017, pp. 145–150. DOI: [10.1109/ICVES.2017.7991916](https://doi.org/10.1109/ICVES.2017.7991916) [110].

The extended version of this paper, published in the IEEE Intelligent Transportation Magazine [111], is partially included in the thesis, as stated before.

- C. Guindel, J. Beltrán, D. Martín, and F. García, “Automatic extrinsic calibration for lidar-stereo vehicle sensor setups,” in *Proceedings of the 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 674–679. DOI: [10.1109/ITSC.2017.8317829](https://doi.org/10.1109/ITSC.2017.8317829) [109].

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.

- C. Guindel, D. Martín, and J. M. Armingol, “Stereo vision-based convolutional networks for object detection in driving environments,” in *Computer Aided Systems Theory – EUROCAST 2017*, R. Moreno-Díaz, F. Pichler, and A. Quesada-Arencibia, Eds., Cham: Springer International Publishing, 2018, pp. 427–434, ISBN: 978-3-319-74727-9. DOI: [10.1007/978-3-319-74727-9_51](https://doi.org/10.1007/978-3-319-74727-9_51) [113].

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, D. Martín, and J. M. Armingol, “Modeling traffic scenes for intelligent vehicles using cnn-based detection and orientation estimation,” in *ROBOT 2017: Third Iberian Robotics Conference*, A. Ollero, A. Sanfeliu, L. Montano, N. Lau, and C. Cardeira, Eds., Cham: Springer International Publishing, 2018, pp. 487–498, ISBN: 978-3-319-70836-2. DOI: [10.1007/978-3-319-70836-2_40](https://doi.org/10.1007/978-3-319-70836-2_40) [112].

The extended version of this paper, published in the Robotics and Autonomous Systems journal [114], is partially included in the thesis, as stated before.

- A. Barrera, C. Guindel, F. García, and D. Martín, “Análisis, evaluación e implementación de algoritmos de segmentación semántica para su aplicación en vehículos inteligentes,” in *Actas de las*

XXXIX *Jornadas de Automática*, Badajoz, Spain, 2018, pp. 983–990, ISBN: 978-84-09-04460-3 [11].

Partially included in the thesis: Annex A. 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. D. L. Escalera, “BirdNet: A 3d object detection framework from lidar information,” in *Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 3517–3523. DOI: [10.1109/ITSC.2018.8569311](https://doi.org/10.1109/ITSC.2018.8569311) [15].

Partially included in the thesis: Chapter 6. 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. Fernández, C. Guindel, N. Salscheider, and C. Stiller, “A deep analysis of the existing datasets for traffic light state recognition,” in *Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 248–254. DOI: [10.1109/ITSC.2018.8569914](https://doi.org/10.1109/ITSC.2018.8569914) [76].

Partially included in the thesis: Annex A. 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, D. Martín, J. M. Armingol, and C. Stiller, “Analysis of the influence of training data on road user detection,” in *Proceedings of the 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, 2018, pp. 21–26. DOI: [10.1109/ICVES.2018.8519510](https://doi.org/10.1109/ICVES.2018.8519510) [115].

Partially included in the thesis: Chapters 4 and 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.

THIRD-PARTY MATERIAL

Material NOT authored or co-authored by the author of this thesis

- Fig. 1.8 was reprinted from [227], which is an open access article distributed under the Creative Commons Attribution (CC BY) license¹.
- Figs. 2.1 (from [261]) and 2.2 (from [265]) were reprinted with permission of the copyright holder, John Wiley and Sons.
- Figs. 2.3, 2.4, 2.5, 2.7, 2.8, 2.9, 2.10, 2.11, 2.12, 2.13, 5.1, 6.3 were reprinted with permission from the copyright holder, the IEEE.
- Fig. 2.6 was reprinted by permission from Springer Nature Customer Service Centre GmbH: Springer Nature, *Nature* 521, 436-444, "Deep learning," Y. LeCun, Y. Bengio, and G. Hinton, © Springer Nature 2015.
- Fig. 3.1 was reprinted from https://commons.wikimedia.org/wiki/File:Epipolar_geometry.svg and is licensed under the Creative Commons Attribution-Share Alike 3.0 Unported (CC BY-SA) license².
- Fig. 4.7 was reprinted from a Bachelor's Degree supervised by the author of this thesis [6] and is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 Spain (CC BY-NC-ND) license³.
- Fig. 5.2 was reprinted by permission from Springer Nature Customer Service Centre GmbH: Springer Nature, *Springer eBook*, "Stereo Vision-Based Semantic 3D Object and Ego-Motion Tracking for Autonomous Driving," Peiliang Li, Tong Qin, and Shaojie Shen, © Springer Nature Switzerland AG 2018.

Material authored or co-authored by the author of this thesis

- Figs. 3.2, 3.3, 5.14, 5.15, 6.4, 6.7, 6.8, 6.9, and 6.10, and Tables 4.1 and 5.10 were reprinted from [114] (*Robotics and Autonomous Systems* journal) with permission of the copyright holder, Elsevier.
- Figs. 3.7, 3.8, and 3.9 (partially) (from [109]); Figs. 5.4, 5.7, 5.8, 5.9, 5.11, 5.12, and 5.13, and Tables 5.2 (partially), 5.3, and 5.4 (from [111]); Figs. 5.10, 5.17, and 5.18, and Tables 4.11, 4.13, 5.7, 5.11,

¹ <http://creativecommons.org/licenses/by/4.0/>

² <https://creativecommons.org/licenses/by-sa/3.0/>

³ <https://creativecommons.org/licenses/by-nc-nd/3.0/es/deed.en>

5.12, 5.13, and 5.14 (from [115]); Figs. 6.13 and 6.14, and Tables 6.5, 6.6, 6.8, and 6.9 (from [15]); and Tables A.2, A.3, A.4, and A.5 (from [76]) were reprinted with permission from the copyright holder, the IEEE.

- Figs. 4.5 and 4.6, and Tables 4.5 and 4.10 (from [113]) were reprinted by permission from Springer Nature Customer Service Centre GmbH: Springer Nature, *Springer eBook*, "Stereo Vision-Based Convolutional Networks for Object Detection in Driving Environments," C. Guindel, D. Martín and J. M. Armingol, © Springer International Publishing AG 2018.
- Fig. A.4 and Tables A.6 and A.7 were reprinted from [9] and are licensed under the Creative Commons Attribution-NonCommercial 3.0 Unported (CC BY-NC) license⁴.

⁴ <https://creativecommons.org/licenses/by-nc/3.0/>

OTHER RESEARCH MERITS

ADDITIONAL PUBLISHED CONTENT

Conference articles

- P. Marín-Plaza, A. Hussein, C. Guindel, F. García, D. Martín, and A. de la Escalera, “Arquitectura basada en ros para el vehículo icab (intelligent campus automobile),” in *Actas de las XXXVII Jornadas de Automática*, Madrid, Spain, 2016, pp. 639–644, ISBN: 978-84-617-4298-1
- 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 International Conference on Intelligent Transportation Systems (ITSC) (to be presented)*

RESEARCH VISITS

- Research visit to the MRT group at Karlsruhe Institute of Technology ([KIT](#)), Germany, under the supervision of Prof. Christoph Stiller, from 05/03/2018 to 24/06/2018.

SUPERVISED ACADEMIC WORKS

Master’s Theses

- C. B. Jaraquemada, “Paralelización de algoritmos de visión estéreo para análisis de entornos de tráfico en GPU mediante CUDA,” Co-supervised with José María Armingol, Master’s Thesis (TFM). Master in Robotics and Automation, Universidad Carlos III de Madrid, Mar. 2017
- A. Barrera, “Estudio de algoritmos de segmentación semántica basados en deep learning para su aplicación en vehículos inteligentes,” Co-supervised with Fernando García, Master’s Thesis (TFM). Master in Robotics and Automation, Universidad Carlos III de Madrid, Sep. 2018
- D. Plaza, “Aplicación de técnicas de visión por computador y SVM para el reconocimiento de líneas en la carretera utilizando imágenes transformadas a vista de pájaro,” Co-supervised with

David Martín, Master's Thesis (TFM). Master in Robotics and Automation, Universidad Carlos III de Madrid, Jun. 2019

Bachelor's Theses

- D. Cabo, "Plataforma integrada de visión estéreo para la reconstrucción 3D de la escena con procesamiento paralelo en GPU," Bachelor's Thesis (TFG), Bachelor's Degree in Industrial Electronics and Automation Engineering, Universidad Carlos III de Madrid, Sep. 2016. [Online]. Available: <http://hdl.handle.net/10016/27085>
- I. Barredo, "Entrenamiento de algoritmos de deep learning para la detección de objetos con la base de datos Cityscapes," Bachelor's Thesis (TFG), Bachelor's Degree in Industrial Electronics and Automation Engineering, Universidad Carlos III de Madrid, Jul. 2017. [Online]. Available: <http://hdl.handle.net/10016/27520>
- P. Rueda, "Implementación de algoritmo de modelado de carriles en la arquitectura de software del vehículo inteligente ivvi 2.0," Bachelor's Thesis (TFG), Bachelor's Degree in Industrial Electronics and Automation Engineering, Universidad Carlos III de Madrid, Sep. 2017. [Online]. Available: <http://hdl.handle.net/10016/27858>
- L. Díaz, "Sistema de detección y clasificación de señales de tráfico basado en deep learning," Bachelor's Thesis (TFG), Bachelor's Degree in Industrial Electronics and Automation, Universidad Carlos III de Madrid, Sep. 2017

ABSTRACT

Few any longer question that autonomous vehicles will be a key element of transportation in the coming decades. Reliable perception of the surroundings of the vehicle is today one of the remaining technical challenges that must be addressed to ensure safe autonomous navigation, especially in crowded environments. This functionality usually relies on onboard sensors, which provide data that must be appropriately processed.

Among the different tasks assigned to the perception suite of an automated vehicle, the detection of other road users that can potentially interfere with the trajectory of the vehicle is particularly critical. However, the identification of agents in sensor data is only the first step. Planning and control modules down the pipeline demand trustworthy information about how the objects are arranged in space. In particular, their orientation and location on the road plane are usually attributes of utmost importance to build a purposeful model of the environment.

This thesis aims to provide close-to-market solutions to these issues taking advantage of the dramatic breakthrough seen in deep neural networks in the past decade. The methods proposed in this thesis are built on top of a popular detection framework, Faster R-CNN, which features high detection accuracy at near real-time rates. Some proposals to enhance the performance of the algorithm in images obtained from onboard cameras are introduced and discussed.

One of the central contributions of the thesis is the extension of the Faster R-CNN framework to estimate the orientation of the detected objects based exclusively on appearance information, which makes the method robust against the different sources of error present in traffic environments. As a natural next step, two algorithms exploiting this functionality to perform 3D object localization are proposed. As a result, the combination of the methods described throughout this thesis leads to a procedure able to provide situational awareness of the potential hazards in the surroundings of the vehicle.

All the proposed methods are analyzed and validated through systematic experimentation using a well-recognized public dataset (the KITTI Vision Benchmark Suite), where notable results were obtained. The viability of the implementation of the solutions in a real vehicle is also discussed.

KEYWORDS: object detection; computer vision; convolutional neural networks; autonomous driving systems

RESUMEN

Pocos cuestionan ya que los vehículos autónomos serán un elemento clave del transporte en las próximas décadas. La percepción fiable del entorno del vehículo es, hoy en día, uno de los retos técnicos que hay que afrontar para garantizar una navegación autónoma segura, especialmente en entornos con muchos agentes. Esta funcionalidad se basa, normalmente, en sensores embarcados, que proporcionan datos que deben ser procesados de forma adecuada.

Entre las diferentes tareas asignadas al sistema de percepción de un vehículo automatizado, la detección de otros usuarios de la vía que puedan interferir potencialmente con la trayectoria del vehículo es particularmente crítica. Sin embargo, la identificación de los agentes en los datos de los sensores es sólo el primer paso. Los módulos de planificación y control del vehículo exigen información fiable sobre la disposición de los objetos en el espacio. En particular, su orientación y ubicación en el plano de la carretera suelen ser atributos de suma importancia para construir un modelo del entorno significativo.

Esta tesis tiene como objetivo proporcionar soluciones comercialmente viables para estos problemas, aprovechando el impresionante avance que han experimentado las redes neuronales profundas en la última década. Los métodos propuestos en esta tesis se basan en un marco de detección popular, Faster R-CNN, que ofrece una alta precisión de detección a velocidades cercanas al tiempo real. Así, se presentan y discuten algunas propuestas para mejorar el rendimiento del algoritmo en las imágenes obtenidas de las cámaras a bordo.

Una de las aportaciones centrales de la tesis es la ampliación de la arquitectura Faster R-CNN para estimar la orientación de los objetos detectados basándose exclusivamente en la información de apariencia, lo que hace que el método sea robusto frente a las diferentes fuentes de error presentes en los entornos de tráfico. Como siguiente paso natural, se proponen dos algoritmos que aprovechan esta funcionalidad para realizar la localización de objetos en 3D. Como resultado, la combinación de los métodos descritos a lo largo de esta tesis permite construir un procedimiento capaz de proporcionar conciencia situacional de los peligros potenciales en los alrededores del vehículo.

Todos los métodos propuestos son analizados y validados mediante experimentación sistemática utilizando una reconocida base de datos pública (*KITTI Vision Benchmark Suite*), donde se han obtenido resultados notables. También se discute la viabilidad de la implementación de las soluciones en un vehículo real.

PALABRAS CLAVE: detección de objetos; visión por computador; redes neuronales convolucionales; sistemas de conducción autónoma

CONTENTS

I	PROBLEM STATEMENT AND LITERATURE REVIEW	
1	INTRODUCTION	3
1.1	Motivation	4
1.1.1	Traffic accidents	4
1.1.2	Air pollution and climate change	8
1.1.3	Congestion	9
1.2	Automated driving	9
1.2.1	Driver assistance systems	10
1.2.2	Autonomous driving	11
1.3	Perception systems	14
1.3.1	Sensors	14
1.3.2	Algorithms	16
1.4	Objectives	18
1.5	Outline of the dissertation	19
2	RELATED WORKS	21
2.1	Historical autonomous driving platforms	21
2.2	Datasets	24
2.2.1	Object recognition	24
2.2.2	Driving environments	25
2.2.3	Synthetic datasets	27
2.3	Sensor calibration	28
2.3.1	Camera intrinsic parameters	28
2.3.2	Extrinsic parameters	29
2.4	Convolutional neural networks	30
2.4.1	Historical evolution	31
2.4.2	Fundamentals	32
2.4.3	Architectures for image recognition	33
2.5	Object detection in images	35
2.5.1	Historical evolution	36
2.5.2	Meta-architectures	37
2.5.3	Training	40
2.5.4	Inference	42
2.6	Perception on automotive platforms	42
2.6.1	Object classification and detection	43
2.6.2	Viewpoint estimation	44
2.6.3	Obstacle 3D localization	46
2.6.4	Multi-tasking and scene understanding	48
2.7	Conclusion	49
II	PROPOSED APPROACHES AND EXPERIMENTAL RESULTS	
3	SENSOR SETUP	53
3.1	Fundamentals	53

3.1.1	Monocular cameras	53
3.1.2	Stereo cameras	56
3.1.3	Lidar	57
3.2	Data representation	58
3.2.1	Stereo matching	58
3.2.2	Lidar	62
3.3	Sensor calibration	63
3.3.1	Intrinsic calibration	64
3.3.2	Extrinsic calibration	65
3.4	Automatic stereo-lidar extrinsic calibration	66
3.4.1	Introduction	66
3.4.2	Feature extraction from stereo data	67
3.4.3	Registration	71
3.4.4	Experimental results	71
3.4.5	Additional remarks	73
3.5	Conclusion	74
4	OBJECT DETECTION	75
4.1	Faster R-CNN paradigm	75
4.1.1	Design principles	76
4.1.2	Training	77
4.2	Tuning for traffic environments	81
4.2.1	KITTI object detection benchmark	81
4.2.2	Hyperparameter tuning of the detection framework	84
4.2.3	Experimental setup and preliminary assessment	89
4.3	Enhanced detection using stereo vision	92
4.3.1	Depth information encoding	92
4.3.2	Experimental results	94
4.4	Analysis of the influence of training data	95
4.4.1	Experimental setup	96
4.4.2	Analysis	98
4.5	Conclusion	99
5	VIEWPOINT ESTIMATION	101
5.1	Problem formulation	101
5.1.1	Orientation in the KITTI dataset	104
5.2	Viewpoint estimation within Faster R-CNN	106
5.2.1	Interpretation of the probability distribution	107
5.2.2	Training	108
5.2.3	Experimental results	109
5.3	Identification of factors affecting the performance	116
5.3.1	Training hyperparameters	118
5.3.2	Number of proposals	119
5.3.3	Viewpoint bins resolution	121
5.3.4	Feature extractor architecture, input scale, and combinations	122
5.3.5	Training data	126

5.4	Improvements in the baseline solution	131
5.4.1	Hybrid viewpoint estimation	132
5.4.2	Validation of the general approach in modern frameworks	136
5.5	Conclusion	139
6	OBJECT LOCALIZATION	141
6.1	Object localization based on stereo data	141
6.1.1	Extrinsic auto-calibration	143
6.1.2	Object 3D localization	144
6.1.3	Experimental results	147
6.1.4	Scene modeling	150
6.2	Object detection and localization based on lidar data	153
6.2.1	Detection and yaw estimation in lidar data	153
6.2.2	Experimental results	155
6.3	Conclusion	160
III CONCLUDING REMARKS		
7	CONCLUSION AND FUTURE WORK	165
7.1	Conclusion	165
7.2	Future work	167
IV APPENDIX		
A	ADDITIONAL CUES FOR SCENE UNDERSTANDING	173
A.1	Lane detection and classification	173
A.2	Road signaling classification	175
A.2.1	Detection	175
A.2.2	Traffic sign classification	176
A.2.3	Traffic light classification	177
A.3	Semantic segmentation	180
BIBLIOGRAPHY		185

LIST OF FIGURES

Figure 1.1	Vehicles in use for the period 2005–2015, worldwide	4
Figure 1.2	Estimated deaths due to road injuries for the period 2000–2016, worldwide	5
Figure 1.3	Road traffic fatalities per 100 000 population, by world region (2013)	5
Figure 1.4	Road traffic deaths by type of road user, worldwide (2013)	6
Figure 1.5	Road traffic fatalities, EU (2015)	7
Figure 1.6	Road traffic collisions with victims and deaths by type of accident, Spain (2017)	7
Figure 1.7	Road traffic collisions and deaths by factor involved, Spain (2017)	8
Figure 1.8	Features of the sensor modalities used in environment perception systems	16
Figure 2.1	Lidar-based terrain detection featured by Stanley (Stanford University) during the 2005 DARPA Grand Challenge	22
Figure 2.2	Sensor setup featured by Boss (Carnegie Mellon University) during the 2007 DARPA Urban Challenge	23
Figure 2.3	Sensor setup featured by TerraMax (Vislab and others) during the 2007 DARPA Urban Challenge	23
Figure 2.4	Recording platform, trajectory, disparity and optical flow maps, and 3D object labels from the KITTI dataset	26
Figure 2.5	Calibration setup used by Geiger et al.	30
Figure 2.6	Convolutional Neural Network (CNN)	32
Figure 2.7	Residual block	35
Figure 2.8	Region-based Convolutional Neural Networks (R-CNN)	38
Figure 2.9	Fast R-CNN	39
Figure 2.10	Faster R-CNN	39
Figure 2.11	Estimation results of the Deep3DBox method . .	46
Figure 2.12	Multi-View 3D object detection network (MV3D)	47
Figure 2.13	3D intersection understanding problem and cues employed by Geiger et al.	49
Figure 3.1	Epipolar geometry	58
Figure 3.2	Stereo matching and disparity maps obtained with two different methods	59
Figure 3.3	3D point clouds obtained with two different stereo matching algorithms	61

Figure 3.4	Three different representations of the lidar information on a scene from the KITTI dataset	64
Figure 3.5	Checkerboard pattern used for intrinsic calibration as seen from a camera with a high positive radial distortion (barrel distortion)	65
Figure 3.6	Calibration target used by the proposed calibration method	67
Figure 3.7	Proposed approach to extract the reference points from the stereo point cloud	68
Figure 3.8	Accuracy of the proposed calibration approach in comparison with other existing methods	73
Figure 3.9	Examples of the calibration result in real scenes .	74
Figure 4.1	Faster R-CNN overview	77
Figure 4.2	Histograms of sizes ($W \times H$) for <i>Hard</i> samples in the KITTI training subset	87
Figure 4.3	Histograms of ratios (H/W) for <i>Hard</i> samples in the KITTI training subset	87
Figure 4.4	Custom RPN anchors	88
Figure 4.5	Proposed approach to incorporate stereo information into the Faster R-CNN framework	92
Figure 4.6	Example of the processed disparity maps on a frame from the KITTI dataset	94
Figure 4.7	Example of the conversion from pixel-wise to bounding box labels	96
Figure 5.1	Illustration of the difference between yaw angle and viewpoint	102
Figure 5.2	Representation of different viewpoints	102
Figure 5.3	Definition of yaw (φ) and viewpoint (θ) angles .	103
Figure 5.4	Example of viewpoint quantization with 8 bins ($N_b = 8$)	104
Figure 5.5	Relationship between orientation similarity and error in orientation estimation	105
Figure 5.6	Proposed approach to incorporate viewpoint estimation capabilities into the Faster R-CNN framework (FRCNN+Or)	106
Figure 5.7	Precision-recall and Recall-IoU curves of the proposed approach on the KITTI <i>Moderate</i> validation subset with and without viewpoint estimation .	113
Figure 5.8	Orientation similarity vs. recall of the proposed approach on the KITTI <i>Moderate</i> validation subset for three alternative viewpoint estimation approaches	114
Figure 5.9	Examples of joint detection and viewpoint estimation on scenes from the KITTI testing set using the proposed approach	116

Figure 5.10	Detection and viewpoint estimation performance (mAP % and mAOS %) of the proposed approach vs. number of training iterations on the KITTI validation subset	118
Figure 5.11	Performance (Average Recall, mAP % and mAOS %) and run time (s) of the proposed approach vs. the number of proposals on the KITTI <i>Moderate</i> validation subset	120
Figure 5.12	Recall of the proposed approach vs. max. distance from the ego-car on the KITTI <i>Hard</i> validation subset for different numbers of proposals . .	120
Figure 5.13	Orientation similarity (OS)-recall and MPPE-recall curves of the proposed approach on the KITTI <i>Moderate</i> validation subset for different values of N_b	122
Figure 5.14	Detection and viewpoint estimation performance (mAP % and mAOS %) of the proposed approach vs. run time on the KITTI <i>Moderate</i> validation subset for different numbers of proposals, architectures and input scales	124
Figure 5.15	Detection and viewpoint estimation performance (mAP % and mAOS %) of the proposed approach vs. run time on the KITTI <i>Moderate</i> validation subset with the VGG-16 backbone for different numbers of bins and input scales	125
Figure 5.16	Images obtained by applying each data augmentation technique to the same Cityscapes frame .	129
Figure 5.19	Example of residual angle (ρ)	132
Figure 6.1	Proposed stereo vision-based object localization method	142
Figure 6.2	Definition of coordinate frames for obstacle localization	143
Figure 6.3	Stereo rig and ground plane in a typical setup .	143
Figure 6.4	Example showing the extrinsic auto-calibration procedure	144
Figure 6.5	Area considered by the proposed object localization approach within an object's bounding box .	145
Figure 6.6	Schematic representation of the BEV of a sample scene with a detected object	146
Figure 6.7	Absolute Euclidean error (m) in the location estimation from the proposed approach for the different categories, using two different stereo matching algorithms	148

Figure 6.8	Absolute Euclidean error (m) in the location estimation from the proposed approach vs. distance from the ego-car using two different stereo matching algorithms	149
Figure 6.9	Localization error (detection minus ground-truth, m) along the depth axis when estimating the center of the object for two different localization methods	149
Figure 6.10	Examples of detections and local scene models obtained from the proposed approach for different values of N_b	151
Figure 6.11	Examples of detections and local scene models obtained from the proposed approach in our IVVI 2.0 platform	152
Figure 6.12	Proposed approach for 3D object detection in lidar data (BirdNet), based on the joint detection and viewpoint estimation network	153
Figure 6.13	Recall of the proposed BEV detection approach on the KITTI validation subset at different IoU thresholds using 300 proposals	159
Figure 6.14	Example of 3D detection results on scenes from the KITTI testing set using the proposed method	161
Figure A.1	Bird’s Eye View (BEV) for lane detection	174
Figure A.2	Example of line and lane detections using the proposed approach	175
Figure A.3	Examples of lane detection and classification results using the proposed approach	176
Figure A.4	Examples of semantic segmentation results for each of the studied models	183

LIST OF TABLES

Table 1.1	Summary of levels of driving automation, as defined by the SAE International	13
Table 2.1	Classification error rates (%) on the ImageNet validation set for different classification models	36
Table 2.2	Orientation estimation performance (AOS %) of selected methods on the KITTI object detection benchmark	46
Table 2.3	BEV and 3D detection performance (AP %) of selected object localization methods on the KITTI object detection benchmark	48

Table 3.1	Error rates and run times of selected stereo matching methods on the KITTI dataset	61
Table 3.2	Parameters for reference points extraction from stereo data	70
Table 4.1	Object occurrence statistics for the custom train and validation KITTI subsets	82
Table 4.2	Definition of the levels of difficulty from the KITTI dataset	83
Table 4.3	Number of CAR, PED, and CYC instances meeting the requirements of the <i>Hard</i> difficulty level in the KITTI dataset	84
Table 4.4	Weights in the infogain matrix for two different sets of categories	86
Table 4.5	Modified settings for RPN anchors	88
Table 4.6	Training hyperparameters for the tuned Faster R-CNN	90
Table 4.7	Detection performance (AP %) of the tuned Faster R-CNN on the KITTI validation subset	91
Table 4.8	NVIDIA Titan Xp GPU specifications	91
Table 4.9	Training hyperparameters for the proposed stereo-vision-capable Faster R-CNN	94
Table 4.10	Comparison of the performance (AP % and AOS %) of the proposed stereo-vision-capable Faster R-CNN with other methods on the KITTI validation subset	95
Table 4.11	Object occurrence stats for the KITTI and Cityscapes subsets used in the analysis	97
Table 4.12	Training hyperparameters for the analysis of the influence of training data on the detection performance	98
Table 4.13	Detection performance (AP %) of Faster R-CNN on the KITTI validation subset for different sets of training data	99
Table 5.1	Set 1 of training hyperparameters for the proposed approach	110
Table 5.2	Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset with and without onboard-oriented tuning	111
Table 5.3	Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset for different alternatives for viewpoint estimation refinement	112
Table 5.4	Comparison of the performance (AP % and AOS %) of the proposed approach with other methods on the KITTI testing set	115

Table 5.5	Set 2 of training hyperparameters for the proposed approach	117
Table 5.6	Set 3 of training hyperparameters for the proposed approach	117
Table 5.7	Detection and viewpoint estimation performance (mAP % and mAOS %) of the proposed approach on the KITTI validation subset with and without dropout	119
Table 5.8	Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset for different numbers of viewpoint bins (N_b)	123
Table 5.9	Faster R-CNN hyperparameters for each of the studied feature extractors	123
Table 5.10	Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset for different scales and difficulty levels	126
Table 5.11	Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset using different sets of training data	127
Table 5.12	Viewpoint classification performance (MPPE %) of the proposed approach on the KITTI validation subset using different training data sets	128
Table 5.13	Detection and viewpoint estimation performance (mAP % and mAOS %) of the proposed approach on the KITTI validation subset with and without pre-training on ImageNet	128
Table 5.14	Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset with and without data augmentation	130
Table 5.15	Mapping of the extended set of categories to the original labels in KITTI and Cityscapes	131
Table 5.16	Training hyperparameters for the residual regression proposal	134
Table 5.17	Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset using different alternatives for viewpoint refinement	135
Table 5.18	Orientation performance (AOS % / AP % ratio) of the proposed approach on the KITTI validation subset using different alternatives for viewpoint refinement	136

Table 5.19	Set of training hyperparameters for the implementation of the proposed approach in Detectron . . .	137
Table 5.20	Detection and viewpoint estimation performance (AP % and AOS %) of the baseline implementation and the enhanced implementation of the proposed approach on the KITTI validation subset	138
Table 6.1	Dimensions of the cuboids assigned to each KITTI category	146
Table 6.2	BEV detection performance (AP %) of the proposed object localization approach	150
Table 6.3	Modified RPN anchors for BEV detection	155
Table 6.4	Training hyperparameters for the proposed BEV detection network	156
Table 6.5	BEV and 3D detection performance (AP BEV % and AP 3D %) of the proposed BEV detection approach on the KITTI validation subset using different weight initialization strategies	156
Table 6.6	BEV detection performance (AP BEV %) on the KITTI validation subset for different variants of the proposed BEV detection approach	157
Table 6.7	BEV detection performance (AP BEV %) of the proposed BEV detection approach on the KITTI validation subset using different data as an input	158
Table 6.8	2D detection and viewpoint estimation performance (AP % and AOS %) of the proposed BEV detection approach on the KITTI testing set	158
Table 6.9	3D detection and BEV detection performance (AP 3D % and AP BEV %) of the proposed BEV detection approach on the KITTI testing set	159
Table 6.10	Comparison of the BEV <i>Car</i> detection performance (AP 3D % and AP BEV %) of the proposed BEV detection approach with other methods on the KITTI validation subset with IoU 0.5	160
Table A.1	Classification performance (accuracy %) of three architectures tested for traffic sign classification	177
Table A.2	Classification performance (mean F1 score) on different validation sets of traffic light color classification using the custom ResNet model	178
Table A.3	Classification performance (F1-score) of traffic light color classification on the combined dataset for the different training sets and models with and without data augmentation	179
Table A.4	Confusion matrix of the MobileNet network for traffic light classification trained and tested on the combined dataset using six labels	180

Table A.5	Confusion matrix of the ResNet model for traffic light classification trained and tested on the combined dataset using six labels	181
Table A.6	Semantic segmentation performance (mean IOU %) of ERFNet for different configurations	182
Table A.7	Comparison of the performance of the studied semantic segmentation methods on the Cityscapes validation set	182

ACRONYMS

ADAS	Advanced Driver-Assistance Systems
AI	Artificial Intelligence
AOS	Average Orientation Similarity
AP	Average Precision
AR	Average Recall
BN	Batch Normalization
BEV	Bird’s Eye View
CNN	Convolutional Neural Network
COCO	Common Objects in COntext
DGT	Dirección General de Tráfico
DL	Deep Learning
DNN	Deep Neural Network
EU	European Union
FC	Fully-Connected
FCN	Fully-Convolutional Network
FOV	Field Of View
FPN	Feature Pyramid Network
GPU	Graphics Processing Unit
HFOV	Horizontal Field Of View
IoU	Intersection over Union

ITS	Intelligent Transportation System
KIT	Karlsruhe Institute of Technology
LSI	Laboratorio de Sistemas Inteligentes
mAOS	mean Average Orientation Similarity
mAP	mean Average Precision
MPPE	Mean Precision in Pose Estimation
NMS	Non-Maximum Suppression
RANSAC	RANdom SAmples Consensus
ReLU	Rectified Linear Unit
ROI	Region Of Interest
ROS	Robot Operating System
RPN	Region Proposal Network
R-CNN	Region-based Convolutional Neural Networks
R-FCN	Region-based Fully Convolutional Networks
SGBM	Semi-Global Block Matching
SGD	Stochastic Gradient Descent
SGM	Semi-Global Matching
SSD	Single Shot Detector
TLR	Traffic Light Recognition
TSR	Traffic Sign Recognition
VFOV	Vertical Field Of View
VRU	Vulnerable Road User
WHO	World Health Organization
YOLO	You Only Look Once

Part I

PROBLEM STATEMENT AND LITERATURE
REVIEW

INTRODUCTION

Private transportation has been taken as granted in modern societies. Millions of people use the car daily, not only when commuting to work, but also for their convenience and leisure trips. Human beings are still the fundamental part of the control loop of vehicles; however, in the last decades, advances in automation technologies are reaching the transportation field, and the aspiration to build autonomous vehicles that dispense with an actual driver is getting closer and closer. Jointly with advances in alternative powertrains, a paradigm shift is expected in the coming years, with beneficial effects on the mitigation of the numerous problems linked to transportation.

Nevertheless, there are still several challenges to be solved. Driving a vehicle in traffic is a complex activity involving a wide range of human skills. The actuation on the vehicle controls (e. g., steering wheel, throttle, or brake) must be carried out concurrently with the processing of high amounts of data that the driver receives through the senses and utilizes to be aware of the conditions of the environment. Therefore, driving decisions are the result of a non-trivial procedure comprising the building of an internal representation of the environment based on this information intake. Since driving environments lack a well-defined structure, and are packed with diverse agents with unpredictable behaviors, these kinds of tasks are beyond the scope of classical automation techniques.

Fortunately enough, artificial intelligence and computer vision are currently experiencing one of the biggest bloomings in history, pushed by recent advances in hardware and, more specifically, in Graphics Processing Units (GPUs). Deep Neural Networks (DNNs), whose training has been enabled by this progress, have changed how data from sensors is processed and apprehended by automated systems. Intelligent Transportation Systems (ITSs) cannot be alien to this trend; on the contrary, this set of methods provide unprecedented opportunities to close the gap between machines and humans in this regard.

This thesis aims to be a step towards a real understanding of the traffic scene by automated systems. The focus will be on the study of different possibilities for object detection, classification, and localization, with an emphasis on systems geared towards traffic scene understanding from onboard processing units. The investigation will show the enormous potential of DNNs to provide decision elements to higher-level driving modules. Analysis and conclusions will be supported by extensive experimentation using real-world data from different sensor modalities.

1.1 MOTIVATION

TRANSPORT
SUSTAINABILITY

Transport, as it is understood today, is associated with several problems affecting health and the economy around the world, such as traffic accidents, congestion, air pollution, and contribution to climate change. Experts coincide in underscoring the fact that the current model is undeniably unsustainable [201], particularly given that the number of people owning a private vehicle is steadily increasing. According to data from the International Organization of Motor Vehicle Manufacturers (OICA) [135], the number of vehicles in use in the world reached 1282 million in 2015, which is an increase of 43.7% in the vehicle fleet from 2005 (Fig. 1.1).

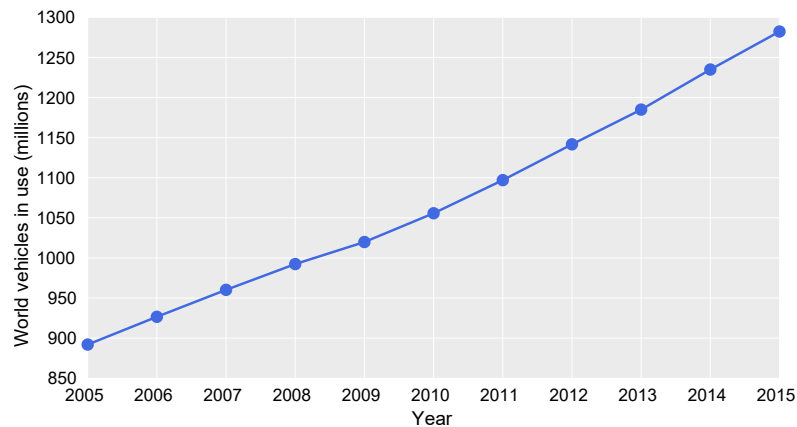


Figure 1.1: Vehicles in use for the period 2005–2015, worldwide. Data from the International Organization of Motor Vehicle Manufacturers (OICA) [135]

The most significant contribution to this increment came from developing countries. While the motorization rate (vehicles per 1000 inhabitants) in the European Union (EU) reaches 581, the number decreases to 105 in the Asia/Oceania/Middle East region and 41 in Africa. Naturally, the total number of vehicles will likely continue to grow in the future in line with the expected economic development in those regions.

TRANSPORT
PROBLEMS

This scenario leads to a worsening of the problems resulting from transportation, whose current impact is analyzed in this section.

1.1.1.1 *Traffic accidents*

SITUATION IN
THE WORLD

According to the World Health Organization (WHO) [278], road-related injuries were responsible for 1.4 million deaths globally in 2016. This number represents an increase of 23.4% in the death toll since 2000, as reported in Fig. 1.2. Overall, road traffic injuries are the 8th cause of death in the world and the first cause of death in the 15-29 age group.

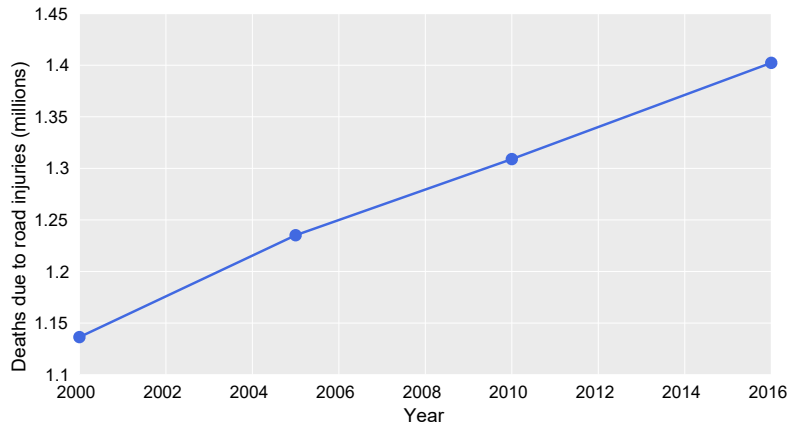


Figure 1.2: Estimated deaths due to road injuries for the period 2000–2016, worldwide. Data from the WHO [278]

Most of the world’s road traffic deaths (90%) take place in low- and middle-income countries, despite having only half of the world’s vehicles. The impact of the economic level on the death rate is especially notorious when the data is broken down into world regions, as shown in Fig. 1.3. Africa is leading the death rate statistic by a large margin, even though its motorization rate is the lowest in the world. This disparity can be explained by poorly designed roads, vehicles that do not meet proper safety standards, and inadequate safety legislation.

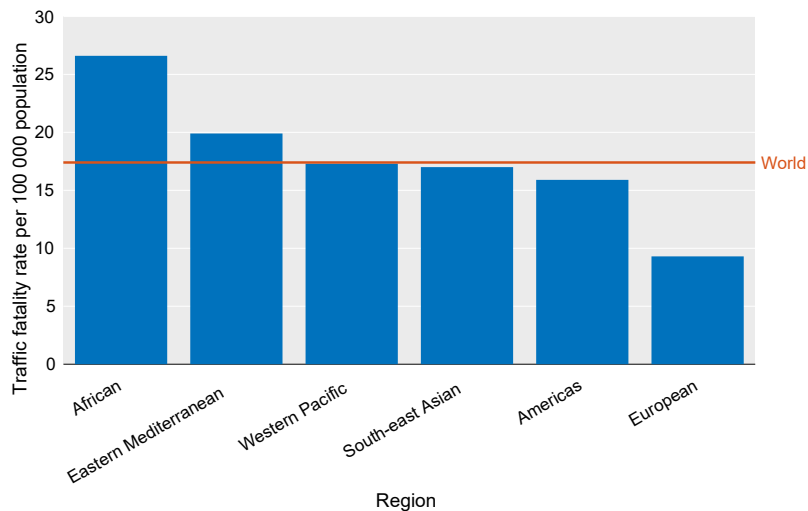


Figure 1.3: Road traffic fatalities per 100 000 population, by world region (2013). Data from the WHO [277]

It is also noteworthy that almost half of all deaths due to road traffic injuries are among the so-called Vulnerable Road Users (VRUs); i. e., traffic participants with limited protection, such as motorcyclists,

cyclists, and pedestrians. Fig. 1.4 depicts the distribution of road traffic deaths by type of road user

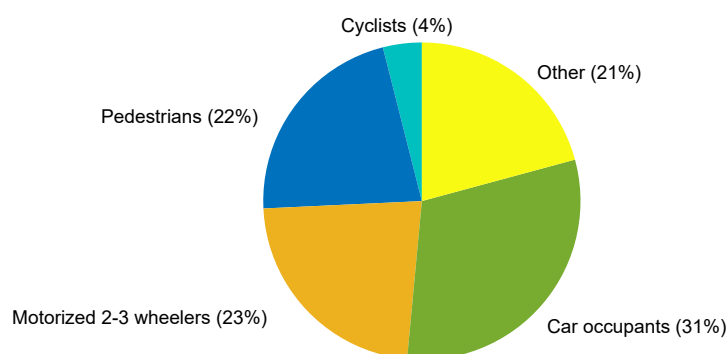


Figure 1.4: Road traffic deaths by type of road user, worldwide (2013). Data from the WHO [277]

The proportion varies by world region, with Africa having the highest proportion of pedestrian and cyclist deaths (43%) and the South-East Asia region, the lowest (16%). Cultural reasons are behind these numbers: whereas walking and cycling are the predominant forms of mobility in the African region, motorcycles are more prevalent in the South-East Asia region.

SITUATION IN EUROPE

Focusing on the EU, the European Commission set an ambitious objective for the period 2010-2020: halving the number of road deaths throughout the Union [67]. Although the number has been declining at a rapid pace (Fig. 1.5), progress has stagnated since 2013, and the European Commission has admitted the difficulty to achieve the target [69]. The latest data available at the time of writing, corresponding to 2017, gives a total of 25 300 deaths and 135 000 severe injuries on European roads. Most fatalities occurred on rural roads (55%) and urban areas (37%), while only 8% of them took place on motorways. It should be noted that the European legislation regarding safety equipment in vehicles is one of the most stringent in the world, requiring the mandatory fitment of a large number of active safety systems.

SITUATION IN SPAIN

Spain is a prime example of the situation in Europe. Although the mortality rate is low in comparison with other countries on the continent (8th lowest mortality rate in the EU), recent years have seen an increase in the number of deaths [60]. Most fatalities in 2017 (1013, 55%) took place in *conventional roads* (undivided highways), while almost half of the deceased (46%) were VRUs. On the other hand, more common types of accidents include angle or side impacts, rear-end collisions, run-off-road collisions (which are particularly lethal), and collisions with pedestrians (Fig. 1.6).

Factors frequently involved in accidents with deaths on the Spanish roads were distractions (33%), inappropriate speed (29%), and driving under the influence of alcohol (26%). Full statistics are presented in Fig. 1.7.

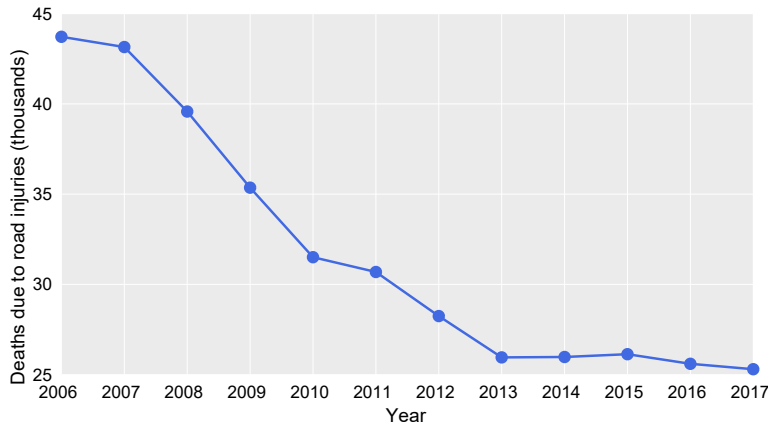


Figure 1.5: Road traffic fatalities, EU (2015). Data from the European Commission [68]

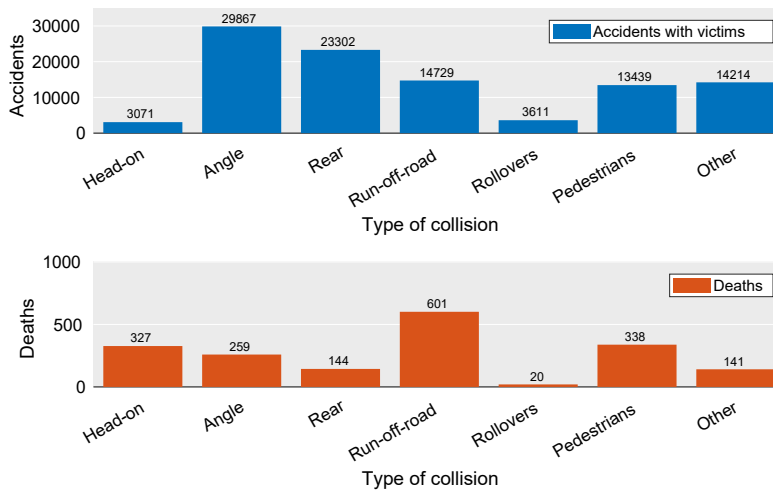


Figure 1.6: Road traffic collisions with victims and deaths by type of accident, Spain (2017). Data from the DGT [60]

Various studies around the world have shown that human errors are the primary cause of an overwhelming majority of traffic accidents. For instance, the National Highway Traffic Safety Administration (NHTSA) showed that 94% of the accidents in the United States between 2005 and 2007 were due to human errors, with a higher prevalence of recognition (41%) and decision (33%) errors [245]. Studies concerning other countries give similar estimations; for instance, 90% of accidents in Spain are reportedly caused by human errors [39].

CAUSES

Apart from the high social cost of traffic accidents, they are also estimated to cause economic losses of around 3% of Gross Domestic Product (GDP) globally [277].

ECONOMIC COST

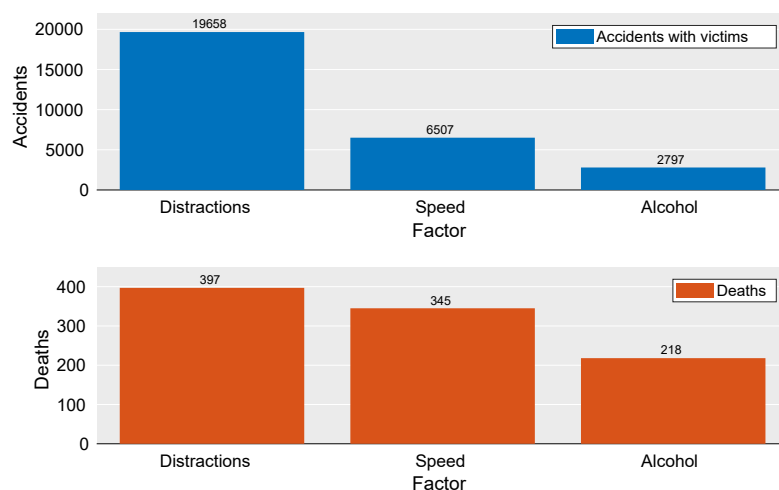


Figure 1.7: Road traffic collisions and deaths by factor involved, Spain (2017). Data from the DGT [60]

1.1.2 Air pollution and climate change

SITUATION IN THE WORLD

The WHO estimates that around 7 million people die annually from exposure to fine particles in polluted air [276], which leads to different diseases such as stroke, lung cancer, and respiratory infections. According to this organization, more than 80% of people living in urban areas where air pollution is monitored are exposed to air quality levels below the recommendations, and 97% of cities in low- and middle-income countries do not meet the air quality guidelines.

One of the main contributors to ambient fine particulate pollution is exhaust fume from diesel vehicles. Whereas emission standards for diesel vehicles in developed countries are getting more and more strict (e. g., *Euro* standards in the EU), the situation is dramatically different in developing countries, where growth in road traffic outpaces the emission regulation.

In addition to the human cost, air pollution has a substantial economic cost. In China, India, and the OECD¹ countries, the estimated cost is \$3 500 000 million, half of which results from road transport pollution [264].

On the other hand, transport is responsible for 23% of the global energy-related greenhouse gas emissions [264]. Emissions from the sector have increased by 1.7% per year on average since 2000, and this increase has been even faster than economic growth [201]. Non-OECD emissions have increased by more than 60% since 2000, and total

¹ Organization for Economic Co-operation and Development. Most OECD members are developed countries.

emissions from transport are projected to rise 70% by 2050. Almost all this growth will take place in developing countries.

In addition to CO₂, diesel engines emit other pollutants (e. g., black carbon) that have also been shown to have a high effect on climate change.

1.1.3 Congestion

The congestion of road networks is a daily problem in many cities. The trivial approach to this issue, namely the expansion of network capacity, is ineffective; actually, the Braess's paradox [27] states that adding a road to a congested road traffic network can indeed increase overall journey time. Costs produced by congestion include the increase in the costs of transport of goods, the decrease in the fuel efficiency of vehicles, and the loss of leisure time, among others. Congestion is a major problem in big cities; for instance, it has been estimated that, in Los Angeles (United States), a driver spends 102 hours a year in congestion, with a total cost for the city of \$19.2 billion [134]. Overall, total (direct and indirect) costs of congestion in the United States alone exceed \$304 billion (more than 1% of the GDP). Likewise, the total cost of congestion in the top five German cities was over €16.4 billion in 2017, and it is estimated at 2% of GDP in Europe [264].

SITUATION IN
THE WORLD

1.2 AUTOMATED DRIVING

Since the invention of motor vehicles, technology has been used to overcome the shortcomings of human drivers. It has been shown that, due to natural limits, human beings are little suitable to perform specific tasks involved in driving, such as [150]:

HUMAN DRIVER
LIMITATIONS

- Routine tasks.
- Simple but time-critical tasks.
- Vision at night and in adverse weather conditions.
- Estimation of distance and speed differences.
- Maintaining a safe and appropriate distance from other road users.

As an example, research conducted by Mercedes-Benz in 1992 showed that more than 90% of drivers fail to brake with enough force in the case of an emergency² [48].

² The findings of the research led to the development of the Emergency Brake Assist (EBA) or Brake Assist (BA) system, which increases automatically braking pressure when the driver is trying to execute an emergency stop.

1.2.1 *Driver assistance systems*ACTIVE SAFETY
SYSTEMS

Technology has been used over the years to improve comfort, efficiency, and, particularly, the safety of transportation. Advances in electronics were vital to enable the development of systems that use an understanding of the vehicle condition to avoid, or at least minimize, the effects of an accident. These systems, encompassed under the name of *active safety* systems, were gradually introduced in vehicles; some noteworthy examples are³ [93]:

- Anti-lock Braking System (ABS) (1966).
- Traction control (1971).
- Stability control (1995).

ADAS

The first active safety systems were mainly devoted to improving the capabilities of the vehicle in case of a dangerous situation, thus helping the driver to manage it safely. However, the final years of the 20th century saw the emergence of new kinds of technological systems with a more ambitious target: helping the driver in the driving process so that dangerous situations could be prevented ahead in time. These Advanced Driver-Assistance Systems (ADAS) were based on exteroceptive sensors able to acquire information from outside the vehicle, differently from the proprioceptive sensors featured by previous solutions, which were aimed at the measurement of the internal status of the vehicle [18]. Further discussion about automotive sensors is deferred until Sec. 1.3.1.

Among the different ADAS developed over the years, the following milestones can be highlighted:

PARKING ASSISTANCE SYSTEMS: Using ultrasonic sensors and, occasionally, cameras, these systems have evolved from warning devices aimed to prevent collisions, to solutions able to relieve the driver of lateral vehicle control during the maneuver.

ADAPTIVE CRUISE CONTROL (ACC): The development of radar technology enabled the emergence of ACC systems, which are intended to adjust the vehicle speed to maintain a safe distance from vehicles ahead.

FORWARD COLLISION PREVENTION SYSTEMS: Lidar (for low-speed applications) and radar sensors (for long-range) are used to avoid head-on collisions by making the driver aware of an impending collision and, eventually (if the driver does not react), activating the vehicle brakes by itself.

³ Note that the provided dates refer to the earliest version of the systems, which can vary significantly from their current implementations.

LANE DEPARTURE WARNING (LDW) SYSTEMS: Cameras were first applied to the mobility sector to provide information to LDW systems and active lane-keeping assistance systems, derived from them. These systems are responsible for letting the driver know when the vehicle is unintentionally leaving the current lane.

A useful compendium of **ADAS** and the main technical challenges associated with them can be found in [273].

1.2.2 *Autonomous driving*

The current trend in the development of automotive systems is naturally and imminently inclined to take a step forward towards eliminating the weakest link in the chain of vehicle control; namely, the driver. Autonomous driving, where decision-making systems take over as drivers without requiring any human intervention, is a future goal on every stakeholder's mind. Some typical use cases for autonomous driving are [269]:

USE CASES

INTERSTATE PILOT: The system takes over the driving task only on interstate-like expressways.

AUTONOMOUS VALET PARKING: The car can park itself at a remote location after the passengers have gotten off and cargo has been unloaded.

FULL AUTOMATION: In permitted areas, the "driver" can devote her/his attention to activities different than driving, although she/he can retake control at any time.

VEHICLE ON DEMAND: The robot car can drive itself in all scenarios with occupants, but also without any payload. This way, the vehicle is available on request at any location.

Widespread adoption of autonomous vehicles as a mode of mobility would probably have profound effects on society. Firstly, it could facilitate the development of ride-sharing and vehicle-sharing services [184] due to the reduction of operational costs. Mobility levels equivalent to the current ones could be reached with up to 90% fewer vehicles in the streets.

FORECASTS

Litman [170] has recently evaluated the expected overall impact. He divides the effects into two types: internal, which affect directly to the user, and external, which impact on others. Beneficial effects belonging to the first group are:

- Reduced drivers' stress and increased productivity due to the ability to rest, play, or work on the move.

- Mobility for non-drivers, including the elderly and people with travel-restrictive medical conditions.
- Reduced driver costs. Paid drivers for taxis and commercial transport would be unnecessary.

On the other hand, the group of external effects includes the following benefits:

- Increased safety. Reduced crash risks since the human factor is removed.
- Increased road capacity and reduced costs. Roadway capacity could be doubled or tripled because of the reduced headways; insurance and fuel costs would be lower.
- Reduced parking costs. Autonomous taxis could reduce parking demand by 90%.
- Increased fuel efficiency and reduced pollution, particularly on platooning schemes.
- Car sharing will be made possible: *vehicle-sharing* and *ride-sharing* will appear as new business models.

According to the experts' opinion [170], autonomous vehicles will likely be expensive novelties during the 2020s and 2030s, interesting only for non-drivers or long-distance drivers. However, they are expected to become widespread by the late 2030s or 2040s.

CLASSIFICATION

In any case, the process is always seen as a gradual change to be completed over many years (or perhaps decades) until achieving a system capable of driving autonomously in all conceivable situations. In this regard, SAE International⁴ proposed a taxonomy intended to understand the process that is needed, from a technical point of view, for the transition towards the new paradigm [233]. The classification relies on the following definitions:

DYNAMIC DRIVING TASK (DDT): All of the real-time operational and tactical functions required to operate a vehicle in on-road traffic (e. g., steering control, acceleration, and deceleration).

OBJECT AND EVENT DETECTION AND RESPONSE (OEDR): Subtask of the DDT that consists of monitoring the driving environment (detecting, recognizing, and classifying objects and events and preparing to respond as needed).

DDT FALLBACK: The response to perform the DDT or, at least, a minimal risk condition after the occurrence of a system failure.

⁴ SAE International is a United States-based automotive standardization body formerly known as the Society of Automotive Engineers (until 2006).

L.	NAME	DDT			
		MOTION	OEDR	FALLBACK	ODD
0	No Driving Automation	Driver	Driver	Driver	n/a
1	Driver Assistance	Driver & system	Driver	Driver	Limited
2	Partial Driving Automation	System	Driver	Driver	Limited
3	Conditional Driving Automation	System	System	Fallback-ready user	Limited
4	High Driving Automation	System	System	System	Limited
5	Full Driving Automation	System	System	System	Unlimited

Table 1.1: Summary of levels of driving automation, as defined by SAE International [233]

OPERATIONAL DESIGN DOMAIN (ODD): The set of specific conditions under which a given driving automation system is designed to function (e.g., geographic limitations or particular driving modes).

Automated driving systems are classified into different levels according to the degree of automation reached regarding these definitions. A summary is reported in Table 1.1.

Some **ADAS**, such as adaptive cruise control or lane-keeping assistance, are classified as Level 1 systems, provided that can act on any vehicle control (i.e., acceleration, brake, or steering). If the control can be performed over all controls at once, thus taking full responsibility for lateral and longitudinal vehicle motion control, the system is rated as Level 2. Some autopilot systems have emerged recently on commercially-available vehicles, such as Tesla Autopilot⁵ and Mercedes-Benz Drive Pilot⁶.

However, as of the date of writing, there are no Level 3 production vehicles available to consumers. Audi aims to offer its Audi AI Traffic Jam Pilot⁷ system in the new A8 sedan, but they are reportedly waiting for legal approval in many countries [133]. Level 4 is currently

⁵ <https://www.tesla.com/autopilot>

⁶ <https://www.mercedes-benz.com/en/mercedes-benz/innovation/the-new-e-class-on-the-road-to-autonomous-driving-video/>

⁷ <https://www.audi-technology-portal.de/en/electrics-electronics/driver-assistant-systems/audi-a8-audi-ai-traffic-jam-pilot>

restricted to some prototypes, whereas Level 5 remains a long-term goal.

The gap between Levels 2 and 3 (and above) is determined by the capability to perform the OEDR tasks; in other words, the ability to monitor the environment and deliver this information in such a way that it can be useful for vehicle control. The current effort is therefore geared towards perception systems with the capability to understand the traffic environment by making the most of data from exteroceptive sensors.

For a more in-depth reflection on autonomous driving concerning its technical, legal, and social dimensions, the reader is referred to the exhaustive 2016 open-access book *Autonomous driving* [180].

1.3 PERCEPTION SYSTEMS

Machine perception is a critical enabler for autonomous driving. The vehicle must be able to perceive and interpret its surroundings and use this information to carry out safe actions. These tasks are the responsibility of the perception system. The perception system is composed of two types of components: sensors (hardware) and algorithms (software).

1.3.1 Sensors

CLASSIFICATION A wide variety of sensors are used in automotive applications to obtain meaningful information about the vehicle and its environment. Proprioceptive sensors measure the internal status of the vehicle, such as the wheel velocity, acceleration or rotational velocity [18], and are essential for almost every active assistance method. However, this thesis focuses on exteroceptive sensors, which are used to acquire information from outside the vehicle.

TECHNOLOGIES Typical exteroceptive sensors used for environment perception in automotive applications are described below [273]:

RADAR: Its operating principle is based on the emission and subsequent reception of radio beams. Not only is it able to measure distances to obstacles, but it also can take advantage of the Doppler effect to measure their speed. Moreover, radar technology is highly robust to weather conditions. On the negative side, radar permits only low solid angle resolution with an acceptable antenna size [272].

LIDAR: Conceptually similar to radar systems, lidar⁸ devices use ultraviolet, infrared, or beams within the visible light spectrum

⁸ 'Lidar' is the acronym of Light Detection And Ranging; however, we will use the lowercase notation *lidar* throughout this document by analogy with similar words such as radar or sonar.

to localize and measure the distance of objects in space [101]. Lidar beams are more strongly attenuated by atmospheric conditions such as fog and rain. The introduction of 360°-capable laser rangefinders [238] was a huge leap forward for this technology and implied its widespread introduction in virtually every autonomous driving research platform. Despite its relatively limited resolution and high cost, lidar devices offer a precise distance estimation and a decent working range. However, concern has been raised that lidar cost and impact on the vehicle design could prevent its adoption for off-the-shelf production [35].

CAMERAS: Humans perceive about 90% of the information required for driving visually [251]. Some of this information has been indeed designed for visual perception and cannot be recognized by any other technology; e. g., traffic signs and lane markings. It, therefore, appears reasonable to use a system similar to the human eye for machine perception of the environment. Cameras, contrarily to the previous technologies, are passive devices that project the light reflected by objects in the 3D space onto a 2D image, thus reducing the available data by a dimension. Despite that and their low robustness against illumination, weather, and other external factors, cameras are inexpensive devices that can provide almost all the information relevant for vehicle control.

STEREO CAMERAS: While stereoscopic vision systems are indeed a subset of the previous category, they merit separate mention since they provide geometrical information on top of the appearance data that is common to all vision systems. These devices leverage epipolar geometry [116] to obtain the 3D coordinates of visible points when two views are available. If the two views are retrieved from two cameras facing the same direction and separated a known distance (*baseline*), calculations are simplified, and the depth coordinate can be straightforwardly obtained. It should be noted, though, that accuracy in the depth estimation is far from perfect and, moreover, depends heavily on the distance from the camera.

A summary of sensor technologies used in autonomous driving can be found in [227]. Fig. 1.8, extracted from that survey, represents the properties of the different sensor technologies cited above in comparison with a hypothetical *perfect sensor* fulfilling all the desirable characteristics.

Generally, sensor setups in vehicles feature a combination of devices aimed to take advantage of the complementary properties of the different sensor modalities, on the one hand, and to provide redundancy, on the other hand. *Data fusion* techniques are then used to exploit the advantages obtained by having several sources of information [194].

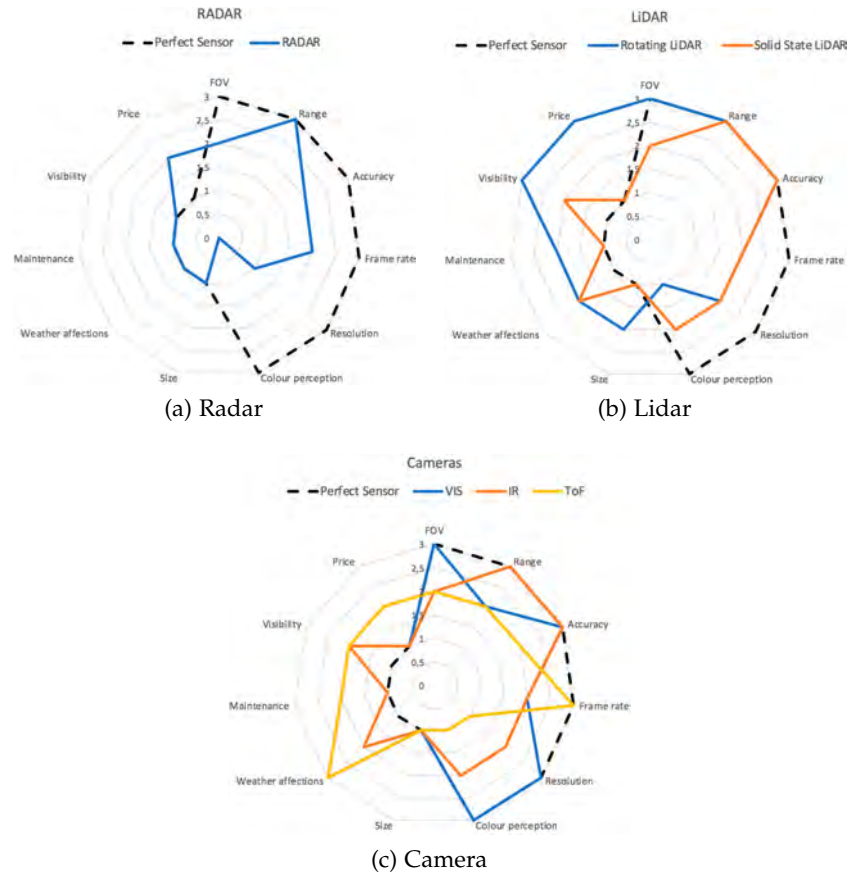


Figure 1.8: Features of the sensor modalities used in environment perception systems. Figure by Rosique et al. from [227] (license CC BY 4.0)

1.3.2 Algorithms

ENVIRONMENT MODELS

The result of the perception system is usually seen as a dynamic vehicle environment model in which individual dynamic motion models represent the vehicle itself and all other road users [58]. This model should also include all pertinent elements of the infrastructure, such as traffic signs/lights, as well as structuring elements such as road markings, crosswalks, or curbs.

In general, information retrieved by perception systems can be represented in two different ways [59]:

OBJECT-BASED REPRESENTATIONS: This kind of representation describes the dynamics of all relevant objects and infrastructure elements in the proximity of the ego vehicle. Each object is assigned a set of variables that usually include position, speed, and 3D object dimensions, and these variables are updated continuously using sensor measurements and filtering (tracking) processes [275].

GRID-BASED REPRESENTATIONS: They use grid maps to divide the environment into fixed, identically-sized cells. The vehicle navigates across this grid, and the perception system provides information on which cells are free and which ones are occupied by an obstacle, providing valuable insight for the subsequent planning algorithms [179].

Whichever model representation is used, detection and localization of objects (namely, road users and infrastructure elements) in the surroundings of the vehicle is perhaps the most critical task of the perception system. For a proper situational awareness, objects should be not only acknowledged but also assigned a semantic meaning; that is, a classification.

Humans can perform this task effortlessly, even at an early age. Machine perception systems, on the other hand, encounter many difficulties with the current state of the technology. Although computer vision is a well-established discipline that has experienced significant advances since its foundation in the 1960s [232], traffic scenarios remain as one of the most challenging applications due to their complexity and lack of structure. Additionally, automotive applications impose tight requirements that increase the intricacy of the problems.

Particularities of these kinds of applications are countless; to mention but one example, Schiele and Wojek [236] established a list of requirements that must be taken into account when designing an onboard pedestrian detection system. The list can be straightforwardly applied to almost every application involving perception in vehicles, especially those using computer vision:

RESOLUTION AND SCALE: Resolution and Field Of View (FOV) of sensors determine the amount of representable information. High-resolution data facilitate the perception tasks and enhance their performance; however, there is a limit on the amount of information that can be processed in order to provide an immediate response.

ROBUSTNESS: Functionality should be achieved in various weather and visual conditions. Furthermore, different instances of the same entity can present dramatic differences in appearance in their representations on sensor data.

VIEWPOINT INVARIANCE: The angle of the camera relative to the instances of interest is variable, and the performance should not be severely affected by this fact.

PARTIAL OCCLUSION: Occlusions between objects are unavoidable in traffic scenarios, especially in complex urban situations.

POSE ESTIMATION: Meaningful decisions regarding the control of the vehicle often require determining not only the presence of

OBSTACLE
DETECTION

CHALLENGES

traffic elements and road users but also a precise estimation of their pose.

2D VERSUS 3D MODELING: While visual information is necessarily represented on a 2D space, global 3D coordinates providing the exact position relative to the own vehicle are often needed.

MACHINE
LEARNING
TECHNIQUES

Machine perception is currently based on relatively complex models of the entities of interest. These models are, in almost every case, learned from examples using machine learning techniques, which are aimed at optimizing the performance of the inference process according to the available samples.

Machine learning, in particular, and Artificial Intelligence (AI), in general, is currently evolving at an unprecedented and most likely increasing rate. Medium- and long-term consequences are still to be determined, but automated systems have been already able to match human performance for some specific tasks (e. g., large-scale classification [119]) that were labeled as virtually impossible for a machine barely a decade ago. Therefore, it is only natural that perception systems aimed at automated driving can benefit from these advances in order to build reliable, accurate environment models to underpin decision making.

1.4 OBJECTIVES

This thesis is framed within the context of perception systems for automated vehicles. The main goal is to investigate the possibilities of DNNs to advance towards a full understanding of the scene around the vehicle by automated driving systems.

Among the different cues required to that end, this thesis focus on object recognition, which is recognized as one of the most critical tasks. This work intends to bring two worlds together: AI, where DNNs have brought dramatic advances, and ITS, which will inevitably come into the spotlight in the decades to come.

To that end, the following objectives are set:

- To address some issues associated with vehicle sensor setups, such as calibration and information representation, allowing an adequate use of the obtained data.
- To study the adequacy of existing all-purpose detection methods for onboard object detection in images.
- To propose specific tuning intended to optimize the performance of these algorithms and deal with the specific challenges posed by traffic scenarios.
- To enhance the detection framework with additional inference tasks that extend the capacities of the perception subsystem to meet the requirements of higher-level navigation modules.

- To perform an extensive analysis of the parameters of the integrated approach to offer different operating points according to the required trade-off between performance and speed.
- To explore creative alternatives to enlarge the available training data and to quantify their effect on the accuracy of the approach.
- To map the information about the obstacles from sensor data to the real 3D space around the vehicle, introducing spatial reasoning from capable sensor modalities. This way, meaningful information is made available for further planning and navigation reasoning.
- To push the detection framework to the limit by using data from sources significantly different from images as input, thus enabling redundancy and robustness in the perception functionality.

These tasks will be addressed from the point of view of their fit into the rest of the applications in the perception stack, as well as the actual needs of the planning and navigation modules.

As the subject of study has a clear applied nature, conclusions are mainly based on experimental analysis. Real-world data will be extensively used, principally from publicly available datasets, which allow fair evaluation and comparison with other state-of-the-art methods.

1.5 OUTLINE OF THE DISSERTATION

This document is structured in seven chapters, including this one, and one appendix. Each chapter (except this one) begins with a brief introduction and ends with some concluding remarks putting together the key findings in that phase. The content of each chapter is summarized below:

- CHAPTER 1 has given an introduction to the thesis by presenting the problems associated with transportation and introducing automated vehicles as a possible solution. The objectives have also been set.
- CHAPTER 2 provides some historical background of automated driving, on the one hand, and a survey of state-of-the-art works related to the matter of research of this thesis, on the other hand.
- CHAPTER 3 deals with the sensors which provide the data from which useful information is inferred. Management of raw data, as well as sensor calibration (including an original calibration method), are discussed.
- CHAPTER 4 introduces the detection meta-architecture that will be used in the remaining chapters. Some novel ideas to enhance

its performance are proposed, along with a study of the influence of training data.

- CHAPTER 5 is devoted to one of the main contributions of the thesis: an extension of the detection framework intended for viewpoint estimation. Extensive experimental results are provided to assess the joint detection and orientation estimation performance, and to analyze its sensitivity against different factors.
- CHAPTER 6 presents two different approaches to introduce spatial reasoning into the detection framework: one based on stereo vision and the other, which is complementary to the central development, on lidar data.
- CHAPTER 7 draws some conclusions from the presented work and proposes future lines of research based on the findings of the research.
- APPENDIX A briefly introduces some applications aimed to provide complementary information for autonomous driving, to which the author of this thesis contributed. Although they fall outside the scope of the central theme of this work, these developments give an overview of additional functionalities that must be considered for autonomous driving.

RELATED WORKS

Machine perception for vehicles has been a strong focus of attention for researchers since the end of the 20th century. Computer vision was soon applied to the problems that arose in the development of automated vehicles [20], and advances in sensor technology led to the integration of new sources of data, such as radar or lidar devices.

The identification of the obstacles relevant to the proper operation of the vehicle has always been a primary line of research in the field of perception systems for vehicles. Methods have been evolving over the years, fed by the progress in computer vision and AI. After years of narrowly application-tailored approaches, the landscape is currently dominated by Deep Neural Networks (DNNs), which enable representation learning without relying on specific-domain priors [17].

An accurate representation of the location and semantic meaning of the relevant instances around the vehicle permits higher-level reasoning and, ultimately, decision-making about control commands to be sent to the actuators.

Supervised learning approaches, which are the basis of almost every machine perception approach today, require a set of samples representative of the real-world use of the function. *Feature learning* methods, such as DNNs, are even more data-intensive since the complexity (or capacity) of the models makes them particularly prone to have low *bias* but high *variance* within the classic bias-variance tradeoff [138]. In this regard, the appearance of public datasets has been instrumental in the progress of machine perception and, not least, has allowed fair benchmarking between methods developed all around the world.

In this chapter, a review of the issue of object perception in traffic environments is provided, along with a survey of relevant approaches proposed in the literature to address it. A comprehensive view of methods is given, along with a mention of the technical aspects that must be taken into account in the process, covering from the sensor selection to the dataset availability.

2.1 HISTORICAL AUTONOMOUS DRIVING PLATFORMS

A significant number of autonomous driving projects have been developed over the years, featuring different design philosophies for the perception stack. This section provides a brief review, with a particular focus on the sensor selection and the motivation behind it.

This chapter includes content from [109].

The idea of autonomous driving has been pursued for decades. The first projects in the 1980-90s included Navlab [260], by the Carnegie Mellon University, and PROMETHEUS, funded by the European Commission [283]. However, the field gained its real momentum in the 2000s, during the *Grand Challenges* held by the Defense Advanced Research Projects Agency (DARPA) of the United States in 2004, 2005, and 2007.

The 2005 DARPA Grand Challenge, the first one to be successfully finished, proclaimed Stanley, from Stanford University, the winner. Stanley's sensor setup included five laser rangefinders pointing forwards along the driving direction of the vehicle (Fig. 2.1), a color camera for long-range road perception, and two radar sensors [261]. The perception system aimed to identify drivable surfaces in the desert surface; computer vision was used to perform the detection in the far range [51]. Other teams used vision sensors as well; for instance, the TerraMax truck¹ relied on a three-camera variable-baseline stereo system to identify obstacles in front of the vehicle [33].

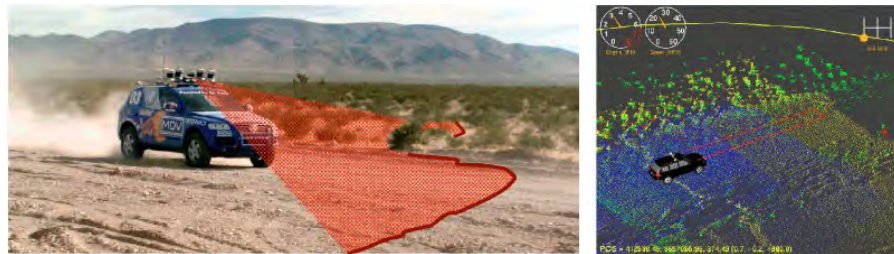


Figure 2.1: Lidar-based terrain detection featured by Stanley (Stanford University) during the 2005 DARPA Grand Challenge [261] © 2006 Wiley Periodicals, Inc.

Boss, from the Carnegie Mellon University, was the winner vehicle of the 2007 DARPA Urban Challenge [265]. The sensor setup was made of four kinds of lidar devices with different characteristics (including a 360°-FOV, 64-layer Velodyne HDL-64), radars, and High Dynamic Range (HDR) cameras with a 45° FOV. The mounting location of the sensors is depicted in Fig. 2.2.

Regarding other participants, the TerraMax vehicle took part in this edition as well and was again heavily dependent on vision algorithms [41]. The vehicle was endowed with a vast set of cameras [31]: a trinocular stereo system to perform obstacle and lane detection, two twin stereo systems to monitor the area close to the truck, two lateral cameras to detect oncoming vehicles at intersections [32], and a rear-view system to monitor the lanes next to the vehicle looking for

¹ The TerraMax vehicle was developed by a team composed by Oshkosh Corporation, a military vehicle manufacturer from the USA, and the VisLab of the University of Parma, Italy, among other industrial companies. Unlike the rest of participants, the TerraMax team entry was a 30 000 pound truck.

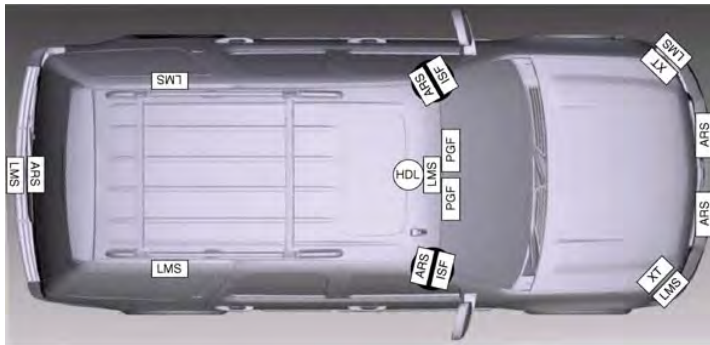


Figure 2.2: Sensor setup featured by Boss (Carnegie Mellon University) during the 2007 DARPA Urban Challenge. The set of sensors included GPS/IMU (APLX), lidar (LMS, HDL, ISF, and XT), radar (ARS), and cameras (PGF) [265] © 2008 Wiley Periodicals, Inc.

overtaking vehicles. FOVs corresponding to the different cameras are shown in Fig. 2.3. Another significant approach was the AnnieWAY entry, which was based on the use of a high-resolution Velodyne HDL-64 lidar supported by some complementary lidar devices for specific maneuvers [144].

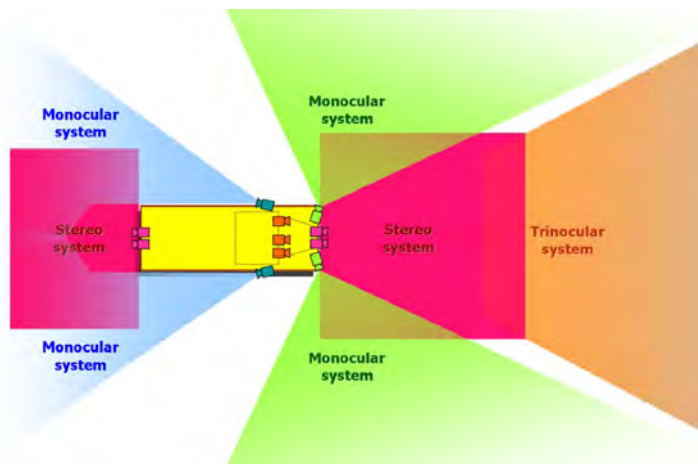


Figure 2.3: Sensor setup featured by TerraMax (Vislab and others) during the 2007 DARPA Urban Challenge [31] © 2010 IEEE

The VisLab research group (established in Parma, Italy) was responsible for the design of the perception system of the TerraMax vehicle in both entries, but their expertise dates back to the 1990s [20]. One of their most recent prototypes is the BRAiVE automobile, where cameras were extensively used over other options, such as lidar devices, in an attempt to deliver a close-to-market approach [30]. BRAiVE's sensor suite is based on ten cameras mounted all around the vehicle: four in the frontal part, looking forward; two over the front wheels, looking sideways; two in the rear-view mirrors, facing backward; and

two over the license plate, also looking backward. Cameras are used to perform various tasks such as traffic sign recognition, lane detection, vehicle detection, and blind-spot monitoring. The BRAiVE platform completed a 13 km public demonstration on public urban roads and freeways in Parma in 2013: the PROUD test [34].

STILLER'S
RESEARCH GROUP

Another primary focus on autonomous vehicle research is the research group lead by Christoph Stiller at the Karlsruhe Institute of Technology (KIT), responsible for the AnnieWay entry on the 2007 DARPA Grand Challenge. The BERTHA prototype, developed in collaboration with Daimler AG, dispensed with the lidar device and featured instead six radar devices, a 35 cm-baseline stereo camera, and two wide-angle monocular color cameras, one looking forward and the other one looking backward [80]. The stereo system was used for lane recognition and 3D scene analysis; the forward-looking wide-angle camera was aimed at traffic light and pedestrian recognition in turning maneuvers; and finally, the backward-looking camera was employed for feature-based localization [299]. A 103 km public test on the Bertha Benz Memorial Route, linking the German cities Mannheim and Pforzheim, was driven autonomously using this sensor setup [298]. Based on the same platform, BerthaOne is additionally endowed with vehicle-to-everything (V2X) communication capabilities [257].

2.2 DATASETS

Automating the driving-related tasks requires the design and implementation of algorithms able to represent reality through a model. As mentioned in Sec. 1.3.2, the creation of this model often involves the use of labeled samples provided by one, or more, datasets. In addition to enabling the re-use of labels, avoiding the duplication of the tedious labeling effort, public datasets are a convenient way to benchmark algorithms by introducing a common assessment framework. Indeed, it is now common that authors of datasets provide the means (e. g., evaluation servers) to promote the comparison of algorithms in the scientific community.

The spreading of deep learning techniques, whose models are endowed with a high degree of complexity and, therefore, require more training data, has led to the introduction of different datasets, created by researchers around the world, and focused on different tasks. Below are some of the most significant datasets currently in use for training and evaluation of perception algorithms.

2.2.1 Object recognition

PRE-DEEP
LEARNING
DATASETS

Classical object classification algorithms were based on handcrafted features and were, often, specific to a single kind of object. Among the different kinds of objects, pedestrians have been frequently a focus

of interest in the computer vision literature due to their importance and relative difficulty to be detected. Typical pedestrian datasets for methods of this kind included INRIA [53], Caltech [61], and Daimler [65], among others. Datasets have been proposed even for the problem of detection at nighttime, as the LSI FIR dataset [196], which provides images from an infrared camera. Pedestrian datasets usually featured cropped image patches containing pedestrians in different poses and backgrounds, intended to be used as positive examples for a pedestrian detection system.

The last years have seen the rise of a panoply of benchmarks with a more global orientation, aimed to provide data of a variety of categories. Thus, the PASCAL Visual Object Classes (VOC) Challenge [71], organized annually from 2005 to 2012, evolved from 4 classes in its first iteration to 20 classes and more than 11 000 images in its last version. This set of classes included a wide variety of labels such as *aeroplane*, *dog*, *table*, and *person*. A leaderboard with the top-performing algorithms was provided [70], enabling a fair comparison.

The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [231] was the first in a new generation of datasets with a massive number of classes. It includes object category classification and detection on hundreds of object categories (up to 1000) and millions of images. The challenge has been run annually since 2010 and has arguably enabled the progress of deep learning because of its vastness [154], which allows for the generation of general-purpose features. Due to the magnitude of the dataset, crowdsourcing was used to collect large-scale annotations.

Sponsored by Microsoft and Facebook, among others, Common Objects in COntext (COCO) [169] is one of the latest additions to this group of massive datasets. It features per-instance segmentations of 91 different object types in 328 000 images, enabling assessment of bounding box and segmentation detection algorithms.

Open Images Dataset [153] is probably the most extreme case of this trend. It consists of approximately 9 million images annotated with image-level labels and object bounding boxes for 600 classes.

A handful of datasets have gone a step further and have included spatial reasoning in 3D; for instance, PASCAL3D+ [282], an extension of the original PASCAL VOC dataset; SUN RGB-D [247], which provides data from an RGB-D sensor; or *Flying Chairs* [62], synthetic data aimed at training of DNNs.

2.2.2 *Driving environments*

Although general-purpose datasets provide an extensive set of samples which is valid to train onboard algorithms, autonomous driving poses specific challenges that are not entirely taken into consideration in catch-all computer vision datasets.

KITTI The KITTI Vision Benchmark Dataset [89], created by the KIT and the Toyota Institute, contains data recorded from a moving vehicle while driving around the German mid-size city of Karlsruhe. Data include RGB images and lidar points that are profusely annotated to enable benchmarking of algorithms performing stereo matching, scene flow, odometry, object detection, and object tracking [90] (Fig. 2.4). It has been later extended to include road segmentation [83] and semantic segmentation [2]; besides, the scene flow evaluation has been updated [182]. The impact of the KITTI dataset on the ITS research community is immeasurable, as evidenced by the fact that it is still today the baseline for a variety of critical tasks such as object 3D localization.

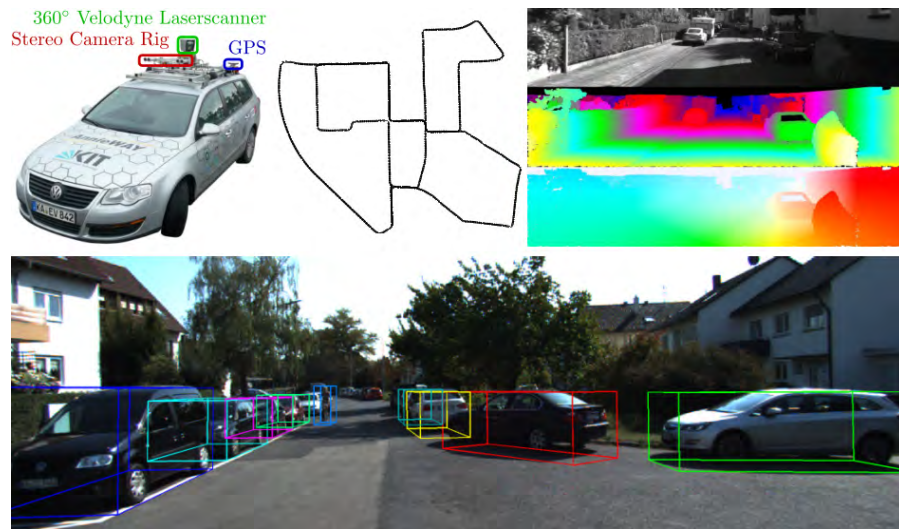


Figure 2.4: Recording platform, trajectory, disparity and optical flow maps, and 3D object labels featured from the KITTI dataset [90] © 2012 IEEE

CITYSCAPES Following the recent progress on semantic segmentation techniques, the Cityscapes dataset [49] provides pixel-level and instance-level semantic annotations of images recorded from a moving vehicle in 50 different Central Europe cities. With 5000 finely-labeled images, the dataset has marked a milestone in semantic segmentation research. On the other hand, CityPersons [294] is built on top of the Cityscapes dataset and provides bounding-box labels for the set of pedestrians, whereas the EuroCity Persons Dataset [28] seeks to increase the diversity of CityPersons regarding places (around different European countries) and times of the year.

OTHERS The last two years have seen the arrival of a multitude of large datasets tailored to autonomous driving. Inspired by Cityscapes, the Mapillary Vistas dataset [191] provides pixel-wise labeled images from all around the world, taken with different devices (e. g., mobile phones or action cameras). Oxford RobotCar [176] contains 100 repetitions of the same route through Oxford, captured throughout a year, thus

encouraging the robustness of the algorithms against weather and illumination conditions. The ApolloScape dataset [132], sponsored by Baidu, features a broad set of images labeled for scene parsing, car instance detection, lane segmentation, and self-localization. Finally, the Berkeley Deep Drive (BDD) 100K [290] dataset claims to be the largest and most diverse driving video dataset and is annotated with image tags, object bounding boxes, drivable areas, lane markings, and full-frame instance segmentation.

Some datasets are focused on particular tasks within the ITS field. For instance, Yang *et al.* [288] dealt with the particular case of fine-grained classification and model verification of vehicles. On the other hand, a dataset developed in the Laboratorio de Sistemas Inteligentes (LSI), Semantic Annotated University Campus Environment (SAUCE) [16], is aimed at the specific case of navigation on off-road environments using pixel-wise-labeled images obtained in the surroundings of a Spanish university campus. On a different level, TorontoCity [270] offers different perspectives of the Toronto metropolitan area, including airborne data captured from airplanes and drones.

The specific task of road sign recognition is of particular interest. In this line, traffic light detection and classification has been nourished by different datasets such as LaRA [54], WPI [47], VIVA [140], and Bosch [12]. DriveU [81] is the latest addition to the group and shows that the interest on the topic remains undiminished. Likewise, traffic signs have also been the subject of some datasets such as the German Traffic Sign Recognition Benchmark (GTSRB) [250] or its detection-oriented counterpart, the GTSDb [129].

ROAD SIGNALING
RECOGNITION

2.2.3 Synthetic datasets

Most existing datasets are built upon data obtained from real sensors, which must be annotated afterward. However, image labeling is a cumbersome and time-consuming task. Advances in the video games industry have fostered the use of virtual worlds to generate datasets that are subsequently used to train AI algorithms. Since worlds are generated by a visual engine, ground-truth labels are straightforwardly available so, once the framework setup is finished, virtually unlimited samples can be obtained. However, the usefulness of these approaches in the computer vision field remains an open question due to the inevitable difference compared to the data obtained with real sensors. This is formally acknowledged as the *domain adaptation* problem [174].

One of the most relevant works regarding autonomous driving research is the SYNTHetic collection of Imagery and Annotations (SYNTHIA) dataset [226], aimed at semantic segmentation. Authors claim that virtual images can be used to enlarge real datasets and, therefore, improve the performance of DNN-based algorithms. The set of samples was later extended with the SYNTHIA-SF dataset [121],

SYNTHIA AND
CARLA

representing extreme cases of road slopes. The simulator used to generate the samples in SYNTHIA was recently released with the name of Car Learning to Act (CARLA) [63]. Based on the Unreal Engine 4 graphical engine, CARLA offers boundless possibilities for researchers to recreate different road conditions (e. g., weather or illumination) and sensor modalities (e. g., cameras or lidars) in order to train and test their perception and control algorithms.

KITTI
EXTENSIONS

Several works have been devoted to the extension of the real-world KITTI dataset [89]. Virtual KITTI [84] tried to replicate the existing frames and create new ones to enlarge the extension of the dataset. On the other hand, KITTI-360 [3] uses augmented reality to populate the uncrowded frames of the KITTI dataset with additional instances of objects.

OTHERS

Computer GRAPHic generated synthetic Traffic Scenes (COnGRATS) [22] uses Blender to generate annotated frames with depth, motion, and semantic labels. Authors claim that the dataset can be used to evaluate the performance of computer vision algorithms, without an apparent difference with real-world data [23]. Richter *et al.* [215] took advantage of the Grand Theft Auto V video game to automatically generate labeled frames aimed at semantic segmentation, instance segmentation, 3D scene layout, visual odometry, and optical flow.

Lately, DNNs have been used to generate realistic virtual frames that could be eventually used to train vision algorithms. An example is the Cascaded Refinement Network [42].

2.3 SENSOR CALIBRATION

Most perception algorithms assume that an accurate estimation of the sensor calibration is available. This prerequisite enables establishing a relationship between the real-world objects and their projections onto the image (intrinsic calibration), on the one hand, and combining data from different sensors that are inevitably located at different positions within the vehicle (extrinsic calibration), on the other hand. Calibration is, generally, a cumbersome and time-consuming task; however, the performance of every application making use of spatial reasoning depends strongly on the accuracy of the estimated calibration parameters.

2.3.1 Camera intrinsic parameters

Most calibration frameworks currently available, such as the Bouguet's Camera Calibration Toolbox for Matlab² and its C implementation included in OpenCV [26], rely on the method proposed by Zhang [295]. A fiducial pattern is used to estimate the set of intrinsic parameters through a procedure involving a closed-form solution (analytical solu-

² http://www.vision.caltech.edu/bouguetj/calib_doc/

tion) and a maximum-likelihood estimation (non-linear optimization technique). The method requires a set of images where the pattern is detected and the coordinates for each corner extracted. The problem is then posed as a series of homographies between the 2D pattern and its projection on the image.

Zhang’s algorithm can also be applied to other fiducial patterns different from the classic checkerboard, such as squared planar markers (e. g., ArUco [223]) or a circles grid, in this latter case reportedly reducing the number of pattern snapshots required to form a well-posed equation system [198].

2.3.2 Extrinsic parameters

Vehicle sensor setups are usually composed of sensors belonging to different modalities (e. g., cameras and lidar/radar rangefinders) whose data must be combined to perceive the environment. Frequently, sensors have overlapping fields of view, and the advantages conferred by their joint use come from the ability to make correspondences between both data representations. To that end, the relative pose of the sensors, given by their extrinsic parameters, must be accurately determined through a calibration process.

The camera-to-range calibration problem is, probably, the most frequently addressed. In the most usual approach, calibration is seen as a process to be performed in a controlled environment, using fiducial markers. Early methods required manual annotation to some extent [235]; however, such a laborious and repetitive task has been naturally a subject of intense research aimed at its full automation.

Most automatic methods are based on establishing correspondences between data from different sensors and, therefore, unambiguous pattern instruments have been used as calibration targets: polygonal boards [199], triangular boards [55], spheres [204], and others. Nevertheless, planar targets are frequent [165], since they are easily visible in both range and intensity data.

Although early methods were designed to deal with limited range information [157], advances in lidar technology have led to devices with the capability to provide a dense 3D point cloud, and calibration methods have advanced accordingly. Assuming a common field of view for the sensors, Geiger *et al.* [91] presented a calibration method based on a single shot per sensor using a setup made of multiple planar checkerboards. The method was tested with two cameras and a multi-layer range sensor. While accurate enough, the setup needed to perform the procedure is rather unwieldy, as shown in Fig. 2.5.

On the other hand, Velas *et al.* [266] use a custom calibration target composed of a plane with four circular holes to provide an estimate of the extrinsic calibration relating a scanner device and a camera. The shape of the target was chosen so that it could be easily detected in

CALIBRATION
PATTERNS

MULTI-LAYER
LIDAR DEVICES



Figure 2.5: Calibration setup used by Geiger *et al.* [91] © 2012 IEEE

both the camera image and the data from a commercial 360° lidar device.

TARGETLESS METHODS

Recently, several methods that perform calibration in the absence of fiducial targets have been proposed. These methods extract features from natural environments; for instance, Moghadam *et al.* [186] use linear features to determine the transformation between sensors. The method is aimed at indoor environments, where linear landmarks are plentiful. In vehicular applications, the ground plane and traffic users have been used to perform camera-lidar calibration [217]. The particular case of non-overlapping fields of view has also been considered [239], [240].

DEEP LEARNING METHODS

DNNs have been employed lately to perform camera-lidar calibration with little or no supervision. RegNet [237] and CalibNet [137] are two of the flag-bearers of this trend. The networks used by these methods are trained to detect and correct miscalibrations using real data where near-ground-truth extrinsic parameters are available (e. g., the KITTI dataset).

ASSESSMENT

A related issue is the assessment of calibration methods. Determining the ground-truth parameters expressing the relative transform between sensors is impractical, and evaluation is often based on custom metrics and inaccurate manual annotations. In this respect, Levinson and Thrun [161] proposed a measure expressing the degree of miscalibration based on the identification of discontinuities in a natural scene.

2.4 CONVOLUTIONAL NEURAL NETWORKS

Data from sensors is not straightforwardly interpretable by computers. A non-trivial processing stage is necessary to extract useful information from the raw data and feed the system higher-level modules.

For decades, this processing has been mostly based on machine-learning methods, which build a mathematical model of sample data in order to make predictions or decisions on unseen data [24]. However, conventional machine-learning technologies are bad at processing data in their raw form; they generally require an intermediate step, built by humans with proper domain expertise, to map the raw data into a suitable representation.

In contrast, representation learning methods aim at building mechanisms able to learn how to deal with raw data and automatically

discover the representations that are the most useful for the machine [17]. Progress in this field has been painfully slow until the emergence of Deep Learning (DL). This term is used to describe a set of representation learning techniques with multiple levels of representation, increasingly abstract, built with simple but non-linear mathematical operations.

DL has represented a significant step forward, not only for representation learning itself but, above all, for machine learning: representations discovered by DNNs have proved far more effective than its hand-crafted counterparts.

2.4.1 Historical evolution

Although multilayer neural networks, trainable through backpropagation, had been widely understood since the mid-1980s, the research community lost interest due to their proneness to get trapped in poor local minima [158]. In practice, poor local minima are not a problem in large networks, and this is one of the reasons why DNNs gained traction again around 2006, being applied to tasks such as digit classification [123]. The use of deep structures was made possible by the advances in GPUs, which accelerated the backpropagation process by 10 or even 20 times.

However, it was soon discovered that there was a particular type of DNN that was relatively easy to train and had an excellent generalization ability: the Convolutional Neural Network (CNN), sometimes referred to as *ConvNet*. Although structures similar to the CNN have been successfully used since 1990 [159], the turning moment took place in 2012, when Krizhevsky *et al.* [154] achieved performance levels for image recognition never seen before during the ILSVRC competition, mentioned before in Sec. 2.2.1. They proposed a CNN design, currently referred to as AlexNet, based on the efficient use of two GPUs and Rectified Linear Units (ReLUs) as non-linearities. The extraordinary performance showed by CNNs in this paper precipitated the rapid adoption of deep learning by the computer vision community.

CNNs are currently the method of choice for virtually all recognition and detection tasks, from image/video captioning [267] to cell segmentation in biomedical images [224]. Most major technology companies in the world are investing heavily in DL projects, and resulting products are already reaching the market (e. g., Google's computational photography³).

CNNs have a neuroscientific basis as some of the fundamental design principles of neural networks came from the brain function [100]. Thus, the *primary visual cortex* (V1) is arranged in a spatial map, contains many simple cells (non-linearities) and also contains many complex cells (pooling units), similar to the CNNs.

FIRST NEURAL
NETWORKS

CONVOLUTIONAL
NETWORKS

3 <https://ai.googleblog.com/search/label/Computational%20Photography>

2.4.2 Fundamentals

BASIC
STRUCTURE

CNNs are arrangements suitable to work with data that has an explicit grid-structured topology and to scale such models to larger sizes; therefore, they have been the most successful on a two-dimensional, image-like topology.

From an architectural point of view, CNNs are similar to conventional neural networks. The basic unit is the *neuron*. Neurons have weights and biases that are learned through an optimization procedure. Each neuron receives an input, performs a dot product (using the weights and bias), and, usually, is followed by a non-linearity (e. g., ReLU) [145].

The general architecture of a CNN is made of a series of stages, as depicted in Fig. 2.6. The earliest stages are made of convolutional layers, on the one hand, and pooling layers, on the other hand. Both structures are usually stacked together:

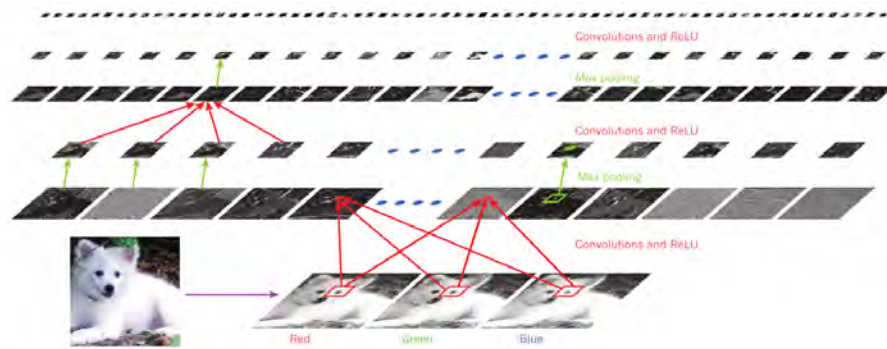


Figure 2.6: Convolutional Neural Network (CNN) [158]

CONVOLUTIONAL LAYERS: They produce *feature maps* from their input, which can be either another feature map or the raw RGB image. These feature maps are generated by discrete convolution of the input data with learnable filters banks (or *kernels*), whose size is typically limited to a small fraction of the spatial dimensions (height and width) of the incoming data. Each unit of a feature map is, therefore, the result of applying the weights in the filter bank to a local patch in the input matrix, and all units share the same filter bank. This design choice is supported by two facts [158]: firstly, local groups of values are frequently correlated since they form distinctive patterns; and secondly, kernels must be invariant to location, since patterns can appear anywhere on the image. Convolutional layers are followed by a non-linearity (e. g., ReLU).

POOLING LAYERS: They compute the maximum of local patches of each feature map. Adjacent pooling units use inputs shifted by more than one row or column, thus effectively decreasing

the size (height/width) of the feature maps. Pooling layers are intended to merge semantically similar features into one, so that differences in the relative position of patterns are omitted [158].

Generally, a CNN is made of various stages of “convolution + non-linearity + pooling” blocks, followed by several Fully-Connected (FC) layers, which perform as a *regular* neural network; that is, each learnable unit performs a weighted sum of the inputs (all of them). This implies, on the one hand, that the 2D structure of the input data is lost at this stage, and, on the other hand, that the last feature map (i. e., the input of the FC layers) must have a fixed size.

Finally, most CNNs perform a *softmax* operation at the last layer. The softmax function normalizes the output vector so that resulting values (σ_j) are in the interval $(0, 1)$, and they add up to 1, thus becoming a probability distribution. To that end, the following operation is performed over every element z_j of the K -length output vector, \mathbf{z} :

$$\sigma(\mathbf{z})_j = \frac{e^{z_j}}{\sum_{k=1}^K e^{z_k}} \quad (2.1)$$

2.4.3 Architectures for image recognition

CNNs can be straightforwardly applied to classification tasks; in this scheme, the output vector is aimed to represent the probability distribution over the possible categories. Therefore, the prediction provided by the network (i. e., the maximum element of the output) is the class label corresponding to the input image.

The model proposed by Krizhevsky *et al.* [154], now referred to as *AlexNet*, paved the way for a manifold of variations proposed to the very same end. AlexNet was a relatively simple model that consisted of 11×11 , 5×5 and 3×3 convolutions, *max* pooling, dropout, Stochastic Gradient Descent (SGD) with momentum, and, last but not least, ReLU activations, instead of the tanh or sigmoid functions that were used traditionally in neural networks. ReLU units are non-linear functions defined as the positive part of its argument:

$$f(x) = \max(0, x) \quad (2.2)$$

DNNs with ReLUs can be trained much faster than their equivalents with other non-linearities. On the other hand, AlexNet was designed to cope with the memory limitations of the GPUs at the time: the network was divided into two symmetrical parts aimed to be trained in different GPUs.

After AlexNet, numerous models were proposed. They are, to a large extent based on the same principles, but introduce variations aimed to increase the accuracy. Generally, as the complexity of the model increases, inference and training times grow, and more data is required

MODELS

ALEXNET

OTHER POPULAR
MODELS

to train the model while avoiding the occurrence of overfitting. Some popular models nowadays are described below:

ZF: The Zeiler and Fergus (ZF) model [291] is a slight variation on the original AlexNet model that modifies the first and second convolutional layers to improve the quality of the resulting features. The proposed changes were based on a study of the features obtained at different levels that makes use of a *deconvolutional network*.

VGG: VGG models [244] are named after the research group where they were developed, the Visual Geometry Group from the University of Oxford. VGG models are an evolution from the original AlexNet that replaced the large kernel filters at the earliest layers with multiple 3×3 filters stacked one after another. This uniform structure enables the generation of more complex representations; however, it also increases the number of parameters. VGG networks can adopt different configurations, being the 16 weight layers version (VGG-16) the most popular one.

MOBILENET: Efficient models specifically designed for mobile applications, MobileNets [130] introduce the concept of *depthwise separable convolutions*, which factorize a standard convolution into a depthwise convolution and a pointwise convolution, increasing the efficiency in the use of parameters. MobileNets have two hyperparameters to control the trade-off between computation time and accuracy, thus adapting to a range of applications. An enhanced iteration, MobileNetV2, was recently presented [234]. They make use of inverted residual structures to further increase the efficiency in memory usage.

INCEPTION: The Inception family of models was designed to explore the limits of the sophistication by using heavily engineered concepts and specific gimmicks to push the performance. The model has undergone several iterations. The main contribution of the original Inception [254] was the *Inception module*, which was made of several filters with multiple sizes operating on the same level. The idea was against the mainstream trend of increasing the depth of the models; they tried to increase their width, instead. Batch Normalization (BN) was later adopted [136] to smooth out the training procedure. Inception-v2 and Inception-v3 [255] were improvements over the baseline model based on the factorization of convolutions and the use of solutions such as an RMSProp optimizer or BatchNorm. Finally, Inception-v4 and Inception-ResNet [253] were an attempt to slim down the complexity of the model, on the one hand, and to include residual connections to improve the performance of the model,

on the other hand. The latter is inspired by the ResNet model described just below.

RESNET: Residual Neural Networks (ResNets) presented by He *et al.* [120] introduced the novel concept of *shortcut connections* to reformulate the network as residual functions with reference to the input layers, instead of unreferenced functions. The building block of the network, the residual block, is shown in Fig. 2.7. The idea was intended to solve the *degradation* phenomenon observed on very deep models, when increasing the number of layers leads to a drop in performance. The most popular variants are the 50-layer (ResNet-50) and 101-layer (ResNet-101) models.

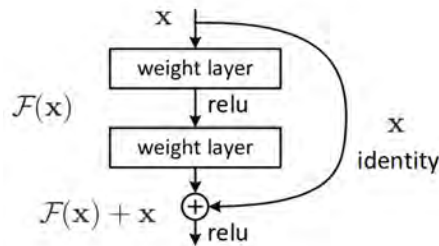


Figure 2.7: Residual block [120] © 2016 IEEE

Classification algorithms are typically evaluated on the ImageNet dataset [231], described in Sec. 2.2.1. A summary of the performance of the models described in this section is presented in Table 2.1, using the Top-1 and Top-5 errors as established by the ILSVRC challenge. Hence, the Top-1 error takes into account all the cases where the ground-truth class has not been predicted as the most probable one, whereas the Top-5 error relaxes the requirement for the ground-truth class to be among the five most probable categories.

BENCHMARK

Included results are purposely extracted from the corresponding original papers, and therefore the available metrics are not entirely homogeneous; for instance, results for modern networks are provided for the average of the classification probabilities over multiple (10–12) crops on the original frames, thus requiring several executions per image. Nevertheless, the evolution of the performance from the early AlexNet/ZF models is notorious.

Further details concerning CNNs will be described in Secs. 2.5.3 and 2.5.4, in regards to their application on detection frameworks.

2.5 OBJECT DETECTION IN IMAGES

Image classification aims to provide a single class label (e.g., *car*, *pedestrian*, or *cyclist*) when given an input image. However, limitations inherent to this task significantly restrain its applicability to real images: only one object is expected to be present in the image, and there

PROBLEM
STATEMENT

MODEL	CROPS	TOP-1 ERR.	TOP-5 ERR.
AlexNet [154]	1	40.7	18.2
ZF [291]	1	33.1	16.5
VGG-16 [244]	1	24.8	7.5
MobileNet [130]	1	29.4	—
MobileNetV2 [234]	1	28	—
BN-Inception [136]	1	25.2	7.82
Inception-v3 [255]	12	19.47	4.48
Inception-v4 [253]	12	18.7	4.2
Inception-ResNet-v2 [253]	12	18.7	4.1
ResNet-50 [120]	10	22.85	6.71
ResNet-101 [120]	10	21.75	6.05

Table 2.1: Classification error rates (%) on the ImageNet validation set for different classification models

is no information about its location within the frame. Object localization goes a step further and intends to estimate a set of coordinates determining the position of the object unambiguously. However, the limitation concerning the number of objects in the image still holds.

Meanwhile, object detection is intended to offer an estimate of the position and category of the objects represented in the image. As follows from the definitions above, object detection entails a higher degree of complexity than classification and requires the use of algorithms specifically designed for the application.

Until a few years ago, detection algorithms were based on classification methods applied sequentially over patches from the input image. However, the emergence of CNNs brought new methods where the search space was reduced, and others where the processing can be performed concurrently over all the areas in the image.

A brief review of the most relevant aspects considered in the literature about this issue is provided below, including some specific concerns regarding training and inference. For further insight, the reader is referred to the comprehensive review by Agarwal *et al.* [1], which covers almost every aspect of CNN-based object detection.

2.5.1 Historical evolution

Before the widespread adoption of CNNs, detection algorithms relied on the *sliding window* approach: crops of fixed size were extracted from all positions and scales of the image and fed to a classifier to determine the presence of an object.

Classical detectors used together with this approach were based on hand-crafted features, whose complexity remained tractable. Because

of this, they were suitable for repeated application, as required by the sliding-window procedure.

As feature selection was deemed as the most critical stage of a classification algorithm, a manifold of different features were proposed over the years. Two of them can be highlighted because of its relevance:

HAAR-LIKE FEATURES: First proposed by Viola and Jones in the popular Viola–Jones face detection framework [268], Haar-like features sum up the pixel intensities in adjacent rectangular regions and compute the difference between these sums.

HISTOGRAMS OF ORIENTED GRADIENTS (HOG): They were introduced by Dalal and Triggs for human detection [53], but showed compelling performance for other classification tasks, becoming the dominant approach for several years. The method consists of counting occurrences of gradient orientation in local areas of the image and is based on the intuition that local object appearance and shape is described by the distribution of intensity gradients and edge directions.

The Deformable Part-based Model (DPM) proposed by Felzenszwalb *et al.* [74] embedded HOG features into a structure that takes into account the relative positions of the different parts that make up an object. Since it solved some pressing problems related to the little robustness of HOG features against changes in pose, it was rapidly adopted and become the last widespread approach before the introduction of CNNs.

These and other features were used to perform classification through machine learning algorithms such as boosting [82], which seeks to convert a set of weak learners to a final strong classifier, and, mainly, Support Vector Machines (SVM) [50], aimed to find a representation where samples belonging to different categories can be easily discriminated. Classical detection algorithms were generally targeted to a single kind of object, so the detection was posed as a binary classification problem.

Since the groundbreaking work by Krizhevsky *et al.* [154] in image classification, the focus was on applying CNNs to object detection, thus avoiding the need for manually designing features. CNNs were first integrated into sliding-window approaches [241], but that paradigm led to suboptimal solutions. It soon became evident the necessity of designing tailored approaches to exploit the representation power of CNNs in object detection.

MODERN
DETECTION
APPROACHES

2.5.2 Meta-architectures

A large number of methods have been proposed in the computer vision literature to take advantage of CNNs in object detection. However,

TAXONOMY

Huang *et al.* [131] realized that virtually all modern convolutional detection systems were based on the same underlying principles. Hence, they can be analyzed as *meta-architectures* making use of a CNN acting as a *feature extractor*, often referred to as the *backbone*.

The feature extractor is, as its name implies, responsible for computing the features used for detection and classification. These features are gathered from the feature maps generated by a plain CNN, like those described in Sec. 2.4. However, the different methods differ in their use of the feature maps. Huang *et al.* found three main meta-architecture designs: R-CNN, R-FCN, and single-shot detectors.

2.5.2.1 R-CNN

R-CNN The Region-based Convolutional Neural Networks (R-CNN) paradigm was proposed in 2014 by Girshick *et al.* [96]. The method is based on the classification of previously proposed image patches using a CNN. The initial approach required to run the whole CNN for every image patch or *Region Of Interest (ROI)* [97], as depicted in Fig. 2.8. The proposals came from a classic segmentation method (selective search [263]) and were abundant. Unsurprisingly, the computational cost of the algorithm was prohibitive.

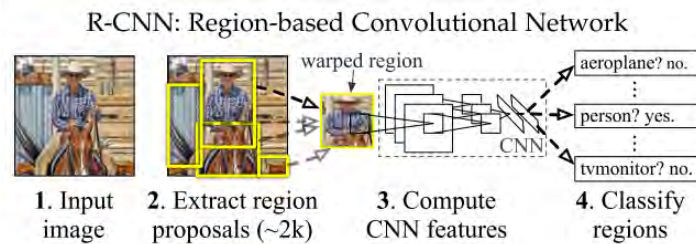


Figure 2.8: Region-based Convolutional Neural Networks (R-CNN) [97]
© 2016 IEEE

FAST R-CNN Soon after, the algorithm was largely improved to become *Fast R-CNN* [95]. Fast R-CNN was an important step forward since it only required to compute the feature maps once, and therefore all the objects could be classified with a single execution of the feature extractor, as shown in Fig. 2.9. The only layers that were not shared and therefore required as many executions as proposals were the FC layers. As described in Sec. 2.4, FC layers require a fixed input size; because of this, one of the most significant contributions of Fast R-CNN is the *ROI pooling* layer, which computes a fixed-length feature vector using an arbitrary size image patch as input. As a result, execution times became more tractable.

FASTER R-CNN Finally, in 2015, Ren *et al.* proposed *Faster R-CNN* [213], an end-to-end framework that dispenses with the external proposal generator. Faster R-CNN introduced the idea of Region Proposal Network (RPN), which

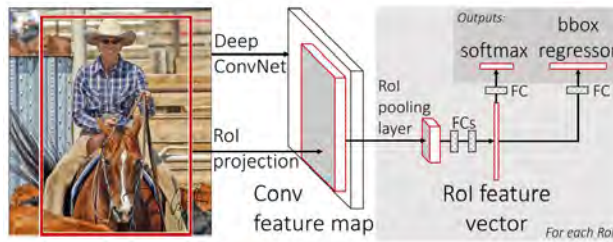


Figure 2.9: Fast R-CNN [95] © 2015 IEEE

is a small network responsible for determining which proposal boxes contain objects. As the RPN is extremely lightweight, a large number of proposals covering the whole image at different scales and aspect ratios are used, but only those containing objects are subsequently classified using the Fast R-CNN approach (Fig. 2.10). Faster R-CNN features a multi-task loss function that allows training both the RPN and the classification *head* simultaneously [214].

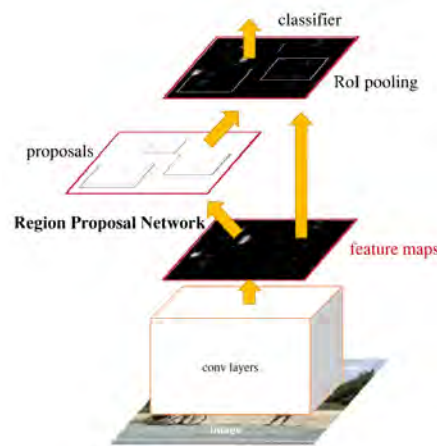


Figure 2.10: Faster R-CNN [214] © 2015 IEEE

The Faster R-CNN approach was recently extended to provide a semantic mask of the object within the detection box. The resulting method, known as Mask R-CNN [118], is currently one of the top-performing methods for instance semantic segmentation. Mask R-CNN also replaced the classic ROI pooling layer by an ROI align layer, which solved the problems of misalignments caused by the ROI pooling computations using bilinear interpolation.

2.5.2.2 R-FCN

Region-based Fully Convolutional Networks (R-FCN) [52] is another meta-architecture, based on Faster R-CNN, which aimed to reduce the per-ROI computation to a minimum. To that end, it got rid of the FC layers and therefore performed the pooling at the last layer of features prior to

prediction. The cropping mechanism was position-sensitive to incorporate translation variance, thus preventing the loss of information at the ROI pooling stage in Faster R-CNN.

2.5.2.3 Single-shot detectors

In contrast with the previous methods, where classification was performed in two stages, there is another family of approaches that avoid the proposal-specific stage and therefore pose the classification problem as an end-to-end computation where raw pixels are converted to bounding box coordinates and class probabilities.

YOLO The first single-stage method was *You Only Look Once (YOLO)* [210]. YOLO modeled the detection problem as a regression by dividing the image into a grid and letting the network estimate bounding boxes and class probabilities for each of the grid cells. YOLO has been further improved as YOLOv2 [211], which used multi-scale training, and YOLOv3 [212], where several design tweaks were added.

SSD In the same spirit, *Single Shot Detector (SSD)* [171] adopted the structure of the RPN introduced in Faster R-CNN and extended it to provide a proper classification, instead of the *objectness* score for which the RPN was designed. This way, the per-region classification could be entirely removed. To handle objects of different sizes, they also added additional convolutional layers to the feature extractor so that multi-scale feature maps were computed.

COMPARISON
WITH TWO-STAGE
METHODS

Generally, single-stage detectors are much faster than their double-stage counterparts, but their detection performance is notably worse. The drop in performance is particularly noticeable in small objects, which single-shot detectors struggle to find [131].

2.5.3 Training

Training is the process through which network weights are learned from labeled samples. The training procedure critically determines the quality of the resulting model. Several approaches concerning different aspects of the training process have been proposed in the literature; some of them are particular to the detection problem, but others are common to every method using DNNs.

TRANSFER
LEARNING

For instance, the pre-training of CNNs was soon identified as one of the critical aspects to improve the performance of recognition algorithms. The underlying concept behind this is that features generated by CNNs are so powerful that they can be useful for tasks different from the one for which they were trained. *Transfer learning* was quickly exploited for classification [209], and it is used in virtually every detection method to initialize the network weights of the feature extractor before starting the optimization process. However, recent works [117] have questioned this practice since similar results can be reportedly obtained by increasing the number of training iterations.

Those layers which are not initialized from a previous model should be assigned random values, as usual in neural networks. The most straightforward approach consists of drawing the random values from either a uniform or a Gaussian distribution. Parameters of the source distributions can be selected according to policies such as Xavier (or Glorot) [98] or MSRA (or He) [119] to guarantee the adequacy of the values.

WEIGHT
INITIALIZATION

Optimization, on the other hand, can make use of different strategies; however, SGD [230] and Adam [147] are still the most popular. Most training hyperparameters, such as the learning rate or the training schedule, are still selected by hit-and-trial. So is batch size, although the vast majority of detection algorithms use small batches to keep the GPU memory usage under control.

OPTIMIZATION
METHODS

Some techniques have been proposed to improve the effectiveness of input data. Most datasets have imbalanced classes; that is, there are many more samples from some categories than others. More importantly, detection algorithms are based on the classification of a large number of proposals, most of whom correspond to the background. Therefore, the imbalance is often a problem since the resulting model risks being biased towards the majority class. In this regard, Online Hard Example Mining (OHEM) [243] selects the worst-performing samples (hard examples) to calculate the gradients. On the other hand, *focal loss* [168] improved the detection performance by penalizing hard examples more strongly than the easy ones.

CLASS
BALANCING

CNNs are generally large networks prone to overfitting. Regularization methods try to avoid this phenomenon by discouraging learning an overcomplicated model. For instance, *dropout* [248] randomly removes units from the network during training. Batch Normalization (BN) [136] normalizes the layer inputs by the mean and variance computed within a training batch; this way, the internal covariate shift is reduced, and the gradients are less dependent on the scale of the parameters or their initial values, which can benefit generalization. Group Normalization (GN) [279] solves some drawbacks of BN, such as the dependency on the batch sizes, by dividing the channel into groups and computing the mean and variance within each group.

REGULARIZA-
TION

Increasing the amount of training data has a positive effect on the generalization ability of the models and prevents overfitting. In order to enhance the available data without using additional annotation efforts, *data augmentation* strategies can be adopted. These techniques apply transformations over the training images to artificially enlarge the existing dataset. The original AlexNet paper employed data augmentation by applying random translations, horizontal mirroring, and pixel-wise addition of multiples of the principal components of the images [154]. In general, typical transformations for data augmentations can be divided [258] into geometric (e.g., scale, resize, translation,

DATA
AUGMENTATION

rotation, or mirroring) and photometric (e. g., contrast, color, hue, or saturation).

2.5.4 Inference

NON-MAXIMUM SUPPRESSION

Object detection pipelines perform classification of a large number of proposals or anchors which are, to a greater or lesser extent, overlapped. Because of this, several bounding boxes are labeled as positive for each real instance in the image. A post-processing stage is, therefore, necessary to filter out all those boxes corresponding to the same entity except for one, which provides the most accurate representation of the real entity. This step is crucial to avoid the existence of duplicate detections that might degrade the performance of other modules down the pipeline (e. g., tracking).

Most approaches are based on the *greedy* Non-Maximum Suppression (NMS) method employed by Dalal and Triggs on their HOG-based person detection system [53]. The NMS algorithm selects the prediction box with the highest confidence and rules out all the other boxes that are overlapped with an Intersection over Union (IoU) higher than a fixed threshold. It should be noted that the threshold value is selected manually, and the optimal setting usually varies depending on the dataset.

Some attempts have been made recently to learn NMS in a CNN. Hosang *et al.* showed its feasibility, first using results from a greedy NMS as input [127], and later getting rid of the NMS and taking into account double and neighboring detections [128]. They reportedly achieved non-negligible gains of performance in the object detection task.

On the other hand, Soft-NMS [25] aimed to increase the performance of greedy NMS with minimal changes. Detections are rescored according to a Gaussian function of overlap, instead of discarded. This approach does not require training and can be easily implemented, so it was rapidly adopted by the object detection community.

TEST-TIME AUGMENTATION

Generally, the performance of detection methods can be improved on inference time by aggregating the results of several feed-forward passes, each performed on a different image scale or crop. Detections must be later aggregated using some variant of NMS. This technique can be used when the inference running time is not a limiting factor; for instance, it is widely used nowadays in competitions such as those discussed in Sec. 2.2.1.

2.6 PERCEPTION ON AUTOMOTIVE PLATFORMS

Autonomous driving modules are one of the most demanding applications of perception algorithms. The construction of the environment representation is a complex task that requires a high level of reasoning

ability, as discussed in Sec. 1.3.2. For this reason, new methods developed within the computer vision and AI fields are always welcomed and rapidly adopted.

As the reader may have noted, methods introduced in previous sections were mainly designed to handle 2D images from cameras. However, these approaches can be applied to other data sources, such as lidar, provided that the information is conveniently represented.

Apart from coping with the specific challenges of traffic scenes (see the list by Schiele and Wojek in Sec. 1.3.2), algorithms intended for automotive applications should be designed with a particular emphasis on efficiency. Results must be computed under real-time constraints on onboard processing platforms (e. g., embedded computers) with limited computation capabilities.

The following sections are devoted to reviewing modern techniques for traffic scene understanding. The focus is on the analysis of methods geared towards an object-based representation.

2.6.1 Object classification and detection

Before the emergence of DNNs, methods based on hand-crafted features (as those cited in Sec. 2.5.1) were extensively used in object detection for autonomous applications, often focused on the detection of cars [86], [195], [289] and pedestrians [19], [200]. The reader is referred to [246] and [65] for a review of onboard vehicle and pedestrian detection algorithms before the emergence of DL.

Nevertheless, the performance of these approaches was generally subpar due to the difficulties involved in traffic scenarios (e. g., occlusions), which classical algorithms could not deal with adequately. Modern techniques based on DNN brought significant improvements in performance that were promptly leveraged to tackle the problems which stem from the lack of structure of traffic environments.

CNNs have a straightforward application on specific, well-defined tasks such as Traffic Light Recognition (TLR) and Traffic Sign Recognition (TSR). Classical image processing methods, where features were often based on color or shape [54], [185], gave way to CNN-based methods [142] soon after the influential work by Krizhevsky *et al.* [154]. As feature extraction was now embedded into the learning process, these methods featured enhanced robustness against variations such as illumination.

Whereas the application of CNNs to provide a classification of the kind of the traffic sign or the status of the traffic light (i. e., red, yellow or red) is straightforward, ROIs to be classified can be obtained through several approaches: digital maps [143], external proposal generation methods [297] or as an output of the method itself [175], [271]. In any case, the well-defined structure of road signaling generally facilitates that stage.

PRE-DEEP
LEARNING
APPROACHES

ROAD SIGNALING
RECOGNITION

OBJECT
DETECTION

When the aim of the system is the detection of all kinds of objects, including dynamic agents, the complexity of the problem increases as no size or shape assumptions can be easily made. Object detection methods have been profusely applied to this problem, but the particular characteristics of the traffic environments are a source of additional problems that must be often approached through specific solutions.

PROPOSAL
GENERATION

Prominent among these challenges is the adequacy of the proposals sent to the classification stage on two-stage methods, such as Faster R-CNN [213]. In this regard, Yang *et al.* introduced scale-dependent pooling and cascade rejection classifiers to improve the detection performance on different scales [285]. The former aimed to exploit different convolutional features for each proposal depending on its size, whereas the latter was introduced to avoid the duplication of computations by removing the easy negatives early in the process. Similarly, MS-CNN [38] tried to overcome the limitations of the fixed receptive field of the Faster R-CNN's RPN by extracting features at different levels of the backbone network.

Based on those and other similar works, Lin *et al.* proposed the Feature Pyramid Network (FPN) [167] as an efficient solution to detect objects at different scales based on the use of different features maps in a top-down architecture with lateral connections.

METHODS USING
LIDAR

Another group of works employs stereo or lidar information to enhance the proposal generation. Such is the case of 3DOP [44], which generate 3D proposals using object size priors, ground plane, and depth-informed features, taking advantage of the information from a stereo system. DeepStereoOP [205] uses a simplified version of 3DOP and re-ranks the resulting proposals using the disparity map from the stereo system.

Nonetheless, information demanded by planning and control modules usually exceeds the scope of conventional object detection methods. The following sections deal with the pose estimation and 3D localization of objects. It should be noted, however, that these methods often built upon reliable detection methods such as those presented above.

2.6.2 Viewpoint estimation

PROBLEM
STATEMENT

Apart from object detection, estimation of the 3D pose of objects is another matter of interest when dealing with traffic scenes. Although the concept of pose usually includes both the position and orientation of objects in space, the study of the former will be delayed to the next section and, therefore, focus here is on the determination of the orientation.

In the most general case, the description of the orientation of objects in space needs three angles; e. g., pitch (rotation around the transversal axis), roll (rotation around the longitudinal axis), and yaw (rotation

around the vertical axis). However, the road surface can be reasonably approximated by a plane on which dynamic objects (e. g., other cars) move. Under this assumption, pitch and roll angles are negligible and, in any case, irrelevant to describe the dynamics of the objects. The same cannot be said of the yaw angle; knowing the orientation of objects, e. g., pedestrians [78], on the 2D plane defined by the road provides valuable information about their current and future movements and, therefore, has been a subject of study in the literature.

When information comes from a single camera, the estimation of the pose must be done using only appearance features. In that particular case, it is more reasonable to estimate the relative orientation between the camera and the object instead of the absolute yaw. This magnitude is, in fact, the *viewpoint* or *observation angle*. A more detailed discussion about this issue will be provided in Chapter 5.

Both López-Sastre *et al.* [173] and Pepik *et al.* [202], [203] explored the possibilities to enrich a conventional part-based detector with the ability to estimate the viewpoint of the objects. The former also introduced a measure of the accuracy of the orientation prediction, the Mean Precision in Pose Estimation (MPPE).

The problem of the estimation of the pose of objects got renewed interest with the emergence of DNNs. It was soon evident that convolutional features could be useful for this task [94]. However, the availability of training data is one of the main concerns of these methods. Rendered 3D models are used in some cases to obtain different views of the objects [252].

In general, the viewpoint estimation problem can be posed in two ways: as classification into discrete *bins* resulting from the quantization of the full circle, as is the case with the works shown above, or as a continuous regression aimed at providing the exact value of the viewpoint [177], [259]. Yang *et al.* [287] proposed a CNN for discrete viewpoint estimation that was further enhanced by performing a regression based on the scores obtained for each bin and the Kullback-Leibler distance.

Pose-RCNN [29] extends the R-CNN architecture to handle the orientation estimation. Proposals are generated using clustering on lidar data or *stixels* on stereo data. On the other hand, SubCNN [281] proposes a *subcategory-aware* CNN and embed the viewpoint into it as a subcategory to be predicted along with the detections.

Later works are, generally, more ambitious, and aim to estimate full 3D boxes (or cuboids) enclosing the object. An example of this is Deep3DBox [188], represented in Fig. 2.11, which takes 2D detection bounding boxes and enhance them to become oriented 3D bounding boxes. Murthy *et al.* [189] go even further and try to reconstruct the 3D shape of cars using a single frame. When spatial information (i. e., stereo or lidar) is added, the estimation of the location of objects becomes more affordable, as will be discussed in the next section.

VIEWPOINT
ESTIMATIONDEEP LEARNING
APPROACHESCONTINUOUS VS.
DISCRETECUBOID
ESTIMATION

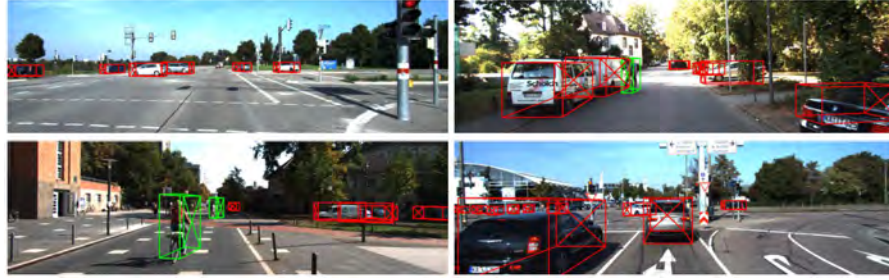


Figure 2.11: Estimation results of the Deep3DBox method [188] © 2017 IEEE

METHOD	AOS CAR	AOS PED.	AOS CYC.
DPM-VOC+VP [203]	63.27	39.83	23.22
Pose-RCNN [29]	75.35	59.89	62.25
SubCNN [281]	88.43	66.28	63.41
Deep3DBox [188]	88.56	—	59.37

Table 2.2: Orientation estimation performance (AOS %) of selected methods on the KITTI object detection benchmark. Results are given for the *Moderate* difficulty level on the testing set⁵

BENCHMARK

A summary of the performance of some of the cited methods can be found in Table 2.2. Results are expressed in terms of the Average Orientation Similarity (AOS) measure, proposed by the KITTI dataset [90] as a way to jointly evaluate the detection and orientation estimation performance⁴.

2.6.3 Obstacle 3D localization

PROBLEM STATEMENT

The object detection problem is usually formulated as the localization of the visible instances within the image frame. However, important though it is, this task does not provide a complete picture of the scene. Navigation through a traffic environment requires precise information not only about the presence of objects, but also about their dimensions and, particularly, their 3D location. As with the orientation, road environments usually allow assuming that objects move along the ground plane and, therefore, effort focus on the estimation of the 2D coordinates within that plane, together with the dimensions of the 3D bounding box.

VISION-BASED METHODS

The last couple of years has seen the proliferation of a large number of methods designed to provide 3D detections. The information can be extracted from different data modalities. Although some methods, such as Mono3D [43] or MonoPSR [156], have shown that geomet-

⁴ AOS will be described in detail in Sec. 5.1.1.

⁵ http://www.cvlibs.net/datasets/kitti/eval_object.php

rical information can be inferred from monocular data, additional sources are usually employed. For instance, spatial information can be retrieved from a set of monocular images [228] through Structure from Motion (SfM) techniques. Spatial reasoning can also be obtained from stereo data, as in the 3DOP approach [44], [45], which employs a CNN that exploits context and depth information to obtain 3D bounding boxes.

Nevertheless, most of the works aimed at 3D detection nowadays make use of lidar data. In this line, VeloFCN [163] and later 3D FCN [162] employed Fully-Convolutional Networks (FCNs) on the data obtained by the lidar to generate 3D detections. The former used a representation later known as *front view* (FV), in which the 3D points are projected onto a cylinder plane, whereas the latter used 3D convolutions.

Generally, the key issue is the representation of the 3D lidar data in such a way that it can be correctly handled by CNNs. In this regard, one of the most influential works was the one by Chen *et al.*, MV3D [46]. They used the front view from VeloFCN and complemented it with a Bird's Eye View (BEV), which is a grid map built using the lidar information⁶. The RGB image was also used as an input, and features were extracted from the three different representations. The approach is depicted in Fig. 2.12.

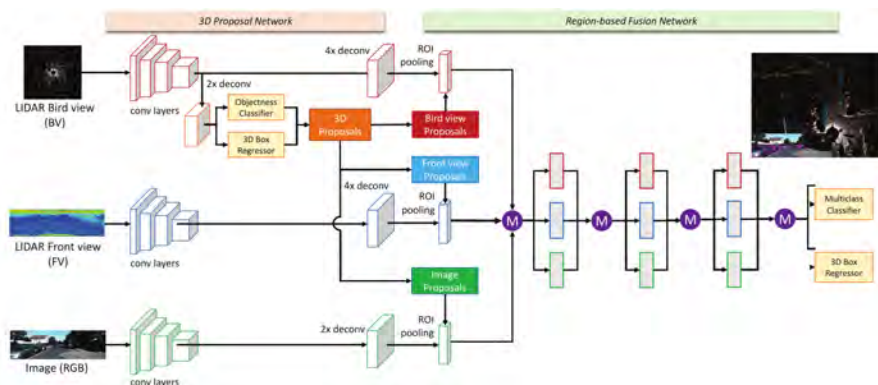
LIDAR-BASED
METHODSLIDAR DATA
REPRESENTATION

Figure 2.12: Multi-View 3D object detection network (MV3D) [46] © 2017 IEEE

Since then, a wide array of works have adopted the BEV representation [166], [274], [292], each of them making use of different features per cell (e. g., max height or intensity). The vast majority of works in this group focus on car detection; this is because VRUs are generally smaller and, therefore, more scarcely represented on the BEV. An exception is AVOD [155], which uses features extracted separately from

BEV-BASED
METHODS

⁶ A detailed description of the BEV representation will be provided in Sec. 3.2.2

METHOD	AP BEV	AP 3D
TopNet [274]	53.71	12.58
RT3D [292]	42.10	21.27
MV3D [46]	76.90	62.35
UberATG-ContFuse [166]	85.83	66.22
F-PointNet [208]	84.00	70.39
AVOD-FPN [155]	83.79	71.88
PC-CNN [64]	86.10	73.80
PointRCNN [242]	86.04	75.42

Table 2.3: BEV and 3D detection performance (AP %) of selected object localization methods on the KITTI object detection benchmark. Results are given for the *Car* category and the *Moderate* difficulty level on the testing set⁸

the RGB image and the lidar BEV to generate 3D proposals (using a modified RPN) and classify them⁷.

3D-CLOUD-BASED
METHODS

Some approaches deal with the raw point cloud, dispensing with the necessity to engineer 2D representations manually [242]. Voxel-Net [296] divides the point cloud into equally-spaced 3D voxels and represents the points within each one using a specific encoding.

PREVIOUSLY
DETECTED
INSTANCES

Another group of methods use the 2D detections to extract the corresponding 3D points and then estimate the 3D box and its location. F-PointNet [208] uses various stages to filter the points located inside the *detection frustum* and provide an estimate of the geometry of the object. On the other hand, Du *et al.* [64] propose a model-fitting algorithm followed by a refinement CNN to provide the 3D detections.

BENCHMARK

Results obtained by most of the cited algorithms on the KITTI dataset are reported in Table 2.3. The measure chosen is the Average Precision (AP), as introduced in [71]. *AP BEV* takes into account 2D detections on the BEV, whereas *AP 3D* is more restrictive since it considers all the dimensions of the 3D bounding boxes.

A comprehensive revision on this matter can be found in the recent review by Arnold *et al.* [4].

2.6.4 Multi-tasking and scene understanding

MULTI-TASK
FRAMEWORKS

The ultimate goal of onboard perception systems is the generation of an environment model, as mentioned in Sec. 1.3.2. The building of this representation requires knowledge about different cues describing the environment. These data are usually obtained by different modules,

⁷ It should be noted, however, that the authors of AVOD use a different model for each category; for instance, three models were trained for evaluation on the KITTI benchmark, each corresponding to one of the three main categories of the dataset.

⁸ http://www.cvlibs.net/datasets/kitti/eval_object.php

such as the ones described so far in this chapter, and combined afterward. However, it is sometimes useful to combine different tasks into the same reasoning process so that they can make use of the same medium-level information.

For instance, Hoiem *et al.* [125] showed that object detection and scene geometry recovery are coupled and should be faced together. Some recent works making use of CNNs have tried to extend their capabilities to perform various tasks simultaneously. For instance, Deep MANTA uses several cascaded CNNs [40] to carry out 3D detection in images; the system is ultimately designed to provide several outputs, including 2D box regression, parts visibility estimation, or template similarity measurement.

In general, multi-task is well suitable for onboard perception tasks, since it performs several inference tasks at the same time, avoiding duplications and, therefore, optimizing the processing requirements. In this line, the work by Oeljeklaus *et al.* aims to provide different cues using a single feed-forward run of a CNN. They have shown the viability of the multi-task approach in two use cases: simultaneous semantic segmentation and road topology [192], and joint road segmentation, object detection, and pose estimation [193]. The performance obtained in both cases is on par with algorithms performing the different tasks separately.

A small group of works aimed at extracting higher-level information from the different cues available. For instance, Geiger *et al.* [88] offered jointly reasoning about the 3D scene layout of intersections, as well as the location and orientation of vehicles in them, using a probabilistic model. Fig. 2.13 shows the cues employed and the expected outcome of the approach. Zhang *et al.* [293] went a step forward and included high-level semantics in the form of traffic patterns to avoid unfeasible combinations.

SCENE
UNDERSTANDING

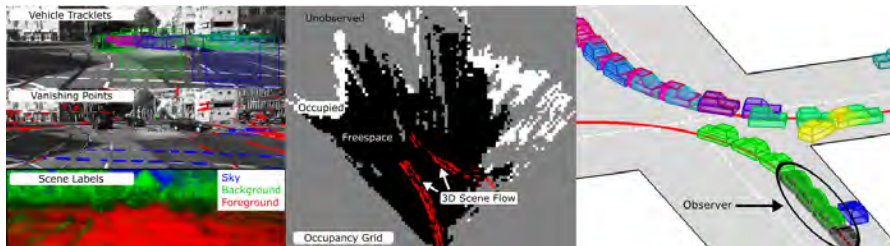


Figure 2.13: 3D intersection understanding problem and cues employed by Geiger *et al.* [88] © 2014 IEEE

2.7 CONCLUSION

A survey covering the different lines of research that will be addressed in this thesis has been introduced. A historical review of autonomous

driving and foundational techniques in computer vision was presented, together with an up-to-date selection of works whose scope overlaps, to a greater or lesser extent, with the approaches proposed in this thesis.

Based on the conducted literature review, object recognition has attracted considerable attention in recent years, pushed by the advances in *DNNs*. Autonomous vehicles, which have made use of computer vision techniques since the initial prototypes, are still one of the largest consumers of these kinds of methods.

The specific challenges which arise in traffic environments have led to the emergence of a large number of methods tailored to onboard perception systems. In this context, onboard detection frameworks usually face additional requirements, such as the estimation of the pose of the objects, which have been widely discussed in the *ITS* literature.

This thesis is intended to tackle some daunting challenges that are still pending. Among them are automatic calibration, efficient and robust object detection, and accurate object localization. The final goal is enabling the creation of complete scene understanding methods that make vehicle navigation safer and more comfortable.

Part II

PROPOSED APPROACHES AND
EXPERIMENTAL RESULTS

This chapter addresses some fundamental issues that must be borne in mind when designing and implementing an onboard perception system, as a prerequisite to obtain meaningful input data for the algorithms. Among the automotive-ready sensor modalities discussed in Sec. 1.3, the radar is set aside as it is out of the scope of this thesis, so the analysis focus instead on vision and lidar devices.

Data from each modality have specific characteristics that determine the processing approaches that must be adopted down the pipeline to extract useful information and feed it to the medium- and high-level navigation modules. In this chapter, an overview of the existing approaches to interpreting raw sensor data and making it suitable for subsequent processing is provided.

First of all, models that explain the relationship between real-world entities and their respective representations on sensor data are introduced. Later, different strategies for data representation, depending on the source, are discussed. Throughout this dissertation, it will be shown that this issue is of paramount importance and, therefore, must be carefully chosen to optimize the performance of the perception algorithms. Finally, the topic of sensor calibration will be addressed. Within that context, a novel approach to process 3D data to perform automatic lidar-camera calibration will be presented and validated on both synthetic and real data.

3.1 FUNDAMENTALS

Sensors provide input data to the different algorithms in the perception stack. However, in order to make valuable reasoning about the environment, the relationship between real points in the environment and their representation on sensor data must be known. Different models can be used depending on the modality in use; a summary is provided below.

3.1.1 *Monocular cameras*

Cameras are endowed with CCD or CMOS sensors that capture the light beams reflected by the objects in their field of view and transform them into electrical signals that can be stored and processed as 2D arrays of numbers. Lenses are the other constituent element of the

VISION DEVICES

This chapter includes content from [114] and [109].

vision system; they are responsible for focusing the light beams to the sensor element.

The pinhole perspective projection model, proposed by Brunelleschi at the beginning of the fifteenth century, is widely used to relate 3D points in the scene with their projection onto the image plane [79].

From a practical point of view, the method establishes that the relationship between a 3D point $\mathbf{P} = (X, Y, Z)^T$ and its projection onto the image plane, $\mathbf{p} = (x, y)^T$, is given by the intrinsic camera matrix, \mathcal{K} ¹:

$$\mathcal{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}, \quad (3.1)$$

where f_x , f_y , c_x , and c_y are the *intrinsic parameters* of the vision system:

f_x, f_y focal lengths expressed in pixel units

c_x, c_y position in pixel units of the retinal coordinate system or principal point

For a monocular camera, the *projection matrix*, \mathcal{P} , can be defined as:

$$\mathcal{P} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = [\mathcal{K} \mid 0] \quad (3.2)$$

Then, the projection $\mathbf{p} = (x, y)^T$ of a point $\mathbf{P} = (X, Y, Z)^T$ onto the rectified image is given by the following set of equations, in homogeneous coordinates:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \mathcal{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (3.3)$$

$$x = \frac{u}{w}$$

$$y = \frac{v}{w}$$

It is important to note that all these equations hold as long as \mathbf{P} is expressed in a local coordinate frame whose origin coincides with the principal point of the camera. In this thesis, the following *camera* frame is mainly adopted²:

x points to the right in the image

y points down in the image

z points into the plane of the image (forward)

¹ Please note that a simplified version of the intrinsic camera matrix is used here, where the skew between axes is assumed to be nil.

² The proposed *camera* frame is the usual in computer vision publications and libraries; e.g., http://wiki.ros.org/image_pipeline/CameraInfo

On the other hand, lenses usually introduce distortion that displaces the projected points from their ideal positions. When using the general *rational polynomial distortion model*, distortion is defined by eight parameters [197]: six of them refer to the radial component ($k_1, k_2, k_3, k_4, k_5, k_6$), and the other two account for the tangential component (p_1, p_2). The simpler *Plumb Bob* model considers only the first three radial coefficients ($k_4 = k_5 = k_6 = 0$). When distortion is considered, the projection of \mathbf{P} involves the following procedure:

1. Projection onto the normalized undistorted image:

$$\begin{aligned} x' &= \frac{X}{Z} \\ y' &= \frac{Y}{Z} \end{aligned} \quad (3.4)$$

2. Distortion coefficients are used to move the point to its distorted position (in a normalized image). This step is also known as rectification. Using the rational polynomial distortion model:

$$\begin{aligned} x'' &= x' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + 2p_1 x' y' + p_2 (r^2 + 2x'^2) \\ y'' &= y' \frac{1 + k_1 r^2 + k_2 r^4 + k_3 r^6}{1 + k_4 r^2 + k_5 r^4 + k_6 r^6} + p_1 (r^2 + 2y'^2) + 2p_2 x' y', \end{aligned} \quad (3.5)$$

where $r^2 = x'^2 + y'^2$.

3. The normalized image is converted to a pixel-coordinate image by applying the intrinsic camera matrix to each image point:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathcal{K} \begin{bmatrix} x'' \\ y'' \\ 1 \end{bmatrix} \quad (3.6)$$

Note that Eq. 3.3 is merely a particular case of this procedure, where all the distortion parameters are zero.

Obviously, the selection of the camera-lens combination must be carefully made based on both compatibility (mainly, regarding the sensor format) and the requirements of the particular application. Two of the most important characteristics to take into account when evaluating the adequacy of the vision system to the application are:

1. The **FOV**, which express the area around the vehicle which is covered by the sensor
2. The area occupied by the subjects of interest in the resulting image, which determines the effective range of the system.

Both properties are determined by the characteristic parameters of the vision system (lens and camera) and can be obtained by making use of the pinhole model. Hence, for non-fisheye cameras, Horizontal Field Of View (**HFOV**) and Vertical Field Of View (**VFOV**) angles can be obtained using the following equations:

$$\begin{aligned} \text{HFOV} &= 2 \cdot \arctan\left(\frac{w_{\text{sensor}}}{2 \cdot f_x}\right) \\ \text{VFOV} &= 2 \cdot \arctan\left(\frac{h_{\text{sensor}}}{2 \cdot f_y}\right), \end{aligned} \quad (3.7)$$

where $w_{\text{sensor}} \times h_{\text{sensor}}$ is the physical size of the CCD or CMOS sensing element, and f is the focal length in mm. Usually, it is safe to assume that $f_x = f_y = f$; then, the value of f is, in practice, given by the lens.

On the other hand, an object of dimensions $w_{\text{obj}} \times h_{\text{obj}}$ placed at a distance d_{obj} from the camera will span $N_w \times N_h$ pixels in the image, where N_w and N_h can be obtained as follows:

$$\begin{aligned} N_w &= \frac{f_x \cdot w_{\text{image}} \cdot w_{\text{obj}}}{w_{\text{sensor}} \cdot d_{\text{obj}}} \\ N_h &= \frac{f_y \cdot h_{\text{image}} \cdot h_{\text{obj}}}{h_{\text{sensor}} \cdot d_{\text{obj}}}, \end{aligned} \quad (3.8)$$

where $w_{\text{image}} \times h_{\text{image}}$ is the image resolution.

It is noteworthy that these equations have been expressed in terms of physical lengths, differently from Eqs. 3.1-3.6, which were written in terms of pixel units. The relationship is given by the size of each pixel within the sensor. Usually, pixels are square, so if the pixel size is $w_{\text{pixel}} = h_{\text{pixel}} = l_{\text{pixel}}$; then $w_{\text{sensor}} = w_{\text{image}} \cdot l_{\text{pixel}}$ and $h_{\text{sensor}} = h_{\text{image}} \cdot l_{\text{pixel}}$.

Of course, the resulting image is dependent on other characteristics that must be considered as well, such as the aperture of the lens and the dynamic range of the sensor.

3.1.2 Stereo cameras

While all the equations described in Sec. 3.1.1 hold for each of the cameras of a stereoscopic system, devices of this kind present some particularities that must be considered.

First of all, stereo rectification involves not only the transformation into an image with the distortion corrected but also a transformation to align the two images of the stereo pair so that they lie in the same plane and have coincident epipolar lines[225].

When this transformation is performed, the definition of projection matrix given by Eq. 3.2 no longer applies, as the leftmost 3×3 submatrix in \mathcal{P} become different from \mathcal{K} . Additionally, the projection matrix has an additional parameter, the translational offset $\mathbf{t} = (T_x, T_y, 0)^T$,

which reflects the external position of the right camera relative to the left camera. The projection matrix \mathcal{P} then becomes:

$$\mathcal{P} = \begin{bmatrix} f'_x & 0 & c'_x & T_x \\ 0 & f'_y & c'_y & T_y \\ 0 & 0 & 1 & 0 \end{bmatrix} = \mathcal{K}' \left[\mathcal{I} \mid \mathbf{t} \right], \quad (3.9)$$

with \mathcal{I} being the identity matrix, and \mathcal{K}' , the intrinsic matrix of the rectified image. It makes sense to use a local camera frame attached to one of the two cameras, so that, for that camera, $T_x = T_y = 0$. Furthermore, in the most usual (canonical) setup, the two cameras are looking perpendicular to the line joining both camera centers; in that case, $T_y = 0$ and $T_x = -f_x \cdot B$, where B is the distance separating both cameras, known as the *baseline*. The baseline is the most critical parameter to take into account in the selection of a stereo vision system; in general, wider baselines are suitable for more distant operation ranges, as will be shown in Sec. 3.2.1.

3.1.3 Lidar

The operation principle of most modern multi-layer, 360° lidars is based on several light emitters, vertically arranged, that perform rotation (all together) around a slightly displaced vertical axis.

Lidar raw data is a set of 3D points that can be represented in a *point cloud*. A point cloud is an arrangement of points defined by their x , y , and z coordinates. Each point can also include additional information, such as color, which can be represented using RGB values. As with the camera, points are in a local reference frame whose origin is located in the center of the lidar sensor. A typical lidar reference system is the one described below³:

- x points forward
- y points to the left
- z points upward

This specification assumes that the lidar is positioned in such a way that scan planes are roughly parallel to the ground plane.

When selecting a lidar device, prominent features are the resolution and the FOV. The number of scan planes determines the vertical resolution; today, devices with 16, 32, 64, and 128 planes can be found in the market⁴. Different strategies can be employed for the distribution of these planes, which has given rise to a variety of devices with different FOVs and plane densities.

LIDAR DEVICES

POINT CLOUD

LIDAR SELECTION
CRITERIA

³ The lidar reference frame introduced here follows the standard ROS coordinate conventions from REP-0103 (<https://www.ros.org/reps/rep-0103.html>) for axis orientation. Note that it differs from the default reference frame used by several manufacturers such as Velodyne.

⁴ For instance, see: <https://www.velodynelidar.com/products.html>

3.2 DATA REPRESENTATION

Through the following chapters of this document, processing based on CNNs will be applied to the sensor data to obtain meaningful information. As mentioned in Sec. 2.4, CNNs are particularly suitable for 2D data arrangements. Whereas images are naturally represented that way, the issue is not so straightforward when dealing with 3D geometrical information from stereo and lidar systems. Stereo vision involves a non-trivial procedure to obtain an estimate of the scene geometry that leads to an image-like structure, while lidar data must be projected into manually-engineered 2D representations. Both issues are discussed below.

3.2.1 Stereo matching

Retrieving the 3D information from a pair of stereo images requires a procedure to find matching pixels in both images and convert their image coordinates into 3D points. That procedure is known as *stereo matching* and is one of the most widely studied and fundamental problems in computer vision [256].

EPIPOLAR
GEOMETRY

If positions and calibration data for the cameras are known, the search space of possible correspondences on the other image for each pixel is reduced to a line, according to the principles of epipolar geometry (Fig. 3.1). If images are *rectified* beforehand, then corresponding horizontal scanlines are epipolar lines, and the search gets even simpler.

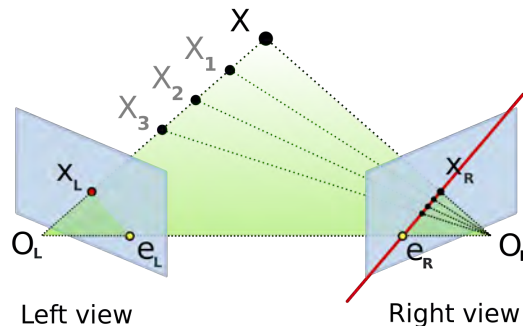


Figure 3.1: Epipolar geometry. Figure by Arne Nordmann in Wikimedia Commons⁵ (license CC BY-SA)

The resulting *standard rectified geometry* is employed in a lot of stereo camera setups and stereo algorithms and leads to an elementary

⁵ https://commons.wikimedia.org/wiki/File:Epipolar_geometry.svg

inverse relationship between depths in the local reference frame (Z) and *disparities* (d) [256]:

$$d = f \frac{B}{Z}, \quad (3.10)$$

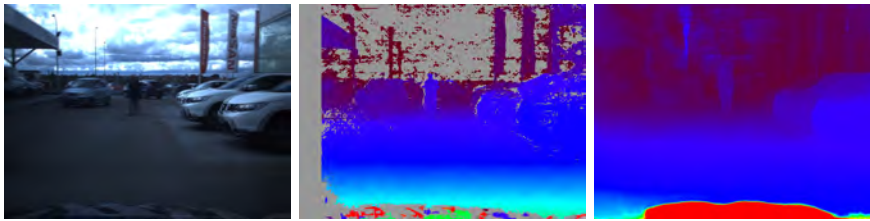
where f is the focal length (in pixels), B , the baseline (distance between cameras) and the disparity d , a value relating the coordinates of the projection of a point onto one of the images, $(x_1, y_1)^T$, with the coordinates of the projection of the same point onto the other image, $(x_2, y_2)^T$:

$$x_2 = x_1 + d(x_1, y_1) \quad (3.11)$$

$$y_2 = y_1 \quad (3.12)$$

Usually, the disparity is computed from left to right and, therefore, subscript 1 refers to the left image and subscript 2, to the right one. *Dense* matching algorithms are supposed to assign a disparity value to each pixel in the image⁶. Then, the resulting information can be represented in a 2D *disparity map* containing the $d(x, y)$ values, as shown in Fig. 3.2.

DISPARITY MAP
COMPUTATION



(a) RGB images overlapped (b) SGM disparity map (c) DispNet disparity map

Figure 3.2: Stereo matching and disparity maps obtained with two different methods [114]

The accuracy and completeness of the depth estimation depend strongly on the stereo matching algorithm. For instance, the disparity map in Fig. 3.2b presents large areas with undefined disparity values (in gray), while the one in Fig. 3.2c has a density of 100%.

Although a wide variety of algorithms have been proposed over the years, three paradigmatic groups of approaches can be cited:

LOCAL ALGORITHMS: Local algorithms, such as the popular Block Matching (BM) [151], search for correspondences taking into account a finite window around each pixel. Usually, the comparison is established in terms of similarity of the intensity values within that window; then, the estimated disparity value is the one that maximizes the resemblance (that is, minimizes a dissimilarity cost) between both images. While stunningly fast

⁶ In practice, some algorithms can be unable to find proper values for a certain percentage of the pixels.

compared to other approaches, local methods struggle with challenging situations such as the lack of texture, which often results in not-so-dense disparity maps with large undefined areas.

GLOBAL ALGORITHMS: Global algorithms make explicit smoothness assumptions and then solve a global optimization problem that provides a solution for the disparity assignment. Most methods are formulated in an energy-minimization framework, where the energy is made of two terms: one considers the agreement of the disparity function with the input image pair, whereas the other encodes the smoothness assumptions. As these methods incorporate information from the whole image, their computational cost is often prohibitive, unless specific hardware or efficiency-oriented implementations are used. A good compromise is offered by the Semi-Global Matching (SGM) algorithm proposed by [124], which only takes into account eight directions (the four cardinal and the four intercardinal directions) from each pixel to solve the energy-minimization problem through scanline optimization (a variant of dynamic programming). The widely-used OpenCV implementation matches pixel blocks instead of individual pixels and can be, therefore, more accurately described as Semi-Global Block Matching (SGBM)⁷. The disparity map in Fig. 3.2b was obtained through this approach.

NEURAL-NETWORK-BASED: Stereo matching is not alien to the general trend towards feature learning brought by DL, and top-performing methods today make use of DNNs in some way or another. A lot of labeled samples are required to optimize the parameters of the network, which is not a trivial requirement. Datasets usually employed to that end include KITTI, which obtains the depth ground-truth from lidar measurements, and the collection of synthetic datasets from the University of Freiburg (e.g., FlyingThings3D or Monkaa)⁸. These very same researchers proposed an encoder-decoder architecture, DispNet [181], designed to perform real-time end-to-end disparity computation based on their previous work, FlowNet [62]. A sample of the resulting disparity map using this method can be found in Fig. 3.2c.

Once the dense disparity map containing the values of $d(x, y)$ is available, the value of the depth coordinate for each pixel within the image is easily obtainable through Eq. 3.10. Hence, stereo data can be represented by a point cloud, similarly to the lidar data. Additionally, color information (RGB) is naturally available for every pixel since, after all, original data were images from two camera devices. A

⁷ https://docs.opencv.org/3.4/d1/d9f/classcv_1_1Stereo_1_1StereoBinarySGBM.html

⁸ <https://lmb.informatik.uni-freiburg.de/resources/datasets>

METHOD	D1-ALL (%)	RUNTIME (s)
SegStereo [286]	2.25	0.6
DispNetC [181]	4.34	0.06
SGM_ROB [124]	6.38	0.11
OCV-BM [151]	25.27	0.1 ^a

^a Method implemented on CPU instead of GPU

Table 3.1: Error rates and run times of selected stereo matching methods on the KITTI dataset. The error rate is expressed as the percentage of stereo disparity outliers over all the ground truth pixels

representation of the XYZRGB stereo point clouds resulting from the disparity maps in Fig. 3.2 is depicted in Fig. 3.3.

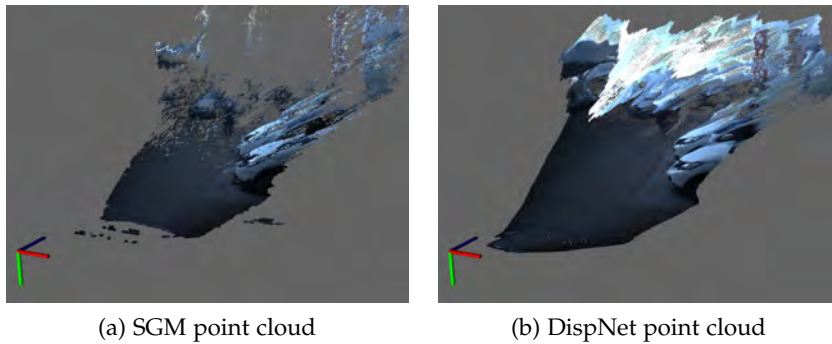


Figure 3.3: 3D point clouds obtained with two different stereo matching algorithms [114]

The analysis of the results in Figs. 3.2 and 3.3 shows that DispNet achieves a higher density than SGM, but is not exempt from spurious estimations that are particularly visible in Fig. 3.3b in the sky area. It is important to note that the accuracy of DNN-based models is strongly dependent on the training data; in this case, the model was trained with the KITTI stereo benchmark [90], whose labels are unevenly distributed due to the properties of the lidar used to generate the ground-truth. Although many other methods have been proposed in recent years, some of them achieving better accuracy than DispNet, very few are suitable for real-time processing due to their high computational cost. Table 3.1 establishes a quantitative comparison of the accuracy of the algorithms cited in this section, using the number of outliers (i.e., clearly wrong estimations) as a measure of performance. A more comprehensive comparison can be found in the KITTI public stereo benchmarks⁹.

When using stereo vision to retrieve geometrical information, estimation accuracy at far distances is known to be worse than in the near

BENCHMARK

MATCHING
ERROR

⁹ http://www.cvlibs.net/datasets/kitti/eval_stereo.php

range. The error made in the estimate, δz , increases with the distance from the camera, z , according to the following equation [73]:

$$\delta z = \frac{z^2}{fB} \delta d, \quad (3.13)$$

where δd is the error made in the stereo matching process itself. Some reasonable values for this error are, for instance, $\delta d \approx 2$ for the SGM method and $\delta d \approx 1$ for the DispNet method¹⁰.

All these considerations will be taken into account in Sec. 6.1, where a method for 3D object localization based on stereo data will be proposed.

3.2.2 Lidar

MOTIVATION

Although lidar measures can be straightforwardly expressed in a point cloud structure, like the one depicted in Fig. 3.4c, this representation may be inefficient depending on the particular application, because of two interrelated reasons:

1. In general, and due to their limited resolution, data from devices of this kind is very sparse, which results in large areas of space being empty. These empty zones waste resources without providing any useful information.
2. 3D data is naturally bulkier than 2D data due to the extra coordinate, which further increases the already high processing requirements.

DATA REPRESENTATIONS

In order to mitigate these problems and, additionally, obtain a 2D representation naturally suitable for CNNs, different approaches have been proposed in the literature. Two of them are described here because of their widespread use in modern object detection approaches:

FRONT VIEW (FV): Lidar data can be projected to obtain a more compact representation; however, it is noteworthy that, differently from the stereo disparity map, this information is not dense when projected onto an image plane. The most straightforward way to create a compact 2D representation from lidar points is, instead, adjusting them to an image where each row contains all the range measurements from the same scan plane. The equations to obtain the (r, c) coordinates of a 3D point (x, y, z) in the so-called *front view* are [46]:

$$\begin{aligned} c &= \lfloor \text{atan2}(y, x) / \Delta\theta \rfloor \\ r &= \lfloor \text{atan2}(z, \sqrt{x^2 + y^2}) / \Delta\phi \rfloor \end{aligned} \quad (3.14)$$

¹⁰ These estimates were found experimentally.

This kind of representation results in image-like 2D structures, like the one shown in Fig. 3.4b. This sample features a 64-layer lidar and, hence, that is the number of rows with information in the FV. Since the device is mounted on the roof of a vehicle, most laser beams point downward¹¹ and collide with the ground, which is, therefore, more densely represented than the rest of the scene. Please note as well that the resulting image from 360° data is considerably wider than the crop depicted in Fig. 3.4b, as every column represents an individual range measure, and horizontal resolution is significantly higher than the vertical.

BIRD'S EYE VIEW (BEV): Occupancy grid representations are a classical approach for robot navigation [187]. Grid maps partition the environment into cells, which are then provided with information about the space represented by each of them. This information can be a binary state representing the occupancy (as in Fig. 3.4d), although when the grid map is created using lidar information, additional features can be included, such as the number of points per cell or the mean value of their reflection intensities. The name Bird's Eye View (BEV) has gained traction in the literature to represent this kind of representation. Despite that this representation does not solve the sparsity problem (as apparent in Fig. 3.4d, where most pixels are still empty), it reduces the dimensionality of the data by pruning out information with little relevance to the goal of traffic scene understanding (the height coordinate), while preserving the x and y coordinates of points.

The data thus processed behaves, to a certain extent, as an image; therefore, it can serve as an input of a CNN, provided that adequate considerations are made. A method for 3D object detection based on the BEV representation will be proposed in Sec. 6.2.

3.3 SENSOR CALIBRATION

Models described in the previous section make use of parameters that are specific to each particular device. Recovering the geometrical information from sensor data, especially cameras (i.e., images), involves determining the *intrinsic parameters* which define the relationship between the world points and its projection. On the other hand, several sensors can be part of the same sensor setup, and the parameters defining the relationship between them is referred to here as *extrinsic parameters*.

TAXONOMY

¹¹ The device is, actually, a Velodyne HDL-64E unit (<https://velodynelidar.com/hdl-64e.html>), with a VFOV from +2.0° to -24.9°

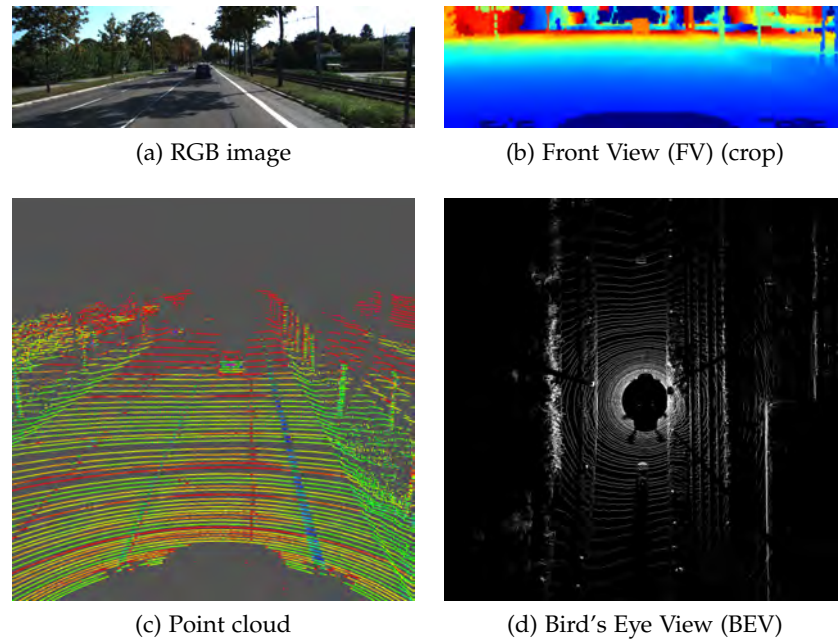


Figure 3.4: Three different representations of the lidar information on a scene from the KITTI dataset

3.3.1 *Intrinsic calibration*

PROBLEM
STATEMENT

Intrinsic parameters are different for each device due to, among other factors, minor differences in manufacturing. Generally, small changes in calibration parameters involve significant changes in accuracy; that is why a calibration process must be carried out before putting the device into operation, in order to have a precise estimation of the intrinsic parameters.

Sometimes, manufacturers perform this procedure before the sensor is sold; this is the case, for example, with lidar devices. However, both elements that make up a vision system, i.e., the lens and the camera, affect the intrinsic parameters. As both parts are bought separately, the intrinsic calibration of vision systems usually falls to the end-user.

ZHANG'S
METHOD

As stated in Sec. 2.3.1, the method by Zhang [295], which uses a fiducial pattern, is widely used nowadays to perform intrinsic calibration. The OpenCV implementation, which was later wrapped in a Robot Operating System (ROS) package¹², is the method-of-choice to perform intrinsic calibration. Values for f_x , f_y , c_x , c_y , as well as the k_i and p_i distortion parameters, are outputs of the procedure.

Fig. 3.5 shows a sample frame that can be used as an input for the Zhang's method, as the checkerboard pattern is completely visible. Generally, the accuracy of the obtained estimation increases with the number and diversity of the samples provided.

¹² http://wiki.ros.org/camera_calibration



Figure 3.5: Checkerboard pattern used for intrinsic calibration as seen from a camera with a high positive radial distortion (barrel distortion)

3.3.2 Extrinsic calibration

Extrinsic calibration between sensors is a prerequisite to establishing correspondences between data representations coming from different devices and, therefore, to make use of all the available sources of information jointly.

MOTIVATION

According to the most general definition, extrinsic parameters relate the sensor's coordinate system to another one (e.g., a fixed world coordinate system or another sensor's local system), thereby specifying its position and orientation in space. When dealing with automotive platforms, the hardest problem is, usually, the determination of the relative pose of sensors with respect to each other.

In that case, the extrinsic calibration problem can be formulated as the estimation, in pairs, of rigid-body transforms, each relating the coordinate system $\{C_1\}$, fixed in one of the sensors, with the coordinate frame $\{C_2\}$, fixed in the other sensor. This transformation is defined by six parameters, $\xi_{12} = (t_x, t_y, t_z, \phi, \theta, \psi)$, which express the translation along the x -, y -, and z -axes, and the rotation around x (roll), y (pitch) and z (yaw).

PROBLEM
FORMULATION

Using homogeneous coordinates, the set of parameters ξ_{12} can also be expressed as a transformation matrix, \mathbf{T}_{12} , which allows transforming a point in the $\{C_1\}$ frame, $\mathbf{p}_1 = (x_1, y_1, z_1, 1)^T$, into a point in $\{C_2\}$ coordinates, $\mathbf{p}_2 = (x_2, y_2, z_2, 1)^T$:

$$\mathbf{p}_2 = \mathbf{T}_{12}\mathbf{p}_1 = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \\ 1 \end{bmatrix}, \quad (3.15)$$

where $r_{11}, r_{12}, \dots, r_{33}$ can be obtained from ϕ, θ , and ψ through algebraic manipulation.

In the most typical setup, both sensors to be calibrated have overlapped FOVs. Features in the area visible from both sensors can then be exploited to perform calibration by finding correspondences between both data representations. These features can be more easily obtained and associated if a fiducial pattern, with distinctive motifs, is employed. While the procedure is usually assumed to be performed offline and in a controlled environment, accuracy requirements make it advisable to minimize the need for human intervention.

3.4 AUTOMATIC STEREO-LIDAR EXTRINSIC CALIBRATION

Among the different sensor pairs that can be found in autonomous vehicles, the camera-lidar combination is one of the most interesting because of the complementarity of both modalities. This fact, which stems from the difference between their respective data, is what makes it one of the most challenging sensor pairs to calibrate as well.

Stereo vision systems provide the same appearance information as a single camera but also allow reconstructing the 3D structure of the environment. In this section, a procedure for extracting features useful for extrinsic calibration from stereo data is proposed. The procedure belongs to a general stereo-lidar calibration framework that was presented in [109] and will also be outlined here for completeness.

3.4.1 Introduction

HIGHLIGHTS

The presented stereo-lidar calibration procedure aims to provide an extrinsic calibration solution that does not require complex setups (e.g., using multiple calibration patterns) and is valid for different commercially-available lidar devices (e.g., with 16, 32 or 64 layers). A short series of synchronized stereo-pair images and lidar scans are used. Intrinsic parameters are assumed to be known (including the stereo camera baseline), and rectification is performed beforehand.

Inspired by the work by Velas *et al.* [266], a custom-made planar calibration target, shown in Fig. 3.6, is used. The marker has a very distinctive, mistake-proofing shape, with a large planar surface and four circular holes that are difficult to confuse in the representations provided by the sensors. The physical realization used in the experiments is made of wood and was manufactured using a CNC wood machine for maximum accuracy in dimensions¹³.

METHOD OVERVIEW

The centers of the four circular holes are used as distinctive features visible by both sensors; therefore, the calibration target must be placed in the common FOV. The placement of the pattern should be selected so that each hole is intersected by two lidar planes and can be perceived from the stereo system.

¹³ The high-quality pattern used in the experiments was kindly provided by Persiman S.L.U. Special thanks to Juan Guindel for his help.

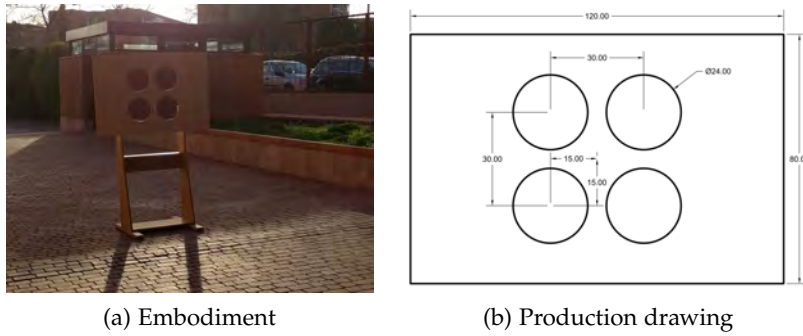


Figure 3.6: Calibration target used by the proposed calibration method

The algorithm can be divided into two main phases: segmentation of reference points (the centers of the circles) in both clouds and registration between them to estimate the transform parameters. Following the notation introduced in Sec. 3.3.2, $\{C\}$ is the reference frame fixed in the camera and $\{L\}$, in the lidar.

3.4.2 Feature extraction from stereo data

The segmentation of reference points from sensor data involves several processing steps. As stated earlier, a particular procedure to handle the stereo data is proposed here. Lidar data requires an equivalent process that takes into account the particularities of the range data; further discussion will be provided later.

The extraction of the reference points from the stereo cloud is done in five stages, which are depicted in Fig. 3.7 and described in detail in the following sections.

3.4.2.1 Pre-processing

In order to retrieve the 3D point cloud from the pair of stereo images, an SGM approach is used. The OpenCV SGBM implementation was found reasonably accurate for depth estimation, which is, indeed, the primary requisite for this application. The accuracy of border localization, a typical problem in stereo matching, has a limited impact on the algorithm since, as will be shown later, it is tackled by using the intensity information. However, the calibration target is assumed to present a minimum of texture (e.g., wood grain) so that the stereo correspondence problem can be solved.

Before the segmentation steps of the algorithm, the stereo cloud is fed through several pass-through filters that remove points out of the normal range of operation, thus limiting the information processing to the area in which the calibration target must be placed.

STEREO
MATCHING

FILTERING

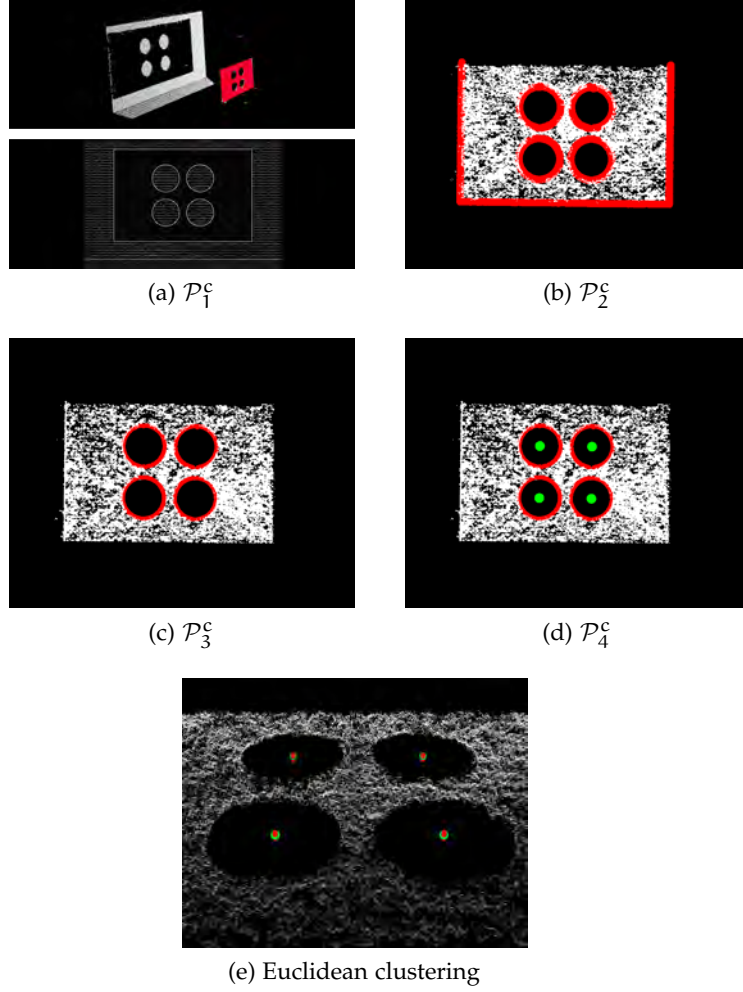


Figure 3.7: Proposed approach to extract the reference points from the stereo point cloud [109] © 2017 IEEE

3.4.2.2 Target segmentation

DEFINITION

Segmentation is used to find subsets of points \mathcal{P}_{i_0} representing a geometrical shape (e.g., a plane) in the input cloud. If several segmentations are applied successively, for each step i_0 , it holds:

$$\mathcal{P}_{i_0}^c = \{(x, y, z)\} \subseteq \mathcal{P}_{i_0-1}^c, \quad (3.16)$$

where the superscript c is used to make clear that clouds are coming from the stereo camera. The extraction of the reference points from the calibration marker involves several segmentation steps, as well as some model fitting procedures. The latter make use of sample consensus-based segmentation methods, i.e., the Random Sample Consensus (RANSAC) family [77], because of its robustness against outliers. On the other hand, the preceding segmentation stages aim to extract points belonging to discontinuities in the calibration target, which will be used to extract the location of the fiducial points.

First of all, taking advantage of the planar shape of the calibration target, points within the stereo cloud that support a plane model are found; then, those points are segmented. Since several planes can be represented in the cloud, such as those belonging to the ground plane or building walls, some restrictions are imposed on the resulting plane. On the one hand, a tight threshold δ_{plane} is used to determine when a point fits the model; and, on the other hand, the plane is required to be parallel to the vertical axis of the sensor reference frame, with an angular tolerance α_{plane} ¹⁴.

PLANAR SURFACE
SEGMENTATION

When the plane model is available, points whose distance to the model is higher than $\delta_{\text{inliers},c}$ are removed, resulting in the cloud segment \mathcal{P}_1^c . An example of the result of the plane segmentation process is shown in the upper part of Fig. 3.7a (in red).

After that, the segmented cloud undergoes a process aimed to filter out the points in the calibration target not belonging to discontinuities. This is effectively carried out by keeping the points in \mathcal{P}_1^c that map to edges in the intensity image. To that end, a Sobel filter is applied over the left image of the stereo pair (lower part of Fig. 3.7a). Points whose projection on the image corresponds to a low value in the Sobel image (smaller than $\tau_{\text{sobel},c}$) are filtered out, as shown in Fig. 3.7b, generating \mathcal{P}_2^c as output.

3.4.2.3 Circle segmentation

The second group of segmentation steps is intended to segment the four circles of the calibration target, whose centers will be used as correspondence keypoints at the registration stage.

In order to avoid introducing noise into the center estimation, points not belonging to the circles are removed. As only discontinuities are present in \mathcal{P}_2^c , outliers to be removed belong, mostly, to the outer edges of the calibration target. For this reason, the cloud is subjected to a filtering process aimed at the elimination of straight lines. Lines are found using a sample consensus segmentation. To avoid removing points belonging to the circles, only those lines compatible with the layout of the calibration pattern are considered. Notwithstanding these considerations, experiments proved that the segmentation method is largely insensitive to the presence of these borders, except for some particular poses of the calibration target. The filtered cloud, \mathcal{P}_3^c , is shown in Fig. 3.7c (in red).

FILTERING

Afterward, remaining points are used to fit four circle models. To that end, a circle segmentation process is performed in the 2D space determined by the plane model π^c . This is effectively implemented by rotating the cloud \mathcal{P}_3^c until the points are aligned with the XY plane and then translating it along the z-axis to reach the value enforced by the plane equation. Later, circles are segmented in the XY subspace

CIRCLE
SEGMENTATION

¹⁴ As shown in Fig. 3.6, the calibration target is intended to stand up on the ground, and the camera is assumed to have a limited pitch deviation from the horizontal.

PARAMETER	VALUE	UNIT
$\delta_{\text{inliers},l}$	10	cm
δ_{cluster}	2	cm
$\tau_{\text{sobel},c}$	128	
α_{plane}	0.55	rad
N	30	

Table 3.2: Parameters for reference points extraction from stereo data

through sample consensus, imposing the known circle radius as a constraint. Center-to-center distances are also checked against the actual dimensions of the calibration pattern to avoid spurious detections. Finally, the obtained centers are transformed back to the 3D sensor's coordinate frame, resulting in the point cloud \mathcal{P}_4^c , depicted in green in Fig. 3.7d. Note that, since the circle segmentation stage is performed in a bidimensional space, the proposed method is also suitable for scarce clouds, such as those provided by low-end lidars.

EUCLIDEAN
CLUSTERING

As a result of the procedure described above, the 3D coordinates of the centers of the four circles, relative to the reference frame fixed in the camera $\{C\}$, are obtained. To make the estimation more robust against the different sources of noise present in the process (e.g., the accuracy of the stereo matching algorithm), centers are effectively cumulated over a window of N frames. The resulting cloud goes through a clustering algorithm that merges points belonging to each circle. The estimated positions of the four centers that are used in the next step of the calibration procedure are, ultimately, the centroids of each cluster. This strategy assumes that the calibration target, as well as the sensors, remains static throughout the selected N-length window. A Euclidean clustering, with a cluster tolerance of δ_{cluster} , is used. This approach can cope reasonably well with outliers; however, additional restrictions can be imposed on both the minimum and the maximum number of points allowed in each cluster, taking into account the length of the window.

Values for the different parameters involved in the segmentation process were selected empirically and are presented in Table 3.2.

LIDAR DATA
PROCESSING

The procedure employed to extract the center of the circles within the lidar data is analogous to the one presented here, with two significant exceptions [109]: discontinuities are segmented using differences in depth between neighboring points, and the outer boundaries of the calibration target are removed instead by restricting the number of points in each ring.

3.4.3 Registration

Registration is responsible for establishing correspondences between the reference points (centers of the circles) from both data representations and obtaining the set of transformation parameters $\hat{\xi}_{CL} = (t_x, t_y, t_z, \phi, \theta, \psi)$ that minimize the distance between both sets.

As presented in [109], a two-stage registration procedure can be employed. First, the optimal transformation assuming the absence of rotation (pure translation) is computed; in homogeneous coordinates:

COARSE
ESTIMATION

$$\mathbf{T}_{CL(1)} = \begin{bmatrix} 1 & 0 & 0 & t_{x(1)} \\ 0 & 1 & 0 & t_{y(1)} \\ 0 & 0 & 1 & t_{z(1)} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.17)$$

This transformation can be obtained by finding the least-squares solution of the overdetermined system of 12 equations provided by the registration of the three coordinates of each of the four reference points. The association is made by assuming that the relative ordering of points in space is the same for both sensors.

After that, the estimation is refined using the well-known Iterative Closest Points (ICP) algorithm [21], which provides a full transformation (a combination of translation and rotation):

FINE
ADJUSTMENT

$$\mathbf{T}_{CL(2)} = \begin{bmatrix} r_{11}(2) & r_{12}(2) & r_{13}(2) & t_x(2) \\ r_{21}(2) & r_{22}(2) & r_{23}(2) & t_y(2) \\ r_{31}(2) & r_{32}(2) & r_{33}(2) & t_z(2) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.18)$$

The final transformation is obtained as the composition of the two partial transformations:

$$\mathbf{T}_{CL}(\hat{\xi}_{CL}) = \mathbf{T}_{CL(2)}\mathbf{T}_{CL(1)} \quad (3.19)$$

3.4.4 Experimental results

The validity of the method for the extraction of the reference points from the stereo data must be assessed regarding its usefulness within the whole calibration pipeline to which it belongs.

The ground-truth transformation relating two sensors is impossible to obtain in practice since it would require knowing the exact position of the sensing elements within both devices. Validation of the calibration algorithm was therefore carried out through two methods:

1. Experiments on a virtual environment created with the open-source simulator Gazebo [149], replicating the exact characteristics of both sensors, including the measuring noise, and the

calibration pattern. In this case, ground-truth parameters are available, and the error can be quantified.

2. Real-world experiments where the validity of the estimated calibration for its use in perception applications could be qualitatively assessed.

3.4.4.1 Evaluation using the synthetic test suite

Tests using the simulation environment were performed on nine different calibration setups with different relative poses between both sensors, and also with respect to the calibration pattern. Some very challenging setups, beyond the typical configurations of automotive applications, were included to generalize the conclusions of the experiments.

EVALUATION
METRICS

Following [91], the performance was evaluated as the difference between both the estimated and the ground-truth transforms, which is separately measured in its linear and angular components:

$$e_t = \|\mathbf{t} - \mathbf{t}_g\| \quad (3.20)$$

$$e_r = \angle(\mathbf{R}^{-1}\mathbf{R}_g), \quad (3.21)$$

where \mathbf{t} is the last column of \mathbf{T}_{CL} in Eq. 3.19, which represents the translation, and \mathbf{R} , the upper-left 3×3 portion of \mathbf{T}_{CL} , which corresponds to the rotation components.

An extensive set of experiments can be found in [109], including some tests for the selection of the N parameter and an analysis of the sensitivity of the method to noise in sensor data. Here, a measure of the accuracy of the obtained transformation on the nine test scenarios of the Gazebo synthetic test suite is presented.

QUANTITATIVE
RESULTS IN
SIMULATION

Results are shown along with those obtained by two comparable methods: the one proposed by Velas *et al.* [266], analyzed using their ROS implementation, and the one by Geiger *et al.* [91], whose results were obtained through their public web toolbox¹⁵. For the sake of fairness, it is important to remark that both approaches are aimed at monocular cameras; besides, the latter one can provide the camera intrinsic parameters as well. Both methods were introduced earlier in Sec. 2.3.2.

Experiments for the method by Geiger *et al.* were conducted on a representative set of settings among the nine possible configurations, in order to avoid overusing the public toolbox. Meanwhile, the method by Velas *et al.* was applied over the whole set of settings, as the source code is available¹⁶. Nevertheless, it was unable to provide valid results for most of the settings since it was designed for smaller magnitudes of the transformation parameters.

¹⁵ <http://www.cvlibs.net/software/calibration/>

¹⁶ http://wiki.ros.org/but_calibration_camera_velodyne

As the calibration algorithm is mainly targeted at low-end lidar devices (in contrast to other methods tailored for 64-layer lidars), performance evaluation was conducted over three devices with a different number of scan planes: 16, 32 and 64. Results, considering only meaningful outcomes, are presented in Fig. 3.8 using box-and-whisker plots.

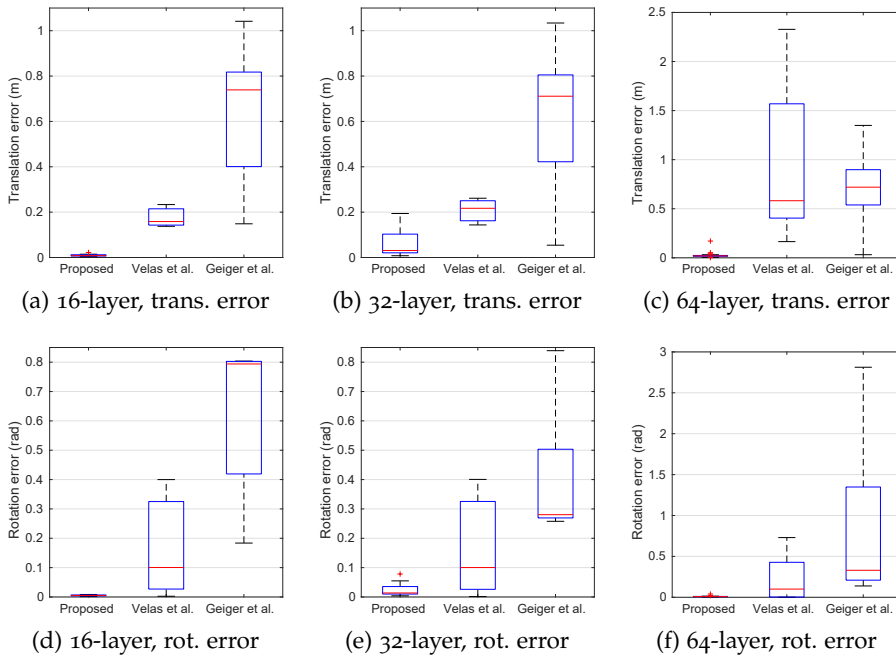


Figure 3.8: Accuracy of the proposed calibration approach in comparison with other existing methods [109] © 2017 IEEE

3.4.4.2 Evaluation in real scenarios

The algorithm was also tested on a real platform with a 16-layer lidar and a 12-cm baseline stereo vision system. After performing calibration with the wooden pattern shown in Fig. 3.6, correspondence between the stereo and lidar clouds could be established, as shown in Fig. 3.9a. Subsequently, correspondence between lidar points and 2D points in the color image from the left camera becomes trivial, as proved in Figs. 3.9b and 3.9c.

The algorithm was implemented in a ROS package and released as open-source software¹⁷ to ease its reproducibility by the scientific community and to provide an off-the-shelf tool to the general public.

3.4.5 Additional remarks

It is important to note that the approach used for registration is mostly agnostic to the procedure used upstream to extract the reference points

QUALITATIVE
RESULTS

¹⁷ https://github.com/beltransen/velo2cam_calibration/

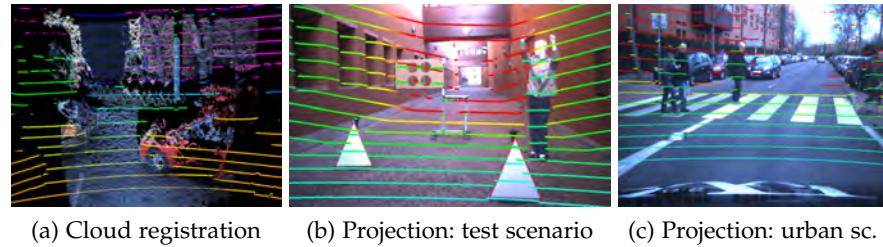


Figure 3.9: Examples of the calibration result in real scenes [109] © 2017 IEEE (Figs. a and b)

from the fiducial marker. Therefore, it would be possible to extend the procedure to deal with almost every kind of sensor, as long as distinctive features from the calibration target can be extracted from the data.

One of the most natural extensions would be the case of monocular cameras, which will be a more general case encapsulating the particular case of stereo vision systems. Although geometrical information (i.e., the depth coordinate) is not straightforwardly available from a single image, using a marker with known-size motifs would enable recovering the missing scale factor and, therefore, using the calibration procedure described above. Such a marker could be one of those described in Sec. 2.3.1 for intrinsic calibration: checkerboards with a fixed square size or squared planar markers.

3.5 CONCLUSION

This chapter has presented an overview of some aspects that must be taken into account when designing the hardware part of a perception system. The mathematical foundations of data projection and stereo vision, as well as some typical lidar data representations, have been described. These elements provide an essential background to face the object localization issue in Chapter 6.

On the other hand, the proposed automatic calibration approach, whose stereo data processing branch was introduced in this chapter, is intended to provide a tool that reduces the effort required to perform extrinsic calibration. Results on simulation prove that errors several orders of magnitude lower than existing approaches can be achieved. The code was released under the GPL-2.0 license and got the attention of a significant number of users around the world¹⁸, thus confirming the convenience of the method.

¹⁸ As of September 2019, the GitHub repository https://github.com/beltransen/velo2cam_calibration had been starred 109 times and forked 46 times.

Object detection is, perhaps, the most critical task to be performed by the perception system of an automated driving system. Dynamic objects may eventually interfere with the trajectory of the vehicle, posing a severe risk; on the other hand, static objects belonging to the infrastructure convey crucial information for vehicle navigation.

This chapter is devoted to the introduction of the object detection framework (in images) that make up the backbone of this thesis. The proposal relies on a well-known object detection meta-architecture, Faster R-CNN [213], which was one of the first real-time-capable DNN-based detection approaches, but it is still today the basis upon which a significant number of detection methods is built.

Some tweaks are proposed later to improve the capabilities of the method when applied to onboard applications. Furthermore, the influence of data used for training is also analyzed in this chapter. It will be shown that training data is indeed one of the factors that have the most influence on the performance of the detection system.

Finally, a proposal for the use of additional sources of information (specifically, stereo-vision depth estimation) in the detection framework is discussed.

4.1 FASTER R-CNN PARADIGM

Object detection in images is the problem of finding all the objects that can be seen in the frame and providing an estimate of their location in image coordinates, as well as a prediction of the category to whom they belong among a list of predefined alternatives.

The rest of the modules down the pipeline expect the object detection system to provide a list of instances defined by a bounding box and an assigned class. Bounding boxes can be encoded using the coordinates of the top-left and bottom-right corners, (x_1, y_1, x_2, y_2) , or the coordinates of the top-left corner and the dimensions of the box, (x, y, w, h) . Classes, on the other hand, are commonly provided as a probability distribution over the possible outcomes.

Faster R-CNN [213] embeds the inference of this list of objects into an end-to-end framework that takes an RGB image as input. The selection of this meta-architecture in this thesis was made due to its compelling set of features:

1. Since it was conceived as an end-to-end framework, the constituent elements of the network (i. e., weights and biases) are

This chapter includes content from [111], [113], [6], [115], and [114].

PROBLEM
FORMULATION

FASTER R-CNN

learned via backpropagation, thus reducing to a minimum the number of tunable parameters.

2. No assumptions are made at any point in the process about the position of the objects in the image, which makes it robust against complicated traffic scenarios.
3. The number of categories considered in the classification stage has a minimal impact on the performance, regarding both the accuracy and the processing time. Increasing the number of classes only raises the number of parameters of the last set of layers.
4. Although it depends on the hardware and the specific implementation in use, the framework was designed with real-time performance in mind and is therefore suitable for onboard applications.

The nuts and bolts of Faster R-CNN are introduced in the following sections¹.

4.1.1 Design principles

STAGES The Faster R-CNN meta-architecture, depicted in Fig. 4.1, is divided into two well-differentiated stages:

REGION PROPOSAL: This stage is responsible for extracting candidates; that is, the bounding boxes that are more likely to contain objects. This task is performed by a *Region Proposal Network (RPN)* that assigns an *objectness* score to a set of predefined *anchors* covering the whole of the image. Classification (and bounding box refinement) of the set of anchors centered in a pixel is performed using a fixed-sized window of features around that pixel.

CLASSIFICATION: The second step assigns a classification label and provides a refinement of the location and size for each bounding box. A fixed-length feature vector describing each proposal is fed into several stacked Fully-Connected (FC) layers, and finally, goes through two function-specific *heads* that estimate probability distributions for the classification and the bounding box regression, respectively.

STRENGTHS Both stages make use of the same data, which comes from a common core: the feature extractor. The feature extractor is made of the first stages of a CNN that acts as the *backbone* of the network; that is,

¹ Information in this section was obtained from the original Fast [95] and Faster R-CNN [213], [214] publications, as well as the source code in <https://github.com/rbgirshick/py-faster-rcnn> and other secondary sources.

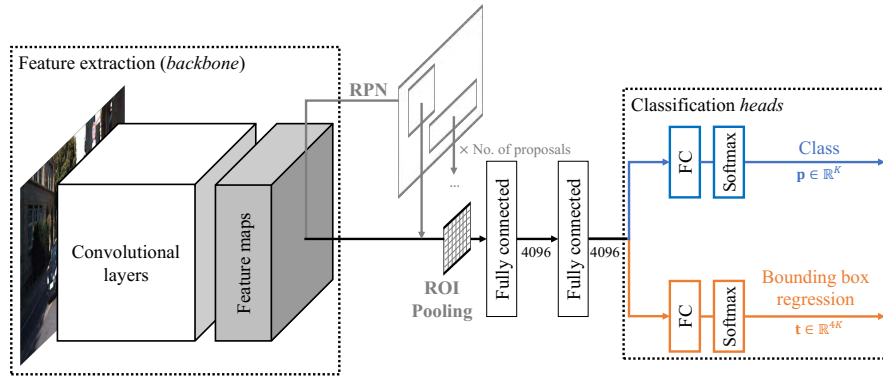


Figure 4.1: Faster R-CNN overview

several stacked blocks of “convolution + non-linearity + pooling” operations. The fact that feature extraction is shared between both stages means that the computationally costly encoding of the image pixels into smaller feature maps is done only once. This property is one of the critical aspects that make this framework suitable for real-time applications.

The link between the region proposal and the classification stages is provided by the *ROI pooling* layer. Anchors selected by the RPN are mapped to the last set of feature maps, and then, a fixed-length feature vector is extracted for each proposal. This operation is effectively performed by dividing the proposal into a fixed number of cells and computing the maximum value of the feature maps within each cell (max pooling). This is a smart way to handle objects at different scales and is one of the reasons why this two-stage scheme can cope well with objects represented by just a few pixels in the image.

On the negative side, it should be noted that every proposal must go through several FC layers. That leads to a hard balance between the recall of the set of proposals and the computing requirements. On the other hand, inference on the RPN is performed based on a limited window of values cropped from the feature maps around the center of the anchor. Therefore, although different anchor scales and aspect ratios are used, the area of the image that is effectively used to perform classification in the RPN, i. e., the *receptive field*, has a fixed size regardless of the actual size of the anchor.

LIMITATIONS

4.1.2 Training

The loss function used to train Faster R-CNN by backpropagation accounts for the different outcomes produced by the multi-task framework, including the intermediate ones from the RPN:

TASKS

1. Proposal objectness. Estimated by the RPN for each anchor, is a categorical probability distribution:

$$\mathbf{a} = (a_0, a_1), \quad (4.1)$$

with a_1 being the probability of the proposal representing an object.

2. Proposal refinement. As the (x, y, w, h) encoding is used throughout the network, the RPN proposes a box regression expressed as an offset relative to each anchor box:

$$\mathbf{b} = (b_x, b_y, b_w, b_h) \quad (4.2)$$

3. Classification. The classification head provides a categorical distribution \mathbf{p} that describes the probabilities of each proposal belonging to each of the K possible categories and an additional catch-all *background* class:

$$\mathbf{p} = (p_0, \dots, p_K), \quad (4.3)$$

where p_0 is the probability of the proposal not being an object, and p_i the probability of the proposal representing an object with the category i .

4. Bounding box refinement. The size and position of every bounding box are regressed using offsets from the original proposal. The regression is class-aware so that four different offset values (t_x, t_y, t_w, t_h) are inferred for each class k :

$$\mathbf{t}^k = (t_x^k, t_y^k, t_w^k, t_h^k), \quad k = 0, \dots, K \quad (4.4)$$

APPROXIMATE JOINT TRAINING

Although the original Faster R-CNN approach trained the RPN and the classification heads separately, it was soon shown [214] that an *approximate joint training* strategy could be used without degrading the performance significantly. This strategy performs training through SGD and, for each iteration, propagates back a combination between both the RPN and the Fast R-CNN losses to the backbone. However, proposal boxes from the RPN are assumed fixed despite being produced by the network as well, so this training strategy is ultimately an approximation.

LOSS FUNCTIONS

The set of parameters of the different stages of the pipeline, i. e., backbone, RPN, and classification heads, is trained through a multi-task loss. Consider the following per-element loss functions:

1. Multinomial logistic loss (or *log* loss), used for multilabel classification problems:

$$L_{\text{cls}}(\hat{\mathbf{p}}_n, \mathbf{l}_n) = -\log \hat{p}_{n, l_n}, \quad (4.5)$$

which assumes that each element n of the batch produces \hat{p}_n , which is a vector whose i th element ($\hat{p}_{n,i}$) represents the estimated probability of the class i , and l_n is the ground-truth class.

2. Smooth-L1 loss, used for box regression:

$$L_{\text{loc}}(\hat{\mathbf{t}}_n^u, \mathbf{v}_n, \sigma) = \sum_{\xi \in \{x, y, w, h\}} \text{smooth}_{L_1}(\hat{t}_{n,\xi}^u - v_{n,\xi}, \sigma), \quad (4.6)$$

where $\hat{\mathbf{t}}_n^u$ is the estimated bounding box regression according to Eq. 4.4, \mathbf{v}_n is the set of regression targets, (v_x, v_y, v_w, v_h) , and smooth_{L_1} is a robust L_1 loss that copes well with outliers:

$$\text{smooth}_{L_1}(x, \sigma) = \begin{cases} 0.5(\sigma \cdot x)^2 & \text{if } |x| < \frac{1}{\sigma^2} \\ |x| - \frac{0.5}{\sigma^2} & \text{otherwise} \end{cases} \quad (4.7)$$

Note that σ is a parameter that controls the resemblance of smooth_{L_1} with a hard L_1 loss². The value of this parameter affects the behavior of the loss function, as will be shown soon.

In Faster R-CNN, the regression targets for a box, $\mathbf{v}_n = (v_x, v_y, v_w, v_h)$, are defined in terms of the proposal, $P = (P_x, P_y, P_w, P_h)$, and the ground-truth box, $G = (G_x, G_y, G_w, G_h)$, encoding parameters:

$$\begin{aligned} v_x &= (G_x - P_x)/P_w \\ v_y &= (G_y - P_y)/P_h \\ v_w &= \log(G_w/P_w) \\ v_h &= \log(G_h/P_h) \end{aligned} \quad (4.8)$$

Hence, the multi-task loss employed to train the network parameters is the sum of four different losses, corresponding to the tasks defined at the beginning of this section:

MULTI-TASK LOSS
FUNCTION

1. Proposal objectness. The RPN proposal classification is defined as a multinomial classification problem with two possible outcomes: *background* and *object*:

$$L_1 = \frac{1}{N_{B_1}} \sum_{j \in B_1} L_{\text{cls}}(\mathbf{a}_j, u_j) \quad (4.9)$$

where \mathbf{a}_j is the probability distribution provided by the network (according to Eq. 4.1), and u_j , the ground-truth class. As shown, the loss is aggregated and normalized over a randomly sampled *mini-batch* of anchors (B_1) ³.

² For large values of σ (e. g., $\sigma > 3$), $\text{smooth}_{L_1}(x, \sigma)$ gradually becomes $|x|$.

³ By default, $B_1 = 256$.

2. Proposal refinement. The loss associated with the RPN regression is:

$$L_2 = \frac{1}{N_{B_1}} \sum_{j \in B_1} u_j L_{\text{loc}}(\mathbf{b}_j, \mathbf{b}_j^*, 3) \quad (4.10)$$

with \mathbf{b}_j representing the offset provided by the network (Eq. 4.2) and \mathbf{b}_j^* , the regression targets as defined in Eq. 4.8. Please note that the ground-truth class of the proposal, u_j , appears as a factor, so the proposals whose ground-truth label is *background* ($u_j = 0$) do not contribute to this loss. This loss is also aggregated over the mini-batch (B_1). The parameter σ of L_{loc} is set to 3 to mitigate the effects of the lack of normalization of the RPN regression targets⁴.

3. Classification. The classification of the proposals from the RPN involves the following loss:

$$L_3 = \frac{1}{N_{B_2}} \sum_{i \in B_2} L_{\text{cls}}(\mathbf{p}_i, v_i) \quad (4.11)$$

where \mathbf{p}_i is the estimated probability distribution over the $K + 1$ possible classes and v_i , the ground-truth class. The loss is normalized over the set of proposals resulting from an NMS performed on the RPN output (B_2).

4. Bounding box refinement. The regression loss is computed by:

$$L_4 = \frac{1}{N_{B_2}} \sum_{i \in B_2} [v_i \geq 1] L_{\text{loc}}(\mathbf{t}_i^{v_i}, \mathbf{t}_i^{v_i^*}, 1) \quad (4.12)$$

where $\mathbf{t}_i^{v_i}$ is the vector of offsets estimated by the network for the ground-truth class⁵ v_i , and $\mathbf{t}_i^{v_i^*}$, the vector of regression targets. Ground-truth regression targets are normalized so that they have zero mean and unit variance and, therefore, the parameter σ of L_{loc} is set here to 1. On the other hand, the *Iverson bracket* applied to a logical proposition P is defined as:

$$[P] = \begin{cases} 1 & \text{if } P \text{ is true} \\ 0 & \text{otherwise} \end{cases} \quad (4.13)$$

Therefore, the term $[v_i \geq 1]$ is included in the loss to prevent the background bounding boxes ($v_i = 0$) from contributing to the regression loss.

⁴ <https://github.com/rbgirshick/py-faster-rcnn/issues/89>

⁵ It should be recalled that the bounding box regression is class-aware, so four offset parameters are separately estimated for each class.

In all cases, the assignment of an anchor or detection to a particular ground-truth annotation is done based on the **IoU** overlap between the two instances. The final multi-task loss to be optimized during training, L , is defined as:

$$L = \sum_{i=1}^4 \alpha_i L_i, \quad (4.14)$$

where α_i are the weights associated with the different terms of the loss. In the proposed approach, $\alpha_i = 1 \forall i \in \{1, 2, 3, 4\}$, so that all the tasks have an equal contribution to the final loss.

4.2 TUNING FOR TRAFFIC ENVIRONMENTS

Even though Faster **R-CNN** is mainly agnostic to the application by not making any assumption, some hyperparameters can be tuned to optimize the performance of the system on the particular case of traffic environments.

In order to adapt the system to the particular characteristics of traffic environments, extensive experimentation based on the KITTI 2D object detection benchmark, one of the components of the KITTI Vision Benchmark Suite [90] introduced in Sec. 2.2.2, has been performed. The accurate and detailed object labels featured by the KITTI dataset make this dataset an ideal choice for this purpose. In this section, a detailed description of the KITTI 2D object detection benchmark is provided, before introducing the set of proposed adjusts to the Faster **R-CNN** framework motivated by the conducted experiments.

4.2.1 KITTI object detection benchmark

The KITTI object detection benchmark [90] consists of 14 999 data frames (synchronized stereo images and lidar point clouds) endowed with labels describing all the relevant objects in the **FOV** of the cameras. Each object is represented by several attributes, such as its bounding box coordinates in the image, its category, its occlusion and truncation states, and its 3D location and dimensions. The possible categories are listed below, each represented by a 3-letter code that will be used hereon in this document:

CAR	<i>Car</i>
PED	<i>Pedestrian</i>
CYC	<i>Cyclist</i>
VAN	<i>Van</i>
TRK	<i>Truck</i>
SIT	<i>Person sitting</i>
TRM	<i>Tram</i>

CATEGORIES

There are two additional categories: a catch-all *Misc* (Miscellaneous) category, and a *DontCare* category. The *DontCare* category is intended to encompass all the regions containing non-labeled objects, usually because they are too far away. These regions are not considered either false positives or false negatives in the evaluation.

TRAIN/VAL
SPLITS

Only labels corresponding to 7481 frames (the *training* set) are publicly available. The rest of the frames (7518 frames, the *testing* set), are used to perform a fair evaluation in a public server, where a leaderboard of methods is on display⁶. During this thesis, experiments have been performed using the *training* set. Eventually, final results have been submitted to the official server to obtain stats on the *testing* set.

As usual in machine learning, the KITTI training set has been conveniently split into a proper training subset and a validation subset. Frames in the KITTI object benchmark come from continuous sequences recorded on different days; ideally, frames in the training subset should belong to different sequences than frames in the validation subset, given that data from the same sequence can be pretty similar. Different train/validation splits are available:

- The split by Xiang *et al.* [280], which features 3682 training images and 3799 validation images (50:50 ratio).
- The split by Chen *et al.* [43], [44], [46], with 3712 training images and 3769 validation images (50:50 ratio), which is lately gaining traction in the literature to establish comparisons between methods.
- A split proposed in this thesis, with 5415 training images and 2065 validation images (70:30 ratio).

As an example, Table 4.1 shows the number of instances in each subset for the last split introduced above. The table contains the aggregated numbers in the official *training* set too.

	CAR	PED	CYC	VAN	TRK	SIT	TRM
Train samples	20 609	2476	1042	2407	965	222	511
Val. samples	8120	2010	584	506	129	—	—
Total	28 729	4486	1626	2913	1094	222	511

Table 4.1: Object occurrence statistics for the custom train and validation KITTI subsets [114]

SENSOR CHARAC-
TERISTICS

Regarding the sensor characteristics, images in the KITTI object detection benchmark have different sizes around 1242×375 . The reason behind this disparity is that, even though all the images were captured

⁶ http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=2d

using the same sensor, different crops were applied later. The 1/2" sensor, together with a 4 mm lens, provides an HFOV of 90°. After the crop, the VFOV is 35°. Although two different stereo rigs are available, the color cameras, with a baseline of 54 cm, will be used throughout this work.

Object labels are divided into three levels of difficulty attending to the size, occlusion, and truncation criteria reported in Table 4.2. Note that, according to this definition, each difficulty level includes all the samples belonging to the lower difficulty levels as well.

LEVELS OF DIFFICULTY

LEVEL	MIN. HEIGHT	MAX. OCCLUSION	MAX. TRUNCATION
Easy	40 px.	Fully visible	15 %
Moderate	25 px.	Partly occluded	30 %
Hard	25 px.	Difficult to see	50 %

Table 4.2: Definition of the levels of difficulty from the KITTI dataset

The evaluation criteria used in the KITTI benchmark for 2D detection is the *Average Precision (AP)*, introduced in [71]. When dealing with the category k , AP is defined as:

METRICS

$$AP_k = \frac{1}{11} \sum_{r \in \{0, 0.1, \dots, 1\}} p_{\text{interp}}(r), \quad (4.15)$$

where $p_{\text{interp}}(r)$ is an interpolation of the precision at each recall level, $p(r)$, for all the recall levels above the current one:

$$p_{\text{interp}}(r) = \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r}) \quad (4.16)$$

Because of this interpolation, $p_{\text{interp}}(r)$ becomes a monotonically increasing function, thus smoothing the precision curve. The AP measure requires high precision at all levels of recall, penalizing methods that focus on a small subset of examples. A condensed version of the measure, the *mean Average Precision (mAP)*, aggregates the AP across all the foreground categories:

$$mAP = \frac{1}{K} \sum_{k=1}^K AP_k, \quad (4.17)$$

Detections are assigned to ground-truth labels according to the IoU overlap. True positives are required an overlap higher than 50% for the *Pedestrian* and *Cyclist* categories, and higher than 70% for the *Car* category. Official evaluation is restricted to those three categories since authors argue that the number of samples for the rest of the categories is not enough to perform comprehensive evaluation⁷.

⁷ This statement is included in the *readme* file of the KITTI 2D object detection benchmark development kit.

Other measures are proposed by the KITTI benchmark to evaluate the performance in pose estimation. They will be discussed in Secs. 5.1.1 and 6.1.3.1.

4.2.2 Hyperparameter tuning of the detection framework

Three main types of strategies have been adopted to optimize the performance of the Faster R-CNN detection framework in the KITTI dataset. The modifications affect, among others, the input scale, the training samples, and the RPN anchors, as described in the following sections.

4.2.2.1 Input scale

Faster R-CNN has proven highly sensitive to the size of the input images [72]: the larger the scale, the better the results, to a certain extent. However, larger input scales slow down the feature map generation. In this work, unless otherwise stated, input images are rescaled to the uniform scale of 500 pixels in height, approximately corresponding to a scaling factor of $1.33\times$. For KITTI images, the resulting resolution is around 1656×500 pixels and is used for both training and validation. This scale offers a good trade-off between accuracy and processing time and is in line with other works [285]. Further discussion will be provided in Sec. 5.3.4.

4.2.2.2 Training samples

HARD SAMPLES

There are two points relevant to the training procedure used with the KITTI dataset. On the one hand, only samples meeting the criteria of the *Hard* difficulty level in Table 4.2 are employed, in order to avoid feeding the network with overcomplicated samples. This filtering leads to the removal of a surprisingly high proportion of samples, as shown in Table 4.3 for the three main classes using the custom split.

	CAR	PED	CYC
Train samples	20 609	2476	1042
Train samples within <i>Hard</i>	15 526	2355	711

Table 4.3: Number of CAR, PED, and CYC instances meeting the requirements of the *Hard* difficulty level in the KITTI dataset

DONTCARE SAMPLES

On the other hand, *DontCare* regions are properly handled to prevent them from corrupting the training. Following the same procedure as the vanilla Faster R-CNN, where anchors and proposals are required

to belong unambiguously to a category⁸, the following measures are adopted:

1. At the *RPN*, the anchors with the highest overlap with each *DontCare* area are discarded.
2. Also at the *RPN*, anchors with an overlap higher than 15% with a *DontCare* region are excluded from the mini-batch.
3. At the classification *heads*, proposals with an overlap higher than 25% with a *DontCare* region are not considered as *background* samples.

This design prevents introducing spurious samples in the training procedure (e. g., distant cars as *background*), as areas of the image with non-labeled objects (i. e., *DontCare* regions) are not employed during the training procedure, neither as positive nor as negative samples.

Additionally, categories not used during the training (e. g., *Misc*), as well as samples harder than the *Hard* difficulty level, are also considered as *DontCare* samples. The reason behind this decision is the difficulty involved in distinguishing the different kinds of traffic participants; for instance, the *Misc* category includes trailers that could be easily confused with cars. These measures aim to avoid ambiguous samples that are difficult to grasp even for a human being, which could hurt the optimization process.

OTHER
CATEGORIES

4.2.2.3 Class balancing

As apparent from Table 4.1, the occurrence frequency of the different categories in the KITTI dataset is strongly unbalanced. This is a common problem in virtually every automotive-oriented dataset and can lead to models biased towards cars at the expense of *VRUs*, which are particularly sensitive from a safety point of view.

In order to deal with this issue, a basic class balancing method is proposed. It is based on a weighted classification loss where samples belonging to different classes contribute differently to the final loss, depending on the frequency of each category in the training set.

To that end, the classification loss L_3 in Eq. 4.11 is modified by replacing the multinomial logistic loss L_{cls} with an *information gain* (or *infogain*) loss L_{inf} , defined as follows⁹:

$$L_{inf}(\hat{\mathbf{p}}_n, \mathbf{l}_n) = \sum_{k=1}^K H_{l_n, k} \log(\hat{p}_{n, k}), \quad (4.18)$$

⁸ In the original Faster *R-CNN*, anchors should have an overlap higher than 70% with a ground-truth box to be considered a positive sample, and an overlap lower than 30% with any ground-truth box to be considered a negative sample. Proposals fed to the classification heads are imposed similar requirements: overlap higher than 50% for the positive ones, and non-existent for the negative ones.

⁹ <http://caffe.berkeleyvision.org/tutorial/layers/infogainloss.html>

where $H_{l_n,k}$ is the element at the location (v_n, k) in the *infogain matrix* H , and $\hat{p}_{n,k}$, the predicted probability of sample n belonging to the class k ; i. e., the k th element of $\hat{\mathbf{p}}_i$.

If H is a diagonal matrix, then the infogain loss becomes:

$$L_{\text{inf}}(\hat{\mathbf{p}}_n, \mathbf{l}_n) = H_{l_n, l_n} \log(\hat{p}_{n, l_n}), \quad (4.19)$$

which can be seen as a weighted version of the multinomial logistic loss, where the weight assigned to the samples belonging to the class k is given by the value of $H_{k,k}$. This way, the underrepresented classes can be assigned higher $H_{k,k}$ values.

In particular, the infogain matrix values are selected according to the frequencies observed for the different categories in the training set using the following equation:

$$H_{k,k} = 2 \cdot \left(\frac{f_{\min}}{f_k} \right)^{\frac{1}{8}}, \quad (4.20)$$

where f_{\min} is the number of occurrences of the less frequent class and f_k , the number of instances of class k . Background samples are assigned a unit weight. Values obtained through this equation are shown in Table 4.4 for two different sets of training categories: the three main categories (CAR, PED, and CYC) and all the categories, respectively.

	BG	CAR	PED	CYC	VAN	TRK	SIT	TRM
CAR+PED+CYC	1	1.38	1.8	2	—	—	—	—
All categories	1	1.14	1.48	1.66	1.48	2	1.65	1.8

Table 4.4: Weights in the infogain matrix for two different sets of categories. BG denotes the weight assigned to *background* samples.

4.2.2.4 RPN anchors

Although the proposal regression performed by the RPN should be capable of handle pretty large deviations from the original anchors through the proposal regression estimation, it has been shown [214] that using different scales and aspect ratios per location improves the performance of the system. Therefore, it makes sense to modify these values to fit the geometries featured by road participants. A statistical analysis of the KITTI samples provides the results shown in the histograms in Figs. 4.2 and 4.3, where W is the width of the annotations, and H , their height. Please note that areas are computed on the original, unscaled image.

Despite that it has been proven that adding more anchors would benefit the accuracy [281], the number of anchors of the proposed setup follows the original model: three scales and three aspect ratios

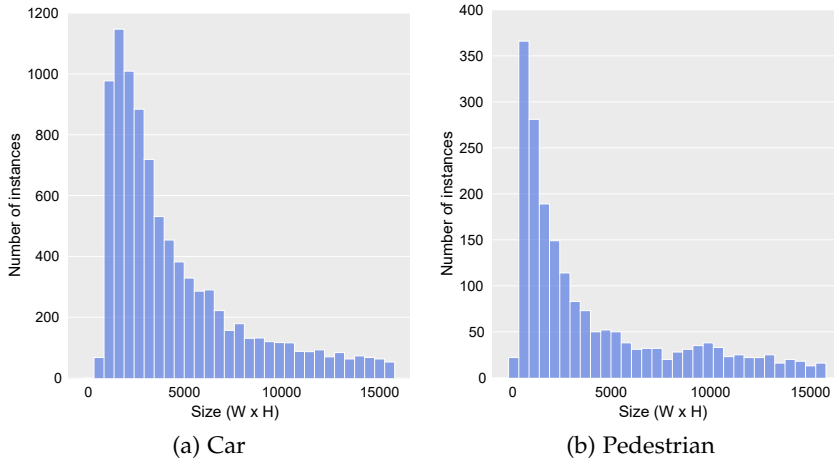


Figure 4.2: Histograms of sizes ($W \times H$) for *Hard* samples in the KITTI training subset

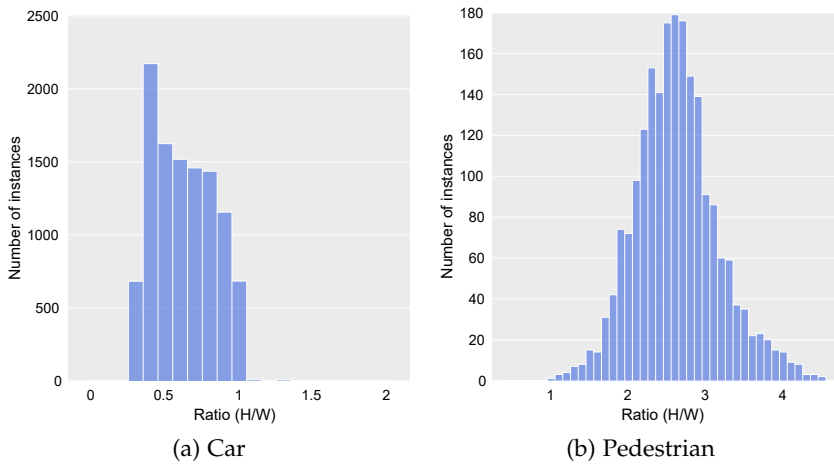


Figure 4.3: Histograms of ratios (H/W) for *Hard* samples in the KITTI training subset

for a total of nine anchors per location. This way, the complexity of the model does not increase, and computation times remain tractable. The new anchors have been selected to fit better the typical shapes and scales without loss of generalization ability. The values are reported in Table 4.5 and graphically represented in Fig. 4.4. In both cases, scales are provided as the areas of the resulting anchors.¹⁰

As apparent from the figure, the new anchor shapes are closer to the ones typically featured by cars (ratios 0.4 and 0.7) and pedestrians (ratio 2.5), which, in general, reduces the magnitude of the regression needed to fit the ground-truth boxes. The new scales, on the other

¹⁰ Anchors, which are rectangles of size $W \times H$, are obtained by modifying the aspect ratio of a base box of 16×16 while maintaining its area. Afterward, a factor is applied to the sides of the box to get the provide them with the desired scale.

	DEFAULT	CUSTOM
Scales (box areas)	$\{128^2, 256^2, 512^2\}$	$\{80^2, 112^2, 144^2\}$
Ratios (H/W)	$\{2 : 1, 1 : 1, 1 : 2\}$	$\{5 : 2, 7 : 10, 2 : 5\}$

Table 4.5: Modified settings for RPN anchors [113]

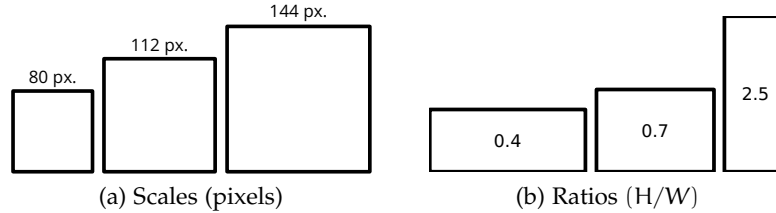


Figure 4.4: Custom RPN anchors

hand, are more suitable for the representation of small and distant objects.

4.2.2.5 Dropout and layer freezing

DROPOUT

As discussed in 2.5.3, *dropout* [248] is a regularization technique aimed at reducing overfitting in DNNs by randomly removing units in the network during training.

Faster R-CNN originally employed dropout in the last two common FC layers before the classification *heads*. However, its effect for this particular setup is dubious, as will be experimentally proven in Sec. 5.3.1. As a result, dropout is not used in the models trained in this thesis. Nevertheless, no overfitting symptoms were observed.

LAYER FREEZING

On the other hand, as is standard practice, initial weights are loaded from a model pre-trained on ImageNet and then fine-tuned. In the baseline Faster R-CNN, however, fine-tuning does not reach every layer in the backbone, as the authors consider that the first convolutional layers are “generic and task-independent,” [95] so ImageNet weights should be valid for other applications.

However, in this thesis, it was observed that, for the considered application, extending backpropagation to the shallowest layers had a significant beneficial effect. Therefore, for all models in this document, fine-tuning was applied to the whole *backbone* network.

4.2.2.6 NMS

A greedy NMS is applied over the resulting bounding boxes of the Faster R-CNN network to remove duplicate detections. The NMS is usually performed on a per-class basis; however, in the particular case of road participants, it makes sense to consider some pairs of categories together based on their similarity. More specifically, two

bundles are considered: *Car*, *Van*, and *Truck*, on the one hand, and *Pedestrian*, *Person sitting*, and *Cyclist*, on the other hand.

This strategy allows removing redundant detections representing a rider as a pedestrian, or a van as a truck, for instance. Note that this is a test-time modification that improves the result provided to higher-level modules, but should not have an impact on the quantitative evaluation performed under the KITTI criteria, which already ignores misclassifications between neighbor categories.

4.2.3 Experimental setup and preliminary assessment

The different findings that will be introduced henceforth are supported by experimental evidence. All the experiments follow the same pattern: a Faster R-CNN-like model is trained¹¹ on one of the KITTI training subsets, and validated on its respective validation subset according (mostly) to the KITTI evaluation criteria.

Apart from the modifications described in the previous section, most parameters and configurations follow the setup used in the original R-CNN papers [95], [96], [214]. Thus, training is performed following the *image-centric* strategy, where all the samples for an SGD iteration are obtained from a single image, and 256 anchors are sampled to calculate the RPN losses. Later, 300 proposals are extracted and fed to the classification heads to compute its losses. In each iteration, frames are horizontally flipped with a certain probability as a way to augment the training data.

The bulk of the experimental analysis of Faster R-CNN and other proposed variants will be provided in Chapter 5 in the context of joint detection and viewpoint estimation. However, a baseline evaluation on the KITTI dataset with the modifications proposed in the previous sections is presented here.

For each set of experiments, the set of hyperparameters and configurations employed to train the network will be specified. In this case, the setup is summarized in Table 4.6.

An explanation of the different parameters in the table is provided below:

- *Feature extractor*: the *backbone* of the model. Usually, the 16-layer VGG (VGG-16) [244] is employed.
- *Pre-training*: it indicates if the model has been trained by fine-tuning a pre-trained model.
- *Anchors*: the set of anchors (scales and aspect ratios) employed by the RPN.

COMMON SETUP

BASELINE
EVALUATION

¹¹ The term *trained* is used here as equivalent of *fine-tuned from a pre-trained model*.

Feature extractor:	VGG-16 [244]
Pre-training:	Yes, on ImageNet
Anchors:	Custom (Table 4.5)
Classification loss:	Infogain
RPN proposals:	300
Classification heads:	Category + bounding box regression
Predicted classes:	3 (CAR, PED, CYC)
Train/val split:	Custom (70:30)
Training schedule:	50k iterations @ lr = 10^{-3} + 30k iterations @ lr = 10^{-4}

Table 4.6: Training hyperparameters for the tuned Faster R-CNN

- *Classification loss*: the type of loss used in L_3 . It can be the regular multinomial logistic loss (L_{cls}) or the weighted version implemented with the infogain loss (L_{inf}).
- *Classification heads*: the sibling FC branches which provide the results of the inference. In addition to the usual heads for category prediction and bounding box regression, new branches can be added to extend the capabilities of the framework, as will be discussed in the next chapter.
- *Predicted classes*: the number of classes for which the model is trained. The output of the FC layer responsible for category prediction is made of this number of elements (plus the one corresponding to the *background* class). Additionally, as the bounding box regression is class-aware, it affects also to the output size of the regression branch.
- *Train/val split*: the split of the publicly available *training* set determining the samples used for training and validation, respectively.
- *Training schedule*: number of iterations and learning rate. A *step decay* schedule is adopted in all the experiments, so the learning rate is reduced by some factor after a set number of training iterations. The learning rate and width (in iterations) corresponding to each step are specified.

Table 4.7 shows the results of the tuned Faster R-CNN trained for the detection of the three main categories on the KITTI dataset. Comparison with other methods is complicated due to the diversity of train/validation splits used in the literature and is, therefore, delayed to the next chapter, where the official testing set will be used. The effect of the different hyperparameters will also be analyzed then.

CLASS	2D DETECTION (AP 2D)		
	EASY	MOD.	HARD
CAR	88.20	77.87	63.64
PED	81.41	70.52	66.39
CYC	72.65	53.73	51.92
mAP	80.76	67.37	60.65

Table 4.7: Detection performance (AP %) of the tuned Faster R-CNN on the KITTI validation subset

4.2.3.1 Implementation and inference time

The tuned Faster R-CNN model, as well as the rest of the modifications described in this and the next chapter, were implemented in Python using the Caffe DL framework [141]¹². This implementation was based on the one by Ross Girshick¹³, which is, in turn, a re-implementation of the original MATLAB code by Shaoqing Ren¹⁴. The modified code has been released under the same MIT license¹⁵ to ease the reproducibility of the results.

SOURCE CODE

Caffe makes use of the NVIDIA CUDA¹⁶ parallel computing platform to perform most of the CNN operations on GPU, thus reducing training and inference times. Experiments in this thesis were performed using a Titan Xp GPU¹⁷. Some of its specs are reported in Table 4.8, including its theoretical performance measured in *float point operations per second* (TFLOPS) for the *single-precision floating-point format* (FP32), which is the one used by the current implementation.

HARDWARE

CUDA cores (shading units):	3840
Boost clock:	1582 MHz
Memory config:	12 GB GDDR5X
Memory bandwidth:	547.7 GB/s
TFLOPS FP32:	12.15 TFLOPS

Table 4.8: NVIDIA Titan Xp GPU specifications¹⁸

The forward-pass time for each KITTI frame is around 83 ms¹⁹; therefore, automotive-capable output rates are doable using dedicated

RUN TIME

¹² Caffe is open source under a BSD 2-Clause license: <https://github.com/BVLC/caffe>

¹³ Open source under the MIT license: <https://github.com/rbgirshick/py-faster-rcnn>. Note that Ross Girshick is one of the authors of Faster R-CNN.

¹⁴ Open source under the MIT license: https://github.com/ShaoqingRen/faster_rcnn.

¹⁵ <https://github.com/cguindel/lsi-faster-rcnn>

¹⁶ <https://developer.nvidia.com/cuda-zone>

¹⁷ The GPUs used in this research were kindly donated by NVIDIA Corporation.

¹⁸ <https://www.techpowerup.com/gpu-specs/titan-xp.c2948>

¹⁹ This time was obtained as the median of forward-pass times for 1 000 frames from the KITTI testing set

implementations. For instance, some GPUs have native *half-precision floating-point format* (FP16) support, which speeds up computations without significant loss of accuracy [183].

4.3 ENHANCED DETECTION USING STEREO VISION

MOTIVATION

While Faster R-CNN features a satisfactory detection performance using only color images as input, additional sources of information can be incorporated into the detection pipeline to further improve the capabilities of the network.

Stereo vision has already been pointed out in previous chapters as a useful strategy to obtain geometrical information from images. In this section, a strategy to leverage stereo information aimed to enhance the solid detection baseline provided by Faster R-CNN is proposed. The method does not require extensive modifications from the original design, which, among other things, avoids an increase in the complexity of the model.

PROPOSAL OVERVIEW

The proposal is summarized in Fig. 4.5. The main idea is the inclusion of stereo depth information to the input image as a *fourth channel*, in addition to the existing R, G, and B color channels.

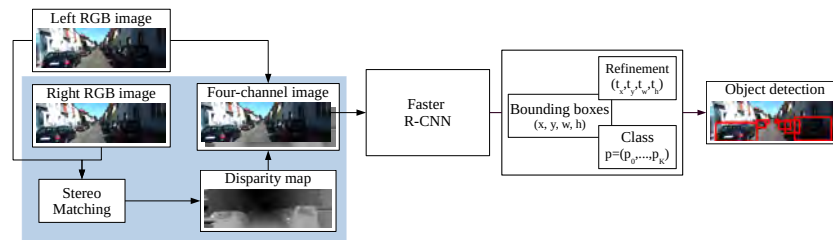


Figure 4.5: Proposed approach to incorporate stereo information into the Faster R-CNN framework [113]

The proposal affects only to the first convolutional layer of the feature extractor, whose filters must be extended to accept four channels as input. Dimensions of all the feature maps, both the intermediate and the final ones, remain unalterable from the original model.

The addition of depth information is aimed to improve mainly the region proposal stage by providing hints about the boundaries of the objects, thus helping to partially overcome the limitations of the RPN due to the fixed receptive field. On the other hand, this approach preserves the end-to-end nature of the Faster R-CNN and does not introduce significant overhead.

4.3.1 Depth information encoding

DISPARITY CHANNEL

Different approaches can be used to represent the depth information in the fourth channel. In this proposal, a scaled disparity map is used.

Following Eq. 3.10, the value of the pixel at coordinates (x, y) in the fourth channel, $I(x, y)$, is given by²⁰;

$$I(x, y) = \text{clip}(k \cdot d(x, y)) = \text{clip}\left(k \cdot f \cdot \frac{B}{Z(x, y)}\right), \quad (4.21)$$

where k is a constant scaling factor, and $\text{clip}(x)$ is a clipping function that limits the pixel value to a maximum of 255; that is, $\text{clip}(x) = \min(x, 255)$. This equation effectively performs normalization of disparity values between 0 and $255/k$, while all pixels with $d(x, y) \geq 255/k$ are assigned $I(x, y) = 255$.

In the experiments, the parameter k was set to 4. This way, only disparities originally in the range between 0 and 64 are distinguishable in the resulting map. In images from the KITTI dataset, that clipping corresponds to depths from 6 m to the infinite, which is an adequate range of detection for the application, given the FOV of the cameras.

The motivation behind using a scaled disparity map as a fourth channel, instead of alternatives such as a depth map, is twofold. On the one hand, the inverse relationship between depth and disparity leads to the disparity map having a higher resolution in closer values (with small $Z(x, y)$), where the criticality of the detection increases. On the other hand, the chosen normalization allows removing extremely close points, which will otherwise dominate most of the scale of the fourth channel, precisely because of the non-uniform resolution.

Two methods, already introduced in Sec. 3.2.1, have been employed to perform the stereo matching: the OpenCV SGBM implementation [124], and DispNet [181]. A background interpolation method is used with the SGBM disparity to remove *holes* (i. e., undefined values) in the disparity map. Every pixel (x_0, y_0) with an undetermined value in the disparity map, $\#d(x_0, y_0)$, is given a value according to:

$$\hat{d}(x_0, y_0) = \min(d(x_0^-, y_0), d(x_0^+, y_0)) \quad (4.22)$$

where $d(x_0^-, y_0)$ and $d(x_0^+, y_0)$ are the disparities of the closest defined pixels at the left and right sides in the same row. Fig. 4.6 depicts an example of the fourth channel arrays obtained through the two different stereo matching methods.

The VGG-16 model [244] is used as a feature extractor. As mentioned before, a common practice in CNN training is to initialize the weights using pre-trained values. Usually, pre-training in Faster R-CNN affects the feature extractor. As the structure of the first convolutional layer has been modified to accept four-channel images, pre-training on a monocular-only dataset, such as ImageNet, is unfeasible. Instead, all but these weights are loaded with ImageNet pre-trained values, and the filters whose structure has changed are initialized with made-up values computed as the mean value of the weights of the color

JUSTIFICATION

STEREO
MATCHING

TRAINING SETUP

²⁰ Please refer to Sec. 3.1.2 for an explanation of the concept of disparity and its mathematical definition.

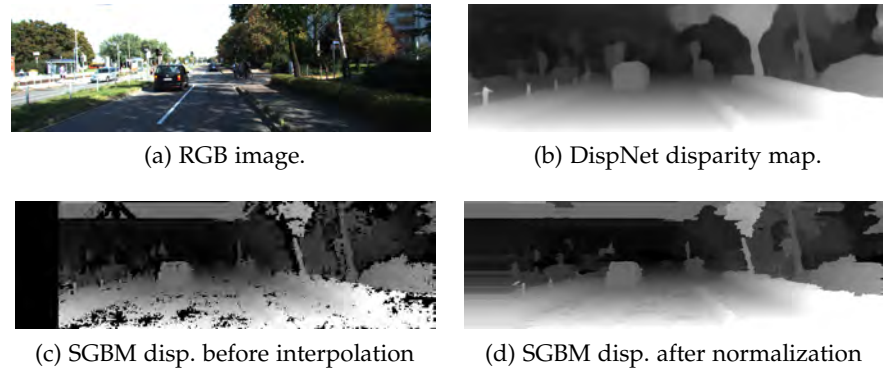


Figure 4.6: Example of the processed disparity maps on a frame from the KITTI dataset [113]

channels. This approach is based on the intuition that discontinuities in depth are usually linked to discontinuities in intensity. In any case, all network parameters, including those in the shallowest layers, are updated during fine-tuning, as explained in Sec. 4.2.2.5, which should help to fit the new nature of the data.

4.3.2 Experimental results

Experiments were conducted on the KITTI object detection benchmark. Training was performed according to the choices summarized in Table 4.9.

Feature extractor:	VGG-16 [244] (slightly modified)
Pre-training:	Yes, on ImageNet
Anchors:	Custom (Table 4.5)
Classification loss:	Multinomial logistic
RPN proposals:	300
Classification heads:	Category + bounding box regression
Predicted classes:	7 (CAR, PED, CYC, VAN, TRK, SIT, TRM)
Train/val split:	Chen et al. Chen <i>et al.</i>
Training schedule:	50k iterations @ $lr = 10^{-3}$ + 30k iterations @ $lr = 10^{-4}$

Table 4.9: Training hyperparameters for the proposed stereo-vision-capable Faster R-CNN

Note that, while the network is trained to predict seven categories, evaluation is restricted to the three considered in the official benchmark. Table 4.10 shows a comparison between the RGB baseline and the two alternatives presented to incorporate the stereo information, making use of two different stereo matching methods.

CLASS	METHOD	2D DETECTION (AP 2D)		
		EASY	MOD.	HARD
CAR	RGB	88.76	77.01	60.81
	RGB+SGBM	89.39	77.99	66.84
	RGB+DispNet	88.82	77.29	66.56
PED	RGB	85.97	68.71	61.41
	RGB+SGBM	87.39	69.16	63.62
	RGB+DispNet	87.70	69.73	64.47
CYC	RGB	65.22	53.67	50.37
	RGB+SGBM	64.07	52.25	49.68
	RGB+DispNet	66.51	55.77	52.26

Table 4.10: Comparison of the performance (AP % and AOS %) of the proposed stereo-vision-capable Faster R-CNN with other methods on the KITTI validation subset [113]

The straightforward addition of stereo data obtained with the DispNet method leads to an improvement of more than +1.1 points in *mAP* for *Moderate* objects and almost +3.6 points for the *Hard* set. The *SGBM* alternative, while generally worse than the DispNet method, also achieves an improvement of around 2.5 points in the *Hard mAP* and gets the highest *AP* values for *Car*.

These improvements are achieved with a negligible impact on the run time of the detection network, as the increase in the number of parameters and operations per frame is minimal. A non-negligible time is spent, however, in the stereo matching process, so this approach makes the most sense when the disparity map is used in other applications within the automated driving pipeline (e.g., traversable area detection [179]) and, therefore, would be computed in any case.

4.4 ANALYSIS OF THE INFLUENCE OF TRAINING DATA

As established before, most experimental results provided throughout this thesis make use of the KITTI 2D object detection benchmark, which is made of real data from onboard sensors that have been profusely annotated and contains a variety of challenging instances with different sizes, poses, and occlusion statuses. Results obtained on the KITTI object detection benchmark are, therefore, representative of the real performance of the algorithms on real situations.

However, the 7481 frames composing the official *training* set may not be enough to fulfill the requirements of modern *DL* algorithms, especially when split into two separated subsets for training and validation. This section is devoted to the study of the influence of training data on the performance of Faster *R-CNN*. The study is limited

here to the quantitative effect of augmenting the available training data. Extended results will be provided in Sec. 5.3.5 in the context of joint detection and orientation estimation.

4.4.1 Experimental setup

DATASETS Two different datasets are used in this analysis: the standard KITTI 2D object detection benchmark and the Cityscapes dataset [49]. Concerning KITTI, the train/validation split by Chen *et al.* [46] is employed, and the validation set is used as the reference testbed, enabling comparison between the different alternatives that will be analyzed.

FORMAT ADAPTATION Regarding the Cityscapes dataset, the 2975 available training frames with *fine* annotations are employed²¹. Cityscapes is aimed at semantic segmentation algorithms, so annotations are pixel-wise. However, instance information is also available, so bounding box labels similar to the ones in the KITTI dataset can be obtained. This adaptation enables the combination of both datasets to study the effect of an increase in the number of training samples²².

To that end, each bounding box is taken as the minimum enclosing rectangle of the set of polygons defining an instantiable object. Fig. 4.7 illustrates the procedure. Although all the instantiable categories can be transformed into bounding box labels, as shown in the figure, only the three main KITTI categories are considered.

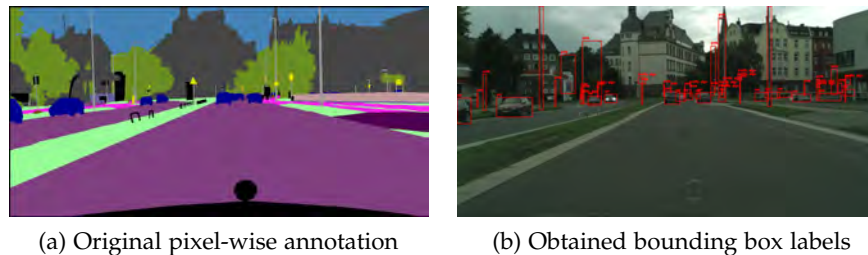


Figure 4.7: Example of the conversion from pixel-wise to bounding box labels. Figures by Iñigo Barredo from [6] (license CC BY-NC-ND 3.0 ES)

CATEGORIES The three main classes of the KITTI dataset are considered. Due to differences in the definition of KITTI and Cityscapes categories, the following criteria have been adopted:

- **CAR**: both definitions agree.
- **PED**: Cityscapes' *person* labels are assumed equivalent to the KITTI *Pedestrian* labels.

²¹ An additional set of 19 998 training images is available featuring *coarse* annotations, but these were found insufficiently precise for this purpose.

²² The adaptation of Cityscapes annotations to KITTI format was designed and performed by Iñigo Barredo during a TFG (Bachelor's Thesis) supervised by the author of this thesis [6].

- **CYC**: each Cityscapes' *rider* sample has been merged with the closest *bicycle* instance, following the *Cyclist* (**CYC**) category specification from the KITTI dataset.

The total number of samples from each dataset considered in the analysis is shown in Table 4.11.

CATEGORY	KITTI			CITYSCAPES
	TRAIN	VAL	TOTAL	TOTAL
<i>Car</i> (CAR)	10 753	10 963	21 716	21 637
<i>Pedestrian</i> (PED)	2104	2172	4276	15 788
<i>Cyclist</i> (CYC)	594	600	1194	1481

Table 4.11: Object occurrence stats for the KITTI and Cityscapes subsets used in the analysis [115] © 2018 IEEE

The adaptation procedure deals with two additional sources of difference between datasets: the occlusion and truncation statuses and the resolution and **FOV**. Regarding the former, assigning occlusion and truncation statuses to every sample enables discarding extremely hard instances from the training procedure, as stated in Sec. 4.2.2.2. Both magnitudes are estimated for every labeled sample from Cityscapes as follows:

OCCLUSION /
TRUNCATION

- **Occlusion**: an instance is labeled as occluded when its bounding box is intersected by another object in the foreground. The foreground-background ordering is available in the Cityscapes annotations. An estimate of the degree of occlusion is provided by computing the ratio between the area of the intersection and the area of the object box.
- **Truncation**: if any of the sides of the bounding box coincides with the image boundaries, the object is assumed truncated.

Although both are rough estimations, they are valid for the intended filtering purposes. Thresholds for the KITTI *Hard* samples were shown in Table 4.2; likewise, the difficulty criteria for the adapted Cityscapes samples have been established according to the occlusion and truncation estimations introduced above so that the maximum admissible occlusion level for *Hard* samples is 75%, and no truncation is allowed.

On the other hand, a comparison between KITTI images (e.g., Fig. 4.6a) and Cityscapes images (e.g., Fig. 4.7b) makes it clear that they are significantly different in terms of resolution and **FOV**. During the training procedure in Faster R-CNN, every input frame is rescaled to a fixed height, so these differences are undesirable. Therefore, an **ROI** of 2048×620 is extracted from the Cityscapes images to make them similar to the KITTI frames regarding the **VFOV**; additionally,

RESOLUTION /
FOV

this cropping removes the hood of the ego-car, which is otherwise visible in the bottommost part of the images.

4.4.2 Analysis

As stated before, the main goal of the analysis is to study the effect of enlarging the training set on the detection capabilities of Faster R-CNN. The procedure consists of adding the 2975 training frames from Cityscapes on top of the KITTI training subset, made of 3682 images. In each training iteration, an image is randomly chosen from the mix of both datasets to perform the SGD. Table 4.12 shows the setup used in the experiment.

Feature extractor:	VGG-16 [244]
Pre-training:	Yes, on ImageNet
Anchors:	Default
Classification loss:	Infogain
RPN proposals:	300
Classification heads:	Category + bounding box regression
Predicted classes:	3 (CAR, PED, CYC)
Train/val split:	Chen <i>et al.</i> (KITTI) and Cityscapes training set
Training schedule:	50k iterations @ $lr = 10^{-3}$ + 30k iterations @ $lr = 10^{-4}$

Table 4.12: Training hyperparameters for the analysis of the influence of training data on the detection performance

Detection performance is analyzed on the KITTI validation subset. Table 4.13 shows the results obtained when training with each of the two datasets separately, and the performance obtained with the mix of both.

As expected, the model trained only on Cityscapes performs worse on the KITTI validation subset than the one trained on the training subset of the same dataset. However, combining the Cityscapes frames with the KITTI samples improves the results from the baseline obtained with the KITTI training subset alone for all the categories and difficulty levels. The overall effect is an increase of +4.24 points in mAP for the *Moderate* level, which is an improvement of around 6.25% with respect to the baseline model.

Results show that the increase in diversity introduced by the additional Cityscapes samples is beneficial for the development of the CNN model, even when the validation set is exclusively made of KITTI frames. The conclusion is especially noteworthy due to the notable differences between both datasets regarding the properties of their respective vision systems. Further discussion will take place in Sec. 5.3.5.

CLASS	TR. DATA	EASY	MOD.	HARD
CAR	KITTI	90.05	79.32	70.04
	Cityscapes	81.37	63.66	53.47
	KITTI + CS	90.31	84.94	70.33
PED	KITTI	75.80	67.17	58.58
	Cityscapes	72.00	63.92	55.33
	KITTI + CS	77.77	68.72	60.05
CYC	KITTI	77.47	56.96	54.64
	Cityscapes	63.09	50.14	46.85
	KITTI + CS	82.90	62.50	58.05

Table 4.13: Detection performance (AP %) of Faster R-CNN on the KITTI validation subset for different sets of training data. KITTI + CS denotes the combined KITTI-Cityscapes dataset [115] © 2018 IEEE

4.5 CONCLUSION

In this chapter, it has been proven that the state-of-the-art object detection meta-architecture Faster R-CNN is suitable to perform detection on images from an onboard camera and provide reliable identification of the obstacles around the vehicle. The conclusion has been reached through a detailed understanding of the design and particularities of the architecture.

Its adequacy can be further enhanced by introducing some improvements specifically targeted at the considered application, though the quantification of the effect of these modifications has been delayed to the next chapter. In any case, results on the KITTI dataset show adequate levels of accuracy.

Data from stereo cameras can be exploited to improve the accuracy of the detection with minimal changes in the architecture. The presented results suggest that the spatial information provided by that source of data can help in the generation of proposals by providing a rough segmentation of the scene.

Furthermore, experimental analyses conducted with two automotive datasets have demonstrated the high sensitivity of the architecture to the size of the training set. The availability of many and varied annotated frames can improve the performance of the algorithm dramatically without any change in its design. Further discussion will be held in the next chapter.

It is important to highlight the relevance of detection methods, such as the one presented here, nowadays. Although a significant part of the computer vision (and related ITS) research has tilted towards semantic segmentation algorithms, object-based models of the environment are still necessary for automated driving. Instance segmentation, a particular subtask of semantic segmentation, aims to separate the

different instances and might be the answer to these needs. However, some of the best performing instance segmentation algorithms rely on robust bounding-box detections. This is particularly the case of Mask R-CNN [118], which is an extension of Faster R-CNN that introduces a new classification head to perform the segmentation task on a per-instance basis. It is, therefore, worth it to continue making efforts to understand and improve the object detection task.

VIEWPOINT ESTIMATION

Once objects are localized within the boundaries of the image, knowledge about the different instances can be enriched with additional attributes that provide the high-level reasoning modules with more detailed insight into the traffic scene. In particular, information about the pose of the objects is of paramount importance for automated driving systems in order to decide the future trajectory to be followed by the vehicle.

As discussed in Sec. 2.6.2, the problem of pose estimation in the context of autonomous driving can often be reduced to the estimation of three magnitudes: on the one hand, the 2D coordinates of objects on the road plane and, in the other hand, their yaw angle, frequently referred to as *heading* angle.

In this chapter, an approach to embed orientation estimation into the Faster R-CNN detection framework is proposed. Orientation is obtained exclusively through appearance features and, therefore, does not rely on motion features that would require performing a robust tracking of objects in the scene. Instead, the method takes advantage of the fact that detection and orientation estimation are linked problems that can be tackled from the same perspective.

In this chapter, a comprehensive analysis of the full detection and viewpoint estimation pipeline is also provided, complementing the results introduced in the previous chapter and providing insightful details that should be taken into account in the design of the perception system.

Finally, a discussion on some possible enhancements to the method is provided, opening new avenues for research. Some preliminary results are presented, implying the future potential of the method beyond the scope of this dissertation.

5.1 PROBLEM FORMULATION

Although *orientation* is an intuitive concept, different definitions can be adopted, so it is important to start establishing the criteria used in this work. Assuming that road users move on a roughly planar surface, the orientation of an object is defined by its rotation around an axis which is normal to the road.

In the most general case, the sought-after magnitude is the value of the angle expressed with respect to a fixed reference frame (e. g., the camera frame), which will be referred to here as the *yaw* angle.

This chapter includes content from [111], [115], and [114].

According to this definition, every car moving in the same direction along a straight road has the same yaw angle.

However, as evident from Fig. 5.1, the representation in the image of objects with the same yaw angle can be significantly different. As we aim to employ appearance features to discriminate among different orientations, it is more convenient to define a new angular magnitude: the *viewpoint* or *observation angle*. Unlike the yaw angle, the viewpoint considers the vector from the origin of the camera coordinate system to the geometrical center of the object, representing the perspective from which the object is perceived by the sensor, as depicted in Fig. 5.2. In that way, instances with identical viewpoint values have the same parts visible in the image (e. g., the front right corner for the viewpoint highlighted in red in Fig. 5.2).



Figure 5.1: Illustration of the difference between yaw and viewpoint [188]
© 2017 IEEE

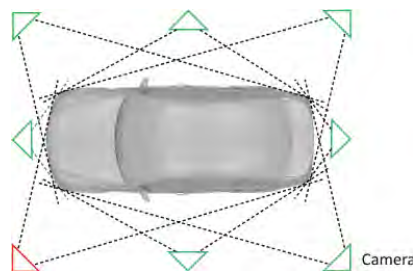


Figure 5.2: Representation of different viewpoints [164]

In this document, both angles are defined as shown in Fig. 5.3, where φ is the yaw angle, representing the rotation around the y-axis of the camera frame¹, and θ is the viewpoint, related to the yaw angle

¹ Note that the canonical camera frame defined in Sec. 3.1.1 is employed here.

and the object position on the plane, (x_0, z_0) , through the following equation:

$$\varphi = \theta + \text{atan2}(z_0, x_0) + 3\pi/2, \tag{5.1}$$

where the atan2 function is defined as follows:

$$\text{atan2}(y, x) = \begin{cases} \arctan(y/x), & \text{if } x > 0 \\ \arctan(y/x) + \pi, & \text{if } x < 0 \text{ and } y \geq 0 \\ \arctan(y/x) - \pi, & \text{if } x < 0 \text{ and } y < 0 \\ +\pi/2, & \text{if } x = 0 \text{ and } y > 0 \\ -\pi/2, & \text{if } x = 0 \text{ and } y < 0 \\ \text{undefined}, & \text{if } x = 0 \text{ and } y = 0 \end{cases} \tag{5.2}$$

Note that, for the car in Fig. 5.3, $\varphi = -90^\circ$, but $\theta \neq -90^\circ$, as the viewpoint varies according to the position of the object with respect to the camera.

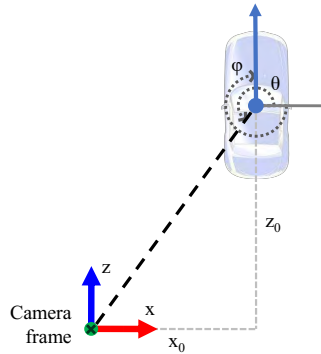


Figure 5.3: Definition of yaw (φ) and viewpoint angles (θ)

Subsequently, the viewpoint estimation problem can be framed in two different ways:

1. As a regression problem to a continuum of values spanning the full 360° circle, so that the viewpoint variable can adopt arbitrary values.
2. As a classification problem in which the circle is quantized into a predefined number of bins, and each real-valued viewpoint is assigned to a bin representing a range of values, as shown in Fig. 5.4. A formal definition of the discrete angle bins can be formulated as follows: if the full circle of possible viewpoints (2π radians) is divided into N_b bins, each bin Θ_l , $l = 0, \dots, N_b - 1$ encompasses a range of viewpoints given by:

$$\Theta_l = \left\{ \theta \in [0, 2\pi) \mid \frac{2\pi}{N_b} \cdot l \leq \theta + \theta_o < \frac{2\pi}{N_b} \cdot (l + 1) \right\}, \tag{5.3}$$

CONTINUOUS VS.
DISCRETE
VIEWPOINT
ESTIMATION

where θ_o is an offset that can be used to control the start and end points of the bins. Unless otherwise stated, $\theta_o = \pi/N_b$ is chosen in this work to make viewpoint bin centers correspond to well-defined orientations; i. e., front, rear, etc.

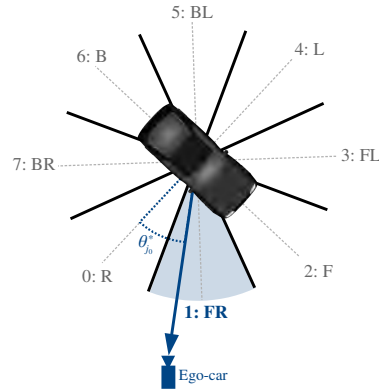


Figure 5.4: Example of viewpoint quantization with 8 bins ($N_b = 8$). The object’s viewpoint falls into the first bin ($\theta_{j_0}^* \in \Theta_1$). Viewpoint bins are named using combinations of the four main orientations: front (F), back (B), left (L), and right (R) [111] © 2018 IEEE

COMPARISON

It is simple to conclude that the regression approach can theoretically provide a finer-grained estimation than the discrete alternative. However, in practice, some aspects favor the use of the discrete approach in machine learning algorithms, such as CNNs. On the one hand, the regression into a full $[0, 2\pi)$ range does not fit well into inference frameworks, where outputs usually represent small variations from some reference values. On the other hand, data available for training is limited, which makes it difficult for the model to learn the diversity of values featured by the viewpoint variable, in contrast with the limited set of possible values in the discrete approach.

Furthermore, the discrete approach has often been proven as adequate for high-level scene understanding (e. g., [92]).

5.1.1 Orientation in the KITTI dataset

ANGLES

The KITTI object detection benchmark [90], whose main features were already introduced in Sec. 4.2.1, is outfitted with both yaw and viewpoint annotations, referred to as *rotation_y* and *alpha*, respectively. Both angles follow the criteria established in Fig. 5.3 and take values in the $[-\pi, \pi]$ range.

METRICS

Regarding the evaluation, the KITTI benchmark poses the viewpoint estimation problem within the frame of object detection and proposes a measure intended to assess both detection and viewpoint estimation

simultaneously. This measure is the *Average Orientation Similarity (AOS)*, which is defined analogously to the *AP* for a category k :

$$\text{AOS}_k = \frac{1}{11} \sum_{r \in \{0, 0.1, \dots, 1\}} s_{\text{interp}}(r) \quad (5.4)$$

A condensed metric that expresses the mean of the *AOS* values over all the categories can also be defined here as *mean Average Orientation Similarity (mAOS)*. As with the *AP*, $s_{\text{interp}}(r)$ results from the interpolation of a measure, in this case, the *orientation similarity*, at each recall level, $s(r)$, for all the recall levels above the current one:

$$s_{\text{interp}}(r) = \max_{\tilde{r}: \tilde{r} \geq r} s(\tilde{r}) \quad (5.5)$$

The orientation similarity is a unitary normalized variant of the cosine similarity, whose output is, in turn, limited to the $[-1, 1]$ range:

$$s(r) = \frac{1}{|\mathcal{D}(r)|} \sum_{i \in \mathcal{D}(r)} \frac{1 + \cos \Delta_{\theta}^{(i)}}{2} \delta_i, \quad (5.6)$$

where $\mathcal{D}(r)$ is the set of object detections at recall r and $\Delta_{\theta}^{(i)}$ is the angular error between the estimated orientation of detection i and its ground-truth value. On the other hand, δ_i ensures that only the detections corresponding to a ground-truth object are taken into account in the *AOS* computation, given that $\delta_i = 0$ unless the detection i has an overlap over the defined *IoU* threshold with any ground-truth box.

The orientation similarity is a smooth function of $\Delta_{\theta}^{(i)}$ on the domain $[-2\pi, 2\pi]$, which encompasses the range of possible values, and is able to deal naturally with the fact that π and $-\pi$ represent the same direction, as shown in Fig. 5.5.

ORIENTATION
SIMILARITY

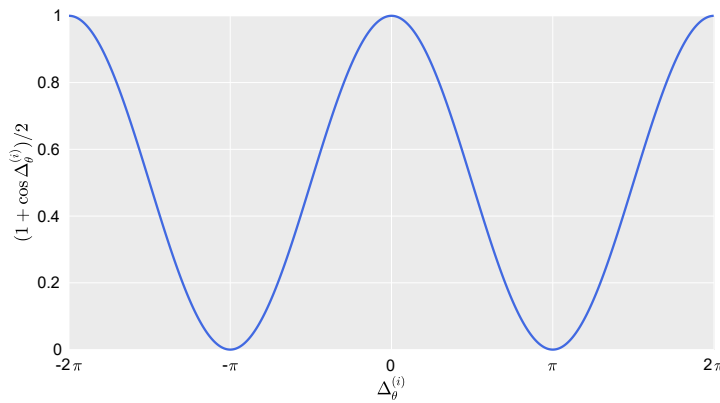


Figure 5.5: Relationship between orientation similarity and error in orientation estimation

As apparent from Eq. 5.6, when the orientation is perfectly estimated, the orientation similarity $s(r)$ becomes the precision, $p(r)$.

Therefore, the AOS value is upper-bounded by the AP and hence, assesses the detection and the orientation estimation performances jointly.

5.2 VIEWPOINT ESTIMATION WITHIN FASTER R-CNN

INTUITION

A method to embed the viewpoint estimation into the Faster R-CNN framework is proposed in this thesis. The approach is based on two hypotheses inspired by intuition:

1. The orientation of objects can be straightforwardly approximated from their appearance on a monocular image. The set of visible parts (e.g., the headlights of a car or the face of a pedestrian) and the relative positions among them can be used as cues.
2. Features employed for object detection and classification are also useful for viewpoint estimation.

APPROACH

In Faster R-CNN, the feature maps are concurrently used by the RPN and the classification heads (Fast R-CNN) to generate proposals, refine their bounding boxes, and provide a classification. According to the hypotheses stated above, it should be possible to additionally exploit these feature maps to provide a viewpoint estimation using an additional classification head. This idea is the foundation of the proposed approach, outlined in Fig. 5.6.

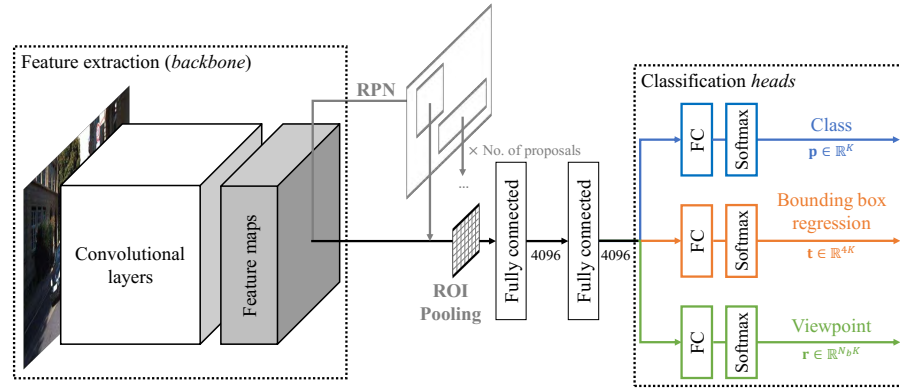


Figure 5.6: Proposed approach to incorporate viewpoint estimation capabilities into the Faster R-CNN framework (FRCNN+Or)

A discrete approach is adopted, so the viewpoint estimation problem is viewed as a classification task with N_b categories, which are indeed bins defined as in Eq. 5.3. The target output of the viewpoint estimation branch for an object classified into the class k is a categorical distribution over N_b possible outcomes described by $\mathbf{r}^k \in \Delta^{N_b-1}$, with Δ^N being the unit N -simplex:

$$\Delta^N = \left\{ \mathbf{x} \in \mathbb{R}^{N+1} \mid \sum_{i=1}^{N+1} x_i = 1 \wedge \forall i: x_i \geq 0 \right\} \quad (5.7)$$

The superscript k shows that, in this approach, the viewpoint prediction is class-aware, so the actual output of the viewpoint classification head is made of $K + 1$ different unit $(N_b - 1)$ -simplex sets, each describing the probability distribution for a category (plus the *background* class). Therefore, \mathbf{r}^k is the N_b -length prediction corresponding to the class k . The desired raw output \mathbf{r} of the viewpoint estimation branch is made of the concatenation of the different \mathbf{r}^k vectors. If the symbol \oplus is used to represent vector concatenation, then:

$$\mathbf{r} = \mathbf{r}^0 \oplus \mathbf{r}^1 \oplus \dots \oplus \mathbf{r}^K \in \mathbb{R}^{(K+1) \cdot N_b} \quad (5.8)$$

Hence, each element r_i^k of the probability distribution \mathbf{r}^k expresses the likelihood of the object's orientation being within the range spanned by the bin with index i (Eq. 5.3), as long as the object belongs to the class k .

The viewpoint estimation branch has the same basic structure as its siblings for classification and bounding box regression: the vector resulting from the stack of common FC layers is fed to a specific FC layer, and then, a softmax operation is applied. Naturally, the softmax is applied here over each \mathbf{r}^k sub-vector instead of using the full \mathbf{r} output; in practice, only the softmax of $\mathbf{r}^{\hat{k}}$, with \hat{k} being the predicted class, is computed.

As with the other classification heads, the viewpoint estimation branch provides a predicted value for each proposal from the RPN. However, its overall impact on the inference time is minor since all the classification heads use the same feature maps; besides, only $N_b \times (K + 1) \times 4096$ new parameters are added to the CNN.

5.2.1 Interpretation of the probability distribution

As stated above, the target output of the new viewpoint branch in the proposed CNN is a vector describing the probability of each viewpoint bin for each RPN proposal. This set of numbers has to be eventually mapped into a single-number value that can be assigned to the object model. This mapping means converting from the discrete space of angle bins to the continuum of values in $[-\pi, \pi]$ so that the viewpoint estimate becomes a decimal number representing the object's angle in rad.

From the definition of \mathbf{r}^k and Eq. 5.3, it follows that the predicted bin Θ_i for a proposal classified with class \hat{k} is given by the index of the maximum element in $\mathbf{r}^{\hat{k}}$; that is, $\hat{l} = \arg \max_i (r_i^{\hat{k}})$.

The most straightforward way to provide a prediction of the viewpoint magnitude ($\hat{\theta}$) is using the center of the predicted angle bin, $Z(\Theta_i)$:

$$\hat{\theta} = Z(\Theta_i) = \frac{\pi(2\hat{l} + 1)}{N_b} - \theta_o \quad (5.9)$$

CLASS-AWARE
ESTIMATION

VIEWPOINT
ESTIMATION
HEAD

SIMPLE
ESTIMATION

PSEUDO-
CONTINUOUS
ESTIMATION

As an alternative approach, an interpolation step can be performed to provide a finer-grained estimation. Following [287], the final prediction can be computed as the weighted average of two adjacent viewpoint bin centers, using their respective probabilities as weights. In particular, the bin with the highest probability and its most probable neighbor are used here. The viewpoint value is then obtained as:

$$\hat{\theta} = \frac{r_{\hat{l}}^k \cdot Z(\Theta_{\hat{l}}) + r_{\hat{l}_{\pm}}^k \cdot Z(\Theta_{\hat{l}_{\pm}})}{r_{\hat{l}}^k + r_{\hat{l}_{\pm}}^k}, \quad (5.10)$$

with \hat{l}_{\pm} being the index of the bin adjacent to \hat{l} with the highest probability:

$$\hat{l}_{\pm} = \arg \max_{l \in \{\hat{l}-1, \hat{l}+1\}} (r_l^k). \quad (5.11)$$

Of course, bins Θ_0 and Θ_{N_b-1} are assumed adjacent to this purpose. In this way, the one-hot representation given by r^k can be mapped to a continuous value in $[Z(\Theta_{\hat{l}}) - \pi/N_b, Z(\Theta_{\hat{l}}) + \pi/N_b)$, thus increasing the resolution of the prediction and mitigating the potential confusion in some extreme cases (e. g., viewpoints at the end of bin intervals).

5.2.2 Training

VIEWPOINT
OUTPUT

The output \mathbf{r} is included as an additional outcome of the Faster R-CNN framework and is handled the same way that the rest of the outputs listed in Sec. 4.1.2; namely, the RPN's objectness \mathbf{a} and box refinement \mathbf{b} , and the classification head's category estimation \mathbf{p} and bounding box refinement \mathbf{t} .

For each classified proposal, only one of the probability distributions contained in \mathbf{r} is relevant: the one corresponding to its class. At inference time, r^k is employed, as mentioned before. During training, however, the distribution corresponding to the ground-truth class, r^{v_i} , is used. In either case, the list of outputs provided in Sec. 4.1.2 is extended to include a fifth element:

$$\mathbf{r}^k = (r_0^k, \dots, r_{N_b-1}^k), \quad k = 0, \dots, K \quad (5.12)$$

VIEWPOINT LOSS

The same training scheme as in Chapter 4 is adopted, and therefore, the loss function is the one in Eq. 4.14 with an extra summand taking into account the viewpoint estimation. As this prediction is indeed a classification into the N_b possible angle bins, the multinomial logistic loss is used²:

$$L_5 = \frac{1}{N_{B_2}} \sum_{i \in B_2} [v_i \geq 1] L_{\text{cls}}(\mathbf{r}_i^{v_i}, \mathbf{w}_i), \quad (5.13)$$

² Please note that, in this equation, the subscript i of $r_i^{v_i}$ refers to the index of the batch element instead of the index of the element within the vector.

where w_i is the index of the ground-truth bin³ for the detection i . Background samples ($v_i = 0$) do not contribute to the loss. Besides, the loss is normalized over the set of proposals from the RPN (B_2).

Finally, training is also performed through the approximate joint training strategy, so the resulting multi-task loss to be optimized by the SGD is:

FINAL
MULTI-TASK LOSS

$$L = \sum_{i=1}^5 L_i, \quad (5.14)$$

with the definitions of L_1 , L_2 , L_3 , and L_4 given in Sec. 4.1.2, and L_5 as described in Eq. 5.13.

5.2.3 Experimental results

In order to assess the validity of the approach, quantitative experiments have been carried out on the KITTI dataset. The aim is to demonstrate the validity of the general approach introduced in this chapter, as well as the technical soundness of the design choices presented in Sec. 4.2 within the framework of the proposed joint detection and viewpoint estimation method.

5.2.3.1 Criteria

Quantitative results are expressed in terms of the already introduced AP and AOS measures (Sec. 4.2.1 and Sec. 5.1.1), but other ways of measuring the performance are used as well:

METRICS

- Precision-recall and orientation similarity-recall curves, representing the interpolated precision and orientation similarity, respectively, for each recall value, $p_{\text{interp}}(r)$, as in Eq. 4.16. AP and AOS are indeed aimed to condense the information provided by these curves.
- Evolution of the recall with the number of proposals. Employed, among others, by the COCO dataset [169], this measure is especially interesting to assess the effectiveness of region proposal methods. It can be averaged similarly to AP and AOS to provide the *Average Recall (AR)*.
- Mean Precision in Pose Estimation (MPPE). Introduced by [173], the MPPE is defined as the mean of the elements on the main diagonal of the confusion matrix when viewpoint estimation is viewed as a bin classification problem. While MPPE is usually computed for a fixed detection threshold, it can also be obtained for every recall value, as with precision and orientation similarity.

³ As with the other losses, detections are assigned ground-truth annotations according to the IoU overlap between them.

Criteria established by the KITTI dataset and described in Sec. 4.2.1, including threshold IoUs and levels of difficulty, are adopted here.

DETECTION
SCORE

It must be noted that, for almost every metric used, detected instances are assumed to be ranked according to a confidence score assigned by the detection algorithm. In this work, the score is naively obtained as the class probability provided by the category vector; that is, the \hat{k} th element of \mathbf{p} , $p_{\hat{k}}$. This score is also used in the final NMS procedure that is performed over the set of output detections. Therefore, the confidence value of the viewpoint estimation, provided by $r_{\hat{l}}^{\hat{k}}$, is not considered at any time. This design decision is intended to favor the performance of detection over viewpoint estimation in case of disagreement.

TRAINING SETUP

The set of settings shown in Table 5.1 is employed, unless otherwise stated. Likewise, the number of viewpoint bins is set to $N_b = 8$ for most of the experiments.

Feature extractor:	VGG-16
Pre-training:	Yes, on ImageNet
Anchors:	Custom (Table 4.5)
Classification loss:	Infogain
RPN proposals:	300
Classification heads:	Category + b. box regression + viewpoint estimation
Predicted classes:	3 (CAR, PED, CYC)
Train/val split:	Custom (70:30)
Training schedule:	50k iterations @ $\text{lr} = 10^{-3}$ + 50k iterations @ $\text{lr} = 10^{-4}$ + 50k iterations @ $\text{lr} = 10^{-5}$

Table 5.1: Set 1 of training hyperparameters for the proposed approach

When comparing alternatives, the analyzed models are trained and tested *ceteris paribus*, i. e., “all other things being equal,” to isolate the effect of the variables under study.

5.2.3.2 Parameter tuning and onboard-oriented adjustments

In this section, a justification of two of the solutions adopted to fit Faster R-CNN to traffic environments (Sec. 4.2.2) is provided. In particular, the use of custom RPN anchors (AN) and the *infogain* loss for class balancing (IL) are evaluated. Table 5.2 shows the effect of the changes introduced in terms of AP and AOS statistics.

As apparent from the data, the impact of custom anchors on the performance is limited, although results suggest that detection of *Hard* samples benefits from it (+0.87 points in mAP). Hence, the modification is suitable given that the use of custom anchors does not affect the inference time as long as the number of anchors per location is maintained.

CL.	AN	IL	2D DETECTION (AP 2D)			2D DET. AND OR. (AOS)		
			EASY	MOD.	HARD	EASY	MOD.	HARD
CAR	✓	✗	89.24	77.95	60.65	87.64	76.37	59.11
	✗	✓	89.87	79.07	66.96	88.41	77.62	65.37
	✓	✓	89.18	78.67	61.21	87.60	77.11	59.70
PED	✓	✗	80.73	68.78	62.96	71.95	61.13	56.00
	✗	✓	79.55	68.19	61.34	72.01	61.14	54.94
	✓	✓	80.30	69.03	62.98	74.52	63.35	57.37
CYC	✓	✗	64.38	49.39	47.15	54.48	41.82	40.03
	✗	✓	72.77	51.07	48.50	58.01	39.74	37.69
	✓	✓	68.98	51.50	49.99	53.86	41.58	40.55

Table 5.2: Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset with and without onboard-oriented tuning: custom anchors (AN) and infogain loss (IL) [111] © 2018 IEEE

Regarding the use of the infogain loss for class balancing, results show a notable improvement that reaches +1.03 points in mAP for *Moderate* samples. Infogain weights were assigned to favor the *Pedestrian* and *Cyclist* classes at the expense of the *Car* category, whose contribution to the final loss is effectively reduced; nevertheless, it is noteworthy that performance on *Car* detection is also ameliorated for *Moderate* and *Hard* difficulties.

5.2.3.3 Alternatives for viewpoint estimation

This section evaluates the interpolation method (IN) introduced in Sec. 5.2.1, along with the use of a convenient offset angle, $\theta_o = \pi/N_b$ in Eq. 5.3 (OF). In particular, as $N_b = 8$, $\theta_o = \pi/8$, and the centers of the bins coincide with the cardinal and intercardinal directions. Table 5.3 shows the AP and AOS values when adopting these measures. Please note that the interpolation method is a test-time modification that affects only the viewpoint estimation, whereas modifying the offset θ_o requires training a new model; that is why AP gets affected in this latter case.

The interpolation approach shows a consistent improvement of around +0.20 points in mAP across all levels of difficulty. On the other hand, a carefully selected offset implies enhancing not only the viewpoint estimation performance (more than +1 mAOS point for *Moderate* samples) but also the detection performance (+0.65 mAP points for *Moderate* samples). The proper separation among different viewpoint bins introduced by the offset probably benefits the whole model, including the detection branch.

CL.	OF	IN	2D DETECTION (AP 2D)			2D DET. AND OR. (AOS)		
			EASY	MOD.	HARD	EASY	MOD.	HARD
CAR	X	X	88.50	77.72	60.41	87.90	76.90	59.33
	X	✓				87.93	76.94	59.37
	✓	X	89.18	78.67	61.21	87.53	77.03	59.64
	✓	✓				87.60	77.11	59.70
PED	X	X	80.89	69.37	63.98	75.17	62.89	57.96
	X	✓				75.51	63.30	58.33
	✓	X	80.37	69.03	62.98	74.07	63.00	56.97
	✓	✓				74.52	63.35	57.37
CYC	X	X	63.73	50.16	49.68	50.93	38.38	38.01
	X	✓				51.18	38.60	38.25
	✓	X	68.98	51.50	49.99	53.65	41.41	40.42
	✓	✓				53.86	41.58	40.55

Table 5.3: Performance (AP % and AOS %) on the KITTI validation subset of the proposed approach for different alternatives for viewpoint estimation refinement: offset $\theta_o = \pi/N_b$ (OF) and interpolation (IN) [111] © 2018 IEEE

5.2.3.4 Assessment and benchmarking of the general approach

COMPARISON
WITH BASELINE

In this section, evidence of the validity and effectiveness of the proposed approach is provided. First of all, the method is compared with the baseline Faster R-CNN, with detection-only capabilities. The comparison is shown in Fig. 5.7 in terms of the recall-IoU curve, intended for assessing the effectiveness of the proposals, and the precision-recall curve, which gives an overall estimate of the performance of the detection method.

As evident from the figure, the introduction of viewpoint estimation capabilities has a minor impact on the quality of the proposals, whereas the overall detection performance of both variants is comparable. These results validate the initial hypothesis that the same set of features can be used for detection and viewpoint orientation simultaneously.

VALIDATION OF
THE DISCRETE
APPROACH

For the assessment of the effectiveness of the discrete viewpoint classification approach adopted in this thesis, the proposed method is compared now with two alternative approaches that pose the problem as a direct regression of the observation angle. Two different loss functions have been used:

- Smooth-L1 loss. In this variant, the Smooth-L1 loss (Eq. 4.7) is employed to learn the regression parameters, similarly to other works in the literature [44], [205]. The output of the viewpoint estimation head is, in this case, a $(K + 1)$ -length vector s' where

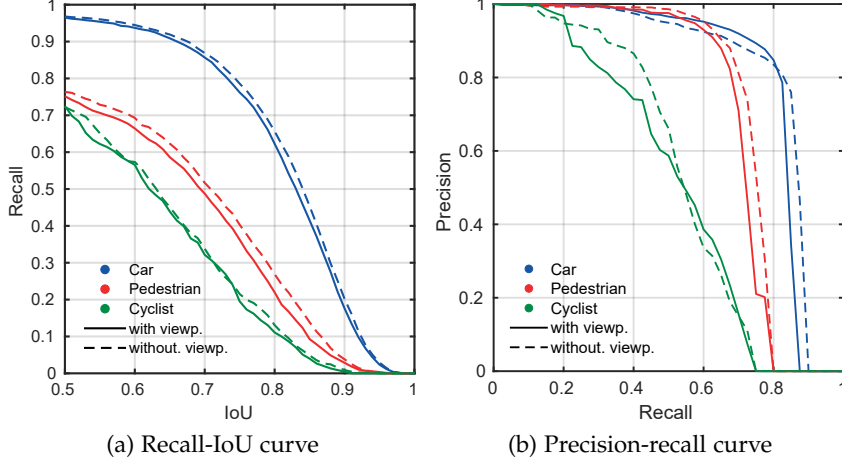


Figure 5.7: Precision-recall and Recall-IoU curves of the proposed approach on the KITTI *Moderate* validation subset with and without viewpoint estimation [111] © 2018 IEEE

element s'_k provides a direct estimation of the viewpoint for the class k . The final viewpoint value is the one corresponding to the predicted class, s'_k . Therefore, for this alternative, L_5 in Eq. 5.14 becomes:

$$L_5 = \frac{1}{N_{B_2}} \sum_{i \in B_2} [v_i \geq 1] \text{smooth}_{L_1}(s'_{i,v_i} - s_i^*, 1), \quad (5.15)$$

with the definition of the smooth-L1 function provided in Eq. 4.7 and s_i^* being the ground-truth viewpoint angle, in radians.

- **Cosine similarity loss.** In an attempt to optimize the AOS results of the regression approach, a loss inversely proportional to the *orientation similarity*, defined in Eq. 5.6, is used in the SGD. This prediction is class-agnostic, so the output is indeed a real value s'' that can be straightforwardly interpreted as the predicted viewpoint. Then, the loss contribution of the viewpoint estimation is:

$$L_5 = \frac{1}{N_{B_2}} \sum_{i \in B_2} L_{\cos}(s''_i, s_i^*), \quad (5.16)$$

where s_i^* is used to denote the ground-truth observation angle for the sample i , and L_{\cos} for element n is defined as:

$$L_{\cos}(\hat{s}_n, s_n^*) = 1 - \frac{1 + \cos(\hat{s}_n - s_n^*)}{2}, \quad (5.17)$$

which is a smooth function which ranges from 0, when the predicted angle coincides with the ground-truth, to 1, when both angles are opposite each other.

In both cases, the only modifications performed on the proposed method are the output provided by the viewpoint head, which is no longer a probability distribution over viewpoint bins but instead a direct estimation of the angle value, and the corresponding component of the multi-task loss. The results of the two tested regression approaches in comparison with the proposed approach are provided in Fig. 5.8.

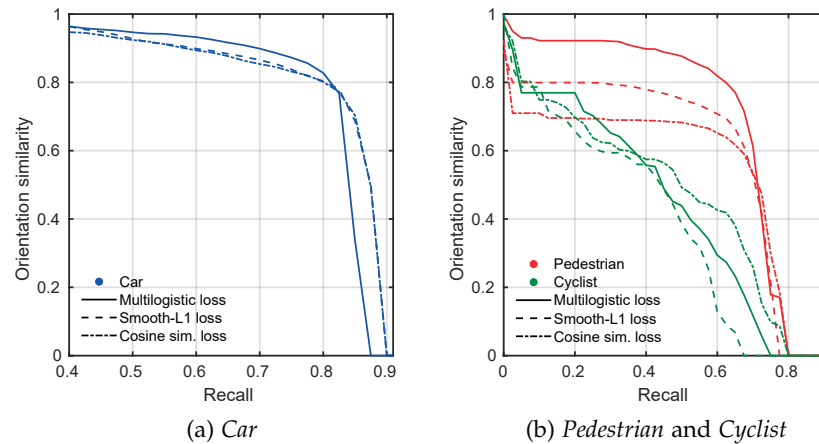


Figure 5.8: Orientation similarity vs. recall of the proposed approach on the KITTI *Moderate* validation subset for three alternative viewpoint estimation approaches. The proposed method uses the *multilogistic* (multinomial logistic) loss [111] © 2018 IEEE

As observed, the proposed classification approach, whose resolution is limited by the number of bins, outperforms the two alternative regression methods. The difference is especially relevant in the *Pedestrian* class, although the overall *mAOS* score of the discrete approach is +3.86 points higher than the *mAOS* of the orientation-similarity-based regression and +4.23 points higher than the *mAOS* of the smooth-L1-based regression in the *Moderate* validation subset. These results show that the proposed discrete approach is sound and fits well with the Faster *R-CNN* detection framework.

COMPARISON WITH OTHER METHODS

A comparison of the detection and viewpoint orientation results with other methods is provided in Table 5.4. Values, including the ones for the proposed method, are obtained from the official KITTI object detection benchmark⁴ and are therefore referred to the KITTI testing set, whose annotations are not publicly available. Results are available on the KITTI website as FRCNN+Or⁵.

All the included methods, except for DPM-VOC, are modern *DNN*-based methods able to jointly estimate detection and observation angle for all the categories in the KITTI set:

⁴ http://www.cvlibs.net/datasets/kitti/eval_object.php

⁵ http://www.cvlibs.net/datasets/kitti/eval_object_detail.php?&result=28dc29642148933e41c6daccff01d35b1bdf5ecf

- Multi-view deformable part model trained with structured bounding box and viewpoint loss (DPM-VOC) [203]
- Monocular 3D object detection for autonomous driving (Mono3D) [43]
- 3D bounding box estimation using deep learning and geometry (Deep3DBox) [188]
- Subcategory-aware convolutional neural networks for object proposals and detection (SubCNN) [281]

CL.	METHOD	2D DETECTION (AP 2D)			2D DET. AND OR. (AOS)			TIME (s)
		EASY	MOD.	HARD	EASY	MOD.	HARD	
CAR	DPM-VOC	80.45	66.25	49.86	77.51	63.27	47.57	8 ^a
	Mono3D	90.27	87.86	78.09	89.00	85.83	76.00	4.2 ^b
	Deep3DBox	90.47	88.86	77.60	90.39	88.56	77.17	1.5 ^b
	SubCNN	90.75	88.86	79.24	90.61	88.43	78.63	2 ^c
	Proposed	89.87	78.95	68.97	88.52	77.61	67.69	0.09 ^d
PED	DPM-VOC	59.60	44.86	40.37	53.66	39.83	35.73	8 ^a
	Mono3D	77.30	66.66	63.44	68.58	58.12	54.94	4.2 ^b
	Deep3DBox	—	—	—	—	—	—	1.5 ^b
	SubCNN	83.17	71.34	66.36	78.33	66.28	61.37	2 ^c
	Proposed	71.18	56.78	52.86	66.84	52.62	48.72	0.09 ^d
CYC	DPM-VOC	43.65	31.16	28.29	31.24	23.22	21.62	8 ^a
	Mono3D	75.22	63.85	58.96	65.74	53.11	48.87	4.2 ^b
	Deep3DBox	82.65	73.48	64.11	68.58	59.37	51.97	1.5 ^b
	SubCNN	77.82	70.77	62.71	71.39	63.41	56.34	2 ^c
	Proposed	68.81	55.80	50.52	63.41	50.91	45.46	0.09 ^d

^a CPU; ^b GPU 2.5 GHz; ^c GPU 3.5 GHz; ^d Titan Xp GPU (1.6 GHz)

Table 5.4: Comparison of the performance (AP % and AOS %) of the proposed approach with other methods on the KITTI testing set [111] © 2018 IEEE

The running time for a single frame is provided in the last column of Table 5.4. These processing times are dependent on the particular implementations and hardware used by the respective authors, although it can be assumed that modern GPUs were used for CNN-based approaches. The proposed approach takes around 90 ms to perform a forward pass, a time significantly lower than the one featured by the other methods.

Therefore, it is safe to conclude that the proposed approach obtains comparable results to state-of-the-art methods while being significantly faster than them. Some qualitative results on frames from the KITTI testing set are shown in Fig. 5.9, where the predicted viewpoint is represented as an arrow.

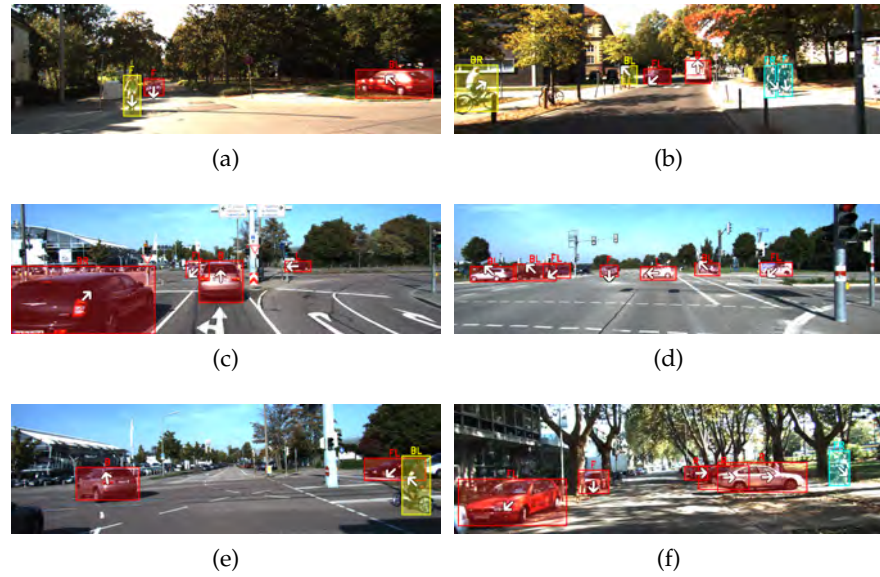


Figure 5.9: Examples of joint detection and viewpoint estimation on scenes from the KITTI testing set using the proposed approach. Class is encoded using color: *Car*, *Cyclist*, and *Pedestrian* classes are represented as red, yellow, and blue, respectively. The interpolated viewpoint estimation is depicted as an oriented white arrow at the center of the obstacle, while the predicted bin is provided above each bounding box as a combination of *front* (F), *back* (B), *left* (L) and *right* (R) [111] © 2018 IEEE

It is clear from the results that the proposed method provides a methodology to perform joint detection and viewpoint estimation that is focused on efficiency and, therefore, is especially suitable for onboard applications. The foundation principle of the approach, based on taking advantage of the same set of convolutional features to perform both tasks, allows for reduced inference times. Furthermore, most design choices, such as the input scale of the image, were made prioritizing framerate over accuracy. Consequently, it is still possible to modify the accuracy/performance trade-off through different hyperparameters that will be profusely studied in the next section.

5.3 IDENTIFICATION OF FACTORS AFFECTING THE PERFORMANCE

The method introduced in this chapter for simultaneous object detection and viewpoint orientation leaves room for several setting adjustments that allow tailoring the approach to the required performance. The analysis introduced in the previous section was performed for a point of operation that was deemed adequate for onboard applications. However, it is also interesting to further analyze the effect of the different tunable parameters on the performance and the inference time of the method, so that different trade-offs can be reached without sub-

stantially modifying the design. A set of experimental results aimed to quantify the influence of different hyperparameters is provided in this section.

As before, the analysis is based on the KITTI dataset and the criteria introduced in Sec. 5.2.3.1 still holds. However, two new sets of training parameters are used in this section in addition to the Set 1 presented in Table 5.1. These two sets are shown in Tables 5.5 and 5.6⁶.

Feature extractor:	VGG-16 [244] (by default), ZF [291], or MobileNet [130]
Pre-training:	Yes, on ImageNet
Anchors:	Custom (Table 4.5)
Classification loss:	Infogain
RPN proposals:	300 (unless otherwise stated)
Classification heads:	Category + b. box regression + viewpoint estimation
Predicted classes:	7 (CAR, PED, CYC, VAN, TRK, SIT, TRM)
Train/val split:	Custom (70:30)
Training schedule:	VGG-16: 50k it. @ lr = 10^{-4} + 50k it. @ lr = 10^{-5} + 50k it. @ lr = 10^{-6} ZF: 50k it. @ lr = $5 \cdot 10^{-4}$ + 30k it. @ lr = $5 \cdot 10^{-5}$ MobileNet: 50k it. @ lr = 10^{-3} + 50k it. @ lr = 10^{-4} + 50k it. @ lr = 10^{-5}

Table 5.5: Set 2 of training hyperparameters for the proposed approach

Feature extractor:	VGG-16 [244]
Pre-training:	Yes, on ImageNet (unless otherwise stated)
Anchors:	Default
Classification loss:	Infogain
RPN proposals:	300
Classification heads:	Category + b. box regression + viewpoint estimation
Predicted classes:	3 (CAR, PED, CYC)
Train/val split:	Chen <i>et al.</i> [46] (KITTI) and Cityscapes training set
Training schedule:	50k iterations @ lr = 10^{-3} + 30k iterations @ lr = 10^{-4} (unless otherwise stated)

Table 5.6: Set 3 of training hyperparameters for the proposed approach

⁶ It can be assumed that every experiment in this chapter resulted from a separate training procedure. Therefore, small differences in results due to the randomness involved in training can occur between experiments even if all the hyperparameters coincide.

5.3.1 Training hyperparameters

TRAINING
ITERATIONS

Firstly, the performance of the algorithm at different stages of the training procedure is analyzed for signs of either bias or variance error. Fig. 5.10 shows the evolution of the detection and viewpoint estimation accuracy from 0 to 150 000 iterations. With the KITTI train/validation split in use, the upper limit of the range corresponds to around 40 epochs.

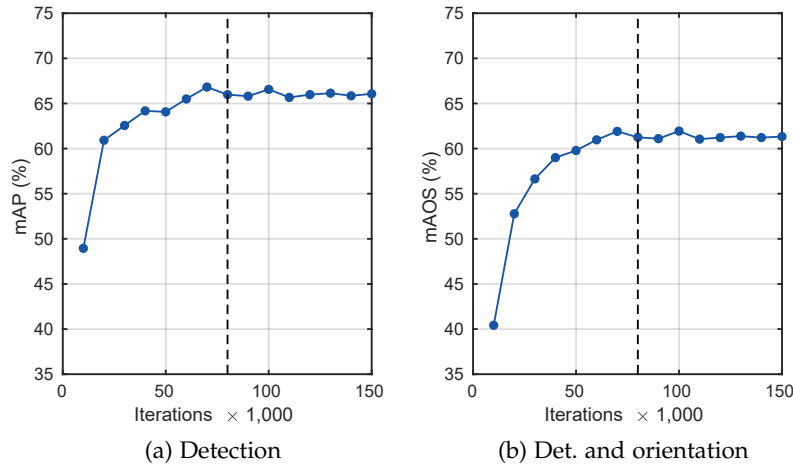


Figure 5.10: Detection and orientation estimation performance (mAP % and mAOS %) of the proposed approach vs. number of training iterations on the KITTI validation subset (hyperparameters: set 3) [115] © 2018 IEEE

Both curves show that the training procedure is most effective during the first 80k iterations, and plateaus after that. Although the training set is relatively small, no signs of overfitting are observed.

DROPOUT

Despite this, the use of dropout was also tested. Dropout [248] is a technique that randomly ignores some neurons during training to improve the generalization capabilities of a DNN. The probability of a neuron to be dropped is given by the dropout ratio, p .

A comparison of the obtained mAP and mAOS stats for models trained with and without dropout, other things being equal, is shown in Table 5.7. The dropout ratio is fixed to $p = 0.5$, and it is set up to affect the two fully connected (FC) layers that are shared by all the classification heads, as in the original Faster R-CNN proposal.

It is apparent from the table that the effect of dropout in the proposed method is negligible or even negative, which is particularly noticeable in the viewpoint estimation. This result validates the training procedure without dropout proposed in Sec. 4.2.2.5.

CL.	DR	2D DETECTION (MAP 2D)			2D DET. AND OR. (MAOS)		
		EASY	MOD.	HARD	EASY	MOD.	HARD
mean	✗	79.51	65.98	59.44	74.43	61.25	55.02
	✓	79.20	65.34	58.43	73.77	60.73	54.16

Table 5.7: Detection and viewpoint estimation performance (mAP % and mAOS %) of the proposed approach on the KITTI validation subset with and without dropout (hyperparameters: set 3) [115] © 2018 IEEE

5.3.2 Number of proposals

The number of proposals generated by the RPN and subsequently classified by the respective heads is a hyperparameter of vital importance on the inference time given that, according to the R-CNN paradigm, every proposal has to be individually processed at the classification stage.

Some studies suggest that, contrary to what was expected, the number of proposals from the RPN can be significantly reduced from the proposed initially (300) without resulting in a significant loss of recall [131]. For that reason, some tests were performed to assess the effect of this hyperparameter on the proposed detection and viewpoint estimation algorithm.

Firstly, the quality of the resulting set of proposals for different values of this parameter is studied. The AR metric, proposed in [126], is computed for a range between 10 and 1000 proposals. AR is computed by averaging the total recall across different IoU requirements, usually between 0.5 and 1, and gives information about the proposal performance. Results for each of the three main KITTI classes are provided in Fig. 5.11a. It can be observed that the AR does not drop severely until less than 50 proposals are considered

This performance translates to the overall mAP/mAOS results, as shown in Fig. 5.11b. Thus, the mAP is reduced by only 1.33 points when 50 proposals are used. Meanwhile, the effect on the inference time is presented in the same figure. From the baseline of 300 proposals, runtime per frame gets reduced by 9% with 100 proposals, and by 13.6% with 50 proposals.

From these results, it follows that the performance of the method using 100 or even 50 proposals is satisfactory enough to provide a faster alternative to the baseline when inference time is critical.

The degradation in recall caused by the reduction of the number of proposals affects almost exclusively to distant objects, which are less visible in the images. Taking advantage of the 3D annotations available in the KITTI dataset, Fig. 5.12 provides results of the maximum recall achieved if the evaluation is limited at a certain distance from the

OVERALL EFFECT

EFFECT ON
DISTANT OBJECTS

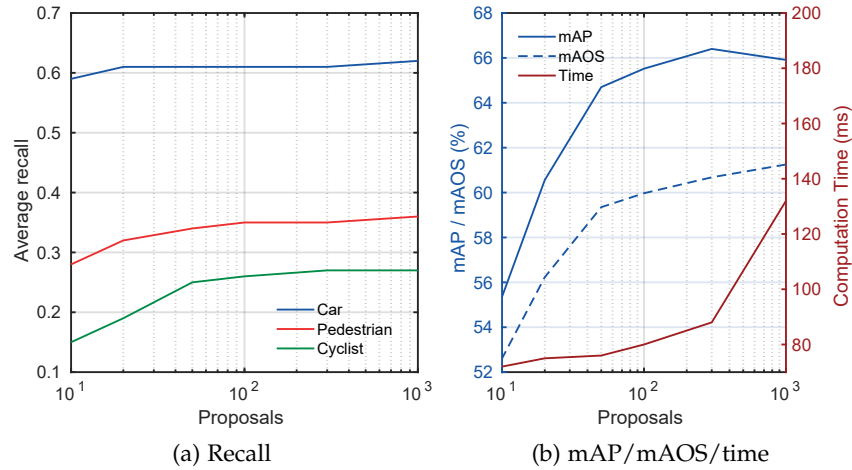


Figure 5.11: Performance (Average Recall, mAP % and mAOS %) and run time (s) of the proposed approach vs. the number of proposals on the KITTI *Moderate* validation subset (hyperparameters: set 1) [111] © 2018 IEEE

camera. Curves are provided for 50, 100, and 300 RPN proposals. The *Hard* subset is used here to take into account all the annotations.

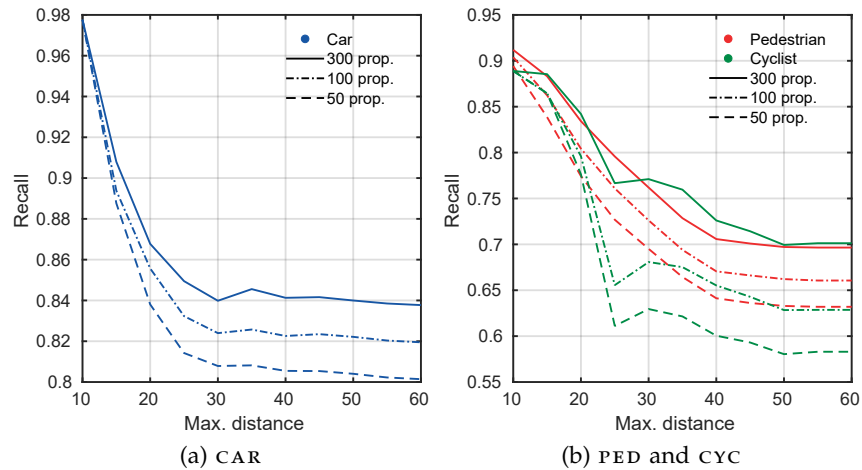


Figure 5.12: Recall of the proposed approach vs. max. distance from the ego-car on the KITTI *Hard* validation subset for different numbers of proposals (hyperparameters: set 1) [111] © 2018 IEEE

As apparent from the figure, samples farther than 20 m get affected the most by the lower number of proposals. The effect is particularly significant in the *Cyclist* category, and almost negligible for *Car*⁷.

Fig. 5.12 also provides further insight into the baseline approach with 300 proposals: according to the results, at least 75% of objects from each category placed within the range 0 m to 30 m ahead of the vehicle are successfully identified.

⁷ Please note the different scale in the y-axis of Fig. 5.12a.

5.3.3 Viewpoint bins resolution

The most critical hyperparameter in the new viewpoint estimation branch proposed in this thesis is the number of viewpoint bins, N_b , whose value has been fixed to $N_b = 8$ up to this section. The discrete approach adopted to embed the viewpoint inference task into the Faster R-CNN framework naturally limits the maximum resolution to $2\pi/N_b$. It might seem logical to increase the N_b value as much as possible to improve the resolution of the algorithm; however, this comes with two major drawbacks that must be taken into account:

1. As the size of the training set remains unchanged, increasing N_b implies reducing the number of samples available for training each of the possibilities.
2. For larger N_b values, differences in appearance between different bins get smaller. As a multinomial logistic loss is used for bin classification, confusions between a bin and its neighbor, which are more likely to occur as N_b increases, can lead to overly high losses and hurt the training procedure.

In this section, the influence of N_b on the performance of the proposed method is investigated. Different tests are performed for $N_b \in \{4, 8, 16, 32\}$. Orientation similarity-recall and MPPE-recall curves are shown in Fig. 5.13 for the four alternatives using the KITTI *Moderate* samples.

From Fig. 5.13a, it follows that the performance in terms of AOS does not get overly affected by changes in N_b . Probably, this is due to the definition of AOS, which uses the smooth cosine similarity function, as shown before in Fig. 5.5. Still, the $N_b = 8$ curve is slightly above the other alternatives, which proves that it is a valid trade-off choice for the viewpoint estimation problem.

On the other hand, the MPPE curves (Figs. 5.13b-5.13d) offer insight into the accuracy of the bin classification problem depending on the number of divisions. As apparent from these results, as N_b increases, misclassifications are more frequent, thus counterbalancing the improvement in resolution.

Table 5.8 shows the numerical results in terms of AP and AOS for the considered alternatives. Overall, $N_b = 8$ outperforms its closest alternatives (4 and 16) by around 1 mAOS point for the *Hard* difficulty level, thus confirming its adequacy. However, $N_b = 16$ also offers a good trade-off, especially for *Car*. This is likely due to the larger size, in pixels, of the samples belonging to this class compared to the other two, which enables a finer-grained classification. *Easy* samples, which are also generally bigger, also get more significant improvements when increasing N_b . Despite all, for $N_b = 32$, the degradation of the classification performance is too substantial, which leads to lower

RESOLUTION VS.
CLASSIFICATION
ACCURACY

RESULTS

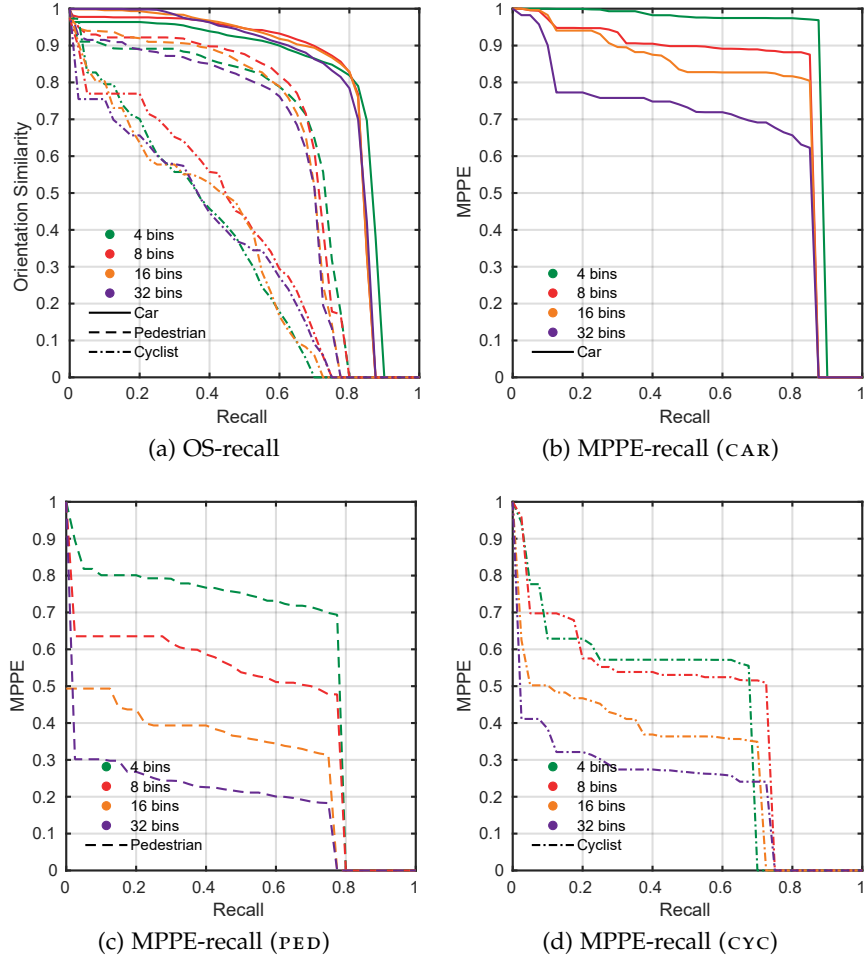


Figure 5.13: Orientation similarity (OS)-recall and $MPPE$ -recall curves of the proposed approach on the KITTI *Moderate* validation subset for different values of N_b (hyperparameters: set 1) [111] © 2018 IEEE

detection performance (-1.11 mAP for *Moderate*) and, therefore, lower $mAOS$ (around -2.5 points for all difficulty levels).

In the next section, the influence of N_b will be further analyzed in combination with other parameters, such as the backbone architecture and the input scale.

5.3.4 Feature extractor architecture, input scale, and combinations

FEATURE EXTRACTOR

Faster R -CNN, including the modified version proposed in this thesis, is a meta-architecture that admits many alternatives for the feature selection part. As stated in the previous chapter, VGG-16 [244] is used in this work as the baseline backbone for its compelling performance. However, virtually any of the multiple existing CNN architectures could be adapted to fulfill this role. Significant differences in perfor-

CL.	N_b	2D DETECTION (AP 2D)			2D DET. AND OR. (AOS)		
		EASY	MOD.	HARD	EASY	MOD.	HARD
CAR	4	88.86	78.32	65.15	85.97	75.39	61.99
	8	89.18	78.67	61.21	87.60	77.11	59.70
	16	88.77	78.00	62.72	88.42	77.53	61.87
	32	88.13	76.98	59.90	88.00	76.69	59.39
PED	4	82.00	69.30	63.92	74.16	61.96	56.89
	8	80.37	69.03	62.98	74.52	63.35	57.37
	16	80.02	67.96	61.54	73.56	61.75	55.80
	32	78.79	67.63	60.94	70.69	60.14	54.13
CYC	4	69.19	48.32	47.36	52.99	36.37	35.49
	8	68.98	51.50	49.99	53.86	41.58	40.55
	16	70.79	51.20	49.49	56.28	38.16	36.77
	32	65.38	51.24	49.59	48.61	37.81	36.48

Table 5.8: Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset for different numbers of viewpoint bins (N_b) (hyperparameters: set 1)

mance are expected between feature extractors depending on their representation capabilities.

In this section, two alternatives to the VGG-16 architecture are tested. As the focus is on online applications, they are lightweight models able to achieve even higher framerates than VGG-16 during inference: Zeiler and Fergus (ZF) [291], and MobileNet [130]. The reader is referred to Sec. 2.4.3 for a detailed description of both models; as a brief reminder, the former is a well-studied proposal with a limited number of parameters, whereas the latter is a modern architecture focused on efficiency and ready to be deployed in embedded devices. Table 5.9 shows a short comparison of parameters for the three architectures under study.

BACKBONE	CONV. LAYERS	FEATS. USED BY RPN	ROI POOLING
ZF	5	256	6×6
VGG-16	13	512	7×7
MobileNet	18	512	7×7

Table 5.9: Faster R-CNN hyperparameters for each of the studied feature extractors

Apart from the backbone, the input scale is often shown as one of the most influential hyperparameters on the performance of Faster R-CNN [72]. Here, *scale* refers to the size of the shortest side of the image in pixels. Frames from the KITTI object detection benchmark

have a resolution of around 1242×375 pixels⁸. As stated in Sec. 4.2.2, all experiments so far were performed with a scale of 500 pixels, which corresponds to a $1.33\times$ scaling factor. It is expected that larger scales would improve the detection accuracy, especially for small objects, but that implies higher processing times and a higher GPU memory footprint. In this section, three scales are considered: 375 (approx. $1\times$), 500 (approx. $1.33\times$) and 650 (approx. $1.77\times$). Note that every model is trained with the same scale that will be used for inference to maximize its performance.

RESULTS

Fig. 5.14 is a scatterplot where different alternatives regarding the number of proposals, feature extractor, and input scale, are located according to their performance (y-axis) and run time (x-axis). Times are obtained as the median value of inference times for every frame in the validation subset; on the other hand, performance is given as the mAP and mAOS values for *Moderate* samples.

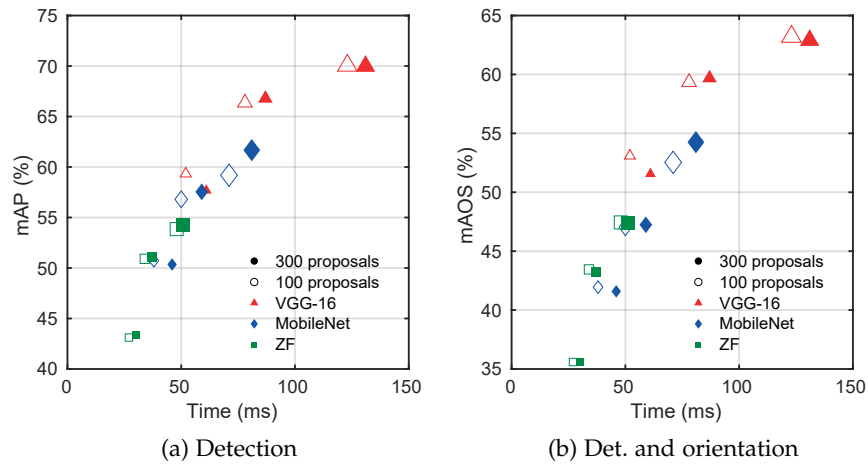


Figure 5.14: Detection and viewpoint estimation performance (mAP % and mAOS %) of the proposed approach vs. run time on the KITTI *Moderate* validation subset for different numbers of proposals, architectures and input scales (375, 500 and 650, indicated through the marker size) (hyperparameters: set 2) [114]

As already proved (Sec. 5.3.2), accuracy is independent, to a large extent, to the number of proposals, especially when the number is above 100. Not surprisingly, the performance of each backbone architecture is strongly linked to its complexity and, therefore, its run time for a forward pass. Hence, VGG-16 is the slowest but provides the best results, whereas ZF is fast but has trouble at modeling the underlying structure of data. Similarly, larger scales produce better results, requiring, in turn, more computational resources.

⁸ There are small differences in size among the images in the object detection benchmark since they were cropped by the dataset authors to avoid pincushion distortion effect [89]

Fig. 5.15 provides a view of the relationship between performance (mAP and mAOS) and run time when the input scale is changed. The same three scale values from the previous experiment are tested. In this case, the two better alternatives for N_b , i. e., 8 and 16 bins, are analyzed.

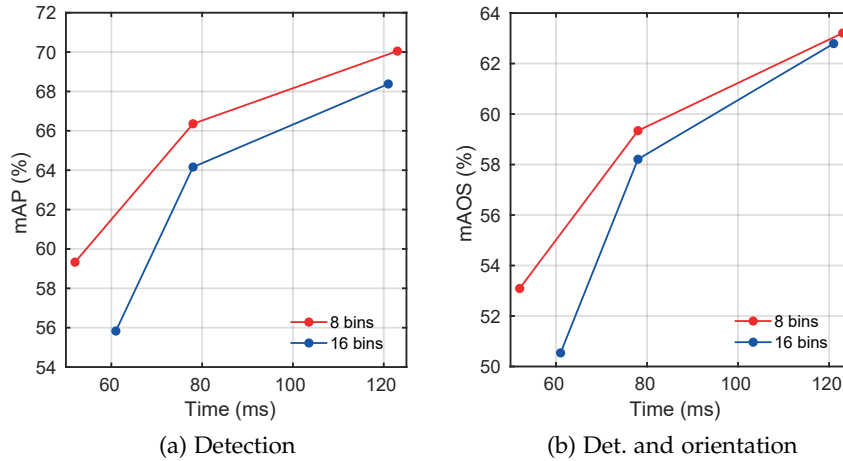


Figure 5.15: Detection and viewpoint estimation performance (mAP % and mAOS %) of the proposed approach vs. run time on the KITTI *Moderate* validation subset with the VGG-16 backbone for different numbers of bins and input scales (hyperparameters: set 2) [114]

According to the results, the improvement achieved by enlarging the image from 500 to 650 pixels is less significant than the experienced when applying the original $1.33\times$ scaling. It can be concluded that performance plateaus for large scales and is no longer worthwhile after 650 pixels considering the increased computational burden.

A breakdown of the performance in terms of AP and AOS for six classes from the KITTI dataset at different scales is presented in Table 5.10. In this case, the number of proposals is fixed to 100 and $N_b = 8$. In addition to providing a detailed review of the effect of the scale on the detection of each category, this is the first time in this document that results are provided for classes other than *Car*, *Pedestrian*, and *Cyclist*⁹.

Predictably, the effect of the input scale in *Easy* samples is lower since all of them are big enough to be well represented in the studied scales (larger than 40 pixels). Overall, results show the suitability of the method for multi-class detection with a large number of categories. Several factors can explain poor results obtained for *Van* and, mainly, *Truck*: the very reduced number of training samples, the high intraclass

⁹ However, it should be noted that every experiment performed with the set 2 of hyperparameters was trained to predict all the available classes; stats for the other classes were just ignored before.

CL.	SCALE (PIX.)	2D DETECTION (AP 2D)			2D DET. AND OR. (AOS)			TIME (ms)
		EASY	MOD.	HARD	EASY	MOD.	HARD	
CAR	375	83.86	71.59	56.33	82.24	69.78	54.37	52
	500	88.54	77.83	60.40	86.88	76.14	58.86	78
	650	90.14	84.49	67.11	88.75	82.92	65.53	123
PED	375	77.53	64.48	59.68	71.13	58.31	53.74	52
	500	79.73	67.77	61.46	73.42	61.42	55.70	78
	650	85.57	70.39	66.34	78.76	64.28	60.24	123
CYC	375	56.88	41.91	40.96	42.02	31.17	30.68	52
	500	73.79	53.47	52.12	54.12	40.45	39.47	78
	650	74.32	55.28	54.23	57.19	42.42	41.78	123
VAN	375	37.16	30.25	30.32	36.59	29.00	28.60	52
	500	44.39	38.87	37.52	43.90	37.16	36.06	78
	650	40.08	34.78	35.58	39.66	32.76	33.60	123
TRK	375	10.58	9.96	6.11	9.04	8.31	5.05	52
	500	15.03	14.77	10.56	11.52	9.94	7.21	78
	650	10.75	12.30	9.17	10.62	10.35	7.19	123

Table 5.10: Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset for different scales and difficulty levels. Results are obtained using the VGG-16 backbone and 100 proposals (hyperparameters: set 2) [114]

variability (e. g., box trucks vs. dump trucks), and the demanding IoU threshold imposed for these categories (70%, the same as *Car*).

5.3.5 Training data

SUMMARY OF THE ANALYSIS

This section is intended to be a continuation of the analysis of the influence of training data which began in Sec. 4.4. In this section, the analysis is extended to the proposed detection and viewpoint estimation method, with more experiments and additional metrics.

As a brief reminder, the goal of this analysis is to explore the potential improvements that can be achieved by moderately increasing the number of training samples. In particular, the KITTI train subset (3682 frames) is extended by introducing samples from the Cityscapes dataset (2975 frames). Please refer to Sec. 4.4 for details about the implementation for the general detection framework.

Regarding the viewpoint estimation part, introduced in this chapter, it is necessary to note that Cityscapes samples are not endowed with orientation annotations. Therefore, at training, the viewpoint component of the multi-task loss is set to nil when processing a frame from the Cityscapes set ($L_5 = 0$).

Table 5.11 shows detection and viewpoint estimation results for three training sets: the KITTI training subset, the combination of KITTI and Cityscapes in the same training procedure (K + CS), and an additional strategy based on finetuning on KITTI a model which was previously trained on Cityscapes, K (CS pret.). As always, testing is performed on the KITTI validation subset.

CL.	TR. DATA	2D DETECTION (AP 2D)			2D DET. AND OR. (AOS)		
		EASY	MOD.	HARD	EASY	MOD.	HARD
CAR	KITTI	90.01	79.03	69.67	88.26	77.35	67.97
	K + CS	90.39	84.59	70.21	88.68	82.79	68.57
	K (CS pret.)	90.33	86.16	70.58	88.63	84.43	69.01
PED	KITTI	71.19	64.05	55.75	65.31	57.62	50.01
	K + CS	76.32	67.98	59.11	67.83	59.65	51.69
	K (CS pret.)	74.54	66.01	57.68	67.33	59.01	51.52
CYC	KITTI	77.33	54.87	52.89	69.73	48.79	47.06
	K + CS	86.11	68.49	63.46	77.66	61.23	56.83
	K (CS pret.)	83.18	60.37	57.35	75.55	54.36	51.74

Table 5.11: Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset using different sets of training data (hyperparameters: set 3). K + CS represents the combined KITTI and Cityscapes dataset, whereas K (CS pret.) refers to the strategy of pre-training with Cityscapes [115] © 2018 IEEE

According to the obtained results, extending the baseline KITTI train subset improves both the AP and AOS for all difficulty levels, with the mixed KITTI + Cityscapes training outperforming the two-stage alternative. It is noteworthy that the improvement in detection performance reaches +7.71 points in mAP for *Moderate* samples, thus exceeding by a large margin the improvement achieved without the viewpoint estimation branch (+4.24 points).

Remarkably, introducing the Cityscapes frames without viewpoint annotations also improves the AOS results significantly. The increase in mAOS reaches +6.64 points for *Moderate* samples. The increase can be partly explained by the fact that AOS jointly assesses detection and viewpoint estimation performance and is upper bounded by the corresponding AP result as only the viewpoint estimated for correctly detected instances is evaluated.

To isolate the effect of the new samples on the viewpoint classification problem itself, MPPE is employed. The average value of MPPE across all recall values (Average MPPE) for the two first alternatives (KITTI alone, and mixed KITTI-Cityscapes) is presented in Table 5.12.

The results confirm that viewpoint classification does not get hurt at all by the lack of viewpoint annotations in some frames of the

OVERALL EFFECT

EFFECT ON BIN
CLASSIFICATION
ACCURACY

CL.	TR.	VIEWP. CLAS. (A-MPPE)		
		DATA	EASY	MOD.
CAR	KITTI	92.24	80.93	69.29
	K + CS	92.13	83.40	71.72
PED	KITTI	59.03	51.02	43.71
	K + CS	57.74	51.35	44.41
CYC	KITTI	70.71	49.84	49.00
	K + CS	64.95	51.60	49.11

Table 5.12: Viewpoint classification performance (MPPE %) of the proposed approach on the KITTI validation subset using different training data sets (hyperparameters: set 3) [115] © 2018 IEEE

combined dataset. Hence, frames from the KITTI train subset suffice to train the viewpoint estimation capability, whereas the new samples conveniently improve the detection functionality of the algorithm.

PRE-TRAINING

As is standard practice, all experiments are performed starting from the weights of a model pre-trained on ImageNet [231], made of a large number of diverse images that enable the generation of a baseline model with high representation capabilities. The experiment shown in Table 5.13 evidences that the enlargement of the training dataset is still not enough to replace the generalization ability provided by the ImageNet pre-training.

CL.	P	TR.	2D DETECTION (MAP 2D)			2D DET. AND OR. (MAOS)		
			DATA	EASY	MOD.	HARD	EASY	MOD.
mean	✓	KITTI	79.51	65.98	59.44	74.43	61.25	55.02
	✗	K+CS	53.80	42.99	37.25	47.93	39.13	33.00

Table 5.13: Detection and viewpoint estimation performance (mAP % and mAOS %) of the proposed approach on the KITTI validation subset with and without pre-training (P) on ImageNet (hyperparameters: set 3). K+CS denotes the combined KITTI-Cityscapes dataset [115] © 2018 IEEE

DATA AUGMENTATION

Training data can also be enlarged through data augmentation techniques, which expand the number of samples by creating modified versions of images in the dataset. All models in this thesis are trained with horizontal flipping augmentation; however, in this section, more techniques are examined to create variations of the original images. Following [221], four *texture augmentations* are employed:

- Intensity addition: a random value between -40 and $+40$ is added to all pixels in the image.

- Intensity multiplication: all pixels in the image are multiplied by a factor randomly picked from the range $[0.5, 1.5]$.
- Additive Gaussian noise: a small jitter is added to each pixel, following a Gaussian distribution with mean 0 and standard deviation randomly chosen from $[0, 5.1]$
- Hue and saturation addition: a random value in the range $[-20, 20]$ is added to the H and S channels of the image when expressed in HSV color space.

These transformations, whose effects for their respective extreme values are illustrated in Fig. 5.16, not only increase the number of training samples, thus avoiding overfitting, but also improve the robustness of the model against changes of illumination, shadows, and other noise effects caused by the sensing elements (i. e., lens and sensor).

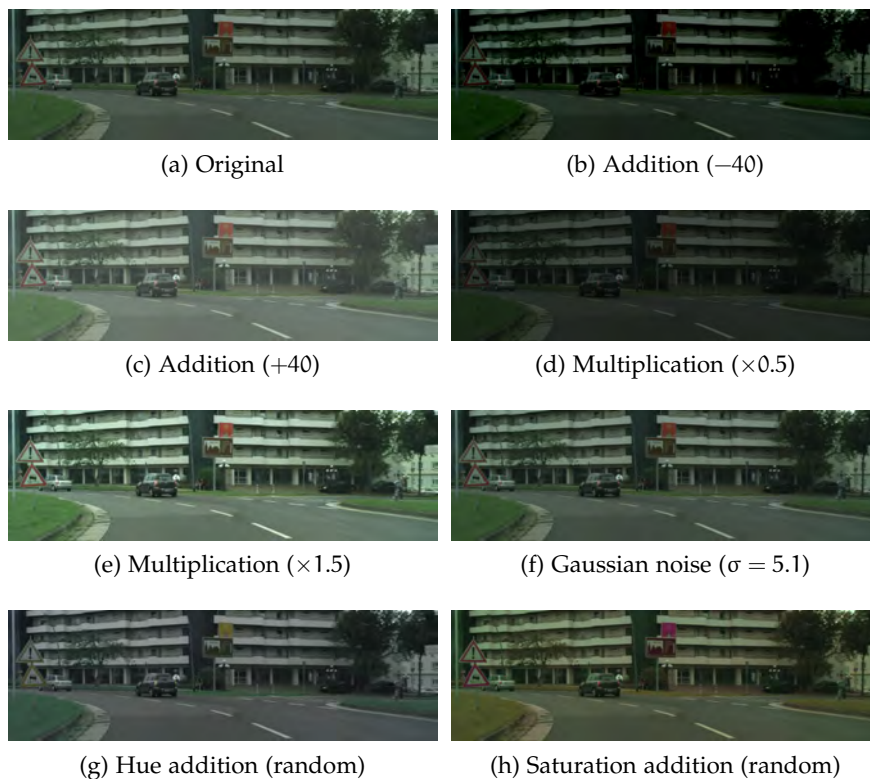


Figure 5.16: Images obtained by applying each data augmentation technique to the same Cityscapes frame

During training, a random quantity between 0 and 4 of these transformations is applied over the input image, increasing the diversity of possibilities. Two different training sets have been augmented and tested: the KITTI training subset and the mixture between KITTI and Cityscapes samples. Results on the KITTI validation subset are shown in Table 5.14.

TR.	AUG	2D DETECTION (MAP 2D)			2D DET. AND OR. (MAOS)		
		EASY	MOD.	HARD	EASY	MOD.	HARD
KITTI	✗	79.51	65.98	59.44	74.43	61.25	55.02
	✓	80.39	65.87	58.96	74.56	61.00	54.38
K+CS	✗	84.27	73.69	64.26	78.06	67.89	59.03
	✓	83.96	74.14	65.16	77.95	68.09	59.59

Table 5.14: Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset with and without data augmentation (hyperparameters: set 3) [115] © 2018 IEEE

As observed, the effect of data augmentation techniques is reduced for both training sets when testing on the KITTI validation set, with a maximum improvement of +0.9 points in mAP for the *Hard* difficulty level. A plausible explanation is that KITTI validation frames lack diversity (e. g., illumination conditions and sensor devices) and, therefore, the improved robustness of the model cannot be exploited.

Fig. 5.17 shows some examples of the improvement in detection performance introduced by the new Cityscapes samples and data augmentation. The baseline detections are in the left column, while the images in the right column have been processed with the model trained with the mixed datasets. It is apparent that, with the enhanced training data, some false negatives become correctly identified.

QUALITATIVE
RESULTS

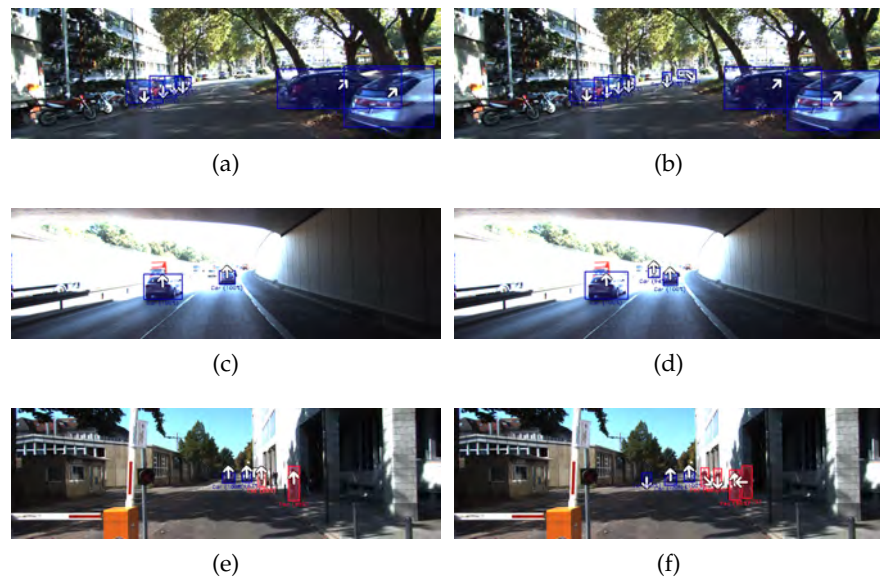


Figure 5.17: Selected examples of object detection and viewpoint estimation results on the KITTI test dataset. (a, b, c) with a model trained on the KITTI training split; (d, e, f) with a model trained on the combined dataset [115] © 2018 IEEE

Finally, the possibility of training the proposed approach with *heterogeneous annotations* is investigated. Some of the categories in KITTI and Cityscapes are different from each other; for instance, *Tram* in KITTI and *train* in Cityscapes, or *traffic sign*, which is labeled in Cityscapes but is not in KITTI. The idea is to create an extended set of classes useful for traffic scene understanding by combining both groups. Category mapping is described in Table 5.15.

HETEROGENEOUS
LABELS

CATEGORY	KITTI CATEGORY	CITYSCAPES CATEGORY
<i>Car</i>	<i>Car</i>	<i>car</i>
<i>Truck</i>	<i>Truck and Van</i>	<i>truck</i>
<i>Pedestrian</i>	<i>Pedestrian and Person_sitting</i>	<i>person</i>
<i>Cyclist</i>	<i>Cyclist</i>	<i>bicycle + rider</i>
<i>Train</i>	<i>Tram</i>	<i>train</i>
<i>Traffic Sign</i>	N.A.	<i>traffic sign</i>

Table 5.15: Mapping of the extended set of categories to the original labels in KITTI and Cityscapes

Qualitative results, shown in Fig. 5.18, prove the validity of the approach. The particular case of traffic signs is especially relevant since the model is able to capture the appearance of the class even if they are labeled as *background* in KITTI samples.

QUALITATIVE
RESULTS USING
HETEROGENEOUS
LABELS



Figure 5.18: Selected examples of object detection and viewpoint estimation results on the KITTI test dataset with additional categories [115]
© 2018 IEEE

5.4 IMPROVEMENTS IN THE BASELINE SOLUTION

This section aims to introduce some ideas that might dramatically enhance the performance of the proposed detection and viewpoint estimation method but could not be analyzed in full detail due to time considerations. The first proposal aims to improve the viewpoint

estimation by adding a regression step to the existing bin classification procedure, thus giving rise to a *hybrid* viewpoint estimation approach. The second issue to be discussed in this chapter is the adequacy of the approach to be part of modern detection architectures. Both ideas boost the performance of the method to the maximum, and should, therefore, be seriously considered as future research lines.

5.4.1 Hybrid viewpoint estimation

OVERVIEW

The orientation estimation provided by the method proposed in the previous sections has proven satisfactory for traffic scene modeling, as will be discussed in Sec. 6.1. Nevertheless, the estimation can be further refined by introducing a complementary regression step. In the proposal introduced in the previous sections, viewpoint estimation is seen as a classification problem into discrete angle bins. An additional regression branch can be added to estimate the residual between the center of the predicted bin and the object's viewpoint, as represented in Fig. 5.19.

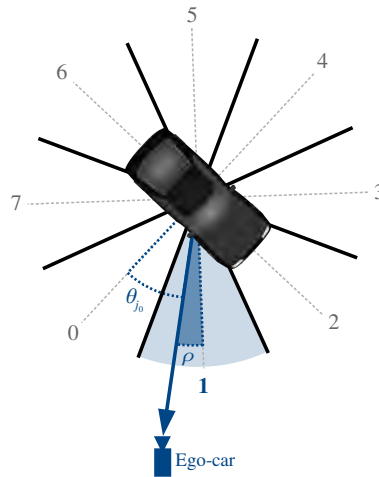


Figure 5.19: Example of residual angle (ρ). The object's viewpoint falls into the first bin

SINGLE VALUE ENCODING

Two alternatives for residual regression will be proposed. The first one estimates the unit-variance-scaled residual value explicitly. As with the bin classification, residual regression is class-aware. Therefore, the output of the residual regression branch of the CNN is a set of $K + 1$ vectors s^k , one per class:

$$\mathbf{s}^k = (s_0^k, \dots, s_{N_b-1}^k), \quad k = 0, \dots, K, \quad (5.18)$$

where only the element corresponding to the index of the most probable bin according to the viewpoint classification branch, $s_{\hat{1}}^k$, is consid-

ered, representing the estimated distance between the center of that bin ($Z(\Theta_{\hat{l}})$) and the actual viewpoint (θ):

$$s_{\hat{l}}^k \propto Z(\Theta_{\hat{l}}) - \theta \quad (5.19)$$

Note that $Z(\Theta_{\hat{l}})$ was the estimation obtained by the viewpoint classification branch through Eq. 5.9. A smooth-L1 loss is used to take into account this new task:

$$L_6 = \frac{1}{N_{B_2}} \sum_{i \in B_2} [v_i \geq 1] \text{smooth}_{L_1}(s_{i, w_i}^{v_i}, x_i, 1), \quad (5.20)$$

where $s_{i, \hat{l}}^{v_i}$ is the \hat{l} th element of s^k for the sample i from the mini-batch B_2 , and x_i , the residual target. Only residuals belonging to a foreground class ($v_i \geq 1$) and with the same index as the ground-truth viewpoint bin (w_i) contribute to the loss. Residual targets are normalized using the expected standard deviation as a scaling factor so that they have unit variance. Then, if it is assumed that ground-truth viewpoints follow a uniform distribution:

$$x_i = \frac{Z(\Theta_{w_i}) - \theta}{\frac{2\pi}{N_b \sqrt{12}}}, \quad (5.21)$$

with w_i being the index of the ground-truth bin.

It can be said that the viewpoint residual regression performed according to this approach is not only class-aware but also *bin-aware* since a separated residual is estimated for each possible class-bin combination.

The second alternative for residual regression encodes them using the sine and cosine of the angle, following [274]. In this particular case, the output of the residual regression branch for class k is as follows:

$$\mathbf{s}^k = (s_{0(s)}^k, s_{0(c)}^k, s_{1(s)}^k, s_{1(c)}^k \dots, s_{N_b-1(s)}^k, s_{N_b-1(c)}^k), \quad k = 0, \dots, K, \quad (5.22)$$

Hence, the inference output for category k is a $N_b \times 2$ -element vector. Each element $s_{l(s)}^k$ represents the sine of the scaled residual for bin l , whereas $s_{l(c)}^k$ refers to the cosine of that same scaled residual:

$$s_{l(s)}^k = \sin(N_b \cdot (Z(\Theta_{\hat{l}}) - \theta)) \quad (5.23)$$

$$s_{l(c)}^k = \cos(N_b \cdot (Z(\Theta_{\hat{l}}) - \theta)) \quad (5.24)$$

Because of the scaling by a factor of N_b , this encoding provides a smooth and unique mapping from the range of possible values of the residual, $[-\pi/N_b, \pi/N_b]$. The estimated residual ($Z(\Theta_{\hat{l}}) - \theta$) can be then obtained as:

$$Z(\Theta_{\hat{l}}) - \theta = \frac{1}{N_b} \arctan \left(\frac{s_{\hat{l}(s)}^k}{s_{\hat{l}(c)}^k} \right) \pm \frac{\pi}{N_b} \quad (5.25)$$

SINE/COSINE
ENCODING

The loss component of residual regression becomes then:

$$L_6 = \frac{1}{N_{B_2}} \sum_{i \in B_2} [v_i \geq 1] \sum_{\xi \in s, c} \text{smooth}_{L_1}(s_{i, \hat{l}(\xi)}^{v_i}, x_{i(\xi)}, 1), \quad (5.26)$$

where the regression targets $x_{i(s)}$ and $x_{i(c)}$ are computed analogously to $s_{i(s)}^k$ and $s_{i(c)}^k$ (Eqs. 5.23 and 5.24) with the ground-truth bin index (w_i) instead.

In both approaches, the new L_6 loss is added as a summand in the multi-task loss of Eq. 5.14, with a weight of 4; that is, the new multi-task loss is:

$$L = \sum_{i=1}^6 \alpha_i L_i, \quad (5.27)$$

with $\alpha_i = 0 \forall i \in \{1, 2, 3, 4, 5\}$ and $\alpha_6 = 4$. This increased weight allows for the smaller magnitude of the loss associated with residual estimation.

Experiments to prove the adequacy of the solution were conducted on the KITTI dataset, according to the training setup reported in Table 5.16. This setup largely follows the set 3 of hyperparameters introduced in Table 5.6.

EXPERIMENTAL
RESULTS AND
DISCUSSION

Feature extractor:	VGG-16 [244]
Pre-training:	Yes, on ImageNet
Classification loss:	Infogain
RPN proposals:	300
Classification heads:	Category + b. box regression + viewpoint estimation + residual regression
Predicted classes:	3 (CAR, PED, CYC)
Train/val split:	Chen <i>et al.</i> [46]
Training schedule:	50k iterations @ lr = 10^{-3} + 30k iterations @ lr = 10^{-4} (unless otherwise stated)

Table 5.16: Training hyperparameters for the residual regression proposal

Table 5.17 shows the results in terms of AP and AOS on the KITTI validation subset. Four alternatives are used in the comparison: the baseline method using the center of the estimated bin, the interpolation method introduced in 5.2.1, and the two residual regression alternatives proposed in this subsection.

Experimental results show an improvement of +1.55 points in mAOS (Moderate samples) from the non-refined variant for the first approach. The second alternative, which uses the sine/cosine encoding, enlarges the increase in mAOS to +1.92 points. Overall, the detection performance does not get hurt despite the increased number of components of the multi-task loss and even gets improved in some configurations.

CL.	V. REFINE.	2D DETECTION (AP 2D)			2D DET. AND OR. (AOS)		
		EASY	MOD.	HARD	EASY	MOD.	HARD
CAR	None	90.01	79.03	69.67	88.26	77.35	67.97
	Interp.				88.33	77.42	68.04
	Res. (v. 1)	90.14	79.15	69.75	89.99	78.81	69.23
	Res. (v. 2)	89.99	78.95	69.54	88.92	77.82	68.42
PED	None				65.31	57.62	50.01
	Interp.	71.19	64.05	55.75	65.46	57.77	50.15
	Res. (v. 1)	72.08	64.45	55.99	65.40	57.65	49.98
	Res. (v. 2)	73.32	65.20	56.86	66.68	58.56	51.11
CYC	KITTI				69.73	48.79	47.06
	Interp.	77.33	54.87	52.89	69.89	48.92	47.18
	Res. (v. 1)	77.79	54.21	51.85	73.60	50.44	48.29
	Res. (v. 2)	80.24	57.43	54.60	72.44	52.03	49.35

Table 5.17: Detection and viewpoint estimation performance (AP % and AOS %) of the proposed approach on the KITTI validation subset using different alternatives for viewpoint refinement: no refinement (None), interpolation (Interp.), and the two proposed methods for residual regression (Res. (v. 1) and Res. (v. 2))

As already discussed, the AOS measure assesses joint detection and orientation estimation, which is convenient to represent the real performance of the algorithm but makes it difficult to decouple both problems. With the proposed residual regression approaches, the viewpoint estimation problem is no longer posed as a multi-class classification; therefore, the MPPE metric is not valid to study the pure orientation estimation performance. Instead, the AOS/AP ratio is computed and reported in Table 5.18. As the AOS only takes into account positive detections, it is upper bounded by the AP, and the AOS/AP ratio gives an overview of the orientation estimation performance on the correctly identified instances.

Contrary to what the AOS results seem to suggest, the direct regression approach offers the best overall orientation prediction performance. It is particularly interesting to analyze the effect of both residual regression alternatives on the *Cyclist* class: the first one largely improves the pure orientation performance, whereas the second one leads to a significant increase in the pure detection performance, which raises the AOS further.

However, regardless of the viewpoint refinement technique, the proposed architecture achieves AOS/AP ratios above 97% for *Car*, and 89% for *Pedestrian* and *Cyclist*. Although the smoothness of the AOS

CL.	V. REFINE.	O/D RATIO (AOS/AP 2D)		
		EASY	MOD.	HARD
CAR	None	98.06	97.87	97.57
	Interp.	98.13	97.96	97.67
	Res. (v. 1)	99.83	99.56	99.25
	Res. (v. 2)	98.81	98.57	98.40
PED	None	91.74	89.96	89.70
	Interp.	91.95	90.20	89.94
	Res. (v. 1)	90.73	89.45	89.27
	Res. (v. 2)	90.95	89.82	89.88
CYC	KITTI	90.17	88.93	88.97
	Interp.	90.38	89.16	89.19
	Res. (v. 1)	94.61	93.06	93.14
	Res. (v. 2)	90.27	90.61	90.38

Table 5.18: Orientation performance (AOS % / AP % ratio) of the proposed approach on the KITTI validation subset using different alternatives for viewpoint refinement: no refinement (None), interpolation (Interp.), and the two proposed methods for residual regression (Res. (v. 1) and Res. (v. 2))

metric might conceal some inaccuracies, these results confirm the adequacy of the selected approach.

5.4.2 Validation of the general approach in modern frameworks

INTRODUCTION

The approach introduced in the preceding sections for embedding the viewpoint estimation into a two-stage detection architecture is a conceptually simple, general framework that is not limited to the use case presented in the experimental setup of this thesis but, instead, admits a wide range of setups.

DETECTRON SOFTWARE SYSTEM

In January 2018, Facebook AI Research (FAIR) released *Detectron*¹⁰, a suite based on the DL framework Caffe 2 that includes the official implementation of Mask R-CNN [118], which, as mentioned in Chapter 4, is a top-performing instance segmentation method. As Mask R-CNN is, essentially, an extension of Faster R-CNN, *Detectron* enables the training and testing of Faster R-CNN models that incorporate the latest advances in DNN-based detection, including:

- Residual architectures [120] as a feature extractor. ResNets, introduced in Sec. 2.4.3, are nowadays the backbone of choice for a wide range of applications. The use of residual blocks allows increasing the number of layers and leads to significant gains in the representation ability of the feature extractor.

¹⁰ <https://github.com/facebookresearch/Detectron>

- Feature Pyramid Network (FPN) [167]. As described in 2.6.1, FPNs improve the feature extraction step by using feature maps at different scales in both the RPN and the classification heads.
- ROI align [118]. The ROI pooling operation featured by the original Faster R-CNN has been shown to present notable design flaws. ROI align, originally intended to improve the performance of pixel-wise mask prediction, provides an elegant replacement for extracting per-ROI feature maps.

These advances and others of equal importance were introduced concurrently with the development of this thesis, which is the reason why they were not employed in the experimental setup of the previous sections. However, the proposed viewpoint estimation functionality is a general approach and, therefore, is compatible with all these complementary techniques.

This section is intended to provide insight into the potential of the method when using contemporary bells and whistles in the detection framework. As will be shown, results make it clear that the approach is not tailored to a specific setup, but, instead, can be considered a generic plug-in with application in a diversity of architectural variants.

In order to prove this claim, a comparison between the baseline implementation (*lsi-faster-rcnn*, featuring a VGG-16 model as feature extractor) used in the previous sections and another one based on Detectron (with a ResNet-50 backbone) is presented. The experimental setup employed in the experimentation is summarized in Table 5.19.

EXPERIMENTAL
RESULTS

Feature extractor:	VGG-16 [244] (<i>lsi-faster-rcnn</i>), ResNet-50 [120] with FPN [167] (Detectron)
Pre-training:	Yes, on ImageNet (<i>lsi-faster-rcnn</i>) and COCO (Detectron)
Anchors:	Default (3 scales in <i>lsi-faster-rcnn</i> and 4 in Detectron)
Classification loss:	Infogain (<i>lsi-faster-rcnn</i>), cross-entropy loss (Detectron)
RPN proposals:	300 (<i>lsi-faster-rcnn</i> and Detectron), 1 000 (Detectron)
Classification heads:	Category + b. box regression + viewpoint estimation
Predicted classes:	3 (CAR, PED, CYC)
Train/val split:	Chen <i>et al.</i> [46]
Training schedule:	50k iterations @ lr = 10^{-3} + 30k iterations @ lr = 10^{-4} (<i>lsi-faster-rcnn</i>), 60k iterations @ lr = 10^{-3} (Detectron)

Table 5.19: Set of training hyperparameters for the implementation of the proposed approach in Detectron

Some settings, such as the training schedule or the COCO pre-training, were selected according to the results provided by the Mask R-CNN authors on the Cityscapes dataset [118], which is similar in characteristics and number of samples to the KITTI dataset employed here. As suggested in that paper, weights corresponding to each KITTI category in the last classification layer are initialized from the corresponding categories in the pre-trained COCO model.

On the other hand, it is noteworthy that both the number of anchor scales and the number of RPN proposals have been increased in the Detectron implementation; despite this, the improvements in the code implementation and the use of a more efficient framework (Caffe 2) lead to even faster run times, as will be shown later. Nevertheless, experiments with a reduced number of proposals (300) have also been performed.

Accuracy and speed results of the newest implementation (*Det.*) in comparison with the baseline version (*lsi-faster-rcnn*) are reported in Table 5.20. Two alternatives are provided for the Detectron option regarding the number of proposals (1000 or 300) fed to the classification heads by the RPN.

CL.	IMPLEM.	2D DETECTION (AP 2D)			2D DET. AND OR. (AOS)			TIME (ms)
		EASY	MOD.	HARD	EASY	MOD.	HARD	
CAR	lsi-faster-rcnn	90.01	79.03	69.67	88.26	77.35	67.97	90
	Det. (1000 p.)	90.37	87.63	78.83	88.77	85.65	76.61	72
	Det. (300 p.)	90.37	87.79	78.93	88.75	85.81	76.70	65
PED	lsi-faster-rcnn	71.19	64.05	55.75	65.31	57.62	50.01	90
	Det. (1000 p.)	75.32	71.29	63.09	65.50	61.29	54.00	72
	Det. (300 p.)	75.36	67.40	63.41	65.50	58.36	54.25	65
CYC	lsi-faster-rcnn	77.33	54.87	52.89	69.73	48.79	47.06	90
	Det. (1000 p.)	82.12	61.40	59.86	73.70	55.24	53.68	72
	Det. (300 p.)	82.32	61.33	57.13	73.61	54.87	51.11	65

Table 5.20: Detection and viewpoint estimation performance (AP % and AOS %) of the baseline implementation (*lsi-faster-rcnn*) and the enhanced implementation (*Det.*) of the proposed approach on the KITTI validation subset

Predictably, the performance of the Detectron alternative is consistently better; indeed, these results represent the peak performance that can be currently achieved using the proposed approach. It should be noted that these results were achieved without a careful tuning of the hyperparameters; in fact, some overfitting problems were observed during the experiments. A more in-depth analysis of the parameters, especially those newly introduced in Detectron (e. g., those corresponding to the FPN), might easily lead to further improvements.

It is also highly remarkable that the proposed approach can be straightforwardly used within the Mask R-CNN framework. As Mask

R-CNN is already implemented in the Detectron suite, the only factor nowadays that has prevented training a multi-task model able to perform simultaneous detection, viewpoint estimation, and (instance) semantic segmentation according to the guidelines proposed in this thesis is the lack of annotated datasets, as the 3D pose of objects (and, therefore, their orientation) has not been a matter of interest in most datasets until very recently¹¹.

Including the proposed viewpoint estimation method into the Mask R-CNN framework would not only provide an additional output of interest for onboard perception systems, but would also improve the detection (and, possibly, the viewpoint estimation) performance of the method. This phenomenon was observed by the authors of Mask R-CNN, who attributed it to the new semantic cues introduced by the mask component of the multi-task loss [118].

5.5 CONCLUSION

This chapter has been devoted to introducing and discussing the approach proposed in this thesis to enhance the scene understanding of automated vehicles through the estimation of the heading angle of objects on the road plane. The method is a straightforward, natural extension of Faster R-CNN, which allows taking advantage of its high detection accuracy while providing valuable insight into the intentions and future trajectories of potentially dangerous objects.

The method has been designed to provide an estimate of the orientation of objects relative to the camera. Both detection and viewpoint estimation are performed from a shared set of appearance features, thus exploiting the close relationship between the two tasks. As motion features are not employed, they can be introduced later in the pipeline as a redundant source to increase the robustness of the method.

Extensive experiments have shown the validity of the approach and its superior performance over similar alternatives. Run times were compatible with onboard requirements, even though suboptimal implementations were used.

Furthermore, the sensitivity of the results against changes in the most critical hyperparameters has been analyzed in depth. Several alternatives have been proposed to reach the desired point of operation along the speed/accuracy trade-off. On the other hand, the positive effects of introducing new training data have also been analyzed.

The approach introduced in this thesis has achieved notable results on the renowned KITTI benchmark, which contains works from scientists all around the world; e. g., as of September 2019, it is still in 12th place in the official *Object Detection and Orientation Estimation*

¹¹ Further research is required to assess if some datasets released in 2019, such as nuScenes [37], do meet the requirements to perform the subsequent multi-task training.

Evaluation for the *Pedestrian* category¹², despite featuring one of the fastest run times among all the ranked methods. Additionally, the source code of the implementation used in Chapters 4 and 5 of this thesis was released under the MIT license to promote reproducibility among the scientific community¹³.

¹² http://www.cvlibs.net/datasets/kitti/eval_object_detail.php?&result=28dc29642148933e41c6daccff01d35b1bdf5ecf

¹³ <https://github.com/cguindel/lsi-faster-rcnn>

The preceding two chapters have introduced a reliable paradigm for object detection and orientation estimation. Nevertheless, the application of this methodology to a real automated vehicle is still hampered by a rather glaring gap: the output provided by the proposed approach is composed of detections localized within the boundaries of the image; however, higher-level, decision-making modules need spatial awareness about the 3D location of objects in the surroundings of the vehicle.

As introduced in Sec. 2.6.3, a manifold of methods have been used to estimate the position of objects in space. It is not unusual that object localization is considered as a subsequent step aimed to endow a set of already existing detections with geometrical information. In particular, when it comes to onboard perception methods, the magnitudes of interest are the 2D position of obstacles in the ground surface and their heading. Likewise, the dimensions of road users are also worthwhile since they enable the definition of obstacles in the environment model as 3D cuboids centered at a point in the coordinate frame attached to the ego-vehicle.

Although 3D pose can be estimated from a single frame [43], [156], accurate localization of obstacles usually makes it advisable to incorporate additional sources of information enabling spatial reasoning. Two of the most typical in automotive applications are stereo-vision systems and lidar scanners. In this chapter, methods introduced in the previous chapters are leveraged to obtain meaningful 3D detections from both modalities.

6.1 OBJECT LOCALIZATION BASED ON STEREO DATA

Firstly, a method to perform object localization using stereo information is presented. This method assumes that objects have been detected and provided with an estimated orientation using the method introduced in Chapters 4 and 5. An overview of the approach is presented in Fig. 6.1.

STEREO VISION

Foundations of stereo vision were widely discussed in Chapter 3. As a brief reminder, two images representing the same scene and acquired from different points of view can be used to perform 3D reconstruction through a procedure known as *stereo matching*. DispNet and SGBM, introduced in Sec. 3.2.1, are used in this chapter to perform stereo matching.

This chapter includes content from [114] and [15].

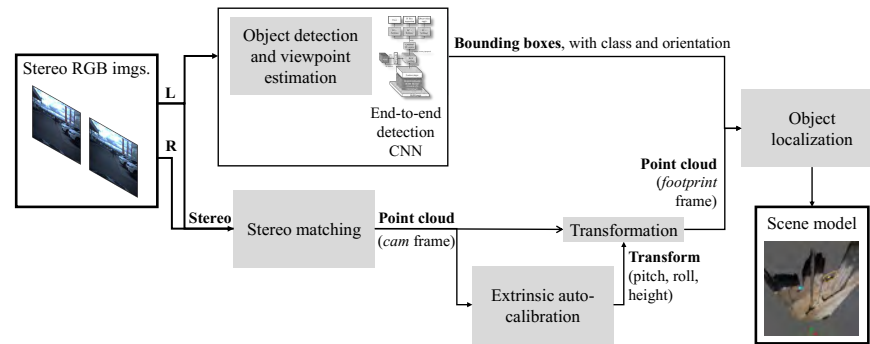


Figure 6.1: Proposed stereo vision-based object localization method

DATA FUSION

The basic idea behind the proposed object localization approach is that it is possible to establish a correspondence between 2D detections in the image and 3D points from stereo matching. The association is straightforward if the stereo system is appropriately calibrated (e. g., the baseline is known), and the resulting 3D cloud is expressed in local coordinates of the camera used to perform detection; in this work, the leftmost camera of the stereo pair is chosen as the reference frame.

In the proposed object localization framework, correspondence between points in both representations is preserved due to the point clouds being organized and projectable. The former implies that they are organized as image-like structures (with rows and columns), and the latter means that this arrangement convey actual information of the correspondence with the image, so that the point with (u, v) index in the cloud contains the 3D location of the (u, v) pixel in the image [207].

REFERENCE
FRAMES

The point cloud resulting from the stereo matching procedure is expressed in a reference frame attached to the optical center of the camera, the *camera* frame. However, high-level modules making use of the information from the perception system require the obstacles to be expressed in a reference frame whose axes are aligned with the main directions of the road plane. The difference is shown in Fig. 6.2, where a *footprint* frame has been defined just below the local *camera* frame¹, aligned with the road plane and with its *x*-axis pointing forward. Please note that the *footprint* frame is located on the ground plane but moves together with the ego-car.

When obstacles are expressed in *footprint* coordinates, measurements represent distances along the road plane, as desirable. Nonetheless, it should be noted that the transformation relating the *camera* and the *footprint* frames does not remains constant over time when the vehicle is moving. Unevennesses of the terrain can alter the relationship between both frames; for instance, speed bumpers significantly modify

¹ In the figure, the camera is assumed to be mounted at the top of the windscreen, facing forwards.

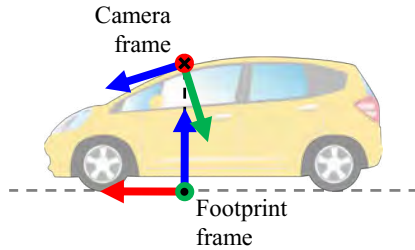


Figure 6.2: Definition of coordinate frames for obstacle localization: local camera frame and footprint frame

the relative pitch angle. An auto-calibration procedure is required to reestimate the transform at each time step.

6.1.1 Extrinsic auto-calibration

The auto-calibration procedure proposed to obtain the transform between the *camera* and the *footprint* frames is based on previous works developed in the LSI [66], [190], [216]. The method obtains the parameters by detecting and segmenting the ground plane in the data from the stereo system, as shown in Fig. 6.3. The ground is assumed flat up to a certain distance.

OVERVIEW

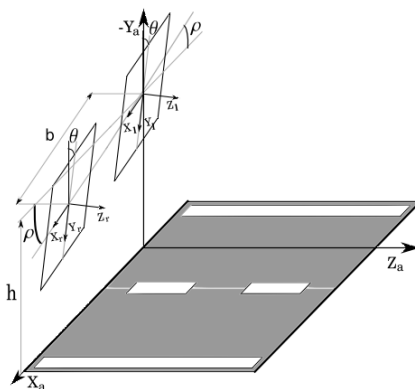


Figure 6.3: Stereo rig and ground plane in a typical setup [190] © 2014 IEEE

In order to avoid spurious detections, only a small patch of ground in front of the camera is taken into account to perform calibration. Successive pass-through filters are applied to the stereo point cloud to filter out points closer than 2 m and further than 20 m, as well as those outside a width range of 12 m around the depth axis. Within these ranges, the flatness assumption is fulfilled with a high probability.

POINT CLOUD
PRE-PROCESSING

Then, the resulting point cloud is downsampled using a voxel grid of size $20 \times 20 \times 20$ cm. In addition to reducing the amount of data to be processed, this filter normalizes the point density along the depth (z) axis. An example of a filtered cloud is depicted in Fig. 6.4a.

PLANE
SEGMENTATION

From this point cloud, the coefficients defining the ground plane can be computed. A RANSAC-like consensus approach [77] is used to fit the points to a plane. Following [66], a tight threshold of 1.5 cm is used. In addition, the search is reduced to planes roughly perpendicular to the vertical axes (y) of the camera, with a tolerance of 0.35 rad. Fig. 6.4b depicts the normal to the ground plane obtained from the exemplary point cloud.

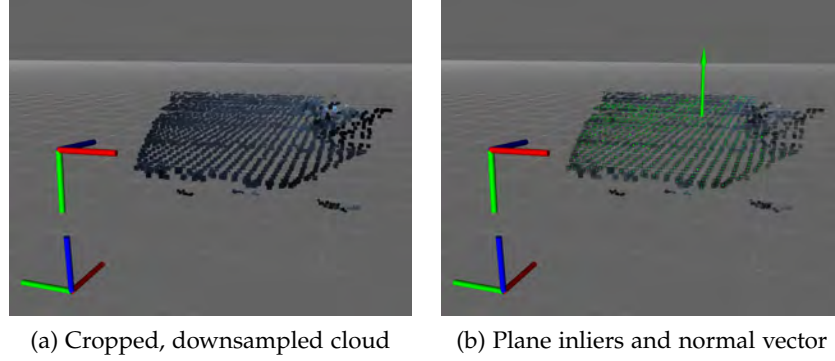


Figure 6.4: Example showing the extrinsic auto-calibration procedure. The pictures show the downsampled cloud, the normal vector to the estimated plane and the *camera* (top) and *footprint* (bottom) coordinate systems for a certain frame [114]

COEFFICIENTS OF
THE PLANE

It can be shown [66] that, given a road plane $ax_c + by_c + cz_c + d = 0$, with (x_c, y_c, z_c) being the coordinates of a point belonging to the plane, roll (ψ), pitch (ϕ) and height (h) defining the camera pose with respect to the ground plane can be obtained as:

$$\psi = \arcsin(a) \quad (6.1)$$

$$\phi = \arctan\left(\frac{-c}{b}\right) \quad (6.2)$$

$$h = d \quad (6.3)$$

Yaw angle cannot be extracted solely from the plane, and thus, it is assumed to be nil. It is noteworthy that this method also produces as a by-product a coarse estimation of the free space in front of the vehicle. This outcome is straightforwardly given by the inliers of the plane segmentation (in green in Fig. 6.4b).

6.1.2 Object 3D localization

PROBLEM
STATEMENT

As stated before, it is possible to retrieve the 3D points corresponding to each obstacle detected in the image. Nevertheless, there is a significant nuance in what the sensor data represents: stereo clouds, as any other sensor data, contain information representing only the visible parts of the objects; in this case, their surfaces. Therefore, recovering the full 3D geometry of the object is far from evident since, in practice,

it involves estimating the non-visible parts. In this section, a method to recover the 3D cuboid describing the obstacle from the stereo data is proposed.

First of all, points in the stereo cloud can be expressed in *footprint* coordinates by using the transform obtained in the auto-calibration procedure. Then, points belonging to the ground, as well as those that are too close to the camera (less than 3 m), are removed.

Another issue that arises due to the use of bounding box representations in 2D detection is that not every 3D point falling inside the bounding box corresponds, actually, to the object. Fig. 6.5 shows clearly the problem: in that case, the peripheral areas of the bounding box represent mostly the background behind the object. In order to deal with this problem, only the innermost part of the bounding box is taken into account for computing the object location. In particular, for a $H \times W$ -pixel bounding box, a smaller rectangle of $5H/7 \times 5W/7$ located in the center of the original proposal is considered, as shown in Fig. 6.5. The dimensions of the inner square have been selected through statistical analysis. It is noteworthy that this approach deals naturally with different scales.

POINT CLOUD
TRANSFORMA-
TIONPOSITION-BASED
SEGMENTATION

Figure 6.5: Area considered by the proposed object localization approach within an object's bounding box

The method aims to obtain the location of the center of each object and its dimensions. To handle the difference between the stereo data, representing surfaces, and the desired measure, fixed dimensions are assumed for the cuboid of every object in each class. Table 6.1 provides the values selected for each category, obtained from aggregated statistical data from the KITTI dataset.

DIMENSIONS

Hence, the final estimation of the cuboid representing the object and its location, as shown in Fig. 6.6, is based on the following inputs:

3D BOX
ESTIMATION

1. The x and y coordinates, in the *footprint* frame, of the surface of the object. The surface is represented by a point $p_s = (x_s, y_s)$ that is obtained from the stereo data by ranking two lists containing the x and y locations of the 3D points corresponding to the inner box depicted in Fig. 6.5, and computing the first quartile of each to provide the estimated x_s and y_s coordinates. The use of the

CATEGORY	LENGTH (m)	WIDTH (m)	HEIGHT (m)
Car (CAR)	3.88	1.63	1.50
Pedestrian (PED)	0.88	0.65	1.77
Cyclist (CYC)	1.76	0.60	1.75
Truck (TRK)	10.81	2.63	3.34
Person sitting (SIT)	0.75	0.59	1.26
Tram (TRM)	14.66	2.60	3.61

Table 6.1: Dimensions of the cuboids assigned to each KITTI category

first quartile is intended to eliminate the influence of outliers and ensures that the estimated point belongs to the surface of the object.

2. The size of the object, $L_0 \times W_0 \times H_0$, assigned according to the values in Table 6.1²
3. The viewpoint estimation θ that was computed together with the bounding box detections based on appearance features from image data.

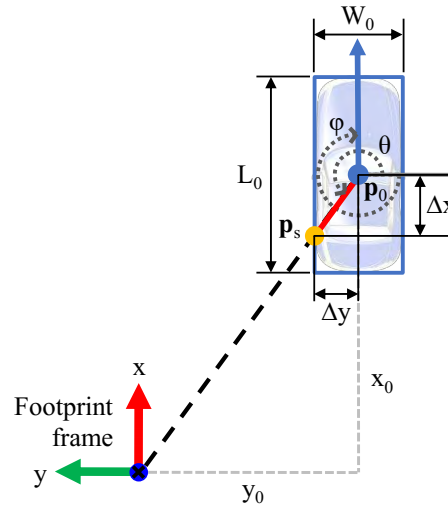


Figure 6.6: Schematic representation of the BEV of a sample scene with a detected object (a car facing towards)

From this, the size and orientation of the cuboid are determined, and its location, $p_0 = (x_0, y_0)$, can be obtained by extending the direction defined by the vector that joins the origin of the *footprint* frame with p_s . The extension has magnitude Δs , where $\Delta s = \sqrt{\Delta x^2 + \Delta y^2}$. Both Δx and Δy are given by the dimensions of the cuboid and the viewpoint

² H_0 , which is not depicted in the top-view representation of Fig. 6.6, is the height of the cuboid.

angle. Once p_0 is known, it is straightforward to convert the viewpoint angle, θ into the yaw angle, φ , using Eq. 5.1:

$$\varphi = \alpha + \operatorname{atan2}(x_0, y_0) + \frac{3\pi}{2}, \quad (6.4)$$

Note that, as discussed in Sec. 5.1, the yaw angle expresses the rotation of the object around a local vertical axis.

6.1.3 Experimental results

Experiments to assess the validity of the approach have been conducted on the KITTI object detection benchmark, where reliable annotations of the 3D location and dimensions of labeled instances are available. Most existing datasets lack these annotations, making the KITTI dataset the right choice for this task, again.

6.1.3.1 3D object localization in the KITTI dataset

The evaluation of the 3D detection task proposed by the KITTI benchmark³ [90] is based on two different kinds of representations: full 3D cuboids⁴, and 2D bounding boxes on BEV representation⁵. The former requires precise estimation of the six dimensions defining the pose of every object, while the latter is focused on the estimation of the measures that are the most important for vehicle navigation: the 2D coordinates on the road plane and the yaw angle. Please refer to Sec. 3.2 for a detailed description of the BEV representation.

Unless otherwise stated, the evaluation follows the criteria established in Sec. 4.2.1, including the IoU requirements regarding the overlap between detections and ground-truth annotations. It is noteworthy that for 3D evaluation, intersection and union are volumes, whereas for BEV evaluation they are areas, although intersection takes into account the box rotation given by the yaw angle. Again, the AP measure is used for quantitative evaluation.

Note also that the baseline of KITTI stereo cameras is approximately 54 cm. Taking into account the intrinsic parameters of the cameras, the stereo matching error from Eq. 3.13 becomes:

$$\delta z[\text{m}] = \frac{z[\text{m}]^2}{389.63} \delta d, \quad (6.5)$$

which expresses the relation between the error made in the estimation, δz , and the distance to the object, z (in m), through δd , which is a constant magnitude that depends on the stereo matching method in use.

³ 3D evaluation was introduced in 2017 as an extension of the existing 2D object detection benchmark.

⁴ http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=3d

⁵ http://www.cvlibs.net/datasets/kitti/eval_object.php?obj_benchmark=bev

EXPERIMENTAL
SETUP

CRITERIA AND
METRICS

STEREO CAMERAS

6.1.3.2 Localization accuracy

MODEL HYPERPA-
RAMETERS

The experiments in this section rely on the outcome of the detection and viewpoint estimation method described in Chapter 5. In particular, a model trained with the set 2 of hyperparameters is employed. The input scale is set to 650, and 100 proposals are classified in each frame. A score threshold of 0.2 is established to filter improbable detection results.

LOCALIZATION
ERROR

Fig. 6.7 shows the distribution of the localization error per category for each of the two stereo matching methods used, considering only true positive detections. The error is computed as the absolute Euclidean distance on the xz -plane of the *camera* frame.

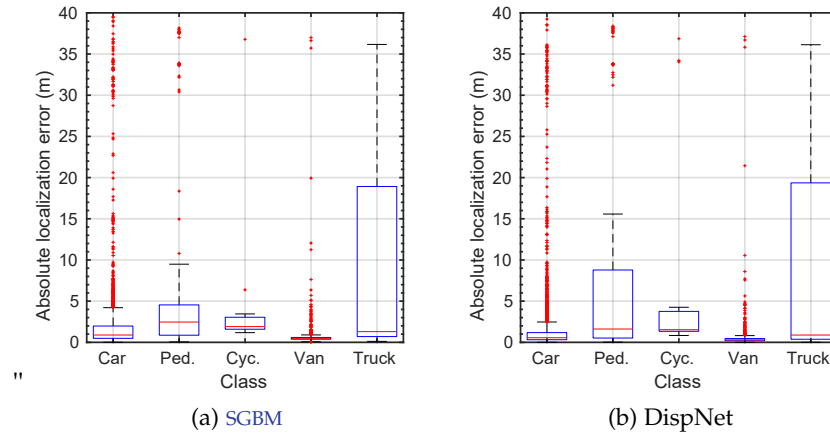


Figure 6.7: Absolute Euclidean error (m) in the location estimation from the proposed approach for the different categories, using two different stereo matching algorithms. The central mark represents the median, and the bottom and top edges of the box indicate the first and third percentiles, respectively; outliers are represented outside the whiskers [114]

Despite the presence of a substantial amount of outliers, the median of the localization error is 77.1 cm when using SGBM and 51.7 cm with the DispNet method. It should be noted that the DispNet model was fine-tuned in the KITTI dataset; overall, the difference is not very significant, and both methods perform reasonably well in the localization of the obstacles ahead of the vehicle for the three main categories of the dataset.

EFFECT ON
DISTANT OBJECTS

From Eq. 6.5, it follows that the depth information error grows quadratically with the distance from the stereo system. Fig. 6.8 studies the distribution of error as a function of the distance along the depth (z) axis of the *camera* frame. The error in depth estimation given by Eq. 6.5 is also depicted.

As apparent, the localization method is reasonably accurate, especially for obstacles closer than 20 m from the camera. Outliers become

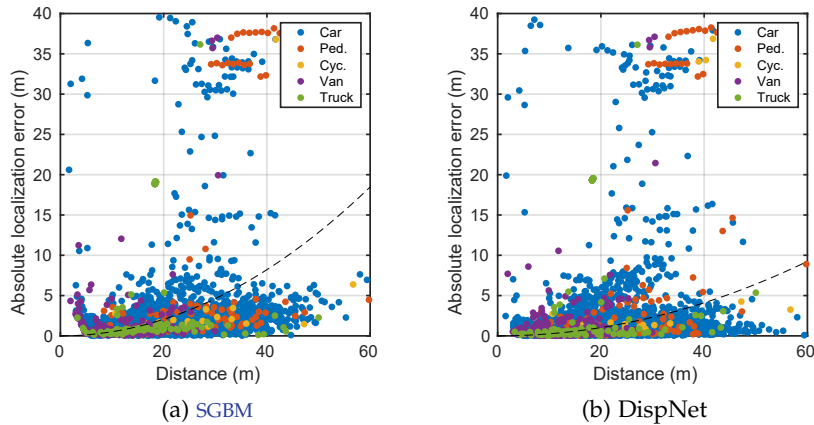


Figure 6.8: Absolute Euclidean error (m) in the location estimation from the proposed approach vs. distance from the ego-car using two different stereo matching algorithms. The dashed line represents the estimated stereo matching error [114]

increasingly abundant at long distances, where occlusions are more likely to occur (e. g., pedestrians behind vehicles).

To study the effectiveness of the approach proposed for obtaining the center of the obstacles from the stereo data, Fig. 6.9 offers a comparison with a naive approach where the predicted location is given by the average of the points within the object bounding box. The non-absolute error is employed to highlight the measurement bias introduced by using surface points as the only source for estimating the location.

COMPARISON
WITH A TRIVIAL
APPROACH

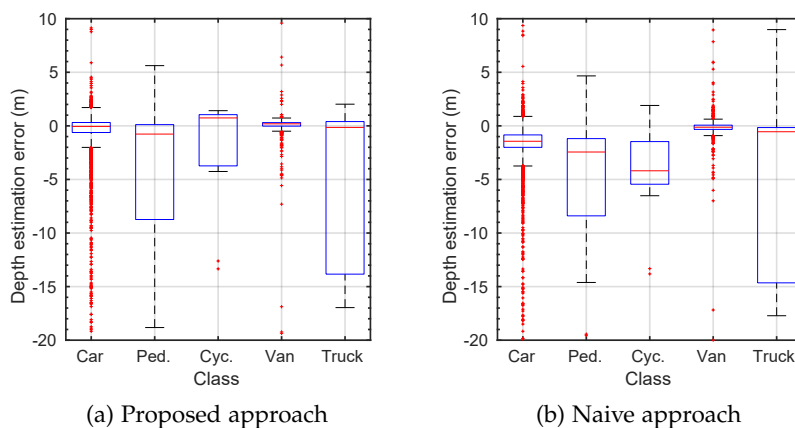


Figure 6.9: Localization error (detection minus ground-truth, m) along the depth axis when estimating the center of the object for two different localization methods. DispNet disparity used [114]

It can be observed that the median of the error is closer to 0 with the proposed approach, whereas the error of the naive approach is

biased towards negative values, as expected. These results prove the soundness of the method to obtain accurate measures.

BEV DETECTION
PERFORMANCE

The problem can be posed as a BEV classification, following the BEV KITTI evaluation. 2D detection stats on BEV perspective are shown in Table 6.2. IoU requirements have been relaxed to obtain a representative view of the performance of the localization method; indeed, results are given for two different minimum IoU levels.

MIN IOU	DIFFIC.	CAR	PED	CYC	VAN	TRK
20%	Easy	77.67	38.55	32.72	25.55	18.68
	Moder.	66.49	31.71	22.22	21.09	12.73
	Hard	49.88	31.42	22.02	15.38	11.16
40%	Easy	70.64	12.64	8.35	22.16	8.68
	Moder.	53.23	12.95	5.19	18.84	8.18
	Hard	38.18	9.17	5.17	13.77	4.55

Table 6.2: BEV detection performance (AP %) of the proposed object localization approach [114]

It is important to note that, even though BEV detection is effectively carried out by the method described in this chapter, the final accuracy is constrained by the performance of the image detection and viewpoint estimation algorithm presented in the previous chapters.

6.1.4 Scene modeling

DEFINITION

As depicted in Fig. 6.1, a pipeline made of the combination of the joint detection and viewpoint estimation CNN with all the steps for object localization described in this chapter can lead to the generation of an object-based model of the environment, where all the potential obstacles are accurately located on the road plane.

KITTI SAMPLES

Two examples of the resulting model for frames from the KITTI testing set are provided in Fig. 6.10. As the resolution in viewpoint estimation has an impact on the resulting model, two different configurations are tested: $N_b = 8$ and $N_b = 16$.

The upper part of each sample depicts the left image, including the bounding boxes representing image detections and an overlay with an estimate of the *free space* in front of the car (in green). This free space estimation is made of the projection of the stereo 3D points located around the road plane, with a threshold of 10 cm. On the other hand, the lower part of the pictures is the *model* of the scene, where a top view of the space in front of the vehicle is represented. Rectangles representing the estimated position of the obstacles' cuboids are depicted over the projection of the 3D stereo cloud. Ground-truth labels are also represented with a thinner line. For scale, the cell size is 10/3 m (i. e., 3.33 m).

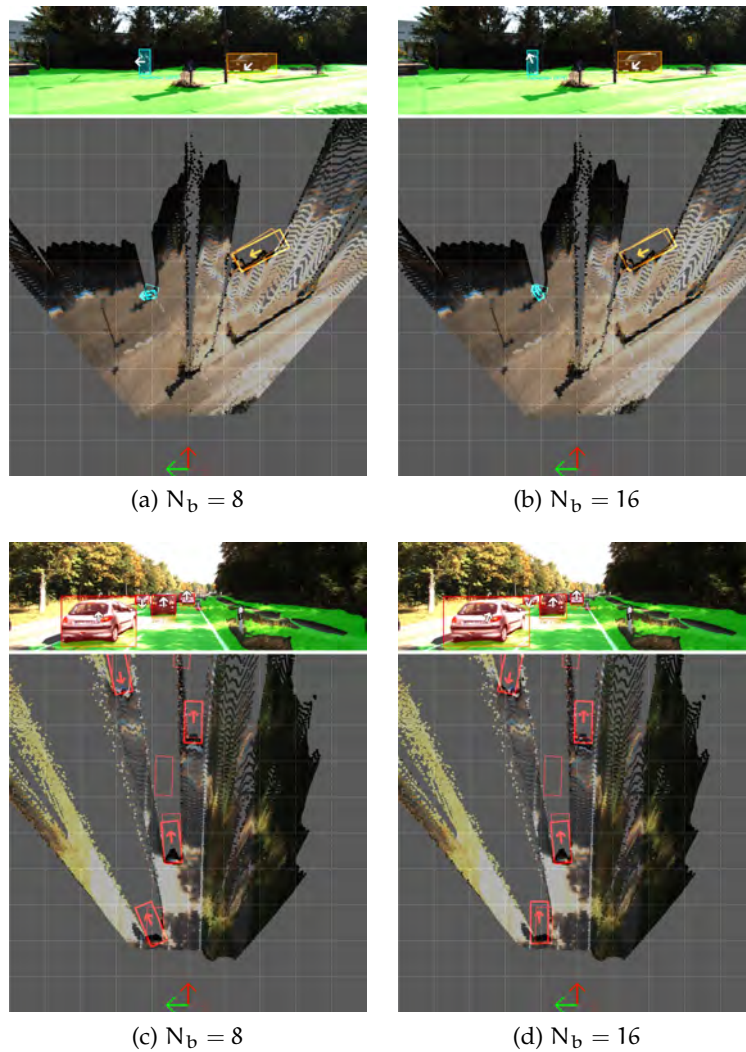


Figure 6.10: Examples of detections and local scene models obtained from the proposed approach for different values of N_b : 8 bins (left) and 16 bins (right). Color code: red for *Car*, blue for *Pedestrian* and orange for *Van* [114]

As seen in the figure, the approach has the potential to provide valuable spatial awareness of the environment. Besides, the method is mainly agnostic to the specific sensor setup. An additional set of experiments has been performed on the LSI's IVVI 2.0 platform, a vehicle endowed with a 12-cm-baseline stereo system, showing the robustness of the proposed approach against changes regarding the sensor device and its positioning. Some examples are provided in Fig. 6.11; in this case, the cell size is 5 m.

It should be noted that, except for the stereo matching stage, the localization method presented here introduces a negligible overhead on top of the detection and viewpoint estimation framework. Concerning the disparity map computation, run times are heavily dependent

IVVI 2.0 SAMPLES

RUN TIME

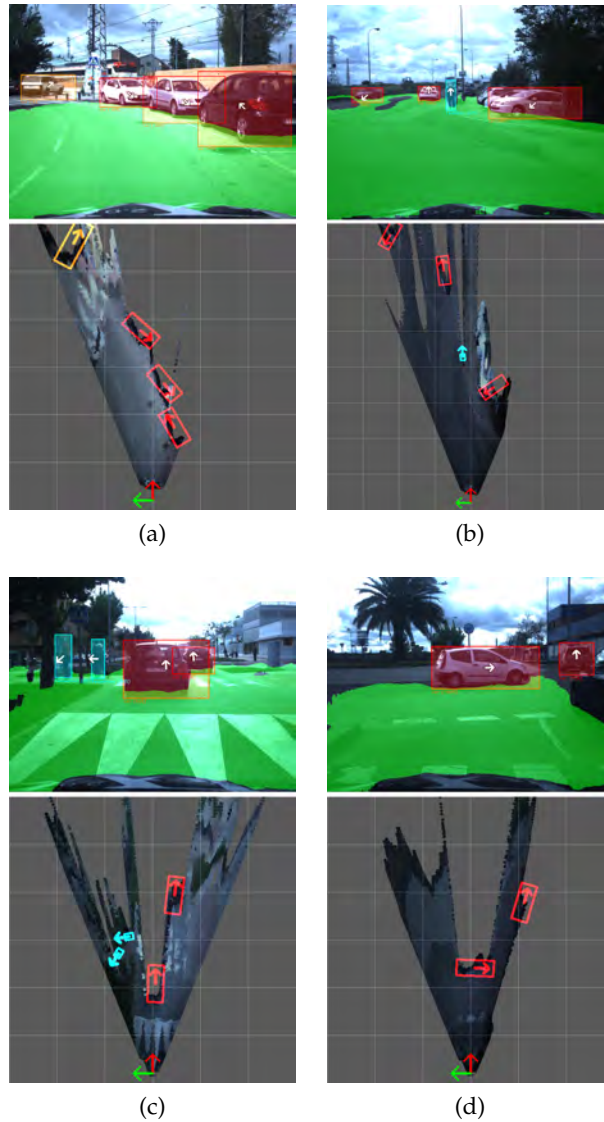


Figure 6.11: Examples of detections and local scene models obtained from the proposed approach in our IVVI 2.0 platform. Color code: red for *Car*, blue for *Pedestrian* and orange for *Van* [114]

on the implementation; in the version used in this thesis, times for DispNet were around 80 ms. Nevertheless, both the stereo matching and the object detection and viewpoint inference can be performed concurrently due to the design of the approach, as shown in Fig. 6.1. This way, information can be delivered at rates around 10 Hz, enabling a fast response to unexpected situations.

6.2 OBJECT DETECTION AND LOCALIZATION BASED ON LIDAR DATA

The approach for object localization introduced in the previous section is intended to complement and enhance the information provided by the image detection CNN from Chapters 4 and 5. However, the CNN framework presented in this thesis can alternatively be used to perform end-to-end object detection and localization (or, in other words, 3D object detection) based on lidar data.

The use of lidar information leads to some improvements over the stereo-based solution, such as increased accuracy at long distances, robustness against fog or illumination, and ease of obtaining information in a 360° range. However, the representation of lidar data also conveys some challenges that are critical for the accuracy of the object detection task.

6.2.1 Detection and yaw estimation in lidar data

In this section, data from a 360° lidar scanner is employed to perform 3D object detection. The general procedure, depicted in Fig. 6.12, was introduced in [15] as *BirdNet*. Here, the analysis is limited to the detection part (in gray), which is indeed the cornerstone of the method.

OVERVIEW

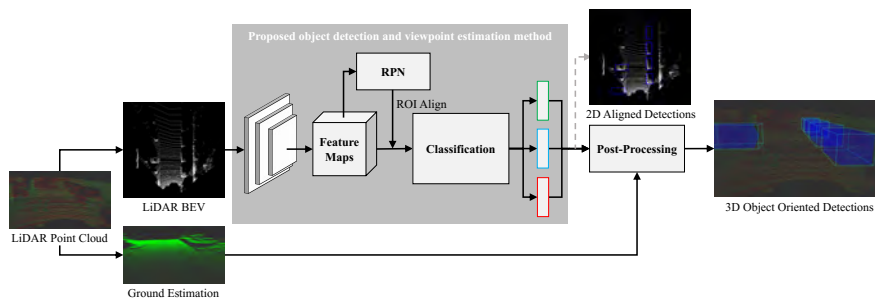


Figure 6.12: Proposed approach for 3D object detection in lidar data (BirdNet), based on the joint detection and viewpoint estimation network © 2018 IEEE [15]

The goal is to take advantage of the Faster R-CNN variant developed for image detection and viewpoint estimation by extending its capabilities to the lidar modality. While it is designed to admit RGB images as input, it can also be seen as a general framework that enables detection over arbitrary 2D, image-like structures.

In this case, the BEV representation introduced in Sec. 3.2 is used to express lidar data as a 2D data structure. As a reminder, the BEV representation is built by discretizing the space around the vehicle into infinitely high cells and providing each cell with a set of variables describing the lidar measurements that fall into its volume. Unless

LIDAR DATA REPRESENTATION

otherwise stated, 5 cm square cells are adopted. Following [46], each cell encodes the following features:

- Height (H), representing the height of the highest point at each cell. Height is measured from the ground, whose position must be therefore estimated beforehand [15].
- Intensity (I), encoding the average reflectance of points in each cell.
- Density (D), computed as the number of points in each cell, normalized by the maximum possible. A novel method for normalization agnostic to the lidar device in use is employed [15].

As each feature is stored in a different channel, this arrangement is structurally identical to an RGB image, so the inference process is not affected. The aim is to perform 2D detection in the BEV, which can be then interpreted as detections in 3D space except for the information in the height axis. The height of the objects, as well as their z location (in *footprint* frame), cannot be recovered, but they can be reasonably estimated given the position of the ground. Besides providing a convenient structure for lidar data, the BEV representation normalizes the resolution of the lidar device so that the method is, in principle, agnostic to the number of measuring layers.

BEV DETECTION
WITHIN FASTER
R-CNN

When a BEV is fed into the network instead of an image, outputs are reinterpreted: bounding boxes are the minimum axis-aligned rectangles enclosing the object in BEV perspective, and the viewpoint estimation from the custom branch is used now as a prediction of the yaw angle. Contrary to the image detection case, bounding boxes do not fit well BEV detection, since most objects appear rotated. However, as a yaw estimation is also provided, an accurate rotated bounding box can be computed in a subsequent post-processing step by assuming a fixed width (1.8 m for *Car* and 0.6 m for *Pedestrian* and *Cyclist*) [15].

CHANGES IN THE
DETECTION
MODEL

The detection CNN uses the VGG-16 backbone as a feature extractor. Throughout this document, feature maps are obtained from the last convolutional layer, *conv5*, as in the original approach. They are, therefore, 16 times smaller than the input image [244]. Since objects, especially pedestrians, are typically represented with a handful of pixels in the BEV, the resolution of the feature maps from the last stage is not suited to this task. Following [46], the last pooling layer in the VGG-16 backbone, *pool4*, is removed. Hence, the downsampling factor from the original image is reduced to 8, providing enough resolution for the detection.

Similarly, ROI pooling is replaced by ROI align, which was proposed as an alternative in [118]. ROI align is intended to fix some quantization issues present in ROI pooling, which caused misalignments between the extracted features and the actual ROI. Due to the small size of BEV objects, small translations could lead to significant reductions in performance.

Anchors were also modified according to the expected sizes in BEV, in the same way as in Sec. 4.2.2.4. Scales and ratios are shown in Table 6.3. Naturally, smaller anchors are chosen, as objects cannot be represented in the BEV by a large number of pixels (exceptions, such as trains, aside).

	DEFAULT	BEV ANCHORS
Scales (box areas)	{128 ² , 256 ² , 512 ² }	{16 ² , 48 ² , 80 ² }
Ratios (H/W)	{2 : 1, 1 : 1, 1 : 2}	{2 : 1, 1 : 1, 1 : 2}

Table 6.3: Modified RPN anchors for BEV detection

The interpolation approach introduced in Sec. 5.2.1 is adopted here to smooth out the predicted yaw values.

6.2.2 Experimental results

Experiments were performed on the KITTI object detection benchmark using, mainly, the BEV evaluation introduced in Sec. 6.1.3, which is indeed the most suitable for evaluating the proposed framework. KITTI lidar data comes from a Velodyne HDL-64E device, with 64 layers. It is noteworthy that annotations are only present in the FOV of the cameras; therefore, although the lidar provides measures in a 360° range, only the frontal ~110° are considered for training and evaluation. On the other hand, the BEV is cropped at 35 m ahead of the vehicle. Given the 5 × 5 cm cells, input images have a resolution of 1400 × 700.

The aim of the investigation carried out in this section is twofold: firstly, to study the influence of both the design decisions and the different input features on the final performance, and secondly, to provide an assessment of the effectiveness of the approach. Training parameters are presented in Table 6.4.

6.2.2.1 Hyperparameter tuning and ablation experiments

As in the previous use cases, the detection CNN was fine-tuned in the KITTI dataset, starting from a set of initial weights from ImageNet. While this made sense when training for image detection, as features were expected to be similar, BEV arrays are radically different from RGB images. Still, results in Table 6.5 prove that initializing the weights from a pre-trained ImageNet model produces significantly better results than using random values.

On the other hand, Table 6.6 investigates the effect of three alternatives:

- With or without the *pool4* layer (P4). As explained before, removing the last pooling operation from the backbone leads to

EXPERIMENTAL
SETUP

TRANSFER
LEARNING

MODEL
ALTERNATIVES

Feature extractor:	VGG-16
Pre-training:	Yes, on ImageNet (unless otherwise stated)
Anchors:	Custom (Table 6.3)
Classification loss:	Infogain
RPN proposals:	300
Classification heads:	Category + b. box regression + viewpoint (yaw) estimation
Predicted classes:	3 (CAR, PED, CYC)
Train/val split:	Chen et al. [46]
Training schedule:	50k iterations @ $lr = 10^{-3}$ + 50k iterations @ $lr = 10^{-4}$ + 50k iterations @ $lr = 10^{-5}$

Table 6.4: Training hyperparameters for the proposed BEV detection network

WEIGHTS	BEV DET. (MAP BEV)			3D DET. (MAP 3D)		
	EASY	MOD.	HARD	EASY	MOD.	HARD
ImageNet	54.46	41.61	40.57	22.92	18.02	16.92
Gaussian	41.89	30.77	29.92	19.76	15.04	14.75

Table 6.5: BEV and 3D detection performance (AP BEV % and AP 3D %) of the proposed BEV detection approach on the KITTI validation subset using different weight initialization strategies [15] © 2018 IEEE

feature maps with better resolution. This modification translates into higher accuracies for the *Pedestrian* and *Cyclist* categories.

- With or without ground (GR). An alternative approach to build the BEV representation without including points belonging to the ground was tested in order to avoid spurious detections in empty areas. However, due to the algorithm used for floor removal, some planar surfaces from the objects were also filtered, thus destroying valuable information for the CNN. The drop in performance is notable.
- Number of bins (N_b). As has been the custom in other parts of this dissertation, the influence of the number of yaw bins has also been studied for two alternatives: $N_b = 8$ and $N_b = 16$. Results show little difference between both configurations, although $N_b = 16$ has a slight advantage.

From now on, results refer to the variant without the *pool4* layer, with ground points (i. e., floor removal is not applied) and 16 bins.

Finally, the relevance of each of the three input features is studied in Table 6.7 by analyzing the performance of models trained with single-channel BEVs. Notably, the model trained with intensity (I) information performs much worse than the other two alternatives,

CLASS	P4	GR	N _b	BEV DETECTION (AP BEV)		
				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

Table 6.6: BEV detection performance (AP BEV %) on the KITTI validation subset for different variants of the proposed BEV detection approach, regarding pool4 layer (P4), ground points (GR) and number of bins (N_b) [15] © 2018 IEEE

density (D) and height (H). Neither of these two models is significantly better than the other. However, the baseline model using the three features achieves the best results for all the categories, thus proving the adequacy of the selected features.

6.2.2.2 Detection performance

Unlike most existing methods, the proposed 3D object detection method can estimate the cuboids corresponding to objects from the three main categories in the KITTI dataset, thus enabling evaluation on both the BEV and the 3D detection benchmarks. Additionally, if the calibration between camera and lidar is known, cuboids can be projected onto the image, and the approach can be additionally assessed as a 2D detection method, including a predicted viewpoint value computed from the yaw by using Eq. 6.4.

In this thesis, the focus is on BEV detection performance. Nevertheless, Tables 6.8 and 6.9 show, respectively, the 2D and 3D stats obtained

CLASS	I	D	H	BEV DETECTION (AP BEV)		
				EASY	MOD.	HARD
CAR	✓	✓	✓	72.32	54.09	54.50
	✓	✗	✗	55.04	41.16	38.56
	✗	✓	✗	70.94	53.00	53.30
	✗	✗	✓	69.80	52.90	53.69
PED	✓	✓	✓	43.62	39.48	36.63
	✓	✗	✗	36.25	30.43	28.37
	✗	✓	✗	38.21	32.72	29.58
	✗	✗	✓	38.37	34.04	32.37
CYC	✓	✓	✓	47.44	31.26	30.57
	✓	✗	✗	33.09	22.83	21.79
	✗	✓	✗	43.77	28.62	26.99
	✗	✗	✓	48.06	31.21	30.40

Table 6.7: BEV detection performance (AP BEV %) of the proposed BEV detection approach on the KITTI validation subset using different data as an input [15] © 2018 IEEE

on the testing set of the KITTI benchmark for reference. Results were computed in the official evaluation server and are publicly available⁶.

CLASS	2D DETECTION (AP 2D)			2D ORIENTATION (AOS)		
	EASY	MOD.	HARD	EASY	MOD.	HARD
CAR	78.18	57.47	56.66	50.85	35.81	34.90
PED	36.83	30.90	29.93	21.34	17.26	16.67
CYC	64.88	49.04	46.61	41.48	30.76	28.66
mean	59.96	45.80	44.4	37.89	27.94	26.74

Table 6.8: 2D detection and viewpoint estimation performance (AP % and AOS %) of the proposed BEV detection approach on the KITTI testing set [15] © 2018 IEEE

As expected, cars are more reliably detected than pedestrians and cyclists due to their size. The three levels of difficulty defined by the KITTI dataset do not correspond well with the actual complexity in BEV detection, given that they are defined according to image features. That is why differences between levels are less noticeable than in image detection. 3D detection results and, to a lesser extent, BEV detection results, are profoundly affected by the high IoU overlapping required for true positives by KITTI criteria. In practice, lower IoU levels provide acceptable predictions that can be useful for vehicle navigation.

⁶ http://www.cvlibs.net/datasets/kitti/eval_object_detail.php?&result=62a5c7d933e853e8049c9975fa25d8749f258778

CLASS	3D DETECTION (AP 3D)			BEV DETECTION (AP BEV)		
	EASY	MOD.	HARD	EASY	MOD.	HARD
CAR	14.75	13.44	12.04	75.52	50.81	50.00
PED	14.31	11.80	10.55	26.07	21.35	19.96
CYC	18.35	12.43	11.88	38.93	27.18	25.51
mean	15.80	12.56	11.49	45.84	33.11	31.82

Table 6.9: 3D detection and BEV detection performance (AP 3D % and AP BEV %) of the proposed BEV detection approach on the KITTI testing set [15] © 2018 IEEE

This issue is further studied in Fig 6.13, where recall-IoU curves for the three categories are presented. As apparent from the figures, recall drops abruptly for high IoU values. However, this does not mean that objects are not identified; instead, the problem is that predicted cuboids do not fit perfectly the labels in the dataset, which might be a minor issue depending on the application.

SENSITIVITY TO
IOU THRESHOLDS

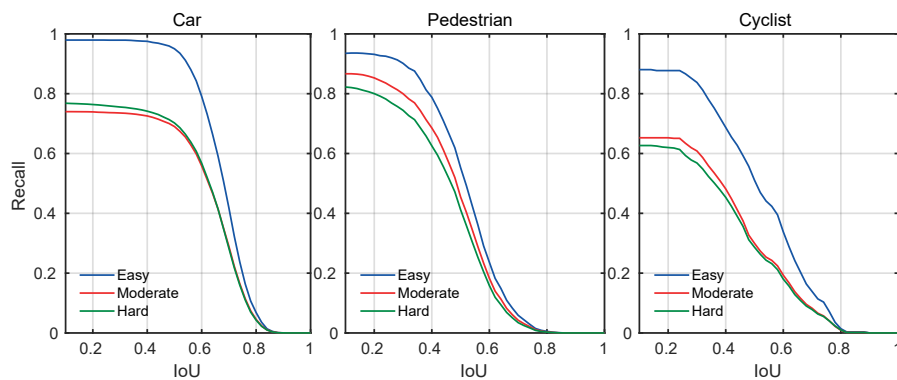


Figure 6.13: Recall of the proposed BEV detection approach on the KITTI validation subset at different IoU thresholds using 300 proposals [15] © 2018 IEEE

A fair comparison with other methods in the literature is mostly impractical due to two notable peculiarities of our method:

COMPARISON
WITH OTHER
METHODS

1. Most methods focus on the detection of cars, which is a more affordable task due to size constraints.
2. Typically, several sources of data, i. e., images and lidar, are combined to perform detection through sensor fusion. Instead, the proposed method performs inference from lidar data, exclusively.

Despite this, Table 6.10 establishes a comparison with other 3D detection methods in the KITTI dataset for *Car* detection. The minimum IoU has been set to 50%, which is adequate for the application. Results

come from their own published papers; fortunately, they all use the same validation subset (Chen et al. [46]) and provide results for IoU 0.5. The compared methods are: MV3D, using only visual information (BEV+FV) [46]; VeloFCN [163]; and PC-CNN [64], which fuses RGB and lidar data.

METHOD	3D DETECTION (AP 3D)			BEV DETECTION (AP BEV)			TIME (s)
	EASY	MOD.	HARD	EASY	MOD.	HARD	
MV(BV+FV)	95.74	88.57	88.13	86.18	77.32	76.33	0.24
VeloFCN	67.92	57.57	52.56	79.68	63.82	62.80	1
PC-CNN*	87.16	87.38	79.40	90.36	88.46	84.75	0.5
Proposed	88.92	67.56	68.59	90.43	71.45	71.34	0.11

* Fuses RGB and LiDAR data.

Table 6.10: Comparison of the BEV Car detection performance (AP 3D % and AP BEV %) of the proposed BEV detection approach with other methods on the KITTI validation subset with IoU 0.5 [15] © 2018 IEEE

Detection results of the proposed method are competitive; it improves VeloFCN by a large margin in both BEV and 3D detection and is on par with the other two methods. It is noteworthy that it achieves these results requiring much lower processing time, and providing detections for all the available categories, which are unique qualities of the approach that make it suitable for onboard applications.

Some qualitative examples of the performance of the method are shown in Fig. 6.14. The run time per KITTI frame using the current implementation (see Sec. 4.2.3.1) is around 110 ms using an NVIDIA Titan Xp GPU. The forward-pass time is highly dependent on the cell size and the detection range; both have been chosen to favor accuracy against speed in this work, so it is expected that computation time could be further reduced depending on the requirements of the application.

6.3 CONCLUSION

This chapter has been aimed to close the gap between the perception suite and the planning and control modules by endowing the already available set of detections with spatial reasoning. This is a necessary step to achieve a proper situational awareness that allows the automated vehicle to navigate in all kinds of traffic environments.

Two alternative approaches have been studied: one intended to complement the object detection and viewpoint estimation method introduced in the preceding chapters, and another aimed to replace the functionality of the entire pipeline. The former takes advantage of stereo information, whereas the former uses lidar data.

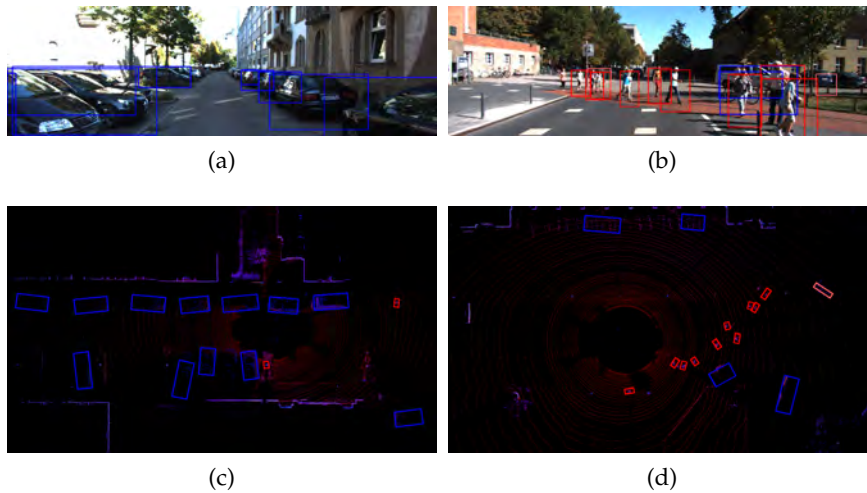


Figure 6.14: Example of 3D detection results on scenes from the KITTI testing set using the proposed method [15] © 2018 IEEE

The first method has proven effective despite the limitations of stereo matching, which affect mainly to the accuracy at long distances. A complete model of the traffic scene, including the position of all road users with respect to the vehicle and the free space ahead of it, can be obtained through this approach.

On the other hand, the lidar-based method has emerged as a viable alternative to provide 3D detections based solely on lidar data, thus serving as a redundant method which can perform in parallel to the main detection pipeline. BirdNet, which is the name given to this approach, was one of the first methods in the KITTI object detection benchmark capable of providing results for all the categories⁷ and has attracted significant interest in the related literature, receiving 17 citations from reputable authors (e. g., [284]) in less than one year after publication⁸.

This chapter proves that the methods proposed in this thesis are aligned with the current trends in onboard perception research, which is geared lately towards robust and accurate 3D object detection approaches

⁷ http://www.cvlibs.net/datasets/kitti/eval_object_detail.php?&result=62a5c7d933e853e8049c9975fa25d8749f258778

⁸ Citation count provided by Google Scholar. There were no self-citations.

Part III

CONCLUDING REMARKS

CONCLUSION AND FUTURE WORK

The achievements made in this thesis and their relevance within the topics addressed are summarized in this chapter. Besides, some future research endeavors aimed to lend continuity to the lines of inquiry started in this work are also suggested.

7.1 CONCLUSION

The perception suite of an automated vehicle should be able to provide the planning and navigation modules with situational awareness of traffic scenes. A proper environment model must take into consideration the rest of the road users around the vehicle and their dynamics. Within this general goal, this thesis has focused on the identification and localization of objects from onboard sensor data. Different detection and pose estimation approaches have been proposed taking advantage of modern [DNNs](#), which reach high levels of accuracy thanks to their feature learning capabilities.

The most important aspects to be considered in the selection of the sensor suite have been discussed, but the work has not been limited to a single modality. Instead, it has dealt with different sources of data, handling the particularities of each one through convenient data representations.

Hence, a significant number of approaches have been proposed and validated throughout this document. The main contributions are summarized below:

- A method to extract features from stereo vision data, and employ them in an automatic extrinsic calibration framework, has been introduced. The resulting calibration approach largely eliminates the need for human intervention in the procedure of obtaining the transform that relates a stereo camera and a multi-layer lidar device. Both sensors are typically part of the sensor setup in automated vehicles, and a precise calibration between them is crucial to perform data association. A novel evaluation procedure based on synthetic data has proven that the accuracy of the proposed calibration method exceeds that of existing methods.
- The applicability of a modern, [DNN](#)-based object detector ([Faster R-CNN](#)) to onboard obstacle identification has been demonstrated. Several fine-tuning measures have been proposed to improve the performance of the detection framework in the context of this particular application. The detector uses color images as the sole

input and does not make any bold assumption, thus offering superior robustness against different variability sources.

- A method to take advantage of stereo data to complement intensity information in object detection has been proposed. The approach has been shown to improve the detection accuracy of Faster R-CNN without the need for severe structural changes and with a minor effect on the computational burden. The method is suitable for enhancing the detector capabilities when stereo information is available.
- An approach to performing joint detection and heading estimation based on the Faster R-CNN detection framework has been presented. The intuition behind the method is that both detection and orientation estimation can be performed from the same set of appearance features. This idea has been validated through plentiful experimentation. As a result, the proposed multi-task approach is effective but also efficient and provides valuable information about the position and intentions of other road users.
- Different combinations of model alternatives, hyperparameter settings, and design choices have been thoroughly tested to analyze their effect on the speed and accuracy of the method discussed in the previous point. The results allow to conveniently modify the speed-accuracy trade-off according to the requirements of the particular application. Additionally, the sensitivity of the method to the size and diversity of training data has also been investigated, suggesting that small increases in the number of training samples might lead to significant improvements in performance.
- A method to localize the detected objects in 3D space using stereo vision information has been proposed. The approach makes use of the orientation estimation provided by the multi-task framework from the previous points to estimate a cuboid representing the geometry and location of each object with respect to the ego vehicle. Despite the inherent limitations of stereo matching, the method has been proven adequate for object-based modeling of traffic scenes.
- A standalone approach to perform 3D object detection in lidar data has been introduced. The method takes advantage of the proposed detection and orientation estimation framework to process lidar data represented as an image-like structure. Due to the particularities of lidar scanners, the proposal can perform detection on a wide 360° range around the vehicle.

All these contributions are inextricably linked as parts of a whole: an onboard perception suite intended to build an object-based model

with enhanced information about the dynamic objects in the scene. The investigation conducted in this thesis falls within the topic of traffic scene understanding, which is one of the most critical challenges faced by upcoming autonomous vehicles.

7.2 FUTURE WORK

The set of approaches introduced in this thesis, albeit aimed at a common end, does not intend to offer a comprehensive perception stack. The application of this methodology to a real automotive platform is subject to the presence of complementary modules and integration efforts that fall outside the scope of this work. Among them, it is worth highlighting the following:

- The set of detections from the presented approaches would benefit hugely from a subsequent *tracking* stage [99]. This component is widely used in ITS applications to increase the robustness of detections by introducing time as a variable in the reasoning process. This way, the correlation between consecutive frames is exploited to make predictions, which makes it possible to filter inference errors made by single-frame detectors. The topic has been extensively addressed in the literature, giving rise to a manifold of methods ranging from basic Kalman filters [87] to very sophisticated approaches [262]. Furthermore, the orientation estimation method presented in this thesis might aid the movement prediction, thus obtaining additional benefit from the tracking phase.
- The implementation of the algorithms presented in this thesis was geared towards experimentation, and therefore, ease of operation and versatility took precedence over efficiency. Implementations tailored to automotive hardware should make it possible to achieve higher framerates with lower energy consumption, applying little or no modification to the presented models. For instance, most embedded platforms for deep learning inference are optimized for half-precision float arithmetic (FP16) or even eight-bit computations (INT8). Recent works have shown that comparable accuracy levels can be achieved under these low-precision setups [183], thus extending the validity of the conclusions reached in this work to embedded inference.
- Although all the proposed methods have been designed to be virtually agnostic to the characteristics of the sensor device in use, experimentation has been limited to a restricted number of setups. Given the growing availability of camera and lidar scanner devices, the adequacy of the proposed approaches for every specific setup should be confirmed on a case-by-case basis.

- Industrial-level safety standards (e.g., ASIL-D, ISO 26262, etc.) impose strict verification requirements which are challenging to meet for DNNs [152]. Advances are being made in the matter [146], but this fundamental issue requires an in-depth analysis which is out of the scope of this thesis.
- Putting the developed algorithms into operation would necessarily involve the participation of other complementary modules in the perception stack, so that fundamental tasks such as traffic signaling detection, traversable space estimation, and ego-localization [122] are fulfilled. Some of these tasks will be addressed in Annex A.

Regarding the core topics of this thesis, several improvements can be carried out to enhance and extend the introduced proposals. Some of them are listed below:

- The exceptionally rapid progress in DNNs has led to the emergence of multiple alternatives in the literature to improve each of the stages of the detection framework. Examples include, but are not limited to, the *focal loss* proposed in [168] to mitigate class imbalance at training time, and Soft-NMS [25], which aims to improve the duplicate filtering at inference time. These and other structural modifications were discussed in Sec. 2.5 and, along with the use of recent models for the feature extractor part, are expected to lead to significant improvements in the performance of the proposed methods, as hinted in Sec. 5.4.2.
- The link between detection and localization according to the methods proposed in this thesis takes place at a high level. Alternatives for a low-level fusion of appearance and geometric information can be explored as a potential source of improvement for 3D detection, as suggested by the results in Sec. 4.3.
- A low-level fusion scheme such as the one proposed in the previous point would make all the sensor data available before, or during, feature extraction. That configuration would enable embedding 3D localization into the Faster R-CNN framework as another task, resulting in an end-to-end 3D detection scheme similar to the ones that have become popular in recent times (see Sec. 2.6.3).
- Inversely, the current dependency on prior geometrical data could be dropped by introducing depth estimation into the framework. Instead of explicitly feeding the network with spatial information, stereo matching could be embedded in the network architecture, which would then accept the stereo pair as input. Alternatively, 3D reconstruction techniques from monocular images [43], [156] could be applied, avoiding the need for additional sources of data.

- It is a well-known fact that detection can benefit from information about the scene geometry and the camera position [125]. Thus, the use of additional mid-level cues within the proposed multi-task paradigm could be explored as a way of taking advantage of the (loose) structure of traffic environments. As an example, road users are likely to be found on the road surface, which is a fact that could be exploited by the system to reduce the false-positive rate.

There are reasons to believe that traffic scene understanding will be an active research topic in the coming decades. AI and DNNs should be the driving forces to achieve the robustness and accuracy goals imposed by autonomous vehicles. This work is, ultimately, a modest contribution towards that objective, which will undoubtedly require a Herculean research effort to be reached.

Part IV

APPENDIX

As has been repeatedly made clear during this dissertation, object detection and localization is paramount for the automated navigation of a vehicle. However, as natural, that application solely does not cover the entire range of functionalities that the onboard perception stack is required to provide.

Classical applications that fall outside the scope of object detection include road (or lane) detection and traffic light classification, for instance, whose functionality is also critical for autonomous driving.

Recently, semantic segmentation has gained traction as a *catch-all* alternative thanks to the fine-grained knowledge provided by its pixel-wise classification capabilities. Theoretically, semantic segmentation has the potential to replace several onboard applications with a one-shot approach [222].

Instance segmentation [118] and, more recently, panoptic segmentation [148] aim to combine the pixel-wise estimation provided by semantic segmentation with the instance-level classification from object detectors.

In this annex, some applications developed in the context of the thesis, but which fall outside the scope of its main topic, are briefly presented. All of them are oriented towards the common objective of providing information to improve the understanding of the traffic environment, thus complementing the main developments proposed in the thesis.

A.1 LANE DETECTION AND CLASSIFICATION

An algorithm for lane detection and classification was proposed in [219] based on classical computer vision techniques¹. The procedure performs three different tasks: it finds lines in the image, infers the most probable lanes given those lines, and classify the lines according to their traversability.

The information is extracted from a stereo pair. The pipeline is composed of the following steps:

1. Bird's Eye View (BEV) perspective transform: As usual in lane detection, the image is transformed to a BEV perspective so that lines defining a road lane are represented parallel in the image.

This annex includes content from [219], [57], [76] and [11].

¹ The work described in this section was mostly developed by Dr. César H. Rodríguez during his Ph.D. The author of this thesis was responsible for its implementation in ROS.

OVERVIEW

PROCEDURE

Note that this BEV is not equivalent to the one described before in this document for lidar data; instead, this one is obtained by a homographic transformation which gives, as an output, the image that would be obtained by a *virtual camera* placed above the road and looking down to it. An example is shown in Fig. A.1. The homography is given by the extrinsic parameters of the camera with respect to the ground and is therefore obtained through an auto-calibration method similar to the one presented in Sec. 6.1.1.

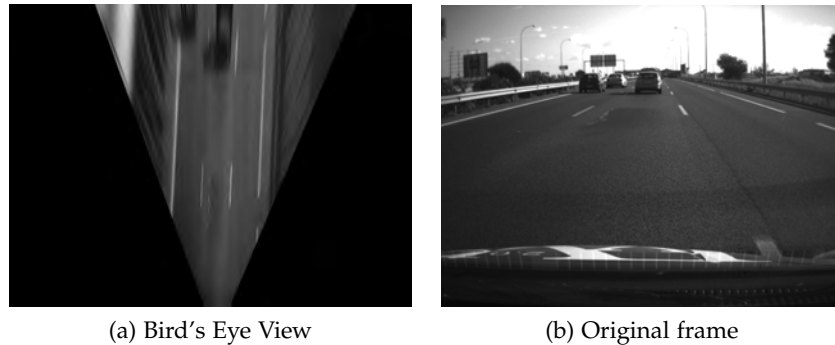


Figure A.1: Bird's Eye View (BEV) for lane detection [219]

2. Road segmentation: Pixels whose 3D positions lie outside a certain distance from the detected plane are removed, so only pixels belonging to the road are taken into account for subsequent processing.
3. Road markings mask: A mask of candidate line pixels is created by computing gradients in the image and keeping only points meeting some heuristic criteria.
4. Line detection: the Hough transform is used to detect straight lines in the BEV image, as shown in Fig. A.2a.
5. Lane identification: The set of Hough lines is filtered to keep only lines meeting geometrical constraints. All the resulting lines follow the same direction and are separated by a distance equivalent to the width of a lane. Then, each pair of adjacent lines define a lane. An example is shown in Fig. A.2b.
6. Line classification: A set of descriptors is extracted from each line, describing its mean intensity value, its length, and its frequency peaks. Lines are then classified according to a set of heuristic rules into one of the following classes: *solid*, *dashed*, *merging*, and *unknown*.
7. Temporal consistency: Lanes are tracked through several frames to increase the robustness against occlusions and noise. To that

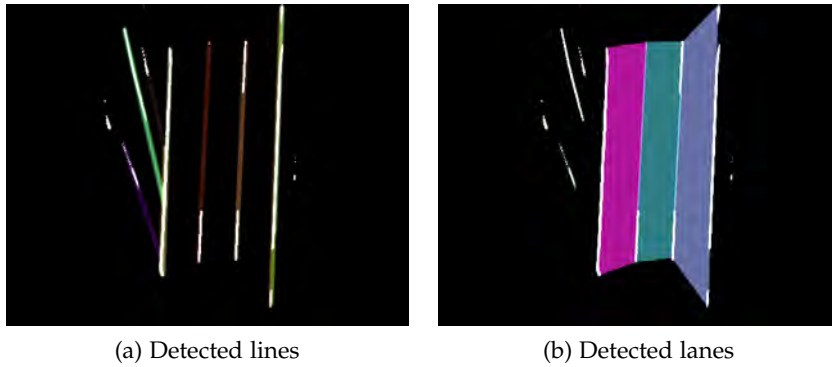


Figure A.2: Example of line and lane detections using the proposed approach

end, lines are assigned an *energy* value that decays over time if they are no longer detected. Likewise, a line is required to be detected over several frames to validate the detection.

Tests on a custom dataset showed precision values of around 88% for lane detection and 85% for line classification [216]. Some qualitative results are shown in Fig. 7.

RESULTS

A.2 ROAD SIGNALING CLASSIFICATION

Traffic Sign Recognition (TSR) and Traffic Light Recognition (TLR) are two critical applications for enabling the use of current infrastructures by autonomous vehicles. As road signaling is targeted to human drivers, who perceive information through the sense of sight, cameras are nearly always used. In this section, two methods are presented, each of them dealing with one of the two problems.

A.2.1 Detection

The approaches described in this section are aimed at the classification of previously detected candidate regions. Thus, they serve the same purpose as the *classification head* in Faster R-CNN.

Naturally, in the previous step of the pipeline, an RPN could be used to generate proposals. However, due to the particularities of these objects, other options are also available. For instance, a semantic segmentation network could be trained on a dataset with these categories to provide the areas of the image where a sign or a light is likely to be present. This way, a dedicated proposal network will not be necessary, and semantic segmentation could also fulfill other functions (e.g., free space detection).

On the other hand, both signs and lights are, differently from cars or pedestrians, static objects belonging to the infrastructure. It is possible

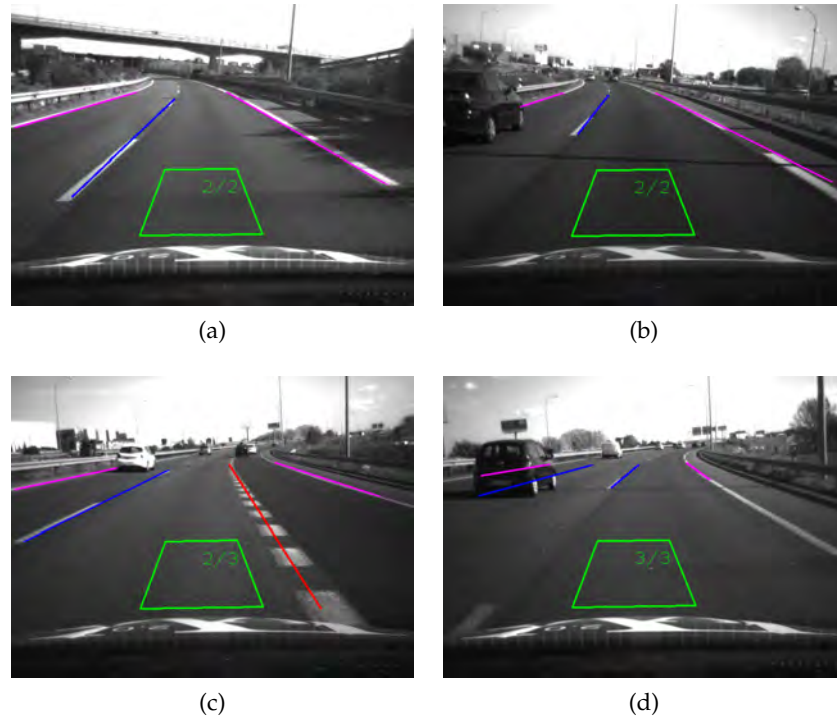


Figure A.3: Examples of lane detection and classification results using the proposed approach. Color code: pink for *solid*, blue for *dashed* and red for *merging*

to take advantage of this fact by using maps provided with both entities to determine the areas of the image that should be classified.

A.2.2 Traffic sign classification

A traffic sign detection framework was proposed in [57]². Proposal generation was based on color, taking advantage of the unique hue values featured by traffic signs. However, the most relevant results were from the classification step, which validated the adequacy of CNNs to perform traffic sign classification.

Experiments were performed using three different architectures (sorted by increasing complexity): LeNet [160], AlexNet [154], and GoogLeNet [254].

Tests used the German Traffic Sign Recognition Benchmark (GTSRB) [249], a renowned dataset that comprises 43 classes typically found in European roads. A 90:10 train-validation split was performed on the dataset.

Images are resized to 28×28 (in grayscale) for the LeNet network, and to 256×256 (in color) for AlexNet and GoogLeNet. Results in

² This work was developed by Lidia Díaz during a TFG (Bachelor's Thesis) supervised by the author of this thesis [57]

Table A.1 prove the high potential of CNNs to perform reliable traffic sign classification.

LeNet	AlexNet	GoogLeNet
98.90	98.39	99.05

Table A.1: Classification performance (accuracy %) of three architectures tested for traffic sign classification [57]

A.2.3 Traffic light classification

Traffic light status recognition might seem a simple task; however, the variety of appearances and typologies of these elements pose a challenge to onboard perception systems. In this regard, the data used for training have a significant influence on the performance of the resulting model.

In [76], a detailed study of the public datasets aimed at traffic light recognition is presented³. Besides, different CNNs were applied to perform traffic light status classification. The aim was twofold: firstly, to identify the datasets which, when used for training, allow obtaining the best performance; secondly, to test the accuracy of two CNN models for multi-state classification.

In this section, the experimental procedure for the evaluation of the CNN models is introduced, along with the obtained results. Five datasets were used (sorted by publication date):

- LARA traffic light recognition benchmark [54].
- WPI Traffic Light Dataset [47].
- LISA Traffic Light Dataset [140]. For purposes of analysis, day-light images (LISA-D) are separated from the nighttime ones (LISA-N).
- Bosch Small Traffic Lights Dataset (BSTLD) [12].

Additionally, some samples from the Cityscapes dataset [49] were annotated. This set is employed in combination with the others to generate a meta-dataset composed of all the labeled samples together (ALL).

As stated before, two CNN models were employed:

1. A custom ResNet network, made of only two residual blocks. Each block contains two 3×3 convolution layers (plus their respective non-linearities). The first block has 16 channels and the second, 32.

³ This work was co-authored by the author of this thesis during his research visit at the KIT (Germany) under the supervision of Prof. Christoph Stiller and with the guidance of Dr. Carlos Fernández.

PROBLEM
STATEMENT

DATASETS

MODELS

2. MobileNet v2 [234]: The standard model, with a width multiplier of 1, was used.

METRICS

The mean F1-score (mF_1) across the N evaluated categories is used to allow easy comparison between alternative cases. Contrary to raw accuracy, this metric assigns the same importance to each category, including the underrepresented *yellow*:

$$mF_1 = \frac{1}{N} \sum_{i=1}^N 2 \frac{P_i R_i}{P_i + R_i} \quad (\text{A.1})$$

EXPERIMENTAL
SETUP

Traffic light crops from all sources are resized to 64×32 before entering the CNN. Yellow samples are weighted with a factor of 3 in the classification loss due to the low number of samples available. All datasets were split into training and validation subsets with a ratio of 70 : 30. In order to increase the significance of the results, three different training procedures were performed for each configuration, and the best result is provided.

TRAINING/VAL.
COMBINATIONS

First of all, results for the custom ResNet model for the different combinations of train and validation datasets are shown in Table A.2, including the dataset composed of the mixture of all the others (ALL). In this case, only color classification is evaluated (i.e., classes are *red*, *yellow*, and *green*). Note that this experiment was aimed to show the ability of each dataset to train robust models, on the one hand, and to assess the real performance of the methods, on the other hand. Also, note that WPI results are not directly comparable with the others since it lacks *yellow* labels.

		TRAINING					
		BSTLD	LARA	LISA-D	LISA-N	WPI*	ALL
TESTING	BSTLD	85.03	71.95	77.64	70.83	90.73	84.84
	LARA	98.35	99.95	95.10	99.04	99.69	99.95
	LISA-D	94.50	87.81	99.82	86.31	87.58	99.68
	LISA-N	98.40	94.85	96.19	99.47	99.72	99.62
	WPI*	99.61	98.90	99.61	96.38	100.00	100.00
	ALL	92.76	83.86	88.28	86.29	92.77	96.23

* WPI is evaluated only on the available *red* and *yellow* classes.

Table A.2: Classification performance (mean F1 score) on different validation sets of traffic light color classification using the custom ResNet model [76] © 2018 IEEE

Obviously, the models perform better when evaluated on the same dataset used for training. Apart from this, BSTLD arises as a challenging dataset able to generate all-terrain models. Evaluation on LISA-N does not yield reliable results due to the prominence of traffic light bulbs in nighttime conditions. Overall, the model trained with the

combined dataset obtained the best results when evaluated on every other dataset, except for BSTLD.

Some data augmentation techniques were tested to improve the performance of the models:

1. Horizontal flip.
2. Random affine transformations (to simulate different points of view).
3. Saturation and value (HSV color space) jittering.

The effect of this augmentation, together with a comparison of both models, for color classification is investigated in Table A.3. Again, different models trained in each dataset are considered. Overall, the performance of MobileNet is slightly better than that of the custom ResNet, although its complexity is also higher. On the other hand, augmentation has a small positive effect on the results.

AUG:	MOBILENET		RESNET	
	✗	✓	✗	✓
BSTLD	88.3	92.3	92.8	92.9
LARA	74.8	88.8	83.9	88.8
LISA-D	91.0	84.0	88.3	93.3
LISA-N	74.7	60.2	86.3	90.6
WPI*	84.1	92.5	92.8	97.3
ALL	99.7	99.8	96.2	96.3

* WPI is evaluated only on the available *red* and *yellow* classes.

Table A.3: Classification performance (F1-score) of traffic light color classification on the combined dataset for the different training sets and models with and without data augmentation (AUG) [76] © 2018 IEEE

Finally, arrow traffic signs are differentiated, and the problem is posed as a 6-class classification into *green*, *yellow*, *red*, *left green*, *left yellow*, and *left red*. Only left turns were considered since they are the most frequent in the available datasets and also in right-hand-drive traffic environments⁴

In this case, due to the reduced number of arrow samples, an additional factor of 3 is applied to the loss corresponding to these categories. Confusion matrices, one for each model, are shown in Tables A.4 and A.5. The combined dataset has been used for both training and testing.

There are a significant number of samples that were misclassified regarding the existence of an arrow; however, most of them were

⁴ Note that, when a green circle is on, it usually allows to go straight and to turn right. However, turn left is not allowed due to oncoming traffic.

		PREDICTED						REC. (%)	
		●	●	●	←	←	←		
		9069	562	10572	3453	104	824		
GROUND TRUTH	●	9569	8906	75	32	553	0	3	93.1
	●	523	30	467	3	10	13	0	89.3
	●	10710	16	7	10463	7	0	217	97.7
	←	3012	117	7	2	2878	1	7	95.6
	←	96	0	6	0	0	89	1	92.7
	←	674	0	0	72	5	1	596	88.4
PRECISION (%)		98.2	83.1	99.0	83.3	85.6	72.3	91.1*	

* Mean F1-score between classes.

Table A.4: Confusion matrix of the MobileNet network for traffic light classification trained and tested on the combined dataset using six labels [76] © 2018 IEEE

assigned to the correct color. Again, the performance of the MobileNet model is slightly better than that of the ResNet. However, the run time per frame of the former is 7 ms (14 ms on CPU), whereas it increases to 24 ms (29 ms on CPU) for the latter.

CONCLUSION

Results prove the benefits of combining different datasets to train a traffic light classification CNNs. Additionally, data augmentation has been shown to improve the model capabilities further. Finally, the feasibility of applying a specific traffic light classification network on top of a previous proposal method has been demonstrated.

A.3 SEMANTIC SEGMENTATION

PROBLEM STATEMENT

Semantic segmentation aims to provide a label to each pixel in the image. The past few years have seen a flood of research works dealing with this task [85], which gives valuable information for autonomous driving.













In [11], a comparative study of four semantic segmentation methods is presented, along with an analysis of relevant hyperparameters and setup options that influence their performance⁵.

METHODS

The methods under study are listed below:

FULLY CONVOLUTIONAL NETWORK (FCN) [172]: FCN was the first DNN-based method proposed for semantic segmentation. Its encoder-decoder architecture, where feature maps are extracted from the images and then upsampled to provide the final estimation, is the basis of most of the subsequent approaches. Transpose convolutions are used to carry out the upsampling,

⁵ This work was developed by Alejandro Barrera during a TFM (Master's Thesis) supervised by the author of this thesis [9].

		PREDICTED						REC. (%)	
									
		9069	562	10572	3453	104	824		
GROUND TRUTH		9569	9069	79	18	398	1	4	95.1
		523	31	469	4	9	10	0	89.7
		10710	17	6	10467	6	1	213	97.7
		3012	144	4	3	2857	3	1	94.9
		96	0	8	0	0	88	0	91.7
		674	4	0	52	2	0	616	91.4
PRECISION (%)		97.9	82.9	99.3	87.3	85.4	73.9	91.1*	

* Mean F1-score between classes.

Table A.5: Confusion matrix of the ResNet model for traffic light classification trained and tested on the combined dataset using six labels [76] © 2018 IEEE

whereas *skips* are employed to add low-level features at middle stages of the decoding procedure. Different alternatives were proposed; the FCN-8s all-at-once is used in this section.

BAYESIAN SEGNET [5]: Similar to the previous one, this architecture features an encoder-decoder architecture (SegNet [5]); however, they introduce dropout layers to estimate the uncertainty of the estimation.

U-NET [224]: Originally intended for biomedical image processing, it concatenates features at different levels to propagate low-level information to the final layers.

ERFNET [220]: It introduces a novel block, *Non-bottleneck-1D*, to reduce the computation burden while achieving high accuracy. The design was geared towards onboard systems.

The cited architectures were reimplemented on a common framework, and tested under the same conditions, to enable a fair comparison. The Cityscapes dataset [49] was employed for training and testing. Regarding the hyperparameters analysis, the following factors were analyzed:

- Class imbalance: Weighting the different categories according to their frequency of occurrence in the training set has a positive effect on the performance.
- Transfer learning: When possible, a model pre-trained for recognition on ImageNet can be used to initialize the weights of the encoder part. Despite the difference between tasks, initial features have been shown beneficial for semantic segmentation models.

- Preprocessing and data augmentation: The following data augmentation techniques were evaluated: Gaussian jitter, horizontal flip, and random crop. Overall, they increase the accuracy and robustness of the trained models.
- Input scale. Increasing the size of input images has a dramatic effect on the final accuracy, at the expense of higher computation times.
- Number of classes. Coarse-grained categories (e.g., vehicle), which can be enough to represent the environment in specific situations, lead to better results than fine-grained classes (e.g., car, van, or truck).

Tests were performed using the ERFNet architecture. Results, in terms of the widely-used mean **IoU** stat [172], are shown in Table A.6.

CL.B.	TR.L.	AUG	HD	MIOU
✗	✗	✗	✗	40.23
✓	✗	✗	✗	40.42
✓	✓	✗	✗	43.51
✓	✗	✓	✗	42.14
✓	✗	✗	✓	45.43

Table A.6: Semantic segmentation performance (mean **IoU** %) of ERFNet for different configurations: class balancing (CL. B.), transfer learning (TR. L.), data augmentation (AUG), and high resolution (HD). Table by Barrera et al. from [11] (license CC-BY-NC)

COMPARISON AMONG MODELS

On the other hand, Table A.7 reports a comparison between the studied architectures using different evaluation metrics. Run times per frame are also included. Overall, the ERFNet architecture achieves results on par with the other methods with much lower processing time. Some examples of the results obtained with the analyzed methods are depicted in Fig. A.4, using a frame obtained from the LSI’s IVVI 2.0 platform as an input.

MODEL	OV. ACC.	M REC.	M ACC.	M IOU	F. W. IOU	TIME (ms)
FCN	83.51	50.44	70.57	43.27	76.11	170
B. SegNet	84.00	54.58	68.84	44.25	76.77	182
U-Net	82.81	49.94	62.80	41.31	75.52	135
ERFNet	85.14	51.79	67.10	43.34	78.21	61

Table A.7: Comparison of the performance of the studied semantic segmentation methods on the Cityscapes validation set. Stats: overall accuracy (OV. ACC.), mean recall (M REC.), mean accuracy (M ACC.), mean **IoU** (M IOU), and frequency-weighted **IoU** (F. W. IOU). Table by Barrera et al. from [11] (license CC-BY-NC)

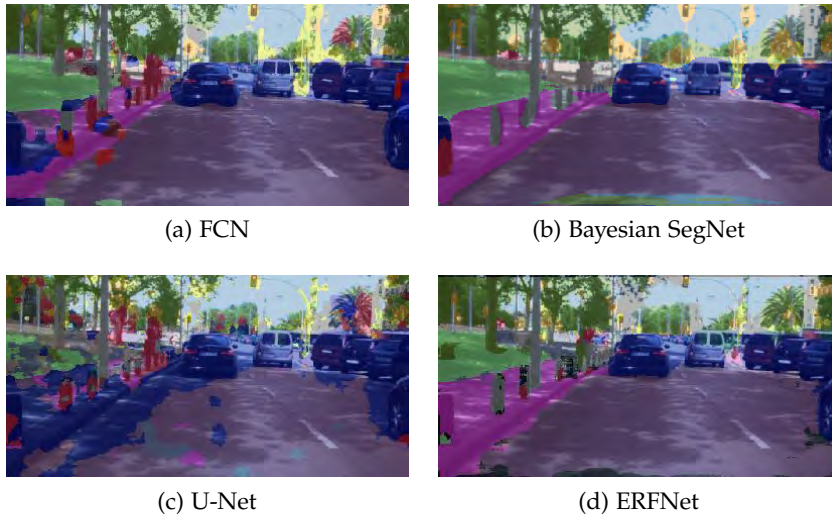


Figure A.4: Examples of semantic segmentation results for each of the studied models. Figures by Barrera et al. from [11] (license CC-BY-NC)

In short, semantic segmentation has shown enormous potential to help to understand the traffic scene. However, as already discussed in Sec. 4.5, it will not probably offer an all-in-one solution to all the traffic problems. Instead, it is likely that the vehicular perception systems will evolve in a way that they will make the most of the combined use of object detection and semantic segmentation methods. The former is suitable to generate an object-based model of the environment where dynamic obstacles (i. e., *things*) are present. On the other hand, the latter is more appropriate to provide information about the infrastructure (i. e., *stuff*). The task aimed at combining both, named *panoptic segmentation* [148], is destined to become an important line of research over the next few years.

CONCLUDING
REMARKS

BIBLIOGRAPHY

- [1] S. Agarwal, J. O. D. Terrail, and F. Jurie, "Recent advances in object detection in the age of deep convolutional neural networks," *arXiv:1809.03193 [cs.CV]*, 2018.
- [2] H. A. Alhaja, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother, "Augmented reality meets deep learning for car instance segmentation in urban scenes," in *Proc. British Machine Vision Conference (BMVC)*, 2017.
- [3] —, "Augmented reality meets computer vision," *International Journal of Computer Vision*, vol. 126, no. 9, pp. 961–972, 2018.
- [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*, vol. 20, no. 10, pp. 3782–3795, 2019.
- [5] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.
- [6] I. Barredo, "Entrenamiento de algoritmos de deep learning para la detección de objetos con la base de datos cityscapes," Bachelor's Thesis (TFG), Universidad Carlos III de Madrid, Jul. 2017.
- [9] A. Barrera, "Estudio de algoritmos de segmentación semántica basados en deep learning para su aplicación en vehículos inteligentes," Master's Thesis (TFM), Universidad Carlos III de Madrid, Sep. 2018.
- [11] A. Barrera, C. Guindel, F. García, and D. Martín, "Análisis, evaluación e implementación de algoritmos de segmentación semántica para su aplicación en vehículos inteligentes," in *XXXIX Jornadas de Automática*, 2018, pp. 983–990.
- [12] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1370–1377.
- [15] 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.

- [16] J. Beltrán, C. Jaraquemada, B. Musleh, A. de la Escalera, and J. M. Armingol, "Dense semantic stereo labelling architecture for in-campus navigation," in *Proc. International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP) - Volume 5: VISAPP*, 2017, pp. 266–273.
- [17] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.
- [18] K. Bengler, K. Dietmayer, B. Färber, 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.
- [19] M. Bertozzi, A. Broggi, M. del Rose, M. Felisa, A. Rakotomamonjy, and F. Suard, "A pedestrian detector using histograms of oriented gradients and a support vector machine classifier," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2007, pp. 143–148.
- [20] M. Bertozzi, A. Broggi, and A. Fascioli, "Vision-based intelligent vehicles: State of the art and perspectives," *Robotics and Autonomous Systems*, vol. 32, no. 1, pp. 1–16, 2000.
- [21] P. J. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, 1992.
- [22] D. Biedermann, M. Ochs, and R. Mester, "COngRATS: Realistic simulation of traffic sequences for autonomous driving," in *Proc. International Conference on Image and Vision Computing New Zealand (IVCNZ)*, 2015.
- [23] —, "Evaluating visual ADAS components on the COngRATS dataset," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 1067–1072.
- [24] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.
- [25] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS – improving object detection with one line of code," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 5562–5570.
- [26] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Really, 2008.
- [27] D. Braess, A. Nagurney, and T. Wakolbinger, "On a paradox of traffic planning," *Transportation science*, vol. 39, no. 4, pp. 446–450, 2005.

- [28] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila, "EuroCity Persons: A novel benchmark for person detection in traffic scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 41, no. 8, pp. 1844–1861, 2019.
- [29] M. Braun, Qing Rao, Y. Wang, and F. Flohr, "Pose-RCNN: Joint object detection and pose estimation using 3D object proposals," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2016, pp. 1546–1551.
- [30] A. Broggi, M. Buzzoni, S. Debattisti, P. Grisleri, M. C. Laghi, P. Medici, and P. Versari, "Extensive tests of autonomous driving technologies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1403–1415, 2013.
- [31] A. Broggi, A. Cappalunga, C. Caraffi, S. Cattani, S. Ghidoni, P. Grisleri, P. P. Porta, M. Posterli, and P. Zani, "TerraMax vision at the Urban Challenge 2007," *IEEE Transactions on Intelligent Transportation Systems*, vol. 11, no. 1, pp. 194–205, 2010.
- [32] A. Broggi, A. Cappalunga, S. Cattani, and P. Zani, "Lateral vehicles detection using monocular high resolution cameras on TerraMax," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2008, pp. 1143–1148.
- [33] A. Broggi, C. Caraffi, P. P. Porta, and P. Zani, "The single frame stereo vision system for reliable obstacle detection used during the 2005 DARPA Grand Challenge on TerraMax," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2006, pp. 745–752.
- [34] A. Broggi, P. Cerri, S. Debattisti, M. C. Laghi, P. Medici, M. Panciroli, and A. Prioletti, "PROUD—public road urban driverless test: Architecture and results," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2014, pp. 648–654.
- [35] A. Broggi, P. Grisleri, and P. Zani, "Sensors technologies for intelligent vehicles perception systems: A comparison between vision and 3D-lidar," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2013, pp. 887–892.
- [37] 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," *arXiv:1903.11027 [cs.LG]*, 2019.
- [38] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos, "A unified multi-scale deep convolutional neural network for fast object detection," in *Computer Vision - ECCV 2016 (LNCS, vol. 9908)*, 2016, pp. 354–370.

- [39] E. Cano, "Solo el 10 por ciento de los accidentes responden a fallos técnicos," *ABC*, Mar. 14, 2016. [Online]. Available: https://www.abc.es/motor/reportajes/abci-causas-accidentes-trafico-solo-10-ciento-accidentes-responden-fallos-tecnicos-201603140141_noticia.html (visited on 11/15/2018).
- [40] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, 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.
- [41] Y. Chen, V. Sundareswaran, C. Anderson, A. Broggi, P. Grisleri, P. P. Porta, P. Zani, and J. Beck, "TerraMax: Team Oshkosh urban robot," *Journal of Field Robotics*, vol. 25, no. 10, pp. 841–860, 2008.
- [42] Q. Chen and V. Koltun, "Photographic image synthesis with cascaded refinement networks," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 1520–1529.
- [43] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun, "Monocular 3D object detection for autonomous driving," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2147–2156.
- [44] X. Chen, K. Kundu, Y. Zhu, A. Berneshawi, H. Ma, S. Fidler, and R. Urtasun, "3D object proposals for accurate object class detection," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 424–432.
- [45] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun, "3D object proposals using stereo imagery for accurate object class detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 5, pp. 1259–1272, 2018.
- [46] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3D object detection network for autonomous driving," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 6526–6534.
- [47] Z. Chen, Q. Shi, and X. Huang, "Automatic detection of traffic lights using support vector machine," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2015, pp. 37–40.
- [48] K. Chirantana and S. S. Kanth, "Collision warning with automatic braking system for electric cars," *International Journal of Mechanical Engineering Research*, vol. 5, no. 2, pp. 153–165, 2015.
- [49] 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.

- [50] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.
- [51] H. Dahlkamp, A. Kaehler, D. Stavens, S. Thrun, and G. Bradski, "Self-supervised monocular road detection in desert terrain," in *Proc. Robotics Science and Systems (RSS)*, 2006.
- [52] J. Dai, Y. Li, K. He, and J. Sun, "R-FCN: Object detection via region-based fully convolutional networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2016, pp. 379–387.
- [53] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2005, pp. 886–893.
- [54] R. De Charette and F. Nashashibi, "Traffic light recognition using image processing compared to learning processes," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2009, pp. 333–338.
- [55] S. Debattisti, L. Mazzei, and M. Panciroli, "Automated extrinsic laser and camera inter-calibration using triangular targets," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 696–701.
- [57] L. Díaz, "Sistema de detección y clasificación de señales de tráfico basado en deep learning," Bachelor's Thesis (TFG), Universidad Carlos III de Madrid, Sep. 2017.
- [58] K. Dietmayer, "Predicting of machine perception for automated driving," in *Autonomous Driving*. Springer, 2016, ch. 20, pp. 407–424.
- [59] K. C. J. Dietmayer, S. Reuter, and D. Nuss, "Representation of fused environment data," in *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*, H. Winner, S. Hakuli, F. Lotz, and C. Singer, Eds. Springer, 2016, ch. 24, pp. 567–604.
- [60] Dirección General de Tráfico, "Las principales cifras de la siniestralidad vial en España 2017," 2018.
- [61] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian detection: A benchmark," in *IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2009, pp. 304–311.
- [62] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbas, V. Golkov, P. van der Smagt, D. Cremers, and Thomas Brox, "Flownet: Learning optical flow with convolutional networks," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2758–2766.
- [63] 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.

- [64] X. Du, S. Karaman, and D. Rus, "A general pipeline for 3D detection of vehicles," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3194–3200.
- [65] M. Enzweiler and D. M. Gavrila, "Monocular pedestrian detection: Survey and experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2179–2195, 2009.
- [66] A. de la Escalera, E. Izquierdo, D. Martín, B. Musleh, F. García, and J. M. Armingol, "Stereo visual odometry in urban environments based on detecting ground features," *Robotics and Autonomous Systems*, vol. 80, no. June, pp. 1–10, 2016.
- [67] European Commission, "Towards a European road safety area: Policy orientations on road safety 2011-2020," 2010.
- [68] European Commission: Directorate General for Transport, "Annual accident report," Jun. 2017.
- [69] European Commission: Directorate-General Mobility and Transport, "Road safety in the European Union – trends, statistics and main challenges," 2018.
- [70] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2014.
- [71] M. Everingham, L. Van Gool, C. K. I. 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.
- [72] Q. Fan, L. Brown, and J. Smith, "A closer look at Faster R-CNN for vehicle detection," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 124–129.
- [73] O. Faugeras, *Three-dimensional computer vision: a geometric viewpoint*. MIT Press, 1993.
- [74] P. F. Felzenszwalb, R. Girshick, D. McAllester, and D. Ramanan, "Object detection with discriminatively trained part-based models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 9, pp. 1627–1645, 2010.
- [76] C. Fernández, C. Guindel, N.-O. Salscheider, and C. Stiller, "A deep analysis of the existing datasets for traffic light state recognition," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 248–254.
- [77] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

- [78] F. Flohr, M. Dumitru-Guzu, J. F. P. Kooij, and D. M. Gavrila, "A probabilistic framework for joint pedestrian head and body orientation estimation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 4, pp. 1872–1882, 2015.
- [79] D. A. Forsyth and J. Ponce, *Computer Vision: A Modern Approach*. Prentice Hall, 2012.
- [80] U. Franke, D. Pfeiffer, C. Rabe, C. Knoeppel, M.ENZweiler, F. Stein, and R. G. Herrtwich, "Making Bertha see," in *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2013, pp. 214–221.
- [81] A. Fregin, J. Müller, U. Kreßel, and K. Dietmayer, "The DriveU traffic light dataset: Introduction and comparison with existing datasets," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3376–3383.
- [82] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal-Japanese Society For Artificial Intelligence*, vol. 14, no. 771-780, p. 1612, 1999.
- [83] J. Fritsch, T. Kuhn, and A. Geiger, "A new performance measure and evaluation benchmark for road detection algorithms," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2013, pp. 1693–1700.
- [84] 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.
- [85] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied Soft Computing*, vol. 70, pp. 41–65, 2018.
- [86] F. García, D. Martín, A. de la Escalera, and J. M. Armingol, "Sensor fusion methodology for vehicle detection," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 123–133, 2017.
- [87] F. García, J. Urdiales, J. Carmona, D. Martín, and José M. Armingol, "Mobile based pedestrian detection with accurate tracking," in *IEEE Intelligent Vehicles Symposium (IV) - Workshop on Human Factors in Intelligent Vehicles (HFIV'16)*, 2016, pp. 44–48.
- [88] A. Geiger, M. Lauer, C. Wojek, C. Stiller, and R. Urtasun, "3D traffic scene understanding from movable platforms," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 36, no. 5, pp. 1012–1025, 2014.
- [89] 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.

- [90] 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.
- [91] 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.
- [92] A. Geiger, C. Wojek, and R. Urtasun, "Joint 3D estimation of objects and scene layout," in *Advances in Neural Information Processing Systems (NIPS)*, 2011.
- [93] GEM Motoring Assist. The history of automobile safety, [Online]. Available: <https://blog.motoringassist.com/history-of-automobile-safety/> (visited on 10/15/2018).
- [94] A. Ghodrati, M. Pedersoli, and T. Tuytelaars, "Is 2D information enough for viewpoint estimation?" In *Proc. British Machine Vision Conference (BMVC)*, 2014.
- [95] R. Girshick, "Fast R-CNN," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.
- [96] 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.
- [97] —, "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, 2016.
- [98] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," *Journal of Machine Learning Research*, vol. 9, pp. 249–256, 2010.
- [99] M. . Gómez-Silva, J. M. Armingol, and A. de la Escalera, "Multi-object tracking with data association by a similarity identification model," in *Proc. International Conference on Imaging for Crime Detection and Prevention (ICDP)*, 2016, 25 (6 .)-25 (6 .)(1).
- [100] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT Press, 2016.
- [101] H. Gotzig and G. Geduld, "Automotive LIDAR," in *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. Springer, 2016, ch. 18, pp. 405–430.

- [109] 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.
- [110] C. Guindel, D. Martín, 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.
- [111] —, “Fast joint object detection and viewpoint estimation for traffic scene understanding,” *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 4, pp. 74–86, 2018.
- [112] —, “Modeling traffic scenes for intelligent vehicles using CNN-based detection and orientation estimation,” in *ROBOT 2017: Third Iberian Robotics Conference: Volume 2*, 2018, pp. 487–498.
- [113] —, “Stereo vision-based convolutional networks for object detection in driving environments,” in *Computer Aided Systems Theory - EUROCAST 2017*, 2018, pp. 427–434.
- [114] C. Guindel, D. Martín, and J. M. Armingol, “Traffic scene awareness for intelligent vehicles using ConvNets and stereo vision,” *Robotics and Autonomous Systems*, vol. 112, pp. 109–122, 2019.
- [115] C. Guindel, D. Martín, J. M. Armingol, and C. Stiller, “Analysis of the influence of training data on road user detection,” in *2018 IEEE International Conference on Vehicular Electronics and Safety, ICVES 2018*, 2018, pp. 21–26.
- [116] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge University Press, 2003.
- [117] K. He, R. Girshick, and P. Dollár, “Rethinking ImageNet pre-training,” *Tech. Rep.*, 2018.
- [118] 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.
- [119] K. He, X. Zhang, S. Ren, and J. Sun, “Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification,” in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1026–1034.
- [120] —, “Deep residual learning for image recognition,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

- [121] D. Hernandez-Juarez, L. Schneider, A. Espinosa, D. Vázquez, A. M. López, U. Franke, M. Pollefeys, and J. C. Moure, "Slanted stixels: Representing San Francisco's steepest streets," in *Proc. British Machine Vision Conference (BMVC)*, 2017.
- [122] N. Hernández, A. Hussein, D. Cruzado, I. Parra, and J. M. Armingol, "Applying low cost WiFi-based localization to in-campus autonomous vehicles," in *Proc. IEEE Conference on Intelligent Transportation Systems, Proceedings (ITSC)*, 2017, pp. 848–853, ISBN: 9781538615256. DOI: [10.1109/ITSC.2017.8317780](https://doi.org/10.1109/ITSC.2017.8317780).
- [123] 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.
- [124] 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.
- [125] D. Hoiem, A. A. Efros, and M. Hebert, "Putting objects in perspective," *International Journal of Computer Vision*, vol. 80, no. 1, pp. 3–15, 2008.
- [126] J. Hosang, R. Benenson, P. Dollar, and B. Schiele, "What makes for effective detection proposals?" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, no. 4, pp. 814–830, 2016.
- [127] J. Hosang, R. Benenson, and B. Schiele, "A Convnet for non-maximum suppression," in *German Conference on Pattern Recognition 2016 (GCPR)*, 2016, pp. 192–204.
- [128] —, "Learning non-maximum suppression," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 4507–4515.
- [129] S. Houben, J. Stallkamp, J. Salmen, M. Schlipsing, and C. Igel, "Detection of traffic signs in real-world images: The German traffic sign detection benchmark," in *Proc. International Joint Conference on Neural Networks (IJCNN)*, 2013.
- [130] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861 [cs.CV]*, 2017.
- [131] 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.
- [132] X. Huang, X. Cheng, Q. Geng, B. Cao, D. Zhou, P. Wang, Y. Lin, and R. Yang, "The ApolloScape dataset for autonomous driving," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018, pp. 954–960.

- [133] K. Hyatt and C. Paukert, "Self-driving cars: A level-by-level explainer of autonomous vehicles," *Roadshow by CNET*, Mar. 29, 2018. [Online]. Available: <https://www.cnet.com/roadshow/news/self-driving-car-guide-autonomous-explanation> (visited on 11/15/2018).
- [134] INRIX, "INRIX global congestion ranking," 2018.
- [135] International Organization of Motor Vehicle Manufacturers. World vehicles in use, [Online]. Available: http://www.oica.net/wp-content/uploads//Total_in-use-All-Vehicles.pdf (visited on 10/15/2018).
- [136] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *Proc. International Conference on Machine Learning*, 2015, pp. 448–456.
- [137] G. Iyer, K. R. R., 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.
- [138] A. K. Jain, R. P. W. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.
- [140] M. B. Jensen, M. P. Philipsen, A. Mogelmoose, T. B. Moeslund, and M. M. Trivedi, "Vision for looking at traffic lights: Issues, survey, and perspectives," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 7, pp. 1800–1815, 2016.
- [141] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proc. ACM International Conference on Multimedia*, 2014, pp. 675–678.
- [142] J. Jin, K. Fu, and C. Zhang, "Traffic sign recognition with hinge loss trained convolutional neural networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 1991–2000, 2014.
- [143] V. John, K. Yoneda, B. Qi, Z. Liu, and S. Mita, "Traffic light recognition in varying illumination using deep learning and saliency map," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2014, pp. 2286–2291.
- [144] S. Kammel, J. Ziegler, B. Pitzer, *et al.*, "Team annieway's autonomous system for the 2007 darpa urban challenge," *Journal of Field Robotics*, vol. 25, no. 9, pp. 615–639, 2008.
- [145] A. Karpathy. Cs231n convolutional neural networks for visual recognition, Stanford University, [Online]. Available: <http://cs231n.github.io/convolutional-networks/> (visited on 12/05/2018).

- [146] G. Katz, C. Barrett, D. L. Dill, K. Julian, and M. K. J., "Reluplex: An efficient SMT solver for verifying deep neural networks," in *CAV 2017: Computer Aided Verification (LNCS, vol. 10426)*, 2017, pp. 99–117.
- [147] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *Proc. International Conference on Learning Representations (ICLR)*, 2015.
- [148] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár, "Panoptic segmentation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 9404–9413.
- [149] 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.
- [150] W. König, "Guidelines for user-centered development of DAS," in *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. Springer, 2016, ch. 32, pp. 781–796.
- [151] K. Konolige, "Small vision systems: Hardware and implementation," pp. 203–212, 1998.
- [152] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.
- [153] I. Krasin, T. Duerig, N. Alldrin, *et al.*, "OpenImages: A public dataset for large-scale multi-label and multi-class image classification.," *Dataset available from <https://storage.googleapis.com/openimages/web/index.html>*, 2017.
- [154] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems (NIPS)*, vol. 2, 2012, pp. 1097–1105.
- [155] 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.
- [156] 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.
- [157] K. Kwak, D. F. Huber, H. Badino, and T. Kanade, "Extrinsic calibration of a single line scanning lidar and a camera," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2011, pp. 3283–3289.

- [158] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, 2015.
- [159] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, "Handwritten digit recognition with a back-propagation network," in *Advances in neural information processing systems*, 1990, pp. 396–404.
- [160] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2323, 1998.
- [161] J. Levinson and S. Thrun, "Automatic online calibration of cameras and lasers," in *Robotics: Science and Systems*, 2013.
- [162] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 1513–1518.
- [163] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3D lidar using fully convolutional network," in *Robotics Science and Systems*, 2016.
- [164] P. Li, T. Qin, and S. Shen, "Stereo vision-based semantic 3D object and ego-motion tracking for autonomous driving," in *Computer Vision - ECCV 2018 (LNCS, vol. 11206)*, 2018, pp. 664–679.
- [165] 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.
- [166] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3D object detection," in *Computer Vision - ECCV 2018 (LNCS, vol. 11220)*, 2018, pp. 663–678.
- [167] 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.
- [168] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2980–2988.
- [169] 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 *Computer Vision - ECCV 2014 (LNCS, vol. 8693)*, 2014, pp. 740–755.
- [170] T. Litman, "Autonomous vehicle implementation predictions," Victoria Transport Policy Institute Victoria, Canada, 2017.

- [171] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot multibox detector," in *Computer Vision - ECCV 2016 (LNCS, vol. 9905)*, 2016, pp. 21–37.
- [172] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.
- [173] R. J. López-Sastre, T. Tuytelaars, and S. Savarese, "Deformable part models revisited: A performance evaluation for object category pose estimation," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2011, pp. 1052–1059.
- [174] A. M. López, G. Villalonga, L. Sellart, G. Ros, D. Vázquez, J. Xu, J. Marín, and A. Mozafari, "Training my car to see using virtual worlds," *Image and Vision Computing*, vol. 68, pp. 102–118, 2017.
- [175] H. Luo, Y. Yang, B. Tong, F. Wu, and B. Fan, "Traffic sign recognition using a multi-task convolutional neural network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1100–1111, 2017.
- [176] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The Oxford RobotCar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.
- [177] S. Mahendran, H. Ali, and R. Vidal, "3D pose regression using convolutional neural networks," in *IEEE International Conference on Computer Vision Workshops (ICCVW)*, 2017, pp. 2174–2182.
- [179] P. Marín-Plaza, J. Beltrán, A. Hussein, B. Musleh, D. Martín, A. de la Escalera, and J. M. Armingol, "Stereo vision-based local occupancy grid map for autonomous navigation in ROS," in *Proc. Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications (VISIGRAPP) - Volume 3: VISAPP*, 2016, pp. 703–708.
- [180] M. Maurer, J. C. Gerdes, B. Lenz, H. Winner, *et al.*, *Autonomous Driving*. Springer, 2016.
- [181] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox, "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4040–4048.
- [182] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [183] P. Micikevicius, S. Narang, J. Alben, *et al.*, "Mixed precision training," in *Proc. International Conference on Learning Representations (ICLR)*, 2018.

- [184] D. Milakis, B. Van Arem, and B. Van Wee, "Policy and society related implications of automated driving: A review of literature and directions for future research," *Journal of Intelligent Transportation Systems*, vol. 21, no. 4, pp. 324–348, 2017.
- [185] A. Møgelmoose, M. M. Trivedi, and T. B. Moeslund, "Vision based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 4, pp. 1484–1497, 2012.
- [186] 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.
- [187] H. Moravec and A. Elfes, "High resolution maps from wide angle sonar," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 1985, pp. 116–121.
- [188] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3D bounding box estimation using deep learning and geometry," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 7074–7082.
- [189] J. K. Murthy, G. V. S. Krishna, F. Chhaya, and K. M. Krishna, "Reconstructing vehicles from a single image: Shape priors for road scene understanding," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 724–731.
- [190] B. Musleh, D. Martín, J. M. Armingol, and A. de la Escalera, "Continuous pose estimation for stereo vision based on UV disparity applied to visual odometry in urban environments," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 3983–3988.
- [191] G. Neuhold, T. Ollmann, S. R. Bulo, and P. Kotschieder, "The Mapillary Vistas dataset for semantic understanding of street scenes," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 4990–4999.
- [192] M. Oeljeklaus, F. Hoffmann, and T. Bertram, "A combined recognition and segmentation model for urban traffic scene understanding," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 2292–2297.
- [193] —, "A fast multi-task CNN for spatial understanding of traffic scenes," in *Proc. IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2825–2830.
- [194] S. Ohl, "Static software architecture of the sensor data fusion module of the Stadtpilot project," in *Automotive Systems Engineering*. Springer, 2013, ch. 5, pp. 81–109.

- [195] E. Ohn-Bar and M. M. Trivedi, "Learning to detect vehicles by clustering appearance patterns," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2511–2521, 2015.
- [196] D. Olmeda, C. Premebida, U. Nunes, J. M. Armingol, and A. de la Escalera, "Pedestrian detection in far infrared images," *Integrated Computer-Aided Engineering*, vol. 20, no. 4, pp. 347–360, 2013.
- [197] OpenCV documentation. Camera calibration and 3D reconstruction, [Online]. Available: https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html (visited on 02/28/2019).
- [198] —, Camera calibration with OpenCV, [Online]. Available: https://docs.opencv.org/3.4.2/d4/d94/tutorial_camera_calibration.html (visited on 06/06/2019).
- [199] 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.
- [200] I. Parra Alonso, D. Fernández Llorca, M. Á. Sotelo, L. M. Bergasa, P. Revenga de Toro, J. Nuevo, M. Ocaña, and M. Á. García Garrido, "Combination of feature extraction methods for SVM pedestrian detection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 2, pp. 292–307, 2007.
- [201] Partnership on Sustainable Low Carbon Transport, "Creating universal access to safe, clean and affordable transport," 2013.
- [202] B. Pepik, M. Stark, P. Gehler, and B. Schiele, "Teaching 3D geometry to deformable part models," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 3362–3369.
- [203] —, "Multi-view and 3D deformable part models," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 11, pp. 2232–2245, 2015.
- [204] M. Pereira, D. Silva, V. Santos, and P. Dias, "Self calibration of multiple LIDARs and cameras on autonomous vehicles," *Robotics and Autonomous Systems*, vol. 83, pp. 326–337, 2016.
- [205] C. C. Pham and J. W. Jeon, "Robust object proposals re-ranking for object detection in autonomous driving using convolutional neural networks," *Signal Processing: Image Communication*, vol. 53, pp. 110–122, 2017.
- [207] Point Cloud Library (PCL) documentation. Getting started / basic structures, [Online]. Available: http://pointclouds.org/documentation/tutorials/basic_structures.php (visited on 09/30/2019).

- [208] 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.
- [209] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: An astounding baseline for recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2014, pp. 512–519.
- [210] 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.
- [211] J. Redmon and A. Farhadi, "YOLO9000: Better, faster, stronger," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [212] —, "YOLOv3: An incremental improvement," 2018.
- [213] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.
- [214] —, "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.
- [215] S. R. Richter, Z. Hayder, and V. Koltun, "Playing for benchmarks," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 2232–2241.
- [216] C. H. Rodríguez Garavito, J. Carmona, A. de la Escalera, and J. M. Armingol, "Stereo road detection based on ground plane," in *Computer Aided Systems Theory - EUROCAST 2015*, 2015, pp. 748–755.
- [217] 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.
- [219] C. H. Rodríguez-Garavito, C. Guindel, and J. M. Armingol, "Sistema de asistencia a la conducción para detección y clasificación de carriles," in *XXXVI Jornadas de Automática*, 2015, pp. 26–31.
- [220] E. Romera, J. M. Álvarez, L. M. Bergasa, and R. Arroyo, "ERFNet: Efficient residual factorized ConvNet for real-time semantic segmentation," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 1, pp. 263–272, 2018.

- [221] E. Romera, L. M. Bergasa, J. M. Alvarez, and M. Trivedi, "Train here, deploy there: Robust segmentation in unseen domains," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1828–1833.
- [222] E. Romera, L. M. Bergasa, and R. Arroyo, "Can we unify monocular detectors for autonomous driving by using the pixel-wise semantic segmentation of CNNs?" In *IEEE Intelligent Vehicles Symposium (IV) - DeepDriving Workshop*, 2016.
- [223] F. J. Romero-Ramirez, R. Muñoz-Salinas, and R. Medina-Carnicer, "Speeded up detection of squared fiducial markers," *Image and Vision Computing*, vol. 76, pp. 38–47, 2018.
- [224] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, 2015, pp. 234–241.
- [225] ROS wiki. Image_pipeline package documentation, [Online]. Available: http://wiki.ros.org/image_pipeline (visited on 06/07/2019).
- [226] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. M. Lopez, "The SYNTHIA dataset: A large collection of synthetic images for semantic segmentation of urban scenes," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 4321–4330.
- [227] F. Rosique, P. J. Navarro, C. Fernández, and A. Padilla, "A systematic review of perception system and simulators for autonomous vehicles research," *Sensors*, vol. 19, no. 3, p. 648, 2019.
- [228] C. Rubino, M. Crocco, and A. Del Bue, "3D object localisation from multi-view image detections," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 6, pp. 1281–1294, 2018.
- [230] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," California Univ San Diego La Jolla Inst for Cognitive Science, Tech. Rep., 1985.
- [231] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.
- [232] S. J. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Pearson, 2016.
- [233] SAE International, "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. j3016.," Tech. Rep., 2016.

- [234] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018, pp. 4510–4520.
- [235] 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.
- [236] B. Schiele and C. Wojek, "Camera based pedestrian detection," in Springer, 2016, ch. 22, pp. 525–545.
- [237] N. Schneider, F. Piewak, C. Stiller, and U. Franke, "RegNet: Multimodal sensor registration using deep neural networks," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2017, pp. 1803–1810.
- [238] B. Schwarz, "LIDAR mapping the world in 3D," *Nature Photonics*, vol. 4, no. 7, p. 429, 2010.
- [239] T. Scott, A. A. Morye, P. Piniés, L. M. Paz, I. Posner, and P. Newman, "Exploiting known unknowns: Scene induced cross-calibration of lidar-stereo systems," in *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015, pp. 3647–3653.
- [240] T. Scott, A. A. Morye, P. Piniés, L. M. Paz, I. Posner, and P. Newman, "Choosing a time and place for calibration of lidar-camera systems," in *Proc. IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 4349–4356.
- [241] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. Lecun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2013, pp. 3626–3633.
- [242] 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.
- [243] A. Shrivastava, A. Gupta, and R. Girshick, "Training region-based object detectors with online hard example mining," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 761–769.
- [244] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Proc. International Conference on Learning Representations (ICLR)*, 2015.
- [245] S. Singh, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey," U. S. Department of Transportation. National Highway Traffic Safety Administration., 2015.

- [246] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 4, pp. 1773–1795, 2013.
- [247] S. Song, S. P. Lichtenberg, and J. Xiao, "SUN RGB-D: A RGB-D scene understanding benchmark suite," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 567–576.
- [248] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.
- [249] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition," *Neural Networks*, vol. 32, pp. 323–332, 2012.
- [250] J. Stallkamp, M. Schlipsing, J. Salmen, and C. Igel, "The german traffic sign recognition benchmark: A multi-class classification competition," in *Proc. International Joint Conference on Neural Networks (IJCNN)*, 2011, pp. 1453–1460.
- [251] C. Stiller, A. Bachmann, and A. Geiger, "Fundamentals of machine vision," in *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. Springer, 2016, ch. 20, pp. 461–494.
- [252] H. Su, C. R. Qi, Y. Li, and L. J. Guibas, "Render for CNN: Viewpoint estimation in images using CNNs trained with rendered 3D model views," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 2686–2694.
- [253] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, Inception-ResNet and the impact of residual connections on learning," in *Proc. AAAI Conference on Artificial Intelligence (AAAI)*, 2017, pp. 4278–4284.
- [254] 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)*, 2015, pp. 1–9.
- [255] 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.
- [256] R. Szeliski, *Computer vision: algorithms and applications*. Springer, 2010.

- [257] O. S. Tas, N. O. Salscheider, F. Poggenhans, S. Wirges, C. Bandera, M. R. Zofka, T. Strauss, J. M. Zollner, and C. Stiller, "Making Bertha cooperate—Team AnnieWAY's entry to the 2016 Grand Cooperative Driving Challenge," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 4, pp. 1262–1276, 2018.
- [258] L. Taylor and G. Nitschke, "Improving deep learning using generic data augmentation," in *Proc. IEEE Symposium Series on Computational Intelligence (SSCI)*, 2018, pp. 1542–1547.
- [259] D. Teney and J. Piater, "Continuous pose estimation in 2D images at instance and category levels," in *Proc. International Conference on Computer and Robot Vision*, 2013, pp. 121–127.
- [260] C. Thorpe, M. H. Hebert, T. Kanade, and S. A. Shafer, "Vision and navigation for the Carnegie-Mellon Navlab," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 10, no. 3, pp. 362–373, 1988.
- [261] S. Thrun, M. Montemerlo, H. Dahlkamp, *et al.*, "Stanley: The robot that won the DARPA Grand Challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.
- [262] W. Tian and M. Lauer, "Tracking objects with severe occlusion by adaptive part filter modeling — in traffic scenes and beyond," *IEEE Intelligent Transportation Systems Magazine*, vol. 10, no. 4, pp. 60–73, 2018.
- [263] J. R. R. Uijlings, K. E. A. Van De Sande, T. Gevers, and A. W. M. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, 2013.
- [264] United Nations, "Mobilizing sustainable transport for development. analysis and policy recommendations from the United Nations.," United Nations, 2016.
- [265] C. Urmson, J. Anhalt, D. Bagnell, *et al.*, "Autonomous driving in urban environments: Boss and the Urban Challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.
- [266] 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.
- [267] S. Venugopalan, H. Xu, J. Donahue, M. Rohrbach, R. Mooney, and K. Saenko, "Translating videos to natural language using deep recurrent neural networks," in *Proc. Human Language Technologies: Annual Conference of the North American Chapter of the ACL (NAACL-HLT)*, 2015, pp. 1494–1504.

- [268] 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)*, 2001, pp. 511–518.
- [269] W. Wachenfeld, H. Winner, J. C. Gerdes, B. Lenz, M. Maurer, S. Beiker, E. Fraedrich, and T. Winkle, "Use cases for autonomous driving," in *Autonomous Driving*. Springer, 2016, ch. 2, pp. 9–37.
- [270] S. Wang, M. Bai, G. Mattyus, H. Chu, W. Luo, B. Yang, J. Liang, J. Cheverie, S. Fidler, and R. Urtasun, "TorontoCity: Seeing the world with a million eyes," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2017, pp. 3028–3036.
- [271] M. Weber, P. Wolf, and J. M. Zöllner, "DeepTLR: A single deep convolutional network for detection and classification of traffic lights," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2016, pp. 342–348.
- [272] H. Winner, "Automotive RADAR," in *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. Springer, 2016, ch. 17, pp. 325–404.
- [273] H. Winner, S. Hakuli, F. Lotz, and C. Singer, *Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort*. Springer, 2016.
- [274] 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.
- [275] C. Wojek, S. Walk, S. Roth, K. Schindler, and B. Schiele, "Monocular visual scene understanding: Understanding multi-object traffic scenes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 4, pp. 882–897, 2013.
- [276] World Health Organization. WHO global ambient air quality database, [Online]. Available: <https://www.who.int/airpollution/data/en/> (visited on 10/15/2018).
- [277] —, "Global status report on road safety 2015," 2015.
- [278] —, "Global health estimates 2016: Deaths by cause, age, sex, by country and by region, 2000–2016.," 2018.
- [279] Y. Wu and K. He, "Group normalization," in *Computer Vision - ECCV 2018 (LNCS, vol. 11217)*, 2018, pp. 3–19.
- [280] Y. Xiang, W. Choi, Y. Lin, and S. Savarese, "Data-driven 3D voxel patterns for object category recognition," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1903–1911.
- [281] —, "Subcategory-aware convolutional neural networks for object detection," in *Proc. IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2017, pp. 924–933.

- [282] Y. Xiang, R. Mottaghi, and S. Savarese, "Beyond PASCAL: A benchmark for 3D object detection in the wild," in *Proc. IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014, pp. 75–82.
- [283] M. Xie, L. Trassoudaine, J. Alizon, M. Thonnat, and J. Gallice, "Active and intelligent sensing of road obstacles: Application to the European Eureka-PROMETHEUS project," in *Proc. International Conference on Computer Vision (ICCV)*, IEEE, 1993, pp. 616–623.
- [284] B. Yang, M. Liang, and R. Urtasun, "HDNET: Exploiting HD maps for 3D object detection," in *Proc. Conference on Robot Learning (CoRL)*, 2018, pp. 146–155.
- [285] F. Yang, W. Choi, and Y. Lin, "Exploit all the layers: Fast and accurate CNN object detector with scale dependent pooling and cascaded rejection classifiers," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2129–2137.
- [286] G. Yang, H. Zhao, J. Shi, Z. Deng, and J. Jia, "SegStereo: Exploiting semantic information for disparity estimation," in *Computer Vision - ECCV 2018 (LNCS, vol. 11211)*, 2018, pp. 660–676.
- [287] L. Yang, J. Liu, and X. Tang, "Object detection and viewpoint estimation with auto-masking neural network," in *Computer Vision - ECCV 2014 (LNCS, vol. 8691)*, 2014, pp. 441–455.
- [288] L. Yang, P. Luo, C. C. Loy, and X. Tang, "A large-scale car dataset for fine-grained categorization and verification," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 3973–3981.
- [289] J. J. Yebes, L. M. Bergasa, R. Arroyo, and A. Lazaro, "Supervised learning and evaluation of KITTI's cars detector with DPM," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2014, pp. 768–773.
- [290] F. Yu, W. Xian, Y. Chen, F. Liu, M. Liao, V. Madhavan, and T. Darrell, "BDD100K: A diverse driving video database with scalable annotation tooling," 2018.
- [291] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision - ECCV 2014 (LNCS, vol. 8689)*, 2014, pp. 818–833.
- [292] Y. Zeng, Y. Hu, S. Liu, J. Ye, Y. Han, X. Li, and N. Sun, "RT3D: Real-time 3D vehicle detection in lidar point cloud for autonomous driving," *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3434–3440, 2018.
- [293] H. Zhang, A. Geiger, and R. Urtasun, "Understanding high-level semantics by modeling traffic patterns," in *Proc. IEEE International Conference on Computer Vision (ICCV)*, 2013, pp. 3056–3063.

- [294] S. Zhang, R. Benenson, and B. Schiele, "CityPersons: A diverse dataset for pedestrian detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 3213–3221.
- [295] 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.
- [296] 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.
- [297] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, "Traffic-sign detection and classification in the wild," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 2110–2118.
- [298] J. Ziegler, P. Bender, M. Schreiber, *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.
- [299] J. Ziegler, H. Lategahn, M. Schreiber, C. Keller, C. Knoppel, J. Hipp, M. Haueis, and C. Stiller, "Video based localization for BERTHA," in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, 2014, pp. 1231–1238.