





Universitat Autònoma de Barcelona

**ADVERTIMENT.** L'accés als continguts d'aquesta tesi queda condicionat a l'acceptació de les condicions d'ús establertes per la següent llicència Creative Commons:  [http://cat.creativecommons.org/?page\\_id=184](http://cat.creativecommons.org/?page_id=184)

**ADVERTENCIA.** El acceso a los contenidos de esta tesis queda condicionado a la aceptación de las condiciones de uso establecidas por la siguiente licencia Creative Commons:  <http://es.creativecommons.org/blog/licencias/>

**WARNING.** The access to the contents of this doctoral thesis it is limited to the acceptance of the use conditions set by the following Creative Commons license:  <https://creativecommons.org/licenses/?lang=en>



**Universitat Autònoma  
de Barcelona**

# Monocular Depth Estimation for Autonomous Driving

A dissertation submitted by **Akhil Gurram** at Universitat Autònoma de Barcelona to fulfil the degree of **Doctor of Philosophy**.

Bellaterra, January 12, 2022

Director	<p><b>Dr. Antonio López Peña</b>  Dept. Ciències de la Computació &amp; Centre de Visió per Computador  Universitat Autònoma de Barcelona (UAB), Spain</p>
Co-Director	<p><b>Dr. Onay Urfalioglu</b>  Huawei Munich Research Center  Munich, Germany</p>
Thesis committee	<p><b>Dr. Arturo de la Escalera</b>  Dpt. Ingeniería de Sistemas y Automática  Universidad Carlos III de Madrid, Spain</p> <p><b>Dr. Maria Vanrell</b>  Dpt. Ciències de la Computació  Universitat Autònoma de Barcelona (UAB), Spain</p> <p><b>Dr. Adrien Gaidon</b>  Machine Learning Group  Toyota Research Institute (TRI), Santa Clara, CA, USA</p>
International evaluators	<p><b>Dr. Adrien Gaidon</b>  Machine Learning Group  Toyota Research Institute (TRI), Santa Clara, CA, USA</p> <p><b>Dr. Cesar Roberto De Souza</b>  Computer Vision Group  NAVER LABS Europe, Meylan, France</p>




---

This document was typeset by the author using  $\LaTeX 2_{\epsilon}$ .

The research described in this book was carried out at the Centre de Visió per Computador, Universitat Autònoma de Barcelona. Copyright © 2022 by **Akhil Gurrām**. All rights reserved. No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopy, recording, or any information storage and retrieval system, without permission in writing from the author.

ISBN: 978-84-124793-0-0

Printed by Ediciones Gráficas Rey, S.L.

# Acknowledgements

PhD has been a life-changing experience for me, and it would not have been possible without the proper support and guidance from my supervisors, colleagues, friends, and family.

To begin with, I would like to express my sincere gratitude to my supervisor Dr. Antonio López, for giving me the opportunity to carry out PhD under his guidance. His professional advice, expertise, encouragement, and invaluable support has helped me develop my scientific endeavor throughout this long journey. While working together, he always made some comments like- “The devil is in the details” OR “The truth lies in the code.” His insightful words inspired me to dig more into the topic and get perfection in my work. I would also like to thank my co-supervisor, Dr. Onay Urfalioglu, for his guidance, understanding, and patience during my research. Both of them have played essential roles in my professional growth, and this thesis would not have been possible without them always being at my side, with positive criticism.

I am incredibly thankful to Universitat Autònoma de Barcelona (UAB), Spain, and Huawei Munich Research Center (Huawei-MRC), Germany, for giving me the platform to pursue my research throughout my PhD and to provide all the necessary facilities. I also would like to thank Huawei-MRC for providing me with the doctoral study fellowship for three years.

I appreciate all the scientific support that I got from the people at the Computer Vision Center at UAB during the progression of my PhD project. I thank Dr. Gabriel Villalonga for the guidance to reach my research milestones and Antoni Bigata, Yi Xiao, and Dr. José A. Iglesias for their support during several successful collaborations.

Besides, I would also like to thank my teammates at Huawei. Starting with Automotive Engineering Lab (AEL), thanks again to Dr. Onay Urfalioglu for introducing me to wonderful people working at Huawei during my last five years. I was fortunate to meet such talented people as Ahmet Faruk Tuna and Fengyi Shen, with whom I could achieve state-of-the-art on Monocular Depth Estimation project. In addition, special thanks to Mehmet, Patrick, Mohammad, Henrique, Dr. Shile Li, Dr. Dzmitry Tsishkou, Dr. Laurent Smadja for their invaluable scientific and technical discussions. Finally, thanks to Rauchfuss Holm for insightful conversations and support in challenging situations. Last but not least, I thank Marek Neumann (director- AEL), Yang Yongwei, and Anna de Arriba Bote for helping me with all the paperwork in the company.

---

In addition, I am incredibly thankful to all my friends for their support and encouragement throughout this long journey. Thanks to Madhu (akka) for all your support and motivation, Manu (the one who has answers to any question) for encouraging me, RajeshK for his positive guidance and fruitful discussions, Sunil for being there to discuss anything, and Shannu & Bhanu for cheering me up constantly in their ways. Even though I have been away from home, people at Pembaur-house (Mallika & Bhargav, Hanne & Manne) have always uplifted me in the tough phases.

Thanks also to Mustafa, Jyothi (cycling buddy), Ramya, Vaishnavi, Srinivas & Sailaja, Samar, Prathemesh, Varsha, Bibhuti, Ruheen, Lipi, Mansi, and Pavan anna. The list continues to the friends from India (far apart, still remained close all throughout) – Balaram, Venu (aka Mayam), Srinivas, and The Fraternity group. Thanks to all of them for supporting me.

Finally, I must thank my family for their continuous encouragement and unfailing support throughout these years. Especially my parents and my siblings for their enduring love and faith in me, to be my backbone in good and bad times, for powering me continuously with optimistic thinking, encouragement, and enthusiasm. And last but not least, my lovely nieces (Ishanvi, Pranavi) and nephew (Samu) always find a way to cheer me up.

Thanks again to everyone who made this thesis possible and with that, I hope to carry all the learnings from my journey throughout my life.

# Abstract

3D geometric information is essential for on-board perception in autonomous driving and driver assistance. Autonomous vehicles (AVs) are equipped with calibrated sensor suites. As part of these suites, we can find LiDARs, which are expensive active sensors in charge of providing the 3D geometric information. Depending on the operational conditions for the AV, calibrated stereo rigs may be also sufficient for obtaining 3D geometric information, being these rigs less expensive and easier to install than LiDARs. However, ensuring a proper maintenance and calibration of these types of sensors is not trivial. Accordingly, there is an increasing interest on performing monocular depth estimation (MDE) to obtain 3D geometric information on-board. MDE is very appealing since it allows for appearance and depth being on direct pixelwise correspondence without further calibration. Moreover, a set of single cameras with MDE capabilities would still be a cheap solution for on-board perception, relatively easy to integrate and maintain in an AV.

Best MDE models are based on Convolutional Neural Networks (CNNs) trained in a supervised manner, i.e., assuming pixelwise ground truth (GT). Accordingly, the overall goal of this PhD is to study methods for improving CNN-based MDE accuracy under different training settings. More specifically, this PhD addresses different research questions that are described below. When we started to work in this PhD, state-of-the-art methods for MDE were already based on CNNs. In fact, a promising line of work consisted in using image-based semantic supervision (i.e., pixel-level class labels) while training CNNs for MDE using LiDAR-based supervision (i.e., depth). It was common practice to assume that the same raw training data are complemented by both types of supervision, i.e., with depth and semantic labels. However, in practice, it was more common to find heterogeneous datasets with either only depth supervision or only semantic supervision. Therefore, our first work was to research if we could train CNNs for MDE by leveraging depth and semantic information from heterogeneous datasets. We show that this is indeed possible, and we surpassed the state-of-the-art results on MDE at the time we did this research. To achieve our results, we proposed a particular CNN architecture and a new training protocol.

After this research, it was clear that the upper-bound setting to train CNN-based MDE models consists in using LiDAR data as supervision. However, it would be cheaper and more scalable if we would be able to train such models from monocular sequences. Obviously, this is far more challenging, but worth to research. Training MDE models using monocular sequences is possible by relying on structure-from-motion (SfM) principles to generate self-supervision. Nevertheless, problems of camouflaged objects,

---

visibility changes, static-camera intervals, textureless areas, and scale ambiguity, diminish the usefulness of such self-supervision. To alleviate these problems, we perform MDE by virtual-world supervision and real-world SfM self-supervision. We call our proposal MonoDEVSNets. We compensate the SfM self-supervision limitations by leveraging virtual-world images with accurate semantic and depth supervision, as well as addressing the virtual-to-real domain gap. MonoDEVSNets outperformed previous MDE CNNs trained on monocular and even stereo sequences. We have publicly released MonoDEVSNets at <https://github.com/HMRC-AEL/MonoDEVSNets>.

Finally, since MDE is performed to produce 3D information for being used in downstream tasks related to on-board perception. We also address the question of whether the standard metrics for MDE assessment are a good indicator for future MDE-based driving-related perception tasks. By using 3D object detection on point clouds as proxy of on-board perception, we conclude that, indeed, MDE evaluation metrics give rise to a ranking of methods which reflects relatively well the 3D object detection results we may expect.

## Resumen

La información geométrica 3D es esencial para percibir el entorno desde un vehículo autónomo (VA) o asistido. Para ello, están equipados con sensores calibrados. Podemos encontrar sensores LiDAR que proporcionan esa información 3D, aunque son relativamente costosos. Dependiendo de las condiciones operativas del VA, los sistemas estereoscópicos también pueden ser suficientes para obtener información 3D, siendo sistemas más baratos y fáciles de instalar. Sin embargo, asegurar un correcto mantenimiento y calibración de este tipo de sensores no es trivial. En consecuencia, existe un interés creciente en realizar una estimación monocular de la profundidad (EMP) para obtener información 3D. La EMP permite que la apariencia visual y el 3D se correspondan a nivel de píxel sin una calibración adicional. Un conjunto de cámaras individuales con capacidad de EMP sería una solución barata para la percepción desde un VA, relativamente fácil de integrar y mantener.

Los mejores modelos de EMP se basan en redes neuronales convolucionales entrenadas de manera supervisada. En consecuencia, el objetivo general de esta tesis doctoral es estudiar métodos para mejorar la precisión de esos modelos en diferentes circunstancias prácticas que encontramos al realizar su entrenamiento. Más concretamente, esta tesis aborda las diferentes cuestiones que se describen a continuación.

Al inicio de esta tesis, una línea de trabajo prometedora para entrenar modelos de EMP consistía en utilizar la supervisión semántica basada en imágenes y supervisión de profundidad basada en LiDAR. Se suponía que los mismos datos de entrenamiento tenían ambos tipos de supervisión asociada, es decir, metainformación de profundidad y semántica. Sin embargo, en la práctica, era más común encontrar conjuntos de datos con solo supervisión de profundidad o solo semántica. Por lo tanto, nuestro primer trabajo fue investigar si podíamos entrenar modelos de EMP aprovechando información de profundidad y semántica proveniente de conjuntos de datos distintos y heterogéneos. Demostramos que esto es posible, y superamos los resultados de vanguardia en EMP de aquel momento. Para ello, propusimos un nuevo protocolo de entrenamiento para los modelos EMP.

Esta investigación también dejó claro que la supervisión basada en LiDAR es la que da lugar a modelos de EMP más precisos. Sin embargo, sería más barato y escalable si pudiéramos entrenar esos modelos a partir de secuencias monoculares. Esto es mucho más complejo ya que requiere utilizar los principios que permiten inferir estructura a partir del movimiento (SfM en inglés), generando así autosupervisión. Sin embargo, numerosos problemas prácticos disminuyen la utilidad de este tipo de autosupervisión.



---

Para aliviar estos problemas, entrenamos modelos de EMP mediante supervisión de imágenes virtuales con información de profundidad asociada y autosupervisión vía SfM de secuencias monoculares reales. A nuestra propuesta la llamamos MonoDEVSNet <<https://github.com/HMRC-AEL/MonoDEVSNet>>. MonoDEVSNet superó la precisión de otros modelos de vanguardia también entrenados en secuencias monoculares e incluso estéreo.

Finalmente, dado que la EMP se aplica para obtener 3D que será utilizado en tareas posteriores de percepción, también abordamos la cuestión de si las métricas estándar para la evaluación de modelos EMP son realmente un buen indicador para esas futuras tareas. Utilizando la detección de objetos en nubes de puntos 3D como ejemplo de percepción, llegamos a la conclusión de que, de hecho, las métricas de evaluación EMP dan lugar a una clasificación de métodos que refleja relativamente bien los resultados esperables en detección 3D de objetos.

## Resum

La informació geomètrica 3D és essencial per percebre l'entorn des d'un vehicle autònom (VA) o assistit. Per això, estan equipats amb sensors calibrats. Podem trobar sensors LiDAR que proporcionen aquesta informació 3D, encara que són relativament costosos. Depenent de les condicions operatives del VA, els sistemes estereoscòpics també poden ser suficients per obtenir informació 3D, i són sistemes més barats i fàcils d'instal·lar. Tot i així, assegurar un correcte manteniment i calibratge d'aquest tipus de sensors no és trivial. En conseqüència, hi ha un interès creixent a fer una estimació monocular de la profunditat (EMP) per obtenir informació 3D. L'EMP permet que l'aparença visual i el 3D es corresponguin a nivell de píxel sense un calibratge addicional. Un conjunt de càmeres individuals amb capacitat d'EMP seria una solució barata per a la percepció des d'un VA, relativament fàcil d'integrar i mantenir.

Els millors models EMP es basen en xarxes neuronals convolucionals entrenades de manera supervisada. En conseqüència, l'objectiu general d'aquesta tesi doctoral és estudiar mètodes per millorar la precisió d'aquests models en diferents circumstàncies pràctiques que trobem en l'entrenament. Més concretament, aquesta tesi aborda les diferents qüestions que es descriuen a continuació.

A l'inici d'aquesta tesi, una línia de treball prometedora per entrenar models d'EMP consistia a utilitzar la supervisió semàntica basada en imatges i la supervisió de profunditat basada en LiDAR. Se suposava que les mateixes dades d'entrenament tenien tots dos tipus de supervisió associada, és a dir, meta-informació de profunditat i semàntica. No obstant això, a la pràctica, era més comú trobar conjunts de dades amb només supervisió de profunditat o només semàntica. Per tant, el nostre primer treball va ser investigar si podíem entrenar models d'EMP aprofitant informació de profunditat i semàntica provinent de conjunts de dades diferents i heterogenis. Demostrem que això és possible, i superem els resultats d'avantguarda a l'EMP d'aquell moment. Per això, vam proposar un nou protocol d'entrenament per als models EMP.

Aquesta investigació també va deixar clar que la supervisió basada en LiDAR és la que dona lloc a models més precisos d'EMP. Tot i això, seria més barat i escalable si poguéssim entrenar aquests models a partir de seqüències monoculares. Això és molt més complex ja que requereix utilitzar els principis que permeten inferir estructura a partir del moviment (SfM en anglès), generant així auto-supervisió. No obstant això, molts problemes pràctics disminueixen la utilitat d'aquest tipus d'auto-supervisió. Per alleujar aquests problemes entrenem models d'EMP mitjançant la supervisió d'imatges virtuals amb informació de profunditat associada i auto-supervisió via SfM

---

de seqüències monoculars reals. Anomenem la nostra proposta com MonoDEVSNNet <<https://github.com/HMRC-AEL/MonoDEVSNNet>>. MonoDEVSNNet va superar la precisió d'altres models d'avantguarda també entrenats en seqüències monoculars i, fins i tot, estèreo.

Finalment, atès que l'EMP s'aplica per obtenir 3D que serà utilitzat en tasques posteriors de percepció, també abordem la qüestió de si les mètriques estàndard per a l'avaluació de models EMP són realment un bon indicador per a aquestes tasques futures. Utilitzant la detecció d'objectes en núvols de punts 3D com a exemple de percepció, arribem a la conclusió que, de fet, les mètriques d'avaluació d'EMP donen lloc a una classificació de mètodes que reflecteix relativament els resultats esperables en detecció 3D d'objectes.

# Contents

<b>Abstract (English/Spanish/Catalan)</b>	<b>iii</b>
<b>List of figures</b>	<b>xiii</b>
<b>List of tables</b>	<b>xix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Autonomous Driving . . . . .	1
1.2 Sensing the Environment . . . . .	3
1.3 Vision-based Depth Computation . . . . .	4
1.4 Learning to Compute Depth from Single Images . . . . .	7
1.5 PhD Objective and Outline . . . . .	9
<b>2 Monocular Depth Estimation by Learning from Heterogeneous Datasets</b>	<b>11</b>
2.1 Introduction . . . . .	11
2.2 Related Work . . . . .	13
2.3 Proposed Approach for Monocular Depth Estimation . . . . .	15
2.3.1 Overall training strategy . . . . .	15
2.3.2 Network Architecture . . . . .	17
2.4 Experimental Results . . . . .	19

2.4.1	Datasets . . . . .	19
2.4.2	Implementation Details . . . . .	19
2.4.3	Evaluation Metrics . . . . .	19
2.4.4	Results . . . . .	21
2.5	Conclusion . . . . .	26
<b>3</b>	<b>Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision</b>	<b>27</b>
3.1	Introduction . . . . .	27
3.2	Related Work . . . . .	29
3.2.1	Supervised MDE . . . . .	29
3.2.2	Self-supervised MDE . . . . .	30
3.2.3	Virtual-world data for MDE . . . . .	32
3.2.4	Relationship of MonoDEVSNets with previous literature . . . . .	33
3.3	Methods . . . . .	33
3.3.1	Training data . . . . .	34
3.3.2	MonoDEVSNets architecture: $\Psi(\theta; x)$ . . . . .	34
3.3.3	Problem formulation . . . . .	34
3.3.4	SfM Self-supervised loss: $\mathcal{L}^{\text{sf}}(\theta, \vartheta^{\text{sf}}, X^r)$ . . . . .	36
3.3.5	Supervised loss: $\mathcal{L}^{\text{sp}}(\theta; X^s, Y^s)$ . . . . .	37
3.3.6	Domain adaptation loss: $\mathcal{L}^{\text{da}}(\theta^{\text{enc}}, \vartheta^{\text{da}}, X^r, X^s)$ . . . . .	37
3.3.7	Overall training procedure . . . . .	38
3.3.8	Absolute depth computation . . . . .	40
3.4	Experimental Results . . . . .	40

3.4.1	Datasets and evaluation metrics . . . . .	40
3.4.2	Implementation details . . . . .	42
3.4.3	Training details . . . . .	43
3.4.4	Results and discussion . . . . .	44
3.5	Conclusion . . . . .	59
<b>4</b>	<b>On the Metrics for Evaluating Monocular Depth Estimation</b>	<b>61</b>
4.1	Introduction . . . . .	61
4.2	Related Work . . . . .	63
4.2.1	2D Object Detection . . . . .	63
4.2.2	3D Object Detection . . . . .	63
4.3	Methods . . . . .	66
4.3.1	Pseudo-LiDAR Generation . . . . .	66
4.3.2	3D object detectors . . . . .	72
4.3.3	Depth estimation methods . . . . .	73
4.4	Experimental Results . . . . .	73
4.4.1	Datasets and evaluation metrics . . . . .	73
4.4.2	Experiment Protocol . . . . .	75
4.4.3	Testing depth estimation . . . . .	76
4.4.4	Evaluation of sampling methods . . . . .	77
4.4.5	Testing 3D object detection . . . . .	79
4.4.6	Depth estimation <i>vs.</i> 3D object detection . . . . .	81
4.5	Conclusion . . . . .	93

## Contents

---

<b>5 Conclusions</b>	<b>95</b>
<b>A Appendix</b>	<b>99</b>
<b>Bibliography</b>	<b>115</b>

# List of Figures

1.1	SAE Levels of Driving Automation. Source: <sae.org> . . . . .	2
1.2	Pictorial representation of the sensor positioning and their applications. Reproduced from [143]. . . . .	3
1.3	Comparison between LiDAR, RADAR, and Camera (covering the visible spectrum) systems. Figure reproduced from [116]. . . . .	4
1.4	Left: Examples of two stereo rigs reproduced from [137]. On top a hand-crafted stereo rig based on two twin cameras, in the bottom a commercial stereo rig. Right: Cameras are mounted with their respective image sensors aligned and separated by a given distance ( <i>baseline</i> ), so that image content overlaps and seems to be horizontally shifted (up to occlusions). The per-pixel content shift is known as <i>disparity</i> and, having the intrinsic parameters of one of the cameras, allows to compute depth by applying triangulation equations. . . . .	5
1.5	By using a monocular system in motion, acquiring overlapping images, we can apply structure-from-motion techniques (SfM) to recover 3D information of the observed scene. Example reproduced from [20]. . . . .	5
1.6	MDE using a CNN by Eigen <i>et al.</i> [30], image reproduced from their paper. . . . .	7
1.7	Top: RGB image with LiDAR points acquired in the same scene on top. The camera and the LiDAR must be calibrated (spatial registration and time synchronization). Bottom: Color-coded 3D information generated by densifying the raw point cloud captured by the LiDAR after its projection to the image plane (Top). . . . .	8
1.8	Data from Virtual KITTI dataset [5]. Top: Synthetic RGB image. Bottom: associated depth (color-coded) ground truth (z-buffer used during the image rendering process). Depth is shown just up to 80m. . . . .	8



## List of Figures

---

2.1	Top to bottom: RGB KITTI images; their depth ground truth (LiDAR); our monocular depth estimation. . . . .	12
2.2	Phase one: conditional backward passes (see main text). We also use skip connections linking convolutional and deconvolutional layers with equal spatial sizes. . . . .	16
2.3	Phase two: the pre-trained (DSC+DC) network is augmented by regression layers for fine-tuning, resulting in the (DSC-DRN) network for depth estimation. . . . .	17
2.4	Details of our CNN architecture at training time. . . . .	18
2.5	Top to bottom, twice: RGB image (KITTI); depth ground truth; our depth estimation. Depth estimation and G.T, expressed in meters, varies in the range [0,50]m as represented by the colorbar notation next to the figure. . . . .	22
2.6	Left to right: RGB image (KITTI), depth ground truth, Godard <i>et al.</i> [41] and our depth estimation results. In this figure, we show on the right side of the image that Godard <i>et al.</i> [41] results yield poor detection quality along with inaccurate depth estimation for specific relevant objects such as cars, tram or poles. On the other hand, our method provides a more accurate depth estimation which can be seen on the right most column. . . . .	24
2.7	Depth estimation results on Cityscapes validation and testing set images. This Cityscapes dataset is used for the task of semantic segmentation and we couldn't provide quantitative results as it doesn't have depth ground truth. <b>Note:</b> The validation set images are not used for training the network.	25
3.1	Training framework for MonoDEVSNet, <i>i.e.</i> , $\Psi(\theta; x)$ . We show the involved images, GT, weights, and losses. Red and blue lines are paths of real and virtual-world data, respectively. The discontinuous line is a common path.	35
3.2	Pyramidal architecture of $\theta^{\text{PYT}}$ . . . . .	43
3.3	Qualitative results on the (KR) Eigen <i>et al.</i> testing slit [30]. From top to bottom: input images, their LiDAR-based depth GT (interpolated for visualization using [95]), DORN, SharinGAN, PackNet-SfM, MonoDEVSNet VK_v1 and VK_v2. . . . .	49

3.4 Abs-rel and rms errors as a function of depth, for KR Eigen testing split [30]. The histogram of depth GT is shown with bars. We compare PackNet-SfM and MonoDEVSNet. . . . .	50
3.5 Qualitative results on KS data. From left to right: input images, PackNet-SfM, MonoDEVSNet. . . . .	51
3.6 Per-class abs-rel and rms errors for KS, computed by averaging over the pixels of each class, for PackNet-SfM and MonoDEVSNet. The % of pixels of each class is shown. . . . .	51
3.7 Point cloud representation on KR Eigen test split [30] and KS data from left to right. From top to bottom: input images, MonoDEVSNet textured point cloud. . . . .	52
3.8 Qualitative results of MonoDEVSNet on Make3D. From left to right: input images, depth GT, MonoDEVSNet. . . . .	53
3.9 Qualitative results on Apollostereo dataset. . . . .	54
3.10 Qualitative results of MonoDEVSNet on Cityscapes dataset. From left to right: input images, MonoDEVSNet estimated depth maps. . . . .	55
3.11 Failure cases in the depth map. . . . .	56
3.12 Comparison between MonoDEVSNet models trained with different network architectures. . . . .	58

4.1	MDE-based 3D object detection in a nutshell. Given an image ( $I$ ) we apply a MDE model ( $\Psi$ ) to generate its corresponding depth map ( $d$ ). With the intrinsic parameters, $\mathcal{K}$ , of the camera capturing the images, as well as a set of parameters, $\mathcal{L}$ , defining a LiDAR-inspired sampling procedure, the depth map can be converted to a 3D point cloud, $\hat{\rho}$ . Then, $\hat{\rho}$ is used to train and test a 3D object detector ( $\Xi$ ) originally developed to work with LiDAR point clouds. The 3D bounding boxes (BBs) of the detected objects are projected to the image just for visualization purposes, but training and inference of the object detectors are done on 3D point clouds. In case of directly working with LiDAR point clouds, only the 3D object detection in the bottom box (discontinuous blue) is required (we may not even have corresponding images to project the BBs). In case of working with stereo data, the processing within the top box (discontinuous red) consists of disparity estimation from the usual left-right stereo pair of images and the final depth computation. . . . .	67
4.2	Top: depth map. Mid-Bottom: white pixels would be sampled according to the LiDAR-inspired procedure, for $N_b = 64$ (Mid) and $N_b = 16$ (Bottom). . .	69
4.3	Point R-CNN 3D object detector [118]. . . . .	70
4.4	Voxel R-CNN 3D object detector [27]. . . . .	70
4.5	CenterPoint 3D object detector [144]. . . . .	70
4.6	MonoDELSNet-St uses stereo self-supervision and LiDAR supervision for training. See Fig. 3.1 to compare with MonoDEVSNNet (in this chapter called MonoDEVSNNet-SfM). . . . .	71
4.7	MonoDELSNet-SfM uses SfM self-supervision and LiDAR supervision for training. See Fig. 3.1 to compare with MonoDEVSNNet (in this chapter called MonoDEVSNNet-SfM). . . . .	71

4.8 Experiment protocol. DE stands for depth estimation (monocular or stereo based). TS and VS stand for training and validation set, respectively. In our experiments, the DE TS is the training set of the Eigen *et al.* [30] split, the 3D-OD TS and 3D-OD VS are the training and validation sets of the Chen *et al.* [13] split. Thus, the datasets are from KITTI benchmark [38]. Eval DE is based on the standard metrics to evaluate depth estimation (abs-rel, rms, *etc.*), while Eval 3D-OD is based on metrics to assess 3D object detection accuracy ( $AP_{BEV}$ ,  $AP_{3D}$ ). Blue paths work at training time, while red paths work at inference/validation time. In the datasets, images and LiDAR are in correspondence, so that each image has an aligned 3D point cloud. Thus, the same 3D BBs can be used with LiDAR and Pseudo-LiDAR data. Refer to the main text for details. . . . . 74

4.9 3D object detection based on the different MDE methods combined with Point R-CNN. Red BBs are ground truth, green ones are detections. Detections are shown for cars, pedestrians, and cyclists. . . . . 85

4.10 3D object detection based on MonoDELSNet-St combined with either Point R-CNN, or Voxel R-CNN, or CenterPoint. Red BBs are ground truth, green ones are detections. For Point R-CNN and CenterPoint detections are shown for cars, pedestrians, and cyclists; while Voxel R-CNN only detects cars. . . . . 86

4.11 3D object detection based on MonoDELSNet-St combined with Point R-CNN. Red BBs are ground truth, green ones are detections. Detections are shown for cars, pedestrians, and cyclists. . . . . 87

4.12 3D object detection based on MonoDELSNet-St combined with Voxel R-CNN. Red BBs are ground truth, green ones are detections. Detections are shown for cars. . . . . 88

4.13 3D object detection based on MonoDELSNet-St combined with CenterPoint. Red BBs are ground truth, green ones are detections. Detections are shown for cars, pedestrians, and cyclists. . . . . 89

4.14 Comparing rankings: abs-rel (MDE) vs. $AP_{BEV} - mod$ working at high resolution. In the mid column, we have ordered the MDE models from best (top/1) to worse (bottom/8). Then, we have replicated the mid columns as left (voxel-based detectors, which produce the same ranking) and right (Point R-CNN) columns. Afterwards, we have connected the models in left and right columns to its ranking number according to $AP_{BEV} - mod$ . Thus, a perfect correspondence between rankings shows as parallel arrows, and the lower correspondence the more arrows crossing each other. . . . .	90
4.15 Analogous to Fig. 4.14 for low resolution. . . . .	90
4.16 Analogous to Fig. 4.14 using rms metric. . . . .	91
4.17 Analogous to Fig. 4.15 using rms metric. . . . .	91
4.18 Analogous to Fig. 4.14 using $\delta < 1.25$ metric. . . . .	92
4.19 Analogous to Fig. 4.15 using $\delta < 1.25$ metric. . . . .	92

# List of Tables

- 2.1 Results on Eigen *et al.*'s KITTI split. DRN - Depth regression network, DC-DRN - Depth regression model with pretrained classification network, DSC-DRN - Depth regression network trained with the conditional flow approach. Evaluation metrics as follows, rel: avg. relative error, sq-rel: square avg. relative error, rms: root mean square error, rms-log: root mean square log error,  $\log_{10}$ : avg.  $\log_{10}$  error,  $\delta < \tau$ : % of pixels with relative error  $< \tau$  ( $\delta \geq 1$ ;  $\delta = 1$  no error). Godard - K means using KITTI for training, and "+ CS" adding Cityscapes too. Bold stands for **best**, underline for second best. . . . . 23
- 3.1 Comparing ResNet and HRNet as backbone for  $\Psi(\theta; x)$ , training only on SfM self-supervision (relative scale) using the framework in [42]. MW column stands for millions of  $\theta^{\text{enc}}$  weights to be learnt. FPS stands for *frames per second* as required by  $\Psi(\theta; x)$  to process  $x$ , while GFLOPS refers to the *giga floating-point operations per second* required by  $\theta^{\text{enc}}$ ; in both cases using an NVIDIA RTX 2080Ti GPU. The  $1.25^n$  columns,  $n \in \{1, 2\}$ , refer to the  $\tau$  in the usual  $\delta < \tau$  accuracy metrics. In all the tables of Sect. 3.4, bold stands for **best** and underline for second-best. (\*) Currently, HRNet branches do not run in parallel in PyTorch, thus, compromising speed. . . 42
- 3.2 Relative depth results up to 80m on the (KR) Eigen *et al.* [30] testing split. These methods rely on SfM self-supervision. In addition, methods in gray use DA supported by VK. <sup>(1)</sup> MonoDepth2 is based only on SfM self-supervision. . . . . 44

3.3 *Absolute depth* results up to 80m on the (KR) Eigen *et al.* [30] testing split. We divide the results into four blocks. From top to bottom, the blocks refer to: methods based on LiDAR supervision, only virtual-world supervision, stereo self-supervision, SfM self-supervision. In these blocks, we remark best and second-best results per block. Methods in gray use DA supported by VK. We remark some additional comments: <sup>(1)</sup> in addition to LiDAR supervision, it also uses stereo self-supervision; <sup>(2)</sup> it uses stereo and SfM self-supervision; <sup>(3)</sup> in this case, the MDE network is pre-trained on Cityscapes dataset [18] and then fine-tuned on KITTI. . . . . 45

3.4 *Absolute depth* ablative results of MonoDEVSNets (VK\_v2) up to 80m on the (KR) Eigen testing split [30]. Rows 1-6 show the progressive use of the components of our proposal (each row adds a new component). 50/50 refers to mini-batches of 50% real-world samples and 50% or virtual-world ones; not using 50/50 (rows 1-2) means that we alternate mini-batches of pure real- or virtual-world samples. Row 7 corresponds to a simplification of the SfM self-supervised loss.  $\vartheta^S$  (rows 8-9) refers to a GAN-based DA approach. LB (lower bound, row 10) indicates the use of only virtual-world data. UB (upper bound, row 12) indicates the use of KITTI LiDAR-based supervision instead of virtual-world data. Rows 11 and 13 show the difference of our best model (All) with respect to LB and UB, respectively.  $\uparrow D$  means that All is  $D$  units better, while  $\downarrow D$  means that it is  $D$  units worse. All/W18 (row 14) and All/W32 (row 15) refer to using the All configuration by relying on HRNet-W18 and HRNet-W32, respectively. . . . . 46

3.5 *Absolute depth* results on Make3D testing set. All the shown methods use Make3D only for testing (generalization), except <sup>(1)</sup> which fine-tunes on Make3D training set too. . . . . 53

3.6 Absolute depth. We provide final experimental results for different versions of three different architectures. In the upper block of the table we have used the same training set as in previous experiments, *i.e.*, 12K/12K from KR/VK\_v2. In the bottom block, as is usual in the literature, we use all the available training data, *i.e.*,  $\sim 40K$  and  $\sim 22K$ , respectively. . . . . 57

4.1 Models to produce depth from images. For (\*) a stereo rig is needed at testing time, while for the rest just a monocular system. For ( $\dagger$ ) absolute depth is obtained via ego-vehicle velocity. MW column stands for millions of model weights. In all cases, these are camera-based models to obtain Pseudo-LiDAR point clouds. . . . . 72

4.2	Instances of <i>car</i> , <i>pedestrian</i> , and <i>cyclist</i> , in KITTI 3D obj. detect. benchmark.	72
4.3	Working resolutions (in pixels) of the different models we consider in this chapter. (*) It runs on $[1224 - 1242] \times [370 - 376]$ pixels. . . . .	77
4.4	Absolute depth results (up to 80m) for the images of the validation set of the Chen <i>et al.</i> [13] split. We use the models trained by the corresponding authors for PackNet, MonoDepth2, and AdaBins, the rest are trained by us. All the models are based on the same training and validation sets. SDNet must be considered an upper-bound since it uses a stereo pair at testing time, while all the other models work on single images. Bold stands for <b>best</b> and underline for <u>second best</u> . . . . .	78
4.5	3D car detection results ( $AP_{BEV}$ , $AP_{3D}$ ) by using different strategies for sampling depth maps, as well as different 3D object detectors (Point R-CNN, Voxel R-CNN, CenterPoint). The sampling is applied to depth maps obtained from MDE models (AdaBins, MonoDELNet-SfM). $\rho$ stands for Pseudo-LiDAR obtained by using all the pixels of the corresponding depth maps, while $\hat{\rho}$ stands for Pseudo-LiDAR obtained by applying a LiDAR-inspired sampling to the same depth maps (Fig. 4.2, with $N_b = 64$ ). L stands for LiDAR point clouds, thus, for the three detectors this setting produces upper-bound results. These LiDAR point clouds have a one-to-one correspondence with the RGB images used to generate the depth maps with the MDE models. Thus, after sampling these maps, the 3D BBs used with L as object supervision, can be also used with $\rho$ and $\hat{\rho}$ . The images used to generate the depth maps (and so $\rho$ and $\hat{\rho}$ ) are from the training set of the Chen <i>et al.</i> [13] split. The LiDAR training data for the setting L is from the same training set. In all cases, the validation is performed on the validation set of this split. . . . .	80
4.6	Results on 3D car detection ( $AP_{BEV}$ , $AP_{3D}$ ) using Point R-CNN. We complement these results with two depth estimation metrics (abs-rel, rms). Taking Fig. 4.8 as reference, $AP_{BEV}$ and $AP_{3D}$ play the role of Eval 3D-OD, while abs-rel and rms are part of Eval DE. Raw LiDAR P. Cloud refers to training with actual LiDAR 3D point clouds, and SDNet estimates depth from stereo images. Thus, these two methods can be seen as upper-bounds for the rest, which generate the corresponding 3D point clouds after performing MDE. Focusing on the MDE models, bold stands for <b>best</b> and underline for <u>second best</u> within each resolution block. . . . .	82
4.7	Analogous to Table 4.6 but using Voxel R-CNN. . . . .	83



## List of Tables

---

4.8 Analogous to Table 4.6 but using CenterPoint. . . . .	84
---	----

# 1 Introduction

*“Self-driving cars are the natural extension of active safety and obviously something we think we should do.”*

– Elon Musk

---

## 1.1 Autonomous Driving

Before the 1940s, elevators had human *drivers* and people was concerned about being on-board new *driverless* ones. However, during a strike of drivers in New York, the situation was so annoying that people adopted the driverless technology. In the 20th century, all commercial flights had cruise control in the air and a highly automated system for landing in extremely adverse weather conditions (known as CAT III AUTOLAND procedure). The same automation trend can be seen for train, tram, and underground mobility. Therefore, it is natural to aim bringing high levels of automation for road vehicles too. In fact, since the 1980s, Autonomous Vehicles (AVs) on regular roads appear in Science Fiction books and films. Obviously, this is rather challenging since AVs do not move on rails, and have to deal with other vehicles and different traffic participants (pedestrians, bicyclist, *etc.*) in the wild; thus, requiring such high levels of artificial intelligence (AI), that at the beginning of this century we were extremely far away.

Fortunately, during the last decade, several research labs from universities as well as the automotive and AI industries have been pushing forward the state of the art of AVs. In the following decades, more dramatic advances are foreseen given the worldwide interest in moving towards mobility solutions based on efficient AVs.

In fact, according to the World Health Organization (WHO), in the 20th century, 60 million people died due to traffic accidents, 1.3 million per year [122]. About the 93% of the accidents were due to driver errors, which include violating speed limits, being drugged, being distracted, and making wrong predictions. Therefore, AVs are developed under the assumption that removing humans from the driving loop will increase safety even if the AVs are not yet perfect [2]. Beyond traffic accidents, AVs

## Chapter 1. Introduction

can help to reduce CO2 emissions, traffic congestion (30% fewer vehicles on the road), reducing transportation costs by a 40% (cars, fuel, and infrastructure), and improving livability [60]. As a matter of fact, the EU considers AVs as one of the top-ten technologies that will change our lives [135].

**SAE J3016™ LEVELS OF DRIVING AUTOMATION™**  
 Learn more here: [sae.org/standards/content/j3016\\_202104](http://sae.org/standards/content/j3016_202104)

Copyright © 2021 SAE International. The summary table may be freely copied and distributed AS-IS provided that SAE International is acknowledged as the source of the content.

	SAE LEVEL 0™	SAE LEVEL 1™	SAE LEVEL 2™	SAE LEVEL 3™	SAE LEVEL 4™	SAE LEVEL 5™
What does the human in the driver's seat have to do?	You are driving whenever these driver support features are engaged – even if your feet are off the pedals and you are not steering			You are not driving when these automated driving features are engaged – even if you are seated in "the driver's seat"		
	You must constantly supervise these support features; you must steer, brake or accelerate as needed to maintain safety			When the feature requests, you must drive	These automated driving features will not require you to take over driving	
Copyright © 2021 SAE International.						
	These are driver support features			These are automated driving features		
What do these features do?	These features are limited to providing warnings and momentary assistance	These features provide steering OR brake/acceleration support to the driver	These features provide steering AND brake/acceleration support to the driver	These features can drive the vehicle under limited conditions and will not operate unless all required conditions are met	This feature can drive the vehicle under all conditions	
Example Features	<ul style="list-style-type: none"> <li>• automatic emergency braking</li> <li>• blind spot warning</li> <li>• lane departure warning</li> </ul>	<ul style="list-style-type: none"> <li>• lane centering OR</li> <li>• adaptive cruise control</li> </ul>	<ul style="list-style-type: none"> <li>• lane centering AND</li> <li>• adaptive cruise control at the same time</li> </ul>	<ul style="list-style-type: none"> <li>• traffic jam chauffeur</li> </ul>	<ul style="list-style-type: none"> <li>• local driverless taxi</li> <li>• pedals/steering wheel may or may not be installed</li> </ul>	<ul style="list-style-type: none"> <li>• same as level 4, but feature can drive everywhere in all conditions</li> </ul>

Figure 1.1 – SAE Levels of Driving Automation. Source: <sae.org>

Unfortunately, developing AVs is not trivial and we cannot go from human-driven vehicles to fully-automated ones in just one step, and in a short period of time. Accordingly, a committee of experts from the Society of Automotive Engineers (SAE) categorized road vehicles into six groups regarding their level of automation, as shown in Fig. 1.1. Level 0 corresponds to human driving with intelligent driver assistance, while Level 5 corresponds to fully replacing human drivers everywhere and in all conditions. As a reference, popular Tesla vehicles are somewhere between Levels 2 and 3.

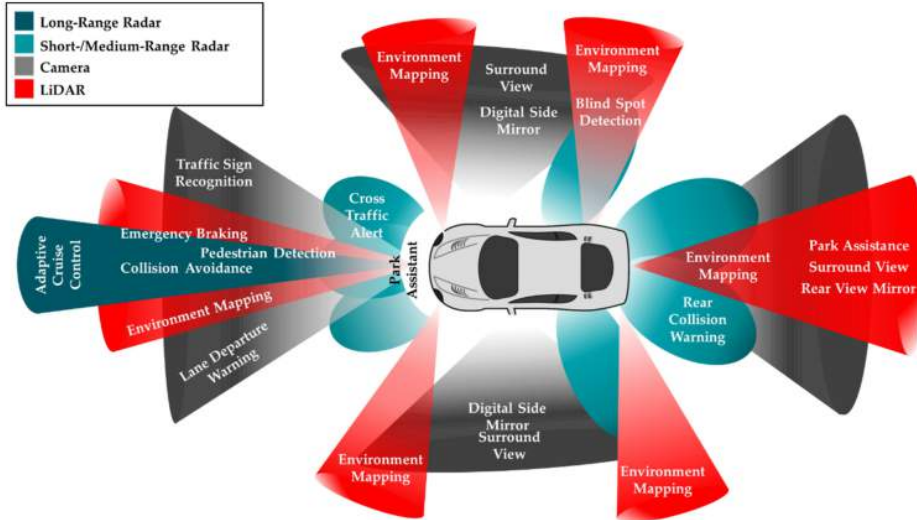


Figure 1.2 – Pictorial representation of the sensor positioning and their applications. Reproduced from [143].

## 1.2 Sensing the Environment

In order to reach the different levels of automation, AVs must be equipped with sensors that capture information about the environment surrounding them. Sensors can be passive such as cameras working on the visible or far-infrared (FIR) spectrum, as well as inertial measurement units (IMUs), or active such as the radio detecting and ranging (RADAR), the light detecting and ranging (LiDAR), and global navigation satellite systems (GNSS). The usual placement of the sensors with their respective target applications is shown in Fig. 1.2.

Ideally, we would like to use only camera sensors since they are passive, relatively easy to install and maintain, and at least two orders of magnitude cheaper than others such as LiDARs. Moreover, the visual quality of the captured images, the robustness against adversarial illumination conditions, the frame rate, and the resolution, are features constantly improved since cameras are also key assets in the global consumer market (smartphones, personal computers, gaming, etc.). Images contain a dense and rich information about the captured scene. The bottleneck, however, is to being able to interpret its semantic and geometric content in real-time and, in the case of AVs, with on-board processing hardware. In the last years, GPU technology and Convolutional

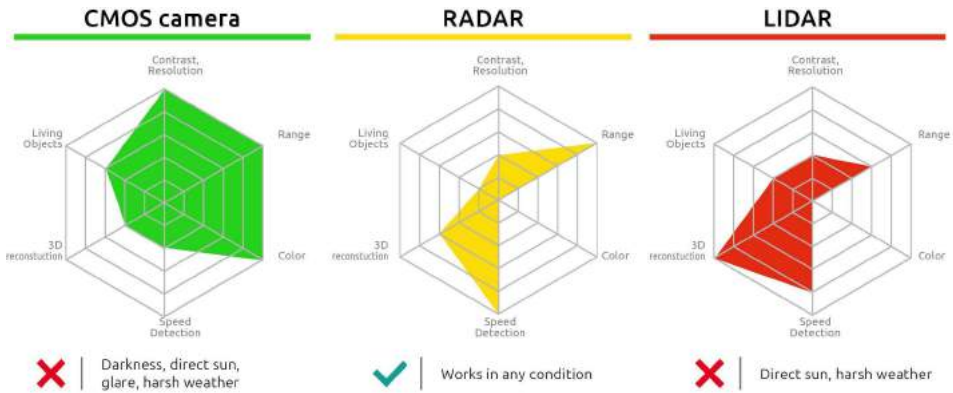


Figure 1.3 – Comparison between LiDAR, RADAR, and Camera (covering the visible spectrum) systems. Figure reproduced from [116].

Neural Networks (CNNs), have significantly increased the performance on visual tasks such as 2D object detection [102, 106] and semantic segmentation [10, 51]. However, when thinking about measuring distances or providing 3D bounding boxes of traffic participants (*e.g.*, vehicles and pedestrians), which is essential for AVs, sensors such as LiDAR [27, 118, 144] and RADAR [84, 87, 114] are more accurate and cover a longer distance than camera-based approaches such as stereo rigs [19, 73, 131]. Therefore, in the sake of reliability, most AVs of today are based on the fusion of heterogeneous sensor suites that include cameras, LiDARs, and RADARs [143]. Figure 1.3 summarizes the pros and cons of this type of sensors according to [116]. On the other hand, note that not using cameras would make much more difficult to solve crucial tasks such as detecting horizontal and vertical traffic information (regulatory signals and lights, lane marks, etc.) since, nowadays, these have been designed to be well-seen by human drivers. Tracking traffic participants (vehicles, pedestrians) [123, 155] or detecting their intentions [47] can be extremely difficult, if not impossible, without using cameras. Some AVs require pre-generated high-definition (HD) 3D maps to navigate [4]. In this case, the global positioning system (GPS) and inertial measurement units (IMUs) complement cameras, LiDARs, and RADARs, to compute the 3D ego-motion and localization of the AV.

### 1.3 Vision-based Depth Computation

Figure 1.3 tell us that working towards vision-based 3D reconstruction would allow us to simplify on-board sensor suites by, for instance, not using LiDAR technology; after

### 1.3. Vision-based Depth Computation

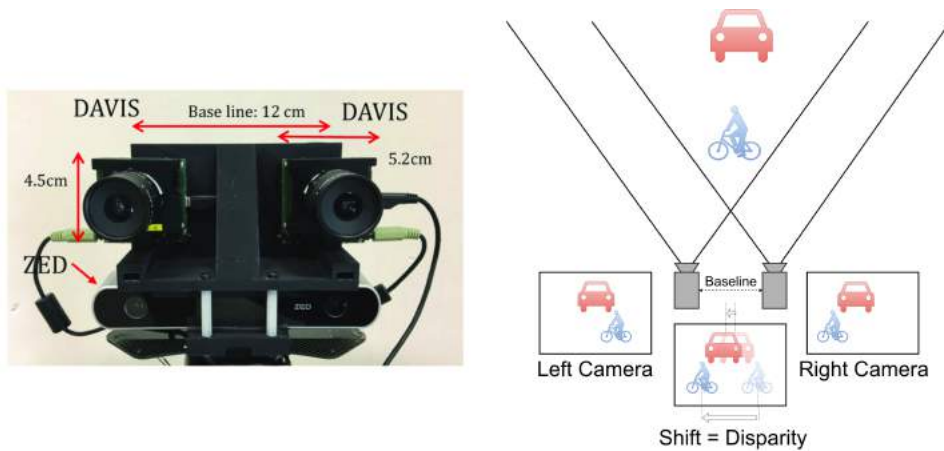


Figure 1.4 – Left: Examples of two stereo rigs reproduced from [137]. On top a hand-crafted stereo rig based on two twin cameras, in the bottom a commercial stereo rig. Right: Cameras are mounted with their respective image sensors aligned and separated by a given distance (*baseline*), so that image content overlaps and seems to be horizontally shifted (up to occlusions). The per-pixel content shift is known as *disparity* and, having the intrinsic parameters of one of the cameras, allows to compute depth by applying triangulation equations.

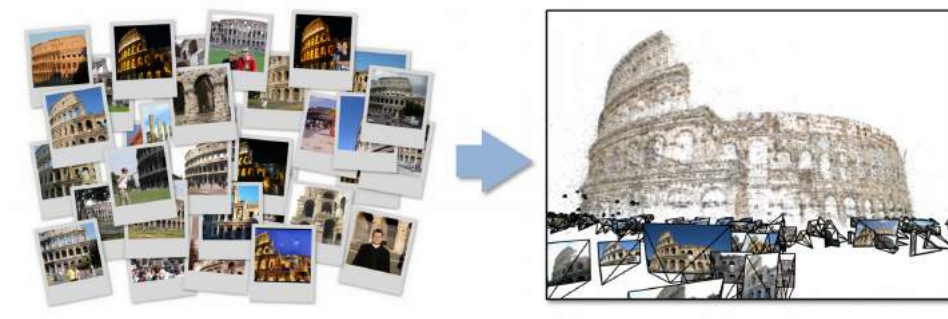


Figure 1.5 – By using a monocular system in motion, acquiring overlapping images, we can apply structure-from-motion techniques (SfM) to recover 3D information of the observed scene. Example reproduced from [20].

all, the core sense in human driving is vision. In the end, we *only* need to understand the 3D structure of the world surrounding the vehicle (which is local) up to the level of allowing AVs to perform the next maneuver safely. This translates to being able to obtain vision-based depth for complementing vision-based semantics. For indoor controlled environments, depth from vision was approached with the help of additional active elements such as structured light [31, 63], or time-of-flight technology [50]. However, these approaches are not suitable for AVs driving outdoors in regular traffic conditions. In fact, vision-based on-board depth computation has been traditionally based on a stereo rig (Fig. 1.4), *i.e.* a physical system with two properly aligned cameras (left and right) separated by a baseline (*e.g.*, 12cm, 24cm) which influences the rig's working range (*i.e.*, where depth computation is reliable). Basically, depth computation from a stereo rig involves an offline calibration process of the left and right cameras [149], and a real-time feature matching between left and right images to compute the so-called *disparity map*, either using traditional methods [148] or deep learning [68]. Using depth-from-stereo, higher-level geometric representations of the imaged scene have been also used in AVs. In particular, relying on the so-called *stixels* [1, 55, 90], which can be combined with semantic information too [19, 113]. In practice, most of the inaccuracies of stereo-based depth computation come from feature matching errors and from the deterioration of the parameters obtained by the offline calibration procedure, which usually requires on-board corrections [133].

On the other hand, some vehicles are only equipped with a single camera to run driver assistance functionalities (SAE Level 0 or 1). In these cases, depth has been usually computed in a more indirect or adhoc manner. For instance, assuming that the vehicle moves in a flat road and computing the horizon line it is possible to estimate distances to other traffic participants using a monocular system [93]. It is also possible to learn depth ranges even using handcrafted features or visual semantics and additional restrictions coming from projective geometry [9, 57, 67, 77]. Besides, since an on-board monocular system moves with the ego-vehicle, a priori, structure-from-motion (SfM) [160] (Fig. 1.5) can also be used to compute depth. However, inaccuracies due to camouflage (objects moving as the camera may not be distinguished from background), visibility changes (occlusion changes, non-Lambertian surfaces), static-camera cases (*i.e.*, stopped ego-vehicle), textureless areas, and scale ambiguity (depth could only be estimated up to an unknown scale factor), limits the usefulness of SfM as a stand-alone paradigm to obtain depth from an on-board single camera, especially if we aim at obtaining dense enough depth maps.

## 1.4. Learning to Compute Depth from Single Images

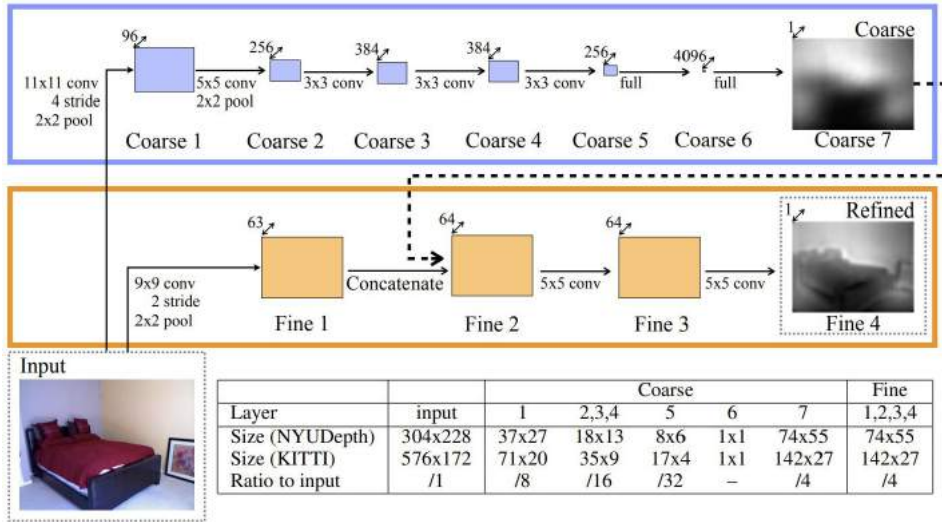


Figure 1.6 – MDE using a CNN by Eigen *et al.* [30], image reproduced from their paper.

## 1.4 Learning to Compute Depth from Single Images

Since purely relying on geometric principles (stereo, SfM) for vision-based depth estimation shows the above outlined problems, related research over the past few years has pivoted to a different paradigm. In particular, vision-based depth estimation is now approached as a machine learning problem. More specifically, the objective is to train a model capable of estimating the depth at which the elements of a scene are located with respect to the camera that captures it, at the pixel level. In other words, to train a monocular depth estimation (MDE) model. This task can be tackled thanks to Convolutional Neural Networks (CNNs). To the best of our knowledge, the first attempt to use CNNs for MDE comes from Eigen *et al.* [30] (Fig. 1.6), since then many other researchers have been focusing on this setting [3, 6, 33, 37, 41, 42, 44–46, 48, 49, 52, 53, 66, 69, 78, 91, 101, 110, 138, 146, 152, 154, 160]. Note that these are just some significant references we can find in the literature and that we cite as related work in further chapters of this PhD document, but not a complete list. As we will see, beyond the MDE accuracy reached by each proposal, which, fortunately, is constantly increasing, one major question is the paradigm used to train the corresponding CNNs.

A priori, the most favorable case for training a CNN to perform MDE corresponds to the situation in which we are able to collect images calibrated with LiDAR point



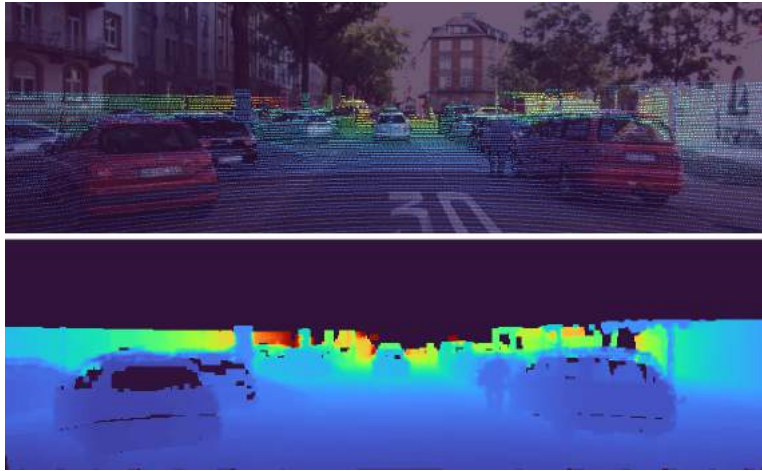


Figure 1.7 – Top: RGB image with LiDAR points acquired in the same scene on top. The camera and the LiDAR must be calibrated (spatial registration and time synchronization). Bottom: Color-coded 3D information generated by densifying the raw point cloud captured by the LiDAR after its projection to the image plane (Top).

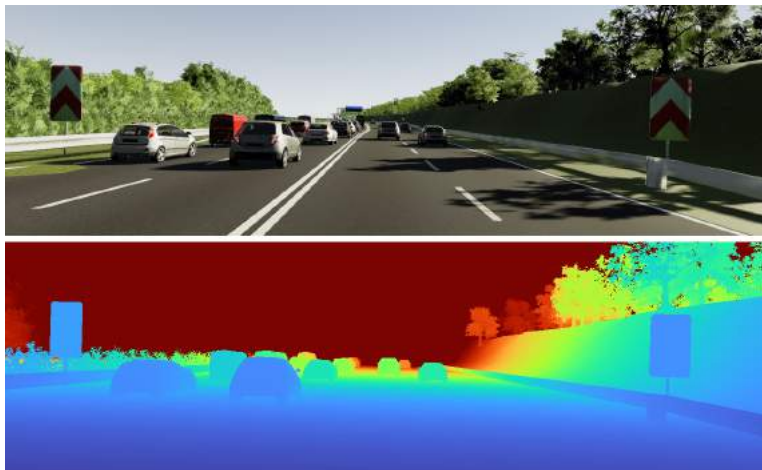


Figure 1.8 – Data from Virtual KITTI dataset [5]. Top: Synthetic RGB image. Bottom: associated depth (color-coded) ground truth (z-buffer used during the image rendering process). Depth is shown just up to 80m.

clouds (Fig. 1.7 Top). Since LiDAR point clouds can be densified in image space (Fig. 1.7 Bottom), LiDAR can act as *supervision* (*i.e.*, as *ground truth*) for training the MDE CNN. On the other hand, the most challenging situation is when, at training time, only a monocular system is available and, then, using *self-supervision* from SfM is the only hope to train the MDE CNN, which is really challenging. Another training setting consists in having a calibrated stereo rig to collect self-supervision. In fact, then SfM and stereo self-supervision can be combined. We will cite and review works following these different approaches within the remaining chapters of this PhD.

## 1.5 PhD Objective and Outline

Once established MDE based on CNNs as a key task for developing AVs, we can state that **the overall goal of this PhD is to study methods for improving CNN-based MDE accuracy under different training settings**. More specifically, in this PhD we address different research questions organized per chapter as follows.

*Chapter 2.* When we started to work in this PhD, state-of-the-art methods for MDE were already based on CNNs. In fact, a promising line of work consisted in using image-based semantic supervision (*i.e.*, pixel-level class labels) while training CNNs for MDE using LiDAR-based supervision (*i.e.*, depth). It was common practice to assume that the same raw training data are complemented by both types of supervision, *i.e.*, with depth and semantic labels. However, in practice, it was more common to find heterogeneous datasets with either only depth supervision or only semantic supervision. Therefore, **our first work on MDE was to research if we could train CNNs for depth estimation by leveraging the depth and semantic information from heterogeneous datasets**. In fact, as we will show in this chapter, this is indeed possible, and we surpassed the state-of-the-art results, on MDE at the time we did this research. To achieve our results, we proposed a particular CNN architecture and a new training protocol.

*Chapter 3.* After performing the research of previous chapter, it was clear that the upper-bound setting to train CNN-based MDE models consists in using LiDAR data as supervision. However, it would be cheaper and more scalable if we would be able to train such models from monocular sequences. Obviously, this is far more challenging, but worth to research. As we have mentioned before, training MDE models using monocular sequences is possible by relying on SfM principles to generate self-supervision. Nevertheless, problems of camouflaged objects, visibility changes, static-camera intervals, textureless areas, and scale ambiguity, diminish the usefulness of such self-supervision. In this chapter, **we perform monocular depth estimation by virtual-world supervision (MonoDEVs) and real-world SfM self-supervision. We compensate the SfM self-supervision limitations by leveraging virtual-world images with accurate semantic and depth supervision (Fig. 1.8), as well as addressing the virtual-to-real do-**

**main gap.** At the moment of finishing this research, MonoDEVSNets outperformed previous MDE CNNs trained on monocular and even stereo sequences. We have publicly released MonoDEVSNets at <https://github.com/HMRC-AEL/MonoDEVSNets>.

*Chapter 4.* During the research conducted in previous chapters, we had to compare quantitative and qualitative results between different MDE models. However, the targeted task is not MDE. Instead, MDE is performed to produce 3D information that can be used in downstream tasks related to on-board perception of AVs or driver assistance. Therefore, **the question that arose was whether the standard metrics for MDE assessment are a good indicator for future MDE-based driving-related perception tasks.** This is the question that we address in this chapter. In particular, we take the task of 3D object detection on point clouds as proxy of on-board perception. We train and test Point R-CNN [118], Voxel R-CNN [27], and CenterPoint [144] 3D object detectors, using point clouds coming from MDE models. We confront the ranking of object detection results with the ranking given by the depth estimation metrics of the MDE models. Fortunately, we conclude that, indeed, MDE evaluation metrics give rise to a ranking of methods which reflects relatively well the 3D object detection results we may expect. Among the different metrics, the so-called *absolute relative (abs-rel)* error seems to be the best for that purpose.

We have written these chapters to be self-contained, following the usual structure of a paper, *i.e.* abstract, introduction, related work, proposed method, experimental work with associated discussion, and summarizing conclusions. Chapter 5 summarizes the main contributions of this PhD and draws lines of continuation. Finally, we have added Appendix A listing the publications and patents done while working in this PhD; some of them from the work here presented, others from additional collaborations.

## 2 Monocular Depth Estimation by Learning from Heterogeneous Datasets

---

Depth estimation provides essential information to perform autonomous driving and driver assistance. In particular, monocular depth estimation is interesting from a practical point of view, since using a single camera is cheaper than many other options and avoids the need for continuous calibration strategies as required by stereo-vision approaches. State-of-the-art methods for monocular depth estimation are based on Convolutional Neural Networks (CNNs). A promising line of work consists of introducing additional semantic information about the traffic scene when training CNNs for depth estimation. In practice, this means that the depth data used for CNN training is complemented with images having pixel-wise semantic labels, which usually are difficult to annotate (e.g. crowded urban images). Moreover, so far it is common practice to assume that the same raw training data are associated with both types of ground truth, i.e., depth and semantic labels. The main contribution of this chapter is to show that this hard constraint can be circumvented, i.e., that we can train CNNs for depth estimation by leveraging the depth and semantic information from heterogeneous datasets. In order to illustrate the benefits of our approach, we combine KITTI depth and Cityscapes semantic segmentation datasets, outperforming state-of-the-art results on monocular depth estimation.

---

### 2.1 Introduction

Depth estimation provides essential information at all levels of driving assistance and automation. Active sensors such as a RADAR and LiDAR provide sparse depth information. Post-processing techniques can be used to obtain dense depth information from such sparse data [95]. In practice, active sensors are calibrated with cameras to perform scene understanding based on depth and semantic information. Image-based object detection [141], classification [159], and segmentation [51], as well as pixel-wise semantic segmentation [150] are key technologies providing such semantic information.

Since a camera sensor is often involved in driving automation, obtaining depth

## 2. Monocular Depth Estimation by Learning from Heterogeneous Datasets

---

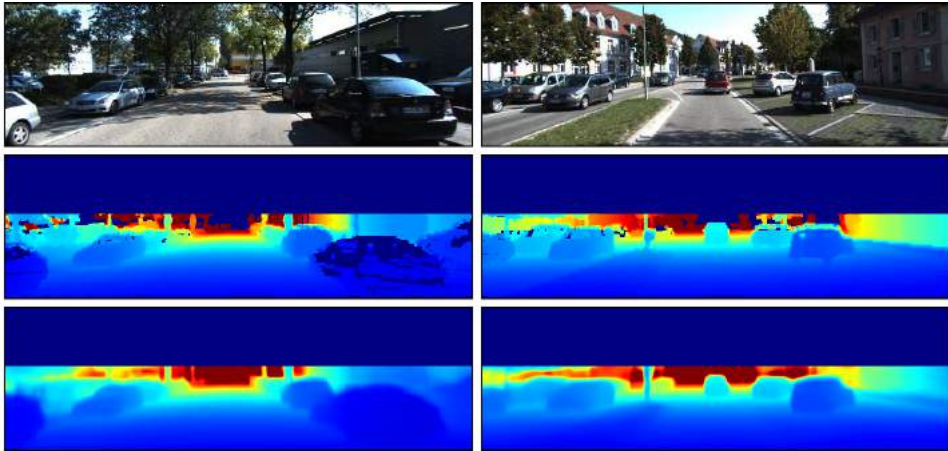


Figure 2.1 – Top to bottom: RGB Kitti images; their depth ground truth (LiDAR); our monocular depth estimation.

directly from it is an appealing approach and so has been a traditional topic from the very beginning of ADAS<sup>1</sup> development. Vision-based depth estimation approaches can be broadly divided in stereoscopic and monocular camera based settings. The former includes attempts to mimic binocular human vision. Nowadays, there are robust methods for dense depth estimation based on stereo vision [56], able to run in real-time [54]. However, due to operational characteristics, the mounting and installation properties, the stereo camera setup can loose calibration. This can compromise depth accuracy and may require to apply on-the-fly calibration procedures [23, 105].

On the other hand, monocular depth estimation would solve the calibration problem. Compared to the stereo setting, one disadvantage is the lack of the scale information, since stereo cameras allow for direct estimation of the scale by triangulation. Though, from a theoretical point of view, there are other depth cues such as occlusion and semantic object size information which are successfully determined by the human visual system [22]. These cues can be exploited in monocular vision for estimating scale and distances to traffic participants. Hence, monocular depth estimation can indeed support detection and tracking algorithms [9, 58, 93]. Dense monocular depth estimation is also of great interest since higher level 3D scene representations, such as the well-known Stixels [1, 55] or semantic Stixels [113], can be computed on top. Attempts to address dense monocular depth estimation can be found based on either super-pixels [111] or pixel-wise semantic segmentation [77]; but in both cases relying on hand-crafted

---

<sup>1</sup>ADAS: Advanced Driver Assistance Systems

features and applied to photos mainly dominated by static traffic scenes.

State-of-the-art approaches to monocular dense depth estimation rely on CNNs [6, 32, 41, 66]. Recent works have shown [61, 86] that combining depth and pixel-wise semantic segmentation in the training dataset can improve the accuracy. These methods require that each training image has per-pixel association with depth and semantic class ground truth labels, e.g., obtained from a RGB-D camera. Unfortunately, creating such datasets imposes a lot of effort, especially for outdoor scenarios. Currently, no such dataset is publicly available for autonomous driving scenarios. Instead, there are several popular datasets, such as KITTI containing depth [38] and Cityscapes [18] containing semantic segmentation labels. However, none of them contains both depth and semantic ground truth for the same set of RGB images.

Depth ground truth usually relies on a 360° LIDAR calibrated with a camera system, and the manual annotation of pixel-wise semantic classes is quite time consuming (e.g. 60-90 minutes per image). Furthermore, in future systems, the 360° LIDAR may be replaced by four-planes LIDARs having a higher degree of sparsity of depth cues, which makes accurate monocular depth estimation even more relevant.

Accordingly, in this chapter we propose a new method to train CNNs for monocular depth estimation by leveraging depth and semantic information from multiple *heterogeneous* datasets. In other words, the training process can benefit from a dataset containing only depth ground truth for a set of images, together with a different dataset that only contains pixel-wise semantic ground truth (for a different set of images). In Sect. 3.2 we review the state-of-the-art on monocular dense depth estimation, whereas in Sect. 3.3 we describe our proposed method in more detail. Sect. 3.4 shows quantitative results for the KITTI dataset, and qualitative results for KITTI and Cityscapes datasets. In particular, by combining KITTI depth and Cityscapes semantic segmentation datasets, we show that the proposed approach can outperform the state-of-the-art in KITTI (see Fig. 2.1). Finally, in Sect. 3.5 we summarize the presented work and draw possible future directions.

## 2.2 Related Work

First attempts to perform monocular dense depth estimation relied on hand-crafted features [67, 77]. However, as in many other Computer Vision tasks, CNN-based approaches are currently dominating the state-of-the-art, and so our approach falls into this category too.

Eigen *et al.* [30] proposed a CNN for coarse-to-fine depth estimation. Liu *et al.* [78] presented a network architecture with a CRF-based loss layer which allows end-to-end training. Laina *et al.* [69] developed an encoder-decoder CNN with a reverse Huber loss layer. Cao *et al.* [6] discretized the ground-truth depth into several bins (classes) for train-

## 2. Monocular Depth Estimation by Learning from Heterogeneous Datasets

---

ing a FCN-residual network that predicts these classes pixel-wise; which is followed by a CRF post-processing enforcing local depth coherence. Fu *et al.* [32] proposed a hybrid model between classification and regression to predict high-resolution discrete depth maps and low-resolution continuous depth maps simultaneously. Overall, we share with these methods the use of CNNs as well as tackling the problem as a combination of classification and regression when using depth ground truth, but our method also leverages pixel-wise semantic segmentation ground truth during training (not needed in testing) with the aim of producing a more accurate model, which will be confirmed in Sect. 3.4.

There are previous methods using depth and semantics during training. The motivation behind is the importance of object borders and, to some extent, object-wise consistency in both tasks (depth estimation and semantic segmentation). Arsalan *et al.* [86] presented a CNN consisting of two separated branches, each one responsible for minimizing corresponding semantic segmentation and depth estimation losses during training. Jafari *et al.* [61] introduced a CNN that fuses state-of-the-art results for depth estimation and semantic labeling by balancing the cross-modality influences between the two cues. Both [86] and [61] assume that for each training RGB image it is available pixel-wise depth and semantic class ground truths. Training and testing is performed in indoor scenarios, where a RGB-D integrated sensor is used (neither valid for outdoor scenarios nor for distances beyond 5 meters). In fact, the lack of publicly available datasets with such joint ground truths has limited the application of these methods outdoors. In contrast, a key of our proposal is the ability of leveraging disjoint depth and semantic ground truths from different datasets, which has allowed us to address driving scenarios.

The works introduced so far rely on deep supervised training, thus eventually requiring abundant high quality depth ground truth. Therefore, alternative unsupervised and semi-supervised approaches have been also proposed, which rely on stereo image pairs for training a disparity estimator instead of a depth one. However, at testing time the estimation is done from monocular images. Garg *et al.* [37] trained a CNN where the loss function describes the photometric reconstruction error between a rectified stereo pair of images. Godard *et al.* [41] used a more complex loss function with additional terms for smoothing and enforcing left-right consistency to improve convergence during CNN training. Kuznietsov *et al.* [66] proposed a semi-supervised approach to estimate inverse depth maps from the CNN by combining an appearance matching loss similar to the one suggested in [41] and a supervised objective function using sparse ground truth depth coming from LIDAR. This additional supervision helps to improve accuracy estimation over [41]. All these approaches have been challenged with driving data and are the current state-of-the-art.

Note that autonomous driving is pushing forward 3D mapping, where 360° LIDAR sensing plays a key role. Thus, calibrated depth and RGB data are regularly generated.

Therefore, although, unsupervised and semi-supervised approaches are appealing, at the moment we have decided to assume that depth ground truth is available; focusing on incorporating RGB images with pixel-wise class ground truth during training. Overall, our method outperforms the state-of-the-art (Sect. 3.4).

## 2.3 Proposed Approach for Monocular Depth Estimation

### 2.3.1 Overall training strategy

As we have mentioned, in contrast to previous works using depth and semantic information, we propose to leverage heterogeneous datasets to train a single CNN for depth estimation; *i.e.* training can rely on one dataset having only depth ground truth, and a different dataset having only pixel-wise semantic labels. To achieve this, we divide the training process in two phases. In the first phase, we use a multi-task learning approach for pixel-wise depth and semantic CNN-based classification (Fig. 2.2). This means that at this stage depth is discretized, a task that has been shown to be useful for supporting instance segmentation [124]. In the second phase, we focus on depth estimation. In particular, we add CNN layers that perform regression taking the depth classification layers as input (Fig. 2.3).

Multi-task learning has been shown to improve the performance of different visual models (*e.g.* combining semantic segmentation and surface normal prediction tasks in indoor scenarios; combining object detection and attribute prediction in PASCAL VOC images) [85]. We use a network architecture consisting of one common sub-net followed by two additional sub-net branches. We denote the layers in the common sub-net as DSC (depth-semantic classification) layers, the depth specific sub-net as DC layers and the semantic segmentation specific sub-net as SC layers. At training time we apply a conditional calculation of gradients during back-propagation, which we call *conditional flow*. More specifically, the common sub-net is always active, but the origin of each data sample determines which specific sub-net branch is also active during back-propagation (Fig. 2.2). We alternate batches of depth and semantic ground truth samples.

Phase one mainly aims at obtaining a depth model (DSC+DC). Incorporating semantic information provides cues to preserve depth ordering and per object depth coherence (DSC+DS). Then, phase two uses the pre-trained depth model (DSC+DC), which we further extend by regression layers to obtain a depth estimator, denoted by DSC-DRN (Fig. 2.3). We use standard losses for classification and regression tasks, *i.e.* cross-entropy and L1 losses respectively.



## 2. Monocular Depth Estimation by Learning from Heterogeneous Datasets

---

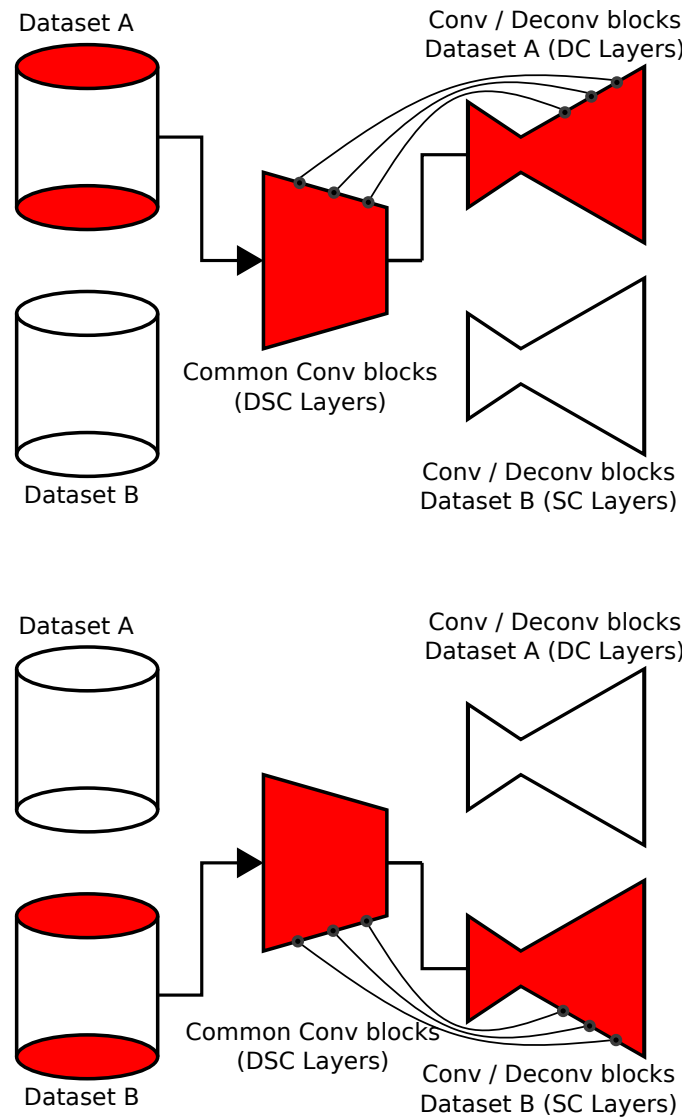


Figure 2.2 – Phase one: conditional backward passes (see main text). We also use skip connections linking convolutional and deconvolutional layers with equal spatial sizes.

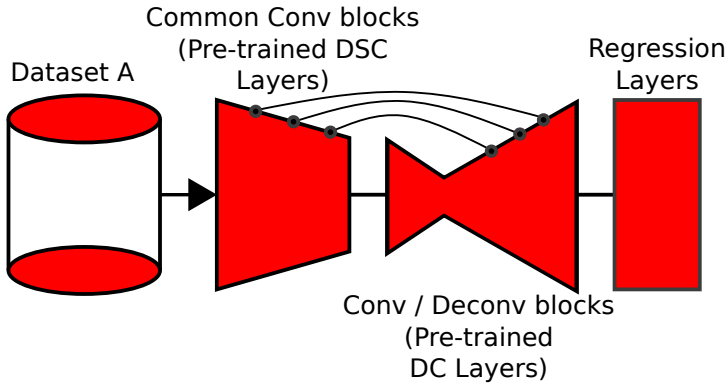


Figure 2.3 – Phase two: the pre-trained (DSC+DC) network is augmented by regression layers for fine-tuning, resulting in the (DSC-DRN) network for depth estimation.

#### 2.3.2 Network Architecture

Our CNN architecture is inspired by the FCNDROPOUT of Ros [109], which follows a convolution-deconvolution scheme. Fig. 2.4 details our overall CNN architecture. First, we define a basic set of four consecutive layers consisting of Convolution, Batch Normalization, Dropout and ReLu. We build *convolutional blocks* (ConvBlk) based on this basic set. There are blocks containing a varying number of sets, starting from two to four sets. The different sets of a block are chained and put in a pipeline. Each block is followed by an average-pooling layer. *Deconvolutional blocks* (DeconvBlk) are based on one deconvolution layer together with skip connection features to provide more scope to the learning process. Note that to achieve better localization accuracy these features originate from the common layers (DSC) and are bypassed to both the depth classification (DC) branch and the semantic segmentation branch (SC). In the same way we introduce skip connections between the ConvBlk and DeconvBlk of the added regression layers.

At phase 1, the network comprises 9 ConvBlk and 11 DeconvBlk elements. At phase 2, only the depth-related layers are active. By adding 2 ConvBlk with 2 DeconvBlk elements to the (DSC+DC) branch we obtain the (DSC-DRN) network. Here, the weights of the (DSC+DC)-network part are initialized from phase 1. Note that at testing time only the depth estimation network (DSC-DRN) is required, consisting of 9 ConvBlk and 7 DeconvBlk elements.



## 2.4 Experimental Results

### 2.4.1 Datasets

We evaluate our approach on KITTI dataset [38], following the commonly used Eigen *et al.* [30] split for depth estimation. It consists of 22,600 training images and 697 testing images, *i.e.* RGB images with associated LiDAR data. We have reprojected the LiDAR data taken from the Velodyne sensor (*i.e.* the 360° LiDAR). Here the LiDAR data are sparse, containing only 15% of the information w.r.t. the whole image. Hence we compute the dense depth maps from the sparse data according to Premebida [95]. We also introduce errors that correlate to the rotation of the Velodyne, motion of the vehicle and surrounding object, we add also incorrect data reading due to the occlusion at the object boundaries. We use a half down-sampled image  $188 \times 620$  for training and testing the model. Moreover, we use 2,975 images from Cityscapes dataset [18] with per-pixel semantic labels.

### 2.4.2 Implementation Details

We implement and train our CNN using MatConvNet [125], which we modified to include the conditional flow back-propagation. We use a batch size of 10 and 5 images for depth and semantic branches, respectively. We use ADAM, with a momentum of 0.9 and weight decay of 0.0003. The ADAM parameters are pre-selected as  $\alpha = 0.001$ ,  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ . Smoothing is applied via  $L_0$  gradient minimization [139] as pre-processing for RGB images, with  $\lambda = 0.0213$  and  $\kappa = 8$ . We include data augmentation consisting of small image rotations, horizontal flip, blur, contrast changes, as well as salt & pepper, Gaussian, Poisson and speckle noises. For performing depth classification we have followed a linear binning of 24 levels on the range [1,80]m.

### 2.4.3 Evaluation Metrics

For comparing the obtained results with state-of-the-art we compute several metrics commonly used in the literature, including:

Average relative error:

$$\text{abs-rel} := \frac{1}{n} \sum_{ij} \frac{|d_{gt} - d_p|}{d_{gt}} \quad (2.1)$$

## 2. Monocular Depth Estimation by Learning from Heterogeneous Datasets

---

Square average relative error:

$$\text{sq-rel} := \frac{1}{n} \sum_{ij} \frac{|d_{gt} - d_p|^2}{d_{gt}} \quad (2.2)$$

Root mean square error:

$$\text{rms} := \sqrt{\frac{1}{n} \sum_{ij} (d_{gt} - d_p)^2} \quad (2.3)$$

Root mean square log error:

$$\text{rms-log} := \sqrt{\frac{1}{n} \sum_{ij} (\log(d_{gt}) - \log(d_p))^2} \quad (2.4)$$

Average  $\log_{10}$  error:

$$\log_{10} := \frac{1}{n} \sum_{ij} |\log_{10} d_{gt} - \log_{10} d_p| \quad (2.5)$$

Accuracy with threshold  $\tau$  :

$$\text{Percentage (\%)} \max_{ij} \left( \frac{|d_p|}{d_{gt}}, \frac{|d_{gt}|}{d_p} \right) := \delta < \tau; \quad (2.6)$$

where  $ij$  stands for pixel coordinates running on all considered images,  $n$  is the total number of valid pixels (*i.e.* containing depth ground truth) in those images,  $d_{gt}$  and  $d_p$  are the ground truth and predicted depths at a given  $ij$ , respectively. The abs-rel error and the  $\delta < \tau$  are percentage measurements,  $\log_{10}$ , sq-rel and rms are reported in meters, and rms-log is similar (reported in meters) to rms but applied to logarithm depth values.

### 2.4.4 Results

We compare our approach to supervised methods such as Liu *et al.* [78] and Cao *et al.* [6], unsupervised methods such as Garg *et al.* [37] and Godard *et al.* [41], and semi-supervised method Kuznietsov *et al.* [66]. Liu *et al.*, Cao *et al.* and Kuznietsov *et al.* did not release their trained model, but they reported their results on the Eigen *et al.* split as us. Garg *et al.* and Godard *et al.* provide a Caffe model and a Tensorflow model respectively, trained on our same split (Eigen *et al.*'s KITTI split comes from stereo pairs). We have followed the author's instructions to run the models for estimating disparity and computing final depth by using the camera parameters of the KITTI stereo rig (focal length and baseline). In addition to the KITTI data, Godard *et al.* also added 22,973 stereo images coming from Cityscapes; while we use 2,975 from Cityscapes semantic segmentation challenge (19 classes). Quantitative results are shown in Table 2.1 for two different distance ranges, namely [1,50]m (cap 50m) and [1,80]m (cap 80m). As for the previous works, we follow the metrics proposed by Eigen *et al.* as shown in Section 2.4.3. Note how our method outperforms the state-of-the-art models in all metrics but one (being second best).

In Table 2.1 we also assess different aspects of our model. In particular, we compare our depth estimation results with (DSC-DRN) and without the support of the semantic segmentation task. In the latter case, we distinguish two scenarios. For the first one, which we denote as DC-DRN, we discard the SC subnet from the 1<sup>st</sup> phase so that we first train the depth classifier and later add the regression layers for retraining the network. In the second scenario, noted as DRN, we train the depth branch directly for regression, *i.e.* without pre-training a depth classifier. We see that for both cap 50m and 80m, DC-DRN and DRN are on par. However, we obtain the best performance when we introduce the semantic segmentation task during training. Without the semantic information, our DC-DRN and DRN do not yield comparable performance. This suggests that our approach can exploit the additional information provided by semantic information to learn a better depth estimator.

Fig. 2.5 shows qualitative results on KITTI. Note how well relative depth is estimated, also how clear are seen vehicles, pedestrians, trees, poles and fences. In addition, we show more qualitative results on KITTI compared with Godard *et al.* [41] in Fig. 2.6. Fig. 2.7 shows similar results for Cityscapes; illustrating generalization since the model was trained on KITTI. In this case, images are resized at testing time to KITTI image size ( $188 \times 620$ ) and the result is resized back to Cityscapes image size ( $256 \times 512$ ) using bilinear interpolation.

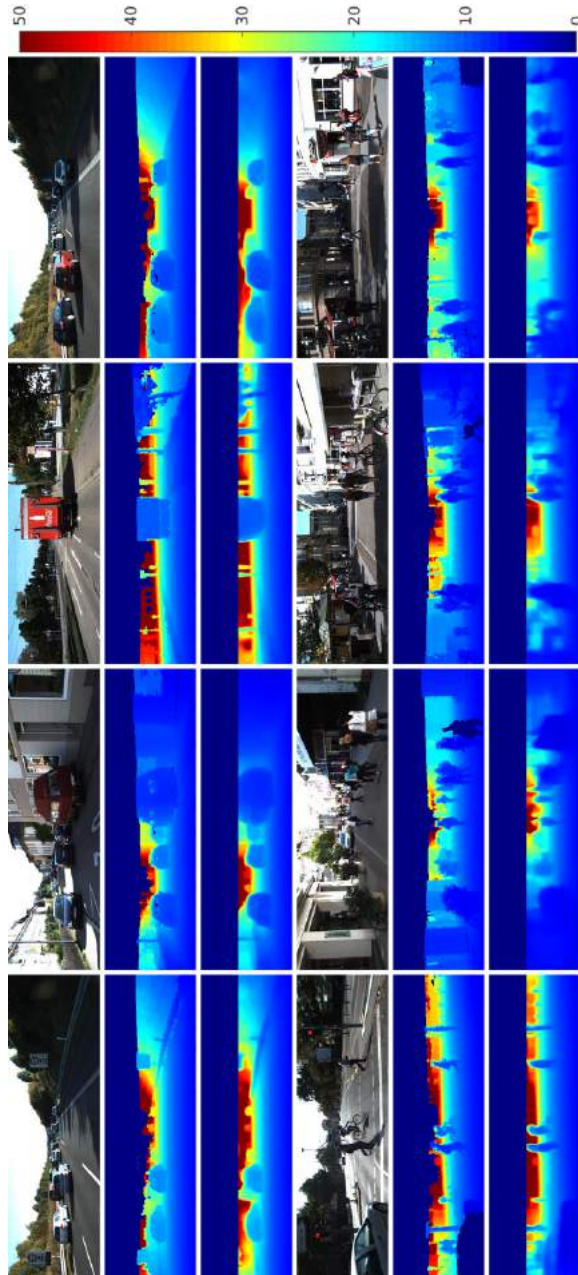


Figure 2.5 – Top to bottom, twice: RGB image (KITTI); depth ground truth; our depth estimation. Depth estimation and G.T, expressed in meters, varies in the range  $[0,50]$ m as represented by the colorbar notation next to the figure.

Table 2.1 – Results on Eigen *et al.*'s KITTI split. DRN - Depth regression network, DC-DRN - Depth regression model with pretrained classification network, DSC-DRN - Depth regression network trained with the conditional flow approach. Evaluation metrics as follows, rel: avg. relative error, sq-rel: square avg. relative error, rms: root mean square error, rms-log: root mean square log error,  $\log_{10}$ : avg.  $\log_{10}$  error,  $\delta < \tau$ : % of pixels with relative error  $< \tau$  ( $\delta \geq 1$ ;  $\delta = 1$  no error). Godard – K means using KITTI for training, and " + CS " adding Cityscapes too. Bold stands for **best**, underline for second best.

Approaches	metrics	Lower the better					Higher the better			
		cap (m)	abs-rel	sq-rel	rms	rms-log	$\log_{10}$	$\delta < 1.25$	$\delta < 1.25^2$	$\delta < 1.25^3$
Liu fine-tune [78]		80	0.217	1.841	6.986	0.289	-	0.647	0.882	0.961
Godard – K [41]		80	0.155	1.667	5.581	0.265	0.066	0.798	0.920	0.964
Godard – K + CS [41]		80	0.124	1.24	5.393	0.230	<u>0.052</u>	0.855	0.946	0.975
Cao [6]		80	0.115	-	4.712	0.198	-	<b>0.887</b>	<u>0.963</u>	0.982
kuznietsov [66]		80	0.113	<u>0.741</u>	4.621	0.189	-	0.862	0.960	0.986
Ours (DRN)		80	0.112	0.701	4.424	0.188	0.0492	0.848	0.958	0.986
Ours (DC-DRN)		80	0.110	0.698	4.529	0.187	0.0487	0.844	0.954	0.984
<b>Ours (DSC-DRN)</b>		80	<b>0.100</b>	<b>0.601</b>	<b>4.298</b>	<b>0.174</b>	<b>0.044</b>	<u>0.874</u>	<b>0.966</b>	<b>0.989</b>
Garg [37]		50	0.169	1.512	5.763	0.236	-	0.836	0.935	0.968
Godard – K [41]		50	0.149	1.235	4.823	0.259	0.065	0.800	0.923	0.966
Godard – K + CS [41]		50	0.117	0.866	4.063	0.221	<u>0.052</u>	0.855	0.946	0.975
Cao [6]		50	<u>0.107</u>	-	3.605	0.187	-	<b>0.898</b>	<u>0.966</u>	0.984
kuznietsov [66]		50	0.108	<u>0.595</u>	3.518	<u>0.179</u>	-	0.875	0.964	<u>0.988</u>
Ours (DRN)		50	0.109	0.618	3.702	0.182	0.0477	0.862	0.963	0.987
Ours (DC-DRN)		50	0.107	0.602	3.727	0.181	0.0470	0.865	0.963	0.988
<b>Ours (DSC-DRN)</b>		50	<b>0.096</b>	<b>0.482</b>	<b>3.338</b>	<b>0.166</b>	<b>0.042</b>	<u>0.886</u>	<b>0.980</b>	<b>0.995</b>



## 2. Monocular Depth Estimation by Learning from Heterogeneous Datasets

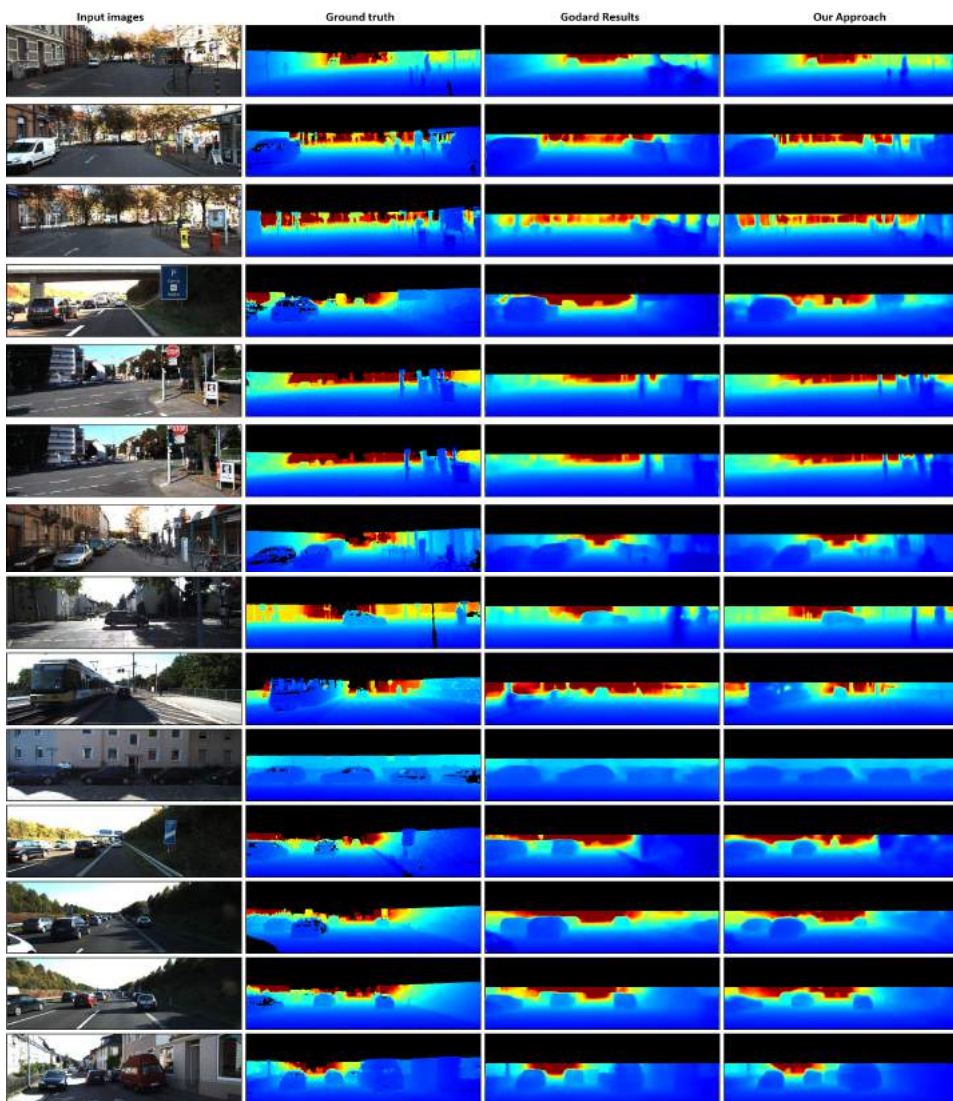


Figure 2.6 – Left to right: RGB image (KITTI), depth ground truth, Godard *et al.* [41] and our depth estimation results. In this figure, we show on the right side of the image that Godard *et al.* [41] results yield poor detection quality along with inaccurate depth estimation for specific relevant objects such as cars, tram or poles. On the other hand, our method provides a more accurate depth estimation which can be seen on the right most column.

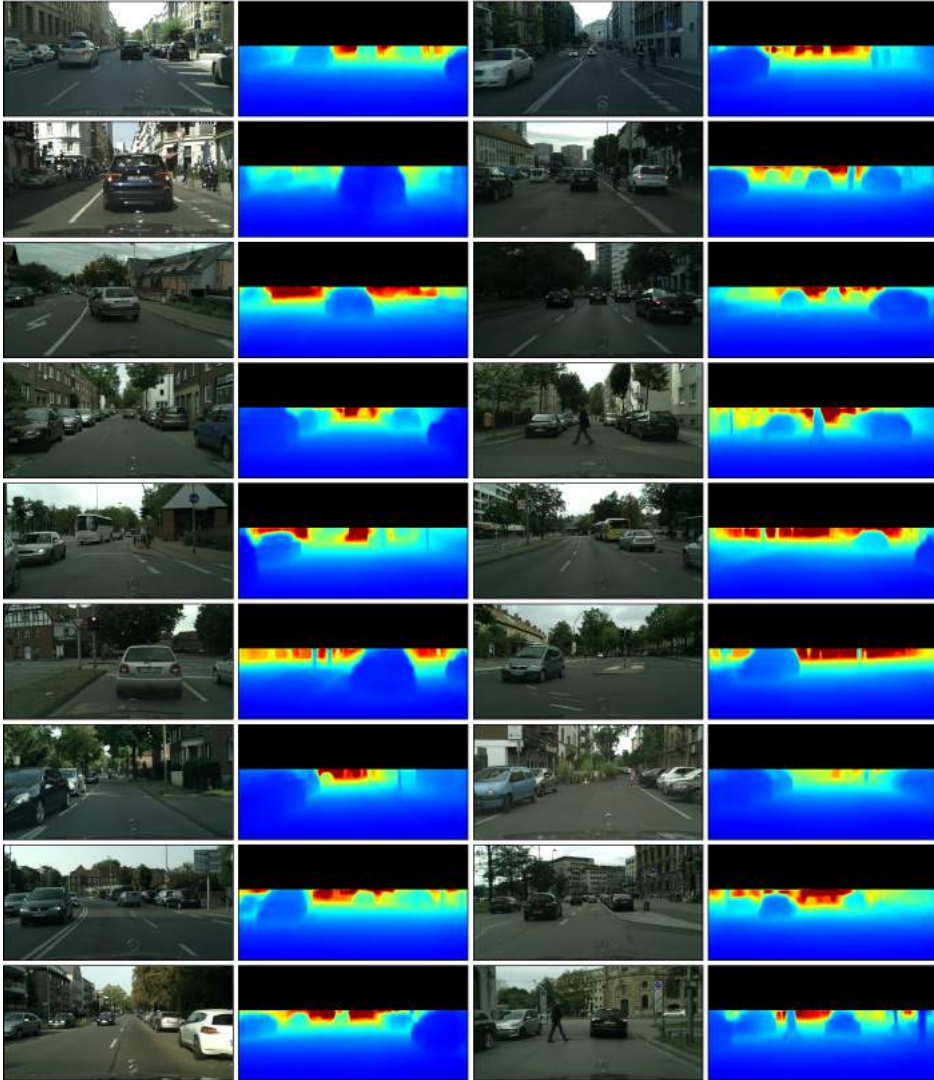


Figure 2.7 – Depth estimation results on Cityscapes validation and testing set images. This Cityscapes dataset is used for the task of semantic segmentation and we couldn't provide quantitative results as it doesn't have depth ground truth. **Note:** The validation set images are not used for training the network.

### 2.5 Conclusion

We have presented a method to leverage depth and semantic ground truth from different datasets for training a CNN-based depth-from-mono estimation model. Thus, up to the best of our knowledge, allowing for the first time to address outdoor driving scenarios with such a training paradigm (*i.e.* depth and semantics). In order to validate our approach, we have trained a CNN using depth ground truth coming from KITTI dataset as well as pixel-wise ground truth of semantic classes coming from Cityscapes dataset. Quantitative results on standard metrics show that the proposed approach improves performance, even yielding new state-of-the-art results.

# 3 Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---

Depth information is essential for on-board perception in autonomous driving and driver assistance. Monocular depth estimation (MDE) is very appealing since it allows for appearance and depth being on direct pixelwise correspondence without further calibration. Best MDE models are based on Convolutional Neural Networks (CNNs) trained in a supervised manner, *i.e.*, assuming pixelwise ground truth (GT). Usually, this GT is acquired at training time through a calibrated multi-modal suite of sensors. However, also using only a monocular system at training time is cheaper and more scalable. This is possible by relying on structure-from-motion (SfM) principles to generate self-supervision. Nevertheless, problems of camouflaged objects, visibility changes, static-camera intervals, textureless areas, and scale ambiguity, diminish the usefulness of such self-supervision. In this chapter, we perform *monocular depth estimation* by virtual-world supervision (MonoDEVS) and real-world SfM self-supervision. We compensate the SfM self-supervision limitations by leveraging virtual-world images with accurate semantic and depth supervision, and addressing the virtual-to-real domain gap. Our MonoDEVSNets outperforms previous MDE CNNs trained on monocular and even stereo sequences.

---

## 3.1 Introduction

Augmenting semantic information with depth is essential for on-board perception in autonomous driving and driver assistance. In this context, active sensors such as LiDAR and RADAR, or passive ones such as stereo rigs, are traditionally used to obtain depth information. For instance, in [28] RADAR and V2V communications are used to detect vehicles and estimate their distance to the ego-vehicle; LiDAR can be used for the same purpose [157], and it allows to perform road border detection too [25]; finally, note also that recent advances in deep stereo computation [16] can bring stereo rigs as a LiDAR alternative for some driving use cases. However, due to cost and maintenance considerations, we wish to predict depth from the same single camera used to predict

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---

semantics, so having a direct pixelwise correspondence without further calibration. Therefore, in this chapter, we focus on monocular depth estimation (MDE) on-board vehicles in outdoor traffic. Recent advances on MDE rely on Convolutional Neural Networks (CNNs). Let  $\Psi$  be a CNN architecture for MDE with weights  $\theta$ , which takes a single image  $x$  as input, and estimates its pixelwise depth map  $d$  as output, *i.e.*,  $\Psi(\theta; x) \rightarrow d$ . The  $\Psi$ 's can be trained in a supervised manner, *i.e.*, finding the values of  $\theta$  by assuming access to a set of images with pixelwise depth ground truth (GT). Usually, such a GT is acquired at training time through a multi-modal suite of sensors, at least consisting of a camera calibrated with a LiDAR or some type of 3D laser scanner variant [6, 30, 33, 49, 52, 69, 78, 110, 138, 145]. Alternatively, we can use self-supervision based on either a calibrated stereo rig [37, 41, 91, 112], or a monocular system and structure-from-motion (SfM) principles [44, 146, 152, 154], or a combination of both [42]. Combining stereo self-supervision and LiDAR supervision has been also analyzed [46, 53, 66]. The cheaper and simpler the suite of sensors used at training time, the better in terms of scalability and general access to the technology; however, the more challenging training a  $\Psi$ . Currently, supervised methods tend to outperform self-supervised ones [24], thus, improving the latter is an open challenge worth to pursue.

We are interested in the most challenging setting, namely, when at training time we only have a single on-board camera allowing for SfM-based self-supervision. However, using only such a self-supervision may give rise to depth estimation inaccuracies due to camouflage (objects moving as the camera may not be distinguished from background), visibility changes (occlusion changes, non-Lambertian surfaces), static-camera cases (*i.e.*, stopped ego-vehicle), and textureless areas, as well as to scale ambiguity (depth could only be estimated up to an unknown scale factor). In fact, an interesting approach to compensate for these problems could be leveraging virtual-world images (RGB) with accurate pixelwise depth (D) supervision. Using virtual worlds [5, 29, 34, 82, 107, 108, 115], we can acquire as many RGBD virtual-world samples as needed. However, these virtual-world samples can only be useful provided we address the virtual-to-real domain gap [15, 88, 92, 151, 153], which links MDE with visual domain adaptation (DA), a realm of research in itself [21, 129, 134].

Accordingly, our contributions to MDE are the following:

- We propose a CNN architecture to perform MDE by training on virtual-world supervision and real-world SfM self-supervision, *i.e.*, requiring just monocular sequences even at training time. We show that this architecture can accommodate different feature extraction backbones.
- We reduce domain discrepancies between supervised (virtual world) and semi-supervised (real world) data at the space of the extracted features (backbone bottleneck) by using the idea of gradient reversal layer (GRL) [35, 36]. Thus, not

adding computational burden at testing time.

- Despite using SfM-based semi-supervision, thanks to the virtual-world supervised data, we can compute a global scaling factor at training time, which allows us to output absolute depth at testing time.

In summary, we propose to perform *monocular depth estimation* by virtual-world supervision (MonoDEVS) and real-world SfM self-supervision, estimating depth in absolute scale. By relying on standard benchmarks, we show that our MonoDEVSNet outperforms previous ones trained on monocular and even stereo sequences. We think our released code and models<sup>1</sup> will help researchers and practitioners to address applications requiring on-board depth estimation, also establishing a strong baseline to be challenged in the future.

In the following, Section 3.2 summarizes previous works related to ours. Section 3.3 details our proposal. Section 3.4 describes the experimental setting and discusses the obtained results. Finally, Section 3.5 summarizes the presented work and conclusions, and draws the work we target for the near future.

## 3.2 Related Work

MDE was first based on hand-crafted features and shallow machine learning [67, 77, 112, 120]. Nowadays, best performing models are based on CNNs [24], thus, we focus on them.

### 3.2.1 Supervised MDE

Relying on depth GT, Eigen *et al.* [30] developed a  $\Psi$  architecture for coarse-to-fine depth estimation with a scale-invariant loss function. This pioneering work inspired new CNN-based architectures to MDE [6, 33, 52, 69, 78, 110, 138], which also assume depth GT supervision. MDE has been also tackle as a task on a multi-task learning framework, typically together with semantic segmentation as both tasks aim at producing pixelwise information and, eventually, may help each other to improve their predictions at object boundaries. For instance, this is the case of some  $\Psi$ 's for indoor scenarios [61, 62, 86]. These proposals assume that pixelwise depth and class GT are simultaneously available at training time. However, this is expensive, being scarcely available for outdoor scenarios. In order to address this problem, Gurram *et al.* [49] proposed a training framework which does not require depth and class GT to be available for the same images. Guizilini *et al.* [45] used an out-of-the-box CNN for semantic segmentation to train semantically-guided depth features while training  $\Psi$ .

<sup>1</sup><https://github.com/HMRC-AEL/MonoDEVSNet>

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---

The drawback of these supervised approaches is that the depth GT usually comes from expensive LiDARs, which must be calibrated and synchronized with the cameras; *i.e.*, even if the objective is to use only cameras for the functionality under development. Moreover, LiDAR depth is sparse compared to available image resolutions. Besides, surfaces like vehicle glasses or dark vehicles may be problematic for LiDAR sensing. Consequently, depth self-supervision and alternative sources of supervision are receiving increasing interest.

#### 3.2.2 Self-supervised MDE

Using a calibrated stereo rig to provide self-supervision for MDE is a much cheaper alternative to camera-LiDAR suites. Garg *et al.* [37] pioneered this approach by training  $\Psi$  with a warping loss involving pairs of stereo images. Godard *et al.* [41] introduced epipolar geometry constraints with additional terms for smoothing and enforcing consistency between left-right image pairs. Chen *et al.* [11] improved MDE results by enforcing semantic consistency between stereo pairs, via a joint training of  $\Psi$  and semantic segmentation. Pillai *et al.* [91] implemented sub-pixel convolutional layers for depth super-resolution, as well as a novel differentiable layer to improve depth prediction on image boundaries, a known limitation of stereo self-supervision. Other authors [53, 66] still complement stereo self-supervision with sparse LiDAR supervision.

SfM principles [160] can be also followed to provide self-supervision for MDE. In fact, in this setting we can assume a monocular on-board system at training time. Briefly, the underlying idea is that obtaining a frame,  $x_t$ , from consecutive ones,  $x_{t\pm 1}$ , can be decomposed into jointly estimating the scene depth for  $x_t$  and the camera pose at time  $t$  relative to its pose at time  $t \pm 1$ ; *i.e.*, the camera ego-motion. Thus, we can train a CNN to estimate (synthesize)  $x_t$  from  $x_{t\pm 1}$ , where, basically, the photo-metric error between  $x_t$  and  $\hat{x}_t$  acts as training loss, being  $\hat{x}_t$  the output of this CNN (*i.e.*, the synthesized view). After the training process, part of the CNN can perform MDE up to a scale factor (relative depth).

Zhou *et al.* [154] followed this idea, adding an explainability mask to compensate for violations of SfM assumptions (due to frame-to-frame changes on the visibility of frame’s content, textureless surfaces, *etc.*). This mask is estimated by a CNN jointly trained with  $\Psi$  to output a pixelwise belief on the synthesized views. Later, Yin *et al.* [146] proposed GeoNet, which aims at improving MDE by also predicting optical flow to explicitly consider the motion introduced by dynamic objects (*e.g.*, vehicles, pedestrians), *i.e.* a motion that violates SfM assumptions. However, this was effective on predicting occlusions, but not in significantly improving MDE accuracy. Godard *et al.* [42] followed the idea of having a mask to indicate stationary pixels, which should not be taken into account by the loss driving the training. Such pixels typically appear on vehicles moving at the same speed as the camera, or can even correspond to full

frames in case the ego-vehicle stops and, thus, the camera becomes stationary for a while. Pixels of similar appearance in consecutive frames are considered as stationary. A simple definition which can work because, instead of using a training loss based on absolute photo-metric errors (*i.e.* on minimizing pairwise pixel differences), it is used the structure similarity index measurement (SSIM) [132]. Moreover, within the so-called MonoDepth2 framework, Godard *et al.* [42] combine SfM and stereo self-supervision to establish state-of-the-art results. Alternatively, Guizilini *et al.* [45] addressed the presence of dynamic objects by a two-stage MDE training process. The first stage ignores the presence of such objects, returning a  $\Psi$  trained with a loss based on SSIM. Then, before running the second stage, the training sequences are processed to filter out frames that may contain erroneous depth estimations due to moving objects. Such frames are identified by applying  $\Psi$ , a RANSAC algorithm to estimate the ground plane from their estimated depth, and determining if there is a significant number of pixels that would be projected far below the ground plane. Finally, in the second stage,  $\Psi$  is retrained from scratch without the filtered frames.

Zhao *et al.* [152] focused on avoiding scale inconsistencies among frames as produced by SfM self-supervision, specially when they are from sequences whose underlying depth range is too different. Depth and optical flow estimation CNNs are trained, but not a pose estimation one. Instead, the optical flow between two frames is used to find robust pixel correspondences between them, which are used to compute their relative camera pose, computing the fundamental matrix by the 8-point algorithm, and then performing triangulation between the corresponding pixels of these frames. Overall, a sparse depth pseudo-GT is estimated and used as supervision to train  $\Psi$ . However, even robustifying scale consistency among frames, this method still outputs just relative depth. To avoid this problem, Guizilini *et al.* [46] used sparse LiDAR supervision with SfM self-supervision, relying on depth and pose estimation networks. More recently, Guizilini *et al.* [44] relied on ego-vehicle velocity to solve scale ambiguity in a pure SfM self-supervision setting. A velocity supervision loss trains the pose estimation CNN to learn scale-aware camera translation which, in turn, enables scale-aware depth estimation.

Overall, this literature shows the relevance of achieving MDE via SfM self-supervision and strategies to account for violation of SfM assumptions, as well as to obtain absolute depth values. Among these strategies, complementing SfM self-supervision with supervision (depth GT) coming from additional sensors such as a LiDAR and/or a stereo rig seems to be the most robust approach to address all the problems at once. However, then, a single camera would not be enough at training time. In this chapter, we also complement SfM self-supervision with accurate depth supervision. However, instead of relying on additional sensors, we use virtual-world data.



#### 3.2.3 Virtual-world data for MDE

Training  $\Psi$  on virtual-world images to later perform on real-world ones, requires to address the virtual-to-real domain gap. Many approaches perform a virtual-to-real image-to-image translation coupled to the training of  $\Psi$ . This translation usually relies on generative adversarial networks (GANs) [17, 43], since to train them only unpaired and unlabeled sets of real- and virtual-world images are required.

Zheng *et al.* [153] proposed  $T^2$ Net. In this case, a GAN and  $\Psi$  are jointly trained, where the GAN aims at performing virtual-to-real translation while acting as an auto-encoder for real-world images. The translated images are the input for  $\Psi$  since they have depth supervision. Additionally, a GAN operating on the encoder weights (features) of  $\Psi$  was incorporated during training to force similar depth feature distributions between translated and real-world images. However, this feature-level GAN worsen MDE results in outdoor scenarios. Kundu *et al.* [88] proposed AdaDepth, which trains a common feature space for real- and virtual-world images, *i.e.*, a space where it is not possible to distinguish the domain of the input images. Then, depth estimation is trained from this feature space. To achieve this, adversarial losses are used at the feature space level as well as at the estimated depth level.

Cheng *et al.* [15] proposed  $S^3$ Net, which extends  $T^2$ Net with SfM self-supervision. In this case, GAN training involves semantic and photo-metric consistency. Semantic consistency between the virtual-world images and their GAN-translated counterparts is required, which is measured via semantic segmentation (which involves also to jointly train a CNN for this task). Photo-metric consistency is required for consecutive GAN-translated images, which is measured via optical flow. Note that semantic segmentation and optical flow GT is available for virtual-world images.  $\Psi$  uses the GAN-translated images as input and is trained end-to-end with the GAN. Then, a further fine-tuning step of  $\Psi$  is performed using only the real-world sequence and SfM self-supervision, *i.e.*, involving the training of a pose estimation CNN while fine-tuning. During this process, a masking mechanism inspired in [42] is also used to compensate for SfM-adverse scenarios. Contrary to AdaDepth and  $T^2$ Net,  $S^3$ Net just outputs relative depth.

Zhao *et al.* [151] proposed GASDA, which leverages real-world stereo and virtual-world data. In this case, the CycleGAN idea [158] is used to perform DA, which actually involves two GANs, one for virtual-to-real image translation and another for real-to-virtual. Two  $\Psi$ 's are trained coupled to CycleGAN, one intended to process images with real-world appearance (actual real-world images or GAN-translated from the virtual domain), the other to process images with synthetic appearance (actual virtual-world images or GAN-translated from the real domain). In fact, at testing time, the most accurate depth results are obtained by averaging the output of these two  $\Psi$ 's, which also involves to translate the real-world images to the virtual domain by the corresponding GAN. Thanks to the stereo data, left-right depth and geometry consistency losses are

also included during training aiming at obtaining a more accurate  $\Psi$ . PNVR *et al.* [92] proposed SharinGAN for training a DA GAN coupled to a specific task. One of the selected tasks is MDE with stereo self-supervision, as in [151]. In this case, real- and virtual-world images are transformed to a new image domain where their appearance discrepancies are minimized to perform MDE from them, *i.e.* the GAN and the  $\Psi$  are jointly trained end-to-end. SharinGAN outperformed GASDA. However, at testing time, before performing the MDE, the real-world images must be translated by the GAN to the new image domain. Both GASDA and SharinGAN produce absolute scale depth.

### 3.2.4 Relationship of MonoDEVSNet with previous literature

In term of operational training conditions, the most similar paper to ours is  $S^3$ Net [15]. However, contrary to  $S^3$ Net, our MonoDEVSNet can estimate depth in absolute scale. On the other hand, for the SfM self-supervision we leverage from the state-of-the-art proposal in [42]. Note that methods based on pure SfM self-supervision such as [42] (only SfM setting), [154], [146], and [45], just report relative depth. In order to compare MonoDEVSNet with them, we have estimated relative depth too. We will see how we outperform these methods, proving the usefulness of leveraging depth supervision from virtual worlds. In fact, regarding relative depth, we also outperform  $S^3$ Net. Methods leveraging virtual-world data such as GASDA [151] and SharinGAN [92], rely on real-world stereo data at training time, while we only require monocular sequences. On the other hand, our training framework can be extended to accommodate stereo data if available, although it is not our current focus.  $S^3$ Net, GASDA, SharinGAN,  $T^2$ Net [153], and AdaDepth [88], leverage ideas from GAN-based DA to reduce the virtual-to-real domain gap, either in image space ( $S^3$ Net, GASDA, SharinGAN,  $T^2$ Net) or in feature space (AdaDepth). We have analyzed both, image and feature based DA, finding that the later outperforms the former. In particular, by using the Gradient-Reversal-Layer (GRL) DA strategy [35, 36], up to the best of our knowledge, not previously applied to MDE. Currently, we outperform the SfM self-supervision framework in [44] thanks to the virtual-world supervision and our GRL DA strategy. However, using vehicle velocity to obtain absolute depth as in [44], is a complementary strategy that could be also incorporated in our framework, although it is not the focus on this chapter.

## 3.3 Methods

In this section, we introduce MonoDEVSNet, which aims at leveraging virtual-world supervision to improve real-world SfM self-supervision. Since we train from both real- and virtual-world data jointly, we describe our supervision and self-supervision losses, the loss for addressing the virtual-to-real domain gap, and the strategy to obtain depth

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---

in absolute scale. Our proposal is visually summarized in Fig. 3.1.

#### 3.3.1 Training data

For training MonoDEVSNet, we assume two sources of data. On the one hand, we have image sequences acquired by a monocular system on-board a vehicle while driving in real-world traffic. We denote as  $x_t^r$  one of such frames acquired at time  $t$ . We denote these data as  $X^r = \{x_t^r\}_{t=1}^{N^r}$ , where  $N^r$  is the number of frames from the real-world sequences. These frames do not have associated GT. On the other hand, we have analogous sequences but acquired on a virtual world, *i.e.*, on-board a vehicle immersed in a traffic simulation. We denote as  $x_t^s$  one of such virtual-world frames acquired at time  $t$ . We refer to these data as  $X^s = \{x_t^s\}_{t=1}^{N^s}$ , where  $N^s$  is the number of frames from the virtual-world sequences. The images in  $X^s$  do have associated GT, since it can be automatically generated. In particular, as it is commonly available in today’s simulators, we assume pixelwise depth and semantic class GT. We define  $Y^s = \{< d_t^s, c_t^s >\}_{t=1}^{N^s}$  to be this GT; *i.e.*, given  $x_t^s$ ,  $d_t^s$  is its depth GT, and  $c_t^s$  its semantic class GT.

#### 3.3.2 MonoDEVSNet architecture: $\Psi(\theta; x)$

MonoDEVSNet, *i.e.*, our  $\Psi(\theta; x)$ , is composed of three main blocks: a encoding block of weights  $\theta^{\text{enc}}$ , a multi-scale pyramidal block,  $\theta^{\text{pyr}}$ , and a decoding block inspired in [42],  $\theta^{\text{dec}}$ . Therefore, the total set of weights is  $\theta = \{\theta^{\text{enc}}, \theta^{\text{pyr}}, \theta^{\text{dec}}\}$ . Here,  $\theta^{\text{enc}}$  acts as a backbone of features. Moreover, since we aim at evaluating several encoders, the role of the multi-scale pyramid block is to adapt the bottleneck of the chosen encoder to the decoder. At testing time  $\Psi(\theta; x)$  will process any real-world image  $x$  acquired on-board the ego-vehicle, while at training time either  $x \in X^r$  or  $x \in X^s$ .

#### 3.3.3 Problem formulation

Training  $\Psi(\theta; x)$  consists in finding the optimum weight values,  $\theta^*$ , by solving the problem:

$$\theta^* = \min_{\theta} \mathcal{L}(\theta; X^r, X^s, Y^s) ,$$

where  $\mathcal{L}$  is a loss function, and  $X^s, Y^s$  indicates the use of the virtual-world frames with their GT. As we are going to detail,  $\mathcal{L}$  relies on three different losses, namely,  $\mathcal{L}^{\text{sf}}$ ,  $\mathcal{L}^{\text{sp}}$  and  $\mathcal{L}^{\text{da}}$ . The loss  $\mathcal{L}^{\text{sf}}$  focuses on training  $\theta$  based on SfM self-supervision, thus, only relying on real-world data sequences. The SfM self-supervision is achieved with the support of a camera pose estimation task performed by a CNN, T, of weights  $\theta^{\text{sf}}$ . Thus, we have  $\mathcal{L}^{\text{sf}}(\theta, \theta^{\text{sf}}; X^r)$ . The loss  $\mathcal{L}^{\text{sp}}$  focuses on training  $\theta$  with virtual-world supervision,

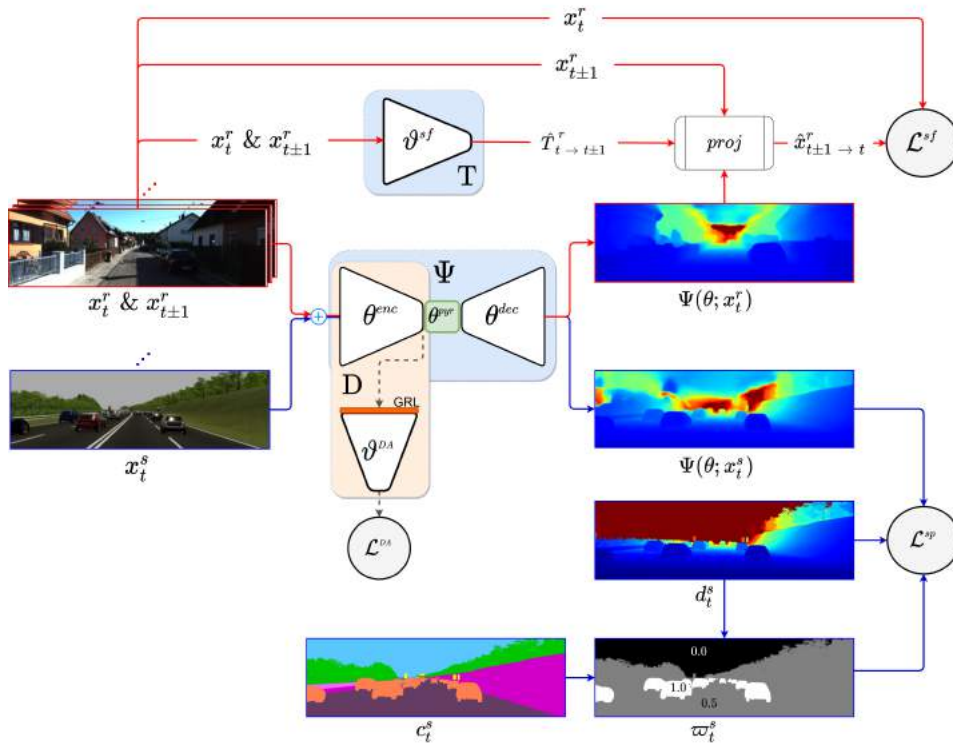


Figure 3.1 – Training framework for MonoDEVSNet, *i.e.*,  $\Psi(\theta; x)$ . We show the involved images, GT, weights, and losses. Red and blue lines are paths of real and virtual-world data, respectively. The discontinuous line is a common path.

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---

in particular, using depth and semantic GT from virtual-world sequences. Therefore, we have  $\mathcal{L}^{\text{sp}}(\theta; X^s, Y^s)$ . Finally,  $\mathcal{L}^{\text{DA}}$  focuses on creating domain-invariant features  $\theta^{\text{enc}}$  as part of  $\theta$ . In particular, we rely on a binary real/virtual domain-classifier CNN, D, of weights  $\{\theta^{\text{enc}}, \vartheta^{\text{DA}}\}$ . Thus, we have  $\mathcal{L}^{\text{DA}}(\theta^{\text{enc}}, \vartheta^{\text{DA}}; X^r, X^s)$ .

#### 3.3.4 SfM Self-supervised loss: $\mathcal{L}^{\text{sf}}(\theta, \vartheta^{\text{sf}}, X^r)$

Since we focus on improving MDE by the additional use of virtual-world data, for the SfM self-supervision we leverage from the state-of-the-art proposal in [42], which we briefly summarize here for the sake of completeness as:

$$\mathcal{L}^{\text{sf}}(\theta, \vartheta^{\text{sf}}, X^r) = \sum_{t=2}^{N^r-1} P_t^r(\theta, \vartheta^{\text{sf}}) + \lambda S_t^r(\theta) . \quad (3.1)$$

As introduced in [41], the term  $\lambda S_t^r(\theta)$  is a constant weighted loss to force local smoothness on  $\Psi(\theta; x_t^r)$ , taking into account the edges of  $x_t^r$ . The term  $P_t^r(\theta, \vartheta^{\text{sf}})$  is the actual SfM-inspired loss. It involves the joint training of the depth estimation weights,  $\theta$ , and the relative camera pose estimation weights,  $\vartheta^{\text{sf}}$ . Figure 3.1 illustrates the CNN, T, associated to these weights, which takes as input two consecutive frames, *e.g.*,  $(x_t^r, x_{t+1}^r)$ , and outputs the pose transform (rotation and translation),  $\hat{T}_{t \rightarrow t+1}^r = T(\vartheta^{\text{sf}}; x_t^r, x_{t+1}^r)$ , between them. Then, as can be seen in Fig. 3.1, a projection module takes  $\hat{T}_{t \rightarrow t+1}^r$ ,  $x_{t+1}^r$ , and the depth estimation  $\Psi(\theta; x_t^r)$ , to generate the synthesized frame  $\hat{x}_{t+1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}})$  which, ideally, should match  $x_t^r$ . In fact, both frames adjacent to  $x_t^r$  are considered for robustness. Thus, the SfM-inspired component of  $\mathcal{L}^{\text{sf}}$  is defined as:

$$P_t^r(\theta, \vartheta^{\text{sf}}) = \overline{cpe}(x_t^r, \hat{x}_{t \pm 1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}}), x_{t \pm 1}^r) ,$$

where  $cpe()$  is a pixelwise conditioned photo-metric error and  $\overline{cpe}()$  its average over the pixels. Obtaining  $cpe()$  starts by computing two pixelwise photo-metric error measurements,  $pe(x_{t-1}^r, x_t^r, x_{t+1}^r)$  and  $pe(\hat{x}_{t-1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}}), x_t^r, \hat{x}_{t+1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}}))$ , where  $pe(x_{-1}^r, x_0^r, x_{+1}^r) = \min(pe(x_0^r, x_{-1}^r), pe(x_0^r, x_{+1}^r))$ , and  $pe(x_A^r, x_B^r)$  is the pixelwise photo-metric error between  $x_A^r$  and  $x_B^r$  defined in [41], *i.e.*, based on local structural similarity (SSIM) and pixelwise photo-metric absolute differences between  $x_A^r$  and  $x_B^r$ . Thus,  $\min()$  applies pixelwise. Then, a pixelwise binary *auto-mask* [42] is computed as:

$$\omega_t^r(x_t^r, \hat{x}_{t \pm 1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}}), x_{t \pm 1}^r) = [pe(\hat{x}_{t-1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}}), x_t^r, \hat{x}_{t+1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}})) < pe(x_{t-1}^r, x_t^r, x_{t+1}^r)]_1,$$

where  $[ ]_I$  denotes the Iverson bracket applied pixelwise. Finally,  $cpe()$  is computed as:

$$cpe(x_t^r, \hat{x}_{t\pm 1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}}), x_{t\pm 1}^r) = \omega_t^r(x_t^r, \hat{x}_{t\pm 1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}}), x_{t\pm 1}^r) \odot pe(\hat{x}_{t-1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}}), x_t^r, \hat{x}_{t+1 \rightarrow t}^r(\theta, \vartheta^{\text{sf}})) ,$$

where  $\odot$  stands for pixelwise multiplication. The auto-mask  $\omega_t^r()$  conditions which pixels of  $pe()$  are considered during the gradient computation of  $\mathcal{L}^{\text{sf}}$ . As explained in [42], the aim of  $\omega_t^r()$  is to remove, during training, the influence of pixels which remain the same between adjacent frames because they are assumed to often indicate SfM violations such as a static camera, objects moving as the camera, or low texture regions. The support of  $\vartheta^{\text{sf}}$  is not needed at testing time.

### 3.3.5 Supervised loss: $\mathcal{L}^{\text{sp}}(\theta; X^s, Y^s)$

In this case, since we address an estimation problem and we have accurate GT, we base  $\mathcal{L}^{\text{sp}}$  on the L1 metric. On the other hand, MDE is specially interesting to determine how far is the ego-vehicle from vehicles, pedestrians, etc. Accordingly, since  $Y^s$  includes semantic class GT, we use it to increase the relevance of accurately estimating the depth for such major traffic protagonists. Moreover, since virtual-world depth maps are based on the Z-buffer involved on image rendering, the range of depth values available as GT tend to be over-optimistic even for active sensors such as LiDAR. For instance, there can be depth values larger than 300  $m$  in the Z-buffer. Since we do not aim at estimating depth beyond a reasonable threshold (in  $m$ ),  $d^{\text{max}}$ , to compute  $\mathcal{L}^{\text{sp}}$  we will also discard pixels  $p$  with  $d_t^s(p) \geq d^{\text{max}}$ . For each  $x_t^s$ , both the semantic class relevance and the out-of-range depth values, can be codified as real-valued weights running on  $[0, 1]$  and arranged on a mask,  $\omega_t^s$ . Thus,  $\omega_t^s$  depends on  $d_t^s$ ,  $d^{\text{max}}$ , and  $c_t^s$ . However, contrarily to  $\omega_t^r()$ , we can compute  $\omega_t^s$  offline, *i.e.*, before starting the training process. Taking all these details into account, we define our supervised loss as:

$$\mathcal{L}^{\text{sp}}(\theta; X^s, Y^s) = \sum_{t=1}^{N^s} \|\omega_t^s \odot (\Psi(\theta; x_t^s) - d_t^s)\|_1 . \quad (3.2)$$

### 3.3.6 Domain adaptation loss: $\mathcal{L}^{\text{da}}(\theta^{\text{enc}}, \vartheta^{\text{da}}; X^r, X^s)$

As can be seen in Fig. 3.1, we aim at learning depth features,  $\theta^{\text{enc}}$ , so that it cannot be distinguished whether they were generated from a real-world input frame (target domain) or a virtual-world one (source domain); in other words, learning a domain invariant  $\theta^{\text{enc}}$ . Taking into account that we do not have accurate depth GT in the target

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---

domain, while we do have it for the source domain, we need to apply an unsupervised DA technique to train  $\theta^{\text{enc}}$ . In addition, as part of  $\theta$ , the training of  $\theta^{\text{enc}}$  must result on an accurate  $\Psi(\theta; x)$ . Achieving this accuracy and domain invariance are adversarial goals. Accordingly, we propose to use the Gradient-Reversal-Layer (GRL) idea introduced in [35], which, up to the best of our knowledge, has not been applied before for DA in the context of MDE. In this approach, the domain invariance of  $\theta^{\text{enc}}$  is measured by a binary target/source domain-classifier CNN,  $D$ , of weights  $\{\theta^{\text{enc}}, \vartheta^{\text{DA}}\}$ . In [35], a logistic loss is proposed to train the domain classifier. In our case, this is set as:

$$\mathcal{L}^{\text{DA}}(\theta^{\text{enc}}, \vartheta^{\text{DA}}; X^r, X^s) = \sum_{t=1}^{N^r} \log(D(\theta^{\text{enc}}, \vartheta^{\text{DA}}; x_t^r)) + \sum_{t=1}^{N^s} \log(1 - D(\theta^{\text{enc}}, \vartheta^{\text{DA}}; x_t^s)) , \quad (3.3)$$

where we assume that  $D(\theta^{\text{enc}}, \vartheta^{\text{DA}}; x)$  outputs 1 if  $x \in X^r$  and 0 if  $x \in X^s$ . The GRL has no parameters and connects  $\theta^{\text{enc}}$  with  $\vartheta^{\text{DA}}$  (see Fig. 3.1). Its behavior is exactly as explained in [35]. This means that during forward passes of training, it acts as an identity function, while, during back-propagation, it reverses the gradient vector passing through it. Both the GRL and  $\vartheta^{\text{DA}}$  are required at training time, but not at testing time.

#### 3.3.7 Overall training procedure

Algorithm 1 summarizes the steps to compute the needed gradient vectors for mini-batch optimization. In particular, we need the gradients related to MonoDEVSNet weights,  $\theta = \{\theta^{\text{enc}}, \theta^{\text{pyr}}, \theta^{\text{dec}}\}$ , and the weights of the auxiliary tasks, *i.e.*,  $\vartheta^{\text{sf}}$  for SfM self-supervision, and  $\vartheta^{\text{DA}}$  for DA. Regarding gradient computation, we do not need to distinguish  $\theta^{\text{pyr}}$  from  $\theta^{\text{dec}}$ , so we define  $\theta^{\text{pyde}} = \{\theta^{\text{pyr}}, \theta^{\text{dec}}\}$ . In Alg. 1, we introduce an equalizing factor between supervised and self-supervised losses,  $\omega^{\text{sf}} \in \mathbb{R}$ , which aims at avoiding one loss dominating over the other. A priori, we could set a constant factor. However, in practice, we have found that having an adaptive value is more useful. Therefore, inspired by the GradNorm idea [14], we use the ratio between the supervised and self-supervised losses. Algorithm 1 also introduces the scaling factor  $\omega^{\text{DA}} \in \mathbb{R}$  which, following [35], controls the trade-off between optimizing  $\theta^{\text{enc}}$  to obtain an accurate  $\Psi(\theta; x)$  model versus being domain invariant. Finally,  $\mathcal{L}^{\text{DA}}(\theta^{\text{enc}}, \vartheta^{\text{DA}}; \emptyset, X_B^s)$  and  $\mathcal{L}^{\text{DA}}(\theta^{\text{enc}}, \vartheta^{\text{DA}}; X_B^r, \emptyset)$  indicate whether this loss must be computed only using virtual- or real-world data, respectively.

**Algorithm 1:** Computing the gradients  $\Delta_{\theta^{\text{enc}}}$ ,  $\Delta_{\theta^{\text{pyde}}}$ ,  $\Delta_{\theta^{\text{sf}}}$ ,  $\Delta_{\theta^{\text{DA}}}$  for a mini-batch  $X_B^r \subset X^r$ ,  $X_B^s, Y_B^s \subset X^s, Y^s$ .  $\nabla_{\xi_i} F(\xi_1, \xi_2)$  refers to back-propagation on  $F(\xi_1, \xi_2)$  with respect to weights  $\xi_i \subset \xi_1 \cup \xi_2$ .  $\emptyset$  is the empty set.

Forward Passes with  $\{X_B^s, Y_B^s\}$

$$\begin{aligned} \ell^{\text{SP}}(\theta) &\leftarrow \mathcal{L}^{\text{SP}}(\theta; X_B^s, Y_B^s) \\ \ell^{\text{DA},s}(\theta^{\text{enc}}, \vartheta^{\text{DA}}) &\leftarrow \omega^{\text{DA}} \mathcal{L}^{\text{DA}}(\theta^{\text{enc}}, \vartheta^{\text{DA}}, \emptyset, X_B^s) \end{aligned}$$

Back-propagation for Supervision & DA

$$\begin{aligned} \Delta_{\theta^{\text{pyde}}}^s &\leftarrow \nabla_{\theta^{\text{pyde}}} \ell^{\text{SP}}(\theta) \\ \Delta_{\theta^{\text{enc}}}^s &\leftarrow \nabla_{\theta^{\text{enc}}} (\ell^{\text{SP}}(\theta) - \ell^{\text{DA},s}(\theta^{\text{enc}}, \vartheta^{\text{DA}})) \\ \Delta_{\vartheta^{\text{DA}}}^s &\leftarrow \nabla_{\vartheta^{\text{DA}}} \ell^{\text{DA},s}(\theta^{\text{enc}}, \vartheta^{\text{DA}}) \end{aligned}$$

Forward Passes with  $X_B^r$

$$\begin{aligned} \ell^{\text{sf}}(\theta, \vartheta^{\text{sf}}) &\leftarrow \mathcal{L}^{\text{sf}}(\theta, \vartheta^{\text{sf}}; X_B^r) \\ \ell^{\text{DA},r}(\theta^{\text{enc}}, \vartheta^{\text{DA}}) &\leftarrow \omega^{\text{DA}} \mathcal{L}^{\text{DA}}(\theta^{\text{enc}}, \vartheta^{\text{DA}}, X_B^r, \emptyset) \end{aligned}$$

Back-propagation for Self-supervision & DA

$$\begin{aligned} \Delta_{\theta^{\text{pyde}}}^r &\leftarrow \nabla_{\theta^{\text{pyde}}} \ell^{\text{sf}}(\theta, \vartheta^{\text{sf}}) \\ \Delta_{\theta^{\text{enc}}}^r &\leftarrow \nabla_{\theta^{\text{enc}}} (\ell^{\text{sf}}(\theta, \vartheta^{\text{sf}}) - \ell^{\text{DA},r}(\theta^{\text{enc}}, \vartheta^{\text{DA}})) \\ \Delta_{\vartheta^{\text{DA}}}^r &\leftarrow \nabla_{\vartheta^{\text{DA}}} \ell^{\text{DA},r}(\theta^{\text{enc}}, \vartheta^{\text{DA}}) \end{aligned}$$

Setting the final gradient vectors

$$\begin{aligned} \Delta_{\theta^{\text{sf}}} &\leftarrow \nabla_{\theta^{\text{sf}}} \ell^{\text{sf}}(\theta, \vartheta^{\text{sf}}) \\ \Delta_{\theta^{\text{DA}}} &\leftarrow \Delta_{\theta^{\text{DA}}}^s + \Delta_{\vartheta^{\text{DA}}}^r \\ \omega^{\text{sf}} &\leftarrow \ell^{\text{SP}}(\theta) / \ell^{\text{sf}}(\theta, \vartheta^{\text{sf}}) \\ \Delta_{\theta^{\text{pyde}}} &\leftarrow \Delta_{\theta^{\text{pyde}}}^s + \omega^{\text{sf}} \Delta_{\theta^{\text{pyde}}}^r \\ \Delta_{\theta^{\text{enc}}} &\leftarrow \Delta_{\theta^{\text{enc}}}^s + \omega^{\text{sf}} \Delta_{\theta^{\text{enc}}}^r \end{aligned}$$



#### 3.3.8 Absolute depth computation

The virtual-world supervised data trains  $\Psi(\theta; x)$  on absolute depth values, while the real-world SfM self-supervised data trains  $\Psi(\theta; x)$  on relative depth values. Thanks to the unsupervised DA, the depth features  $\theta^{\text{enc}}$  are trained to be domain invariant. However, according to our experiments, this is not sufficient for  $\Psi(\theta; x)$  producing accurate absolute depth values at testing time. Fortunately, thanks to the use of virtual-world data, we can still compute a global scaling factor,  $\psi \in \mathbb{R}$ , so that  $\psi\Psi(\theta; x)$  is accurate in absolute depth terms. For that, we assume that the sequences in  $X^s$  are acquired with a camera analogous to the one used to acquire the sequences in  $X^r$ . Here analogous refers to using the same number of pixels, field of view, frame rate, and mounted on-board in similar heading directions. Note that simulators are flexible enough for setting these camera parameters as needed. Accordingly, we train a  $\Psi(\theta; x)$  model using only data from  $X^s$  and SfM self-supervision, *i.e.* as if we would not have supervision for  $X^s$ . Then, we find the median depth value produced by this model on the virtual-world data,  $\hat{d}^{s,m} \in \mathbb{R}$ . Finally, we set  $\psi = d^{s,m} / \hat{d}^{s,m}$ , where  $d^{s,m} \in \mathbb{R}$  is the median depth value of the GT. This pre-processing step is performed once and the model discarded afterwards. Other works apply a similar approach [15, 42, 45, 152, 154] but relying on LiDAR data as GT reference, while we only rely on virtual-world data.

## 3.4 Experimental Results

We start by defining the datasets and evaluation metrics used in our experiments. After, we provide relevant implementation and training details of MonoDEVSNets. Finally, we present and discuss our quantitative and qualitative results, comparing them with those from previous literature as well as performing an ablative analysis over MonoDEVSNets components.

### 3.4.1 Datasets and evaluation metrics

We use publicly available datasets and metrics which are *de facto* standards in MDE research. In particular, we use KITTI Raw (KR) [38] and Virtual KITTI (VK) [5] as real- and virtual-world sequences, respectively. We follow Zhou *et al.* [154] training-testing split. From the training split we select 12K monocular triplets, *i.e.*, samples of the form  $\{x_{t-1}^r, x_t^r, x_{t+1}^r\}$ . The testing split consists of 697 isolated images with LiDAR-based GT, actually introduced by Eigen *et al.* [30]. In addition, for considering the semantic content of the images in the analysis of results, we also use KITTI Stereo 2015 (KS) [83] for testing. This dataset consists of 200 isolated images with enhanced depth maps and semantic labels. VK is used only for training, we also use 12K monocular triplets (non-rotated camera subset) with associated depth GT. In this case, the triplets are used to calibrate

the global scaling factor  $\psi$  (see Sect. 3.3.8), while for actual training supervision only 12K isolated frames are used. As the depth GT of VK ranges up to  $\sim 655\text{m}$ , to match the range of KR’s LiDAR-based GT, we clip it to  $80\text{m}$  ( $d^{\max}$ ). VK includes similar weather conditions as KR/KS, and adds situations with fog, overcast, and rain, as well as sunrise and sunset illumination.

As is common practice since [41], we use Make3D dataset [111] for assessing generalization. It contains photographs of urban and natural areas. Thus, Make3D shows views and content pretty much different from those on-board a vehicle as KR, KS, and VK. The images come with depth GT acquired by a 3D scanner. There are 534 images with depth GT, organized in a standard split of 400 for training and 134 for testing. We use the latter, since we rely on Make3D only for testing.

In order to assess quantitative MDE results, we use the standard metrics introduced by Eigen *et al.* [30] and described in Section 2.4.3, *i.e.*, the average absolute relative error (abs-rel), the average squared relative error (sq-rel), the root mean square error (rms), and the rms log error (rms-log). For these metrics, the lower the better. In addition, the accuracy (termed as  $\delta$ ) under a threshold  $\tau \in \{1.25, 1.25^2, 1.25^3\}$  is also used as metric. In this case, the higher the better. We remind that the abs-rel error and the  $\delta < \tau$  are percentage measurements, sq-rel and rms are reported in meters, and rms-log is similar (reported in meters) to rms but applied to logarithm depth values.

These metrics are applied to absolute depth values for MDE models trained with depth supervision coming from either LiDAR [6, 30, 33, 46, 49, 52, 69, 78, 110, 138, 145], stereo [37, 41, 42, 91, 112], real-world stereo and virtual-world depth [92, 151], or stereo and LiDAR [53, 66]. However, MDE models trained on pure SfM self-supervision can only estimate depth in relative terms, *i.e.*, up to scale. Moreover, the scale factor varies from image to image, a problem known as scale inconsistency. In this case, before computing the above metrics, it is applied a per-image correction factor computed at testing time [15, 42, 45, 146, 152, 154]. In particular, given a test image  $x$  with GT and estimated depth  $d(x)$  and  $\hat{d}(x)$ , respectively, the common practice consists of computing a scale  $\psi(x) \in \mathbb{R}$  as the ratio  $\text{median}(d(x))/\text{median}(\hat{d}(x))$ , and then compare  $\psi(x)\hat{d}(x)$  with  $d(x)$ . On the other hand, SfM self-supervision with the help of additional information can train models able to produce absolute scale in testing time. For instance, [44] uses the ego-vehicle speed and, in fact, virtual-world supervision can help too [88, 153]. The latter approach is the one followed in this chapter, especially thanks to the procedure presented in Sect. 3.3.8. Therefore,  $\Psi(\theta; x)$  will be evaluated in relative scale terms, and  $\psi\Psi(\theta; x)$  in absolute terms. Please, note that our  $\psi \in \mathbb{R}$  scaling factor is constant for all the evaluated images and computed at training time. In the following, when presenting quantitative results, we will make clear if they are in relative or absolute terms.

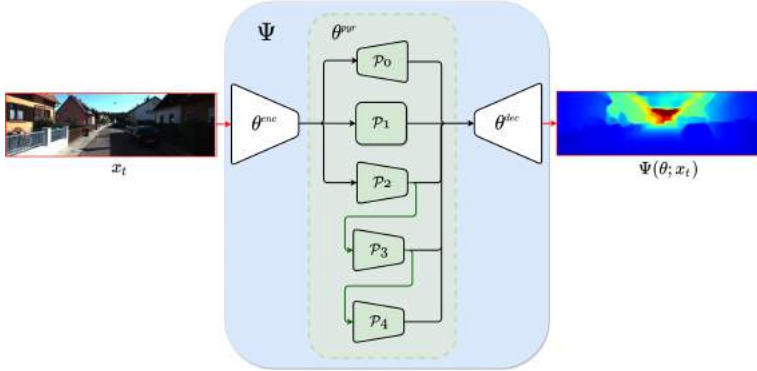
### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

Table 3.1 – Comparing ResNet and HRNet as backbone for  $\Psi(\theta; x)$ , training only on SfM self-supervision (relative scale) using the framework in [42]. MW column stands for millions of  $\theta^{\text{enc}}$  weights to be learnt. FPS stands for *frames per second* as required by  $\Psi(\theta; x)$  to process  $x$ , while GFLOPS refers to the *giga floating-point operations per second* required by  $\theta^{\text{enc}}$ ; in both cases using an NVIDIA RTX 2080Ti GPU. The  $1.25^n$  columns,  $n \in \{1, 2\}$ , refer to the  $\tau$  in the usual  $\delta < \tau$  accuracy metrics. In all the tables of Sect. 3.4, bold stands for **best** and underline for second-best. (\*) Currently, HRNet branches do not run in parallel in PyTorch, thus, compromising speed.

$\theta^{\text{enc}}$ Backb.	MW	GFLOPS	FPS	abs-rel	sq-rel	rms	1.25	1.25 <sup>2</sup>
ResNet-18	<u>11.6</u>	<b>4.47</b>	<b>141.2</b>	0.115	0.882	4.701	0.879	0.961
ResNet-50	25.5	10.14	<u>77.06</u>	0.110	0.831	4.642	0.883	0.962
ResNet-101	44.5	19.29	43.26	0.110	0.809	4.712	0.878	0.960
ResNet-152	60.2	28.47	30.71	0.107	<u>0.800</u>	<u>4.629</u>	0.885	0.962
HRNet-W18	<b>9.5</b>	<u>8.29</u>	15.79*	<u>0.107</u>	0.846	4.671	<u>0.887</u>	<u>0.962</u>
HRNet-W32	29.3	19.50	15.53*	0.107	0.881	4.794	0.886	0.961
HRNet-W48	65.3	40.04	15.48*	<b>0.105</b>	<b>0.791</b>	<b>4.590</b>	<b>0.888</b>	<b>0.963</b>

#### 3.4.2 Implementation details

We start by selecting the actual CNN layers to implement  $\Psi(\theta; x)$ . Since we leverage the SfM self-supervision idea from [42], a straightforward implementation would be to use its ResNet-based architecture as it is. However, the High-Resolution Network (HRNet) architecture [128], exhibits better accuracy in visual tasks such as semantic segmentation and object detection, suggesting that it can be a better backbone than ResNet. Thus, we decided to start our experiments by comparing ResNet and HRNet backbones using the SfM self-supervision framework provided in [42]. In particular, we assess different ResNet/HRNet architectures for  $\theta^{\text{enc}}$ , while using the proposal in [42] for  $\theta^{\text{dec}}$ . Then, when using ResNet we have  $\theta^{\text{pyt}} = \emptyset$ , while for HRNet  $\theta^{\text{pyt}}$  consists of pyramidal layers adapting the  $\theta^{\text{enc}}$  and  $\theta^{\text{dec}}$  CNN architectures under test. For these experiments, we rely on KR. Table 3.1 shows the accuracy (in relative scale terms) of the tested variants and their number of weights. We see how HRNet outperforms ResNet, being HRNet-W48 the best. Indeed, HRNet is slower than ResNet, and HRNet-W48 is the one requiring more GFLOPS by far. However, at this stage of our research we target the architecture which potentially can provide higher depth estimation accuracy. Thus, for our following experiments, we will rely on HRNet-W48 although being the heaviest. We show the corresponding pyramidal architecture of  $\theta^{\text{pyt}}$  in Fig. 3.2. It is composed of five blocks ( $\mathcal{P}_i$ ), where each block is a pipeline of three consecutive layers consisting of convolution, batch normalization, and ReLU. As a deep learning framework, we use PyTorch 1.5v [89].

Figure 3.2 – Pyramidal architecture of  $\theta^{\text{pyr}}$ .

In order to train the camera pose estimation network,  $T(\vartheta^{\text{sf}}; x_t^r, x_{t\pm 1}^r)$ , we follow [42] but using ResNet-50 instead of ResNet-18 since the former is more accurate. Four convolutional layers are used to convert the ResNet-50 bottleneck features to the 6-DoF relative pose vector (3D translation and rotation). For training the classification block of  $D(\theta^{\text{enc}}, \vartheta^{\text{DA}}; x)$ , *i.e.*,  $\vartheta^{\text{DA}}$ , we use a standard classification pipeline based on convolutions, ReLU and fully connected layers. Finally, we remark that these networks are not required at testing time.

### 3.4.3 Training details

The input images are processed (at training and testing time) at a resolution of  $640 \times 192$  ( $W \times H$ ), where LANCZOS interpolation is performed from the  $\sim 1242 \times 375$  original resolution. As optimizer, we use Adam [64] with learning rate  $lr = 10^{-4}$ , and the rest of its hyper-parameters set to default values. The weights  $\theta^{\text{enc}}$  are initialized from available ImageNet [26] pre-training,  $\theta^{\text{pyr}}$ ,  $\theta^{\text{dec}}$ , and  $\vartheta^{\text{DA}}$  are randomly initialized with Kaiming weights, while the ResNet-50 part of  $\vartheta^{\text{sf}}$  is also initialized with ImageNet and the rest (convolutional layers to output the pose vector) following Kaiming. The mini-batch size is of 16 images, 50%/50% from real/virtual domains. To minimize over-fitting, we apply standard data augmentation such as horizontal flip, a 50% chance of random brightness, contrast, saturation, and hue jitter with ranges of  $\pm 0.2$ ,  $\pm 0.2$ ,  $\pm 0.2$ , and  $\pm 0.1$ , respectively. Remaining hyper-parameters were set as  $\lambda = 0.001$  in Eq. 3.1,  $\omega^{\text{DA}} = 10$  in Alg. 1, and in Eq. 3.2 our mask  $\omega_i^z$  is set to have values of 1.0 for traffic participants (vehicles, pedestrians, *etc.*), 0.5 for static infrastructure (buildings, road, vegetation, *etc.*), and 0.0 for the sky and pixels with depth over  $d^{\text{max}}$  (here 80m).

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

Table 3.2 – *Relative depth* results up to 80m on the (KR) Eigen *et al.* [30] testing split. These methods rely on SfM self-supervision. In addition, methods in gray use DA supported by VK. (<sup>1</sup>) MonoDepth2 is based only on SfM self-supervision.

Method	abs-rel	sq-rel	rms	rms-log	1.25	1.25 <sup>2</sup>	1.25 <sup>3</sup>
[154] (Zhou <i>et al.</i> )	0.183	1.595	6.709	0.270	0.734	0.902	0.959
[146] GeoNet	0.149	1.060	5.567	0.226	0.796	0.935	0.975
[42] MonoDepth2 <sup>1</sup>	0.115	0.903	4.863	0.193	0.877	0.959	0.981
[152] (Zhao <i>et al.</i> )	0.113	0.704	4.581	0.184	0.871	0.961	<b>0.984</b>
[45] (Guizilini <i>et al.</i> )	<b>0.102</b>	<u>0.698</u>	<u>4.381</u>	<b>0.178</b>	<b>0.896</b>	0.964	<b>0.984</b>
[15] S <sup>3</sup> Net (VK_v1)	0.124	0.826	4.981	0.200	0.846	0.955	0.982
MonoDEVSNNet / VK_v1	<u>0.105</u>	0.753	4.389	<u>0.179</u>	0.890	<u>0.965</u>	<u>0.983</u>
MonoDEVSNNet / VK_v2	<b>0.102</b>	<b>0.685</b>	<b>4.303</b>	<b>0.178</b>	<u>0.894</u>	<b>0.966</b>	<b>0.984</b>

#### 3.4.4 Results and discussion

##### Relative depth assessment

We start by assessing MDE in relative terms. Table 3.2 presents MonoDEVSNNet results (Ours) and those from previous works based on SfM self-supervision. From this table we can draw several observations. Regarding DA, MonoDEVSNNet (VK\_v1) outperforms S<sup>3</sup>Net (VK\_v1) in all metrics. The new version of VK (VK\_v2) allows us to obtain even better results. MonoDEVSNNet with virtual-world supervision outperforms the version with only SfM self-supervision (best result in Table 3.1) in all metrics, no matter the VK version we use. Overall, MonoDEVSNNet outperforms most previous methods, being on par with [45].

##### Absolute depth assessment

While assessing depth in relative terms is a reasonable option to compare methods purely based on SfM self-supervision, the most relevant evaluation is in terms of absolute depth. These are presented in Table 3.3. The first (top) block of this table shows results based on depth supervision from LiDAR, thus, a priori they can be thought of as upper-bounds for methods based on self-supervision. The second block shows methods that only use virtual-world supervision. The third and fourth (bottom) blocks show results based on stereo and SfM self-supervision, respectively. Methods in gray use DA supported by VK. We can draw several observations from this table. MonoDEVSNNet (Ours) is the best performing among those leveraging supervision from VK\_v1 and, consistently with the results on relative depth, by using VK\_v2 we improve MonoDEVSNNet results. In fact, MonoDEVSNNet based on VK\_v2 outperforms all self-supervised methods, including

Table 3.3 – *Absolute depth* results up to 80m on the (KR) Eigen *et al.* [30] testing split. We divide the results into four blocks. From top to bottom, the blocks refer to: methods based on LiDAR supervision, only virtual-world supervision, stereo self-supervision, SfM self-supervision. In these blocks, we remark best and second-best results per block. Methods in gray use DA supported by VK. We remark some additional comments: <sup>(1)</sup> in addition to LiDAR supervision, it also uses stereo self-supervision; <sup>(2)</sup> it uses stereo and SfM self-supervision; <sup>(3)</sup> in this case, the MDE network is pre-trained on Cityscapes dataset [18] and then fine-tuned on KITTI.

Method	abs-rel	sq-rel	rms	rms-log	1.25	1.25 <sup>2</sup>	1.25 <sup>3</sup>
[30] (Eigen <i>et al.</i> )	0.203	1.548	6.307	0.282	0.702	0.890	0.890
[78] (Liu <i>et al.</i> )	0.217	1.841	6.986	0.289	0.647	0.882	0.961
[6] (Cao <i>et al.</i> )	0.115	N/A	4.712	0.198	<u>0.887</u>	0.963	0.982
[66] (Kuzni. <i>et al.</i> ) <sup>1</sup>	0.113	0.741	4.621	0.189	0.862	0.960	<u>0.986</u>
[138] (Xu <i>et al.</i> )	0.122	0.897	4.677	N/A	0.818	0.954	0.985
[49] (Gurram <i>et al.</i> )	<u>0.100</u>	<u>0.601</u>	<u>4.298</u>	<u>0.174</u>	0.874	<u>0.966</u>	<b>0.989</b>
[33] DORN	<b>0.098</b>	<b>0.582</b>	<b>3.666</b>	<b>0.160</b>	<b>0.899</b>	<b>0.967</b>	<u>0.986</u>
[88] AdaDepth / VK_v1	<b>0.167</b>	<b>1.257</b>	<b>5.578</b>	<b>0.237</b>	<b>0.771</b>	<b>0.922</b>	<b>0.971</b>
[153] T <sup>2</sup> Net / VK_v1	<u>0.174</u>	<u>1.410</u>	<u>6.046</u>	<u>0.253</u>	<u>0.754</u>	<u>0.916</u>	<u>0.966</u>
[37] (Garg <i>et al.</i> )	0.169	1.512	5.763	0.236	0.836	0.935	0.968
[91] SuperDepth	0.112	0.875	4.958	<u>0.207</u>	0.852	0.947	0.977
[42] MonoDepth2	<u>0.109</u>	<u>0.873</u>	4.960	0.209	<u>0.864</u>	<u>0.948</u>	0.975
[42] MonoDepth2 <sup>2</sup>	<b>0.106</b>	<b>0.806</b>	<b>4.630</b>	<b>0.193</b>	<b>0.876</b>	<b>0.958</b>	<b>0.980</b>
[151] GASDA / VK_v1	0.120	1.022	5.162	0.215	0.848	0.944	0.974
[92] SharinGAN / VK_v1	0.116	0.939	5.068	0.203	0.850	<u>0.948</u>	<u>0.978</u>
[44] PackNet-SfM	0.111	0.829	4.788	0.199	0.864	0.954	0.980
[44] PackNet-SfM <sup>3</sup>	<u>0.108</u>	0.803	4.642	0.195	<u>0.875</u>	0.958	0.980
MonoDEVSNNet / VK_v1	<u>0.108</u>	<u>0.775</u>	<u>4.464</u>	<u>0.188</u>	<u>0.875</u>	<u>0.961</u>	<u>0.982</u>
MonoDEVSNNet / VK_v2	<b>0.104</b>	<b>0.721</b>	<b>4.396</b>	<b>0.185</b>	<b>0.880</b>	<b>0.962</b>	<b>0.983</b>

those using stereo rigs instead of monocular systems. We are not yet able to reach the performance of the best methods supervised with LiDAR data. However, it is clear that our proposal is able to successfully combine real-world SfM self-supervision and virtual-world supervision. Thus, we think it is worth to keep this line of research until reaching the LiDAR-based upper-bounds.

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

Table 3.4 – *Absolute depth* ablative results of MonoDEVSNets (VK\_v2) up to 80m on the (KR) Eigen testing split [30]. Rows 1-6 show the progressive use of the components of our proposal (each row adds a new component). 50/50 refers to mini-batches of 50% real-world samples and 50% or virtual-world ones; not using 50/50 (rows 1-2) means that we alternate mini-batches of pure real- or virtual-world samples. Row 7 corresponds to a simplification of the SfM self-supervised loss.  $\vartheta^G$  (rows 8-9) refers to a GAN-based DA approach. LB (lower bound, row 10) indicates the use of only virtual-world data. UB (upper bound, row 12) indicates the use of KITTI LiDAR-based supervision instead of virtual-world data. Rows 11 and 13 show the difference of our best model (All) with respect to LB and UB, respectively.  $\uparrow D$  means that All is  $D$  units better, while  $\downarrow D$  means that it is  $D$  units worse. All/W18 (row 14) and All/W32 (row 15) refer to using the All configuration by relying on HRNet-W18 and HRNet-W32, respectively.

Configuration	abs-rel	sq-rel	rms	rms-log	1.25	1.25 <sup>2</sup>	1.25 <sup>3</sup>
1. $\{X^r, X^s, Y^s\}$	0.368	2.601	8.025	0.514	0.080	0.478	0.883
2. $+\psi$	0.140	0.876	4.915	0.217	0.828	0.950	0.980
3. +50/50	0.128	0.880	4.618	0.198	0.844	0.957	0.982
4. $+\vartheta^{DA}$	0.110	0.724	4.450	0.187	0.873	0.960	0.983
5. $+\omega^{sf}$	0.106	0.716	4.441	0.188	0.876	0.962	0.982
6. $+\omega_t^s$ (All)	<b>0.104</b>	<b>0.721</b>	<b>4.396</b>	<b>0.185</b>	<b>0.880</b>	<b>0.962</b>	<b>0.983</b>
7. Simplified $\mathcal{L}^{sf}$	0.105	0.736	4.471	0.190	0.875	0.960	0.981
8. All+ $\vartheta^G$ ; $-\vartheta^{DA}$	0.119	0.809	4.654	0.196	0.857	0.958	0.982
9. All+ $\vartheta^G$	0.106	0.748	4.503	0.191	0.873	0.959	0.981
10. LB	0.165	1.280	5.628	0.248	0.777	0.916	0.965
11. $\uparrow$ All vs. $\downarrow$ LB	$\uparrow 0.061$	$\uparrow 0.559$	$\uparrow 1.232$	$\uparrow 0.063$	$\uparrow 0.103$	$\uparrow 0.046$	$\uparrow 0.018$
12. UB	0.088	0.583	3.978	0.164	0.906	0.970	0.986
13. $\uparrow$ All vs. $\downarrow$ UB	$\downarrow 0.016$	$\downarrow 0.138$	$\downarrow 0.418$	$\downarrow 0.021$	$\downarrow 0.026$	$\downarrow 0.008$	$\downarrow 0.003$
14. All/W18	0.109	0.773	4.524	0.190	0.871	0.960	0.982
15. All/W32	0.107	0.754	4.510	0.188	0.875	0.960	0.982

#### Ablative analysis of MonoDEVSNets

It is also worth to analyze the contribution of the main components of our proposal. In rows 1-6 of Table 3.4, we add one component at a time showing performance for absolute depth. The 1st row corresponds to using the real-world data with SfM self-supervision and the virtual-world images with only depth supervision, *i.e.*, without using neither semantic supervision ( $\omega_t^s$ ), nor gradient equalization ( $\omega^{sf}$ ), nor domain adaptation ( $\vartheta^{DA}$ ), nor mixed mini-batches (50/50), nor the global scaling factor ( $\psi$ ). By

comparing 1st and 2nd rows (*i.e.*, w/o  $\psi$  and w/  $\psi$ , resp.), we can see how relevant is obtaining a good global scaling factor to output absolute depth. In fact, adding  $\psi$  to the virtual-world depth supervision shows the higher improvement among all the components of our proposal. Then, using mixed mini-batches of real- and virtual-world data improves the performance over alternating mini-batches of only either real- or virtual-world data. This can be seen by comparing 2nd and 3rd rows (*i.e.*, w/o 50/50 and w/ 50/50, resp.). If we alternate the domains, the optimization of a mini-batch is dominated by self-supervision (real-world data), and the optimization of the next mini-batch is dominated by supervision (virtual-world data). Thus, there is not an actual joint optimization of SfM self-supervised and supervised losses, which turns to be relevant. Yet, as can be seen in 4th row, when we add the DA component ( $\vartheta^{\text{DA}}$ ) we improve further the depth estimation results. As can be seen in 5th row, adding the equalization ( $\omega^{\text{sf}}$ ) between gradients coming from supervision and self-supervision also improves the depth estimation results. Finally, adding the virtual-world mask ( $\omega_t^{\text{v}}$ ) leads to the best performance in 6th row. Overall, this analysis shows how all the considered components are relevant in our proposal. We also remark that these components are needed only to train  $\theta$ , but only  $\psi$  and  $\theta$  are required at testing time. Additionally, we have assessed the effect of simplifying the SfM self-supervised loss that we leverage from [42], here summarized in Sect. 3.3.4. In particular, we neither use the auto-mask ( $\omega_t^{\text{r}}$ ), nor the multi-scale depth loss, and we replaced the minimum re-projection loss by the usual average re-projection loss (*i.e.*, we re-define  $pe(x_{-1}^r, x_0^r, x_{+1}^r)$  in Sect. 3.3.4). Results are shown in the 7th row. The metrics show worse values than in 6th row (All), but still outperforming or being on pair with PackNet-SfM and the stereo self-supervised methods of Table 3.3.

We also did additional experiments changing the DA mechanism. Instead of taking direct real- and virtual-world images as input to train  $\Psi(\theta; x)$ , a GAN-based CNN,  $\mathcal{G}$ , processes them to create an image space in which (hopefully) it is not possible to distinguish the domain. We train a CNN,  $\Psi(\theta; \mathcal{G}(\vartheta^{\mathcal{G}}; x))$ , where  $x$  can come from either the real or the virtual domain, and  $\vartheta^{\mathcal{G}}$  are the weights of  $\mathcal{G}$ . These weights are jointly trained with  $\theta$ ,  $\vartheta^{\text{sf}}$ , and  $\vartheta^{\text{DA}}$  to optimize depth estimation and minimize the possibility of discriminating the original domain of a sample  $x^{\mathcal{G}} = \mathcal{G}(\vartheta^{\mathcal{G}}; x)$ . Table 3.4 shows results using this GAN when removing  $\vartheta^{\text{DA}}$  (8th row) and when keeping it (9th row). As we can see, this approach does not improve performance. Moreover, the training is more complex and  $\mathcal{G}(\vartheta^{\mathcal{G}}; x)$  would be required at testing time. Thus, we discarded it.

We also assessed the improvement of our proposal with respect a lower-bound model (LB) trained on virtual-world images and their depth GT ( $X^s \cdot Y^s$ ), but neither using real-world data ( $X^r$ ), nor DA ( $\vartheta^{\text{sf}}$ ), nor the mask ( $\omega_t^{\text{v}}$ ). Results are shown in 10th row of Table 3.4, and we explicitly show the improvement of our proposal over such LB in 11th row. Likewise, we have trained an upper-bound model (UB) replacing VK data by KR data with LiDAR-based supervision, so that DA is not required. Results are shown in 12th row,



### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---

and the distance of our model to this UB is explicitly shown in 13th row. Comparing 11th and 13th rows we can see how we are clearly closer to the UB than to the LB.

Finally, we have done experiments using HRNet-W18 and HRNet-W32. The results are shown in 14th and 15th rows of Table 3.4, respectively. Indeed, as it happens with the results on relative depth (Table 3.1), HRNet-W48 outperforms these more lightweight versions of HRNet. However, by using HRNet-W18 and HRNet-W32 we still outperform or are on par with the state-of-the-art self-supervised methods shown in Table 3.3, *i.e.*, those based on stereo self-supervision and PackNet-SfM.

#### Qualitative results

Figure 3.3 presents qualitatively results relying on the depth color map commonly used in the MDE literature. We show results for representative methods in Table 3.3, namely, DORN (LiDAR supervision), SharinGAN (stereo self-supervision and virtual-world supervision), PackNet-SfM (SfM self-supervision and ego-vehicle speed supervision), and MonoDEVSNet (Ours) using VK\_v1 and VK\_v2 (SfM self-supervision and virtual-world supervision). We also show the corresponding LiDAR-based GT. This GT shows that for LiDAR configurations such as the one used to acquire KITTI dataset, detecting some close vehicles may be problematic since only a few LiDAR points capture their presence. Despite being trained on LiDAR supervision, DORN provides more accurate depth information in these corner cases than the raw LiDAR, which is an example of the relevance of MDE in general. However, DORN shows worse results in these corner cases than the rest (SharinGAN/PackNet-SfM/Ours), even being more accurate in terms of MDE metrics, which focus on global assessment. SharinGAN has more difficulties than PackNet-SfM and our proposal for providing sharp borders in vertical objects/infra-structure (*e.g.*, vehicles, pedestrians, traffic signs, trees). An interesting point to highlight is also the qualitative difference that we observe on our results depending on the use of VK version. In VK\_v1 data, vehicle windows appear as transparent to depth, like in many cases happens with LiDAR data, while in VK\_v2 they appear as solid. This is translated to the MDE results as we can observe comparing the two bottom rows of Fig. 3.3. Technically, we think the qualitative results of VK\_v2 make more sense since the windows are there at the given depth. However, what we would like to highlight is that we can select one option or another thanks to the use of virtual-world data.

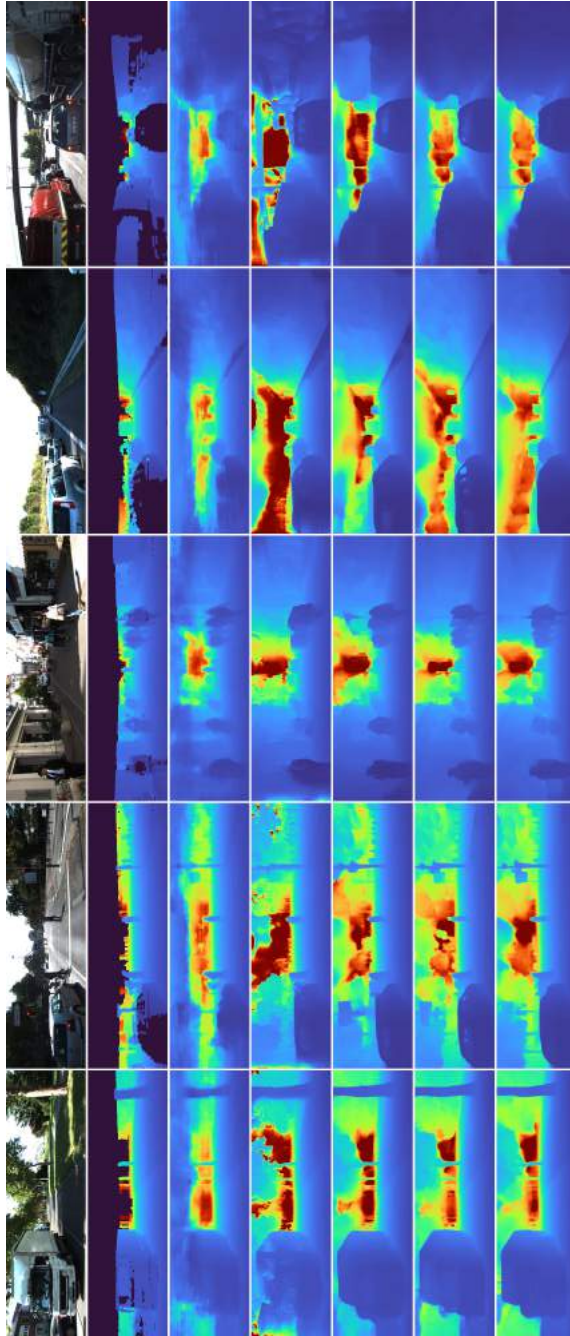


Figure 3.3 – Qualitative results on the (KR) Eigen *et al.* testing slit [30]. From top to bottom: input images, their LIDAR-based depth GT (interpolated for visualization using [95]), DORN, SharinGAN, PackNet-SfM, MonoDEVSNet VK\_v1 and VK\_v2.

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

#### Additional insights

In terms of qualitative results we think the best performing and most similar approaches are PackNet-SfM and MonoDEVSNets, both relying on real-world monocular systems. Thus, we perform a deeper comparison of them. First, following PackNet-SfM article [44], Fig. 3.4 plots the abs-rel error as a function of depth. Since this is a relative error, we also plot rms. Results are similar within a close range of up to 20m. Within 20m and 70m, our proposal clearly outperforms PackNet-SfM and beyond 70m both methods perform similarly. How these differences translate to abs-rel and rms global scores depends on the number of pixels falling in each distance range, which we show as an histogram in the same plot. We see how for the KR testing set most of the pixels fall in the 5 – 20m depth range, where both methods perform more similarly. Second, we provide further comparative insights by using KS data since it has associated per-class semantic GT. Note that, although KS is a different data split than the one used in the experiments shown so far (KR), still is KITTI data; thus, we are not yet facing experiments about generalization. Figure Fig. 3.5 compares qualitative results of PackNet-SfM *vs.* MonoDEVSNets. We can see how PackNet-SfM misses some vehicles that our proposal does not. We believe that these vehicles may be moving at a similar speed w.r.t the ego-vehicle, which may be problematic for pure SfM-based approaches and we hypothesize that virtual-world supervision can help to avoid this problem. Figure Fig. 3.6 shows the corresponding abs-rel metric per-class, focusing on the most relevant classes for driving. Note how the main differences between PackNet-SfM and MonoDEVSNets are observed on vehicles, especially on cars.

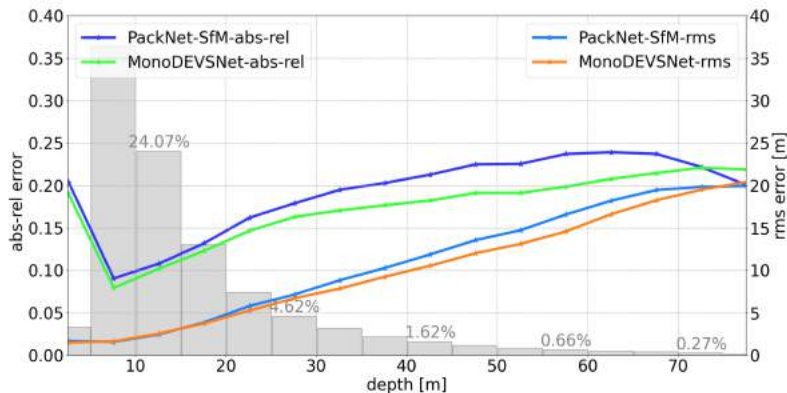


Figure 3.4 – Abs-rel and rms errors as a function of depth, for KR Eigen testing split [30]. The histogram of depth GT is shown with bars. We compare PackNet-SfM and MonoDEVSNets.

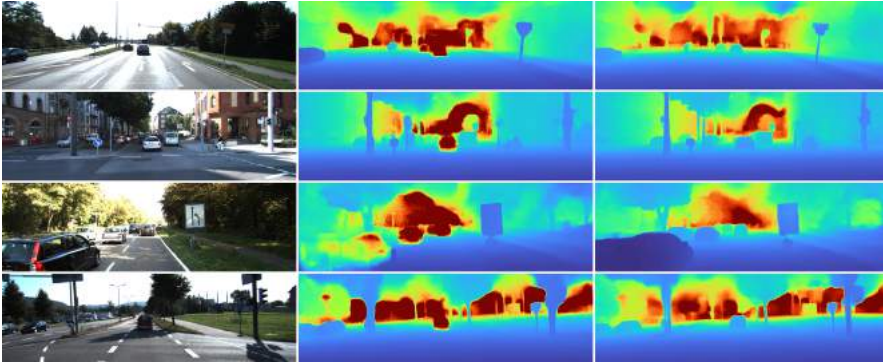


Figure 3.5 – Qualitative results on KS data. From left to right: input images, PackNet-SfM, MonoDEVSNet.

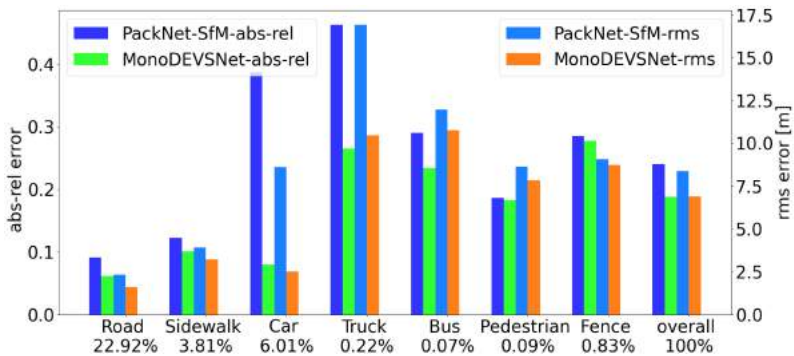


Figure 3.6 – Per-class abs-rel and rms errors for KS, computed by averaging over the pixels of each class, for PackNet-SfM and MonoDEVSNet. The % of pixels of each class is shown.

Additional qualitative results are added in Fig. 3.7, where we can see how original images from KR and KS can be rendered as a textured point cloud. In particular, the viewpoint of these renders can change with respect to the original images thanks to the absolute depth values obtained with MonoDEVSNet.

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---



Figure 3.7 – Point cloud representation on KR Eigen test split [30] and KS data from left to right. From top to bottom: input images, MonoDEVSNet textured point cloud.

#### Generalization results

As done in the previous literature using VK to support MDE [88, 92, 151, 153], we assess generalization on Make3D dataset. As in this literature, we follow the standard data conditioning (cropping and resizing) for models trained on KR, as well as the standard protocol introduced in [41] to compute MDE evaluation metrics (*e.g.* only depth below 70m is considered). Table 3.5 presents the quantitative results usually reported for Make3D, and ours. Note how, in generalization terms, our method also outperforms the rest. Moreover, Fig. 3.8 shows how our proposal captures the depth structure even better than the depth GT, which is build from  $55 \times 305$  depth maps acquired by a 3D scanner. In addition, we show qualitative results on two datasets: ApolloStereo [75] and Cityscapes [18] datasets which are recording in a real-world driving setup similar to car-mounted videos of KITTI. A visual survey showing the results of MonoDEVSNet on ApolloStereo and Cityscapes dataset is summerized in Fig. 3.9 and Fig. 3.10 respectively. However, its difficult to compute MDE evaluation metrics on these datasets, as the depth maps from Cityscapes are precomputed by using SGM and as the ApolloStereo doesnt provide baseline for the conversion of disparity to depth maps.



Table 3.5 – *Absolute depth* results on Make3D testing set. All the shown methods use Make3D only for testing (generalization), except (<sup>1</sup>) which fine-tunes on Make3D training set too.

Method	abs-rel	sq-rel	rms
[153] $T^2$ Net / VK_v1	0.508	6.589	8.935
[88] AdaDepth-S <sup>1</sup> / VK_v1	0.452	5.71	9.559
[151] GASDA / VK_v1	0.403	6.709	10.424
[92] SharinGAN / VK_v1	0.377	4.900	8.388
MonoDEVSNets / VK_v1	0.381	3.997	<b>7.949</b>
MonoDEVSNets / VK_v2	<b>0.377</b>	<b>3.782</b>	<u>8.011</u>

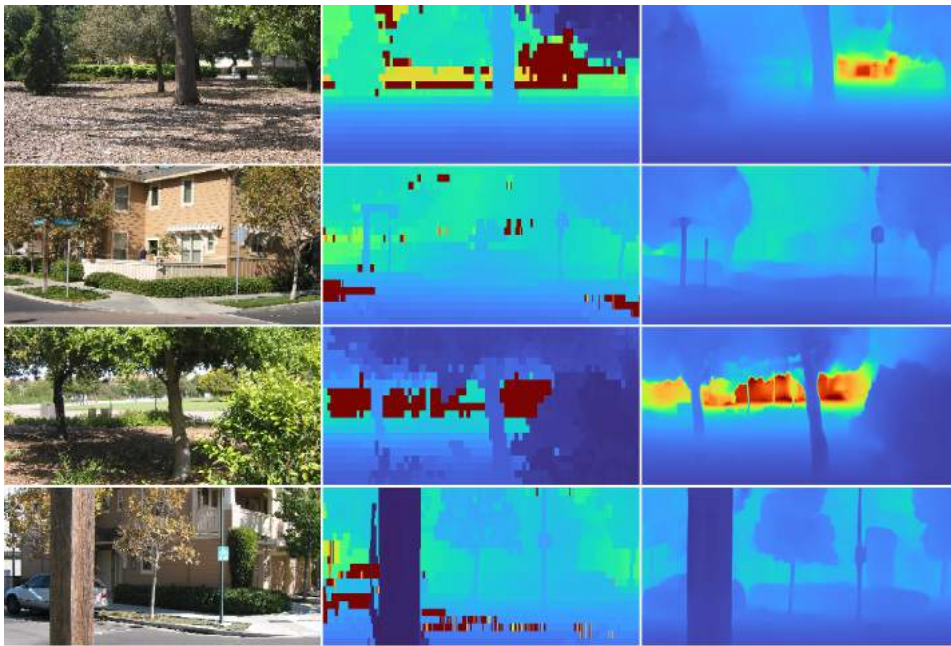


Figure 3.8 – Qualitative results of MonoDEVSNets on Make3D. From left to right: input images, depth GT, MonoDEVSNets.

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---

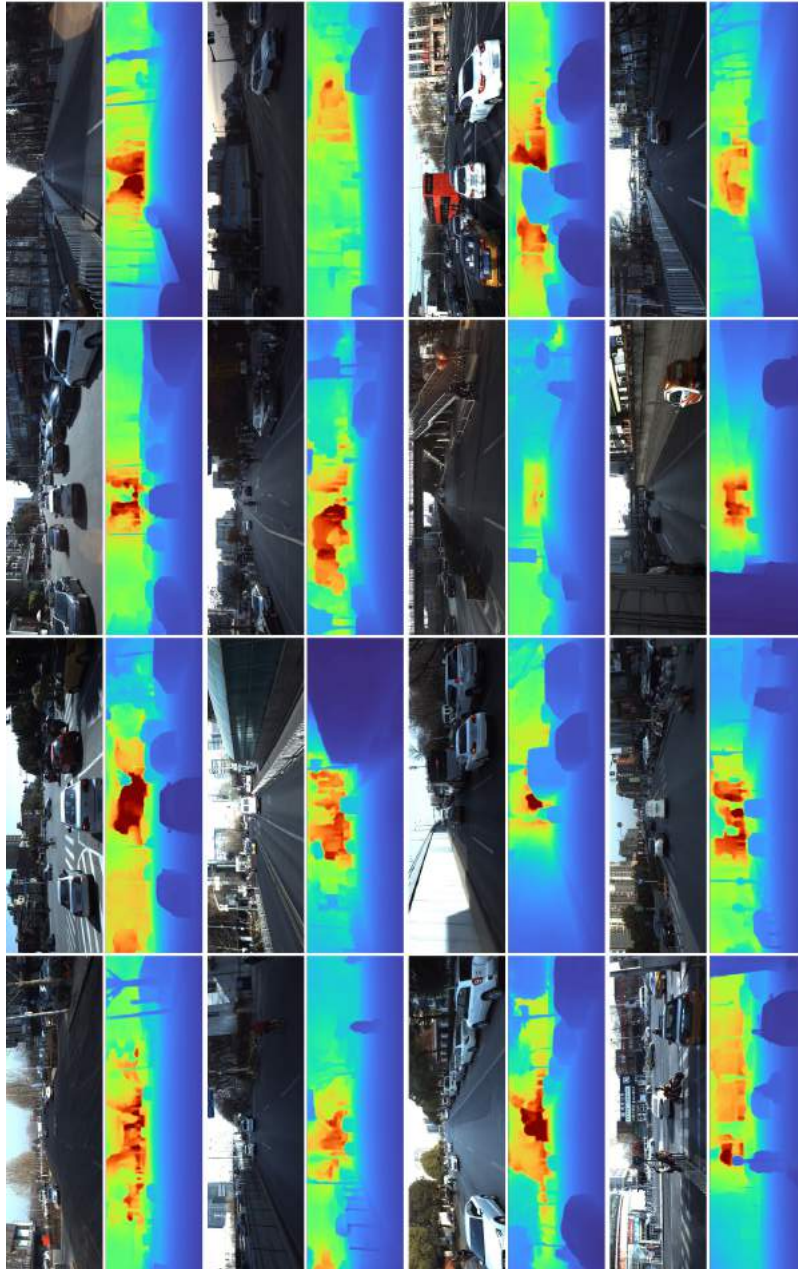


Figure 3.9 – Qualitative results on ApolloStereo dataset.



Figure 3.10 – Qualitative results of MonoDEVSNet on Cityscapes dataset. From left to right: input images, MonoDEVSNet estimated depth maps.

### Failure cases

Fig. 3.11 shows qualitative results where some errors in the depth map are highlighted: (1) overexposed pixel columns lead to hallucinate a vertical structure; (2) saturated fence segments are not seen; (3) pedestrians are visible but with an approximate silhouette; (4) a bridge is not seen; (5) saturated skies do not appear as faraway. Thinking in the information required to drive and assuming that depth estimation is combined with semantic segmentation, we think that the cases (3) and (5) are not a problem and the (2) would be only if the unseen segment is too large (which is not the case in the shown example). The case (4) could be a problem for an autonomous bus/truck provided it does not fit below the bridge, but usually those would have predefined routes where this should not happen. However, (1) can be a problem depending where the hallucinated structure appears, *e.g.*, in the example probably it would not be a problem, but it would be if the structure appears in the middle of the road. Behind some errors, we can find the lack of training data (*e.g.*, for the bridge not seen). Behind others, we find extreme imaging conditions (like overexposed image areas). In the former case, we need to re-train by taking these cases into account; while, in the latter case, we need to prevent such undesired effects (*e.g.* by using HDR camera settings). It is worth to mention that



### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---

we have seen similar errors in other methods in the literature.

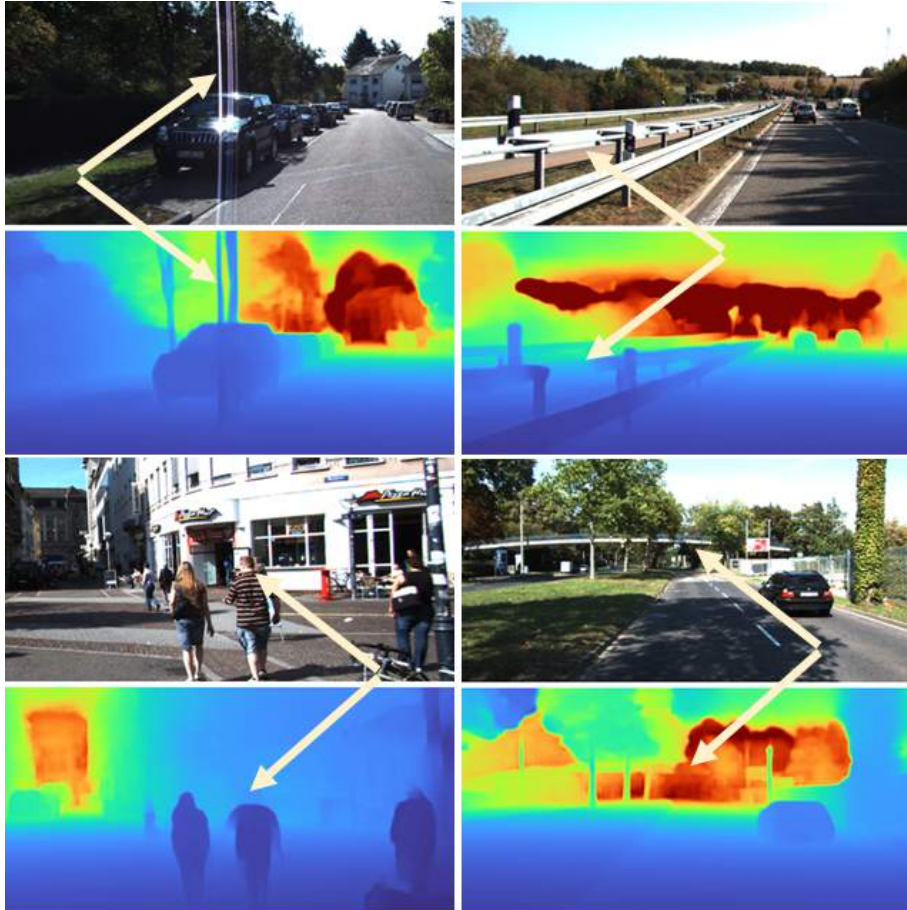


Figure 3.11 – Failure cases in the depth map.

### Revisiting $\theta^{\text{enc}}$ architectures

We selected HRNet-W48 because it provides the most accurate results, however, we may need to sacrifice accuracy to reduce the computational burden. Thus, we have run more experiments with the final MonoDEVS training approach, just changing  $\theta^{\text{enc}}$ . These include representative architectures of ResNet and HRNet types, as well as, DenseNet [59]. As ResNet, DenseNet does not need adding the pyramidal blocks ( $\theta^{\text{pyr}}$ ). Moreover, as we mentioned in Sect. 3.4.1, to keep our experimental work manageable, we used 12K triplets (samples) from the real- and virtual-world training sets; however, it is possible to use  $\sim 40\text{K}$  samples from KR, which is the common practice in the literature (as those using KR in Tables 3.1-3.3). Likewise, we use  $\sim 22\text{K}$  samples from VK\_v2 as other literature methods in Table 3.5 do with VK\_v1. Table 3.6 presents the corresponding results. The block  $\sim 40\text{K}/22\text{K}$  can be directly compared to the literature in Table 3.3). We see how, indeed, HRNet-W48 is the best in term of accuracy metrics. However, we see that DenseNet-121 offers the best trade-off between memory (MW) and computational (GFLOPS) requirements, offering real-time (FPS) with accuracy close to the state-of-the-art. If we need to reduce the computational burden and significantly increase the FPS, then ResNet-18 is a reasonable alternative.

Table 3.6 – Absolute depth. We provide final experimental results for different versions of three different architectures. In the upper block of the table we have used the same training set as in previous experiments, *i.e.*, 12K/12K from KR/VK\_v2. In the bottom block, as is usual in the literature, we use all the available training data, *i.e.*,  $\sim 40\text{K}$  and  $\sim 22\text{K}$ , respectively.

$\theta^{\text{enc}}$ Backb.	MW	GFLOPS	FPS	abs-rel	sq-rel	rms	1.25	1.25 <sup>2</sup>
ResNet-18	11.6	<b>4.47</b>	<b>141.2</b>	0.116	0.836	4.735	0.860	0.954
ResNet-152	60.2	19.29	30.71	<u>0.108</u>	<u>0.759</u>	4.559	0.870	0.960
HRNet-W18	<u>9.5</u>	8.29	15.79	0.109	0.773	4.524	<u>0.871</u>	0.960
HRNet-W48	65.3	40.04	15.48	<b>0.104</b>	<b>0.721</b>	<b>4.396</b>	<b>0.880</b>	<b>0.962</b>
DenseNet-121	<b>6.9</b>	<u>7.09</u>	<u>32.60</u>	0.116	0.812	4.646	0.854	0.960
DenseNet-161	26.5	19.21	24.87	0.111	0.763	<u>4.516</u>	0.864	<u>0.960</u>
ResNet-18	11.6	<b>4.47</b>	<b>141.2</b>	0.114	0.838	4.734	0.860	0.954
ResNet-152	60.2	19.29	30.71	<u>0.104</u>	0.784	4.560	<u>0.878</u>	0.960
HRNet-W18	<u>9.5</u>	8.29	15.79	0.105	<u>0.745</u>	4.470	0.877	0.961
HRNet-W48	65.3	40.04	15.48	<b>0.101</b>	<b>0.703</b>	<b>4.413</b>	<b>0.882</b>	<b>0.962</b>
DenseNet-121	<b>6.9</b>	<u>7.09</u>	<u>32.60</u>	0.111	0.786	4.536	0.870	0.960
DenseNet-161	26.5	19.21	24.87	0.109	0.760	<u>4.440</u>	0.873	<u>0.962</u>

### 3. Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision

---

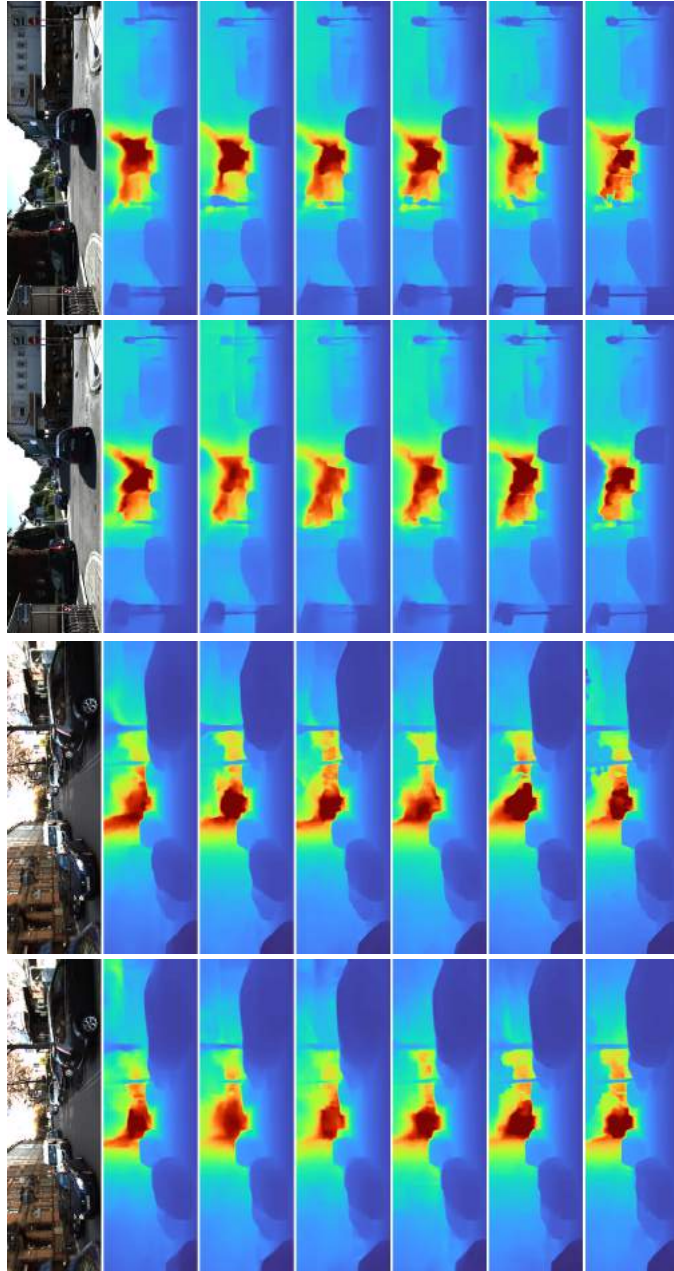


Figure 3.12 – Comparison between MonoDEVSNet models trained with different network architectures.

## 3.5 Conclusion

For on-board perception, we have addressed monocular depth estimation by virtual-world supervision (MonoDEVS) and real-world SfM-inspired self-supervision; the former compensating for the inherent limitations of the latter. This challenging setting allows to rely on a monocular system not only at testing time, but also at training time; a cheap and scalable approach. We have designed a CNN, MonoDEVSNet, which seamlessly trains on real- and virtual-world data, exploiting semantic and depth supervision from the virtual-world data, and addressing the virtual-to-real domain gap by a relatively simple approach which does not add computational complexity in testing time. We have performed a comprehensive set of experiments assessing quantitative results in terms of relative and absolute depth, generalization, and we show the relevance of the components involved on MonoDEVSNet training. Our proposal yields state-of-the-art results within the SfM-based setting, even outperforming stereo-based self-supervised approaches. Qualitative results also confirm that MonoDEVSNet properly captures the depth structure of the images. As a result, we show the usefulness of leveraging virtual-world supervision to ultimately reach the upper-bound performance of methods based on LiDAR supervision.



## 4 On the Metrics for Evaluating Monocular Depth Estimation

---

In previous chapters, we evaluated different MDE models. However, MDE is performed to produce 3D information that can be used in downstream tasks related to on-board perception of AVs or driver assistance. Therefore, the question that arises is whether the standard metrics for MDE assessment are a good indicator for future MDE-based driving-related perception tasks. We address this question in this chapter. In particular, we take the task of 3D object detection on point clouds as proxy of on-board perception. We train and test state-of-the-art 3D object detectors using 3D point clouds coming from MDE models. We confront the ranking of object detection results with the ranking given by the depth estimation metrics of the MDE models. We conclude that, indeed, MDE evaluation metrics give rise to a ranking of methods which reflects relatively well the 3D object detection results we may expect. Among the different metrics, the absolute relative (abs-rel) error seems to be the best for that purpose.

---

### 4.1 Introduction

In previous chapters, we address the problem of MDE from different perspectives defined by the data available at training time. In Chapter 2, the focus is on leveraging different types of supervision from different datasets (*e.g.*, semantics from cityscapes and depth from KITTI). In Chapter 3, the focus is on using SfM self-supervision (monocular sequences from an on-board camera) and depth supervision from virtual images, thus, not requiring the use of expensive sensors such as LiDAR. While doing the many experiments involved in this research, we reported our results and compare them with those in the state-of-the-art. This has been done by using *de facto* standard metrics for evaluating depth estimation, *e.g.*, abs-rel, rms, *etc.* (see Equations 2.1-2.6). We can see examples of such comparison of results both in Table 2.1 and Table 3.3. Looking at these quantitative results, we can rank the different MDE models according to one or several metrics. For instance, we can rank them by abs-rel or rms metrics. However, the difference is not too large even the way of training the model is quite different. For

## 4. On the Metrics for Evaluating Monocular Depth Estimation

---

instance, let us focus on the abs-rel values for MonoDEVNet/VK\_v2 and DORN in Table 3.3. We can see that for DORN abs-rel=0.098, while for MonoDEVNet abs-rel=0.104. The former is based on LiDAR supervision, the latter does not use LiDAR for training, just monocular sequences and virtual images. In other words, eventually DORN requires a much elaborated sensor suite setting (calibrated camera and LiDAR) for training. Thus, does the difference of  $\sim 0.06$  points justify the use of a LiDAR-based setting? This kind of question was around, but not the focus, in previous chapters. Note that, when performing MDE on-board an autonomous or assisted vehicle, obtaining depth estimation maps is just an intermediate step of a perception stack. Thus, one may wonder if those differences on depth estimation will be consolidated once such MDE models are used to support the targeted perception task. The aim of this chapter is to do a step forward to answer this question, using 3D object detection as targeted perception task.

More specifically, we use different MDE models to generate depth maps. These depth maps are then used to generate 3D point clouds, which we term as Pseudo-LiDAR in analogy with LiDAR point clouds. Pseudo-LiDAR is used for training and testing 3D object detectors. We compare the ranking of MDE models according to their performance estimating depth, and their performance supporting 3D object detection through the generation of Pseudo-LiDAR. We consider eight MDE models working at two resolutions (low and high), as well as three different CNN architectures for 3D object detection (Point R-CNN [118], Voxel R-CNN [27], CenterPoint [144]). After analysing our experimental results, based on KITTI benchmark [38], we have seen that the abs-rel metric is well aligned with 3D object detection results in terms of ranking the MDE methods. What remains as future work is to investigate if we can predict accuracy improvements in 3D object detection from the improvements observed in the abs-rel metric. Otherwise, we recommend to incorporate 3D object detection as part of the evaluation of MDE models.

The rest of this chapter is organized as follows. In Sect. 4.2, we review the state-of-the-art on 3D object detection. Note that we have reviewed the state-of-the-art on MDE in previous chapters, while 3D object detection is new in this chapter. In Sect. 4.3, we present the models and methods we use in our experimental work, namely, the MDE models, the 3D object detection models, and the procedure to generate Pseudo-LiDAR from depth maps. In Sect. 4.4, we present quantitative and qualitative results. In particular, we present the performance of the MDE models for both estimating depth and supporting 3D object detection. This allows to compare the rankings of MDE generated by MDE metrics *vs.* 3D object detection metrics, which is also done in this section. Finally, Sect. 4.5 summarizes the work presented and the conclusions derived.

## 4.2 Related Work

### 4.2.1 2D Object Detection

Given an image, object detection is the task of locating instances of semantic classes. With the recent advancement in deep learning, object detection has witnessed significant improvement over time. There have been two important contributions to solving the 2D object detection problems, namely, R-CNN [40] and YOLO [103].

The Region-based Convolutional Neural Networks (R-CNNs)—known as a two-stage detector—consists of two main stages: region proposal and detection network. Initially, in the region proposal stage, the network provides the region of interest (ROI) of an image based on some assumptions, such as texture, size, and pixel intensities. Later the extracted feature vector for each region-proposal candidate is passed to the detection module to classify the object and create a bounding box (BB) for each object. Nevertheless, training time is expensive as it has to process each region proposal individually at a time. Considering this drawback of R-CNN, Fast R-CNN [39] was developed to reduce the training time by running the neural network once on the whole image. This pioneering work inspired developing faster networks to solve 2D object detection [51, 94, 106].

Another significant contribution to 2D object detection is YOLO, a single-stage detector. Unlike the region proposal-based methods, YOLO sees the entire image by dividing it into  $N \times N$  grids and produces a class probability map to each grid cell. YOLO builds the confidence value with the possible object class per grid and localizes with a BB in a single step using a regression model. This work inspired further extensions to develop end-to-end frameworks [76, 79, 102, 104, 121, 126, 127]. Hence, the single-stage and two-stage detectors have emerged as a classical 2D and 3D object detection model by combining prior knowledge and labeled datasets.

### 4.2.2 3D Object Detection

3D object detection plays a significant role in autonomous driving tasks such as path planning, motion prediction, collision avoidance, *etc.* Like in 2D, 3D object detection can also be solved as a single-stage detector [70, 71, 140, 157] which tends to be faster at the inference time, while the two-stage detector [7, 12, 13, 27, 73, 118, 144, 147] tends to be more accurate. Besides, We can further categorize the 3D object detection task into two types based on the sensor generating the input data, namely, LiDAR [27, 74, 118, 142, 144, 157] and vision [7, 12, 13, 72, 73, 99, 130, 147] based. Currently, LiDAR-based performs better than vision-based ones. But due to the high sparseness, irregularity of the point cloud, computationally expensive, and limited semantic information, the research focus is shifting towards vision-based methods [100]. Since in this chapter we strive for accuracy, we analyze LiDAR and vision-based two-stage detectors in the rest of this section.



## 4. On the Metrics for Evaluating Monocular Depth Estimation

---

### LiDAR-based Methods

With the progress in deep learning, training a supervised 2D/3D object detector has become reliable. But by directly applying CNNs on the LiDAR point cloud data, the models suffer to learn shape information due to sparsity and high variance of the configuration of the 3D points. Hence learning feature vectors from the sparse point cloud is a fundamental task for LiDAR-based methods, which leads to the development of 3D CNNs [74] (inspired from PointNet [97]). Moreover, these methods can be classified as point-based and voxel-based according to how they represent the raw 3D point clouds.

**LiDAR point-based methods:** These methods generally work directly with the unstructured form of the raw point clouds coming from LiDARs. They preserve the geometry information of the 3D scene as much as possible. Nevertheless, retrieving a 3D point from the point cloud data is computationally expensive compared with volumetric grids [80]. PointNet pioneered using point cloud data to learn 3D representation for classification and segmentation tasks without converting input data into voxels or another structure. PointNet++ [98] extended it further by running PointNet recursively in a hierarchical manner to capture local structures and granular patterns of the point cloud. Hence, point-based methods such as Point R-CNN [118], 3DSSD [142] started using PointNet++ as a backbone network to extract pointwise features for 3D object detection tasks. In Point R-CNN, the extracted 3D features from PointNet++ helps to produce object proposals as 3D BBs. Using additional 3D CNNs, the new features categorize 3D BB proposals and, then, each proposal is classified, and its BB is refined too (aiming for a better adjustment). The limitation of Point R-CNN is the inference time: both the backbone (PointNet++) and refinement modules in stage two are time-consuming.

**LiDAR voxel-based methods:** These methods generally transform the raw point clouds to the a volumetric representation (voxel) in compact shape to efficiently extract point features for 3D object detection via 3D CNNs (also known as voxelization). The voxel-based methods are computationally efficient, sometimes with a drop of accuracy because of information loss during the data quantization process [27,117]. VoxelNet [157] was the first end-to-end trainable network to learn the informative features by dividing the point cloud into equally spaced 3D voxels and processing through Voxel Feature Encoding (VFE) network to extract the features. These 3D features are fed to the region proposal network to construct probability scores and regress 3D BBs. Later, VoxelNet was used as a 3D backbone network for several 3D object detection architectures [27, 144]. Voxel R-CNN [27] uses the voxel grid to compute the 3D features from the VoxelNet and then transform them into 2D features using a 2D CNN. The output of this network is processed by Region Proposal Networks (RPNs) to propose object candidates in the form of 3D BBs. Then, given the 3D features and the 3D BBs, a voxel grouping is performed by a process called *voxel query* (inspired in Ball query [98]). In addition, a PointNet inspired process is applied to obtain grid point features, which are fed to fully connected networks

(FCNs) to perform the final refinement and classification of the BBs. CenterPoint [144] inspired by CenterNet [156], converts the 3D point cloud into voxels or pillars, using a LiDAR-based backbone network *i.e.* VoxelNet or PointPillars [70]. For instance, using VoxelNet as backbone network, voxels are the input to construct 3D features and flatten them to produce 2D features. Then, the CenterNet keypoint detector is applied on the 2D features to find the object center. After, the anchor-free network makes heatmaps to extract other object properties such as 3D size, orientation, and velocity from the center location. In the second stage, based on the estimated geometric structure of the 3D center, the light-weighted point network extracts the point features at the center of each side of the 3D BBs. All the point-feature vectors, including the feature vector of the center point, are concatenated to pass through MLP layers to improve the confidence of the classification output and regress the 3D BBs.

### Vision-based Methods

These methods only take monocular or stereo images to produce 3D BBs for each object instance. There are two main categories to predict the 3D BBs: template matching and geometric properties-based methods. In the template matching-based methods, we rely extensively on the depth maps computed from stereo images [13], sampling 3D proposals produced by Fast R-CNN and sliding window [12], and the object templates obtained from CAD models [7]. Instead of relying on the templates or 3D proposals, geometric properties-based methods use accurate 2D BBs and estimate 3D pose from the geometric and semantic information acquired by monocular images [72]. In comparison, Stereo R-CNN [73] exploits 3D semantic information by using left and right features to extract better RoI features for the 3D object detection task.

Since the recent high-performing point cloud-based methods [27, 118, 144] are accessible, another way to develop a vision-based method is to replace LiDAR point cloud with reprojected estimated depth maps into 3D space, which is known as Pseudo-LiDAR. Pseudo-LiDAR was first introduced in [130] by producing 3D points in camera coordinates using depth estimation models ((monocular) [33] or (stereo) [8]) to mimic LiDAR, then using these 3D points as input to train 3D object detectors [65, 96]. The conversion from estimated depth maps to 3D points may produce noise that creates misalignment and artifacts. To reduce their impact, Pseudo-LiDAR++ [147] uses depth cost volumes instead of disparity to improve the stereo depth estimation results and later uses cheaper LiDAR sensors to rectify the artifacts to enhance the 3D object detection results. However, these two modules are trained separately. Later, Pseudo-LiDAR-e2e [99] has combined the depth estimator and 3D object detection as a single end-to-end framework by using differentiable Change of Representation (CoR) modules.

In order to perform the research addressed in this chapter, we rely on Point R-CNN [118] (Fig. 4.3), Voxel R-CNN [27] (Fig. 4.4), and CenterPoint [144] (Fig. 4.5).

### 4.3 Methods

In order to assess the usefulness of MDE methods for performing 3D object detection, we consider different MDE approaches and 3D object detectors which work on 3D point clouds. While LiDAR already provides such point clouds, for MDE we have to produce the so-called Pseudo-LiDAR [130] point clouds by properly sampling the respective depth maps. For MDE we consider recent state-of-the-art methods based on self-supervision such as MonoDepth2 [42] and PackNet [44], as well as our MonoDEVSNet (Chapter 3). We also consider state-of-the-art methods based on LiDAR supervision, such as AdaBins [3], and MonoDELSNet, a modification of MonoDEVSNet where we replace virtual-world supervision by LiDAR supervision. In the role of upper-bounds, we consider dense LiDAR data as well as Pseudo-LiDAR based on depth-from-stereo computed by SDNet [147]. Regarding 3D object detection, we consider three relatively different approaches which are Point R-CNN [118], Voxel R-CNN [27], and CenterPoint [144]. Accordingly, in Sect. 4.3.1 we introduce Pseudo-LiDAR generation, in Sect. 4.3.2 we summarize the above mentioned 3D object detectors, and Sect. 4.3.3 introduces our MonoDELSNet variants for MDE. Figure 4.1, briefly illustrates the overall idea of performing 3D object detection from monocular images.

#### 4.3.1 Pseudo-LiDAR Generation

##### From a depth map to a 3D point cloud: $\rho$

In order to generate a Pseudo-LiDAR point cloud,  $\rho$ , from a depth map,  $d$ , estimated from an image,  $I$ , we need the intrinsic parameters,  $\mathcal{K}$ , of the camera that generated this image. More specifically, we need its optical center  $(C_u, C_v)$  and focal length<sup>1</sup>  $F$ , which can be obtained by well-established camera calibration methods [81]. Therefore, we have  $\mathcal{K} = \{C_u, C_v, F\}$ . Given this information, we can assign a 3D point,  $(x, y, z)$ , to each pixel,  $(u, v)$ , of the depth map (and so of the input image) as follows:

$$\begin{aligned} z &\leftarrow d_{u,v} , \\ y &\leftarrow (z/F)(v - C_v) , \\ x &\leftarrow (z/F)(u - C_u) . \end{aligned} \tag{4.1}$$

Therefore,  $\rho$  is generated by applying Eq. 4.1 in all pixels. Afterwards, we can even visualize the 3D point cloud  $\rho$  from different viewpoints.

---

<sup>1</sup>In practice, calibration software allows to estimate a different focal length parameters per image axis, *i.e.*,  $F_u$  and  $F_v$ . However, it is expected that  $F_u \approx F_v$ , since this is basically a numerical trick. Thus, for the sake of simplicity, we keep the idea of using a single focal length parameter.

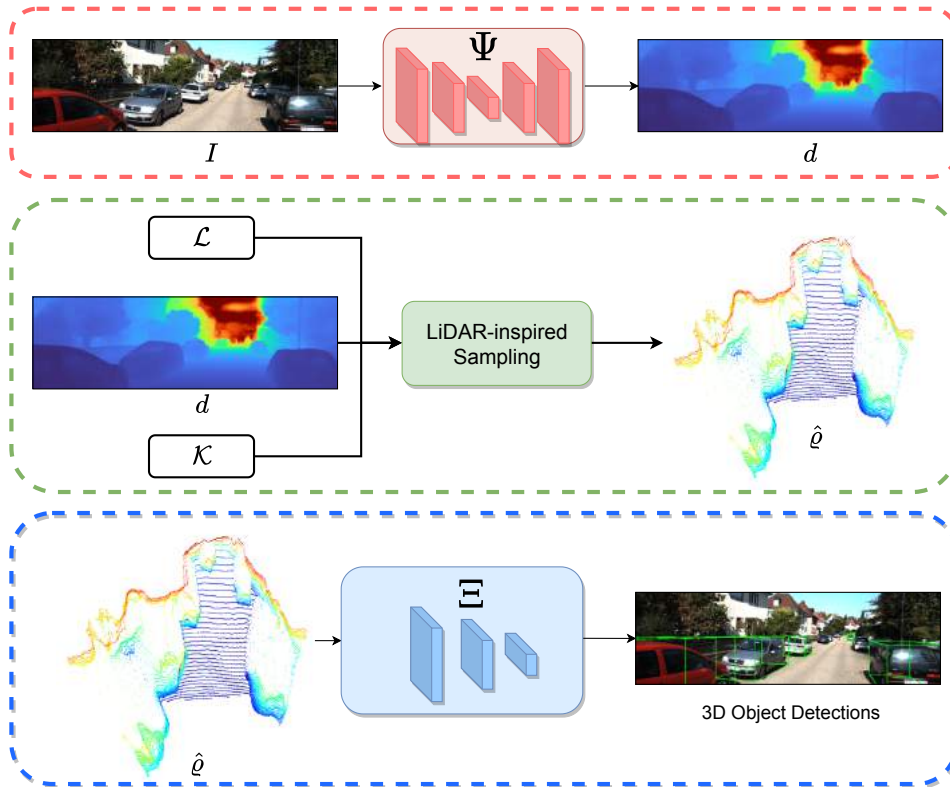


Figure 4.1 – MDE-based 3D object detection in a nutshell. Given an image ( $I$ ) we apply a MDE model ( $\Psi$ ) to generate its corresponding depth map ( $d$ ). With the intrinsic parameters,  $\mathcal{K}$ , of the camera capturing the images, as well as a set of parameters,  $\mathcal{L}$ , defining a LiDAR-inspired sampling procedure, the depth map can be converted to a 3D point cloud,  $\hat{p}$ . Then,  $\hat{p}$  is used to train and test a 3D object detector ( $\Xi$ ) originally developed to work with LiDAR point clouds. The 3D bounding boxes (BBs) of the detected objects are projected to the image just for visualization purposes, but training and inference of the object detectors are done on 3D point clouds. In case of directly working with LiDAR point clouds, only the 3D object detection in the bottom box (discontinuous blue) is required (we may not even have corresponding images to project the BBs). In case of working with stereo data, the processing within the top box (discontinuous red) consists of disparity estimation from the usual left-right stereo pair of images and the final depth computation.

## 4. On the Metrics for Evaluating Monocular Depth Estimation

---

### Sampled Pseudo-LiDAR: $\hat{\rho}$

As we will see in Sect. 4.4 (Table 4.5), directly working with  $\rho$  drives to poor 3D object detection results. We believe that this is because the design of the state-of-the-art 3D object detectors is biased towards the typical 3D pattern distributions present in point clouds captured by actual LiDARs. Therefore, we introduce a LiDAR-inspired sampling procedure which aims at making Pseudo-LiDAR point clouds to be more similar to LiDAR ones. Note that here we are not addressing a domain adaptation problem, since training and testing data will come from the same domain. Instead, we aim at adjusting our generated point clouds to be better suited for training and testing models such as Point R-CNN, Voxel R-CNN, and CenterPoint (Sect. 4.3.2).

Let us introduce the parameters,  $\mathcal{L}$ , required for such LiDAR-inspired sampling. We assume a rotational LiDAR mounted with the rotation axis mainly perpendicular with respect to the road plane. The Velodyne LiDAR HDL 64e used in KITTI dataset, is an example. We term as  $N_b$  the number of beams of the LiDAR under consideration. We term as  $\mathcal{V}$  and  $\mathcal{H}$  the vertical and horizontal field of view (FOV) of the LiDAR, respectively. The vertical angle resolution is  $\mathcal{V}/N_b$  and the horizontal angle resolution,  $\theta_{\mathcal{H}}$ , depends on the rotation mechanism. For instance, for the mentioned Velodyne used in KITTI dataset, we have  $N_b = 64$ ,  $\mathcal{V} \approx 26.9^\circ$ ,  $\mathcal{H} = 360^\circ$ , and  $\theta_{\mathcal{H}} \approx 0.08^\circ$ . Moreover,  $d^{\max}$  and  $h^{\max}$  denote the maximum depth and height we want to consider above the camera, respectively. Finally, it is common to discard the rows of the depth map above a threshold  $r^{\min}$ . Therefore, we have  $\mathcal{L} = \{N_b, \mathcal{V}, \mathcal{H}, \theta_{\mathcal{H}}, d^{\max}, h^{\max}, r^{\min}\}$ .

Now we can think of the sampling procedure as follows. We have a *virtual ray* originated in the camera optical center ( $C_u, C_v$ ). This ray samples the image plane (which is at a distance  $F$  from the principal point), by increments of  $\theta_{\mathcal{H}}$  in its horizontal-component motion, and increments of  $\mathcal{V}/N_b$  in its vertical-component motion. In addition,  $\mathcal{V}, \mathcal{H}, d^{\max}$  and  $h^{\max}$  set bounds in the 3D space to be considered, while  $r^{\min}$  sets a bound in the image space. For the research carried out in this chapter, we have set  $\mathcal{V}$  and  $\mathcal{H}$  so that we consider the full image area,  $d^{\max} = 80\text{m}$  (points with greater depth are not considered),  $h^{\max} = 1\text{m}$  (points with higher height above the camera are not considered), and  $r^{\min}$  is set to discard the top 40% rows of the depth maps. Note also that the mentioned ray will intersect the image plane in sub-pixel coordinates, however, we take the nearest neighborhood approach to select corresponding pixel coordinates. Fig. 4.2 shows what pixels from a depth map would be sampled to generate the final 3D point cloud following Eq. 4.1. We term this Pseudo-LiDAR 3D point cloud as  $\hat{\rho}$ .

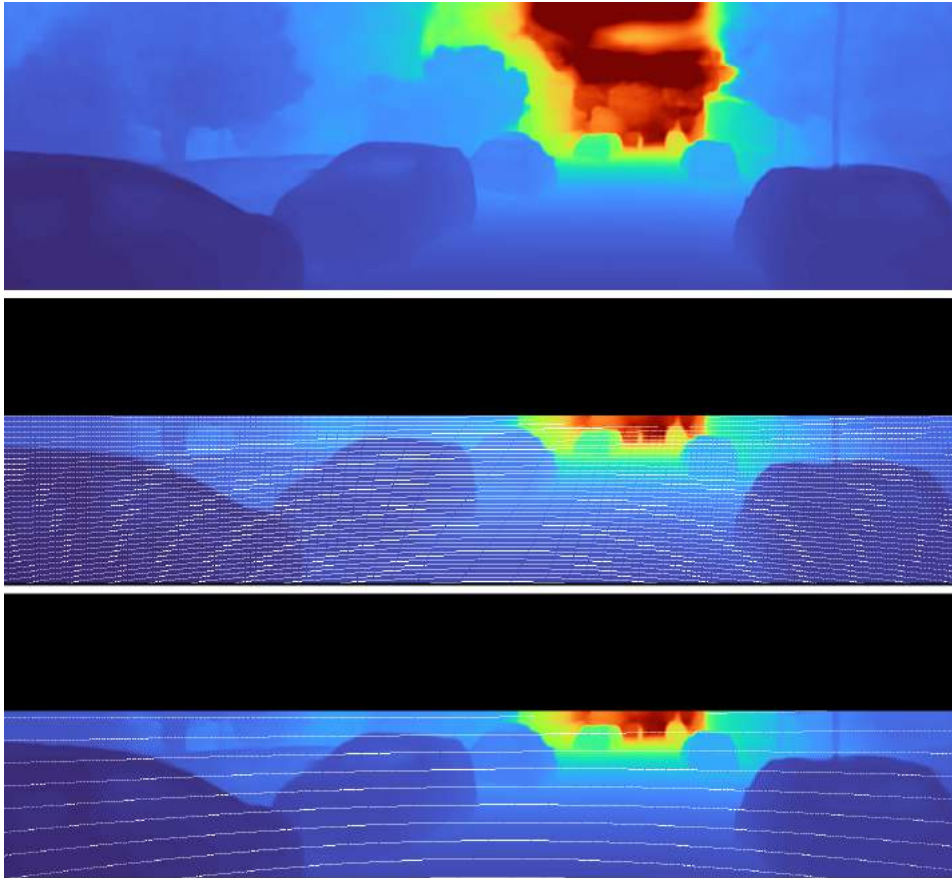


Figure 4.2 – Top: depth map. Mid-Bottom: white pixels would be sampled according to the LiDAR-inspired procedure, for  $N_b = 64$  (Mid) and  $N_b = 16$  (Bottom).

#### 4. On the Metrics for Evaluating Monocular Depth Estimation

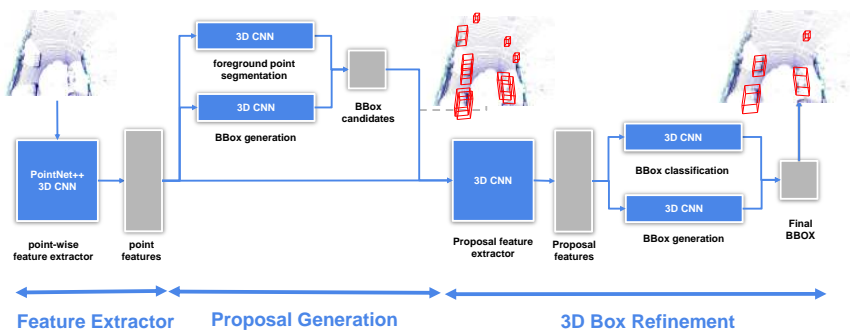


Figure 4.3 – Point R-CNN 3D object detector [118].

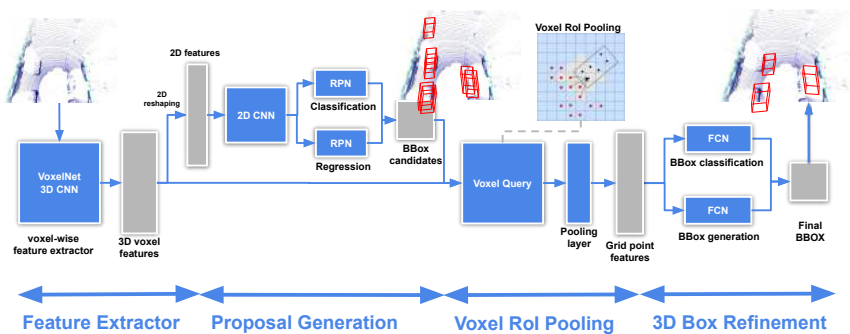


Figure 4.4 – Voxel R-CNN 3D object detector [27].

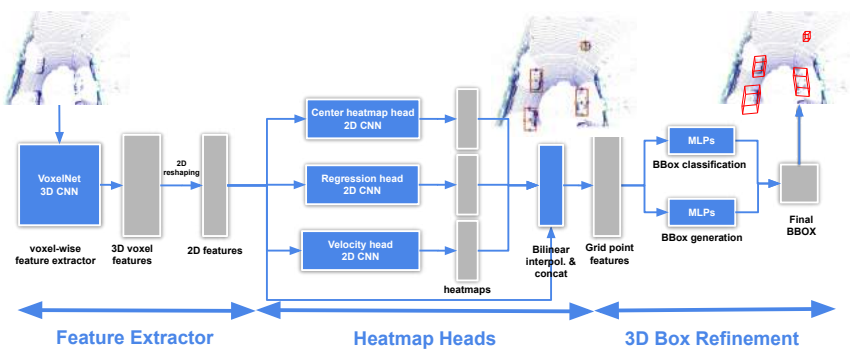


Figure 4.5 – CenterPoint 3D object detector [144].

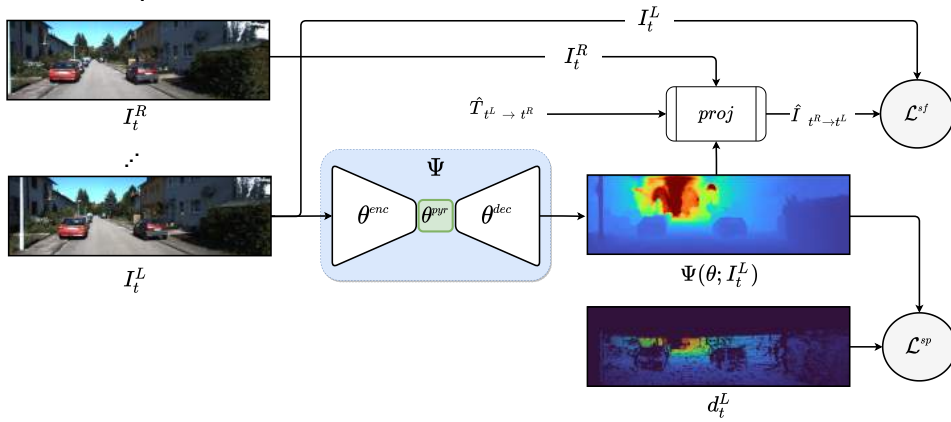


Figure 4.6 – MonoDELSNet-St uses stereo self-supervision and LiDAR supervision for training. See Fig. 3.1 to compare with MonoDEVSNets (in this chapter called MonoDEVSNets-SfM).

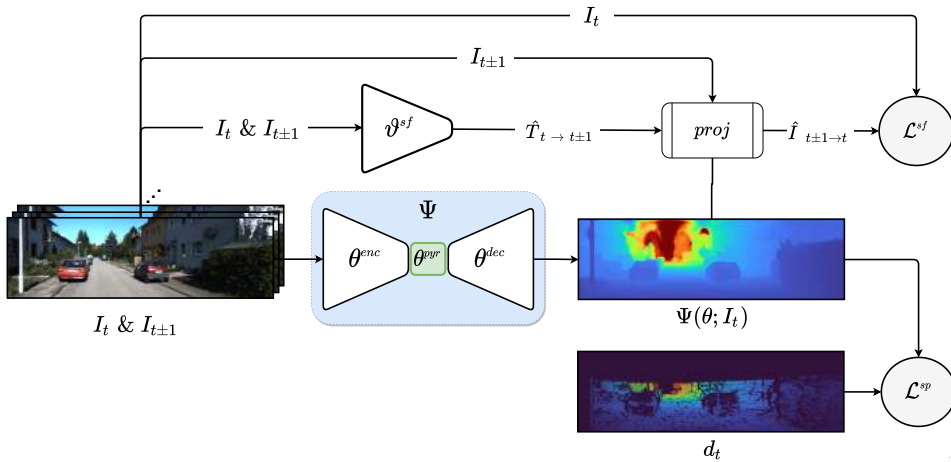


Figure 4.7 – MonoDELSNet-SfM uses SfM self-supervision and LiDAR supervision for training. See Fig. 3.1 to compare with MonoDEVSNets (in this chapter called MonoDEVSNets-SfM).



#### 4. On the Metrics for Evaluating Monocular Depth Estimation

Table 4.1 – Models to produce depth from images. For (\*) a stereo rig is needed at testing time, while for the rest just a monocular system. For (†) absolute depth is obtained via ego-vehicle velocity. MW column stands for millions of model weights. In all cases, these are camera-based models to obtain Pseudo-LiDAR point clouds.

Model	Training Data	Backbone Encoder	MW
SDNet*	LiDAR & Stereo	PSMNet	5.22
AdaBins	LiDAR	EfficientNet B5	78.25
MonoDepth2-St	Stereo	ResNet-18	14.84
MonoDepth2-St+SfM	Stereo & Monocular Seq.	ResNet-18	14.84
PackNet-SfM†	Monocular seq.	PackNet	129.88
MonoDEVSNNet-SfM	Virtual depth & Monocular Seq.	HRNet-w48	93.34
MonoDELSNet-St	LiDAR & Stereo	HRNet-w48	93.34
MonoDELSNet-SfM	LiDAR & Monocular Seq.	HRNet-w48	93.34
MonoDELSNet-SfM/RN	LiDAR & Monocular Seq.	ResNet-18	14.84

Table 4.2 – Instances of *car*, *pedestrian*, and *cyclist*, in KITTI 3D obj. detect. benchmark.

Class	Total	Training set	Validation set
Car	28,742	14,357	14,385
Pedestrian	4,487	2,207	2,280
Cyclist	1,627	734	893

#### 4.3.2 3D object detectors

In order to perform the research of this chapter, we have selected three state-of-the-art 3D object detectors, namely, Point R-CNN [118], Voxel R-CNN [27], and CenterPoint [144]. In terms of Fig. 4.2, any of these 3D object detectors can play the role of  $\Xi$ . Beyond differences in their respective CNN architectures, an important difference for our study is the fact that they rely on different strategies to represent the 3D point clouds. More specifically, Point R-CNN assumes a point-based representation, while Voxel R-CNN and CenterPoint rely on a voxel-based representation. The representation conditions the design of the respective CNN architectures, however, as we can see in Fig. 4.3 (Point R-CNN), Fig. 4.4 (Voxel R-CNN), and Fig. 4.5 (CenterPoint), globally they share a similar approach borrowed from 2D object detection. They have a first stage for extracting features from the point cloud. In the case of Point R-CNN feature extraction relies on PointNet++ [98], while for Voxel R-CNN and CenterPoint it relies on VoxelNet [140, 157]. After feature extraction, there is a stage for proposing candidates to be classified later as objects or rejected as such. The candidates are proposed in the form of 3D bounding boxes (BBs) with attached features. Then there is a final stage where the proposed BBs are classified and their coordinates are refined too.

### 4.3.3 Depth estimation methods

In order to generate point clouds for 3D object detection, we will use the following MDE methods already introduced in Chapter 3: MonoDepth2 [42], PackNet [44], and MonoDEVSNet. In addition, we also consider a recent state-of-the-art method for MDE, AdaBins [3], which uses LiDAR supervision. We also consider two variants of MonoDEVSNet, which we call MonoDELSNet-SfM and MonoDELSNet-St. In the two variants, we replace the supervision coming from the virtual data to supervision coming from LiDAR data, this is why we use the term MonoDELSNet instead of MonoDEVSNet. Moreover, we have also considered the use of stereo-based self-supervision, and we use the term MonoDELSNet-St (Fig. 4.6) in this case, while we use the term MonoDELSNet-SfM (Fig. 4.7) in case of keeping the original SfM-based self-supervision. As we mentioned in Chapter 3, MonoDepth2 can also use self-supervision from either SfM or SfM plus stereo. Therefore, in this chapter, for the sake of terminology coherence, we will use the terms MonoDepth2-SfM and MonoDepth2-St+SfM. For the same reason, we will use the terms MonoDEVSNet-SfM and PackNet-SfM. Obviously, for training the 3D object detectors, we will use raw LiDAR point clouds as upper-bound. Therefore, for completeness we will use a state-of-the-art depth-from-stereo model. In particular, we will use the so-called SDNet [147]. SDNet requires stereo and LiDAR information at training time. At testing time only the stereo images are required to estimate the depth. Thus, comparing to MDE models we can consider SDNet also as an upper-bound. Table 4.1 summarizes the models we consider in this chapter for producing 3D point clouds (Pseudo-LiDAR).

## 4.4 Experimental Results

### 4.4.1 Datasets and evaluation metrics

To train the MDE models in Table 4.1, we follow Chapter 3 in terms of protocol and datasets. Thus, we use the training set of the Eigen *et al.* [30] split of the KITTI Raw [38] dataset, considering the subset established by Zhou *et al.* [154] when using SfM self-supervision. Note that, regarding Table 4.1, the models corresponding to AdaBins, MonoDepth2, and PackNet-SfM, are taken from the authors, while the rest are trained by us. Overall, we ensure that all the models are trained and validated on the same splits.

For evaluating 3D object detectors, we consider KITTI object detection benchmark [38]. Moreover, we use the Chen *et al.* [13] split, consisting of 3,712 images for training and 3,769 for validation. As we have mentioned before, we use Point R-CNN, Voxel R-CNN, and CenterPoint as 3D object detectors. In order to make the most of the hyperparameter tuning done by the respective authors, we use their corresponding framework settings. This implies to train Voxel R-CNN only for the class *car*, while Point R-CNN and Center point will also include the classes *pedestrian* and *cyclist*. Table 4.2 shows the

#### 4. On the Metrics for Evaluating Monocular Depth Estimation

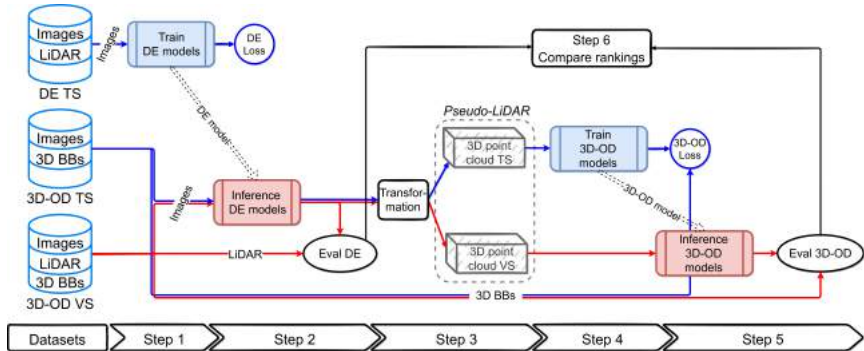


Figure 4.8 – Experiment protocol. DE stands for depth estimation (monocular or stereo based). TS and VS stand for training and validation set, respectively. In our experiments, the DE TS is the training set of the Eigen *et al.* [30] split, the 3D-OD TS and 3D-OD VS are the training and validation sets of the Chen *et al.* [13] split. Thus, the datasets are from KITTI benchmark [38]. Eval DE is based on the standard metrics to evaluate depth estimation (abs-rel, rms, *etc.*), while Eval 3D-OD is based on metrics to assess 3D object detection accuracy ( $AP_{BEV}$ ,  $AP_{3D}$ ). Blue paths work at training time, while red paths work at inference/validation time. In the datasets, images and LiDAR are in correspondence, so that each image has an aligned 3D point cloud. Thus, the same 3D BBs can be used with LiDAR and Pseudo-LiDAR data. Refer to the main text for details.

number of instances of each class in the training and validation sets with associated 3D bounding boxes (BBs). An important question for our experiments is that this total of 7,481 images come with stereo pairs and, even more essential, with corresponding 3D point clouds based on a 64-beams Velodyne LiDAR. Camera calibration matrices are also available. In the rest of the chapter, we term all these data as KITTI-3D-OD.

As in Chapter 3, we use the *de facto* standard metrics to evaluate depth estimation (abs-rel, rms, *etc.*). For 3D object detection we use KITTI metrics [38]. More specifically, we report Average Precision (AP) for 3D and bird-eye-view (BEV), *i.e.*,  $AP_{3D}$  and  $AP_{BEV}$ . Specific results for the detection-difficulty categories used with KITTI metrics are also reported, *i.e.*, for *easy*, *moderate*, and *hard*. In order to compute these APs, we consider an intersection-over-union (IoU) threshold equal to 0.7 (over 1.0) for the easy case, and 0.5 for the moderate and hard cases. Moreover, as we have mentioned above, the *car* class is the only in common for the default configurations of Point R-CNN, Voxel R-CNN, and CenterPoint. We also trained Voxel R-CNN for the multi-class task. However, performance on pedestrians and cyclists is poor. In fact, for Point R-CNN and CenterPoint neither is so good. Thus, we focus the quantitative analysis on car detection, while qualitative results are presented for the classes that each detector considers.

### 4.4.2 Experiment Protocol

The protocol to conduct our experiments is summarized in Fig. 4.8 together with the following description of the **Steps**:

1. **Training depth estimation.** The depth estimation models (Table 4.1) are trained on the training set of the Eigen split (from KITTI Raw dataset).
2. **Testing depth estimation.** The trained depth estimation models are applied to the images of the validation set of the Chen split. Their performance is assessed with the LiDAR-based ground truth (depth) of the same validation set. This is done by collecting the *de facto* metrics on depth estimation accuracy (abs-rel, rms, *etc.*).
3. **Generating 3D point clouds.** The trained depth estimation models are applied to the images of the training set of the Chen split (from KITTI-3D-OD dataset). This generates the corresponding depth maps, which are converted into 3D point clouds as explained in Sect. 4.3.1. We do the same for the images of the validation set of the Chen split. These 3D point clouds are the Pseudo-LiDAR data. Note that KITTI images and LiDAR data are calibrated, so it is right to assume that we can use the same 3D BBs for LiDAR and Pseudo-LiDAR data. Note that, in case of using an on-board vision-based system to fully replace LiDAR, these 3D BBs should be directly annotated in the Pseudo-LiDAR.
4. **Training 3D object detection.** The Pseudo-LiDAR obtained from the images of the training set of the Chen split are used to train the 3D object detection models summarized in Sect. 4.3.2 (*i.e.*, Point R-CNN, Voxel R-CNN, CenterPoint), focusing on the classes of Table 4.2 (*i.e.*, cars, pedestrians, and cyclists).
5. **Testing 3D object detection.** The trained 3D object detectors are applied to the Pseudo-LiDAR obtained from the images of the validation set of the Chen split. The performance is assessed by computing the *de facto* metrics on 3D object detection accuracy ( $AP_{BEV}$ ,  $AP_{3D}$ ).
6. **Depth estimation vs. 3D object detection.** Select a metric on depth estimation and rank the depth estimation models accordingly. Assess if this ranking matches with the rankings that would result from metrics on 3D object detection. The better is the matching, the better the depth estimation metric is for comparing depth models regarding its expected performance when used for 3D object detection.

At the moment of developing this research, there were not other publicly available datasets with the proper ground truth to follow the above mentioned steps. Still, this protocol is not as perfect as we would like. The reason is that there is a  $\sim 25\%$  overlapping

## 4. On the Metrics for Evaluating Monocular Depth Estimation

---

between the training set of the Eigen *et al.* split (used to train the depth estimation models, not the 3D object detectors) and the validation set of the Chen *et al.* split (used in this chapter to test the depth estimation models and the 3D object detectors). This was also recently noted in [119], who propose another split avoiding the problem. Unfortunately, this new split was not publicly available to perform the research of this chapter on time. However, we believe this  $\sim 25\%$  overlapping will not affect our conclusions, since at the end we are comparing rankings.

An additional detail to take into account concerns image resolution, which turns out not to be equal for all images of the KITTI dataset. We can find differences of  $\sim 6$  pixels in the number of rows, and 22 pixels in the number of columns. The LiDAR-based depth maps (ground truth) are generated to have the same resolution as their associated images. This fact together with the differences among the CNN architectures for depth estimation have provoked differences on the working resolution of the depth estimation models. In order to clarify this, we must consider Table 4.3. We see that all the models except AdaBins are prepared to work on two resolutions, namely, high and low. The high one varies a bit from model to model. The low one is the same for all models and has been the standard one used in the literature to research on MDE. Since the resolution of KITTI original images runs on  $[1224 - 1242] \times [370 - 376]$  pixels, working with both high and low resolution involves resizing the KITTI images (*e.g.*, using LANCZOS interpolation) for training and inference. For the high resolution, the resizing is relatively small, for the low resolution there is a strong down-scaling.

For evaluating the depth estimation results (using abs-rel, rms, *etc.*), no matter the working resolution of the depth estimation models, nearest neighbourhood is applied to compare each estimated depth with the ground truth. As we have mentioned before, for each KITTI image its LiDAR-based depth map (ground truth) has the same resolution. Thus, these ground truth maps are of higher resolution than the estimated ones, specially in case of using the low resolution setting. On the other hand, for generating Pseudo-LiDAR, the estimated depth maps are resized to the resolution of the original KITTI images using nearest neighbourhood too. Again, when working at high resolution, this resizing is relatively small, while for low resolution this implies a more significant up-scaling. Finally, AdaBins handles the images of KITTI at their original resolution at training and inference times.

### 4.4.3 Testing depth estimation

After **training depth estimation** models, we can perform the step of **testing depth estimation**. The results are shown in Table 4.4, for both low (standard) and high resolutions. Comparing resolutions, we see that the higher the resolution the better the performance of the same models. For both resolutions MonoDELSNet-SfM outperforms MonoDEVNet-SfM. This was expected since the supervision of the former comes from

Table 4.3 – Working resolutions (in pixels) of the different models we consider in this chapter. (\*) It runs on  $[1224 - 1242] \times [370 - 376]$  pixels.

Model	High Resolution
SDNet	1248 × 384
MonoDepth2-St & MonoDepth2-St+SfM	1024 × 320
PackNet-SfM, MonoDEVSNNet-SfM, MonoDELSNet-St, MonoDELSNet-SfM, & MonoDELSNet-SfM/RN	1280 × 384
	<b>Low Resolution</b>
All models above	640 × 192
AdaBins	KITTI Original*

LiDAR (L) while for the latter comes from virtual (V) depth data. On the other hand, the best performing variant is MonoDELSNet-St, which, in addition to LiDAR supervision, replaces structure-from-motion (SfM) by stereo (St) self-supervision. We remark that this is done at training time, since at testing time these are MDE models. In fact, in the standard resolution, MonoDELSNet-St is the best performing model. At the higher resolution, it still performs the best together with AdaBins, which is also trained with LiDAR supervision. For PackNet, MonoDepth2, and AdaBins we are using models trained by the authors at the respective resolutions. However, for all models the training and testing sets corresponds to the same splits. This was not the case of SDNet, thus, we trained this model for a fair comparison, using high resolution. Since SDNet uses stereo pairs at testing time, it outperforms all the other models (which work on single images). In order to have more models, we have also trained and tested MonoDELSNet-SfM/RN, *i.e.*, using ResNet-18 instead of HRNet-w48 (see Table 4.1). It performs worse than MonoDELSNet-SfM, which is expected since HRNet-w48 uses  $\sim 93$ M weights while ResNet-18 uses  $\sim 18$ M weights. However, this model is still useful in this chapter because we aim at having different MDE models showing different performance to further investigate if this has an impact in 3D object detection.

#### 4.4.4 Evaluation of sampling methods

In order to bound the number of experiments on 3D object detection, we have started by selecting the best performing method for sampling depth maps and so generating 3D point clouds from images. In Sect. 4.3.1 we described two alternatives. One consists in

#### 4. On the Metrics for Evaluating Monocular Depth Estimation

Table 4.4 – Absolute depth results (up to 80m) for the images of the validation set of the Chen *et al.* [13] split. We use the models trained by the corresponding authors for PackNet, MonoDepth2, and AdaBins, the rest are trained by us. All the models are based on the same training and validation sets. SDNet must be considered an upper-bound since it uses a stereo pair at testing time, while all the other models work on single images. Bold stands for **best** and underline for second best.

Model	abs-rel	sq-rel	rms	rms-log	1.25	1.25 <sup>2</sup>	1.25 <sup>3</sup>
Low (standard) resolution (see Table 4.3)							
PackNet-SfM	0.103	0.804	4.780	0.198	0.882	0.954	0.977
MonoDepth2-St	0.097	0.852	4.919	0.205	0.875	0.948	0.974
MonoDepth2-St+SfM	0.096	0.795	4.746	0.192	0.885	0.957	0.978
MonoDEVSNNet-SfM	0.094	0.660	4.297	0.180	<u>0.896</u>	<u>0.964</u>	0.982
MonoDELSNet-SfM	<u>0.082</u>	<u>0.535</u>	<u>3.915</u>	<u>0.169</u>	<b>0.910</b>	<b>0.966</b>	<u>0.983</u>
MonoDELSNet-St	<b>0.079</b>	<b>0.527</b>	<b>3.900</b>	<b>0.167</b>	<b>0.910</b>	<b>0.966</b>	<b>0.984</b>
High resolution (see Table 4.3)							
PackNet-SfM	0.101	0.901	4.774	0.199	0.888	0.955	0.972
MonoDepth2-St	0.096	0.788	4.719	0.198	0.883	0.952	0.976
MonoDepth2-St+SfM	0.095	0.769	4.609	0.188	0.892	0.959	0.979
MonoDEVSNNet-SfM	0.090	0.617	4.107	0.177	0.903	0.966	0.982
MonoDELSNet-SfM	<u>0.077</u>	<u>0.511</u>	3.837	0.172	0.911	0.966	0.982
MonoDELSNet-St	<b>0.073</b>	<b>0.495</b>	<b>3.761</b>	<u>0.166</u>	<u>0.918</u>	<u>0.967</u>	<u>0.983</u>
MonoDELSNet-SfM/RN	0.079	0.537	3.904	0.171	0.909	0.966	0.982
AdaBins	0.080	0.512	<u>3.821</u>	<b>0.164</b>	<b>0.920</b>	<b>0.969</b>	<b>0.984</b>
SDNet	<b>0.044</b>	<b>0.365</b>	<b>3.050</b>	<b>0.136</b>	<b>0.962</b>	<b>0.978</b>	<b>0.987</b>

using all the pixels of the generated depth map for obtaining the corresponding Pseudo-LiDAR, which we called  $\rho$ . The other consists in using a LiDAR-inspired sampling to obtain the Pseudo-LiDAR, which we called  $\hat{\rho}$ . Since we have seen that by working at high resolution we obtain the best depth estimation results (Table 4.4), we focus on this setting to select the sampling method. Moreover, we select two MDE models to bound the number of experiments, namely, AdaBins and MonoDELSNet-SfM. The former is trained with LiDAR supervision (Table 4.1) and is one of the two top-performing methods in depth estimation using high resolution (Table 4.4), while the later also uses SfM self-supervision (Table 4.1) and is the top-performing MDE among those using SfM (Table 4.4). To detect objects, we consider the three approaches introduced in Sect. 4.3.2, *i.e.*, Point R-CNN, Voxel R-CNN, and CenterPoint. Table 4.5 summarizes the results for their common class (car). We can see clearly that the LiDAR-inspired sampling gives rise to significantly better performing object detectors, particularly, when using voxel-based representations as is the case of Voxel R-CNN and CenterPoint.

#### 4.4.5 Testing 3D object detection

According to previous results, the rest of experiments on 3D object detection will rely on the LiDAR-inspired sampling method. Now we consider all the depth estimation methods in Table 4.1, for low and high resolution (Table 4.3), and for Point R-CNN, Voxel R-CNN, and CenterPoint. Table 4.6, Table 4.7, and Table 4.8 present the corresponding quantitative results for the class car. The images used to generate the depth maps (and so the Pseudo-LiDAR  $\hat{\rho}$ ) are from the training set of the Chen *et al.* [13] split, while the validation is performed on the validation set of this split too. Concerning 3D object detection, we can see that MonoDELSNet-SfM and MonoDELSNet-St are consistently outperforming the rest of MDE models, both for low and high resolution. The results are still significantly far from the two upper-bounds, one obtained by working with actual LiDAR point clouds, the other with Pseudo-LiDAR from stereo-based depth estimation (SDNet). If we pay attention to MonoDEVSNet-SfM, the MDE model we developed in Chapter 3 for challenging MDE training conditions, we see that still there is quite a lot of room for improvement if the final target is 3D car detection. Comparing the best performing MDE models, we see that resolution matters, since given the same MDE approach, training on higher resolution gives rise to higher detection performance. Considering only the detection results corresponding to the use of actual LiDAR point clouds, voxel-based methods (*i.e.*, Voxel R-CNN and CenterPoint) outperform Point R-CNN. This is not as clear when using Pseudo-LiDAR, most probably because those must reach higher performances to show such differences.

In addition to the quantitative analysis, Figures 4.9-4.13 present qualitative results. In Fig. 4.9, we show results based on the different MDE methods we consider in this chapter, combined with Point R-CNN. In Fig. 4.10, we show results based on MonoDELSNet-St,



#### 4. On the Metrics for Evaluating Monocular Depth Estimation

Table 4.5 – 3D car detection results ( $AP_{BEV}$ ,  $AP_{3D}$ ) by using different strategies for sampling depth maps, as well as different 3D object detectors (Point R-CNN, Voxel R-CNN, CenterPoint). The sampling is applied to depth maps obtained from MDE models (AdaBins, MonoDELNet-SfM).  $\rho$  stands for Pseudo-LiDAR obtained by using all the pixels of the corresponding depth maps, while  $\hat{\rho}$  stands for Pseudo-LiDAR obtained by applying a LiDAR-inspired sampling to the same depth maps (Fig. 4.2, with  $N_b = 64$ ). L stands for LiDAR point clouds, thus, for the three detectors this setting produces upper-bound results. These LiDAR point clouds have a one-to-one correspondence with the RGB images used to generate the depth maps with the MDE models. Thus, after sampling these maps, the 3D BBs used with L as object supervision, can be also used with  $\rho$  and  $\hat{\rho}$ . The images used to generate the depth maps (and so  $\rho$  and  $\hat{\rho}$ ) are from the training set of the Chen *et al.* [13] split. The LiDAR training data for the setting L is from the same training set. In all cases, the validation is performed on the validation set of this split.

Model	Input	$AP_{BEV}$			$AP_{3D}$		
		easy	<b>mod.</b>	hard	easy	<b>mod.</b>	hard
Point R-CNN							
	L	90.63	89.55	89.35	90.62	89.51	89.28
AdaBins	$\rho$	65.33	41.16	33.42	61.03	39.50	32.27
	$\hat{\rho}$	70.86	48.31	41.12	65.66	45.98	39.56
	Difference $\hat{\rho} - \rho$	↑05.53	↑07.15	↑07.70	↑04.63	↑06.48	↑07.29
MonoDELSNet-SfM	$\rho$	67.87	48.68	40.72	66.21	46.26	38.71
	$\hat{\rho}$	75.58	56.07	48.68	71.15	53.31	45.92
	Difference $\hat{\rho} - \rho$	↑07.71	↑07.39	↑07.96	↑04.94	↑07.05	↑07.21
Voxel R-CNN							
	L	97.33	89.71	89.35	97.29	89.70	89.33
AdaBins	$\rho$	20.22	14.13	13.22	18.07	12.72	11.21
	$\hat{\rho}$	70.77	52.34	46.46	66.18	47.44	43.90
	Difference $\hat{\rho} - \rho$	↑50.55	↑38.21	↑33.24	↑48.11	↑34.72	↑32.69
MonoDELSNet-SfM	$\rho$	12.18	08.72	08.10	09.15	07.19	06.47
	$\hat{\rho}$	74.91	56.07	53.17	71.51	53.11	47.06
	Difference $\hat{\rho} - \rho$	↑63.72	↑48.67	↑46.13	↑64.82	↑48.45	↑42.26
CenterPoint							
	L	95.25	89.88	89.30	95.17	89.85	89.21
AdaBins	$\rho$	11.21	09.09	09.09	09.91	09.09	09.09
	$\hat{\rho}$	67.92	47.21	43.04	63.62	44.85	38.63
	Difference $\hat{\rho} - \rho$	↑56.71	↑38.12	↑33.95	↑53.71	↑35.76	↑29.54
MonoDELSNet-SfM	$\rho$	10.74	09.81	09.09	10.06	09.09	09.09
	$\hat{\rho}$	69.23	52.87	46.46	63.60	48.56	43.29
	Difference $\hat{\rho} - \rho$	↑58.49	↑43.06	↑37.37	↑53.54	↑39.47	↑34.20

combined with Point R-CNN, Voxel R-CNN, and CenterPoint. Figures 4.11, 4.12, and 4.13, show more results using MonoDELSNet-St too. Overall, despite we can see errors (false positives and negatives), we think these are very promising results taking into account we are able to provide relatively accurate 3D object BBs (for the true positives) from single images.

#### 4.4.6 Depth estimation vs. 3D object detection

Despite examining the results of Table 4.6, Table 4.7, and Table 4.8 has a great interest in itself, in this chapter we are even more interested in assessing if ranking MDE models by depth estimation metrics correlates with ranking them according to 3D object detection performance, *i.e.*, using Pseudo-LiDAR generated by MDE. In order to compare rankings, we focus on three different an relevant MDE metrics, namely, abs-rel (which is relative error), rms (in meters), and  $\delta < 1.25$  (which is a %). Concerning the 3D object detection metrics, it is common practice to focus in the moderate setting (**mod.**). Moreover, we see that  $AP_{BEV}$  and  $AP_{3D}$  give rise to analogous rankings in terms of 3D object detection under the moderate setting. Thus, we just focus on  $AP_{BEV}$  for such setting. Accordingly, Fig. 4.14 and Fig. 4.15, compare the  $AP_{BEV} - mod$  ranking with the abs-rel ranking, for all the considered 3D object detection models, both for low and high resolution settings. Fig. 4.16 and Fig. 4.17 are analogous using the rms metric, while Fig. 4.18 is based on  $\delta < 1.25$  metric. Briefly, the correlation between the MDE and object detection rankings can be visually observed by looking at the arrows in these figures. A perfect correspondence between rankings shows as parallel arrows, while the lower the correspondence the more arrows crossing each other. When working on low resolution, it seems that the correspondence is higher than when working at high resolution. However, high resolution is more interesting since we have seen that the produced Pseudo-LiDAR gives rise to better performing 3D object detectors. Then, if we focus in this case, it seems that the abs-rel metric still corresponds well with 3D object detection rankings, while the others (rms,  $\delta < 1.25$ ) are less informative.

#### 4. On the Metrics for Evaluating Monocular Depth Estimation

Table 4.6 – Results on 3D car detection ( $AP_{BEV}$ ,  $AP_{3D}$ ) using Point R-CNN. We complement these results with two depth estimation metrics (abs-rel, rms). Taking Fig. 4.8 as reference,  $AP_{BEV}$  and  $AP_{3D}$  play the role of Eval 3D-OD, while abs-rel and rms are part of Eval DE. Raw LiDAR P. Cloud refers to training with actual LiDAR 3D point clouds, and SDNet estimates depth from stereo images. Thus, these two methods can be seen as upper-bounds for the rest, which generate the corresponding 3D point clouds after performing MDE. Focusing on the MDE models, bold stands for **best** and underline for second best within each resolution block.

Model	$AP_{BEV}$			$AP_{3D}$			DE	
	easy	<b>mod.</b>	hard	easy	<b>mod.</b>	hard	abs-rel	rms
Raw LiDAR P. Cloud	90.63	89.55	89.35	90.62	89.51	89.28	-	-
Low resolution (see Table 4.3)								
PackNet-SfM	43.27	28.93	24.20	39.48	26.92	23.17	0.103	4.780
MonoDepth2-St	59.48	38.93	32.08	54.37	36.28	30.94	0.097	4.919
MonoDepth2-St+SfM	54.97	35.84	30.77	51.09	31.55	27.69	0.096	4.746
MonoDEVSNet-SfM	64.36	44.30	38.86	60.00	39.54	36.22	0.094	4.297
MonoDELSNet-SfM	<u>65.43</u>	<u>47.01</u>	<u>39.82</u>	<u>62.08</u>	<u>43.61</u>	<u>37.94</u>	<u>0.082</u>	<u>3.915</u>
MonoDELSNet-St	<b>66.23</b>	<b>47.24</b>	<b>40.13</b>	<b>63.85</b>	<b>44.42</b>	<b>38.31</b>	<b>0.079</b>	<b>3.900</b>
High resolution (see Table 4.3)								
PackNet-SfM	47.86	32.53	30.50	45.74	31.39	27.84	0.101	4.774
MonoDepth2-St	64.34	40.88	37.21	59.96	39.36	32.41	0.096	4.719
MonoDepth2-St+SfM	58.81	37.68	31.63	53.14	35.37	30.10	0.095	4.609
MonoDEVSNet-SfM	66.17	47.66	45.30	64.33	45.93	40.01	0.090	4.107
MonoDELSNet-SfM	<b>75.58</b>	<b>56.07</b>	<b>48.68</b>	<b>71.15</b>	<b>53.31</b>	<b>45.92</b>	<b>0.073</b>	3.837
MonoDELSNet-St	<u>74.57</u>	<u>54.35</u>	<u>47.25</u>	<u>68.13</u>	<u>47.42</u>	<u>42.84</u>	<u>0.077</u>	<b>3.761</b>
MonoDELSNet-SfM/RN	70.50	48.97	45.41	64.89	47.15	39.87	0.079	3.904
AdaBins	70.86	48.31	41.12	65.66	45.98	39.56	0.080	<u>3.821</u>
SDNet	89.79	77.98	69.62	89.59	75.61	67.36	0.044	3.050

#### 4.4. Experimental Results

Table 4.7 – Analogous to Table 4.6 but using Voxel R-CNN.

Model	$AP_{BEV}$			$AP_{3D}$			DE	
	easy	<b>mod.</b>	hard	easy	<b>mod.</b>	hard	abs-rel	rms
Raw LiDAR P. Cloud	97.33	89.71	89.35	97.29	89.70	89.33	-	-
Low resolution (see Table 4.3)								
Packnet-SfM	48.11	33.71	29.27	42.31	29.68	26.97	0.103	4.780
MonoDepth2-St	59.17	41.88	36.32	54.85	37.38	34.01	0.097	4.919
MonoDepth2-St+SfM	56.12	37.27	34.52	53.01	34.60	29.74	0.096	4.746
MonoDEVSNNet-SfM	63.96	45.08	42.62	60.42	42.47	37.51	0.094	4.297
MonoDELSNet-SfM	<u>65.04</u>	<u>46.85</u>	<u>43.62</u>	<u>62.22</u>	<u>44.68</u>	<u>38.72</u>	<u>0.082</u>	<u>3.915</u>
MonoDELSNet-St	<b>69.32</b>	<b>47.77</b>	<b>44.80</b>	<b>64.20</b>	<b>45.67</b>	<b>42.40</b>	<b>0.079</b>	<b>3.900</b>
High resolution (see Table 4.3)								
PackNet-SfM	55.23	37.12	35.80	52.19	34.71	33.86	0.101	4.609
MonoDepth2-St	65.21	45.74	42.64	62.21	42.54	37.47	0.096	4.719
MonoDepth2-St+SfM	57.98	37.72	35.09	52.87	35.40	30.54	0.095	4.774
MonoDEVSNNet-SfM	65.73	46.90	45.24	63.06	44.69	42.84	0.090	4.107
MonoDELSNet-SfM	<u>74.91</u>	<u>56.07</u>	<u>53.17</u>	<u>71.51</u>	<u>53.11</u>	<u>47.06</u>	<u>0.077</u>	3.837
MonoDELSNet-St	<b>75.90</b>	<b>57.39</b>	<b>54.23</b>	<b>73.97</b>	<b>55.64</b>	<b>48.73</b>	<b>0.073</b>	<b>3.761</b>
MonoDELSNet-SfM/RN	69.94	52.59	46.62	65.42	47.68	43.87	0.079	3.904
AdaBins	70.77	52.34	46.46	66.18	47.44	43.90	0.080	<u>3.821</u>
SDNet	90.35	79.15	76.35	90.28	78.39	70.35	0.044	3.050

#### 4. On the Metrics for Evaluating Monocular Depth Estimation

Table 4.8 – Analogous to Table 4.6 but using CenterPoint.

Model	$AP_{BEV}$			$AP_{3D}$			DE	
	easy	<b>mod.</b>	hard	easy	<b>mod.</b>	hard	abs-rel	rms
Raw LiDAR P. Cloud	95.25	89.88	89.30	95.17	89.85	89.21	-	-
Low resolution (see Table 4.3)								
PackNet-SfM	42.34	29.74	25.92	36.55	26.09	23.25	0.103	4.780
MonoDepth2-St	56.50	39.79	34.97	52.63	35.92	31.93	0.097	4.919
MonoDepth2-St+SfM	54.00	35.49	32.01	50.29	32.32	28.31	0.096	4.746
MonoDEVSNNet-SfM	57.46	40.86	36.28	51.71	38.02	33.85	0.094	4.297
MonoDELSNet-SfM	62.48	<b>45.61</b>	<u>40.99</u>	<u>56.97</u>	<u>42.25</u>	<u>37.14</u>	<u>0.082</u>	<u>3.915</u>
MonoDELSNet-St	<b>62.75</b>	<u>45.38</u>	<b>41.23</b>	<b>58.80</b>	<b>42.69</b>	<b>37.36</b>	<b>0.079</b>	<b>3.900</b>
High resolution (see Table 4.3)								
PackNet-SfM	49.89	35.10	32.10	45.44	32.33	29.10	0.101	4.774
Monodepth2-St	62.14	43.22	37.59	56.43	39.82	34.96	0.096	4.719
MonoDepth2-St+SfM	51.00	35.12	31.72	47.01	31.50	28.12	0.095	4.609
MonoDEVSNNet-SfM	61.13	44.63	42.16	56.66	41.63	37.10	0.090	4.107
MonoDELSNet-SfM	<u>69.23</u>	<u>52.87</u>	<u>46.46</u>	<u>63.60</u>	<u>48.56</u>	<u>43.29</u>	<u>0.077</u>	3.837
MonoDELSNet-St	<b>74.04</b>	<b>56.14</b>	<b>51.96</b>	<b>68.69</b>	<b>53.58</b>	<b>47.27</b>	<b>0.073</b>	<b>3.761</b>
MonoDELSNet-SfM/RN	66.72	49.65	44.82	62.57	45.65	41.14	0.079	3.904
AdaBins	67.92	47.21	43.04	63.62	44.85	38.63	0.080	<u>3.821</u>
SDNet	89.78	77.76	73.75	89.27	75.70	68.51	0.044	3.050

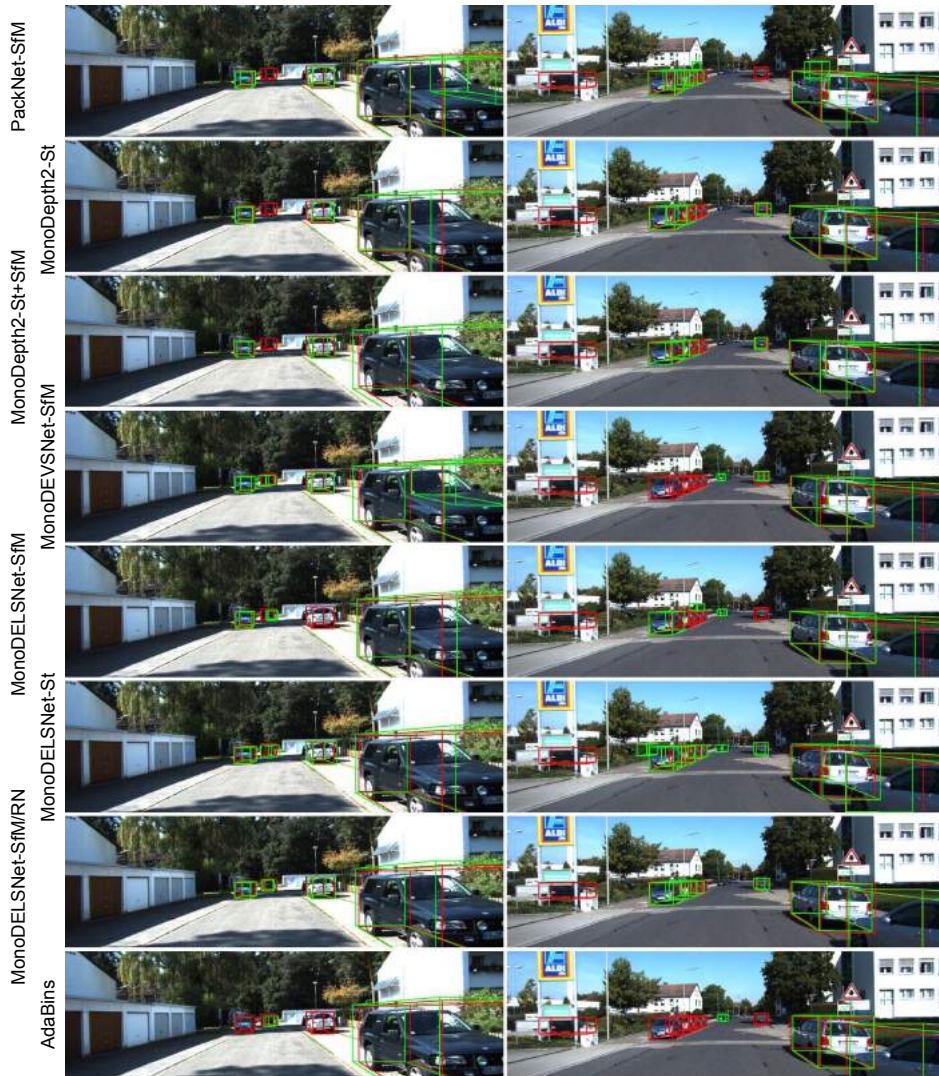


Figure 4.9 – 3D object detection based on the different MDE methods combined with Point R-CNN. Red BBs are ground truth, green ones are detections. Detections are shown for cars, pedestrians, and cyclists.



#### 4. On the Metrics for Evaluating Monocular Depth Estimation

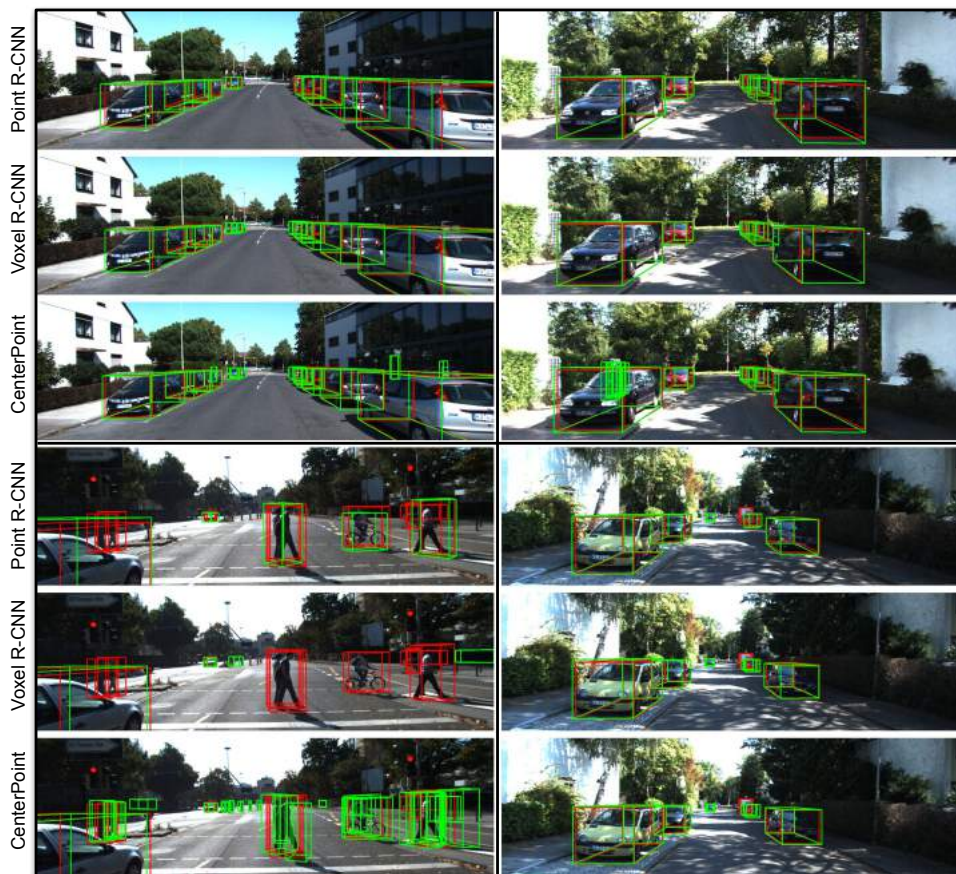


Figure 4.10 – 3D object detection based on MonoDELSNet-St combined with either Point R-CNN, or Voxel R-CNN, or CenterPoint. Red BBs are ground truth, green ones are detections. For Point R-CNN and CenterPoint detections are shown for cars, pedestrians, and cyclists; while Voxel R-CNN only detects cars.



Figure 4.11 – 3D object detection based on MonoDELSNet-St combined with Point R-CNN. Red BBs are ground truth, green ones are detections. Detections are shown for cars, pedestrians, and cyclists.



#### 4. On the Metrics for Evaluating Monocular Depth Estimation

---



Figure 4.12 – 3D object detection based on MonoDELSNet-St combined with Voxel R-CNN. Red BBs are ground truth, green ones are detections. Detections are shown for cars.



Figure 4.13 – 3D object detection based on MonoDELSNet-St combined with CenterPoint. Red BBs are ground truth, green ones are detections. Detections are shown for cars, pedestrians, and cyclists.

#### 4. On the Metrics for Evaluating Monocular Depth Estimation

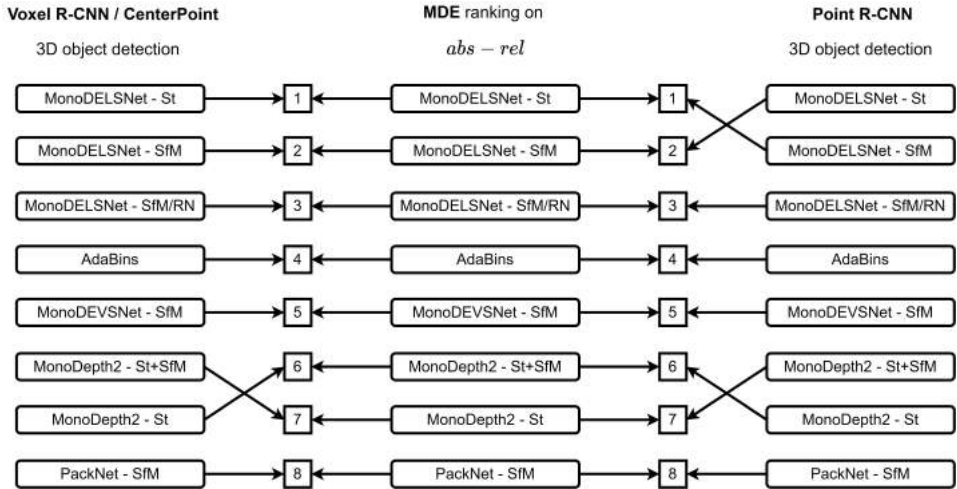


Figure 4.14 – Comparing rankings:  $abs - rel$  (MDE) *vs.*  $AP_{BEV} - mod$  working at high resolution. In the mid column, we have ordered the MDE models from best (top/1) to worse (bottom/8). Then, we have replicated the mid columns as left (voxel-based detectors, which produce the same ranking) and right (Point R-CNN) columns. Afterwards, we have connected the models in left and right columns to its ranking number according to  $AP_{BEV} - mod$ . Thus, a perfect correspondence between rankings shows as parallel arrows, and the lower correspondence the more arrows crossing each other.

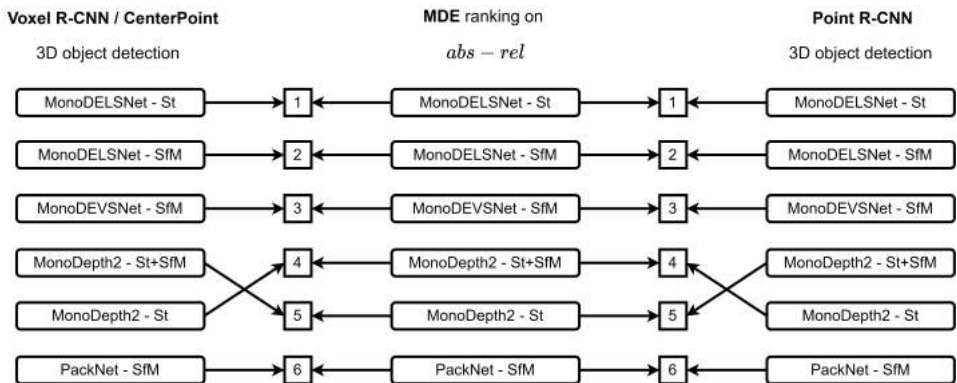


Figure 4.15 – Analogous to Fig. 4.14 for low resolution.

## 4.4. Experimental Results

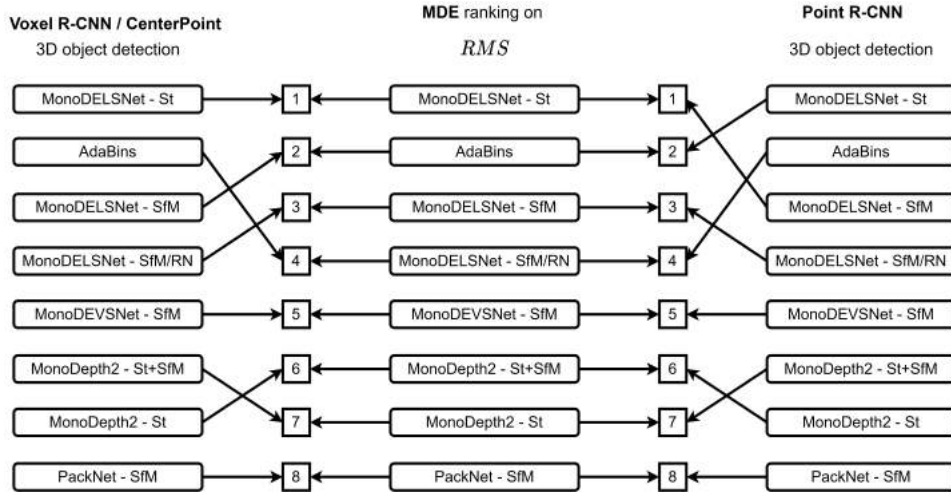


Figure 4.16 – Analogous to Fig. 4.14 using rms metric.

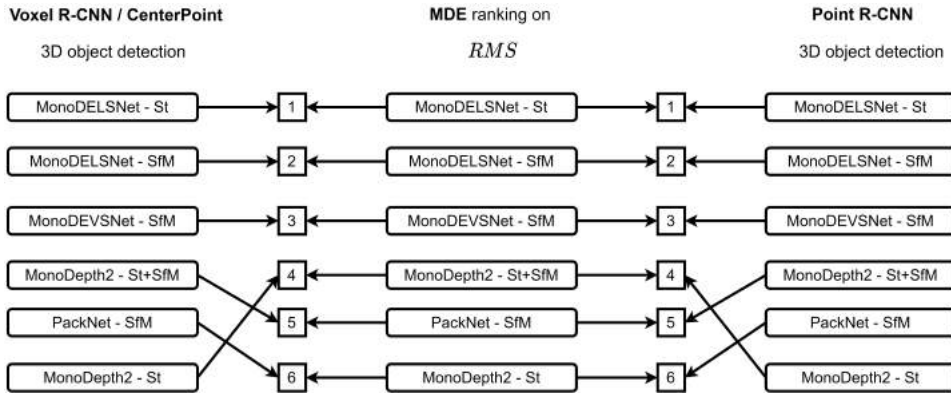


Figure 4.17 – Analogous to Fig. 4.15 using rms metric.



#### 4. On the Metrics for Evaluating Monocular Depth Estimation

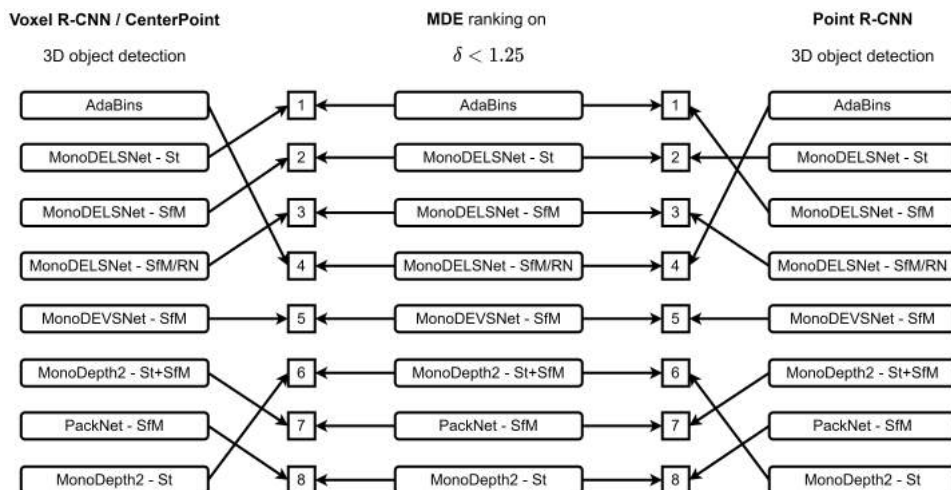


Figure 4.18 – Analogous to Fig. 4.14 using  $\delta < 1.25$  metric.

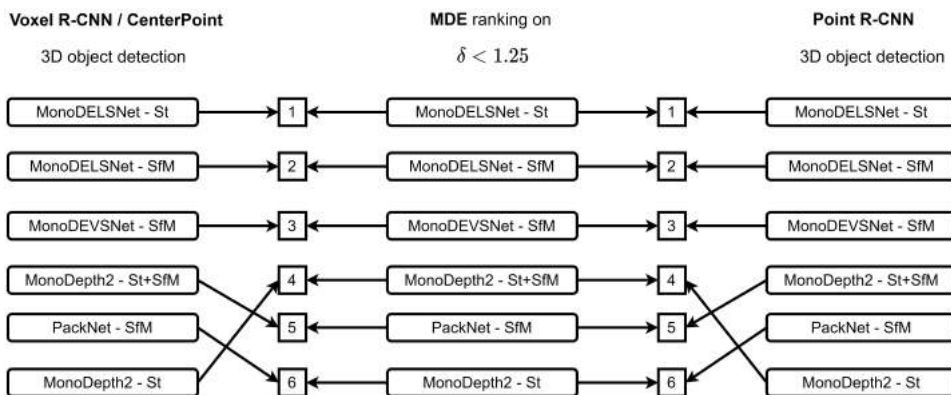


Figure 4.19 – Analogous to Fig. 4.15 using  $\delta < 1.25$  metric.

## 4.5 Conclusion

When performing MDE on-board an autonomous or assisted vehicle, obtaining depth estimation maps is just an intermediate step of a perception stack. For instance, the perception goal may be to perform semantic segmentation or/and object detection with attached depth information to allow for actual vehicle navigation. On the other hand, reviewing the literature of MDE, it is common to see relatively small quantitative differences among the evaluated MDE models. Thus, one may wonder if those differences will be consolidated once such MDE models are used in the targeted perception task. We have addressed this question by using 3D object detection as target perception task. Depth maps based on different MDE models have been converted to a Pseudo-LiDAR 3D point clouds, where 3D object detectors can be trained and tested. We have considered eight MDE models working at two resolutions (low and high), as well as three different CNN architectures for 3D object detection (Point R-CNN, Voxel R-CNN, CenterPoint). Using KITTI benchmark data, we have seen that, indeed, the abs-rel metric commonly used in MDE assessment, is well aligned with 3D object detection results in terms of ranking the MDE methods. What remains as future work is to investigate if we can predict accuracy improvements in 3D object detection (in absolute terms), from the improvements observed in the abs-rel metric. In case this is not possible, we recommend to incorporate 3D object detection as part of the evaluation of MDE models. It is worth to mention that we have also seen that the way depth maps are sampled to produce Pseudo-LiDAR matters because of the dependency that 3D object detectors have on this.



## 5 Conclusions

During the last decade, several research labs from universities as well as the automotive and AI industries have been pushing forward the state of the art of autonomous vehicles (AVs). Given the worldwide interest in mobility solutions based on efficient AVs, more dramatic advances are envisioned in the following decades. In general, AVs are equipped with a suite of sensors to capture appearance (cameras), measure the distance (LiDAR), speed (RADAR), and for better localization (GPS, IMU). These sensor suites are expensive and not trivial to maintain as to be used by millions of AVs. For building efficient AVs, being able to use only cameras would be ideal since these are significantly cheaper and easier to install and maintain compared to other sensors such as LiDARs. However, using only cameras challenges the capture of the 3D information of the driving environment. Note that both semantic and corresponding 3D information is essential to drive safely. For this reason, to capture 3D information, the most common approach so far has been to use LiDARs. Depending on the operational conditions for the AV, calibrated stereo rigs may also be sufficient for obtaining 3D information, being these rigs less expensive and easier to install than LiDARs. However, ensuring proper maintenance and calibration of these rigs is not trivial. Hence, in order to obtain 3D information, the research focus is shifting towards monocular depth estimation (MDE). Accordingly, in this PhD thesis, we researched and developed novel MDE frameworks to improve accuracy under different training settings. Besides, we further analyze several MDE approaches applicable to the downstream tasks, namely, 3D object detection.

Ultimately, high-performing MDE models are based on Convolutional Neural Networks (CNNs) trained with depth supervision. With the aim of obtaining more accurate MDE models, we can include additional supervision while training the MDE CNN; in particular, semantic information about the traffic scene. Depth supervision is usually obtained by having LiDAR and camera calibrated sensors, *i.e.*, depth is not manually annotated. However, pixel-wise semantic supervision relies on manual annotations, which are typically cumbersome to obtain. In practice, it is common to find different datasets with different supervision information. However, the same dataset does not always have all the required supervision (here depth and semantics) associated with the same raw data. Therefore, in the second chapter of this PhD thesis, we show that this



## 5. Conclusions

---

challenging circumstance can be circumvented by training CNNs for MDE by leveraging the depth and semantic information from multiple heterogeneous datasets in a conditional flow approach. In other words, the training process can benefit from a dataset containing only depth supervision for a set of images, together with a different dataset that only contains pixelwise semantic supervision for another set of images. To validate our proposal, we trained CNNs utilizing depth supervision from the KITTI dataset and pixel-level semantic segmentation coming from the Cityscapes dataset. We obtained state-of-the-art results at the time we did this research, thus, confirming that semantic supervision can complement depth supervision even if it comes from different data sources.

On the other hand, as we have mentioned, depth supervision comes from LiDAR sensors. Acquiring it through a calibrated multi-modal suite of sensors is an expensive process not affordable for many teams. Alternatively, training CNNs using monocular cameras is cheaper and highly scalable, and this is possible using depth self-supervision from Structure-from-Motion (SfM) principles. However, this is not enough since SfM can produce depth only up to an unknown scale factor. Besides, SfM-based self-supervision may produce more issues which might lead to depth inaccuracies due to visibility changes (occlusion changes, non-Lambertian surfaces), camouflage (objects moving as the camera may not be distinguished from background), static-camera cases (*i.e.* stopped ego-vehicle), and textureless areas. An interesting approach to compensate for these problems could be training CNNs using a virtual-world dataset with pixelwise accurate depth and semantic segmentation supervision along with SfM-based self-supervision on the real-world data. This approach is what we explore in the third chapter of this PhD thesis. More specifically, the main tasks we performed are the following:

- We designed a CNN, MonoDEVNet, which effortlessly train SfM-based self-supervision using real-world monocular sequences and supervision using precise pixelwise RGB-depth images taken from a virtual world.
- We reduce domain differences between the virtual-world supervision and real-world SfM self-supervision by using the gradient reversal layer (GRL) technique at training time, *i.e.*, this is not needed at testing time.
- We show several experiments with different backbone CNNs to analyze quantitative results on relative and absolute depth. In addition, we describe the importance of each component involved in MonoDEVNet training through an ablative analysis.

In summary, we show MonoDEVNet outperforms state-of-the-art MDE models trained with monocular or stereo self-supervision. According to the qualitative results, MovoDE-

---

VSNet accurately captures the 3D geometric structure of the images, further supporting quantitative results. We think our released code and models<sup>1</sup> will help researchers and practitioners to address applications requiring on-board depth estimation, also establishing a strong baseline to be challenged in the future.

In the second and third chapters of this PhD thesis, we assess MDE models based on standard metrics for this purpose, which focus on MDE as a standalone task. However, MDE is performed to produce 3D information that can be used in downstream tasks related to on-board perception of AVs or driver assistance. Thus, in the fourth chapter, we wonder if the slight performance differences among MDE models usually seen in terms of MDE metrics actually have an impact in the targeted perception task. Due to its great relevance, we consider 3D object detection as such task. We use eight different MDE models working at two image resolutions (low and high) with diverse training protocols to generate MDE-based depth maps. We considered three state-of-the-art frameworks for 3D object detection, namely, Point R-CNN, Voxel R-CNN, and CenterPoint. The estimated depth maps are transformed into point cloud representations (Pseudo-LiDAR) to train and evaluate these 3D object detectors. Using KITTI benchmark data, we show that the performance ranking on 3D object detection is well aligned with one of the metrics used for MDE evaluation, in particular, with the abs-rel metric. Such alignment is worse with others such as RMS or  $\delta < 1.25$ . Overall, we advise to use the 3D object detection task as a component for assessing the performance of MDE models.

During the research conducted in this thesis, we have developed several MDE models by comparing quantitative and qualitative results. Besides, we also examined whether the standard metrics are a good criterion for developing future MDE-based AV-related perception tasks. As a result, several ideas emerge for improving MDE results, which we believe are worth seeking as future work. We think our MonoDEVSNets framework can be further extended with a multi-task setting. Currently, the models take monocular images as input and produce depth as output. However, as a multi-task framework, we think adding further task-specific decoders for semantic and/or instance segmentation, 2D/3D object detection and tracking, scene flow estimation, *etc.*, could provide higher scope to the encoder to learn 3D scene structure while training with the virtual-world dataset. We would also like to improve the performance of 3D object detection based on Pseudo-LiDAR by additionally including training data from virtual-world datasets, *i.e.*, containing 3D object bounding boxes and accurate depth. Besides, we would like to extend our analysis on how MDE performance affects the further downstream tasks by addressing end-to-end driving [136], *i.e.*, rather than assessing perception models assessing sensorimotor models for driving. Finally, in line with our work, we would like

---

<sup>1</sup><https://github.com/HMRC-AEL/MonoDEVSNets>

## 5. Conclusions

---

to apply the ideas to the challenging indoor environment for humanoid robots maneuver or VR/AR applications.

# A Appendix

## List of Contributions

### International Journals and Magazines

- **Akhil Gurram**, Ahmet Faruk Tuna, Fengyi Shen, Onay Urfalioglu, and Antonio M. López. "Monocular Depth Estimation through Virtual-world Supervision and Real-world SfM Self-Supervision." in *IEEE Transactions on Intelligent Transportation Systems*, 2021.
- **Akhil Gurram**, Onay Urfalioglu, Ibrahim Halfaoui, Fahd Bouzaraa, and Antonio M. Lopez. "Semantic Monocular Depth Estimation Based on Artificial Intelligence." in *IEEE Intelligent Transportation Systems Magazine*, 2020.
- Yi Xiao, Felipe Codevilla, **Akhil Gurram**, Onay Urfalioglu, and Antonio M. López. "Multimodal end-to-end Autonomous Driving." in *IEEE Transactions on Intelligent Transportation Systems*, 2020.

### International Conferences

- Fengyi Shen, **Akhil Gurram**, Ahmet Faruk Tuna, Onay Urfalioglu, and Alois Knoll. "TridentAdapt: Learning Domain-invariance via Source-Target Confrontation and Self-induced Cross-domain Augmentation." in *British Machine Vision Conference (BMVC)*, 2021.
- **Akhil Gurram**, Onay Urfalioglu, Ibrahim Halfaoui, Fahd Bouzaraa, and Antonio M. López. "Monocular Depth Estimation by Learning from Heterogeneous Datasets." in *IEEE Intelligent Vehicles Symposium (IV)*, 2018.

### Patents

- **Akhil Gurram**, Onay Urfalioglu: Domain Adaptation based on Self-supervised

## Appendix A. Appendix

---

Depth and Relative-Pose Estimation. European Patent - 2020. Pending/Filed-86937254.

- Onay Urfalioglu, **Akhil Gurram**, Ibrahim Halfaoui: Sampling-based Self-Supervised Depth and Pose Estimation. European Patent - 2020. Pending/Filed-86934297.
- Onay Urfalioglu, **Akhil Gurram**, Fahd Bouzaraa: Learnable Localization using Images. European Patent - 2019. WO2020182297A1

## Bibliography

- [1] Hernán Badino, Uwe Franke, and David Pfeiffer. The stixel world - a compact medium level representation of the 3D-world. In *DAGM: Joint Pattern Recognition Symposium*, 2009.
- [2] M. Bauman. Why waiting for perfect autonomous vehicles may cost lives. RAND Corporation, 2017.
- [3] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [4] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Trans. on Intelligent Vehicles*, 2(3):194–220, 2017.
- [5] Yohann Cabon, Naila Murray, and Martin Humenberger. Virtual KITTI 2. preprint arXiv:2001.10773, 2020.
- [6] Y. Cao, Z. Wu, and C. Shen. Estimating depth from monocular images as classification using deep fully convolutional residual networks. *IEEE Trans. on Circuits and Systems for Video Technology*, 2017.
- [7] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teuliere, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2040–2049, 2017.
- [8] Jia-Ren Chang and Yong-Sheng Chen. Pyramid stereo matching network. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [9] D. Cheda, D. Ponsa, and A.M. López. Pedestrian candidates generation using monocular cues. In *Intelligent Vehicles Symposium (IV)*, 2012.
- [10] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.

## Bibliography

---

- [11] Po-Yi Chen, Alexander H. Liu, Yen-Cheng Liu, and Yu-Chiang Frank Wang. Towards scene understanding: Unsupervised monocular depth estimation with semantic-aware representation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [12] Xiaozhi Chen, Kaustav Kundu, Ziyu Zhang, Huimin Ma, Sanja Fidler, and Raquel Urtasun. Monocular 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2156, 2016.
- [13] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *Neural Information Processing Systems (NeurIPS)*, 2015.
- [14] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. Grad-Norm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *International Conference on Machine Learning (ICML)*, 2018.
- [15] Bin Cheng, Inderjot Singh Saggu, Raunak Shah, Gaurav Bansal, and Dinesh Bharadia. S3Net: Semantic-aware self-supervised depth estimation with monocular videos and synthetic data. In *European Conference on Computer Vision (ECCV)*, 2020.
- [16] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Tom Drummond, Hongdong Li, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. In *Neural Information Processing Systems (NeurIPS)*, 2020.
- [17] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. StarGAN v2: Diverse image synthesis for multiple domains. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [18] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes dataset for semantic urban scene understanding. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [19] Marius Cordts, Timo Rehfeld, Lukas Schneider, David Pfeiffer, Markus Enzweiler, Stefan Roth, Marc Pollefeys, and Uwe Franke. The stixel world: A medium-level representation of traffic scenes. *Image and Vision Computing*, 68:40–52, December 2017.

- [20] David Crandall, Andrew Owens, Noah Snavely, and Dan Huttenlocher. Discrete-continuous optimization for large-scale structure from motion. In *CVPR 2011*, pages 3001–3008. IEEE, 2011.
- [21] Gabriela Csurka. *A Comprehensive Survey on Domain Adaptation for Visual Applications*, chapter 1. *Advances in Computer Vision and Pattern Recognition*. Springer, 2017.
- [22] J.E. Cutting and P.M. Vishton. Perceiving layout and knowing distances: The integration, relative potency, and contextual use of different information about depth. In W. Epstein and S. Rogers, editors, *Handbook of perception and cognition - Perception of space and motion*. Academic Press, 1995.
- [23] T. Dang, C. Hoffmann, and C. Stiller. Continuous stereo self-calibration by camera parameter tracking. *IEEE Trans. on Image Processing*, 18(7):1536–1550, 2009.
- [24] Raul de Queiroz Mendes, Eduardo Godinho Ribeiro, Nicolas dos Santos Rosa, and Valdir Grassi Jr. On deep learning techniques to boost monocular depth estimation for autonomous navigation. *Robotics and Autonomous Systems*, 136:103701, February 2021.
- [25] Selma E. C. Deac, Ion Giosan, and Sergiu Nedevschi. Curb detection in urban traffic scenarios using lidars point cloud and semantically segmented color images. In *Intelligent Transportation Systems Conference (ITSC)*, 2019.
- [26] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.
- [27] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel R-CNN: Towards high performance voxel-based 3d object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 1201–1209, 2021.
- [28] Sayed H. Dokhanchi, Bhavani S. Mysore, Kumar V. Mishra, and Bjorn Ottersten. Enhanced automotive target detection through RADAR and communications sensor fusion. In *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2021.
- [29] A. Dosovitskiy, G. Ros, F. Codevilla, A.M. López, and V. Koltun. CARLA: An open urban driving simulator. In *Conference on Robot Learning (CoRL)*, 2017.



## Bibliography

---

- [30] D. Eigen, C. Puhrsch, and R. Fergus. Depth map prediction from a single image using a multi-scale deep network. In *Neural Information Processing Systems (NeurIPS)*, 2014.
- [31] Péter Fankhauser, Michael Bloesch, Diego Rodriguez, Ralf Kaestner, Marco Hutter, and Roland Siegwart. Kinect v2 for mobile robot navigation: Evaluation and modeling. In *International Conference on Advanced Robotics (ICAR)*, 2015.
- [32] H. Fu, M. Gong, C. Wang, and D. Tao. A compromise principle in deep monocular depth estimation. preprint arXiv:1708.08267, 2017.
- [33] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [34] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual worlds as proxy for multi-object tracking analysis. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [35] Yaroslav Ganin and Victor Lempitsky. Unsupervised domain adaptation by back-propagation. In *International Conference on Machine Learning (ICML)*, 2015.
- [36] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. Domain-adversarial training of neural networks. *The Journal of Machine Learning Research*, 17(1):2096–2030, 2016.
- [37] R. Garg, V. Kumar, G. Carneiro, and I. Reid. Unsupervised cnn for single view depth estimation: Geometry to the rescue. In *European Conference on Computer Vision (ECCV)*, 2016.
- [38] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: The KITTI dataset. *International Journal of Robotics Research*, 32(11):1231–1237, 2013.
- [39] Ross Girshick. Fast R-CNN. In *International Conference on Computer Vision (ICCV)*, 2015.
- [40] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

- [41] C. Godard, O.M. Aodha, and G.J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [42] Clément Godard, Oisín Mac Aodha, Michael Firman, and Gabriel J. Brostow. Digging into self-supervised monocular depth estimation. In *International Conference on Computer Vision (ICCV)*, 2019.
- [43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Neural Information Processing Systems (NeurIPS)*, 2014.
- [44] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3D packing for self-supervised monocular depth estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [45] Vitor Guizilini, Rui Hou, Jie Li, Rares Ambrus, and Adrien Gaidon. Semantically-guided representation learning for self-supervised monocular depth. In *International Conference on Learning Representation (ICLR)*, 2020.
- [46] Vitor Guizilini, Jie Li, Rares Ambrus, Sudeep Pillai, and Adrien Gaidon. Robust semi-supervised monocular depth estimation with reprojected distances. In *Conference on Robot Learning (CoRL)*, 2020.
- [47] Mahir Gulzar, Yar Muhammad, and Naveed Muhammad. A survey on motion prediction of pedestrians and vehicles for autonomous driving. *IEEE Access*, 9:137957–137969, 2021.
- [48] Akhil Gurram, Ahmet Faruk Tuna, Fengyi Shen, Onay Urfalioglu, and Antonio M. López. Monocular depth estimation through virtual-world supervision and real-world sfm self-supervision. *IEEE Transactions on Intelligent Transportation Systems*, pages 1–14, 2021.
- [49] Akhil Gurram, Onay Urfalioglu, Ibrahim Halfaoui, Fahd Bouzaraa, and Antonio M. López. Monocular depth estimation by learning from heterogeneous datasets. In *Intelligent Vehicles Symposium (IV)*, 2018.
- [50] Miles Hansard, Seungkyu Lee, Ouk Choi, and Radu Patrice Horaud. *Time-of-flight cameras: principles, methods and applications*. Springer Science & Business Media, 2012.
- [51] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask R-CNN. In *International Conference on Computer Vision (ICCV)*, 2017.

## Bibliography

---

- [52] Lei He, Guanghui Wang, and Zhanyi Hu. Learning depth from single images with deep neural network embedding focal length. *IEEE Trans. on Image Processing*, 27(9):4676–4689, 2018.
- [53] Li He, Chuangbin Chen, Tao Zhang, Haife Zhu, and Shaohua Wan. Wearable depth camera: Monocular depth estimation via sparse optimization under weak supervision. *IEEE Access*, 6:41337–41345, 2018.
- [54] D. Hernández, A. Chacón, A. Espinosa, D. Vázquez, J.C. Moure, and A.M. López. Embedded real-time stereo estimation via semi-global matching on the GPU. *Procedia Comp. Sc.*, 80:143–153, 2016.
- [55] D. Hernandez, L. Schneider, P. Cebrian, A. Espinosa, D. Vázquez, A.M. López, U. Franke, M. Pollefeys, and J.C. Moure. Slanted stixels: A way to represent steep streets. *International Journal of Computer Vision*, 127(11):1643–1658, 2017.
- [56] H. Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 30(2):328–341, 2008.
- [57] D. Hoiem, A. Efros, , and M. Hebert. Recovering surface layout from an image. *International Journal of Computer Vision*, 75(1):151–172, 2007.
- [58] D. Hoiem, A.A. Efros, and M. Hebert. Putting objects in perspective. *International Journal of Computer Vision*, 80(1):3–15, 2008.
- [59] Gao Huang, Zhuang Liu, Geoff Pleiss, Laurens Van Der Maaten, and Kilian Weinberger. Convolutional networks with dense connectivity. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, May 2019.
- [60] ITDP. Three revolutions in urban transportation, 2017.
- [61] O.H. Jafari, O. Groth, A. Kirillov, M.Y. Yang, and C. Rother. Analyzing modular CNN architectures for joint depth prediction and semantic segmentation. In *International Conference on Robotics and Automation (ICRA)*, 2017.
- [62] Jianbo Jiao, Ying Cao, Yibing Song, and Rynson Lau. Look deeper into depth: Monocular depth estimation with semantic booster and attention-driven loss. In *European Conference on Computer Vision (ECCV)*, 2018.
- [63] Kourosh Khoshelham and Sander Oude Elberink. Accuracy and resolution of kinect depth data for indoor mapping applications. *Sensors*, 12(2):1437–1454, 2012.

- [64] Diederik P. Kingma and Jimmy L. Ba. Adam : A method for stochastic optimization. In *International Conference on Learning Representation (ICLR)*, 2015.
- [65] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Waslander. Joint 3d proposal generation and object detection from view aggregation. *IROS*, 2018.
- [66] Y. Kuznetsov, J. Stückler, and B. Leibe. Semi-supervised deep learning for monocular depth map prediction. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [67] L. Ladicky, J. Shi, and M. Pollefeys. Pulling things out of perspective. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2014.
- [68] Hamid Laga, Laurent Valentin Jospin, F. Boussaid, and Mohammed Bennamoun. A survey on deep learning techniques for stereo-based depth estimation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2020.
- [69] I. Laina, C. Rupprecht, V. Belagiannis, F. Tombari, and N. Navab. Deeper depth prediction with fully convolutional residual networks. In *Int. Conf. on 3D Vision (3DV)*, 2016.
- [70] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019.
- [71] Bo Li, Tianlei Zhang, and Tian Xia. Vehicle detection from 3d lidar using fully convolutional network. In *Robotics: Science and Systems*, 2016.
- [72] Buyu Li, Wanli Ouyang, Lu Sheng, Xingyu Zeng, and Xiaogang Wang. Gs3d: An efficient 3d object detection framework for autonomous driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1019–1028, 2019.
- [73] Peiliang Li, Xiaozhi Chen, and Shaojie Shen. Stereo R-CNN based 3D object detection for autonomous driving. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [74] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.

## Bibliography

---

- [75] Miao Liao, Feixiang Lu, Dingfu Zhou, Sibozhang, Wei Li, and Ruigang Yang. Dvi: Depth guided video inpainting for autonomous driving. In *European Conference on Computer Vision (ECCV)*, pages 1–17. Springer, 2020.
- [76] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.
- [77] B. Liu, S. Gould, and D. Koller. Single image depth estimation from predicted semantic labels. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.
- [78] F. Liu, C. Shen, G. Lin, and I. Reid. Learning depth from single monocular images using deep convolutional neural fields. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 38(10):2024–2039, 2016.
- [79] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.
- [80] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, 2019.
- [81] Li Long and Shan Dongri. Review of camera calibration algorithms. In Sanjiv K. Bhatia, Shailesh Tiwari, Krishn K. Mishra, and Munesh C. Trivedi, editors, *Advances in Computer Communication and Computational Sciences*, pages 723–732. Springer Singapore, 2019.
- [82] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [83] Moritz Menze and Andreas Geiger. Object scene flow for autonomous vehicle. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2015.
- [84] Michael Meyer and Georg Kuschik. Deep learning based 3D object detection for automotive radar and camera. In *European Radar Conference (EuRAD)*, 2019.
- [85] I. Misra, A. Shrivastava, A. Gupta, and M. Hebert. Cross-stitch networks for multi-task learning. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.

- [86] A. Mousavian, H. Pirsiavash, and J. Košecká. Joint semantic segmentation and depth estimation with deep convolutional networks. In *Int. Conf. on 3D Vision (3DV)*, 2016.
- [87] Ramin Nabati and Hairong Qi. CenterFusion: Center-based radar and camera fusion for 3d object detection. In *Winter conf. on Applications of Computer Vision (WACV)*, 2020.
- [88] Jogendra Nath Kundu, Phani Krishna Uppala, Anuj Pahuja, and R. Venkatesh Babu. AdaDepth: Unsupervised content congruent adaptation for depth estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [89] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Neural Information Processing Systems (NeurIPS)*, 2019.
- [90] David Pfeiffer and Uwe Franke. Towards a global optimal multi-layer stixel representation of dense 3D data. In *British Machine Vision Conference (BMVC)*, 2011.
- [91] Sudeep Pillai, Rareş Ambruş, and Adrien Gaidon. SuperDepth: Self-supervised, super-resolved monocular depth estimation. In *International Conference on Robotics and Automation (ICRA)*, 2019.
- [92] Koutilya PNVR, Hao Zhou, and David Jacobs. SharinGAN: Combining synthetic and real data for unsupervised geometry estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [93] D. Ponsa, A.M. López, F. Lumbreras, J. Serrat, and T. Graf. 3D vehicle sensor based on monocular vision. In *Intelligent Transportation Systems Conference (ITSC)*, 2005.
- [94] Anima Pramanik, Sankar K Pal, J Maiti, and Pabitra Mitra. Granulated rcnn and multi-class deep sort for multi-object detection and tracking. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2021.
- [95] C. Premebida, J. Carreira, J. Batista, and U. Nunes. Pedestrian detection combining RGB and dense LiDAR data. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2014.

## Bibliography

---

- [96] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 918–927, 2018.
- [97] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.
- [98] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30:5099–5108, 2017.
- [99] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End-to-end Pseudo-LiDAR for image-based 3D object detection. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [100] Rui Qian, Xin Lai, and Xirong Li. 3d object detection for autonomous driving: A survey. arXiv preprint arXiv:2106.10823, 2021.
- [101] Rene Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2020.
- [102] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [103] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [104] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.
- [105] E. Rehder, C. Kinzig, P. Bender, and M. Lauer. Online stereo camera calibration from scratch. In *Intelligent Vehicles Symposium (IV)*, 2017.
- [106] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards real time object detection with region proposal networks. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2017.
- [107] S. R. Richter, Z. Hayder, and V. Koltun. Playing for benchmarks. In *International Conference on Computer Vision (ICCV)*, 2017.

- [108] G. Ros, L. Sellart, J. Materzyska, D. Vázquez, and A.M. López. The SYNTHIA dataset: a large collection of synthetic images for semantic segmentation of urban scenes. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [109] Germán Ros. *Visual scene understanding for autonomous vehicles: understanding where and what*. PhD thesis, Comp. Sc. Dpt. at Univ. Autònoma de Barcelona, 2016.
- [110] Anirban Roy and Sinisa Todorovic. Monocular depth estimation using neural regression forest. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [111] A. Saxena, M. Sun, and A.Y. Ng. Make3D: Learning 3D scene structure from a single still image. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 31(5):824–840, 2009.
- [112] Ashutosh Saxena, Jamie Schulte, and Andrew Ng. Depth estimation using monocular and stereo cues. In *Int. Joint Conf. on Artificial Intelligence*, 2007.
- [113] L. Schneider, M. Cordts, T. Rehfeld, D. Pfeiffer, M. Enzweiler, U. Franke, M. Pollefeys, and S. Roth. Semantic stixels: Depth is not enough. In *Intelligent Vehicles Symposium (IV)*, 2016.
- [114] Ole Schumann, Jakob Lombacher, Markus Hahn, Christian Wöhler, and Jürgen Dickmann. Scene understanding with automotive radar. *IEEE Trans. on Intelligent Vehicles*, 5(2):188–203, 2020.
- [115] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. AirSim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics (FSR)*, 2017.
- [116] Yakov Shaharabani. Why FIR sensing technology is essential for achieving fully autonomous vehicles - <https://www.embedded.com/why-fir-sensing-technology-is-essential-for-achieving-fully-autonomous-vehicles/>, Jun 2018.
- [117] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.
- [118] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2019.



## Bibliography

---

- [119] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Peter Kotschieder, and Elisa Ricci. Are we missing confidence in pseudo-lidar methods for monocular 3d object detection? In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [120] V. Srikakulapu, H. Kumar, S. Gupta, and K. S. Venkatesh. Depth estimation from single image using defocus and texture cues. In *National Conference on Computer Vision, Pattern Recognition, Image Processing and Graphics (NCVPRIPG)*, 2015.
- [121] Mingxing Tan, Ruoming Pang, and Quoc V Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10781–10790, 2020.
- [122] W.H.O Team et al. Road traffic injuries - <https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries>, Jun 2021.
- [123] P. Tokmakov, J. Li, W. Burgard, and A. Gaidon. Learning to track with object permanence. In *International Conference on Computer Vision (ICCV)*, 2021.
- [124] J. Uhrig, M. Cordts, U. Franke, and T. Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. In *German Conference on Pattern Recognition (GCPR)*, 2016.
- [125] A. Vedaldi and K. K. Lenc. MatConvNet – convolutional neural networks for MATLAB. In *ACM-MM Conference on Multimedia*, 2015.
- [126] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Scaled-YOLOv4: Scaling cross stage partial network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13029–13038, June 2021.
- [127] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. You only learn one representation: Unified network for multiple tasks. arXiv preprint arXiv:2105.04206, 2021.
- [128] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, April 2020.
- [129] Mei Wang and Weihong Deng. Deep visual domain adaptation: A survey. *Neuro-computing*, 312:135–153, October 2018.
- [130] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-LiDAR from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.

- 
- [131] Yan Wang, Bin Yang, Rui Hu, Ming Liang, and Raquel Urtasun. PLUME: Efficient 3D object detection from stereo images. In *Int. Conf. on Intelligent Robots and Systems (IROS)*, 2021.
- [132] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Trans. on Image Processing*, 13(4):600–612, 2004.
- [133] Michael Warren, David McKinnon, and Ben Upcroft. Online calibration of stereo rigs for long-term autonomy. In *International Conference on Robotics and Automation (ICRA)*, 2013.
- [134] Garrett Wilson and Diane J. Cook. A survey of unsupervised deep domain adaptation. *ACM Transactions on Intelligent Systems and Technology*, 11(5), 2020.
- [135] L. Woensel and G. Archer. Ten technologies which could change our lives: potential impacts and policy implications. European Parliament Research Service, 2015.
- [136] Yi Xiao, Felipe Codevilla, Akhil Gurram, Onay Urfalioglu, and Antonio M. López. Multimodal end-to-end autonomous driving. *IEEE Trans. on Intelligent Transportation Systems*, pages 1–11, 2020.
- [137] Zhen Xie, Shengyong Chen, and Garrick Orchard. Event-based stereo depth estimation using belief propagation. *Frontiers in neuroscience*, 11:535, 2017.
- [138] Dan Xu, Wei Wang, Hao Tang, Hong Liu, Nicu Sebe, and Elisa Ricci. Structured attention guided convolutional neural fields for monocular depth estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [139] L. Xu, C. Lu, Y. Xu, and J. Jia. Image smoothing via l0 gradient minimization. *ACM Trans. on Graphics*, 30(6):174:1–174:12, 2011.
- [140] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.
- [141] F. Yang and W. Choi. Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [142] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3DSSD: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11040–11048, 2020.

## Bibliography

---

- [143] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, Joseph Walsh, et al. Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors*, 21(6):2140, 2021.
- [144] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3D object detection and tracking. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2021.
- [145] Wei Yin, Yifan Liu, Chunhua Shen, and Youliang Yan. Enforcing geometric constraints of virtual normal for depth prediction. In *International Conference on Computer Vision (ICCV)*, 2019.
- [146] Zhichao Yin and Jianping Shi. GeoNet: Unsupervised learning of dense depth, optical flow and camera pose. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [147] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In *International Conference on Learning Representation (ICLR)*, 2019.
- [148] Xiaoxue Zhang and Zhigang Liu. A survey on stereo vision matching algorithms. In *World Congress on Intelligent Control and Automation (WCICA)*, 2014.
- [149] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2020.
- [150] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [151] Shanshan Zhao, Huan Fu, Mingming Gong, and Dacheng Tao. Geometry-aware symmetric domain adaptation for monocular depth estimation. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [152] Wang Zhao, Shaohui Liu, Yezhi Shu, and Yong-Jin Liu. Towards better generalization: Joint depth-pose learning without PoseNet. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [153] Chuanxia Zheng, Tat-Jen Cham, and Jianfei Cai. T2Net: Synthetic-to-realistic translation for solving single-image depth estimation tasks. In *European Conference on Computer Vision (ECCV)*, 2018.

- [154] T. Zhou, M. Brown, N. Snavely, and D. Lowe. Unsupervised learning of depth and ego-motion from video. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [155] X. Zhou, V. Koltun, and P. Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision (ECCV)*, 2020.
- [156] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. arXiv preprint arXiv:1904.07850, 2019.
- [157] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-end learning for point cloud based 3D object detection. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [158] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *International Conference on Computer Vision (ICCV)*, 2017.
- [159] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu. Traffic-sign detection and classification in the wild. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [160] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion. *Acta Numerica*, 26:305–364, 1st May 2017.