UNIVERSITY OF SYDNEY

DOCTORAL THESIS

Image-based 3D Object Detection for Autonomous Driving

Author:
Xinzhu MA

Supervisor:
Prof. Wanli OUYANG
Dr. Dong YUAN
Co-Supervisor:
A. Prof. Luping ZHOU

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Philosophy

in the

Department of Electrical and Information Engineering
Faculty of Engineering

October 24, 2023

Abstract of thesis entitled

Image-based 3D Object Detection for Autonomous Driving

Submitted by

Xinzhu Ma

for the degree of Doctor of Philosophy

at the University of Sydney

in October, 2023

Autonomous driving has the potential to radically change people's lives, improving mobility and reducing travel time, energy consumption, and emissions. Therefore, unsurprisingly, in the last decade both research and industry have put significant efforts to develop self-driving vehicles. As one of the key enabling technologies for autonomous driving, image-based 3D object detection has received a lot of attention and gradually becomes a hot research topic.

In this thesis, we first systematically review the image-based 3D object detection models and propose novel taxonomies to help readers summarize the most commonly used pipelines for image-based 3D detection and deeply analyze each of their components. Then, we build a simple yet compact baseline model to analyze the error types and find the 'localization error' is the bottleneck of this perception task. Consequently, we propose three training strategies (including training samples, labels, and losses) to alleviate this problem.

Except for detecting 3D objects from images directly, we also back-project the image pixels into 3D space and detect the object from the resulting 'pseudo-LiDAR' representation. This scheme outperforms its concurrent counterparts significantly. Furthermore, to explore why 'pseudo-LiDAR' works well, we perform an in-depth investigation and observe that the efficacy of pseudo-LiDAR representation comes from the coordinate transformation, instead of data representation itself. This finding demonstrates the potential of image representation in the 3D object detection task.

Due to the lack of spatial cues, detecting objects in the 3D space from a single image accurately is an ill-posed problem. To mitigate this issue, we propose a simple and effective scheme to introduce the spatial information from LiDAR

signals to the monocular 3D detectors, without introducing any extra cost in the inference phase. In particular, we first transform the LiDAR signals into the image representation and train a LiDAR model with the same architecture as the baseline model. After that, this LiDAR model can serve as the teacher to transfer the learned knowledge to the baseline model, and the experiments show the effectiveness of this scheme. Moreover, to leverage the massive unlabeled data, we also investigate how to apply image-based 3D detection in the semi-supervised setting with the help of LiDAR signals.

In summary, in this thesis, we thoroughly review, analyze, and compare existing image-based 3D detection models. We also propose new image-based 3D detection paradigms with promising performances. Besides, we also show how to use auxiliary LiDAR signals to guide the image-based model learning spatial features and achieve semi-supervised learning. Finally, we discuss some open questions in this research field and point out several promising research directions.

Image-based 3D Object Detection for Autonomous Driving

by

Xinzhu Ma
B.S. and M.S *Dalian University of Technology*

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy

at

University of Sydney
October, 2023

Declaration

I, Xinzhu MA, declare that this thesis titled, "Image-based 3D Object Detection for Autonomous Driving", which is submitted in fulfillment of the requirements for the Degree of Doctor of Philosophy, represents my own work except where due acknowledgement have been made. I further declared that it has not been previously included in a thesis, dissertation, or report submitted to this University or to any other institution for a degree, diploma or other qualifications.

Signed: ____________________________

Date: ____________________________

For Mama and Papa

Acknowledgements

I feel incredibly fortunate to have had a Ph.D. journey that deviated from the stereotypical solitary research experience. Many individuals have contributed to making this journey stimulating, challenging, and enjoyable, and I would like to express my gratitude to them.

First and foremost, I would like to express my deepest appreciation to my lead supervisor, Prof. Wanli Ouyang. His guidance, expertise, and unwavering support have been instrumental in shaping my academic development. I am truly grateful for his countless discussions, feedback, guidance, and encouragement, and I am honored to have had the opportunity to learn from him. Besides, I also feel grateful to work with my co-supervisors Prof. Luping Zhou and Dr. Dong Yuan.

Second, I would like to thank the people who have guided me on my academic journey, including Prof. Zhihui Wang, Prof. Haojie Li, Prof. Dan Xu, and Prof. Cheng Guo. Without the help and support from them, I would not have been able to overcome the difficulties I encountered along the way and achieve my current accomplishments.

I am thankful to the members of SigmaLab and DLUT MM-Lab. They provided a comfortable academic environment and offered their assistance in any aspect whenever needed. I also give my thanks to all my co-authors. Their collaboration and discussions have been an invaluable source of inspiration and knowledge. I am glad to work with them and am proud of the work we have done together.

I would like to acknowledge the support of The University of Sydney for funding my research and SenseTime Research for providing computational resources that significantly facilitated my project.

Finally, I would like to thank my parents, who always steadfastly support me, and give a special thank you to my beloved wife, Yuan Meng, for adding color and joy to my life.

Xinzhu Ma
The University of Sydney
October 24, 2023

List of Publications

Publications:

[1] **Xinzhu Ma**, Zhihui Wang, Haojie Li, Pengbo Zhang, Wanli Ouyang, Xin Fan, "Accurate Monocular 3D Object Detection via Color-Embedded 3D Reconstruction for Autonomous Driving," in *International Conference on Computer Vision (***ICCV***)*, 2019

[2] **Xinzhu Ma**, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, Wanli Ouyang, "Rethinking Pseudo-LiDAR Representation," in *European Conference on Computer Vision (***ECCV***)*, 2020

[3] **Xinzhu Ma**, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, Wanli Ouyang, "Delving into Localization Errors for Monocular 3D Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (***CVPR***)*, 2021

[4] Zhiyu Chong*, **Xinzhu Ma***, Hong Zhang, Yuxin Yue, Haojie Li, Zhihui Wang, Wanli Ouyang, "MonoDistill: Learning Spatial Features for Monocular 3D Object Detection," in *International Conference on Learning Representations (***ICLR***)*, June 2022. (*: equal contribution)

Pre-Prints:

[1] **Xinzhu Ma**, Yuan Meng, Yinmin Zhang, Lei Bai, Jun Hou, Shuai Yi, Wanli Ouyang, "An Empirical Study of Pseudo-Labeling for Image-based 3D Object Detection," *https://arxiv.org/abs/2208.07137*, 2022.

[2] **Xinzhu Ma**, Wanli Ouyang, Andrea Simonelli, Elisa Ricci, "3D Object Detection from Images for Autonomous Driving: A Survey," *https://arxiv.org/abs/2202.02980*, 2022, under review.

Authorship Attribution Statement

Chapter 2 and Chapter 8 are adopted from:

Xinzhu Ma, Wanli Ouyang, Andrea Simonelli, Elisa Ricci, "3D Object Detection from Images for Autonomous Driving: A Survey," *https://arxiv .org/abs/2202.02980*, 2022, under review.

I serve as the first author and am responsible for collecting, comparing, and organizing the literature. Besides, I build the skeleton of this survey paper, propose the taxonomies of the existing works, and write the draft. Dr. Andrea Simonelli also contributes to several sub-sections in preparing the manuscript. Prof. Wanli Ouyang leads and supervises this project. Besides, Prof. Wanli Ouyang, Prof. Elisa Ricci, and Dr. Andrea Simonelli provide insightful comments and revise the manuscript.

Chapter 3 of this thesis is published as:

Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, Wanli Ouyang, "Delving into Localization Errors for Monocular 3D Detection," in *IEEE Conference on Computer Vision and Pattern Recognition (***CVPR***)*, 2021

I serve as the first author and am responsible for proposing the method, designing and implementing the code, conducting the experiments, and writing the draft. Mr. Yinmin Zhang and I develop the codebase together, and he gives me a lot of help with method implementation and draft preparation. Prof. Dan Xu regularly discusses this work with me and provides insightful feedback. Dr. Dongzhan Zhou, Dr. Shuai Yi, and Prof. Haojie Li provide instructive comments for this work. Prof. Wanli Ouyang is the supervisor of this work. He gives lots of valuable comments for this work and puts in a lot of effort in paper revision.

Chapter 4 of this thesis is published as:

Xinzhu Ma, Zhihui Wang, Haojie Li, Pengbo Zhang, Wanli Ouyang, Xin Fan, "Accurate Monocular 3D Object Detection via Color-Embedded 3D Reconstruction for Autonomous Driving," in *International Conference on Computer Vision (**ICCV**)*, 2019

I serve as the first author and am responsible for proposing the method, designing and implementing the code, conducting the experiments, and writing the draft. Prof. Zhihui Wang, Prof. Haojie Li, and Prof. Wanli Ouyang supervise this project, provide lots of valuable feedback, and revise the draft. Mrs. Pengbo Zhang helps me implement the method and prepare the paper draft. Prof. Xin Fan provides insightful comments in several discussions.

Chapter 5 of this thesis is published as:

Xinzhu Ma, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, and Wanli Ouyang, "Rethinking Pseudo-LiDAR Representation," in *European Conference on Computer Vision (**ECCV**)*, 2020

I serve as the first author and am responsible for proposing the method, designing and implementing the code, conducting the experiments, and writing the draft. Mr. Shinan Liu regularly discusses this project with me and provides useful feedback. Mr. Zhiyi Xia helps me pre-process the experimental data and prepare the paper draft. Dr. Hongwen Zhang, Dr. Xingyu Zeng, and Prof. Wanli Ouyang provide valuable comments for this work and also revise the paper for several rounds. Besides, Prof. Wanli Ouyang is the supervisor of this work and provides some key insights.

Chapter 6 of this thesis is published as:

Zhiyu Chong*, **Xinzhu Ma***, Hong Zhang, Yuxin Yue, Haojie Li, Zhihui Wang, Wanli Ouyang, "MonoDistill: Learning Spatial Features for Monocular 3D Object Detection," in *International Conference on Learning Representations (**ICLR**)*, 2022. (*: equal contribution)

As the co-first author, I propose the core idea of this work, provide the first-version code implementation, and revise the paper. Mr. Zhiyu Chong, the co-first author, further implements the idea, conducts the experiments, and prepares the draft. Mrs. Hong Zhang and Mrs. Yuxin Yue also contribute to the method implementation and paper revision. Prof. Zhihui Wang, Prof. Haoji

Li, and Prof. Wanli Ouyang supervise this work, provide valuable feedback regularly, and further revise the paper.

Chapter 7 of this thesis is adopted from:

Xinzhu Ma, Yuan Meng, Yinmin Zhang, Lei Bai, Jun Hou, Shuai Yi, Wanli Ouyang, "An Empirical Study of Pseudo-Labeling for Image-based 3D Object Detection," *https://arxiv.org/abs/2208.07137*, 2022.

I serve as the first author and am responsible for proposing the method, designing and implementing the code, conducting the experiments, and writing the draft. Dr. Yuan Meng provides key insights for this work, help me prepare the draft, and revised the paper. Mr. Yinmin Zhang contributes to the method implementation, experiments, analyses, and paper writing. Dr. Lei Bai, Mrs. Jun Hou, and Dr. Shuai Yi provide instructive comments for this work. Besides, Dr. Lei Bai also helps me revise the paper. Prof. Wanli Ouyang supervises this work, provides valuable feedback regularly, and provides several suggestions to improve this work.

In addition to the statements above, in cases where I am not the corresponding author of a published item, permission to include the published material has been granted by the corresponding author.

As the supervisor for the candidature upon which this thesis is based, I can confirm that the authorship attribution statements above are correct.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter begins by establishing the motivation behind this thesis and providing a clear definition of the image-based 3D object detection task. Subsequently, the key challenges within the field are emphasized, followed by a presentation of our solutions to address these challenges. We then summarize the key contributions made by this thesis. Finally, an outline is provided to offer a comprehensive view of the organization of the subsequent chapters.

1.1 Motivations

Autonomous driving has the potential to radically change people's lives, improving mobility and reducing travel time, energy consumption, and emissions. Therefore, unsurprisingly, in the last decade both research and industry have put significant efforts to develop self-driving vehicles. As one of the key enabling technologies for autonomous driving, 3D object detection has received a lot of attention. Specifically, existing 3D object detection approaches can be roughly categorized into two groups according to whether the input data are images or LiDAR signals (generally represented as point clouds). Although the LiDAR-based methods show promising performances, the expensive and cumbersome sensors restrict the application of these algorithms. Besides, the intrinsic properties of LiDAR sensors also determine that they are difficult to cover some corner cases, *e.g.* long-range objects. Accordingly, the image-based scheme draws lots of attention and gradually becomes a hot topic in both academia and industry. For example, in the past seven years, more than 150 papers have been published in top-tier conferences and journals in this area [151, 123], and the image-based approaches have been applied in real-world automatic/assisted driving systems [181, 182].

1.2 Problem Statement

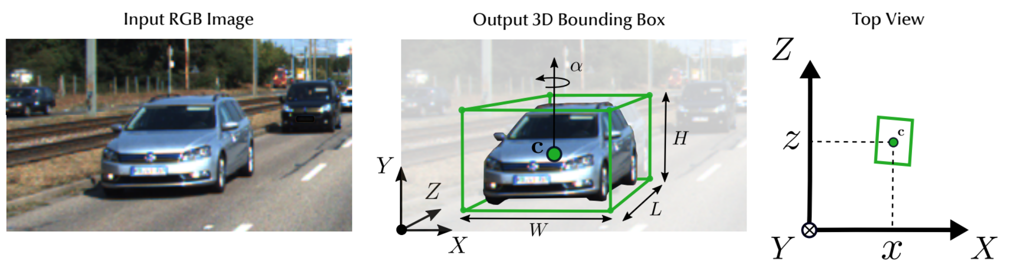


Figure 1.1: Illustration of the image-based 3D object detection task. Given an input image (*left*), it aims to predict a 3D bounding box (represented by its location (x, y, z), dimension (h, w, l), and orientation θ for each object (*middle*). We also show the bird's eye view for better visualization (*right*).

Given the RGB images and corresponding camera parameters, the goal of image-based 3D object detection is to classify and localize the objects of interest. Each object is represented by its category and bounding box in the 3D world space. Generally, the 3D bounding box is parameterized by its location $[x, y, z]$, dimension $[h, w, l]$, and orientation $[\theta, \phi, \psi]$[1] relative to a predefined reference coordinate system (*e.g.* the one of the ego-vehicle which recorded the data). In most autonomous driving scenarios, only the heading angle θ around the up-axis (yaw angle) is considered. Figure 1.1 visualizes an example result on both the 2D image plane and the bird's eye view.

While the general problem of the image-based 3D object detection can be stated as described above, it is worth mentioning that: i): besides the category and 3D bounding boxes, some benchmarks require additional predictions, *e.g.* 2D bounding box for the KITTI dataset [52] and the velocity/attribute for the nuScenes dataset [15]. ii): although only images and camera parameters are initially provided for this task, the adoption of auxiliary data (*e.g.* stereo pairs and LiDAR signals, etc.) is common in this field.

1.3 Issues, Challenges and Our Solutions

Despite significant advancements have been made in the field of image-based 3D object detection [120], there remain numerous issues/challenges that must be addressed to enhance the practicality and applicability of this task.

[1]Location and orientation are also called translation and rotation in some works.

First, 3D object detection from images, which is a fundamental and challenging task in the field of autonomous driving, has garnered significant attention in recent years. With the rapid advancements in deep learning technologies, image-based 3D detection has witnessed remarkable progress, as evidenced by the growing body of research. Particularly, from 2015 to 2021, more than 200 works have explored this problem, covering a wide range of theories, algorithms, and applications. However, despite the extensive research efforts, there is currently a lack of recent surveys that systematically collect and organize this wealth of knowledge. This thesis aims to bridge the existing gap in the literature by presenting the first comprehensive survey of the rapidly expanding research field of image-based 3D detection. It offers a thorough analysis of the most commonly utilized pipelines for this task, delving into the intricacies of each component. Furthermore, a novel taxonomy is proposed to systematically categorize and organize the state-of-the-art methods, providing a more structured and comprehensive review of the existing approaches.

Second, as an emerging and continuously developing field, the community of image-based 3D detection is not well-developed as other tasks, and it is hard for beginners to quickly understand this task and leverage existing materials. To this end, we build a compact and efficient baseline model and quantify the impact introduced by each sub-task through intensive diagnosis experiments. Our diagnosis results give an all-around presentation of the typical 'result-lifting' (see Chapter 2 for the definition) 3D detection. Besides, our baseline model has been open-sourced and is the foundation of several following works [115, 35, 92, 111, 140, 87, 214].

Third, accurately detecting 3D objects from RGB images is an extremely difficult task, and the core problem is the mismatch of the input data and output results. To align this mismatch, we propose to back-project the 2D images into the 3D world space and then detect the objects. This pipeline is significantly superior to its concurrent works in detection accuracy, and we conduct extensive experiments to explore its underlying mechanism. We find the key reason is the 'pseudo-LiDAR' representation lifts the 2D pixels in the 2D image plane into the 3D world space, killing the mismatch between the input and output spaces. In other words, the key point is the space, instead of the representation.

Fourth, as reported in several works [122, 190, 136, 215, 115, 87], the biggest problem for image-based 3D objects (especially for monocular images) is the ambiguity in depth estimation. More particularly, RGB images lack explicit depth information, making it challenging to accurately estimate the 3D position and distance of objects in the scene. Depth ambiguity arises when objects at different distances project similar appearances in 2D images. To alleviate this problem, we propose a detection framework, named MonoDistill, to learn the spatial features for image-based models, under the supervision of LiDAR models with rich geometrical features. The experiments show that the proposed method can significantly boost the accuracy of the depth estimation and the overall performance of 3D object detection. Note that this scheme does not introduce any additional computation cost in the inference phase. Besides, this issue can also be alleviated by geometric modeling, and we also propose our geometric-based solutions in [215, 115] (not included in this thesis).

Last, although collecting raw data for image-based 3D object detection may be relatively straightforward, the manual annotation of objects in 3D space is a complex and labor-intensive process. Moreover, despite advancements, existing models still face challenges in achieving satisfactory performance, and a primary contributing factor is the scarcity of training data [139, 92]. This thesis delves into the potential of a cost-effective alternative, namely pseudo-labeling, to leverage the power of unlabeled data. Extensive experiments are conducted to examine the efficacy of pseudo-labels as a form of supervision for baseline models across different scenarios. The experimental findings not only highlight the effectiveness of the pseudo-labeling mechanism for image-based 3D detection but also uncover several intriguing and significant insights (*e.g.* noisy pseudo-labels work better than the ground truth in the KITTI 3D dataset).

This thesis primarily discusses the aforementioned issues and challenges while presenting our proposed solutions. However, as a rapidly evolving and relatively new field, there are numerous other limitations that necessitate further exploration. Some of these additional limitations are highlighted in Chapter 8. It is our aspiration that this thesis will offer valuable insights and serve as a foundation for future work.

1.4 Contributions

To summarise, the contributions of this thesis are as follows:

- We provide a comprehensive literature review for the image-based 3D object detection field. Specifically, we systemically analyze the pipelines of the existing models and propose novel taxonomy to help the readers acquire knowledge in this new and growing research field. Besides, issues, challenges, and potential future directions of this task are also provided (Chapter 2 and Chapter 8).

- We build a compact and efficient baseline model and conduct intensive diagnostic experiments for monocular 3D object detection based on it. We quantify the overall impact of each sub-task and identify the localization error as the bottleneck of this task. Accordingly, we propose three novel strategies operating on annotations, training samples, and optimization losses to alleviate problems caused by localization errors for boosting detection (Chapter 3).
- We propose a pseudo-LiDAR-based model which first transforms 2D images to 3D point clouds and then detects the objects in the 3D space. This model significantly surpasses its concurrent counterparts in performance. Furthermore, we also conduct extensive analysis for this and build an image representation-based equivalent model, which shows the great potential of image-based representation. (Chapter 4 and Chapter 5).
- Due to the lack of spatial cues, accurately detecting objects in the 3D space from a single image is an extremely difficult task. To mitigate this issue, we propose a flexible and effective image-based 3D detection framework that can learn the spatial features from the LiDAR-based models based on knowledge distillation, thereby improving the detection accuracy significantly (Chapter 6).
- Labeling 3D data is an extremely expensive and time-consuming operation. The existing largest dataset [180] provides 230K labeled frames, still much smaller than those in other fields. To leverage the massive unlabeled data and further push the models with more training samples, we generate pseudo-labels and achieve semi-supervised learning for image-based 3D detection. We show the current state-of-the-art (SOTA) models can be largely improved with more training samples, even though their labels are noisy (Chapter 7).

1.5 Thesis Structure

The outline of this thesis is as follows. First, Chapter 2 presents an overview of this research field. Specifically, it gives the definition of this research problem, introduces the commonly used datasets and evaluation protocols, discusses the mainstream methods and critical milestones, and proposes taxonomies for this task. Second, in Chapter 3, we introduce how to build a simple and strong baseline model for this task and then analyze the source of errors for this task with our baseline model. Chapter 4 introduces how to build an accurate image-based 3D detector with the 'pseudo-LiDAR' representation and thoroughly analyzes why this representation works. Then, Chapter 5 proposes an efficient framework to learn the spatial features for the image-based 3D perception system, based on cross-modality knowledge distillation. Chapter 6 explores the semi-supervised setting with the pseudo-label mechanism to leverage the massive unlabeled data in this field. Finally, we discuss the potential problems of this task, compare the image-based algorithms with the LiDAR-based counterparts, and suggest directions for future research in Chapter 7.

Chapter 2

Datasets, Metrics, and Literature Review

In this Chapter, we aim to help readers, especially beginners, build a systemic understanding of the field of image-based 3D object detection. Particularly, we first introduce the commonly used datasets and the evaluation metrics. Then, we summarize the existing models at the framework level and propose a new taxonomy to group them into three categories. Finally, a detailed comparison of these algorithms is given, discussing each necessary component for 3D detection, such as feature extraction, loss formulation, post-processing, *etc.* The intention is to equip readers with the necessary knowledge and insights to navigate this field effectively.

2.1 Datasets

It is a well-known fact that the availability of large-scale datasets is essential for the success of data-driven deep learning techniques. For image-based 3D object detection in autonomous driving scenario, the main characteristics of the publicly available datasets [52, 15, 177, 21, 82, 138, 143, 180, 48, 54, 105] are summarized in Table 2.1. Among these datasets, the KITTI 3D [52], nuScenes [15], and Waymo Open [180] are the most commonly used and greatly promote the development of 3D detection. In the following, we provide the main information about these benchmarks, in terms of dataset size, diversity, and additional data.

Basic information. For most of the past decade, KITTI 3D was the only dataset to support the development of image-based 3D detectors. KITTI 3D provides front-view images with a resolution of 1280×384 pixels. In 2019 the nuScenes

Dataset	Year	Size				Diversity			Additional data			Benchmark
		# Train	# Val	# Test	# Boxes	# Scenes	# Classes*	Night/Rain	Stereo	Temporal	LiDAR	
KITTI 3D [52]	2017	7,418×1	-	7,518×1	200K	-	3	No/No	Yes	Yes	Yes	Yes
ApolloCar3D [177]	2019	4,036×1	200×1	1,041×1	60K	-	1	Yes/No	Yes	Yes	Yes	Yes
Argoverse [21]	2019	39,384×7	15,062×7	12,507×7	993K	113	15	Yes/Yes	Yes	Yes	Yes	Yes
Lyft L5 [82]	2019	22,690×6	-	27,468×6	1.3M	366	9	No/No	No	Yes	Yes	No
H3D [138]	2019	8,873×3	5,170×3	13,678×3	1.1M	160	8	No/No	No	Yes	Yes	No
A*3D† [143]	2019	39,179×1	-	-	230K	-	7	Yes/Yes	Yes	-	Yes	No
nuScenes [15]	2019	28,130×6	6,019×6	6,008×6	1.4M	1,000	10	Yes/Yes	No	Yes	Yes	Yes
Waymo Open [180]	2019	122,200×5	30,407×5	40,077×5	12M	1,150	3	Yes/Yes	No	Yes	Yes	Yes
CityScapes 3D [48]	2020	2,975×1	500×1	1,525×1	40K	-	6	Yes/Yes	Yes	No	No	Yes
A2D2 [54]	2020	12,497×1	-	-	-	-	14	No/Yes	No	Yes	Yes	No
KITTI-360 [105]	2021	-	-	-	68K	-	2	No/No	Yes	Yes	Yes	Yes

Table 2.1: The summary of datasets that can be used for image-based 3D object detection in autonomous driving scenarios. Some datasets are proposed for multiple tasks, and here we report the numbers for the 3D detection benchmark (*e.g.* KITTI 3D released more than 40K images, and about 15K of them are used for 3D detection). †: not released.

and Waymo Open datasets were introduced. In the nuScenes dataset, six cameras are used to generate 360° view with a resolution of 1600×900 pixels. Similarly, Waymo Open also captures 360° view using five synchronized cameras, and the resolution of the image is 1920×1280 pixels.

Dataset size. The KITTI 3D dataset provides 7,481 images for training and 7,518 images for testing, and it is common practice to split the training data into a training set and a validation set [27, 197, 173]. As the most commonly used one, 3DOP's split [27] includes 3,712 and 3,769 images for training and validation, respectively. The large-scale nuScenes and Waymo Open provide about 40K and 200K annotated frames and use multiple cameras to capture the panoramic view of each frame. In particular, nuScenes provides 28,130 frames, 6,019 frames, and 6,008 frames for training, validation, and testing (six images per frame). Waymo Open, the largest one, gives 122,200 frames for training, 30,407 frames for validation, and 40,077 frames for testing (five images per frame). It is worth mentioning that both these two datasets collect about 1.4M frames raw data, while Waymo Open annotates them at a $5\times$ higher frequency than nuScenes. Besides, all three datasets only release the annotations for the training/validation set, and the evaluation on the test set can only be conducted on their official testing servers.

Note that most papers only use the KITTI 3D dataset (with a focus on the *Car* category) for evaluation, except for the works in [170, 222, 49, 155, 189, 224, 136, 101, 190, 194] reporting performances on nuScenes or Waymo Open. Nevertheless, in future works, evaluating on these large-scale datasets is essential for assessing the effectiveness of the algorithms.

Diversity. The KITTI 3D dataset is captured in Karlsruhe, Germany in the daylight and good weather conditions. It mainly evaluates objects from three categories (Car, Pedestrian, and Cyclist), and divides them into three difficulty levels according to the height of 2D bounding boxes, occlusion, and truncation. The nuScenes dataset consists of 1000 scenes of 20s captured in Boston and Singapore. Differently from the KITTI 3D benchmark, these scenes have been captured at different times of the day (including night) and in different weather conditions (*e.g.* rainy day). There are ten categories of objects for the 3D detection task, and nuScenes also annotates the attribute labels for each category, *e.g.* moving or parked for a car, with or without a rider for a bicycle. These attributes can be regarded as fine-grained class labels, and the accuracy of attribute recognition is also considered in the nuScenes benchmark. The Waymo Open dataset covers 1,150 scenes, shot in Phoenix, Mountain View, and San

Francisco under multiple weather conditions, including night and rainy day. Similar to KITTI 3D, Waymo Open also defines two difficulty levels for the 3D detection task according to the number of LiDAR points contained in each 3D bounding box. The objects of interest in its benchmark include vehicles, pedestrians, and cyclists.

Additional data. In addition to the RGB images and the corresponding camera parameters, these datasets also provide additional data that can be optionally used in the image-based 3D detection task. Specifically, all three datasets provide the LiDAR signals and temporally preceding frames (note that these preceding images may be *unlabelled* because these datasets only annotate the key-frames from the collected videos), and the KITTI 3D dataset also provides the stereo pairs to support 3D object detection from stereo images.

2.2 Evaluation Metrics

Same as 2D object detection, the Average Precision (AP) [1, 43] constitutes the main evaluation metric used in 3D object detection. Starting from its vanilla definition, each dataset has applied specific modifications which gave rise to dataset-specific evaluation metrics. Here, we first review the original AP metric and then introduce its variants adopted in the most commonly used benchmarks, including the KITTI 3D, nuScenes, and Waymo Open.

2.2.1 Review of the AP Metric

To compute AP, the predictions are first assigned to their corresponding ground truths according to a specific measure. The most commonly used one, *i.e.* the Intersection over Union (IoU) between the ground truth A and the estimated 3D bounding box B, is defined as:

$$\mathrm{IoU}(A,B) = \frac{|A \cap B|}{|A \cup B|}. \tag{2.1}$$

The IoU measure is used to judge a matched prediction as a True Positive (TP) or a False Positive (FP) by comparing it with a certain threshold. Then, the recall r and precision p can be computed from the ranked (by confidence) detection results according to:

$$r = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}, \quad p = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \tag{2.2}$$

where the FN denotes the False Negative. The precision can be regarded as a function of recall, *i.e.* $p(r)$. Furthermore, to reduce the impact of "wiggles" in the precision-recall curve [43, 163], the interpolated precision values are used to compute the AP using:

$$\mathrm{AP} = \frac{1}{|\mathbb{R}|} \sum_{r \in \mathbb{R}} p_{interp}(r), \tag{2.3}$$

where $\mathbb{R}$ is the predefined set of recall positions and $p_{interp}(r)$ is the interpolation function defined as :

$$p_{interp}(r) = \max_{r' : r' \geq r} p(r'), \tag{2.4}$$

which means that instead of averaging over the actually observed precision values at recall r, the maximum precision at recall value greater than or equal to r is taken.

2.2.2 Dataset Specific Metrics

KITTI 3D Benchmark. KITTI 3D adopts the AP as the main metric and introduces some modifications. The first one is that the computation of the IoU is done in 3D space. Besides, KITTI 3D adopted the suggestion of Simonelli *et al.* [171] and replaced $\mathbb{R}_{11} = \{0, 1/10, 2/10, 3/10, ..., 1\}$ in Equation 2.3 with $\mathbb{R}_{40} = \{1/40, 2/40, 3/40, ..., 1\}$, which is a more dense sampling with the removal of recall position at 0.

Furthermore, due to the height of the objects is not so important as other items in the autonomous driving scenarios, Bird's Eye View (BEV) detection, also known as 3D localization task in some works [29, 146, 121], can be seen as an alternative to 3D detection. The calculation process of the metric, BEV AP, for this task is the same as the 3D AP, but the IoU is calculated on the ground plane, instead of the 3D space. This task is also included in some other benchmarks, such as Waymo Open [180].

Besides, KITTI 3D also proposed a new metric, Average Orientation Similarity (AOS), to evaluate the accuracy of orientation estimation. AOS is formulated as:

$$\mathrm{AOS} = \frac{1}{|\mathbb{R}|} \sum_{r \in \mathbb{R}} \max_{r' : r' \geq r} s(r'). \tag{2.5}$$

The orientation similarity $s(r) \in [0, 1]$ in Equation 2.5 for recall r is a normalized variant of the cosine similarity defined as:

$$s(r) = \frac{1}{|D(r)|} \sum_{i \in D(r)} \frac{1 + \cos \Delta_{\theta}^{(i)}}{2} \delta_i, \tag{2.6}$$

where $D(r)$ denotes the set of all object detection results at recall rate r and $\Delta_{\theta}^{(i)}$ is the difference in angle between the estimated and ground-truth orientations of detection i. To penalize multiple detections for a single object, KITTI 3D enforces $\delta_i = 1$ if detection i has been assigned to a ground-truth bounding box and $\delta_i = 0$ if it has not been assigned. Note that all the AP metrics are computed independently for each difficulty level and category.

Waymo Open Benchmark. Waymo Open also adopted the AP metric with a minor modification: replacing $\mathbb{R}_{11}$ in Equation 2.3 with $\mathbb{R}_{21} = \{0, 1/20, 2/20, ..., 1\}$. Moreover, considering that accurate heading prediction is critical for autonomous driving and the AP metric does not have a notion of heading, Waymo Open further proposes Average Precision weighted by Heading (APH) as its primary metric. Specifically, APH incorporates heading information into the precision calculation. Each true positive is weighted by the heading accuracy defined as $\min(|\theta - \theta^*|, 2\pi - |\theta - \theta^*|)/\pi$, where θ and θ^* are the predicted heading angle and the ground-truth heading angle in radians within $[-\pi, \pi]$. Note that APH jointly assesses the performance of both *3D detection* and *orientation estimation*, while AOS is only designed for *orientation estimation*.

nuScenes Benchmark. nuScenes proposed a new AP-based metric. In particular, it uses the 2D center distance on the ground plane to match the predictions and ground truths with a certain distance threshold d (*e.g.* 2m), instead of the IoU introduced in Equation 2.1. Besides, nuScenes calculate AP as the normalized area under the precision-recall curve for recall and precision over 10%. Finally, it calculates the mean Average Precision (mAP) over matching thresholds of $\mathbb{D} = \{0.5, 1, 2, 4\}$ meters and the set of classes $\mathbb{C}$:

$$\text{mAP} = \frac{1}{|\mathbb{C}||\mathbb{D}|} \sum_{c \in \mathbb{C}} \sum_{d \in \mathbb{D}} \text{AP}_{c,d}. \tag{2.7}$$

However, this metric only considers the localization of the objects, ignoring the effects of other aspects such as dimension and orientation. To compensate for it, nuScenes also proposed a set of True Positive metrics (TP metrics) designed to measure each predicted error separately using all true positives (determined under the center distance $d = 2m$ during matching). All the five TP metrics are designed to be positive scalars, which are defined as follows [15]:

- *Average Translation Error (ATE)* is the Euclidean distance for object center on the 2D ground plane (units in meters).
- *Average Scale Error (ASE)* is the 3D IoU error $(1 - \text{IoU})$ after aligning orientation and translation.
- *Average Orientation Error (AOE)* is the smallest yaw angle difference between the predictions and ground truths (in radians).
- *Average Velocity Error (AVE)* is the absolute velocity error as the L2 norm of the velocity differences in 2D (in m/s).
- *Average Attribute Error (AAE)* is defined as 1 minus attribute classification accuracy $(1 - acc)$.

Furthermore, for each TP metric, nuScenes also computes the mean TP metric (mTP) over all object categories:

$$\text{mTP}_k = \frac{1}{|\mathbb{C}|} \sum_{c \in \mathbb{C}} \text{TP}_{k,c}, \tag{2.8}$$

where $\text{TP}_{k,c}$ denotes the k^{th} TP metric (*e.g.* k = 1 means the ATE) for category c. Finally, to integrate all the mentioned metrics to a scalar score, nuScenes further proposes the nuScenes Detection Score (NDS) that combines the mAP defined in Equation 2.7 and the mTP_k defined in Equation 2.8:

$$\text{NDS} = \frac{1}{10}[5 \cdot \text{mAP} + \sum_{k=1}^{5}(1 - \min(1, \text{mTP}_k))]. \tag{2.9}$$

2.3 Literature Review in Framework Level

In this section, we summarize the image-based 3D detection methods in terms of the high-level paradigm. Specifically, we first introduce a new taxonomy for this task, and then discuss the existing methods accordingly.

2.3.1 Taxonomy

As shown in Figure 2.1, we propose to group the existing image-based 3D detectors into two categories: (i) *the methods based on 2D features*, and (ii) *the methods based on 3D features*. We believe this taxonomy can help beginners quickly establish a preliminary understanding of the methods in this field. Furthermore,

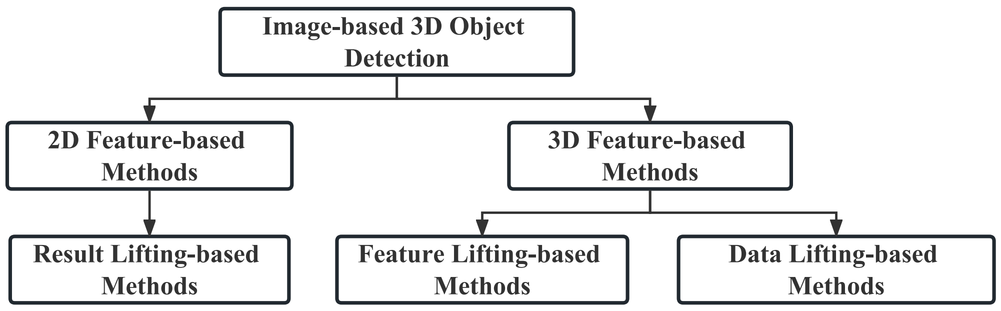


Image-based 3D Object Detection
2D Feature-based Methods
3D Feature-based Methods
Result Lifting-based Methods
Feature Lifting-based Methods
Data Lifting-based Methods

Figure 2.1: The proposed taxonomy for image-based 3D detection has two levels. Specifically, the methods are first divided into '2D feature-based methods' and '3D feature-based methods'. Then they are further grouped into 'result lifting-based methods', 'feature lifting-based methods', and 'data lifting-based methods'.

our taxonomy can be further divided into (i) *the methods based on result lifting*, (ii) *the methods based on feature lifting*, and (iii) *the methods based on data lifting*, which indicates the core problem of image-based 3D object detection: how to generate 3D results from 2D data. Particularly, the result lifting-based methods (*i.e.* the 2D feature-based methods) first estimate the 2D locations (and other items such as orientation, depth, etc.) of the objects in the image plane from the 2D features, and then lift the 2D detections into the 3D space. The feature lifting-based methods generate the 3D features by lifting the 2D features and then predict the final results in the 3D space. Similarly, the data lifting-based methods can also generate the 3D results directly, but they lift the input data from 2D to 3D, instead of the features. Figure 2.2 compares the data flows of these detection pipelines. According to the aforementioned taxonomy, we highlight the milestone methods (with the key benchmarks) in Figure 2.3.

Because there is no specific taxonomy for image-based 3D detection, previous works generally adopt the classic 2D detection taxonomy to divide the 3D object detectors into region-based methods and single-shot methods. We argue the proposed taxonomy is more suitable for image-based 3D detection because: (i) Our taxonomy groups the existing methods based on the feature representations, the foundation of the deep learning-based methods, thus it can help the readers build a structured knowledge quickly. (ii) Our taxonomy indicates how a detector aligns the dimension mismatch between the 2D input data and the 3D results (*i.e.* results lifting, feature lifting, or data lifting), which is the core problem of this task. (iii) Our taxonomy can clearly define the existing methods, while the previous ones can not. For example, the pseudo-LiDAR-based methods (will be introduced in Section 2.3.3) can adopt any LiDAR-based detectors,

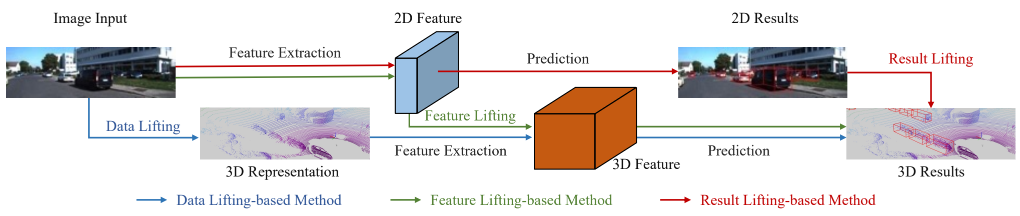


Figure 2.2: Illustration of the image-based 3D object detection pipelines. We show the data flows of data-lifting, feature-lifting, and result-lifting methods with blue, green, and red arrows respectively.

including region-based methods and single-shot methods. Therefore, it is hard to assign these methods to either side.

2.3.2 Methods Based on 2D Features

The first group is the 'methods based on 2D features'. Given an input image, they first estimate the 2D locations, orientations, and dimensions (see Figure 1.1 for the visualization of these items) from the 2D features, and then recover the 3D locations from these intermediate results. Therefore, these methods can also be called 'result lifting-based methods'. In particular, to get an object's 3D location $[x, y, z]$, an intuitive and commonly used solution is to estimate the depth value d using CNNs, and then lift the 2D projection into the 3D space using:

$$\begin{cases} z = d, \\ x = (u - C_x) \times z/f, \\ y = (v - C_y) \times z/f, \end{cases} \tag{2.10}$$

where (C_x, C_y) is the principal point, f is the focal length, and (u, v) is object's 2D location. Also note, these methods only need the depths of the center of the objects, which are different from the methods which require dense depth maps, *e.g.* Pseudo-LiDAR [192]. Furthermore, because the 2D feature-based methods are similar to 2D detectors in the overall framework, we introduce these works with classic taxonomy used in 2D detection field, *i.e.* region-based methods and single-shot methods, for better presentation.

Region-based Methods. The region-based methods follow the high-level idea of the R-CNN series [56, 55, 158] in 2D object detection. In this framework, after generating category-independent region proposals from input images, features are extracted from these regions by CNNs [158, 63]. Finally, R-CNN uses

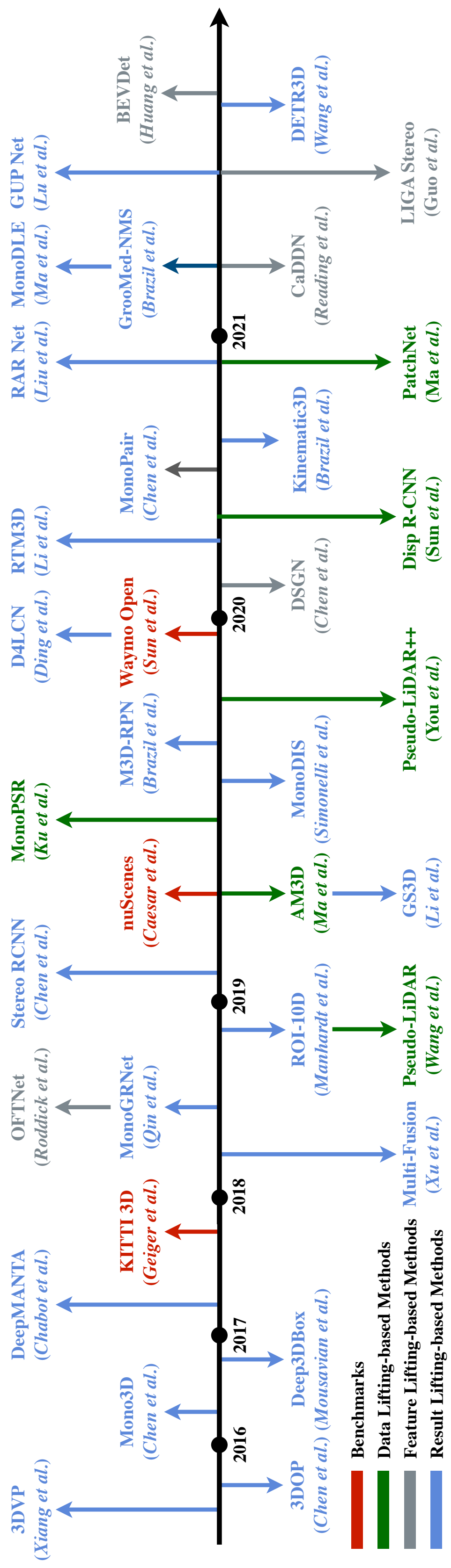


Figure 2.3: Chronological overview of the most relevant image-based 3D detection methods and the profound benchmarks.

these features to further refine the proposals and determine their category labels. Here we summarize the novel designs of the region-based framework for image-based 3D detection.

Proposal generation. Different from the commonly used proposal generation methods [185, 228] in the 2D detection field, a simple method to generate proposals for 3D detection is to tile the 3D anchors (*shape templates of the proposals*) in the ground plane and then project them to the image plane as proposals. However, this design generally leads to a huge computational overhead. To reduce the searching space, Chen *et al.* [26, 27, 28] proposed the pioneering Mono3D and 3DOP by removing the proposals with low confidence using domain-specific priors (*e.g. shape, height, location distribution, etc*) for monocular and stereo based methods respectively. Besides, Qin *et al.* [153] proposed another scheme which estimates an objectness confidence map in the 2D front-view, and only the potential anchors with high objectness confidence are considered in the subsequent steps. In summary, 3DOP [27] and Mono3D [26] compute confidence of proposals using geometric priors, while Qin *et al.* [153] uses a network to predict the confidence map.

With the Region Proposal Network (RPN) [158], the detectors can generate 2D proposals using features from the last shared convolutional layer instead of external algorithms, which saves most of the computational cost, and lots of image-based 3D detectors [131, 4, 199, 2, 171, 170, 97, 3, 168, 142, 153, 179, 204, 145, 53] adopted this design.

Introducing spatial information. Chen *et al.* [97] extended the design of RPN and R-CNN combination to the stereo-based 3D detection. They proposed to extract the features from the left image and right image separately and used the fused feature to generate proposals and predict the final results. This design allows the CNN to implicitly learn disparity/depth cues from stereo pairs and is adopted by the following stereo-based 3D detectors [142, 126, 204]. Also for the same purpose of providing depth information, Xu and Chen [199] proposed another scheme, Multi-Fusion, for monocular 3D detection. In particular, they first generate depth maps for input images using an off-the-shelf depth estimator [57, 47], and then design a region-based detector with multiple information fusion strategies for the RGB images and depth maps. It is worth noting that the strategy of providing depth cues with extra depth estimator for monocular images is embraced by several works [124, 192, 121, 195, 41, 187, 16, 192, 119, 36, 44]. Nevertheless, Stereo R-CNN [97] and Multi-Fusion [199] are similar in the high-level paradigm based on the fact that they both adopt the region-based framework and introduce another image (or map) to provide the spatial cues.

Single-Shot Methods. The single-shot object detectors directly predict class probabilities and regress other items of the 3D boxes from each feature position. As a consequence, these methods generally have faster inference speed than region-based methods, which is vital in the context of autonomous driving. The use of only CNN layers in single-shot methods also facilitates their deployment on different hardware architectures. Besides, some relevant works [222, 189, 107, 183] showed that single-shot detectors can also achieve promising performance. Based on the above reasons, lots of recent methods adopted this framework.

Basic single-shot models. Currently, there are two single-shot prototypes used in image-based 3D detection. The first one is anchor-based, proposed by Brazil and Liu [13]. In particular, this detector is essentially a tailored RPN for monocular 3D detection, and it generates both 2D anchors and 3D anchors for the given images. Different from the category-independent 2D anchors, the shape of 3D anchors generally have a strong correlation to their semantic label, *e.g.* an anchor with a shape of '$1.5m \times 1.6m \times 3.5m$' is usually a car rather than a pedestrian. Therefore, this 3D RPN can be used as the single-shot 3D detector and has been adopted by several methods [41, 187, 14, 88].

Besides, in 2019, Zhou *et al.* [222] proposed an anchor-free single-shot detector named CenterNet, and extended it to image-based 3D detection. In particular, this framework encodes the object as a single point (the center point of the object) and uses key-point estimation to find it. Besides, several parallel heads are used to estimate the other properties of the object, including depth, dimension, location, and orientation. Although this detector seems very simple in architecture, it achieves promising performance across several tasks and datasets. Later on, many following works [189, 224, 31, 122, 113, 99, 115, 116, 217, 114, 174] adopted this design.

2.3.3 Methods Based on 3D Features

Another branch of the proposed taxonomy is the 'methods based on 3D features'. The main feature of these methods is they first generate the 3D features from the images, and then directly estimate all items of the 3D bounding boxes, including the 3D locations, in the 3D space. According to how to get the 3D features, we further group these methods into 'feature lifting-based methods' and 'data lifting-based methods'.

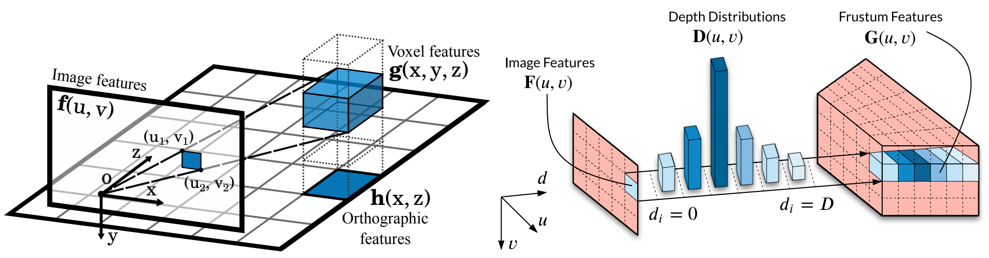


Figure 2.4: An illustration of the feature lifting methods. *Left:* 3D features are generated by accumulating image features over corresponding areas. *Right:* image features are weighted by their depth distribution to lift the 2D features into the 3D space. From [162] and [155].

Feature Lifting-based Methods. The general idea of the feature lifting-based methods is to transform the 2D image features in the image coordinate system into the 3D voxel features in the world coordinate system. Moreover, existing feature lifting-based methods [155, 162, 30, 61, 72] further collapse the 3D voxel features along the vertical dimension, corresponding to the height of objects, to generate the BEV features before estimating final results. For this kind of methods, the key problem is how to transform the 2D image features into the 3D voxel features. We discuss this problem in the following.

Feature lifting for monocular methods. Roddick *et al.* [162] proposed a retrieval-based detection model, named OFTNet, to achieve the feature lifting. They obtain the voxel feature by accumulating the 2D features over the area of the front-view image feature corresponding to the projection of each voxel's top-left corner (u_1, v_2) and bottom-right corner (u_2, v_2):

$$\mathbf{V}(x, y, z) = \frac{1}{(u_2 - u_1)(v_2 - v_1)} \sum_{u=u_1}^{u_2} \sum_{v=v_1}^{v_2} \mathbf{F}(u, v), \tag{2.11}$$

where $\mathbf{V}(x, y, z)$ and $\mathbf{F}(u, v)$ denote the features for the given voxel (x, y, z) and pixel (u, v). Differently, Reading *et al.* [155] achieve feature lifting in a back-projection manner [144]. Firstly, they discretize the continuous depth space to multiple bins and regard the depth estimation as a classification task. In this way, the output of depth estimation is the distribution $\mathbf{D}$ for these bins, instead of a single value. Then, each feature pixel $\mathbf{F}(u, v)$ is weighted by its associated depth bin probabilities in $\mathbf{D}(u, v)$ to generate the 3D frustum feature $\mathbf{G}(u, v)$:

$$\mathbf{G}(u, v) = \mathbf{D}(u, v) \otimes \mathbf{F}(u, v), \tag{2.12}$$

where $\otimes$ denotes the outer product. Note this frustum feature is based on the image-depth coordinate system (u, v, d), this need to be aligned to the 3D world coordinate system (x, y, z) using camera parameters to generate the voxel feature or BEV feature. Recently, Huang *et al.* [72] adopt this lfiting method and build BEV pipeline, which achieves promising performance on the multi-camera setting of image-based 3D detection. Figure 2.4 visualizes these two methods.

Feature lifting for stereo methods. Thanks to the well-developed stereo-matching technologies, building 3D features from stereo pairs is easier to achieve than building them from monocular images. Chen *et al.* [30] proposed the Deep Stereo Geometry Network (DSGN), achieving the feature lifting with stereo images as input. They first extract feature from the stereo pairs and then build 4D plane-sweep volume following the classic plane sweeping approach [37, 45, 208] by concatenating the left image feature and the reprojected right image feature at equally spaced depth values. Then, this 4D volume will be transformed into the 3D world space before generating the BEV map which used to predict the final results.

Data Lifting-based Methods. In the data lifting-based methods, the 2D images are transformed into the 3D data (*e.g.* the point cloud). Then the 3D features are extracted from the resulting data. In this section, we first introduce the pseudo-LiDAR pipeline, which lifts the images to point clouds, and the improvements designed for it. Then we introduce the image representation-based methods and other lifting schemes.

The pseudo-LiDAR pipeline. Thanks to the well-studied depth estimation, disparity estimation, and LiDAR-based 3D object detection, a new pipelin [192, 121, 195] was proposed to build a bridge between the image-based methods and LiDAR-based methods. In this pipeline, we first need to estimate the *dense* depth maps [57, 47] (or disparity maps [126, 19] and then transform them into the depth maps [192]) from images. Then, the 3D location (x, y, z) of the pixel (u, v) can be derived using Equation 2.10. By back-projecting all the pixels into 3D coordinates, the *pseudo-LiDAR* signals $\{(x^{(n)}, y^{(n)}, z^{(n)})\}_{n=1}^{N}$ can be generated, where N is the number of pixels. After that, LiDAR-based detection methods [146, 85, 206, 165] can be applied using the pseudo-LiDAR signals as input. The comparison of data representations used in this pipeline is shown in Figure 2.5. The success of the pseudo-LiDAR pipeline shows the importance of spatial features in this task and breaks the barrier between image-based methods and LiDAR-based methods, which makes it possible to apply the advanced technologies of another field.

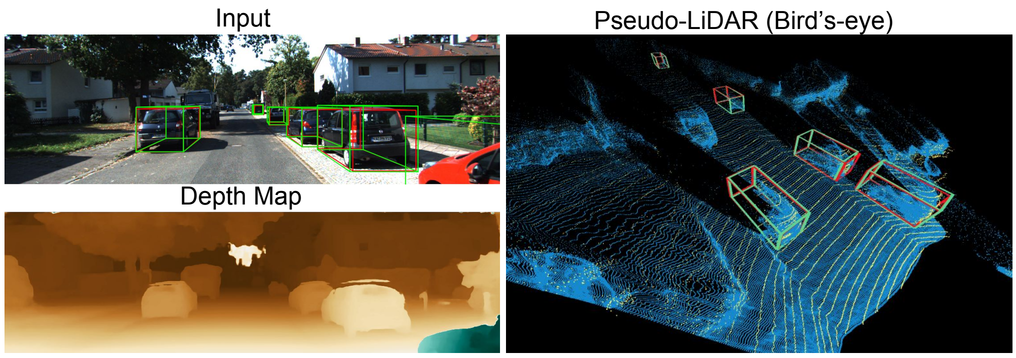


Figure 2.5: Comparison of different data representations: RGB image (*top left*), depth map (*bottom left*), and pseudo-LiDAR (*right*). From [192].

Improving the quality of depth maps (or resulting pseudo-LiDAR signals). Theoretically, the performances of pseudo-LiDAR-based models heavily rely on the quality of depth maps and some works [192, 121, 50] had confirmed this by adopting different depth estimators. Except for the improvement of depth estimation [57, 47, 93] and stereo matching [179, 19, 50], there are some other methods that improve the quality of depth maps. Note that a small error in disparity will lead to a large error in depth for the far-away objects, which is the primary weakness of the pseudo-LiDAR-based methods. To this end, You *et al.* [210] propose to transform the disparity cost volume to depth cost volume and learn depth directly end-to-end instead of through disparity transforms. Peng *et al.* [142] use a non-uniform disparity quantization strategy to ensure a uniform depth distribution, which can also reduce the disparity-depth transformation errors for the far-away objects. Besides, directly improving the accuracy of pseudo-LiDAR signal is another option. For this, You *et al.* [210] propose to use the cheap sparse LiDAR (*e.g.* 4-beam LiDAR) to correct the systematic bias in depth estimator. These designs can significantly boost the accuracy of generated pseudo-LiDAR signals, especially for far-away objects.

Focusing on the foreground objects. The original pseudo-LiDAR model estimates the full disparity/depth maps for the input images. This choice introduces lots of unnecessary computational cost and may distract the networks from the foreground objects, because only the pixels corresponding to the foreground objects are the focus in the subsequent steps. Based on this observation, several methods proposed their improvements. Specifically, similar to the LiDAR-based 3D detector F-PointNet [146], Ma *et al.* [121] use the 2D bounding box to remove the background points. Besides, they also propose a scheme based on dynamic threshold to further remove the noise points. Compared with the 2D bounding box, the methods in [195, 204, 145, 179] adopt instance mask, which is a better filter but requires additional data with ground-truth masks.

Besides, Wang *et al.* [191] and Li *et al.* [96] propose to address this problem in the depth estimation phase. They divide the pixels of the input images into foreground and background using 2D bounding boxes as masks, and apply higher training weight for the foreground pixels. Consequently, the depth values of foreground regions are more accurate than the baseline, thereby improving the 3D detection performance. Note that the confidences of the pixels belong to foreground/background can be used as additional features to augment the pseudo-LiDAR points [96].

Aggregation with other information. As described before, most pseudo-LiDAR-based methods only adopt the resulting pseudo-LiDAR signals as input. Another improvement direction is to enrich the input data with other information. Ma *et al.* [121] fuse the RGB features of each pixel to its corresponding 3D point using an attention-based module. Besides, a RoI-level RGB feature is also used to provide the complementary information to pseudo-LiDAR signals. Pon *et al.* [204] propose to use the pixel-wise part location map to augment the geometric cues for the pseudo-LiDAR signals (similar idea to the LiDAR-based 3D detector [166]). In particular, they use a CNN branch to predict the relative position of each pixel/point of the 3D bounding box, and then use this relative position to enrich the pseudo-LiDAR signals.

End-to-end training. Generally, the pseudo-LiDAR-based methods are clearly divided into two separate parts: depth estimation and 3D detection, and cannot be trained end-to-end. For this problem, Qian *et al.* [150] propose a differentiable Change of Representation (CoR) module that allows the back-propagation of gradients from 3D detection network to depth estimation network, and the whole system can benefit from joint training.

Image representation-based methods. To explore the underlying reasons for the success of pseudo-LiDAR-based methods, Ma *et al.* [119] proposed PatchNet, an image representation-based equivalent implementation of the original pseudo-LiDAR model [192], and achieved the almost same performance. Based on this, Ma *et al.* argue that the data lifting in Equation 2.10, which lifts the 2D location in the image coordinate to the 3D location in the world coordinate, is the key of the success of pseudo-LiDAR family, instead of the data representation. Simonelli *et al.* [173] extend PatchNet by re-scoring the confidence of 3D bounding boxes with a confidence head, and achieve better performance. Note that most of the designs in Section 2.3.3 for pseudo-LiDAR-based methods can be easily used in image representation-based method. Besides, benefited from the well-studied 2D CNN designs, the image-based data lifting model may have greater potentials [119].

Other lifting schemes. Different from previously introduced models which achieve the data lifting by depth estimation and Equation 2.10, Srivastava *et al.* [178] introduce another way for data lifting. Specifically, they transform the front-view images into the BEV maps using Generative Adversarial Nets (GAN) [59, 154], where the generator network aims to generate the BEV maps correspond to the given images and the discriminator network serves to classify the BEV maps are generated or not. Besides, Kim *et al.* [83] propose to use inverse perspective mapping to transform the front-view images into BEV images. After obtaining the BEV images, these two works can use the BEV-based 3D detectors, like MV3D [29] or BirdNet [8] to estimate the final results.

2.4 Literature Review in Modular Level

In this section, we provide a detailed review of 3D object detectors. Compared with the framework-level designs, the following designs are usually modular and can be applied to different algorithms flexibly.

2.4.1 Feature Extraction

Same as other tasks in the CV community, a good feature representation is a key factor in building high-performance image-based 3D detectors. The majority of recent methods use standard CNNs as their feature extractors, while some methods deviated from this introducing better features extraction methods. We will briefly cover them here.

Standard Backbone Nets. Although generally the input data is only the RGB image, the feature lifting-based methods and data lifting-based methods facilitate the use of 2D CNNs [64, 71, 198, 38, 211], 3D CNNs [74, 205], and pointwise CNNs [147, 148, 73] as the backbone networks. Among the standard backbones, DLA [211] and ResNet [64] are generally used to extract 2D features, and the sparse 3D conv [205] is the most popular backbone for 3D feature extraction. Note that the 2D backbones can also be used in the methods based on 3D features. For example, the feature-lifting-based methods, *e.g.* DSGN [30] and CaDDN [155], first use ResNet to extract 2D features from images, and then use 3D convolutions to generate more discriminative features after lifting the 2D features into 3D features.

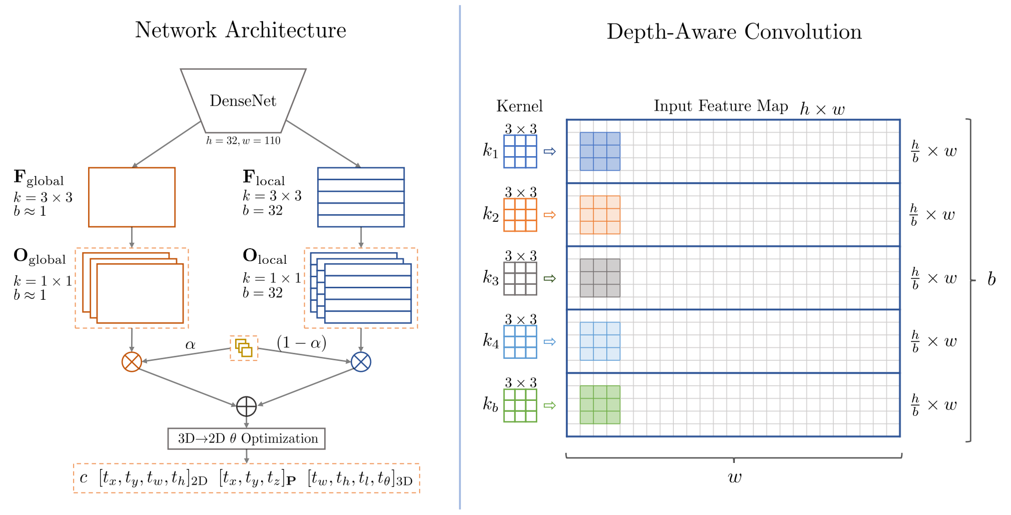

Network Architecture
DenseNet
$h = 32, w = 110$
$\mathbf{F}_{\text{global}}$ $k = 3 \times 3$ $b \approx 1$
$\mathbf{F}_{\text{local}}$ $k = 3 \times 3$ $b = 32$
$\mathbf{O}_{\text{global}}$ $k = 1 \times 1$ $b \approx 1$
$\mathbf{O}_{\text{local}}$ $k = 1 \times 1$ $b = 32$
α
$(1-\alpha)$
3D→2D θ Optimization
c $[t_x, t_y, t_w, t_h]_{2D}$ $[t_x, t_y, t_z]_{\mathbf{P}}$ $[t_w, t_h, t_l, t_\theta]_{3D}$
Depth-Aware Convolution
Kernel
Input Feature Map $h \times w$
3×3
k_1 k_2 k_3 k_4 k_b
$\frac{h}{b} \times w$
b
w

Figure 2.6: *Left:* M3D-RPN uses two parallel branches to extract the global features and local features respectively. *Right:* The proposed depth-aware convolution uses non-shared kernels to extract spatial-aware features for different rows in feature space. From [13].

Local Convolution. As shown in Figure 2.6, Brazil and Liu [13] propose to use two parallel branches to extract the spatial-invariant features and spatial-aware features respectively. In particular, to better capture the spatial-aware cues from monocular images, they further propose a local convolution: the depth-aware convolution. The proposed operation uses non-shared convolution kernels to extract the features for the different rows (roughly corresponding to different depths) in the feature space. Finally, the spatial-aware features are combined with the spatial-invariant ones before estimating the final results. Note that the non-shared kernels will introduce extra computational costs, and [13] also propose an efficient implementation for this scheme.

Feature Attention Mechanism. Since Hu *et al.* [70] introduced the attention mechanism [**attention**] to CNN, lots of attention blocks [70, 196, 7] are proposed. Although the details of these methods are different, they usually share the same key idea: re-weighting the features along a specific dimension, *e.g.* channel dimension.

Qin *et al.* [153] propose an attention scheme for 3D detection from stereo pairs. In particular, they calculate the correlation score s_i for the i^{th} channel of the left-image feature $\mathbf{F}_i^l$ and the right-image feature $\mathbf{F}_i^r$ using the cosine similarity:

$$s_i = \cos < \mathbf{F}_i^l, \mathbf{F}_i^r >= \frac{\mathbf{F}_i^l \cdot \mathbf{F}_i^r}{||\mathbf{F}_i^l|| \cdot ||\mathbf{F}_i^r||}. \tag{2.13}$$

Then, the features are scaled by the scaling factor s_i. Unlike other attention modules that learn the scaling factors in a data driven manner, this scheme updates the features using the correlation between left-image and right-image features, thus more interpretable. Besides, this design has been adopted by other stereo-based 3D detection methods, such as IDA-3D [142].

Depth Augmented Feature Learning. To provide the depth cues unavailable in the RGB images, an intuitive scheme is using the depth maps (generally obtained from an off-the-shelf model or a sub-network) to augment the RGB features [199, 124]. Besides, some efficient depth-augmented feature learning methods are proposed for this purpose. In particular, Ding *et al.* [41] propose a local convolutional network, where they use the depth maps as guidance to learn the dynamic local convolutional filters with different dilated rates for RGB images. Wang *et al.* [187] design a message-passing module between RGB features and depth features based on the graph neural network (GNN). Specifically, they regard the feature vector at each position and its most-relevant neighborhoods as the nodes of GNN. After dynamically sampling the nodes from image and depth features, they use GNN to propagate the depth cues to RGB features. Finally, they apply this module in multiple feature levels and obtain richer features for 3D detection.

Feature Mimicking. Recently, some methods propose to learn the features for image-based models under the guidance of LiDAR-based models. Particularly, Ye *et al.* [44] adopt the pseudo-LiDAR (data lifting) pipeline and enforce the features learned from pseudo-LiDAR signals should be similar to those learned from real LiDAR signals. Similarly, Guo *et al.* [61] apply this mechanism to the feature lifting-based method and conduct the feature mimicking in the transformed voxel features (or BEV features). Furthermore, Chong *et al.* [35] generalize this scheme to the result-lifting methods. They all transfer the learned knowledge from the LiDAR-based models to image-based models in the latent feature space, and the success of these works shows that image-based methods can benefit from feature mimicking.

Feature Alignment. As introduced in Chapter 1, in general, only the yaw angle is considered in the 3D detection task. However, this design will cause a misalignment problem when the roll/pitch angle is not zero, Figure 2.7 illustrates

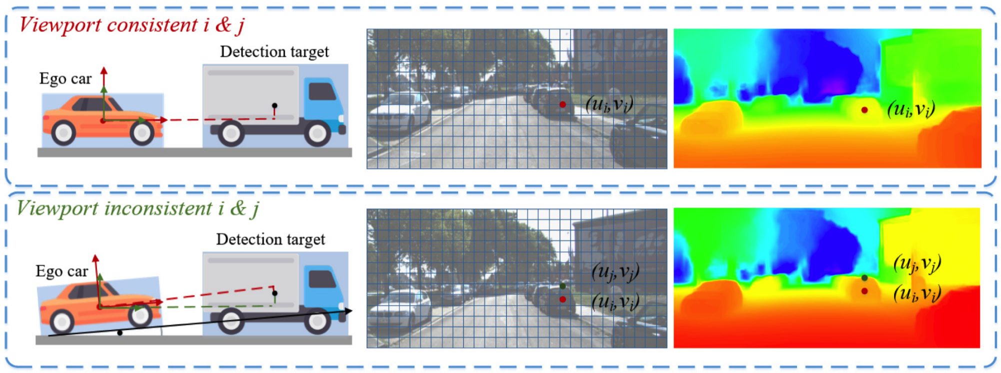


Figure 2.7: 3D detection assumes the ground plane is flat and only the yaw angle is considered (*top*). However, in real-world applications, there are some uneven roads, which leads to a bias between the object's actual position and computed position without considering the roll angle and pitch angle (*bottom*). From [224]

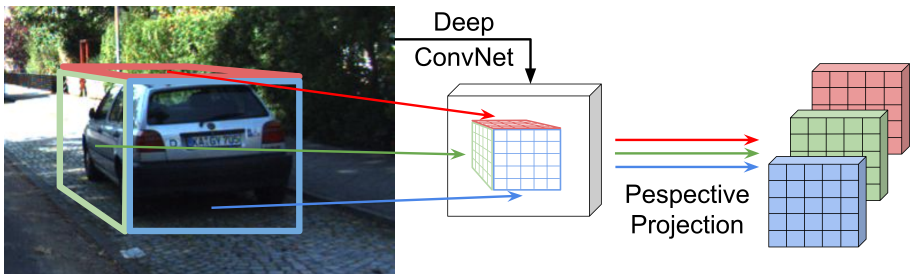


Figure 2.8: GS3D extracts features from the visible surfaces of the projected 3D bounding boxes and uses them to predict the final results. From [95].

this problem. For this problem, Zhou *et al.* [224] propose a feature alignment scheme. In particular, they first estimate the ego-pose using a sub-network, and then design a feature transfer net to align the features in both content level and style level based on the estimated camera pose. Finally, they use the rectified features to estimate the 3D bounding boxes.

Feature Pooling. Li *et al.* [95] propose a new feature pooling scheme for image-based 3D detection. As shown in Figure 2.8, for a given 3D anchor, they extract the features from the visible surfaces and warp them into a regular shape (*e.g.* 7×7 feature map) by perspective transformation. Then, these feature maps are combined and used to refine the proposals to the final results. Note that these features can be further augmented by concatenating the features extracted from 2D anchors using RoI Pool [158] or RoI Align [63].

2.4.2 Result Prediction

After the CNN features are obtained, the 3D detection results are predicted from the extracted features. In this section, we group the novel designs for the result prediction into different aspects and discuss these methods in detail.

Multi-Scale Prediction. A baseline model is to predict the results using the features of the last CNN layer [199, 192, 119, 173]. However, a major challenge of this scheme comes from the varied scales of the objects. Particularly, CNNs commonly extract the features layer by layer, which leads to different receptive fields and semantic levels for features at different layers. Consequently, it is hard to predict all the objects using the features from a specific layer. To address this issue, lots of methods have been proposed, broadly grouped into the layer-level methods and kernel-level methods.

Layer-level methods. The first group of methods mainly operates on the layer-level of CNNs and can be subdivided into the following three sub-groups.

(i) Multi-level prediction-based models. Liu *et al.* [110] and Cai *et al.* [17] propose to use a multi-layer prediction mechanism to address this problem, where each layer focuses on a specific range of scales. Figure 2.9 (*left*) shows the main idea of this design.

(ii) Feature fusion-based models. Another popular solution is to aggregate the features from different layers and predict all samples using this augmented feature map. Figure 2.9 (*middle*) visualizes a typical method [211] of this family. Note that lots of image-based 3D detectors [162, 86, 121, 13, 152, 31, 41, 113, 122, 215, 115, 114, 217, 116, 99] adopted this method for its simple and efficient design.

(iii) Hybrid models. In fact, recent approaches rarely use only one strategy, and the hybrid models are more welcomed. For example, FPN [106], shown in Figure 2.9 (*right*), combines the multi-level prediction and feature fusion scheme, and the FPN-like schemes are embraced by several 3D detection models [171, 170, 142, 124, 97, 189, 136].

Kernel-level methods. Some methods try to solve this problem by adjusting the receptive field in kernel-level. The dilated convolution [23] (also called 'atrous' convolution) is a pioneering work which is initially proposed to extract multi-scale features for semantic segmentation task. It introduces another parameter, the dilation rate, which controls the sampling interval in the convolution operation. This design can enlarge the receptive field without introducing extra computational cost. Ding *et al.* [41] introduce this convolution to monocular

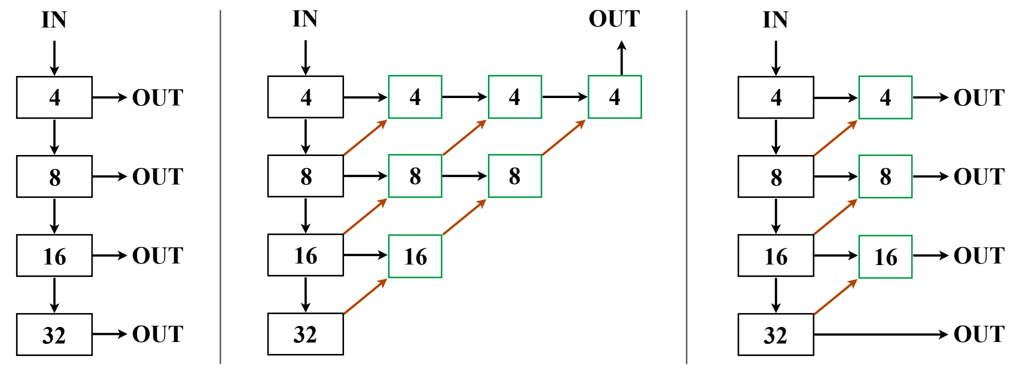


Figure 2.9: Illustration of the multi-level prediction (*left*), feature-fusion (*middle*), and hybrid (*right*) designs for multi-scale object detection. The green rectangle and red arrow respectively denote the feature aggregation node and upsampling operation.

3D detection and propose a scheme to dynamically adjust the dilated rate for each object according to its depth value. Besides, Dai *et al.* [38, 226] propose the deformable convolution, which allows the convolution kernels to learn their sampling positions in a data driven manner. Luo *et al.* [116] propose a variant of deformable convolution, which generates the sampling positions according to the shape of the anchors. Note that the dilated convolution can be regarded as a static special case of the deformable convolution. Besides, the kernel-level designs are orthogonal to the layer-level designs, which means they can collaboratively work in the same algorithm.

Multi-Camera Prediction. To cover all objects in 360° view, the multi-camera detection is adopted by recent large-scale benchmarks [15, 180]. A simple baseline [222] is to treat all views as separate images and predict results from them separately. Then the global Non-Maximum Suppression (NMS) is applied to merge the results from different views and remove the duplicate objects. Recently, a BEV solution [194, 72] is proposed for the multi-camera setting. In particular, this pipeline first lifts the features of different views from the image space to the BEV space and then integrates these BEV maps into a single feature map for the whole scene. After that, the results for all views can be predicted from this integrated BEV feature map. In addition to solving the task of multi-camera detection, this scheme generally generates more discriminative features due to the self-calibration of the features among different views, and similar strategies may be applied to other scenarios in future research, *e.g.* 3D detection from temporal sequences or multi-modality data.

Out-of-Distribution Samples. Due to the range, truncation, occlusion, etc., different objects tend to have different characteristics, and predicting all objects from a unified network may not be optimal. Based on this problem, [119, 173, 217] adopted the self-ensembling strategy. In particular, Ma *et al.* [119] divide the objects into three clusters by their depth values (or the 'difficulty' levels defined by KITTI 3D dataset), and use different heads to predict them in parallel. Simonelli *et al.* [173] extended this design by adding a re-scoring module for each head. Zhang *et al.* [217] decouple the objects into two cases according to their truncation levels and apply different label assignment strategies and loss functions to them.

Besides, Ma *et al.* [122] observe that some far-away objects are almost impossible to localize accurately and reducing their training weights (or directly removing these samples from the training set) can improve the overall performance. The underlying mechanism of this strategy has the same goal as [119, 173, 217], *i.e.* to avoid the distraction from out-of-distribution samples to the model training.

Projective Modeling for Depth Estimation. Compared with a stand-alone depth estimation task, depth estimation in 3D detection has more geometric priors, and the projective modeling is the most commonly used one. In particular, the geometric relationship between the height of 3D bounding box H_{3D} and the height of its 2D projection H_{2D} can be formulated as:

$$d = f \times \frac{H_{3D}}{H_{2D}} \tag{2.14}$$

where d and f respectively denote the depth of the object and the focal length of the camera. The height of the 2D bounding box is used to approximate H_{2D} in [16, 86, 167, 3], so they can compute a rough depth using estimated parameters. However, when the height of the 2D bounding box (denoted as H_{bbox2D}) is used as the H_{2D} in Equation 2.14, extra noise is introduced, because $H_{2D} \neq H_{bbox2D}$. To alleviate this problem, Lu *et al.* [115] propose an uncertainty-based scheme, which models the geometric uncertainty in the projective modeling. Besides, Barabanau *et al.* [4] annotate the key-points of cars with the help of CAD models, and use the height difference of 2D/3D key-points to get the depth. Differently, Zhang *et al.* [215] revise Equation 2.14 by considering the interaction of the locations, dimension and orientations of the objects, and build the relationship between the 3D bounding box and its 2D projection.

In brief, GUPNet [115] captures the uncertainty in the noisy perspective projection modeling, Barabanau *et al.* [4] eliminate the noise by re-labeling, and Zhang *et al.* [215] solve the error by mathematical modeling.

Multi-Task Prediction. *3D detection as multi-task learning.* 3D detection can be seen as a multi-task learning problem because it needs to output the class label, location, dimension, and orientation together. Lu *et al.* [115] propose to dynamically adjust the learning weights of each task for balanced learning. Different from other weight-based multi-task learning methods [32, 79], which assume that each task is independent from each other, there are some dependencies among the sub-tasks in 3D detection, *e.g.,* the height of 2D/3D bounding box can provide hints for depth estimation. Therefore, they build the hierarchical relationship of all tasks, and the training weight of each task is scheduled by its pre-tasks. Besides, Zou *et al.* [229] divide all tasks into the appearance-specific tasks and the localization tasks, and the features for these two groups are learned separately with a message passing module.

Joint training with other tasks. A number of works [200, 216, 212] had shown that the CNN can benefit from joint training with multiple tasks. Similarly, Ma *et al.* [122] observe the 2D detection can serve as an auxiliary task to monocular 3D detection and provide additional geometric cues to the neural network. Besides, Guo *et al.* [61] find this is also effective for stereo 3D detection. Note that the 2D detection is a required component in some methods [13, 171, 192, 121, 119], instead of an auxiliary task. Based on this, Liu *et al.* [111] find extra key-points estimation task can further enrich the CNN features, and the estimated key-points can be used to further optimize the depth estimation sub-task [131, 114, 99]. Besides, depth estimation can also provide valuable cues to the 3D detection model. In particular, some works [136, 30, 61, 35] conduct an extra depth estimation task to guide the shared CNN features to learn the spatial features, and Park *et al.* [137] show that pre-training on the large-scale depth estimation dataset can significantly boost the performance of their 3D detector.

2.4.3 Loss Formulation

Loss function is an indispensable part of the data driven models, and the loss formulation of 3D detection can be simplified to :

$$\mathcal{L} = \mathcal{L}_{\mathbf{cls}} + \mathcal{L}_{\mathbf{loc}} + \mathcal{L}_{\mathbf{dim}} + \mathcal{L}_{\mathbf{ori}} + \mathcal{L}_{\mathbf{joi}} + \mathcal{L}_{\mathbf{conf}} + \mathcal{L}_{\mathbf{aux}}. \tag{2.15}$$

Particularly, the classification loss $\mathcal{L}_{\mathbf{cls}}$ serves to identify the category of a candidate and give the confidence. The location loss $\mathcal{L}_{\mathbf{loc}}$, dimension loss $\mathcal{L}_{\mathbf{dim}}$, and orientation loss $\mathcal{L}_{\mathbf{ori}}$ are designed to regress the components of the parameterization of 3D bounding box, *i.e.* location, dimension, and orientation respectively. The last three loss items are optional. In particular, the loss $\mathcal{L}_{\mathbf{joi}}$, *e.g.* corner loss [124], can jointly optimize the location, dimension, and orientation in a single loss function. The confidence loss $\mathcal{L}_{\mathbf{con}}$ is designed to give better confidence to the detected boxes. Finally, the auxiliary loss $\mathcal{L}_{\mathbf{aux}}$ can introduce additional geometric cues to CNNs. These losses in Equation 2.15 are discussed below.

Classification Loss $\mathcal{L}_{\mathbf{cls}}$. For classification loss, the FocalLoss [107] or its variant [91] is used by most methods. Compared with the standard cross-entropy loss, this loss function reduces the penalty on the easy cases and focuses more on the hard, misclassified examples. In this way, this loss boosts the classification accuracy.

Location Loss $\mathcal{L}_{\mathbf{loc}}$. The feature lifting-based and data lifting-based methods generally regress the locations using L1 loss (or smooth L1 loss, L2 loss, etc. and we omit them in the following part for brevity):

$$\mathcal{L}_{\mathbf{loc}} = \sum_{i \in \{x,y,z\}} ||\mathbf{loc}_i - \mathbf{loc}_i^*||_1, \tag{2.16}$$

where $||\cdot||_1$ denotes the L1 norm. $\mathbf{loc}_i$ and $\mathbf{loc}_i^*$ are the estimated location and corresponding ground-truth location respectively. Generally, the models predict the relative offset to a specific anchor, instead of the absolute position. As for the result lifting-based methods, the 3D location is derived from the 2D location and depth (note most of the feature lifting-based and data lifting-based methods also need depth for their transformations), and the loss function can be formulated as:

$$\mathcal{L}_{\mathbf{loc}} = \mathcal{L}_{\mathbf{loc_{2d}}} + \mathcal{L}_{\mathbf{depth}}, \tag{2.17}$$

where $\mathcal{L}_{\mathbf{loc_{2d}}}$ is the 2D location loss and generally shares the similar formulation as Equation 2.16. The $\mathcal{L}_{\mathbf{depth}}$ is the depth loss. Since some works [122, 190, 137] point out that depth is the key in image-based 3D detection, we mainly review the novel loss formations of this item in image-based 3D detection approaches.

Uncertainty modeling. Following [78, 80], some works [31, 122, 9, 167, 115] model the heteroscedastic aleatoric uncertainty in the depth estimation subtask. Specifically, to capture the uncertaintiy, detectors should simultaneously predict the depth $\mathbf{d}$ and the standard deviation $\mathbf{œ}$ (or variance $\mathbf{œ}^2$):

$$[\mathbf{d}, \mathbf{œ}] = f^{\mathbf{w}}(\mathbf{x}), \tag{2.18}$$

where $\mathbf{x}$ is the input data, f is a convolutional neural network parametrised by the parameters $\mathbf{w}$. Then, the Laplace likelihood is fixed to model the uncertainty, and the loss for the depth estimation sub-task can be formulated by:

$$\mathcal{L}_{\mathbf{depth}} = \frac{\sqrt{2}}{\boldsymbol{\oe}}||\mathbf{d} - \mathbf{d}^*||_1 + \log \boldsymbol{\oe}, \tag{2.19}$$

where $||\cdot||_1$ denotes the L1 norm, and $\mathbf{d}^*$ is the ground-truth depth. Similarly for the Gaussian likelihood [122, 33]:

$$\mathcal{L}_{\mathbf{depth}} = \frac{1}{2\boldsymbol{\oe}^2}||\mathbf{d} - \mathbf{d}^*||_2 + \frac{1}{2}\log \boldsymbol{\oe}^2, \tag{2.20}$$

where $||\cdot||_2$ denotes the L2 norm (the derivation details of Equation 2.19 and Equation 2.20 can be found in [80], page 37). Note that this loss formulation can be applied to any regression task in theory [31, 33, 65]. Besides, the uncertainty has been further utilized in other aspects, such as confidence normalization [115] or post-processing [167].

Discretization. For monocular depth estimation, Fu *et al.* propose DORN [47], which discretizes the continuous depth values into multiple bins and considers the depth estimation as a *ordinal regression* task. This model is often used as a sub-network to provide depth cues for 3D detectors. Besides, the methods in [155, 190] also adopt the discretization strategy, while only regarding depth estimation as a *classification* task. Note that the discretization-based methods usually output the distribution of depth, instead of a single value, which can be used in feature lifting.

Dimension Loss $\mathcal{L}_{\mathbf{dim}}$. A common choice for the dimension loss in 3D detection is L1 loss:

$$\mathcal{L}_{\mathbf{dim}} = \sum_{i\in\{h,w,l\}} ||\mathbf{dim}_i - \mathbf{dim}_i^*||_1, \tag{2.21}$$

where $\mathbf{dim}_i$ denotes the predicted dimension, and $\mathbf{dim}_i^*$ is the corresponding ground truth. The incremental method in [13] computes the mean shape $[H, W, L]$ of each category first, and then estimates the residual offset of these anchors. Furthermore, Simonelli *et al* [171] represent the dimension as $[He^{\delta_h}, We^{\delta_w}, Le^{\delta_l}]$, where $[\delta_h, \delta_w, \delta_l]$ are the outputs of the CNN for the objects' dimension. In this way, they can embed the physical prior, *i.e.* the mean shape of the objects, in the prediction and optimize the CNN's parameters in the exponential space.

Ma *et al.* [122] show that the errors of different elements of the estimated dimension (*e.g.* height, weight, length) have different contribution rates to the

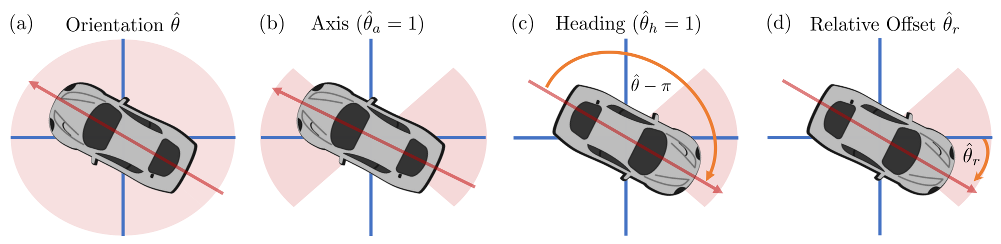

(a) Orientation $\hat{\theta}$
(b) Axis $(\hat{\theta}_a = 1)$
(c) Heading $(\hat{\theta}_h = 1)$
$\hat{\theta} - \pi$
(d) Relative Offset $\hat{\theta}_r$
$\hat{\theta}_r$

Figure 2.10: Illustration of the orientation formulation proposed by Brazil *et al.* They first classify the orientation (*a*) as closer to the horizontal axis or the vertical axis (*b*), and then judge whether it points in the positive or the negative direction (*c*). Finally, they regress an offset to the center of the angle bin (*d*). From [14].

change of IoU. Based on this observation, they dynamically adjust the weight of each term in this loss function, according to its partial derivative w.r.t. the 3D IoU. Besides, they also keep the absolute value of this loss function unchanged to the original L1 loss, which means that their proposed loss is the re-distribution of the standard L1 loss.

Orientation Loss $\mathcal{L}_{\mathbf{ori}}$. Compared with directly regressing the orientation of objects, the hybrid-style (classification and regression) loss formulation is the mainstream in orientation estimation, and the main difference of these losses lies in how to divide the continuous orientation into different bins [184]. In particular, Mousavian *et al.* [129] divide the heading angle into n overlapping bins (n=2 by default), and Qi *et al.* [146] choose a denser, non-overlapping quantization (n=12 in default). These two methods are widely used in the existing 3D detectors, *e.g.* [199, 122, 115, 31, 121, 119, 192, 210, 195, 86]. Besides, as shown in Figure 2.10, Brazil *et al.* [14] propose to divide the orientation into 4 bins and use two classifiers, *i.e.* axis classifier (horizontal or vertical) and heading classifier (positive or negative), to find the angle interval. As for the regression part, an alternative to regressing the residual angle $\Delta\theta$ directly is to regress it in the sine and cosine spaces, *i.e.* sin(■`) and cos(■`), which has been adopted by several works, such as [129, 222, 168]. Besides, Li *et al.* [100] show that orientation estimation can benefit from specific intermediate representation, *i.e.* interpolated cuboid.

Joint Loss $\mathcal{L}_{\mathbf{joi}}$. In this section, we introduce how to jointly optimize the location, dimension, and orientation in a single loss function.

Corner loss. To jointly optimize the location, dimension, and orientation, Manhardt *et al.* [124] recover the 3D coordinates of eight corners using the estimated items, and compute the corner loss:

$$\mathcal{L}_{\textbf{corner}} = \sum_{k=1}^{8} ||P_k - P_k^*||_1, \tag{2.22}$$

where P_k denotes the k^{th} corner of the 3D bounding box. Note that the corner loss can also be used as auxiliary loss and work together with the standard loss formulation [146, 121, 192, 119, 195, 44, 220].

Disentangled corner loss. To avoid the complicated interactions between each item, Simonelli *et al.* [171] propose a disentangling transformation, and this method has been adopted by some other works [170, 174, 113, 136, 173]. In particular, when computing the corners, they only use one estimated item and adopt ground truths for the remaining items (*e.g.* using the predicted dimension and ground-truth location/orientation to compute the corners). They replicate this process for three times to separately back propagate the losses of the location, dimension, and orientation. This design removes the interactions of different items but also keeps the optimization space the same as the corner loss. Also note that this transformation can apply to any metric involving multiple items, such as IoU.

Confidence Loss $\mathcal{L}_{\textbf{conf}}$. A simple baseline method for the confidence estimation is directly adopting the classification confidence as the final score. This strategy is popular in the 2D detection field and also commonly adopted by the image-based 3D detection models, such as [13, 31, 122]. Besides, some confidence estimation methods designed for 2D detectors are also introduced to image-based 3D detection task (*e.g.* following FCOS [183], FCOS3D [189] estimates the 'centerness' for each object and use it to normalize the confidence). However, these methods are generally better at representing the quality of the 2D bounding box, instead of that of the 3D bounding box. Here we represent these confidences as the 2D confidence (in fact the 2D confidence p_{2D} is often adopted as the 3D confidence p_{3D} directly in most works).

To better capture the quality of the estimated 3D bounding boxes, some novel designed are proposed. In particular, Simonelli *et al.* [171] propose to estimate the 3D confidence given a 2D proposal $p_{3D|2D}$, and its ground truth $p^*_{3D|2D}$ is generated by:

$$p^*_{3D|2D} = e^{-\frac{1}{T}\mathcal{L}(B,B^*)}, \tag{2.23}$$

where T is the temperature parameter, and $\mathcal{L}(B, B^*)$ is the disentangled corner loss of bounding box B and its ground-truth B^*. Then the cross-entropy loss is used to optimize the CNNs, and the final 3D confidence is:

$$p_{3D} = p_{2D} \cdot p_{3D|2D}. \tag{2.24}$$

Furthermore, Simonelli *et al.* [173] use the normalized ranking of $\mathcal{L}(B, B^*)$ as $p^*_{3D|2D}$, and report that this relative 3D confidence performs better than the absolute version. Besides, since the depth estimation is the bottleneck of the image-based 3D detection [122, 190, 121, 192, 199], Lu *et al.* [115] use the depth confidence p_{depth} to replace $p_{3D|2D}$. In particular, they capture the depth uncertainty σ using Equation 2.19 and use the normalized uncertainty $e^{-\sigma}$ as the depth confidence, and the final 3D confidence is computed as:

$$p_{3D} = p_{2D} \cdot p_{depth}. \tag{2.25}$$

Auxiliary Loss $\mathcal{L}_{\mathbf{aux}}$. *Dense depth loss.* Although the *dense* depth estimation is unnecessary in the design of most image-based 3D detectors, several works observe that it benefits the detectors in performance. In particular, some feature lifting-based methods [30, 155] find that applying the dense depth supervision is helpful to align the 2D space and the 3D space for the feature lifting models. Park *et al.* [136] find that dense depth estimation can serve as an effective pre-training task for monocular 3D detection. Besides, [35] reports that a separate prediction head supervised by the dense depth maps can effectively introduce the spatial cues to the CNNs, thereby improving the performance.

2D/3D consistency loss. Based on the geometric prior that the projection of the 3D bounding box should tightly fit the 2D bounding box, we can build an auxiliary loss by comparing the consistency of them. Weng and Kitani [195] apply this loss function in their 3D detector, and Brazil and Liu [13] use this loss in their post-processing method.

Others. As introduced in Section 2.4.2, some works report that joint training with some other tasks, such as 2D detection[122, 30] and key-points estimation[111], can boost the performance of 3D detection. From the perspective of loss function, the losses of these tasks can be regarded as the auxiliary losses of 3D detection.

2.4.4 Post-processing

After getting the results from CNNs, some post-processing steps are applied to remove redundant detection results or refine detection results. These steps can broadly be divided into two groups: Non-Maximum Suppression (NMS) and post-optimization.

NMS. *Traditional NMS.* Generally, the original detection results have multiple redundant bounding boxes covering a single object, and NMS is designed so that a single object is only covered by an estimated bounding box. The pseudo-code for the traditional NMS is shown in Algorithm 1. In particular, the bounding box b_m with the maximum score is selected and all other boxes having high overlap with b_m are removed from the detection results. This process is recursively applied on the remaining boxes to get the final results.

Algorithm 1: Traditional NMS

```
Data: $\mathcal{B} = \{b_1, \dots, b_n\}, \mathcal{S} = \{s_1, \dots, s_n\}, \Omega$
$\mathcal{B}$ is the list of initial bounding boxes, $\mathcal{S}$ contains corresponding scores, $\Omega$
is the NMS threshold
Result: $\mathcal{D}$, the set of final results with scores
$\mathcal{D} \leftarrow \varnothing$;
while $\mathcal{S} \neq \varnothing$ do
    $m \leftarrow \arg\max \mathcal{S}$;
    $\mathcal{B} \leftarrow \mathcal{B} - \{b_m\}$;
    $\mathcal{S} \leftarrow \mathcal{S} - \{s_m\}$;
    $\mathcal{D} \leftarrow \mathcal{D} \cup \{(b_m, s_m)\}$;
    for $b_j \in \mathcal{B}$ do
        if $\mathrm{IoU}(b_m, b_j) > \Omega$ then
            $\mathcal{B} \leftarrow \mathcal{B} - \{b_m\}$;
            $\mathcal{S} \leftarrow \mathcal{S} - \{s_m\}$;
        end
    end
end
return $\mathcal{D}$;
```

The variants of NMS. To avoid the removal of valid objects, Bodla *et al.* [11] just reduce the scores of high overlapped objects, instead of discarding them (Soft NMS). Jiang *et al.* [75] observe the mismatch between the classification score and the quality of box, and propose to regress a localization score, *i.e.* IoU score, to play the role of $\mathcal{S}$ in Algorithm 1 (IoU Guided NMS). Since the major issue of monocular based 3D detectors is the localization error [122, 162], where the depth estimation is the core problem to recover object location, Shi *et al.* [167] use Equation 2.19 to capture the uncertainty of estimated depth, and use the depth uncertainty σ_{depth} to normalize the score s to $\frac{s}{\sigma_{depth}}$ when applying the NMS (Depth Guided NMS). It is reported in [135, 133] that the boxes with non-maximum scores may also have high-quality localization and propose to update

b_m by weighted averaging of the boxes b_i with high overlap (Weighted NMS). In particular, they first compute weight by: $w_i = s_i \times \text{IoU}(b_m, b_i)$ and update the bounding box by: $b_i = \sum_i \frac{w_i}{sum(w)} \cdot b_i$, where $sum(w) = \sum_i w_i$. Similarly, He *et al.* [65] also adopt weighted averaging mechanism with an update of the averaging rule. Particularly, they model the uncertainty of each item of bounding box under Gaussian distribution (Equation 2.20) and then set the averaging rule only related to the IoU and uncertainty (Softer NMS). Liu *et al.* [109] propose to use a dynamic NMS threshold Ω for objects with different densities (Adaptive NMS).

Note that some algorithms mentioned above [11, 65, 133, 109, 75, 135] are initially proposed for 2D detection, but they can be easily applied to 3D detection. Besides, [133, 65] can also be regarded as post-optimization methods because they update the predicted results during the NMS process, except for eliminating duplicated detections.

Others. Kumar *et al.* [88] propose a differentiable NMS for monocular 3D detection. With this design, the loss function can directly operate on the results after NMS. Besides, for the multi-camera-based panoramic datasets, *e.g.* nuScenes and Waymo Open, the global NMS is needed to eliminate the duplicate detection results from the overlapping images.

Post-optimization. To boost the quality of detected boxes, some methods choose to further refine the outputs of CNNs by building geometric constraints in the post-optimization step.

Brazil and Liu [13] proposed a post-optimization method to tune the orientation θ based on the consistency between the projected 3D bounding box and 2D bounding box. In particular, they iteratively add a small offset to the predicted orientation θ and project the updated 3D boxes into the 2D image plane. Then, they choose to accept this update or adjust the offset by checking whether the similarity between the 2D bounding box and the projected 3D bounding box increases or decreases.

Another post-optimization method is built on the one-to-one matching of the key-points of objects in the 2D image plane and 3D world space. Specifically, in addition to the 3D boxes, Li *et al.* [99] also estimates the projected corners in the 2D image space. After that, they project the 3D boxes into the image plane and update the estimated parameters by minimizing the pixel distances of the paired pixels, *i.e.* 2D/3D corners, using Gauss-Newton [46] or Levenberg-Marquardt algorithm [128].

Chen *et al.* [31] propose an object-level pair-wise constraint for their post-optimization. In particular, they regard two adjacent objects as an object pair, and additionally estimate the midpoints of the paired objects in their CNN model. After that, they can fine-tune the locations by aligning the paired objects and their midpoint. Further, they also model the uncertainties of location-related items (depth and the center of the 2D projection in [31]) using Equation 2.19, and use the captured uncertainties as weights of the objectives in their post-optimization method. The post-optimization methods in [99, 31] can be efficiently implemented using the open-sourced toolbox g2o [89].

Chapter 3

A Baseline Model and Error Analysis

3.1 Introduction

Remarkable progress has been achieved in 3D detection, especially for LiDAR/stereo-based approaches [223, 90, 165, 30, 192], along with the advances in deep neural networks. In contrast, the accuracy of 3D detection from only monocular images [171, 13, 31, 121, 119, 41] is obviously lower than that from LiDAR or stereo data. In this work, we aim to quantitatively identify the problem and propose our solutions.

To investigate and quantify the underlying factors that restrict the performance of monocular 3D object detection, we conduct intensive diagnostic experiments for this task, inspired by the error identifying methods [91, 222, 68, 12] commonly used in the 2D detection scope. Specifically, we build our baseline model (see Section 3.2.2 for details) based on CenterNet [222] and progressively replace predicted items with their ground-truth values. To better analyze the error patterns, we evaluate the results in a range-wise manner and show the summary of those experiments in Figure 3.1. Based on our investigation, we have the following three observations and corresponding designs.

Observation 1: The most striking feature in Figure 3.1 is the leap in performance when using ground-truth location, reaching a level similar to the state-of-the-art LiDAR-based methods, suggesting that localization error is the key factor in restricting monocular 3D detection. Furthermore, except for depth estimation, detecting the projected center of the 3D object also plays an important role in restoring the 3D position of the object. To this end, we revisit the misalignment between the center of the 2D bounding box and the projected center

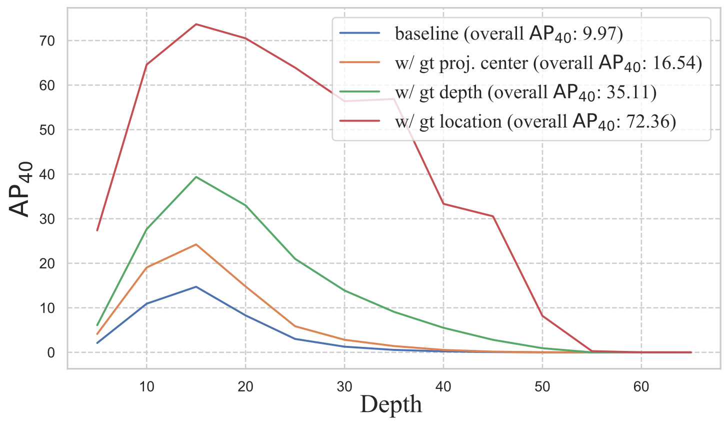


Figure 3.1: Range-wise evaluation on the KITTI-3D *validation* set. Metric is AP_{40} of the Car category under hard setting. The sampling interval is 10 m. For example, the corresponding value at horizontal axis 20 represents the overall performance of all samples between 15 m and 25 m.

of the 3D object. Besides, we also confirm the necessity of keeping 2D detection-related branches in monocular 3D detectors. In this way, 2D detection is used as the correlated auxiliary task to help learn the features shared with 3D detection, which is different from the existing work in [113] that discards 2D detection. *Observation 2*: An apparent trend reflected in Figure 3.1 is that the detection accuracy significantly decreases with respect to the distance (the low performance of very close-range objects will be discussed in supplementary materials). More importantly, all the models cannot output any true positive samples beyond a certain distance. We found that it is almost impossible to detect distant objects accurately with existing technologies due to the inevitable localization errors (see Section 3.3.4 for details). In this case, whether it is beneficial to add these samples to the training set becomes a question. In fact, there is a clear domain gap between 'bad' samples and 'easy-to-detect' samples, and forcing the network to learn from those samples will reduce its representative ability for the others, which will thus impair the overall performance. Based on the observation above, we propose two schemes. The first scheme removes distant samples from the training set and the second scheme reduces the training loss weights of these samples.

Observation 3: We found that, except for localization error, there are also some other vital factors, such as dimension estimation, restricting monocular 3D detection (there is still 27.4% room for improvements even if we use the ground-truth location). Existing methods in this scope tend to optimize each component of the 3D bounding box independently, and the studies in [171, 170] confirm the effectiveness of this strategy. However, the failure to consider the contribution of each loss item to the final metric (*i.e.* 3D IoU) may lead to sub-optimal optimization. To alleviate this problem, we propose an IoU-oriented loss for 3D size estimation. The new IoU-oriented loss dynamically adjusts the loss weight for each side in the sample level according to its contribution rate to the 3D IoU.

In summary, the key contributions of this Chapter are as follows: First, we conduct intensive diagnostic experiments for monocular 3D detection. In addition to finding that the 'localization error' is the main problem restricting monocular 3D detection, we also quantify the overall impact of each sub-task. Second, we investigate the underlying reasons behind localization error and analyze the issues it might bring. Accordingly, we propose three novel strategies operating on annotations, training samples, and optimization losses to alleviate problems caused by localization errors for boosting detection.

Experimental results show the effectiveness of the proposed strategies. In particular, compared with existing best-performing monocular 3D object detection approaches, the proposed method achieves at least 1.6 points AP_{40} improvements on the bird's view detection and 3D object detection in the KITTI-3D dataset.

3.2 Approach

3.2.1 Problem Definition

Given are RGB images and the corresponding camera parameters, our goal is to classify and localize the objects of interest in 3D space. Each object is represented by its category, 2D bounding box $\mathbf{B_{2D}}$, and 3D bounding box $\mathbf{B_{3D}}$. Specifically, $\mathbf{B_{2D}}$ is represented by its center $\mathbf{c^i} = [x', y']_{2D}$ and size $[h', w']_{2D}$ in the image plane, while $\mathbf{B_{3D}}$ is defined by its center $[x, y, z]_{3D}$, size $[h, w, l]_{3D}$ and heading angle γ in the 3D world space.

3.2.2 Baseline Model

Architecture. We build our baseline model based on the anchor-free one-stage detector CenterNet [222]. Specifically, we use standard DLA-34 [211] as our backbone for a better speed-accuracy trade-off. On top of this, seven lightweight heads (implemented by one 3×3 conv layer and one 1×1 conv layer) are used for 2D detection and 3D detection.

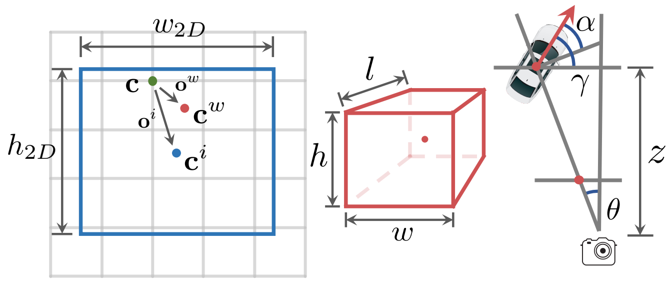


Figure 3.2: Visualization of the notations of 2D bounding box in the feature map scale (*left*), 3D bounding box in the 3D world space (*middle*), and orientation of the object from bird's view (*right*).

2D detection. For 2D detection task, following [156, 222], the proposed model outputs a heatmap to indicate the classification score and the coarse center $\mathbf{c} = (u, v)$ of the object. In existing methods [222, 31, 171], $\mathbf{c}$ is supervised by the ground-truth 2D bounding box center. Another branch predict the offset $\mathbf{o^i} = (\Delta u^i, \Delta v^i)$ between the coarse center and the real center of 2D bounding box, and we can get the final 2D box center location $\mathbf{c^i} = \mathbf{c} + \mathbf{o^i}$. Finally, we use another branch to estimate the size $[w', h']_{2D}$ of 2D bounding box.

3D detection. As for 3D detection, a branch is used for predicting the offset $\mathbf{o^w} = (\Delta u^w, \Delta v^w)$ between the coarse center $\mathbf{c}$ and the center of projected 3D bounding box $\mathbf{c^w} = [x^w\ y^w]^T = \mathbf{c} + \mathbf{o^w}$. With the known camera intrinsic matrix $\mathbf{K} \in \mathbb{R}^{3\times 3}$, we can recover the center of object in 3D world space by:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix}_{3D} = \mathbf{K}^{-1} \begin{bmatrix} \mathbf{c^w} \cdot z \\ z \end{bmatrix} = \mathbf{K}^{-1} \cdot \begin{bmatrix} x^w \cdot z \\ y^w \cdot z \\ z \end{bmatrix}_{2D}, \tag{3.1}$$

where z is the output of depth branch. Finally, the last two branches are used to predict the 3D size $[h, w, l]_{3D}$ and orientation γ, respectively.

Uncertainty modeling. Following [78, 31], we model the heteroscedastic aleatoric uncertainty in the depth estimation sub-task. Specifically, we simultaneously predict the depth $\mathbf{d}$ and the standard deviation $\boldsymbol{œ}$ (or variance $\boldsymbol{œ}^2$):

$$[\mathbf{d}, \boldsymbol{œ}] = f^{\mathbf{w}}(\mathbf{x}), \tag{3.2}$$

where $\mathbf{x}$ is the input data and f is a convolutional neural network parametrised by the parameters $\mathbf{w}$. Then, we fix a Laplace likelihood to model the uncertainty,

baseline	11.12	ground truth	99.97
w/ gt proj. center	23.90	w/o gt proj. center	46.33
w/ gt depth	38.01	w/o gt depth	25.25
w/ gt 3D location	78.84	w/o gt 3D location	12.13
w/ gt 3D size	11.96	w/o gt 3D size	80.50
w/ gt orientation	11.88	w/o gt orientation	70.89

Table 3.1: Error analysis. *Left:* We replace the outputs of 3D detection-related branches with the ground truth values. *Right:* We replace the values of ground truth with the predicted results. Metric is AP_{40} for 3D detection under moderate setting on the KITTI-3D val set. 'proj. center' denotes the projected 3D center $\mathbf{c^w}$ on the image plane.

and the loss for the depth estimation sub-task can be formulated by:

$$\mathcal{L} = \frac{\sqrt{2}}{\text{œ}}||\mathbf{d} - \mathbf{d}^*||_1 + \log \text{œ}, \tag{3.3}$$

where $||\cdot||_1$ denotes the L1 norm and $\mathbf{d}^*$ is the ground truth value for depth $\mathbf{d}$. Similarly for the Gaussian likelihood:

$$\mathcal{L} = \frac{1}{2\text{œ}^2}||\mathbf{d} - \mathbf{d}^*||_2 + \frac{1}{2}\log \text{œ}^2, \tag{3.4}$$

where $||\cdot||_2$ denotes the L2 norm (please refer to [77] for the derivation of Equation 3.3 and Equation 3.4). Note that the uncertainty modeling is not claimed as our contribution.

Losses. There are seven loss terms in total, one for foreground/background sample classification, two (center and size) for 2D detection, and four (center, depth, size, and heading angle) for 3D detection. We adopt the modified Focal Loss used in [91, 222] for the classification sub-task. We use L1 Loss without any anchor for center and size regression in the 2D detection task. For the 3D detection task, uncertainty modeling [78] is used for depth estimation; L1 loss is used for 3D center refinement; and multi-bin loss [146] (we consider 12 non-overlap equal bins) is used for heading angle estimation. Lastly, for 3D size estimation, we use L1 loss in baseline (without anchor), and the proposed IoU loss in our model. The weights for all loss items are set to 1.

3.2.3 Error Analysis

In this section, we explore what restricts the performance of monocular 3D detection. Inspired by CenterNet [222] and CornerNet [91] in the 2D detection field, we conduct an error analysis for different prediction items on KITTI-3D *validation* set via replacing each prediction with ground truth value and evaluating the performance. Specifically, we replace each output head with its ground truth according to the practice of [91, 222]. As shown in Table 3.1, if we replace projected 3D center $\mathbf{c^w}$ predicted from the baseline model with its ground truth, the accuracy is improved from 11.12% to 18.97%. On the other hand, depth can improve the accuracy to 38.01%. If we consider both depth and projected center, *i.e.* replacing the predicted 3D locations $[x, y, z]_{3D}$ with ground-truth results, then the most obvious improvement is observed. Therefore, the low accuracy of monocular 3D detection is mainly caused by localization errors. On the other hand, according to Equation 3.1, depth estimation and center localization jointly determine the position of the object in 3D world space. Compared with the ill-posed depth estimation from a monocular image, improving the accuracy of center detection is a more feasible way.

Table 3.2 shows localization errors introduced by inaccurate center detection. Furthermore, the mean shape of cars in KITTI-3D dataset is $[1.53, 1.63, 3.53]$ meters for $[h, w, l]_{3D}$. Suppose that all other quantities are correct and the localization error is aligned with the length l (resulting in the maximum tolerance), the IoU can be computed by:

$$IoU = \frac{3.53 - \Delta_{loc}}{3.53 + \Delta_{loc}}, \tag{3.5}$$

where Δ_{loc} represents the localization error. According to the official setting, the IoU threshold should be set to 0.7, thus the theoretically acceptable maximum error is $0.62m$. However, an error of only 4-8 pixels in the image (1-2 pixels in $4\times$ down-sampling feature map) will cause the object at 60 meters cannot be detected correctly. Coupled with the errors accumulated by other tasks such as depth estimation (Figure 3.3 shows the errors of depth estimation), it becomes an almost impossible task to accurately estimate the 3D bounding box of distant objects from a single monocular image, unless the depth estimation is accurate enough (not achieved to date).

To better show the importance of center localization, we show the localization error in 3D space caused by shifting the center in the image plane in Table 3.2.

3.2.4 Revisiting Center Detection

Our design for center detection. For estimating the coarse center $\mathbf{c}$, our design is simple. In particular, we 1) use the projected 3D center $\mathbf{c^w}$ as the ground

Δu	Δv	5m	10m	20m	40m	60m
2	2	0.02	0.04	0.08	0.16	0.24
4	2	0.03	0.06	0.13	0.25	0.38
6	2	0.04	0.09	0.18	0.36	0.54
6	4	0.05	0.10	0.20	0.41	0.61
8	2	0.06	0.12	0.23	0.47	0.70
8	6	0.07	0.14	0.28	0.57	0.85

Table 3.2: Localization error (in meter) caused by center shifting in the image plane (in pixel).

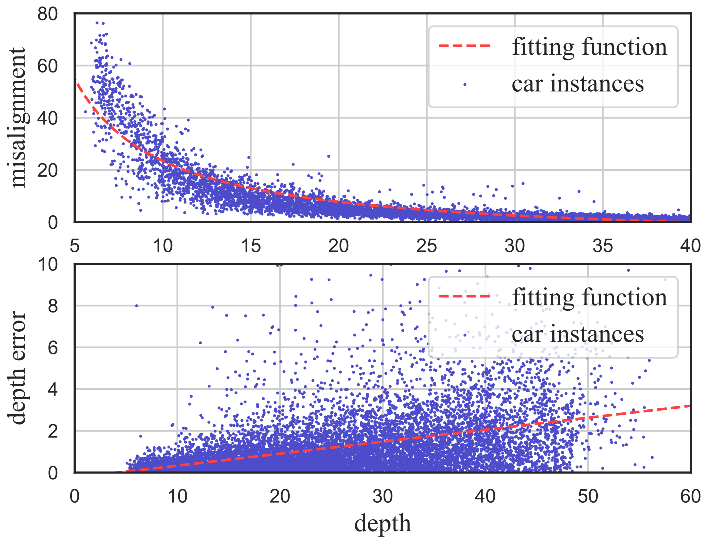


Figure 3.3: Statistics. *Top*: the misalignment (in pixel, collected on the KITTI-3D *trainval* set under moderate setting) between the center of 2D bounding box and the projected 3D center in the image plane. *Bottom*: the depth errors (in meter, trained on the KITTI-3D *training* set, tested on the *validation* set). These two statistics are presented as the function of the depth (x-axis).

truth for the branch estimating coarse center **c** and 2) force our model to learn features from 2D detection simultaneously. This simple design is from our analysis below.

Analysis 1. As shown in Figure 3.4, there is a misalignment between the 2D bounding box center $\mathbf{c^i}$ and the projected center $\mathbf{c^w}$ of the 3D bounding box. According to the formulation in Equation 3.1, the projected 3D center $\mathbf{c^w}$ should be the key for recovering the 3D object center $[x, y, z]_{3D}$. The key problem here is what should be the supervision for the coarse center **c**. Some works [31, 171] choose to use 2D box center $\mathbf{c^i}$ as its label, which is not related to the 3D object center, making the estimation of the coarse center not aware of the 3D geometry of the object. Here we choose to adopt the projected 3D center $\mathbf{c^w}$ as the ground

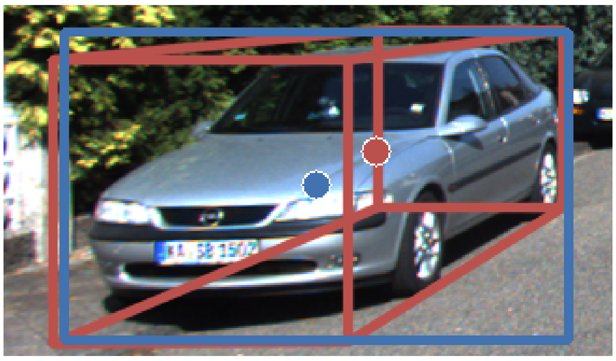


Figure 3.4: Visualization of the misalignment between the center of the 2D bounding box (blue) and the projected 3D center (red) in image plane.

truth for the coarse center $\mathbf{c}$. This helps the branch for estimating the coarse center aware of 3D geometry and more related to the task of estimating 3D object center, which is the key to localization problem.

Analysis 2. Note that SMOKE [113] also use the projected 3D center $\mathbf{c^w}$ as the label of the coarse center $\mathbf{c}$. However, they discard 2D detection-related branches while we preserve them. In our design, the coarse center $\mathbf{c}$ supervised by the projected 3D center $\mathbf{c^w}$ is also used for estimating the 2D bounding box center $\mathbf{c^i}$. With our design, we force a 2D detection branch to estimate an offset $\mathbf{o^i} = \mathbf{c^i} - \mathbf{c}$ between the real 2D center and the coarse 2D center. This makes our model *aware of the geometric information of the object*. Besides, another branch is used to estimate the size of the 2D bounding box so that *the shared features can learn some cues that benefit to depth estimation due to the perspective projection.* In this way, the 2D detection serves as an auxiliary task that helps to learn better 3D-aware features.

3.2.5 Training Samples

Different from [169, 107] which forces network focus on the 'hard' samples, we argue that ignoring some extremely 'hard' cases can improve the overall performance for the monocular 3D detection task. Both the results shown in Figure 3.1 and the analysis conducted in Section 3.3.4 illustrates there is a strong relationship between the distance of the object and the difficulty of detecting it. According to this, two schemes are proposed on how to generate the object-level training weight w_i for sample i.

Scheme 1, hard coding. This scheme discards all samples over a certain distance:

$$w_i = \begin{cases} 1 & \text{if } d_i \leq s \\ 0 & \text{if } d_i > s \end{cases} \tag{3.6}$$

where d_i denotes the depth of sample i, and s is the threshold of depth which is set to 60 meters in our implementation. In this way, the samples with a depth larger than s will not be used in the training phase.

Scheme 2, soft coding. The other one is soft encoding, and we generate it using a reverse sigmoid-like function:

$$w_i = \frac{1}{1 + e^{(d_i - c)/T}}, \tag{3.7}$$

where c and T are the hyper-parameters to adjust the center of symmetry and bending degree, respectively. When $c = s$ and $T \rightarrow 0$, it is equivalent to the hard encoding scheme. When $T \rightarrow \infty$, it is equivalent to using the same weight for all samples. By default, c and T are set to 60 and 1, and the empirical experiments in Section 3.3 find that scheme 1 and scheme 2 are both effective and have similar results.

3.2.6 IoU Oriented Optimization

Recently, some LiDAR based 3D detectors [202, 219] applied the IoU oriented optimization [161]. However, determining the 3D center of the object is a very challenging task for monocular 3D detection, and the localization error often reaches several meters (see Section 3.3.4). In this case, localization-related subtasks (such as depth estimation) will overwhelm others (such as 3D size estimation), if we apply IoU based loss function directly. Moreover, depth estimation from the monocular image itself is an ill-posed problem, and this kind of contradiction will make the training process collapse. Disentangling each loss item and optimizing them independently is another choice [171], but this ignores the correlation of each component to the final result. To alleviate this problem, we propose an IoU-oriented optimization for 3D size estimation. Specifically, suppose all prediction items except the 3D size $\mathbf{s} = [h, w, l]_{3D}$ are completely correct, then we can get:

$$\frac{\partial IoU}{\partial h} : \frac{\partial IoU}{\partial w} : \frac{\partial IoU}{\partial l} \approx \frac{1}{h} : \frac{1}{w} : \frac{1}{l}. \tag{3.8}$$

Accordingly, we can adjust the weight of each side by its partial derivative *w.r.t.* IoU (in magnitude), and the loss function of the 3D size estimation can be modified to:

$$\mathcal{L}_{size} = ||\frac{(\mathbf{s}-\mathbf{s}^*)}{\mathbf{s}}||_1, \tag{3.9}$$

where $||\cdot||_1$ represent the L_1 norm. Note that, compared with the standard 3D size loss $\mathcal{L}'_{size} = ||\mathbf{s}-\mathbf{s}^*||_1$ used in the baseline model, our new loss's magnitude is changed. To compensate for it, we compute $\mathcal{L}'_{size}$ once more, and dynamically generate the compensate weight $w_s = |\mathcal{L}'_{size}/\mathcal{L}_{size}|$, so that the mean value of the final loss function $w_s \cdot \mathcal{L}_{size}$ is equal to the standard one. In this way, the proposed loss can be regarded as a re-distribution of the standard L1 loss.

3.2.7 Proof of Proposition

This part provides the proof of the following proposition, which is used in Equations 3.8 and 3.9 for the IoU-oriented optimization.

Proposition. Suppose all predicted items except the 3D sizes (h, w, l) are completely correct, the contribution ratio of each predicted side to the 3D IoU $\frac{\partial IoU}{\partial h}$: $\frac{\partial IoU}{\partial w}$: $\frac{\partial IoU}{\partial l}$ can be approximated to $\frac{1}{h} : \frac{1}{w} : \frac{1}{l}$.

Proof. Given the above conditions, the 3D IoU metric can be formulated as:

$$IoU = \frac{\prod_{i\in\{h,w,l\}} \min(i, i^*)}{h \times w \times l + h^* \times w^* \times l^* - \prod_{i\in\{h,w,l\}} \min(i, i^*)}, \tag{3.10}$$

where (h^*, w^*, l^*) denotes the ground truth of 3D size (h, w, l). With the different relationship between the prediction and the ground truth of the 3D size, we can obtain the following cases:

Case 1: If $h \leq h^*$, $w \leq w^*$, and $l \leq l^*$, the Equation 3.10 can be simplified as:

$$IoU = \frac{h \times w \times l}{h^* \times w^* \times l^*}, \tag{3.11}$$

and we further compute the partial derivative of 3D IoU with respect to the variable h as

$$\frac{\partial IoU}{\partial h} = \frac{w \times l}{h^* \times w^* \times l^*}, \tag{3.12}$$

where $\frac{\partial IoU}{\partial h}$ represents the partial derivative of 3D *IoU* with respect to the variable h, analogically for $\frac{\partial IoU}{\partial w}$ and $\frac{\partial IoU}{\partial l}$. Then, combining the derivative of 3D IoU with respect to h, w, and l, the contribution ratio of each predicted side can be given as:

$$\frac{\partial IoU}{\partial h} : \frac{\partial IoU}{\partial w} : \frac{\partial IoU}{\partial l} = \frac{1}{h} : \frac{1}{w} : \frac{1}{l}. \tag{3.13}$$

Case 2: If $h > h^*$, $w > w^*$, and $l > l^*$, the Equation 3.10 can be simplified as:

$$IoU = \frac{h^* \times w^* \times l^*}{h \times w \times l}, \tag{3.14}$$

and similar to Equation 3.12 and 3.13, we can derive the same conclusion as *Case 1*.

Case 3: If $h > h^*$, $w \leq w^*$, and $l \leq l^*$, then we represent the 3D IoU as:

$$IoU = \frac{h^* \times w \times l}{h \times w \times l + h^* \times w^* \times l^* - h^* \times w \times l}. \tag{3.15}$$

By calculating the derivative of 3D IoU with respect to h, w, and l respectively, we can get the contribution ratio of each predicted side:

$$\frac{\partial IoU}{\partial h} : \frac{\partial IoU}{\partial w} : \frac{\partial IoU}{\partial l} = \frac{w \times l}{h^* \times w^* \times l^*} : \frac{1}{w} : \frac{1}{l}. \tag{3.16}$$

Case 4: If $h > h^*$, $w > w^*$, and $l \leq l^*$, similarly, we can get the IoU formulation as:

$$IoU = \frac{h^* \times w^* \times l}{h \times w \times l + h^* \times w^* \times l^* - h^* \times w^* \times l}. \tag{3.17}$$

Similar to previous steps, the formulation of each side's contribution rate to the 3D IoU is given as:

$$\frac{\partial IoU}{\partial h} : \frac{\partial IoU}{\partial w} : \frac{\partial IoU}{\partial l} = \frac{1}{h} : \frac{1}{w} : \frac{h^* \times w^* \times l^*}{h \times w \times l \times l}. \tag{3.18}$$

The other cases are similar to *Case 3* and *Case 4*. When $h \approx h^*$, $w \approx w^*$, and $l \approx l^*$, we can get the Equation 3.8 used in the main paper.

3.2.8 Implementation

Training. We train our model on two GTX 1080Ti GPUs with a batch size of 16 in an end-to-end manner for 140 epochs. We use Adam optimizer with an initial learning rate $1.25e^{-3}$ and decay it by ten times at 90 and 120 epochs. The weight decay is set to $1e^{-5}$ and the warmup strategy is also used for the first 5 epochs. To avoid over-fitting, we adopt random cropping/scaling (for 2D detection only) and random horizontal flipping. Under this setting, it takes around 9 hours for the whole training process.

Method	Extra data	3D			BEV			AOS			Runtime
		Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	
Decoupled-3D [16]	Yes	11.08	7.02	5.63	23.16	14.82	11.25	87.34	67.23	53.84	-
AM3D [121]	Yes	16.50	10.74	9.52	25.03	17.32	14.91	-	-	-	~400 ms
PatchNet [119]	Yes	15.68	11.12	10.17	22.97	16.86	14.97	-	-	-	~400 ms
D4LCN [41]	Yes	16.65	11.72	9.51	22.51	16.02	12.55	90.01	82.08	63.98	-
Kinematic3D [14]	Yes	19.07	12.72	9.17	26.69	17.52	13.10	58.33	45.50	34.81	120ms
GS3D [95]	No	4.47	2.90	2.47	8.41	6.08	4.94	85.79	75.63	61.85	~2000 ms
MonoGRNet [152]	No	9.61	5.74	4.25	18.19	11.17	8.73	-	-	-	60 ms
MonoDIS [171]	No	10.37	7.94	6.40	17.23	13.19	11.12	-	-	-	-
M3D-RPN [13]	No	14.76	9.71	7.42	21.02	13.67	10.23	88.38	82.81	67.08	161 ms
SMOKE [113]	No	14.03	9.76	7.84	20.83	14.49	12.75	92.94	87.02	77.12	**30 ms**
MonoPair [31]	No	13.04	9.99	8.65	19.28	14.83	12.89	91.65	86.11	76.45	57 ms
Ours	No	**17.23**	**12.26**	**10.29**	**24.79**	**18.89**	**16.00**	**93.46**	**90.23**	**80.11**	40 ms
Improvement	-	+2.47	+2.27	+1.64	+3.77	+4.06	+3.11	+0.52	+3.21	+2.99	-

Table 3.3: Performance of the Car category on the KITTI-3D *test* set. Methods are ranked by moderate setting (same as the KITTI-3D leaderboard). We highlight the best results in **bold** and the second place in underlined.

Method	3D@IOU=0.7			BEV@IOU=0.7			3D@IOU=0.5			BEV@IOU=0.5		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
CenterNet [222]	0.60	0.66	0.77	3.46	3.31	3.21	20.00	17.50	15.57	34.36	27.91	24.65
MonoGRNet [152]	11.90	7.56	5.76	19.72	12.81	10.15	47.59	32.28	25.50	48.53	35.94	28.59
MonoDIS [171]	11.06	7.60	6.37	18.45	12.58	10.66	-	-		-	-	
M3D-RPN [13]	14.53	11.07	8.65	20.85	15.62	11.88	48.53	35.94	28.59	53.35	39.60	31.76
MonoPair [31]	16.28	12.30	10.42	24.12	18.17	15.76	55.38	42.39	**37.99**	**61.06**	**47.63**	**41.92**
Ours	**17.45**	**13.66**	**11.68**	**24.97**	**19.33**	**17.01**	**55.41**	**43.42**	37.81	60.73	46.87	41.89
Improvement	+1.17	+1.36	+1.26	+0.85	+1.16	+1.25	+0.03	+1.03	-0.18	-0.33	-0.80	-0.03

Table 3.4: Performance of the Car category on the KITTI-3D *validation* set. Methods are ranked by moderate setting (same as KITTI-3D leaderboard). We highlight the best results in **bold** and the second place in underlined.

Inference. During the inference phase, we obtain the prediction results from the parallel decoders. To decode the results, similar to [**zhou2019objects**], we conduct the efficient non-maxima suppression (NMS) on center detection results using a 3×3 max pooling kernel. Then, we recover 2D/3D bounding boxes according to the encoding strategy introduced in Section 3.2.2 and use the score of center detection as the confidence of predicted results. Finally, we discard predictions with confidence less than 0.2.

3.3 Experimental Results

3.3.1 Setup

Dataset. We evaluate our method on the challenging KITTI-D dataset [52, 51], which provides 7,481 images for training and 7,518 images for testing. Since the ground truth for the test set is not available and the access to the test server is limited, we follow the protocol of prior works [26, 27, 29] to divide the training data into a training set (3,712 images) and a validation set (3,769 images). We conduct ablation studies based on this split and also report final results which are trained on all 7,481 images and tested by the KITTI-D official server.

Metrics. The KITTI-3D dataset provides many widely used benchmarks for autonomous driving scenarios, including 3D detection, bird's eye view (BEV) detection, and average orientation similarity (AOS). We report the Average Precision with 40 recall positions (AP_{40}) [171] under three difficultly settings (easy, moderate, and hard) for those tasks. We mainly focus on the **Car** category and also report the performances of the **Pedestrian** and **Cyclist** categories for reference. The default IoU thresholds are 0.7, 0.5, and 0.5 for these categories.

3.3.2 Main Results

Results on the KITTI-3D *test* set. As shown in Table 5.7, we report our results of the **Car** category on KITTI-3D *test* set. Overall, our method achieves superior results over previous methods across all settings under fair conditions. For instance, the proposed method obtains **2.47/2.27/1.64** improvements under easy/moderate/hard settings for 3D detection. Besides, our method achieves **18.89/90.23** in BEV detection/AOS task under moderate setting, improving previous best results by **4.06/4.12** AP_{40}. Compared with the methods with extra data, the proposed method still gets comparable performances, which further proves the effectiveness of our model.

Method	Cat.	Easy	Mod.	Hard
M3D-RPN [13]	Ped.	5.65 / 4.92	4.05 / 3.48	3.29 / 2.94
MonoPair [31]	Ped.	10.99 / 10.02	7.04 / 6.68	6.29 / 5.53
Ours	Ped.	10.73 / 9.64	6.96 / 6.55	6.20 / 5.44
M3D-RPN [13]	Cyc.	1.25 / 0.94	0.81 / 0.65	0.78 / 0.47
MonoPair [31]	Cyc.	4.76 / 3.79	2.87 / 2.12	2.42 / 1.83
Ours	Cyc.	5.34 / 4.59	3.28 / 2.66	2.83 / 2.45

Table 3.5: Benchmark for Pedestrian/Cyclist detection on the KITTI-3D *test* set. Metric is AP_{40} for BEV/3D detection task at 0.5 IoU threshold.

	Easy	Mod.	Hard
baseline	20.29 / 14.51	16.15 / 11.12	14.07 / 9.97
+ p.	23.10 / 15.78	18.15 / 12.65	16.11 / 10.62
+ p.+I.	23.89 / 16.12	18.34 / 12.97	16.69 / 10.99
+ p.+I.+s.	24.97 / 17.45	19.33 / 13.66	17.01 / 11.68

Table 3.6: Results on accumulating the proposed approaches on the KITTI-3D *validation* set. Metric is AP_{40} of the Car category for BEV/3D detection. 'p.' denotes using projected 3D center for supervising the coarse center. 'I.' denotes using our IoU loss design. 's.' denotes the design for discarding distant samples.

Results on the KITTI-3D *validation* set. We also present our model's performance on the KITTI-3D *validation* set in Table 3.4. Note that some methods directly use the pre-trained model provided by DORN [47] as their depth estimator. However, the DORN's training set overlaps with the validation set of KITTI-3D, so we are not comparing these methods here. We can find that the proposed model performs better than all previous methods in the 3D detection task. For BEV detection task, our method outperforms all methods except for MonoPair. Compared with MonoPair, our method is better at detecting objects under strict conditions (0.7 IoU threshold), while MonoPair is slightly better at catching samples under loose conditions (0.5 IoU threshold). Also, note that our method shows better performance consistency between the validation set and the test set. This indicates that our method has better generalization ability, which is of great significance in autonomous/assisted driving.

Latency analysis. We test the proposed model on a single GTX 1080Ti GPU with a batch size of 1 for runtime analysis. As shown in Table 5.7, the proposed method can run at 25 FPS, meeting the requirement of real-time detection. Specifically, our method runs 4× faster than the two-stage detector M3D-RPN. Compared with MonoPair, which shares a similar framework as ours, our

	PC	RF	MT	Easy	Mod.	Hard
a	-	-	✓	98.08 / 1.32	92.31 / 1.04	84.75 / 1.16
b	-	✓	✓	98.08 / 13.98	92.31 / 10.81	84.75 / 9.59
c	✓	✓	-	94.55 / 12.31	88.79 / 10.30	79.29 / 8.82
d	✓	✓	✓	98.42 / 16.08	92.74 / 13.04	83.04 / 11.16

Table 3.7: Analysis for center definition and multitask learning. Metrics are AP_{40} of the Car category for 2D/3D detection tasks. 'PC', 'RF', and 'MT' represent 'projected 3D center', 'refinement', and 'multi-task learning'.

method can still save 16 ms for one image in the inference phase, mainly because: 1) we use standard DLA-34 as our backbone, instead of modified DL4-34 with DCN [38, 226]. 2) we apply fewer prediction heads in our model. 3) we don't need any post-processing. SMOKE [113] can run faster than our method. However, it only conducts 3D detection while the proposed method can perform 2D detection and 3D detection jointly.

Besides, although the detectors with pre-trained depth estimator usually have promising performance, the additional depth estimator introduce lots of computational overheads (*e.g.* the most commonly used DORN [47] takes about 400 ms to process a standard KITTI-3D image. See KITTI-3D Depth Benchmark for more details).

3.3.3 Pedestrian/Cyclist Detection

Here we present the Pedestrian/Cyclist detection results on the KITTI-3D *test* set in Table 5.11. Compared with cars, pedestrians/cyclists are more difficult to detect, and only [13, 31] provide the performances of those categories on the KITTI-3D test set. Specifically, the proposed method performs better than [13] and gets comparable results with [31]. But it is important to note that, since the number of training samples for those two categories is quite small, the performance may fluctuate to some extent.

3.3.4 Analysis

Accumulation of the proposed designs. Table 3.6 shows experimental results evaluating how the proposed designs contribute to the overall performance of this task. Our design in Section 3.2.4, which uses the projected 3D center for supervising center detection and influencing 2D detection ('+p.' in Table 3.6), improves 3D detection accuracy by 1.5. The IoU loss design in Section 3.2.6 further improves the accuracy by 0.3. And the design for discarding distant samples in Section 3.2.5 leads to 0.7 improvements.

	Easy	Mod.	Hard
baseline	16.12	12.97	10.99
+ hard encoding, $s = 40$	14.25	11.25	9.63
+ hard encoding, $s = 60$	17.45	13.66	11.68
+ soft encoding, $c = 40$, T=1	14.50	11.74	9.95
+ soft encoding, $c = 60$, T=1	17.50	13.54	11.32
+ soft encoding, $c = 60$, T=5	17.25	13.03	11.01

Table 3.8: Analysis for training samples. Metrics is AP_{40} of the Car category for 3D detection.

Supervision for coarse center detection and multi-task learning. We show the performance changes caused by center definition and multi-task learning in Table 3.7. Specifically, from setting **a** (used in [222]) and setting **b** (used in [31, 171]) in the table, predicting an offset to compensate for the misalignment between 2D center and projected 3D center can improve the performance of 3D detection significantly. Then, using the projected 3D center as the ground truth for coarse detection (setting **d**, our model) can further improve the performance. Besides, by comparing the setting **c** used in [113] and the setting **d** in our design, we can find the performance of 3D detection benefits from multi-task learning (performing 2D detection and 3D detection jointly). Note that the accuracy of 2D detection under setting **d** is also better than that under setting **c**, which suggests generating 2D bounding boxes from 3D detection may reduce the quality of the 2D detection results. The above conclusions are also reflected in Table 5.7 and 3.4.

Training samples. From Table 3.8, we can find that both removing some samples from the training set appropriately and reducing their training weights of them can improve overall performance. Note that those samples are only a small part of the whole training set and will not affect the representation learning of the network to the whole dataset. For example, in the 7,481 images in *trainval* set, only 1,301/767 samples beyond 60/65 meters, accounting for 4.5%/2.7% of the total 28,742 samples.

Uncertainty modeling. First, from Figure 3.5 and Table 3.9, we can find that uncertainty-based estimation improves the accuracy of the depth map, thereby improving the overall performance of monocular 3D detection. Second, the experimental result also shows that modeling uncertainty based on the Laplace distribution (all models in the main paper adopted this setting) is more suitable for our task than the Gaussian distribution.

Comparison with GIoU loss. We report the improvement introduced by the

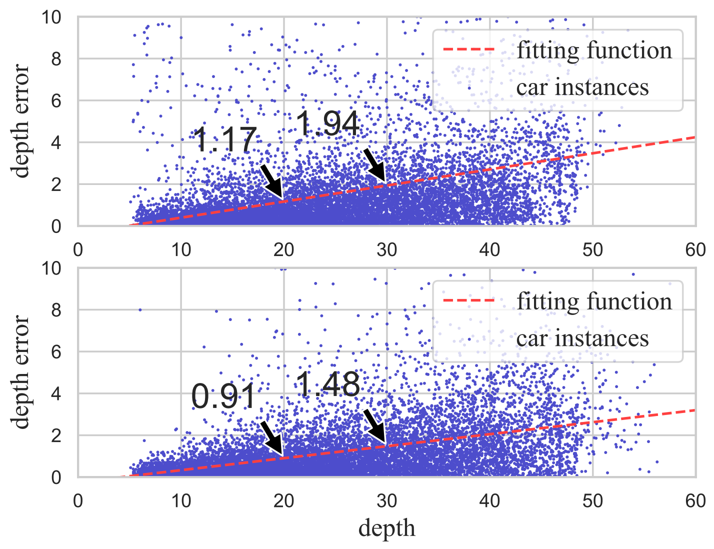


Figure 3.5: Errors of depth estimation. We show the errors of depth estimation as a function of the depth (x-axis) for the plain scheme (*top*) and the uncertainty aware scheme based on the Laplace likelihood (*bottom*).

uncert.	Easy	Mod.	Hard
-	18.56 / 13.18	14.24 / 10.15	12.13 / 8.45
Gaussian	18.68 / 13.20	14.22 / 10.41	12.08 / 8.69
Laplace	20.29 / 14.51	16.15 / 11.12	14.07 / 9.97

Table 3.9: Analysis for the designs of depth estimation. Metrics are AP$_{40}$ of the Car category for BEV/3D detection tasks.

proposed loss function in Table 3.7. To further validate its effectiveness, we also implement the 3D GIoU loss [202] for reference. Specifically, we add the 3D GIoU loss as a regularization item as in [202], investigating different weights considered in our baseline model, and the AP$_{40}$ of cars on the moderate setting on the KITTI-3D *validation* set (Table 3.10) show that our IoU oriented optimization improves accuracy but 3D-GIoU with different weights does not.

Performance of close-range objects. Figure 3.1 provides lots of insights to us. Except for the observations analyzed in the main paper, we also found that the performance degrades for the very close object. Here we provide our analysis for this. In particular, there are three main reasons in total. a) The close-range objects tend to have larger center misalignment (see Figure 3.3 for the statistics). b) The objects at closer ranges are usually more truncated, *e.g.* the red car (depth=3.7, truncation=0.88) and the black car (depth=6.2, truncation=0.34) in Figure 3.7. c) The training samples in the close range are fewer. For example, there are 5,979 cars in $[5m, 15m]$ and 6,707 cars in $[10m, 20m]$ on the KITTI-3D

Baseline	GIoU (w=0.5)	GIoU (w=1)	GIoU (w=5)	Ours
11.12	10.17	10.19	8.48	11.74

Table 3.10: Ablation study for the proposed loss function and 3D GIoU loss on the KITTI-3D *validation* set. The metric is AP_{40} of the Car category under the moderate setting.

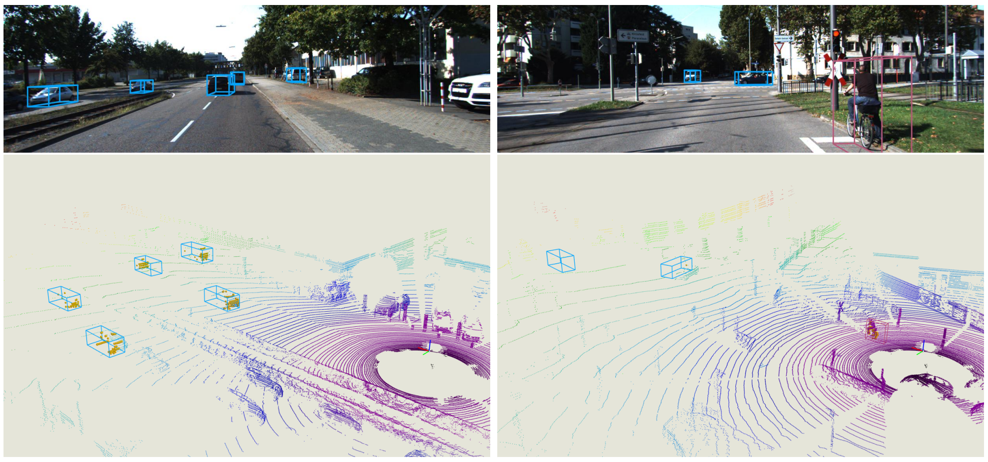


Figure 3.6: Qualitative results on the KITTI-3D *test* set. These results are based on the proposed model trained on the KITTI-3D *trainval* set, running at 25 FPS. We use blue, green, and red boxes to denote cars, pedestrians, and cyclists. LiDAR signals are only used for visualization. Best viewed in color with zoom-in.

trainval set, and the distribution for those samples are summarized in Table 3.11. Note that the KITTI-3D annotates the difficulty of each sample according to its size of the 2D bounding box, occlusion, and truncation. The instance with 'unKnown' tag usually means that it is extremely difficult to detect and is ignored in evaluation. With that in mind, the effective samples of those two ranges are 4,522 and 6,149. In summary, the low performance of the very close objects is caused by the limited training samples (c) and the large proportion of hard cases (a, b).

3.3.5 Qualitative Results

Visualization on the KITTI-3D test set. We visualize some representative outputs of the proposed method in Figure 7.7. To clearly show the object's position in the 3D world space, we also visualize the LiDAR signals. We can observe that our model outputs remarkably accurate 3D bounding boxes for the cases at a reasonable distance. We also find that our model outputs some false positive

Range	Easy	Moderate	Hard	UnKnown	Total
$[5m, 15m]$	2,131	1,428	963	1,457	5,979
$[10m, 20m]$	2,639	1,840	1,670	558	6,707

Table 3.11: Data distribution for the car samples located in $[5m, 15m]$ and $[10m, 20m]$. The data is collected from the KITTI-3D *trainval* set.

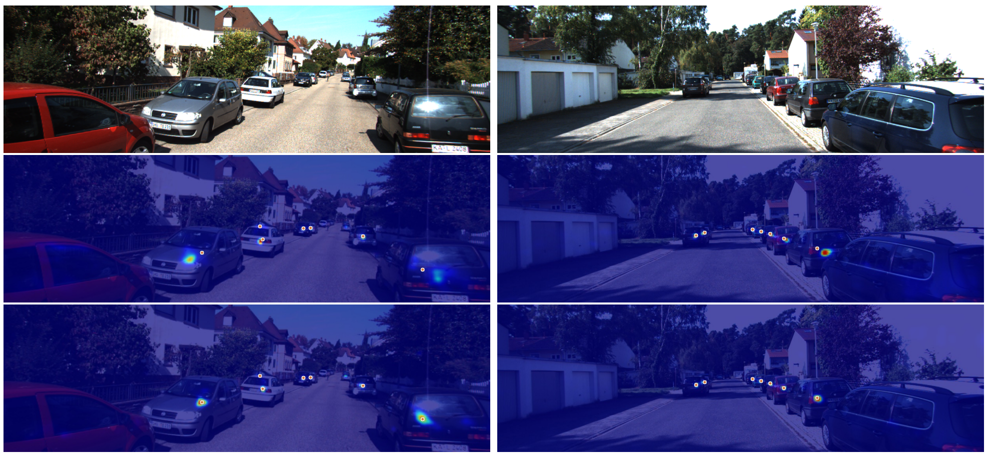


Figure 3.7: Qualitative comparison for the learned features of coarse center detection task on the KITTI-3D *validation* set. *Top:* the input image. *Middle:* the features of the coarse center detection branch supervised by the 2D center. *Bottom:* the features of the coarse center detection branch supervised by the projected 3D center. We use the white circle to highlight the ground truth projected 3D center for better comparison. Best viewed in color with zooming in.

samples, *e.g.* the 3D box on the right in the sixth picture, and the foremost reason for that is the imprecise depth or center estimation. Note that the dimension and orientation estimation for those cases are still accurate.

Visualization of learned features. From Figure 3.4 in the main paper, we can see there is a misalignment between the center of the 2D bounding box and the projected center of the 3D object, especially for close objects (see Figure 3.3 and Figure 3.4). Accordingly, we propose our solution to this problem. Here we visualize the learned features of the coarse center detection branch in Figure 3.7 to show the effectiveness of the proposed method. The qualitative results clearly show that using the projected 3D center as ground truth can make the coarse center more accurate, thereby improving localization accuracy.

3.4 Conclusion

In this Chapter, we systematically analyze the problems in monocular 3D detection and find that localization error is the bottleneck of this task. To alleviate this problem, we first revisit the misalignment between the center of the 2D bounding box and the projected center of the 3D object. We argue that directly detecting projected 3D centers can reduce the localization error and 2D detection is conducive to optimizing 3D detection. Besides, we also find distant samples are almost impossible to detect accurately with the existing technologies, and discarding these samples from the training set will stop them from distracting the network. Finally, we also proposed an IoU-oriented loss for 3D size estimation. Extensive experiments on the challenging KITTI-3D dataset show the effectiveness of the proposed strategies.

Chapter 4

Accurate Pseudo-LiDAR Model with Color-Embedding

4.1 Introduction

In recent years, with the development of technologies in computer vision and deep learning [175, 64], numerous impressive methods are proposed for accurate 2D object detection [56, 55, 158, 63, 107]. However, beyond getting 2D bounding boxes or pixel masks, 3D object detection is eagerly in demand in many applications such as autonomous driving and robotic applications because it can describe objects in a more realistic way. Now, this problem received more and more concern from scholars. Because LiDAR provides reliable depth information that can be used to accurately localize objects and characterize their shapes, many approaches [94, 117, 125, 146, 29, 159, 206] use LiDAR point clouds as their input, and get impressive detection results in autonomous driving scenarios. In contrast, some other studies [18, 27, 26, 199, 129, 176, 95] are devoted to replacing the LiDAR with cheaper monocular cameras, which are readily available in daily life. As LiDAR is much more expensive and inspired by the remarkable progress in image-based depth prediction techniques, this paper focuses on the high-performance detection of 3D objects utilizing only monocular images. However, image-based 3D detection is very challenging, and there is a huge gap between the performance of image-based methods and LiDAR-based methods. We show in this work that we can largely boost the performance of image-based 3D detection by transforming the input data representation.

Typical image-based 3D object detection models [18, 26, 27, 176] adopted the pipeline similar to 2D detectors and mainly focused on RGB features extracted from 2D images. However, these features are not suitable for 3D-related

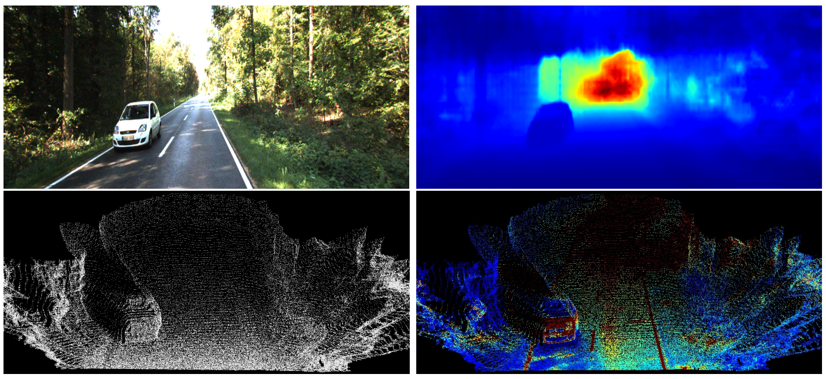


Figure 4.1: Different representations of input data. *Top left:* RGB image. *Top right:* Depth map. *Bottom left:* Point cloud. *Bottom right:* RGB augmenting point cloud (only R-channel is mapped for this visualization). Note that all the representations we mentioned can be generated by a single RGB image.

tasks because of the lack of spatial information. This is one of the main reasons why early studies failed to get better performance. An intuitive solution is that we can use a CNN to predict the depth maps [200, 201, 47] and then use them as input if we do not have the depth data available. Although depth information is helpful to 3D scene understanding, simply using it as an additional channel of RGB images such as [199] does not compensate for the performance difference between image-based methods and LiDAR-based methods. There is no doubt that LiDAR data is much more accurate than estimated depth, here we argue that the performance gap is not only due to the accuracy of the data but also its representation (see Fig. 4.1 for different input representations on monocular 3D detection task). In order to narrow the gap and make the estimated depth a bigger role, we need a more explicit representation form such as the point cloud which describes real-world 3D coordinates rather than depth with a relative position in images. For example, objects with different positions in the 3D world may have the same coordinates in the image plane, which brings difficulties for the network to estimate the final results. The benefits for transform depth maps into point clouds can be enumerated as follow: (1) Point cloud data shows the spatial information explicitly, which makes it easier for the network to learn the non-linear mapping from input to output. (2) Richer features can be learned by the network because some specific spatial structures exist only in 3D space. (3) The recent significant progress of deep learning on point clouds provides a solid building brick, which we can estimate 3D detection results in a more effective and efficient way.

Based on the observations above, a monocular 3D object detection framework is proposed. The main idea for the design of our method is to find a better

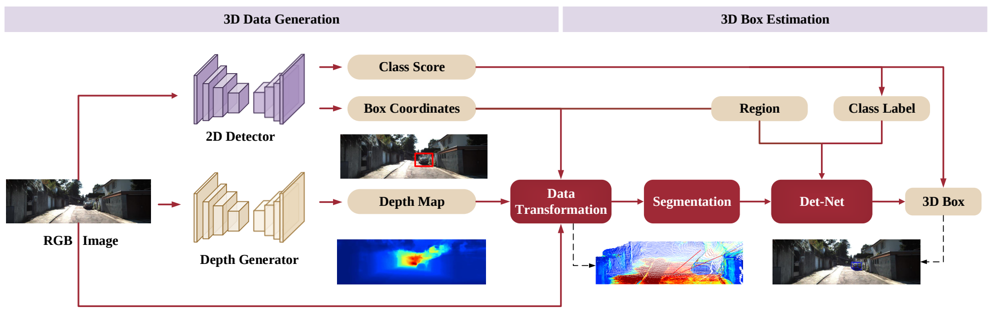


Figure 4.2: The proposed framework for monocular 3D object detection.

input representation. Specifically, we first learn to use front-end deep CNNs and the input RGB data to produce two intermediate tasks involving 2D detection [55, 158] and depth estimation [47, 201] (see Fig. 4.2). Then, we transform depth maps into point clouds with the help of camera calibration files in order to give the 3D information explicitly and used them as input data for subsequent steps. Besides, another crucial component that ensures the performance of the proposed method is the multi-modal features fusion module. After aggregating RGB information which is complementary to 3D point clouds, the discriminative capability of features used to describe the 3D object is further enhanced. Note that, when the optimization of all networks is finished, the inference phase is only based on the RGB input.

The contributions of this Chapter can be summarized as:

- We propose a new framework for monocular 3D object detection which transforms 2D images to 3D point clouds and performs the 3D detection effectively and efficiently.
- We design a features fusion strategy to fully exploit the advantages of RGB cue and point cloud to boost the detection performance, which can be also applied in other scenarios such as LiDAR-based 3D detection.
- Evaluation on the challenging KITTI-3D dataset [52] shows our method outperforms all state-of-the-art monocular methods by around 15% and 11% higher AP on 3D localization and detection tasks, respectively.

4.2 Method

4.3 Proposed Method

In this section, we describe the proposed framework for monocular-based 3D object detection. Specifically, we first present an overview of the proposed method, and then introduce the details of it. Finally, we show the optimization and implementation details for the overall network.

4.3.1 Approach Overview

As shown in Fig. 4.2, the proposed 3D detection framework consists of two main stages. In the 3D data generation phase, we trained two deep CNNs to do intermediate tasks (2D detection and depth estimation) to get position and depth information. In particular, we transfer the generated depth into point cloud which is a better representation for 3D detection, and then we use the 2D bounding box to get the prior information about the location of the RoI (region of interest). Finally, we extract the points in each RoI as our input data for subsequent steps. In the 3D box estimation phase, in order to improve the final task, we design two modules for background points segmentation and RGB information aggregation, respectively. After that, we use PointNet as our backbone net to predict the 3D location, dimension, and orientation for each RoI. Note that the confidence scores of 2D boxes are assigned to their corresponding 3D boxes.

4.3.2 3D Data Generation

Intermediate tasks. As we all know that 3D detection using only monocular images is a very challenging task because image appearance can not determine the 3D coordinates of the object. Therefore, we train two deep CNN to generate depth map and 2D bounding box to provide spatial information and position prior. We adopt some existing algorithms to do these intermediate tasks, and give a detailed analysis of the impact of these algorithms on overall performance in experiment part.

Input representation. This work focuses more on how to use depth information than on how to get them. We believe that one of the main reasons why previous image-based 3D detectors fail to get better results is they don't make good use of depth maps. Simply using the depth map as an additional channel of RGB image such as [199, 124], and then expecting a neural network to extract effective features automatically is not the best solution. In contrast, we transform the estimated depth into point cloud with the help of camera calibration files provided by KITTI-3D (see Fig. 4.1 for different input representations) and then use it as our data input form. Specifically, given a pixel coordinate (u, v) with depth d in the 2D image space, the 3D coordinates (x, y, z) in camera coordinate system can be computed as:

$$\begin{cases} z = d, \\ x = (u - C_x) * z / f, \\ y = (v - C_y) * z / f, \end{cases} \tag{4.1}$$

where f is the focal length of the camera, (C_x, C_y) is the principal point. The input point cloud S can be generated using depth map and 2D bounding box $\mathbf{B}$ as follows:

$$S = \{\, p \mid p \leftarrow \mathbf{F}(v),\ v \in \mathbf{B}\}, \tag{4.2}$$

where v is the pixel in depth map and $\mathbf{F}$ is the transforming function introduced by Eq. 4.1. It should be noted that, like most of the monocular-based methods, we use camera calibration files in our approach. Actually, we can also use a point cloud encoder-decoder net to learn a mapping from (u, v, d) to (x, y, z), thus we don't need the camera parameters during the testing phase anymore. In our measurements, we observe that there is no visible performance difference between these two methods. This is because the error introduced in the point cloud generation phase is much less than the noise contained in the depth map itself.

4.3.3 3D Box Estimation

Point segmentation. After the 3D data generation phase, the input data is encoded as point clouds. However, there are many background points in these data and these background points should be discarded in order to estimate the position of the target accurately. Qi *et al.*[146] propose a 3D instance segmentation PointNet to solve this problem in LiDAR data. But that strategy requires additional pre-processing to generate segmentation labels from 3D object ground truth. More importantly, there will be severe noise even if we use the same labeling method because the points we reconstruct are relatively unstable. For these reasons, we propose a simple but effective segmentation method based on depth prior to segment the points. Specifically, we first compute the depth mean in each 2D bounding box in order to get the approximate position of RoI, and use it as the threshold. All points with the Z-channel value greater than this threshold are considered as background points. The processed point set S' can be expressed as:

$$S' = \{\, p \mid p_v \leq \frac{\sum_{p \in S} p_v}{|S|} + r,\ p \in S\}, \tag{4.3}$$

where p_v denotes the Z-channel value (which is equal to depth) of the point and r is a bias used to correct the threshold. Finally, we randomly select a fixed number of points in the point set S' as the output of this module in order to ensure consistency of the number of subsequent network's input points.

3D box estimation. Before we estimate final 3D results, we follow [146] to predict the center δ of RoI using a lightweight network and use it to update the point cloud as follows:

$$S'' = \{\, p \mid p - \delta,\ p \in S'\}, \tag{4.4}$$

where S'' is the set of points we used to do the final task. Then, we choose PointNet [147] as our 3D detection backbone network to estimate the 3D object which is encoded by its center (x, y, z), size (h, w, l) and heading angle θ. Same as other works, we only consider one orientation because of the assumption that the road surface is flat and the other two angles do not have possible variation. One other thing to note is that the center C we estimate here is a 'residual' center, which means the real center is $C + \delta$. Finally, we assign the confidence scores of the 2D bounding boxes to their corresponding 3D detection results.

4.3.4 RGB Information Aggregation

In order to further improve the performance and robustness of our method, we propose to aggregate complementary RGB information to the point cloud. Specifically, we add RGB information to the generated point cloud by replacing Eq. 4.2 with:

$$S = \{\, p \mid p \leftarrow [\mathbf{F}(v), \mathbf{D}(v)],\ v \in \mathbf{B}\}, \tag{4.5}$$

where $\mathbf{D}$ is a function that outputs the corresponding RGB values of the input point. In this way, the points are encoded as 6D vectors: $[x, y, z, r, g, b]$. However, simply relying on this simple method (we call it 'plain concat' in the experiment part) to add RGB information is not feasible. So, as shown in Fig. 4.3, we introduce an attention mechanism for the fusion task. The attention mechanism has been successfully applied in various tasks such as image caption generation and machine translation for selecting useful information. Specifically, we utilize the attention mechanism for guiding the message passing between the spatial

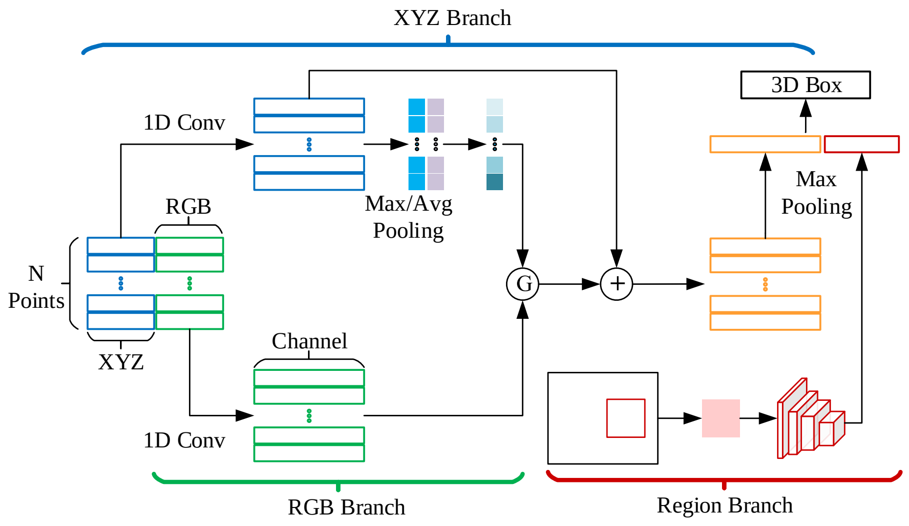


Figure 4.3: 3D box estimation (Det-Net) with RGB features fusion module. **G** is an attention map generated using Eq. 4.6.

features and RGB features. Since the passed information flow is not always useful, the attention can act as a gate function to control the flow, in other words, to make the network automatically learn to focus or to ignore information from other features. When we pass RGB message to its corresponding point, an attention map **G** is first produced from the feature maps **F** generated from XYZ branch as follows:

$$\mathbf{G} \leftarrow \sigma(f([\mathbf{F}^{xyz}_{max}, \mathbf{F}^{xyz}_{avg}])), \tag{4.6}$$

where f is the nonlinear function learned from a convolution layer and σ is a sigmoid function for normalizing the attention map. Then the message is passed with the attention map as follows:

$$\mathbf{F}^{xyz} \leftarrow \mathbf{F}^{xyz} + \mathbf{G} \odot \mathbf{F}^{rgb}, \tag{4.7}$$

where $\odot$ denotes element-wise multiplication. In addition to point-level feature fusion, we also introduce another branch to provide object-level RGB information. In particular, we first crop the RoI from RGB image and resize it to 128×128. Then we use a CNN to extract the object-level feature maps $\mathbf{F}^{obj}$ and the final feature maps set $\mathbf{F}$ obtained from the fusion module is: $\mathbf{F} \leftarrow CONCAT(\mathbf{F}^{xyz}, \mathbf{F}^{obj})$, where $CONCAT$ denotes the concatenation operation.

4.3.5 Implementation Details.

Optimization. The whole training process is performed in two phases. In the first phase, we only optimize the intermediate nets according to the training strategies of the original papers. After that, we simultaneously optimize the two networks for 3D detection jointly with a multi-task loss function:

$$L = L_{loc} + L_{det} + \lambda L_{corner}, \tag{4.8}$$

where L_{loc} is the loss function for the lightweight location net (center only) and L_{loc} is for 3D detection net (center, size and heading angle). We also use the corner loss [146] where the output targets are first decoded into oriented 3D boxes and then smooth L1 loss is computed on the (x, y, z) coordinates of eight box corners directly with regard to ground truth. We train the nets for 200 epochs using Adam optimizer with a batch size of 32. The learning rate is initially set to 0.001 and reduced by half for every 20 epochs. The whole training process can be completed in one day.

Implementation details. The proposed method is implemented based on PyTorch and on Nvidia 1080Ti GPUs. The two intermediate networks of the proposed method naturally support any network structure. We implement some different methods as described in their papers exactly, and the relevant analysis can be found in the experimental part. For the 3D detection nets, we use PointNet as our backbone nets and train them from scratch with random initialization. Moreover, the dropout strategy with a keep rate 0.7 is applied to every fully connected layer except the last one. For the RGB values, we first normalize the range of them to (0, 1) by dividing 255, and then the data distribution of each color channel is regularized into a standard normal distribution. For the region branch in RGB features fusion module, we use ResNet-34 with half channels and global pooling to get the $1 \times 1 \times 256$ features.

4.4 Experimental Results

We evaluate our approach on the challenging KITTI-3D dataset [52] which provides 7,481 images for training and 7,518 images for testing. Detection and localization tasks are evaluated in three regimes: *easy*, *moderate*, and *hard*, according to the occlusion and truncation levels of objects. Since the ground truth for the test set is not available and the access to the test server is limited, we conduct comprehensive evaluation using the protocol described in [26, 27, 29], and subdivide the training data into a *training* set and a *validation* set, which results in 3,712 data samples for training and 3,769 data samples for validation. The split avoids samples from the same sequence being included in both *training* and *validation* set [26].

4.4.1 Comparing with other methods

Baselines. As this work aims at monocular 3D object detection, our approach is mainly compared to other methods with only monocular images as input. Here five methods are chosen for comparisons: Mono3D [26], Deep3DBox [129] and Multi-Fusion [199], ROI-10D [124], MonoGRNet [152] and Pseudo-LiDAR [192].

Method	IoU=0.5			IoU=0.7		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [26]	30.50	22.39	19.16	5.22	5.19	4.13
Deep3DBox [129]	30.02	23.77	18.83	9.99	7.71	5.30
Multi-Fusion [199]	55.02	36.73	31.27	22.03	13.63	11.60
ROI-10D [124]	-	-	-	14.76	9.55	7.57
Pseudo-LiDAR [192]	70.8	49.4	42.7	40.6	26.3	22.9
AM3D (Ours)	**72.64**	**51.82**	**44.21**	**43.75**	**28.39**	**23.87**

Table 4.1: 3D localization performance: Average Precision (AP_{loc}) (in %) of bird's eye view boxes on KITTI-3D *validation* set.

Method	IoU=0.5			IoU=0.7		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Mono3D [26]	25.19	18.20	15.52	2.53	2.31	2.31
Deep3DBox [129]	27.04	20.55	15.88	5.85	4.10	3.84
Multi-Fusion [199]	47.88	29.48	26.44	10.53	5.69	5.39
ROI-10D [124]	-	-	-	10.25	6.39	6.18
MonoGRNet [152]	50.51	36.97	30.82	13.88	10.19	7.62
Pseudo-LiDAR [192]	66.3	42.3	38.5	28.2	18.5	16.4
AM3D (Ours)	**68.86**	**49.19**	**42.24**	**32.23**	**21.09**	**17.26**

Table 4.2: 3D detection performance: Average Precision (AP_{3D}) (in %) of 3D boxes on KITTI-3D *validation* set.

Car. The evaluation results of 3D localization and detection tasks on KITTI-3D *validation* set are presented in Table 4.1 and 4.2, respectively. The proposed method consistently outperforms all the competing approaches across all three difficulty levels. For localization task, the proposed method outperforms Multi-Fusion [199] by ∼**15** AP_{loc} in *moderate* setting. For 3D detection task, our method achieves ∼**12.2** and ∼**10.9** AP_{3D} improvement (moderate) over the recently proposed MonoGRNet [152] under IoU thresholds of 0.5 and 0.7. In the *easy* setting, our improvement is more prominent. Specifically, our method achieves ∼**21.7**

and ∼**18.4** improvement over previous state-of-the-art on localization and detection tasks (IoU=0.7). Note that there is no complicated prior knowledge or constraints such as [26, 27, 124], which strongly confirms the importance of data representation.

Compared with Pseudo-LiDAR [192], which is concurrent to this work, the proposed method has about ∼**1.5** *AP* improvement on each metric. This is because of the modification of the background points segmentation algorithm and the introduction of RGB information. We will discuss this in detail in Sec. 4.4.2.

Table 4.3 shows the results on *testing* set, and more details, such as Precision-Recall curve, can be found on KITTI-3D official server. The *testing* set results also show the superiority of our method in performance compared with others.

Method	Task	Easy	Moderate	Hard
Multi-Fusion [199]	Loc.	13.73	9.62	8.22
RoI-10D [124]	Loc.	16.77	12.40	11.39
Ours	Loc.	**27.91**	**22.24**	**18.62**
Multi-Fusion [199]	Det.	7.08	5.18	4.68
RoI-10D [124]	Det.	12.30	10.30	9.39
Ours	Det.	**21.48**	**16.08**	**15.26**

Table 4.3: AP(%) for 3D localization (Loc.) and 3D detection (Det.) tasks on the KITTI-3D *testing* set.

4.4.2 Detailed analysis of proposed method

In this section we provide analysis and ablation experiments to validate our design choices.

RGB information. We further evaluate the effect of the proposed RGB fusion module, and the baselines are the proposed method without RGB values and using them as additional channels of generated points. Table 4.4 shows the relevant results for *Car* category on KITTI-3D. It can be seen that the proposed module obtains around **2.1** and **1.6** mAP improvement (moderate) on localization and detection task, respectively. The qualitative comparisons can be found in Fig **??**. Quantitative and qualitative results both show the effectiveness of the proposed RGB fusion module. Besides, one thing to note is that incorrect use of RGB information such as plain concat will lead to performance degradation.

Points segmentation. We compare the proposed points segmentation method and the 3D segmentation PointNet which is used in [146]. The baseline is to estimate 3D boxes directly using point clouds with noise which can be regarded as all points are classified into positive samples. As shown in Table 4.5,

	Task	Easy	Moderate	Hard
w/o RGB	Loc.	41.29	26.28	22.75
plain concat	Loc.	36.17	25.34	21.94
ours	Loc.	**43.75**	**28.39**	**23.87**
w/o RGB	Det.	30.73	19.46	16.72
plain concat	Det.	27.20	18.25	16.15
ours	Det.	**32.23**	**21.09**	**17.26**

Table 4.4: Ablation study of RGB information. The metric is $AP_{3D}^{0.7}$ on KITTI-3D *validation* set.

	IoU	Easy	Moderate	Hard
w/o segmentation	0.5	66.42	44.53	40.60
seg-net used in [146]	0.5	67.01	45.51	40.65
ours	0.5	**68.86**	**49.19**	**42.24**
w/o segmentation	0.7	27.04	18.22	16.13
seg-net used in [146]	0.7	29.49	18.70	16.57
ours	0.7	**32.23**	**21.09**	**17.26**

Table 4.5: Ablation study of points segmentation. The metric is $AP_{3D}^{0.7}$ on KITTI-3D *validation* set.

our prior-based method outperforms baseline and segmentation PointNet obviously which proves the effectiveness of the proposed method, and Table 4.6 shows that the proposed method is robust for varying thresholds. Meanwhile, the experimental results also show that the learning-based method is not applicable to approximate the point clouds segmentation task because it's difficult to obtain reliable labels. Besides, the proposed method is also much faster than segmentation PointNet (about 5ms on CPU *v.s.* 20ms on GPU for each proposal).

r	Easy	Moderate	Hard
-0.5	31.13	20.01	16.81
0.0	31.87	20.55	17.03
0.5	**32.23**	**21.09**	**17.26**
1.0	31.93	20.93	17.18

Table 4.6: $AP_{3D}^{0.7}$(%) of different points segmentation threshold r (in meters) for 3D detection on the KITTI-3D *validation* set.

Depth maps. As described in Sec. 4.3, our approach depends on the point clouds generated from the output of the depth generator. In order to study the impact of the quality of depth maps on the overall performance of the proposed method, we implemented four different depth generators [57, 103, 126, 20]. From the results shown in Table 4.7, we find that 3D detection accuracy increases significantly when using more accurate depth (more details about the accuracy of depth maps can be found in the supplement material). It's worth noting that even if we use the unsupervised monocular depth generator [57], the proposed method still outperforms [124] by a large margin.

Depth	Task	Easy	Mod.	Hard
MonoDepth[57]	Loc.	32.42	20.26	17.21
DORN[47]	Loc.	43.75	28.39	23.87
DispNet[126]	Loc.	47.41	30.72	25.66
PSMNet [20]	Loc.	60.18	34.01	30.32
MonoDepth[57]	Det.	23.12	15.45	14.19
DORN[47]	Det.	32.23	21.09	17.26
DispNet[126]	Det.	36.97	23.69	19.25
PSMNet [20]	Det.	45.85	26.03	23.16

Table 4.7: Comparisons of different depth generators. Metrics are $AP_{loc}^{0.7}$ and $AP_{3D}^{0.7}$ on KITTI-3D *validation* set.

Sampling quantity. Some studies such as [147, 148] observe that classification/segmentation accuracy will decrease dramatically as the number of points decreases, and we will show that our approach is not so sensitive to the number of points. In our approach, we randomly select a fixed number (512 points for default configuration) of point clouds to do 3D detection task. Table 4.8 shows the performance of the proposed method under different sampling quantities. According to the results, AP_{3D} will increase as the number of points increases at the beginning. Then, after reaching a certain level ($\sim$512 points), the performance tends to be stable. It is worth noting that we still get a relatively good detection performance even if there are few sampling points.

Sampling Quantity	Easy	Mod.	Hard
64	27.91	19.41	16.31
128	29.72	19.62	16.64
256	30.99	20.71	17.18
512	**32.23**	**21.09**	**17.26**
1024	31.44	21.01	17.23

Table 4.8: Comparisons of different sampling quantities. The metric is $AP_{3D}^{0.7}$(%) on KITTI-3D *validation* set. Note that the number of sample points is consistent in the training and testing phases.

Robustness. We show that the proposed method is robust to various kinds of input corruption. We first set the sampling quantity to 512 in the training phase, but use different values in the testing phase. Fig. 4.4 shows that the

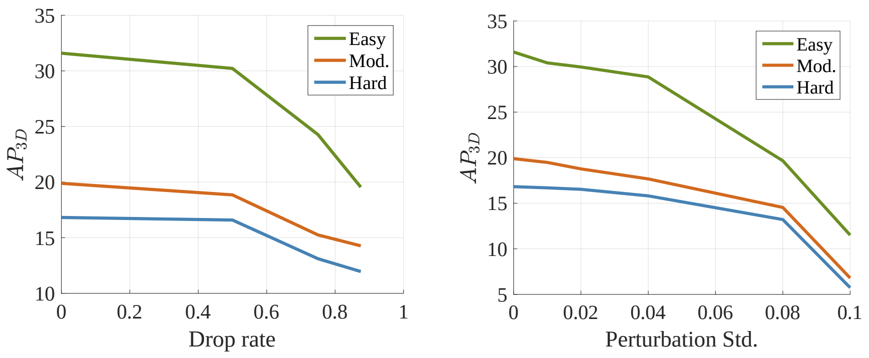


Figure 4.4: *Left*: robustness test of random point dropout. *Right*: robustness test of random perturbations (Gaussian noise is added into each point independently). The metric is $AP_{3D}^{0.7}(\%)$ for *Car* on KITTI-3D *validation* set.

proposed method has more than 70% AP_{3D} even when 80% of the points are missed. Then, we also test the robustness of model-to-point perturbations.

Network architecture. We also investigate the impact of different 3D detection network architectures on overall performance (the previously reported results are all based on PointNet), and the experimental result are shown in Table. 4.9.

	Data	Easy	Mod.	Hard
PointNet [147]	Mono	32.23	21.09	17.26
PointNet++ [148]	Mono	33.17	21.71	17.61
RSNet [73]	Mono	33.93	22.34	17.79

Table 4.9: Comparisons of different 3D detection network architectures. The metric is $AP_{3D}^{0.7}(\%)$ on KITTI-3D *validation* set.

Accuracy of depth maps. Tab. 4.10 and Tab. 4.11 show the accuracy of the monocular and stereo depth prediction methods listed in Tab. 4.7, respectively. Combined with Tab. 4.7, it is evident that 3D detection accuracy increases significantly when using much more accurate depth (or disparity). Note that the metrics for these two kinds of methods are different.

	Abs Rel	Sq Rel	RMSE	RMSE_{log}
MonoDepth	0.097	0.896	5.093	0.176
DORN	0.071	0.268	2.271	0.116

Table 4.10: Accuracy of depth prediction (monocular) on KITTI-3D *validation* set. lower is better.

Extensions of stereo and LiDAR. To further evaluate the proposed method, we

	D1-bg	D1-fg	D1-all
DispNet	4.32 %	4.41 %	4.34 %
PSMNet	1.86 %	4.62 %	2.32 %

Table 4.11: Accuracy of depth prediction (stereo) on KITTI-3D *test* set. lower is better.

extend it to stereo-based and LiDAR-based versions. We select some representational methods based on stereo images (or LiDAR point clouds) and report the comparative results in Table 4.12. The experimental results show that our method is able to give a competitive performance when using LiDAR point clouds or stereo images as input.

Note that the proposed method with LiDAR point cloud input outperforms F-PointNet [146] by **1.8** AP_{3D}, which proves that our RGB fusion module is equally effective for LiDAR-based methods.

Method	Data	Easy	Mod.	Hard
3DOP [27]	Stereo	6.55	5.07	4.10
Multi-Fusion [199]	Stereo	-	9.80	-
ours	Stereo	**45.85**	**26.03**	**23.16**
VoxelNet [223]	LiDAR	81.97	65.46	62.85
FPointNet [146]	LiDAR	83.26	69.28	62.56
ours	LiDAR	**84.53**	**71.07**	**63.49**

Table 4.12: $AP_{3D}^{0.7}$(%) of extended versions of proposed method and related works.

2D detectors. Tab. 4.13 shows the correlation between the performance of 2D detectors and resulting 3D detection performance. We can see that improving the performance of the 2D detector is an effective method to improve the overall detection accuracy. However, the huge gap between the performance of the 2D detector and the final 3D estimator reveals there is still a lot of room for improvement without modifying the 2D detector. The implementation details of the 2D detectors we used can be found in RRC [157] and F-PointNet [146].

	AP_{2D}			AP_{3D}		
	Easy	Mod.	Hard	Easy	Mod.	Hard
[157]	88.4	86.7	76.6	31.1	20.0	16.8
[146]	90.5	89.9	80.7	32.2	21.1	17.3

Table 4.13: Comparisons of different 2D detectors. Metrics are AP_{2D} and AP_{3D} on KITTI-3D *validation* set.

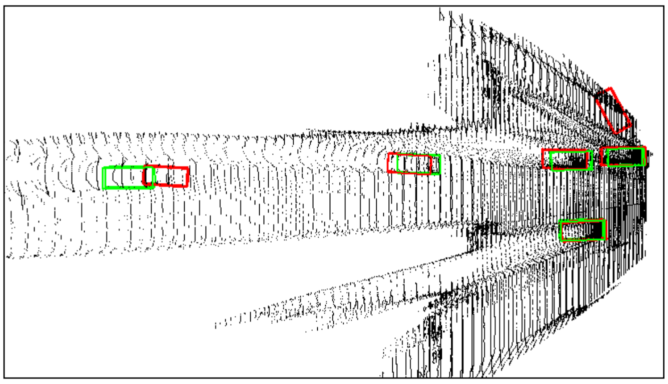


Figure 4.5: A qualitative result of 3D localization : 3D Boxes are projected to the ground plane. Red boxes represent our predictions, and green boxes come from the ground truth.

Pedestrian and cyclist. Most of the previous image-based 3D detection methods only focus on *Car* category as KITTI-3D provides enough instances to train their models. Our model can also get a promising detection performance on *Pedestrian* and *Cyclist* categories because it is much easier and more effective to do data augmentation for point clouds than depth maps used in previous methods. Table 5.11 shows their AP_{loc} and AP_{3D} on KITTI-3D *validation* set.

Category	IoU	Task	Easy	Moderate	Hard
Pedestrian	0.25	Loc.	40.77	34.02	29.83
Pedestrian	0.25	Det.	40.17	33.45	29.28
Pedestrian	0.5	Loc.	14.30	11.26	9.23
Pedestrian	0.5	Det.	11.29	9.01	7.04
Cyclist	0.25	Loc.	28.15	17.79	16.57
Cyclist	0.25	Det.	24.80	15.66	15.11
Cyclist	0.5	Loc.	10.12	6.39	5.63
Cyclist	0.5	Det.	8.90	4.81	4.52

Table 4.14: Benchmarks for Pedestrian and Cyclist. 3D localization and detection AP(%) on KITTI-3D *validation* set for *Pedestrian* and *Cyslist*. The IoU threshold is set to 0.25 and 0.5 for better comparison.

4.4.3 Qualitative Results and Failure Mode

We visualize some detection results of our approach in Fig. 4.6 and a typical localization result in Fig. 4.5. In general, our algorithm can get a good detection result. However, because it's a 2D-driven framework, the proposed method will fail if the 2D box is a false positive sample or missing. Besides, for distant objects, our algorithm is difficult to give accurate results because the depth is not reliable (the leftmost car in Fig. 4.5 is 70.35 meters away from the camera).

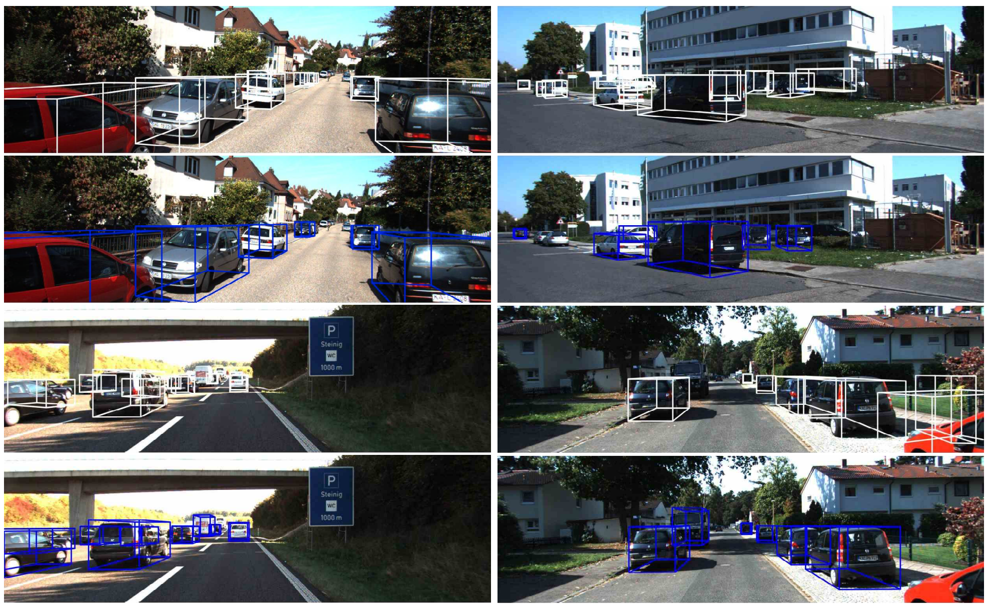


Figure 4.6: Qualitative comparisons of 3D detection results: 3D Boxes are projected to the image plane. White boxes represent our predictions, and blue boxes come from the ground truth.

4.5 Conclusions

We proposed a framework for accurate 3D object detection with monocular images in this Chapter. Unlike other image-based methods, our method solves this problem in the reconstructed 3D space in order to exploit 3D contexts explicitly. We argue that point cloud representation is more suitable for 3D-related tasks than depth maps. Besides, we propose a multi-modal feature fusion module to embed the complementary RGB cue into the generated point cloud representation to enhance the discriminative capability of generated point clouds. Our approach significantly outperforms existing monocular-based methods for 3D localization and detection tasks on KITTI-3D benchmark. In addition, the extended versions verify the design strategy can also be applied to stereo-based and LiDAR-based methods.

Chapter 5

Rethinking Pseudo-LiDAR Representation

5.1 Introduction

3D object detection has received increasing attention from both industry and academia because of its wide applications in various fields such as autonomous driving and robotics. Existing algorithms largely rely on LiDAR sensors, which provide accurate 3D point clouds of the surrounding environment. Although these approaches achieve impressive performance, the excessive dependence on expensive equipment restricts their application prospects.

Compared with fast-developing LiDAR-based algorithms, 3D detection results produced from only RGB images lag considerably behind. This can be attributed to the ill-posed nature of the problem, where a lack of explicit knowledge about the unobserved depth dimension significantly increases the task complexity. An intuitive solution is that we can use a Convolutional Neural Network (CNN) to predict the depth map [47, 57] and then use it to augment the input data if we do not have the available depth information. Although the estimated depth map is helpful to 3D scene understanding, the performance improvement brought by it is still limited.

Several recently proposed algorithms [192, 121, 195] transform the estimated depth map into pseudo-LiDAR representation, and then apply LiDAR-based methods to the transformed data. Surprisingly, this simple yet effective method achieves significant improvement in the 3D detection accuracy on the challenging KITTI-3D dataset. However, it is unclear why such a representation can bring so much performance improvement. According to the empirical explanation of proponents, the choice of representations is the critical success factor of 3D detection systems. Compared with image representation, they believe that pseudo-LiDAR is more suitable for describing the 3D structure of objects, which is the main reason for performance improvement. However, in the absence of direct evidence, the correctness of this statement is still open to doubt.

In this paper, we aim to explore the essential reasons for this phenomenon. Specifically, we carefully build an image representation-based 3D object detector named PatchNet-vanilla, which is an equivalent implementation of pseudo-LiDAR [192] except for the representation of input data. With this detector, we can compare the influence of these two kinds of representations on the 3D detection task in depth. Different from the arguments of other works [192, 121, 195], we observe that the performances of PatchNet-vanilla and pseudo-LiDAR [192] are completely matched, which means that data representation has no effect on 3D detection performance. Moreover, we perform ablation studies on the input data and observe that the real thing matters is coordinate transformation from the image coordinate system to the LiDAR coordinate system, which implicitly encodes the camera calibration information into input data.

PatchNet-vanilla also hints to us that pseudo-LiDAR representation is not necessary to improve the accuracy of image-based 3D detection. By integrating the generated 3D coordinates as additional channels of input data, our 3D detector gets promising performance. More importantly, this approach can be easily generalized to other image-based detectors. Also notice that, as a kind of non-grid structured data, pseudo-LiDAR signals commonly need point-wise CNNs [147, 148] to process. However, the development of these technologies still lags behind the standard CNNs. From this point of view, the image-based detectors should outperform their counterparts based on pseudo-LiDAR. To confirm this hypothesis, PatchNet was proposed by extending our original model (e.g., using more powerful backbone networks [64, 70]), and outperforming other pseudo-LiDAR-based detectors on the KITTI-3D dataset. In addition, there are other benefits from using images directly as the network's inputs, such as allowing us to train an end-to-end 3D detector. Based on the above reasons, we argue that image representation-based 3D detectors have greater development potential.

In summary, the contributions of this Chapter are as follows: First, through sufficient experimental demonstration, we confirm the reason why the pseudo-LiDAR representation is effective is not the data representation itself, but the

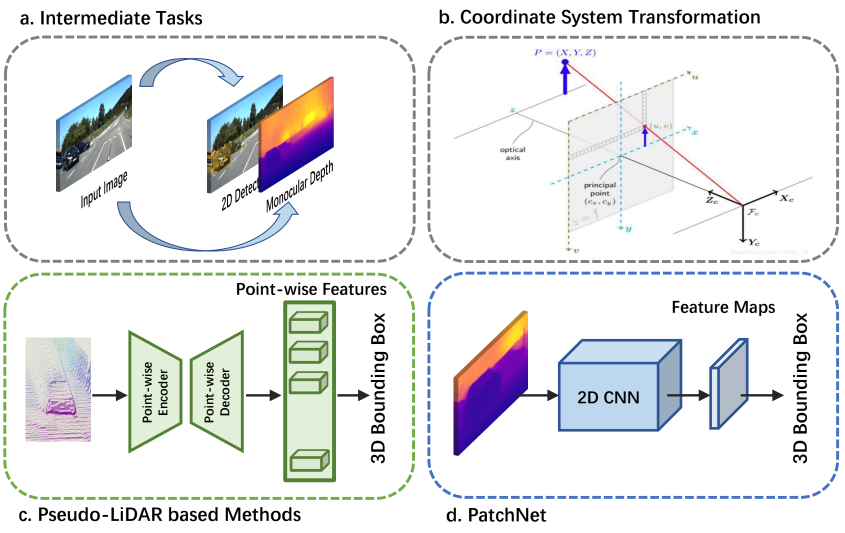


Figure 5.1: Comparison of pseudo-LiDAR based methods [121, 192] and PatchNet. They both generate intermediate tasks using off-the-shelf models (a) and project the image coordinates to the world coordinates (b). Pseudo-LiDAR-based methods treat these data as LiDAR signals and use point-wise networks to predict results from them (c). However, PatchNet organizes them as the image representation for subsequent processing (d).

coordinate system transformation. Second, we find that pseudo-LiDAR representation is not necessary to improve detection performance. After integrating spatial coordinates, image representation-based algorithms can also achieve competitive if not superior the same performance. Third, thanks to more powerful image-based deep learning technologies, we achieve state-of-the-art performance and show the potential of image representation-based 3D detectors.

5.2 Delving into pseudo-LiDAR representation

In this section, we investigate the influence of pseudo-LiDAR representation on 3D detection accuracy. In particular, we first give a brief review of pseudo-LiDAR-based detectors and introduce the technical details of its image-based equivalent detector. Then, we analyze whether data representation is the internal reason for performance improvement by comparing the performance of these two detectors.

5.2.1 Review of pseudo-LiDAR based detectors

Here we take pseudo-LiDAR [192] as an example for analysis, and the paradigm of [192] can be summarized as follows:

Step 1: Depth estimation. Given a single monocular image (or stereo pairs) as input, [192] predict the depth d for each image pixel (u, v) using a stand-alone CNN (Fig 7.1(a)).

Step 2: 2D detection. Another CNN is adopted to generate 2D object region proposals (Fig 7.1(a)).

Step 3: 3D data generation. First, regions of interest (RoIs) are cropped from the depth map generated from Step 1, according to the region proposals generated from Step 2. Then, the 3D coordinates of pixels of each RoI can be recovered by:

$$\begin{cases} z = d, \\ x = (u - C_x) \times z/f, \\ y = (v - C_y) \times z/f, \end{cases} \tag{5.1}$$

where f is the focal length of the camera, (C_x, C_y) is the principal point (Fig 7.1(b)).

Step 4: 3D object detection. Pseudo-LiDAR-based approaches treat the 3D data generated from Step 3 as LiDAR signals and use point-wise CNN to predict results from them (Fig 7.1(c)). In particular, they are treated as an unordered point set $\{x_1, x_2, ..., x_n\}$ with $x_i \in \mathbb{R}^d$, and processed by PointNet, which defines a set function f that maps a set of points to an output vector:

$$f(x_1, x_2, ..., x_n) = \gamma \left(\underset{i=1,...,n}{\mathbf{MAX}} \{h(x_i)\} \right) \tag{5.2}$$

where γ and h are implemented by multi-layer perceptron (MLP) layers.

5.2.2 PatchNet-vanilla: equivalent implementation of pseudo-LiDAR

Analysis. The most significant difference between the pseudo-LiDAR-based methods [121, 192] and other approaches lies in the representation of depth map. The authors of [121, 192] argue that pseudo-LiDAR representation is more suitable for describing the 3D structure of objects, which is the main reason behind the high accuracy of their models. To verify this, we conduct an image representation-based detector, *i.e.* PatchNet-vanilla, which is identical to pseudo-LiDAR [192] except for the input representation.

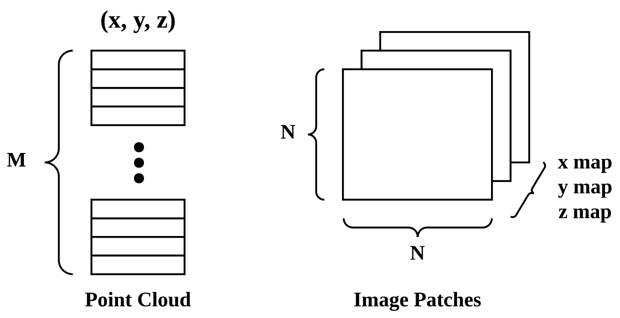


Figure 5.2: Illustration of input data. Pseudo-LiDAR based approaches use point cloud (*left*) as input, while PatchNet use image patches (*right*) as input. We set $M = N \times N$ so that these two kinds of input data contain the same amount of information.

Implementation. Steps 1, 2, and 3 in PatchNet-vanilla are the same as that in the pseudo-LiDAR-based detectors. Therefore, they have the same estimated depth, 2D detection results, and generated 3D data. The main difference is Step 4, which will be analyzed in detail. Specifically, in PatchNet-vanilla, the generated 3D data is organized as image representation (see Fig 5.2), where each pixel location with 3 channels, i.e. (x, y, z) in Eq. 5.1. Different from point-wise CNN used in pseudo-LiDAR counterparts, 2D CNN is used for processing the input data in PatchNet-vanilla (Fig 7.1(d)). Note that we can define the same function as Eq. 5.2 using 2D convolution with 1×1 receptive field and global max pooling. This scheme is also adopted in the official implementation[1] of PointNet.

5.2.3 Preliminary conclusion

Method	Modality	3D detection			BEV detection		
		Easy	Moderate	Hard	Easy	Moderate	Hard
pseudo-LiDAR	pseudo-LiDAR	28.2	18.5	16.4	40.6	26.3	22.9
pseudo-LiDAR*	pseudo-LiDAR	28.9	18.4	16.2	41.0	26.2	22.8
PatchNet-vanilla	image	28.7	18.4	16.4	40.8	26.1	22.8

Table 5.1: Comparison of different input representations. Experiments are conducted on the KITTI-3D *validation* set. $*$ indicates the method is reproduced by ourselves. Metric is $AP|_{R_{11}}$ of the **Car** category.

The performances of PatchNet-vanilla and pseudo-LiDAR are reported in

[1]https://github.com/charlesq34/pointnet.

Tab. 5.1, where we reproduce pseudo-LiDAR to eliminate the impact of implementation details. As can be seen, PatchNet-vanilla achieves almost the same accuracy as pseudo-LiDAR, which means the choice of data representation has no substantial impact on 3D detection tasks. Moreover, we perform ablation studies on data content and observe that coordinate transform is the key factor for performance improvement (experimental results and analysis can be found in Sec. 5.4.2).

The above observations reveal that pseudo-LiDAR representation is not necessary, and after integrating the generated 3D information, image representation has the same potential. More importantly, compared with point-wise CNNs [147, 148], image-based representation can utilize the well-studied 2D CNNs for developing high-performance 3D detectors. Along this direction, we show how the proposed PatchNet framework is used to further improve the detection performance in Sec. 5.3.

5.3 PatchNet

In PatchNet, we first train two deep CNNs on two intermediate prediction tasks (i.e., 2D detection and depth estimation) to obtain position and depth information, which are the same as PatchNet-vanilla and pseudo-LiDAR based detectors (Fig 7.1(a)). Then, as shown in Fig 5.3, for each detected 2D object proposal, we crop the corresponding region from the depth map and recover its spatial information using Eq 5.1. Next, deep features of RoIs are extracted by the backbone network, and filtered by the mask global pooling and foreground mask. Finally, we use a detection head with a difficulty assignment mechanism to predict the 3D bounding box parameterized by $(x, y, z, h, w, l, \theta)$.

Backbone. Most of the existing backbone networks can be used in our method to extract image features. In our implementation, we use the ResNet-18 [64] with Squeeze-and-Excitation (SE) block [70] as the 3D detection backbone. Moreover, we remove all pooling layers in the original SE-ResNet-18 so that its output features have the same size as input image patches. Then we use the masked global pooling operation to extract features from the foreground object.

Mask global pooling. The feature maps $\mathbf{X}$ output from the backbone network will be converted to a feature vector by global pooling. Conventional global pooling takes features of all positions into account and outputs the global feature. To obtain more robust features, we perform global pooling only on those features within foreground regions so that the final feature is corresponding to

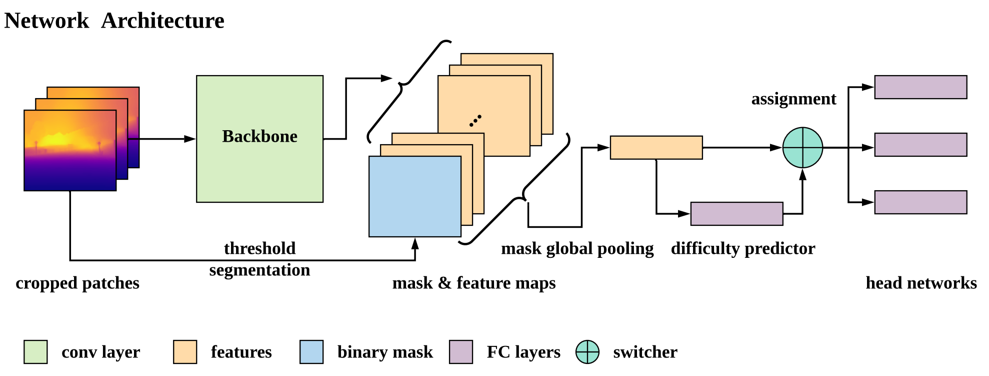


Figure 5.3: Illustration of the network architecture. Given an input patch with $\{x, y, z\}$ channels, we first generate a binary mask according to mean depth and use it to guide the pooling layer to extract the features corresponding to the foreground object. Then, we assign examples to different head networks according to their prediction difficulty of them.

those pixels of interest. Specifically, we additionally generate a binary mask **M** which indicates the foreground region. These masks will be applied to the feature maps **X** to select foreground features before global pooling. Such a mask global pooling encourages the final feature to focus on the regions of interest.

Mask generation. Following the prior work [121], the fore/background binary mask **M** is obtained by setting a threshold to the depth map. Specifically, we empirically add an offset on the mean depth of each patch and set it as the threshold. The regions with depth values smaller than this threshold will be regarded as foreground regions. The binary mask **M** has the same resolution as the input image with its values corresponding to foreground regions set as 1 and otherwise 0.

Head. Inspired by difficulty-wise evaluation adopted by the KITTI-3D dataset, we use three branches to deal with samples of different difficulty levels separately. To select the branch, we need a specific module. Specifically, before sending the feature maps to the three parallel box estimators, we add another branch to predict the difficulty level of each instance.

Note that all three branches are the same in network architecture, and only different in learned parameters for handling different difficulty levels. Besides, in our implementation, all three branches predict results simultaneously, and two of them are blocked according to the output of the difficulty predictor. Theoretically, this does not affect the accuracy of the algorithm and allows all branches to run in parallel with the cost of extra GPU memory.

Loss function. The ground truth box is parameterized by center (x, y, z), size (w, h, l), and heading angle θ. We adopted the loss function proposed by [146] to our baseline model:

$$\mathcal{L} = \mathcal{L}_{center} + \mathcal{L}_{size} + \mathcal{L}_{heading} + \lambda \cdot \mathcal{L}_{corner} \tag{5.3}$$

where $\mathcal{L}_{center}, \mathcal{L}_{size},$ and $\mathcal{L}_{heading}$ respectively denote the loss function for the center, size, and heading angle. λ is an empirical weight, and $\mathcal{L}_{corner}$ is used to alleviate the potential sub-optimal problem. Please refer to [146] for details.

5.4 Experiments

5.4.1 Setup

Dataset. We evaluate our approach on the challenging KITTI-3D dataset [52], which provides 7,481 images for training and 7,518 images for testing. Detection and localization (*i.e.* bird's-eye-view detection) tasks are evaluated in three different subsets: *easy*, *moderate* and *hard*, according to the occlusion and truncation levels of objects. Since the ground truth for the test set is not available and the access to the test server is limited, we follow the protocol of prior works [27, 26, 29] to divide the training data into a training set (3,712 images) and a validation set (3,769 images). We will conduct ablation studies based on this split and also report the final results on the testing set provided by the KITTI-3D server. Due to space limitations, we mainly report the **Car** detection results of **monocular images** in the main paper. The results of **stereo pairs** and **Pedestrian/Cyclist** are also provided for reference.

Metric. Most of the previous works use 11-point interpolated average precision (IAP) metric [52] as follows:

$$AP|_{R_{11}} = \frac{1}{11} \sum_{r \in R_{11}} \max_{\tilde{r} \geq r} \rho(\tilde{r}). \tag{5.4}$$

Recently, to avoid an ostensible boost in performance, KITTI-3D and [171] call for a new 40-point IAP ($AP|_{R_{40}}$) with the exclusion of "0" and four-times denser interpolated prediction for better approximation of the area under the Precision/Recall curve. For fair and comprehensive comparisons with previous and future works, we show both $AP|_{R_{11}}$ and $AP|_{R_{40}}$ in the following experiments.

Method	Modality	3D detection			BEV detection		
		Easy	Moderate	Hard	Easy	Moderate	Hard
pseudo-LiDAR	PL	28.2	18.5	16.4	40.6	26.3	22.9
pseudo-LiDAR*	PL	28.9	18.4	16.2	41.0	26.2	22.8
AM3D	PL	32.2	21.1	17.3	43.8	28.4	23.9
PatchNet-vanilla	image	28.7	18.4	16.4	40.8	26.1	22.8
PatchNet-AM3D	image	32.8	20.9	17.3	43.5	28.2	23.6
PatchNet	image	35.1	22.0	19.6	44.4	29.1	24.1
Improvement	-	+2.9	+0.9	+2.3	+0.6	+0.7	+0.2

Table 5.2: 3D object detection results on KITTI-3D *validation* set. 'PL' denotes 'pseudo-LiDAR' representation. Metrics are AP_{3D} and AP_{BEV} of the **Car** category with 11 recall positions. $*$ indicates the method is reproduced by ourselves.

5.4.2 Investigation of pseudo-LiDAR representation

Analysis of data representation. As shown in Tab. 5.2, PatchNet-vanilla shows comparable results with pseudo-LiDAR, which indicates that *data representation is not the key factor to improve the performance of 3D detectors.* To further validate this claim, we also adjust our image representation-based detector based on AM3D, where we achieve a matched performance again.

input	AP_{3D}			AP_{BEV}		
	Easy	Moderate	Hard	Easy	Moderate	Hard
$\{z\}$	4.51	3.48	3.03	6.31	4.50	3.98
$\{x, z\}$	27.1	18.3	15.8	35.9	23.4	18.3
$\{x, y, z\}$	35.1	22.0	19.6	44.4	29.1	24.1
$\{u, v, z\}$	24.6	15.7	14.6	33.2	21.3	16.7

Table 5.3: Comparison of different input channels on KITTI-3D *validation* set. Metrics are AP_{3D} and AP_{BEV} of the **Car** category with 11 recall positions.

Analysis of data content. We conduct an ablation study on the effect of input channels and report the results in Tab. 5.3. We can see from the results that, using only depth as an input, it is almost impossible to obtain accurate 3D bounding boxes. If other coordinates are used, the accuracy of predicted boxes improves greatly, which validates the importance of generated spatial features. It should be noted that in the absence of y-axis data, this detection accuracy is much worse than our full model. This shows that all coordinates are useful for 3D detection.

In pseudo-LiDAR, the coordinate (u, v) for images is projected to the world coordinate (x, y) using the camera information. Experimental results in Tab. 5.3 also compare the effectiveness of different coordinate systems. According to experimental results, world coordinates (x, y), which utilize the camera information, perform much better than image coordinates (u, v). Through the above experiments, we can observe that *that the real thing matters is coordinate system transformation, instead of data representation itself.*

5.4.3 Boosting the performance of PatchNet

Backbone. Compared with point-wise backbone nets commonly used in the (pseudo) LiDAR-based methods, standard 2D backbones such as [64, 70, 198] can extract more discriminative features, which is a natural advantage of image-based detectors. We investigate the impact of different backbones on the proposed PatchNet, and the experimental results are summarized in Tab. 5.4 (*left*). The original PointNet has only 8 layers. For fair comparison, we construct a PointNet with 18 layers, which is denoted by PointNet-18 in Tab. 5.4. Compared with PointNet-18, using 2D convolution backbones can improve the accuracy of 3D boxes, especially for *hard* setting. This is because these cases are usually occluded/truncated or far away from the camera, and estimating their pose of them is more dependent on context information. However, it is evident that point-wise CNNs are hard to extract local features of data efficiently. From this perspective, image representation-based detectors have greater development potential. Besides, we can see from Tab. 5.4 (*right*) that the accuracy does not improve much when the CNN has more layers from ResNeXt-18 to ResNeXt-50. Compared with ResNeXt-50, ResNeXt-101 performs worse, which can be attributed to over-fitting. All the CNNs are trained from scratch.

Backbone	Easy	Moderate	Hard
PointNet-18	31.1	20.5	17.0
ResNet-18	33.2	21.3	19.1
ResNeXt-18	33.4	21.2	19.2
SE-ResNet-18	33.7	21.5	19.2
ResNeXt-18	32.7	21.2	19.2
ResNeXt-50	32.9	21.4	17.3
ResNeXt-101	31.1	20.9	17.0

Table 5.4: Comparisons of different backbone nets on KITTI-3D *validation* set. Metrics are $AP_{3D}|_{R_{11}}$ for 3D detection task of the **Car** category with IoU threshold = 0.7. Other settings are the same as PatchNet-vanilla.

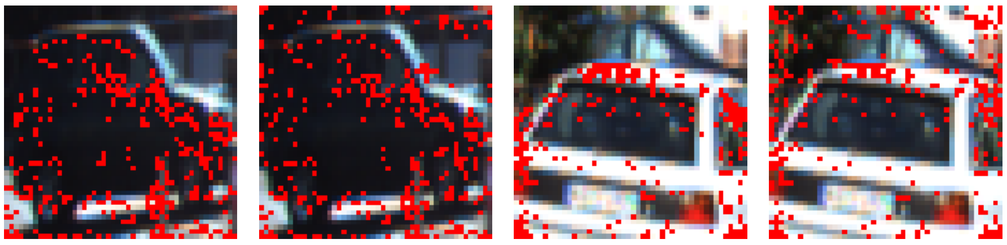


Figure 5.4: Qualitative comparison of max global pooling on KITTI-3D *validation* set. The left/right image in each image pair marks the units activated by mask/standard global pooling.

Mask global pooling. In the PatchNet, we design the mask global pooling operation to force the feature maps must be extracted from a set of pixels of interests, which can be regarded as a hard attention mechanism. Tab. 5.5 shows the effectiveness of this operation, *e.g.* mask global pooling (max) can improve $AP_{3D}|_{11}$ by 1.4% for moderate setting by and 2.7% for easy setting, and max pooling is slightly better than avg pooling. Besides, the visualization result shown in Fig. 5.5 intuitively explains the reason for the performance improvement. Specifically, most activation units filtered by mask global pooling correspond to foreground goals, while the ones from standard global max pooling will have many activation units in the background.

pooling	type	AP_{3D}			AP_{BEV}		
		Easy	Moderate	Hard	Easy	Moderate	Hard
standard	max	32.4	20.6	17.7	41.3	27.0	21.6
mask	avg	34.6	21.6	19.3	43.5	28.7	23.3
mask	max	35.1	22.0	19.6	44.4	29.1	24.1

Table 5.5: Ablation study of mask global pooling on KITTI-3D *validation* set. Metrics are AP_{3D} and AP_{BEV} of the **Car** category with 11 recall positions. Other settings are the same as PatchNet (full model).

Instance assignment. We use a stand-alone module to predict the 'difficulty' of each instance and assign it to its corresponding head network. Tab. 5.6 shows the ablation study of this mechanism. First, we can find that the accuracy of outputs increases with instance assignment. Interestingly, considering that not in all cases we can get the annotations of 'difficulty', we use a simple alternative: using the distance from object to camera to represent the 'difficulty' of objects (our default setting), and the threshold used in this experiment is (30, 50). The experiment shows that this scheme gets a similar performance as predicted difficulty levels.

assignment	switcher	AP_{3D}			AP_{BEV}		
		Easy	Moderate	Hard	Easy	Moderate	Hard
-	-	33.7	21.5	19.2	42.5	28.2	23.5
✓	difficulty	34.7	22.1	19.5	44.1	29.0	24.2
✓	distance	35.1	22.0	19.6	44.4	29.1	24.1

Table 5.6: Ablation study of instance assignment on KITTI-3D *validation* set. Metrics are AP_{3D} and AP_{BEV} of the **Car** category with 11 recall positions.

5.4.4 Comparing with state-of-the-art methods

Method	testing ($AP\|_{40}$)			validation($AP\|_{40}$)			validation ($AP\|_{11}$)		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
OFTNet	1.61	1.32	1.00	-	-	-	4.07	3.27	3.29
FQNet	2.77	1.51	1.01	-	-	-	5.98	5.50	4.75
ROI-10D	4.32	2.02	1.46	-	-	-	10.25	6.39	6.18
GS3D	4.47	2.90	2.47	-	-	-	13.46	10.97	10.38
Shift R-CNN	6.88	3.87	2.83	-	-	-	13.84	11.29	11.08
Multi-Fusion	7.08	5.18	4.68	-	-	-	22.03	13.63	11.60
MonoGRNet	9.61	5.74	4.25	-	-	-	13.88	10.19	7.62
Decoupled-3D*	11.08	7.02	5.63	-	-	-	26.95	18.68	15.82
MonoPSR	10.76	7.25	5.85	-	-	-	12.75	11.48	8.59
MonoPL*	10.76	7.50	6.10	-	-	-	31.5	21.00	17.50
SS3D	10.78	7.68	6.51	-	-	-	14.52	13.15	11.85
MonoDIS	10.37	7.94	6.40	11.06	7.60	6.37	18.05	14.98	13.42
M3D-RPN	14.76	9.71	7.42	-	-	-	20.27	17.06	15.21
PL-AVOD*	-	-	-	-	-	-	19.5	17.2	16.2
PL-FPointNet*	-	-	-	-	-	-	28.2	18.5	16.4
AM3D*	**16.50**	10.74	9.52	28.31	15.76	12.24	32.23	21.09	17.26
PatchNet	15.68	**11.12**	**10.17**	**31.6**	**16.8**	**13.8**	**35.1**	**22.0**	**19.6**

Table 5.7: 3D detection performance of the **Car** category on KITTI-3D dataset. For *testing* set, only $AP|_{R_{40}}$ is provided by the official leaderboard. For *validation* set, we report both $AP|_{R_{40}}$ and $AP|_{R_{11}}$ for better comparisons. The ioU threshold is set to 0.7. $*$ indicates the method is based on pseudo-LiDAR data. Methods are ranked by *moderate* setting (same as the KITTI-3D leaderboard). We highlight the best results in **bold**.

As shown in Tab. 5.7, we report our 3D detection results on the car category on the KITTI-3D dataset, where the proposed PatchNet ranks 1st among all published methods (ranked by *moderate* setting). Overall, our method achieves superior results over other state-of-the-art methods across all settings except for *easy* level of *testing* set. For instance, we outperform the current state-of-the-art AM3D [121] by **0.65/1.56/2.34** under *hard* setting on the listed three metrics,

2D detection	Depth estimation	3D detection
60ms	400ms	28ms

Table 5.8: Runtime of PachNet-vanilla and pseudo-LiDAR.

Backbone	PointNet-18	ResNet-18	ResNeXt-18	SE-ResNet-18
runtime	12ms	23ms	18ms	26ms

Table 5.9: Runtime of PatchNet in 3D detection stage.

which is the most challenging cases in the KITTI-3D dataset. Besides, the proposed method outperforms existing pseudo-LiDAR-based approaches. Note we use the same depth estimator (DORN) as [121, 192, 210, 16] and the pipeline of the proposed method is much simpler than pseudo-LiDAR based counterparts [16, 192]. This shows the effectiveness of our design. We also observe that the proposed model lags behind AM3D [121] under the *easy* setting on *testing* set. This may be attributed to the differences between the 2D detectors. We emphasize that *easy* split contains the least number of examples, so the performance of this setting is prone to fluctuations. Also note that these three splits are containment relationships (e.g., *hard* split contains all instances belonging to *easy* and *moderate* setting).

5.5 Others

5.5.1 Runtime Analysis

In this section, we will analyze the latency of our PatchNet and compare it with some existing methods [121, 192, 195] based on pseudo-LiDAR representation. In general, all the four methods can be divided into three main stages. In our designs, the processing flows of PatchNet-vanilla and pseudo-LiDAR are the same, but the representations of inputs are different. So the runtime of these two methods is almost the same, which is shown as follows (tested on a single 1080 GPU):

PatchNet shares the same 2D detector and depth estimator (note the runtime of different depth estimators varies greatly, see **KITTI-3D Benchmark** for details), and we show its runtime of 3D detection stage for different backbone models as follows:

Method	3D Detection			BEV Detection		
	Easy	Mod.	Hard	Easy	Mod.	Hard
3DOP [27]	6.55	5.07	4.10	12.63	9.49	7.59
Multi-Fusion [199]	-	9.80	-	-	19.54	-
Stereo-RCNN [97]	54.1	36.7	31.1	68.5	48.3	41.5
Pseudo-LiDAR [192]	59.4	39.8	33.5	72.8	51.8	44.0
PatchNet-vanilla	60.8	40.1	33.6	72.7	51.2	43.8
PatchNet	**65.9**	**42.5**	**38.5**	**74.5**	**52.9**	**44.8**
PatchNet-vanilla@$AP\|_{R_{40}}$	61.4	37.6	31.6	73.5	49.8	41.7
PatchNet@$AP\|_{R_{40}}$	**66.0**	**41.1**	**34.6**	**76.8**	**52.8**	**44.3**

Table 5.10: Stereo 3D detection performance of the **Car** category on KITTI-3D *validation* dataset. The IoU threshold is set to 0.7. We highlight the best results in **bold**.

Although we add some extra operations in PatchNet, the runtime of the baseline model (PointNet-18) is 12ms while the runtime of PatchNet-vanilla is 28ms. This is mainly because we remove the foreground segmentation net and use a dynamic threshold to segment the foreground, which can save about 18ms. For the best backbone, the runtime is only 26ms, which has similar runtime of pseudo-LiDAR for 3d detection. Besides, although PatchNet and [121] use the same segmentation method, [121] add another ResNet-34 to extract image features. For [195], it adds a 2D instance segmentation net, which will bring lots of computing overhead (e.g., about 200ms for Mask RCNN [63]).

Overall, PatchNet is more efficient than [121, 195] and has a similar run time as [192].

5.5.2 Extension of Stereo-based Models

Pseudo-LiDAR representation is also widely used in the field of the stereo 3D detection task. In order to verify that the proposed method still works with binocular images, we replace the monocular depth maps with the stereo ones (we use PSMNet [20] as our stereo depth estimator and get the pre-trained model from [192]) and test the performance on KITTI-3D *validation* set using $AP|_{R_{11}}$ for better comparison with previous works. As shown in Tab. 5.10, PatchNet-vanilla has almost the same accuracy as pseudo-LiDAR, while PatchNet achieves better performances. We also report the $AP|_{R_{40}}$ for reference.

Method	Category	3D Detection			BEV Detection		
		Easy	Mod.	Hard	Easy	Mod.	Hard
Pseudo-LiDAR	Pedestrian	33.8	27.4	24.0	41.3	34.9	30.1
PatchNet-vanilla	Pedestrian	34.2	27.9	24.7	-	-	-
PatchNet	Pedestrian	**38.8**	**30.1**	**26.5**	-	-	-
Pseudo-LiDAR	Cyclist	41.3	25.2	24.9	34.9	29.9	27.0
PatchNet-vanilla	Cyclist	43.3	26.9	25.8	-	-	-
PatchNet	Cyclist	**46.8**	**29.0**	**26.8**	-	-	-

Table 5.11: 3D detection performance of the **Pedestrian/Cyclist** category on KITTI-3D *validation* dataset. The IoU threshold is set to 0.5. We highlight the best results in **bold**.

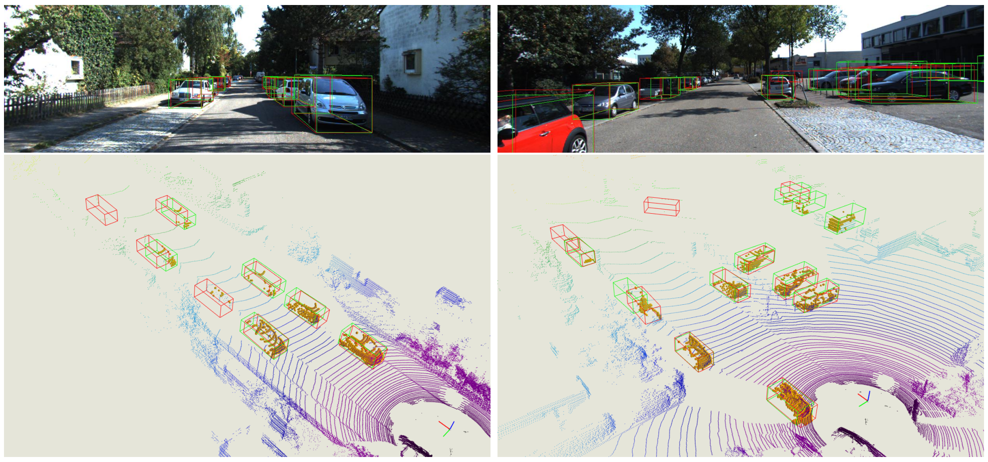


Figure 5.5: Qualitative results on KITTI-3D *validation* set. Red boxes represent our predictions, and green boxes come from the ground truth. LiDAR signals are only used for visualization. Best viewed in color with zoom-in.

5.5.3 Pedestrian and Cyclist

For better comparison, we also report **Pedestrian** and **Cyclist** detection performance on KITTI-3D *validation* set in this part. Specifically, we also use stereo images to conduct these experiments. It can be seen from Tab. 5.11 that these results also support our claims. Note that PatchNet-vanilla should have equal (or similar) performance with pseudo-LiDAR, while the former gets a higher AP on the Cyclist category. This may be caused by insufficient training in pseudo-LiDAR.

5.5.4 Qualitative results

We visualize some representative outputs of our PatchNet model in Fig. 7.7. We can observe that for simple cases at a reasonable distance, our model outputs remarkably accurate 3D bounding boxes. Relatively, for distant objects, our estimates of their size and heading angle are still accurate, although it is difficult to determine their center.

On the other hand, we do observe several failure patterns, which indicate possible directions for future efforts. First, our method often makes mistakes with truncated/occluded objects and often manifests itself as inaccurate heading estimates. Second, sometimes our 2D detector misses objects due to strong occlusion, which will cause these samples to be ignored in the subsequent 3D detection phase.

5.6 Conclusions

In this Chapter, a novel network architecture, namely PatchNet, is proposed to explore the fundamental cause of why pseudo-LiDAR representation-based 3D detectors achieve promising performance. Different from other works, we argue that the key factor is projecting the image coordinates to the world coordinates by the camera parameters, rather than the point cloud representation itself. More importantly, the world coordinate representation can be easily integrated into image representation, which means we can further boost the performance of 3D detectors using more flexible and mature 2D CNN technologies. Experimental results on the KITTI-3D dataset demonstrate our argument and show the potential of the image representation-based 3D detector. We hope these novel viewpoints provide insights to the monocular/stereo 3D object detection community, and promote the development of new 2D CNN designs for image-based 3D detection.

Chapter 6

Learning Spatial Features for Image-based Models

6.1 Introduction

3D object detection is an indispensable component for 3D scene perception, which has wide applications in the real world, such as autonomous driving and robotic navigation. Although the algorithms with stereo [97, 192, 30] or LiDAR sensors [146, 165, 164] show promising performances, the heavy dependence on the expensive equipment restricts the application of these algorithms. Accordingly, the methods based on the cheaper and more easy-to-deploy monocular cameras [199, 121, 122, 13, 41] show great potential and have attracted lots of attention.

As shown in Figure 6.1 (a), some prior works [13, 171, 31] directly estimate the 3D bounding boxes from monocular images. However, because of the lack of depth cues, it is extremely hard to accurately detect the objects in the 3D space, and the localization error is the major issue of these methods [122]. To mitigate this problem, an intuitive idea is to estimate the depth maps from RGB images, and then use them to augment the input data (Figure 6.1 (b)) [199, 41] or directly use them as the input data (Figure 6.1 (c)) [192, 121]. Although these two strategies have made significant improvements in performance, the drawbacks of them can not be ignored: (1) These methods generally use an off-the-shelf depth estimator to generate the depth maps, which introduce lots of computational costs (*e.g.* the most commonly used depth estimator [47] need about 400ms to process a standard KITTI image). (2) The depth estimator and detector are trained separately, which may lead to a sub-optimal optimization. Recently, [155] propose an end-to-end framework (Figure 6.1 (d)) for monocular 3D detection, which can also leverage depth estimator to provide depth cues.

Specifically, they introduce a sub-network to estimate the depth distribution and use it to enrich the RGB features. Although this model can be trained end-to-end and achieves better performance, it still suffers from the low inference speed (630ms per image), mainly caused by the depth estimator and the complicated network architecture. Note that well-designed monocular detectors, like [122, 217, 115], only take about 40ms per image.

In this Chapter, we aim to introduce the depth cues to the monocular 3D detectors without introducing any extra cost in the inference phase. Inspired by the knowledge distillation [67], which can transfer the learned knowledge from a well-trained CNN to another one without any changes in the model design, we propose that the spatial cues may also be transferred in this way from the LiDAR-based models. However, the main problem for this proposal is the difference in the feature representations used in these two kinds of methods (2D image features *vs.* 3D voxel features). To bridge this gap, we propose to project the LiDAR signals into the image plane and use the 2D CNN, instead of the commonly used 3D CNN or point-wise CNN, to train an 'image-version' LiDAR-based model. After this alignment, the knowledge distillation can be friendly applied to enrich the features of our monocular detector.

Based on the above-mentioned motivation and strategy, we propose the **distill**ation based **mono**cular 3D detector (**MonoDistill**): We first train a teacher net using the projected LiDAR maps (used as the ground truths of the depth estimator in previous works), and then train our monocular 3D detector under the guidance of the teacher net. We argue that, compared with previous works, the proposed method has the following two advantages: First, our method directly learns the spatial cues from the teacher net, instead of the estimated depth maps. This design performs better by avoiding information loss in the proxy task. Second, our method does not change the network architecture of the baseline model, and thus no extra computational cost is introduced.

Experimental results on the most commonly used KITTI-3D benchmark, where we rank the 1^{st} place among all monocular-based models by applying the proposed method on a simple baseline, demonstrate the effectiveness of our approach. Besides, we also conduct extensive ablation studies to present each design of our method in detail. More importantly, these experiments clearly illustrate the improvements are achieved by the introduction of spatial cues, instead of other unaccountable factors in CNN.

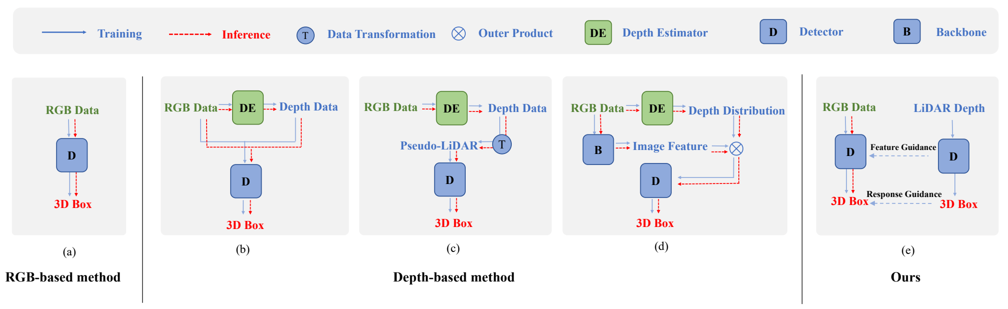


Figure 6.1: Comparison of the high-level paradigms of monocular 3D detectors.

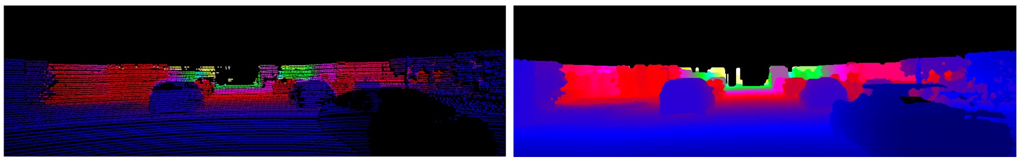


Figure 6.2: Visualization of the sparse LiDAR maps (*left*) and the dense LiDAR maps (*right*).

6.2 Method

6.2.1 Overview

Figure 6.3 presents the framework of the proposed MonoDistill, which mainly has three components: a monocular 3D detector, an aligned LiDAR-based detector, and several side branches which build the bridge to provide guidance from the LiDAR-based detector to our monocular 3D detector. We will introduce how to build these parts one by one in the rest of this section.

6.2.2 Baseline Model

Student model. We use the one-stage monocular 3D detector MonoDLE [122] as our baseline model. Particularly, this baseline model adopts DLA-34 [211] as the backbone and uses several parallel heads to predict the required items for 3D object detection. Due to this clean and compact design, this model achieves good performance with high efficiency. Besides, we further normalize the confidence of each predicted object using the estimated depth uncertainty, which brings about 1 AP improvement [1].

Teacher model. Existing LiDAR-based models are mainly based on the 3D CNN or point-wise CNN. To align the gap between the feature representations of the monocular detector and the LiDAR-based detector, we project the LiDAR

[1]Note that both the baseline model and the proposed method adopted the confidence normalization in the following experiments for a fair comparison.

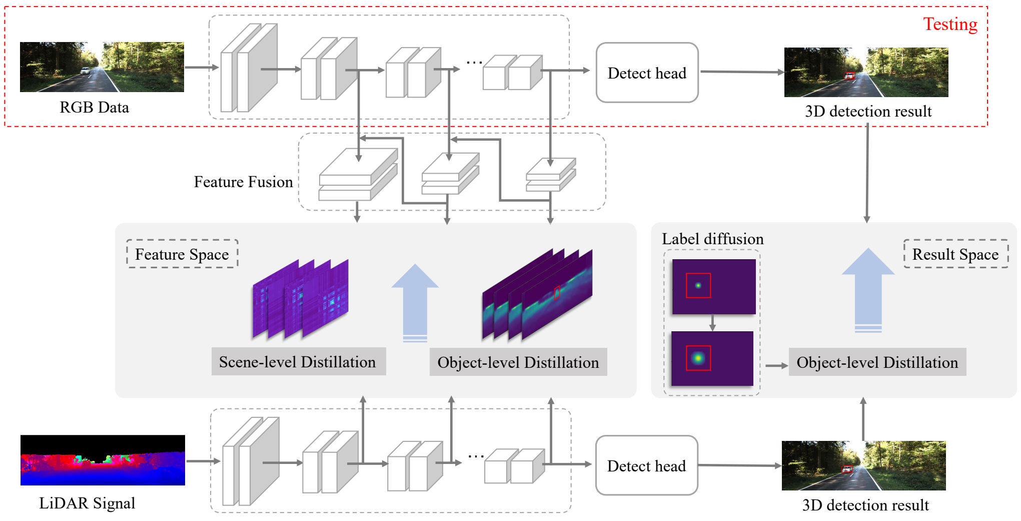


Figure 6.3: Illustration of the proposed MonoDistill. We first generate the 'image-like' LiDAR maps from the LiDAR signals and then train a teacher model using an identical network to the student model. Finally, we propose three distillation schemes to train the student model under the guidance of the well-trained teacher net. In the inference phase, only the student net is used.

points into the image plane to generate the sparse depth map. Further, we also use the interpolation algorithm [84] to generate the dense depth, and see Figure 6.2 for the visualization of generated data. Then, we use these 'image-like LiDAR maps' to train a LiDAR-based detector using the identical network with our student model.

Network architecture. The baseline network is extended from the anchor-free 2D object detection framework, which consists of a feature extraction network and seven detection subheads. We employ DLA-34 [211] without deformable convolutions as our backbone. The feature maps are downsampled by 4 times and then we take the image features as input and use 3x3 convolution, ReLU, and 1x1 convolution to output predictions for each detection head. Detection head branches include three for 2D components and four for 3D components. Specifically, 2D detection heads include the heatmap, offset between the 2D key point and the 2D box center, and the size of the 2D box. 3D components include offset between the 2D key point and the projected 3D object center, depth, dimensions, and orientations. As for objective functions, we train the heatmap with focal loss. The other loss items adopt L1 losses except for depth and orientation. The depth branch employs a modified L1 loss with the assistance of heteroscedastic aleatoric uncertainty. Common *MultiBin* loss is used for the orientation branch. Besides, we propose a strategy to improve the accuracy of the baseline. Inspire by [115], estimated depth uncertainty can provide confidence for each projection depth. Therefore, we normalize the confidence of each predicted box using depth uncertainty. In this way, the score has the capability of indicating the uncertainty of depth.

6.2.3 MonoDistill

In order to transfer the spatial cues from the well-trained teacher model to the student model, we design three complementary distillation schemes to provide additional guidance to the baseline model.

Scene-level distillation in the feature space. First, we think directly enforcing the image-based model learning the feature representations of the LiDAR-based models is sub-optimal, caused by the different modalities. The scene-level knowledge can help the monocular 3D detectors build a high-level understanding of the given image by encoding the relative relations of the features, keeping the knowledge structure, and alleviating the modality gap. Therefore, we train our student model under the guidance of the high-level semantic features provided by the backbone of the teacher model. To better model the structured cues, we choose to learn the affinity map [69] of high-level features, instead of the features themselves. Specifically, we first generate the affinity map, which encodes the similarity of each feature vector pair, for both the teacher and student network, and each element $\mathrm{A}_{\mathbf{i},\mathbf{j}}$ in this affinity map can be computed by:

$$\mathrm{A}_{\mathbf{i},\mathbf{j}} = \frac{\mathbf{f}_{\mathbf{i}}^{\mathbf{T}}\mathbf{f}_{\mathbf{j}}}{||\mathbf{f}_{\mathbf{i}}||_2 \cdot ||\mathbf{f}_{\mathbf{j}}||_2}, \tag{6.1}$$

where $\mathbf{f}_{\mathbf{j}}$ and $\mathbf{f}_{\mathbf{j}}$ denote the $\mathbf{i}^{th}$ and $\mathbf{j}^{th}$ feature vector. After that, we use the L1 norm to enforce the student net to learn the structured information from the teacher net:

$$\mathcal{L}_{\mathbf{sf}} = \frac{1}{\mathrm{K}\times\mathrm{K}}\sum_{i=1}^{K}\sum_{j=1}^{K}||\mathrm{A}_{\mathbf{i},\mathbf{j}}^{\mathbf{t}} - \mathrm{A}_{\mathbf{i},\mathbf{j}}^{\mathbf{s}}||_1, \tag{6.2}$$

where K is the number of the feature vectors. Note the computational/storage complexity is quadratically related to K. To reduce the cost, we group all features into several local regions and generate the affinity map using the features of local regions. This makes the training of the proposed model more efficient, and we did not observe any performance drop caused by this strategy.

Object-level distillation in the feature space. Second, except for the affinity map, directly using the features from the teacher net as guidance may also provide valuable cues to the student. However, there is much noise in feature

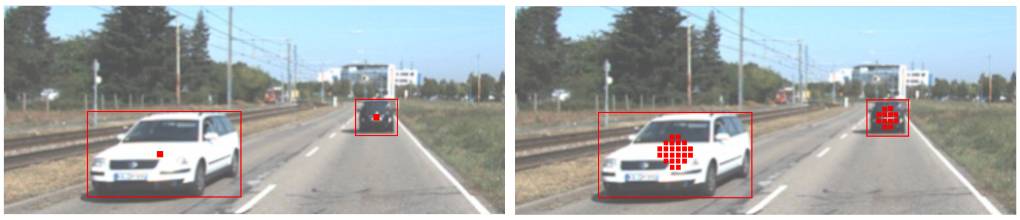


Figure 6.4: *Left:* Regard the center point as the foreground region. *Right:* Generate the foreground region from the center point and the size of the bounding box. Besides, the 2D bounding boxes are used as the foreground region for $\mathcal{L}_{\mathbf{of}}$.

maps, since the background occupies most of the area and is less informative. Distilling knowledge from these regions may make the network deviate from the right optimization direction. To make the knowledge distillation more focused, limiting the distillation area is necessary. Particularly, the regions in the ground-truth 2D bounding boxes are used for knowledge transfer to mitigate the effects of noise. Specifically, given the feature maps of the teacher model and student model $\{\mathbf{F^t}, \mathbf{F^s}\}$, our second distillation loss can be formulated as.

$$\mathcal{L}_{\mathbf{of}} = \frac{1}{\mathrm{N_{pos}}} ||\mathrm{M}_{\mathbf{of}}(\mathrm{F_s} - \mathrm{F_t})||_2^2, \tag{6.3}$$

where $\mathrm{M_{of}}$ is the mask generated from the center point and the size of the 2D bounding box and $\mathrm{N_{pos}}$ is the number of valid feature vectors.

Object-level distillation in the result space. Third, similar to the traditional KD, we use the predictions from the teacher net as extra 'soft label' for the student net. Note that in this scheme, only the predictions on the foreground region should be used, because the predictions on the background region are usually false detection. As for the definition of the 'foreground regions', inspired by CenterNet [222], a simple baseline is regarding the center point as the foreground region. Furtherly, we find that the quality of the predicted value of the teacher net near the center point is good enough to guide the student net. Therefore, we generate a Gaussian-like mask [183, 189] based on the position of the center point and the size of 2D bounding box and the pixels whose response values surpass a predefined threshold are sampled, and then we train these samples with equal weights (see Figure 6.4 for the visualization). After that, our third distillation loss can be formulated as:

$$\mathcal{L}_{\mathbf{or}} = \sum_{k=1}^{N} ||\mathrm{M}_{\mathbf{or}}(\mathrm{y}_{\mathbf{k}}^{\mathbf{s}} - \mathrm{y}_{\mathbf{k}}^{\mathbf{t}})||_1, \tag{6.4}$$

where $\mathbf{M_{or}}$ is the mask which represents positive and negative samples, $\mathbf{y_k}$ is the output of the k^{th} detection head and N is the number of detection heads.

Additional strategies. We further propose some strategies for our method. First, for the distillation schemes in the feature space (*i.e.* $\mathcal{L}_{\mathbf{sf}}$ and $\mathcal{L}_{\mathbf{of}}$), we only perform them on the last three blocks of the backbone. The main motivation for this strategy is: The first block usually is rich in low-level features (such as edges, textures, etc.). The expression forms of the low-level features for LiDAR and image data may be completely different, and enforcing the student net to learn these features in a modality-across manner may mislead it. Second, in order to better guide the student to learn spatial-aware feature representations, we apply the attention-based fusion module (FF in Table 6.1) proposed by [25] in our distillation schemes in the feature space (i.e. $\mathcal{L}_{\mathbf{sf}}$ and $\mathcal{L}_{\mathbf{of}}$).

Loss function. We train our model in an end-to-end manner using the following loss function:

$$\mathcal{L} = \mathcal{L}_{\mathbf{src}} + \lambda_1 \cdot \mathcal{L}_{\mathbf{sf}} + \lambda_2 \cdot \mathcal{L}_{\mathbf{of}} + \lambda_3 \cdot \mathcal{L}_{\mathbf{or}}, \tag{6.5}$$

where $\mathcal{L}_{\mathbf{src}}$ denotes the loss function used in the base model (MonoDLE [122] in default). $\lambda_1, \lambda_2, \lambda_3$ are the hyper-parameters to balance each loss. For the teacher net, only $\mathcal{L}_{\mathbf{src}}$ is adopted.

6.3 Experiments

6.3.1 Setup

Dataset and metrics. We conduct our experiments on the KITTI-3D [52], which is the most commonly used dataset in 3D detection. Specifically, this dataset provides 7,481 training samples and 7,518 testing samples, and we further divide the training data into a train set (3,712 samples) and a validation set (3,769 samples), following prior works [27, 26, 29]. Both 3D detection and Bird's Eye View (BEV) detection are evaluated using $\mathrm{AP}|_{R_{40}}$ [171] as the metric. We report our final results on the *testing* set, while the ablation studies are conducted on the *validation* set. Besides, we mainly focus on the **Car** category, while also presenting the performances of **Pedestrian** and **Cyclist** for reference.

Training details. Our model is trained on 2 NVIDIA 1080Ti GPUs in an end-to-end manner for 150 epochs. We employ the common Adam optimizer with an initial learning rate of $1.25e^{-4}$ and decay it by ten times at 90 and 120 epochs. To stabilize the training process, we also applied the warm-up strategy (5 epochs). As for data augmentations, only random flip and center crop is applied. Same to the common knowledge distillation scheme, we first train the teacher network in advance and then fix the teacher network. As for the student network, we simply train the detection model to give a suitable initialization. We implemented our method using PyTorch. And our code is based on [122].

	SF	OF	OR	FF	3D@IOU=0.7			BEV@IOU=0.7		
					Mod.	Easy	Hard	Mod.	Easy	Hard
a.					15.13	19.29	12.78	20.24	26.47	18.29
b.	✓				16.96	21.99	14.42	22.79	29.76	19.78
c.		✓			16.85	21.76	14.36	22.30	28.93	19.31
d.			✓		17.24	21.63	14.71	23.47	30.52	20.33
e.	✓	✓			17.33	22.34	14.63	22.90	30.02	19.84
f.	✓		✓		17.70	22.59	15.17	23.59	31.07	20.46
g.		✓	✓		17.98	22.58	15.26	23.76	30.98	20.52
h.	✓	✓	✓		18.24	23.82	15.49	25.06	32.66	21.88
i.	✓	✓	✓	✓	**18.47**	**24.31**	**15.76**	**25.40**	**33.09**	**22.16**

Table 6.1: Ablation studies on the KITTI-3D *validation* set. SF, OF, and OR denote the scene-level distillation in feature space, the object-level distillation in feature space, and the object-level distillation in result space, respectively. Besides, FF means the attention-based feature fusion strategy.

6.3.2 Main Results

Ablation studies. Table 6.1 shows the ablation studies of the proposed methods. Specifically, we found that all three distillation schemes can improve the accuracy of the baseline model, and the improvements of them are complimentary. Besides, the feature fusion strategy can also boost accuracy. Compared with the baseline, our full model improves 3D detection performance by **3.34**, **5.02**, **2.98** and improve BEV performance by **5.16**, **6.62**, **3.87** on the moderate, easy and hard settings respectively.

Guidance	Choice	3D@IOU=0.7			BEV@IOU=0.7		
		Mod.	Easy	Hard	Mod.	Easy	Hard
OF	full	16.13	21.52	14.18	22.04	27.85	19.04
	foreground	16.85	21.76	14.36	22.30	28.93	19.31
OR	sparse label	15.51	20.58	13.70	21.47	27.16	18.60
	diffused label	17.24	21.63	14.71	23.47	30.52	20.33

Table 6.2: Evaluation on the KITTI-3D *validation* set for detailed design choice. OF and OR represent the object-level distillation in the feature space and the object-level distillation in the result space.

Detailed design choice. We provide additional experiments in Table 6.2 for our method. First, as for object-level distillation in the feature space, we investigate the different effects of applying distillation on the whole image and foreground regions. Due to the noise in the background, guiding the foreground regions is more effective than the whole image, which improves the accuracy by 0.72 on the moderate settings in 3D detection. Second, as for object-level distillation in the result space, we compare the different effects of point labels and region labels. It can be observed that the generated region can significantly increase performance while guiding only in sparse point labels brings limited improvements. Our proposed label diffusion strategy can increase the number of positive samples for supervision, thus improving performance.

Method	3D@IOU=0.7			BEV@IOU=0.7			Runtime
	Mod.	Easy	Hard	Mod.	Easy	Hard	
M3D-RPN [13]	9.71	14.76	7.42	13.67	21.02	10.23	160 ms
SMOKE [113]	9.76	14.03	7.84	14.49	20.83	12.75	30 ms
MonoPair [31]	9.99	13.04	8.65	14.83	19.28	12.89	60 ms
RTM3D [99]	10.34	14.41	8.77	14.20	19.17	11.99	50 ms
AM3D* [121]	10.74	16.50	9.52	17.32	25.03	14.91	400 ms
PatchNet* [119]	11.12	15.68	10.17	16.86	22.97	14.97	400 ms
D4LCN* [41]	11.72	16.65	9.51	16.02	22.51	12.55	200 ms
MonoDLE† [122]	12.26	17.23	10.29	18.89	24.79	16.00	40 ms
MonoRUn* [22]	12.30	19.65	10.58	17.34	27.94	15.24	70 ms
G-NMS [88]	12.32	18.10	9.65	18.27	16.19	14.05	120 ms
DDMP-3D* [186]	12.78	19.71	9.80	17.89	28.08	13.44	180 ms
CaDDN* [155]	13.41	19.17	11.46	18.91	27.94	17.19	630 ms
MonoEF [224]	13.87	21.29	11.71	19.70	29.03	<u>17.26</u>	30 ms
MonoFlex [217]	13.89	19.94	<u>12.07</u>	19.75	28.23	16.89	30 ms
Autoshape [114]	14.17	<u>22.47</u>	11.36	<u>20.08</u>	<u>30.66</u>	15.59	50 ms
GUPNet [115]	<u>14.20</u>	20.11	11.77	-	-	-	35 ms
Ours*	**16.03**	**22.97**	**13.60**	**22.59**	**31.87**	**19.72**	40 ms
Improvements	+1.83	+0.50	+1.53	+2.51	+1.21	+2.46	-

Table 6.3: Comparison of state-of-the-art methods on the KITTI-3D *test* set. Methods are ranked by moderate setting. We highlight the best results in **bold** and the second place in <u>underlined</u>. Only RGB images are required as input in the inference phase for all listed methods. *: need dense depth maps or LiDAR signals for training. †: our baseline model without confidence normalization.

Comparison with state-of-the-art methods. Table 6.3 and Table 6.5 compare the proposed method with other state-of-the-art methods on the KITTI-3D *test* and *validation* sets. On the *test* set, the proposed method outperforms existing methods in all metrics. We note that, compared with the previous best results, we can obtain **1.83, 0.50, 1.53** improvements on the moderate, easy, and hard settings in 3D detection. Furthermore, our method achieves more significant improvements in BEV detection, increasing upon the prior work by **2.51, 1.21, 2.46** on the moderate, easy, and hard settings. Moreover, compared with the depth-based methods, our method outperforms them in performance by a margin and is superior to theirs in the inference speed. By contrast, our method only takes 40ms to process a KITTI-3D image, tested on a single NVIDIA GTX 1080Ti, while the Fastest of the depth-based methods [121, 119, 41, 155, 186] need 180ms. On the *validation* set, the proposed also performs best, both for the 0.7 IoU threshold and 0.5 IoU threshold. Besides, we also present the performance of the baseline model to better show the effectiveness of the proposed method. Note that we do not report the performances of some depth-based methods [121, 119, 41, 186] due to the data leakage problem [2].

6.3.3 More Discussions

What has the student model learned from the teacher model? To locate the source of improvement, we use the items predicted from the baseline model to replace that from our full model, and Table 6.4 summarizes the results of the cross-model evaluation. From these results, we can see that the teacher model provides effective guidance to the location estimation (b→f), and the improvement of the dimension part is also considerable (c→f). Relatively, the teacher model provides limited valuable cues to the classification and orientation part. This phenomenon suggests the proposed methods boost the performance of the baseline model mainly by introducing spatial-related information, which is consistent with our initial motivation.

	loc.	dim.	ori.	con.	3D@IOU=0.7			BEV@IOU=0.7		
					Mod.	Easy	Hard	Mod.	Easy	Hard
a.	B	B	B	B	15.13	19.29	12.78	20.24	26.47	18.29
b.	B	O	O	O	16.05	20.07	13.47	21.31	27.77	19.14
c.	O	B	O	O	17.91	22.87	15.29	25.09	32.78	21.93
d.	O	O	B	O	18.12	24.02	15.34	25.02	32.85	21.84
e.	O	O	O	B	18.41	24.27	15.55	24.98	32.78	21.81
f.	O	O	O	O	18.47	24.31	15.76	25.40	33.09	22.16

Table 6.4: Cross-model evaluation on the KITTI-3D *validation* set. We extract each required item (location, dimension, orientation, and confidence) from the baseline model (B) and the full model (O), and evaluate them in a cross-model manner.

[2]these methods use the depth estimator pre-trained on the KITTI Depth, which overlaps with the validation set of the KITTI-3D.

Method	3D@IOU=0.7			BEV@IOU=0.7			3D@IOU=0.5			BEV@IOU=0.5		
	Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard
M3D-RPN	11.07	14.53	8.65	15.62	20.85	11.88	35.94	48.53	28.59	39.60	53.35	31.76
MonoPair	12.30	16.28	10.42	18.17	24.12	15.76	42.39	55.38	37.99	47.63	61.06	41.92
MonoDLE†	13.66	17.45	11.68	19.33	24.97	17.01	43.42	55.41	37.81	46.87	60.73	41.89
GrooMeD-NMS	14.32	19.67	11.27	19.75	27.38	15.92	41.07	55.62	32.89	44.98	61.83	36.29
MonoRUn	14.65	20.02	12.61	-	-	-	43.39	59.71	38.44	-	-	-
GUPNet	16.46	22.76	13.72	22.94	31.07	19.75	42.33	57.62	37.59	47.06	61.78	40.88
MonoFlex	17.51	23.64	14.83	-	-	-	-	-	-	-	-	-
Baseline	15.13	19.29	12.78	20.24	26.47	18.29	43.54	57.43	39.22	48.49	63.56	42.81
Ours	**18.47**	**24.31**	**15.76**	**25.40**	**33.09**	**22.16**	**49.35**	**65.69**	**43.49**	**53.11**	**71.45**	**46.94**

Table 6.5: Performance of the Car category on the KITTI-3D *validation* set. We highlight the best results in **bold** and the second place in underlined. †: our baseline model without confidence normalization.

Is the effectiveness of our method related to the performance of the teacher model? An intuitive conjecture is the student can learn more if the teacher network has better performance. To explore this problem, we also use the sparse LiDAR maps to train a teacher net to provide guidance to the student model (see Figure 6.2 for the comparison of the sparse and dense data). As shown in Table 6.6, the performance of the teacher model trained from the sparse LiDAR maps is largely behind by that from dense LiDAR maps (drop to 22.05% from 42.45%, moderate setting), while both of them provides comparable benefits to the student model. Therefore, for our task, the performance of the teacher model is not directly related to performance improvement, while the more critical factor is whether the teacher network contains complementary information to the student network.

	Teacher Model			Student Model			Improvement		
	Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard
sparse maps	22.05	31.67	18.72	18.07	23.61	15.36	+2.94	+4.32	+2.58
dense maps	42.57	58.06	37.07	18.47	24.31	15.76	+3.34	+5.02	+2.98

Table 6.6: Performance of the student model under the guidance of different teacher models. Metric is the $AP|_{40}$ for the 3D detection task on the KITTI-3D *validation* set. We also show the performance improvements of the student model to the baseline model for better comparison.

Do we need depth estimation as an intermediate task? As shown in Figure 6.1, most previous methods choose to estimate the depth maps to provide depth information for monocular 3D detection (information flow: LiDAR data $\rightarrow$estimated depth map$\rightarrow$3D detector). Compared with this scheme, our method directly learns the depth cues from LiDAR-based methods (information flow: LiDAR data$\rightarrow$3D detector), avoiding the information loss in the depth estimation step. Here we quantitatively show the information loss in depth estimation using a simple experiment. Specifically, we use DORN [47] (same as most previous depth augmented methods) to generate the depth maps, and then use them to train the teacher net. Table 6.7 shows the results of this experiment. Note that, compared with setting c, the setting b's teacher net is trained from a larger training set (23,488 *vs.* 3,712) with ground-truth depth maps (ground truth depth maps vs. noisy depth maps). Nevertheless, this scheme still lags behind our original method, which means that there is serious information loss in monocular depth estimation.

Comparison with direct dense depth supervision. According to the ablation studies in Table 6.1, we can find that depth cues are the key factor to affect the performance of the monocular 3D models. However, dense depth supervision

	3D@IOU=0.7			BEV@IOU=0.7			AOS@IOU=0.7			2D@IOU=0.7		
	Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard
a.	15.13	19.29	12.78	20.24	26.47	18.29	90.95	97.46	83.02	92.18	98.37	85.05
b.	17.70	23.21	15.02	23.34	31.20	20.40	91.50	97.77	83.49	92.51	98.54	85.38
c.	18.47	24.31	15.76	25.40	33.09	22.16	91.67	97.88	83.59	92.71	98.58	85.56

Table 6.7: Comparison of using depth estimation as the intermediate task or not. Setting a. and c. denote the baseline model and our full model. Setting b. uses the depth maps generated from DORN [47] to train the teacher model. Experiments are conducted on the KITTI-3D *validation* set.

in the student model without KD may also introduce depth cues to the monocular 3D detectors. Here we conduct the control experiment by adding a new depth estimation branch, which is supervised by the dense LiDAR maps. Note that, this model is trained without KD. Table 6.8 compares the performances of the baseline model, the new control experiment, and the proposed method. From these results, we can get the following conclusions: (i) additional depth supervision can introduce the spatial cues to the models, thereby improving the overall performance; (ii) the proposed KD-based method significantly performs better than the baseline model and the new control experiment, which demonstrates the effectiveness of our method.

	3D@IOU=0.7			3D@IOU=0.5		
	Mod.	Easy	Hard	Mod.	Easy	Hard
Baseline	15.13	19.29	12.78	43.54	57.43	39.22
Baseline + depth supv.	17.05	21.85	14.54	46.19	60.42	41.88
Ours	18.47	24.31	15.76	49.35	65.69	43.49

Table 6.8: Comparison with direct dense depth supervision. Experiments are conducted on the KITTI-3D *validation* set.

6.3.4 More Experiments

Pedestrian/Cyclist detection. Due to the small sizes, non-rigid structures, and limited training samples, pedestrians and cyclists are much more challenging to detect than cars. We first report the detection results on *test* set in Table 6.9. It can be seen that our proposed method is also competitive with current state-of-the-art methods on the KITTI-3D *test* set, which increases 0.69 AP on the hard difficulty level of the pedestrian category. Note that, the accuracy of these difficult categories fluctuates greatly compared with Car detection due to insufficient training samples (see Table 6.10 for the details). Due to the access to the test server is limited, we conduct more experiments for pedestrian/cyclist on the *validation* set for general conclusions (we run the proposed method three times with different random seeds), and the experimental results are summarized in Table 6.11. According to these results, we can find that the proposed method can effectively boost the accuracy of the baseline model for pedestrian/cyclist detection.

Method	Pedestrian			Cyclist		
	Easy	Mod.	Hard	Easy	Mod.	Hard
M3D-RPN	4.92	3.48	2.94	0.94	0.65	0.47
D4LCN	4.55	3.42	2.83	2.45	1.67	1.36
MonoPair	10.02	6.68	5.53	3.79	2.21	1.83
MonoFlex	9.43	6.31	5.26	4.17	2.35	2.04
MonoDLE	9.64	6.55	5.44	4.59	2.66	2.45
CaDDN	**12.87**	8.14	6.76	**7.00**	**3.14**	**3.30**
DDMP-3D	4.93	3.55	3.01	4.18	2.50	2.32
AutoShape	5.46	3.74	3.03	5.99	3.06	2.70
Ours	12.79	**8.17**	**7.45**	5.53	2.81	2.40

Table 6.9: Performance of Pedestrian/Cyclist detection on the KITTI-3D *test* set. We highlight the best results in **bold** and the second place in underlined.

	cars	pedestrians	cyclists
# instances	14,357	2,207	734

Table 6.10: Training samples of each category on the KITTI-3D *training* set.

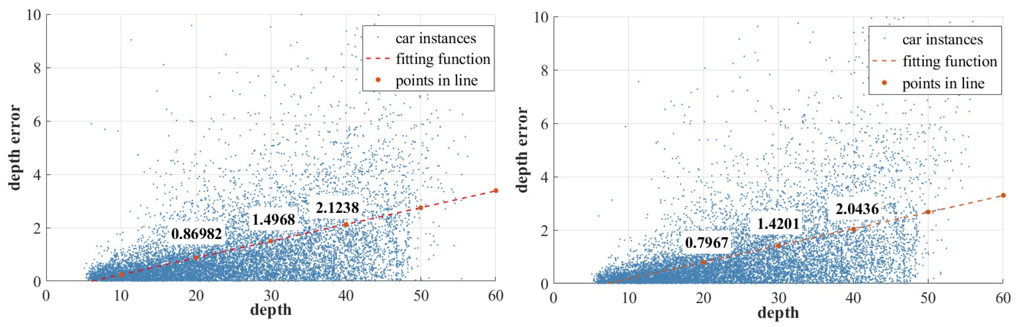


Figure 6.5: Errors of depth estimation. We show the errors of depth estimation as a function of the depth (x-axis) for the baseline model (*left*) and our full model (*right*).

Depth error analysis. As shown in Figure 6.5, we compare the depth error between the baseline and our method. Specifically, we project all valid samples of the Car category into the image plane to get the corresponding predicted depth values. Then we fit the depth errors between ground truths and predictions as a linear function by the least square method. According to the experimental

	Method	3D@IoU=0.25			3D@IoU=0.5		
		Easy	Mod.	Hard	Easy	Mod.	Hard
Pedestrian	Baseline	29.07±0.21	23.77±0.15	19.85±0.14	6.8±0.28	5.17±0.08	4.37±0.15
	Ours	32.09±0.71	25.53±0.55	21.15±0.79	8.95±1.26	6.84±0.81	5.32±0.75
Cyclist	Baseline	21.06±0.46	11.87±0.19	10.77±0.02	3.71±0.49	1.88±0.23	1.64±0.04
	Ours	24.26±1.29	13.04±0.44	12.08±0.68	5.38±0.91	2.67±0.40	2.53±0.38

Table 6.11: Performance of Pedestrian/Cyclist detection on the KITTI-3D *validation* set. Both 0.25 and 0.5 IoU thresholds are considered. We report the mean of several experiments for the proposed methods. ± captures the standard deviation over random seeds.

	3D@IOU=0.7			BEV@IOU=0.7			3D@IOU=0.5			BEV@IOU=0.5		
	Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard	Mod.	Easy	Hard
Baseline	15.13	19.29	12.78	20.24	26.47	18.29	43.54	57.43	39.22	48.49	63.56	42.81
Stereo Depth	18.18	23.54	15.42	24.89	32.26	21.64	49.13	65.18	43.29	52.88	69.47	46.72
LiDAR Depth	18.47	24.31	15.76	25.40	33.09	22.16	49.35	65.69	43.49	53.11	71.45	46.94

Table 6.12: Effects of sstereo/LiDAR guidance. Baseline denotes the baseline model without the guidance of the teacher network. Stereo Depth and LiDAR Depth denote under the guidance of stereo depth maps and LiDAR signals. Experiments are conducted on the KITTI-3D *validation* set.

results, we can find that our proposed method can boost the accuracy of depth estimation at different distances.

The effects of stereo depth. We also explored the changes in performance under the guidance of estimated stereo depth [20], and show the results in Table 6.12. Stereo depth estimation exploits geometric constraints in stereo images to obtain the absolute depth value through pixel-wise matching, which is more accurate compared with monocular depth estimation. Therefore, under the guidance of stereo depth, the model achieves almost the same accuracy as LiDAR signals guidance at 0.5 IoU threshold, and there is only a small performance drop at 0.7 IoU threshold.

Generalizing to other baseline models. To show the generalization ability of the proposed framework, we apply our method on another monocular detector GUPNet [115], which is a two-stage detection method. Experimental results are shown in Table 6.13. We can find that the proposed method can also boost the performance of GUPNet, which confirms the generalization of our method.

	3D@IOU=0.7			3D@IOU=0.5		
	Easy	Mod.	Hard	Easy	Mod.	Hard
GUPNet-Baseline	22.76	16.46	13.72	57.62	42.33	37.59
GUPNet-Ours	24.43	16.69	14.66	61.72	44.49	40.07

Table 6.13: MonoDistill with GUPNet. Experiments are conducted on the KITTI-3D *validation* set.

Generalizing to sparse LiDAR signals. We also explore the changes in performance under different resolutions of LiDAR signals. In particular, following Pseudo-LiDAR++ [210], we generate the simulated 32-beam/16-beam LiDAR signals and use them to train our teacher model (in the 'sparse' setting). We show the experimental results, based on MonoDLE, in Table 6.14. We can see that, although the improvement is slightly reduced due to the decrease of the resolution of LiDAR signals, the proposed method significantly boost the performances of the baseline model under all setting.

Besides, note that the camera parameters of the images on the KITTI-3D *test* set are different from these of the *training/validation* set, and the good performance on the *test* set suggests the proposed method can also generalize to different camera parameters. However, generalizing to the new scenes with different statistical characteristics is a hard task for existing 3D detectors [207, 193], including the image-based models and LiDAR-based models, and deserves further investigation by future works. We also argue that the proposed method can generalize to the new scenes better than other monocular models because our model learns stronger features from the teacher net.

	3D@IOU=0.7			3D@IOU=0.5		
	Mod.	Easy	Hard	Mod.	Easy	Hard
Baseline	19.29	15.13	12.78	43.54	57.43	39.22
Ours - 16-beam	22.49	17.66	15.08	49.39	65.45	43.60
Ours - 32-beam	23.24	17.71	15.19	49.41	65.61	43.46
Ours - 64-beam	23.61	18.07	15.36	49.67	65.97	43.74

Table 6.14: MonoDistill with sparser LiDAR signals. Experiments are conducted on the KITTI-3D *validation* set.

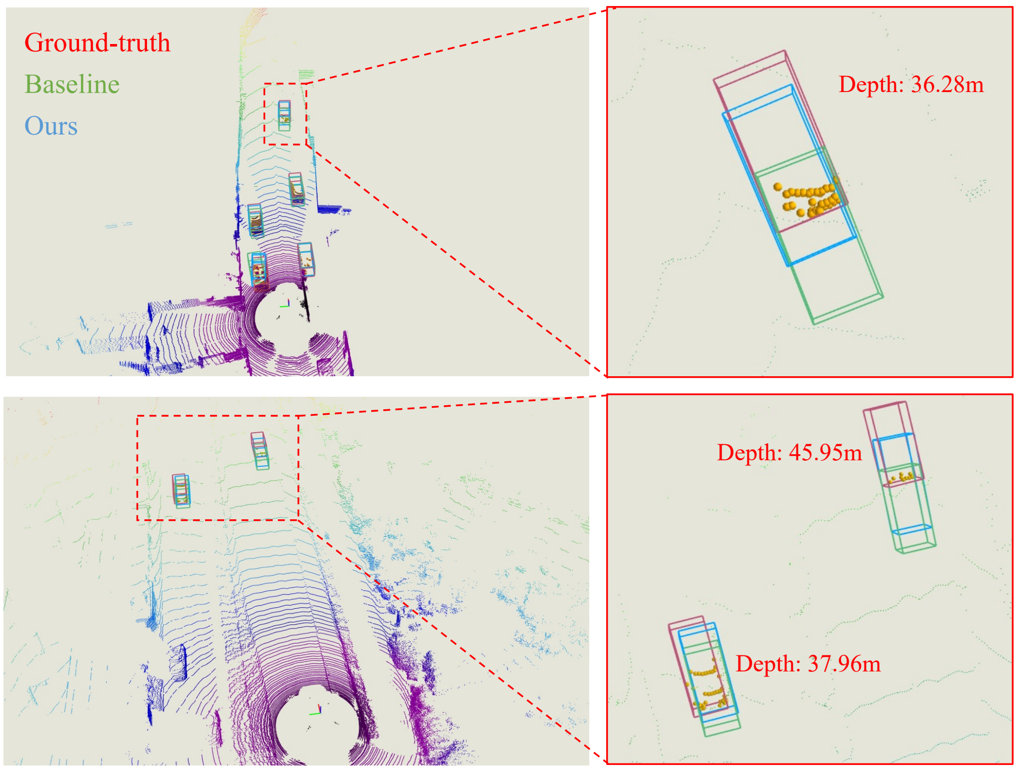


Figure 6.6: Qualitative results of our method for 3D space. The boxes' colors of ground truth, baseline, and ours are highlighted in red, green, and blue, respectively.

6.3.5 Qualitative Results

In Figure 6.7, we show the qualitative comparison of detection results. We can see that the proposed method shows better localization accuracy than the baseline model. Furthermore, in Figure 6.6, we show a comparison of detailed detection results in the 3D space. It can be found that our method can significantly

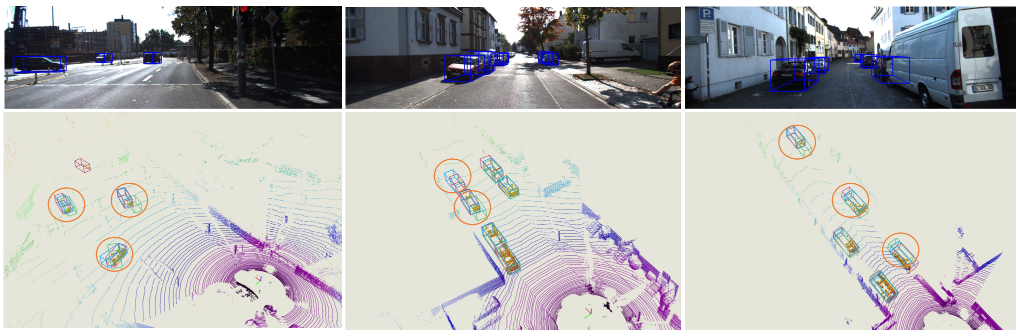


Figure 6.7: Qualitative results. We use green, blue, and red boxes to denote the results from baseline, our method, and ground truth. Besides, we use the red circle to highlight the main differences.

improve the accuracy of depth estimation compared with the baseline.

6.4 Conclusion

In this work, we propose the MonoDistill, which introduces spatial cues to the monocular 3D detector based on the knowledge distillation mechanism. Compared with previous schemes, which share the same motivation, our method avoids any modifications to the target model and directly learns the spatial features from the model rich in these features. This design makes the proposed method perform well in both performance and efficiency. To show an all-around display of our model, extensive experiments are conducted on the KITTI-3D dataset, where the proposed method ranks 1^{st} at 25 FPS among all monocular 3D detectors.

Chapter 7

Semi-Supervised Learning for Image-based 3D Detection

7.1 Introduction

As a crucial component of the self-driving system [52, 15, 180], 3D object detection has attracted extensive attention from both academia and industry. Especially, image-based 3D detection [120] has gradually become a hot problem in recent years. However, although lots of breakthroughs [199, 192, 121, 13, 171, 122, 155, 172, 136, 115, 119, 97, 31, 61, 30, 35, 224, 24, 210] have been made, the performance of the image-based methods still significantly lags behind that of LiDAR-based methods, such as [90, 164, 40, 29, 223, 165], and one of the main reasons for the unsatisfying performances of these methods is the limited training samples.

Unfortunately, although the raw data is relatively easy to collect, manually annotating the objects in the 3D space is a complicated and labor-consuming task. To seek a cheap alternative to the manually annotate labels, we investigate whether the pseudo-labels can provide effective supervision for the image-based 3D object detectors. Particularly, as shown in Figure 7.1, we adopt the following paradigm to train the image-based 3D object detectors: (i) train a teacher model with the annotated key-frames; (ii) generating the pseudo-labels for the unlabeled data using the well-trained teacher model; (iii) training the image-based 3D detectors with the resulting pseudo-labels. Fortunately, our exploratory experiments reveal that the pseudo-labels can play the role of supervisor well, which encourages us to investigate it under more settings and go deeper into this mechanism.

Specifically, we first adopt the LiDAR-based methods [164, 40] as the pseudo-label generators. We argue this choice is meaningful because the LiDAR sweeps

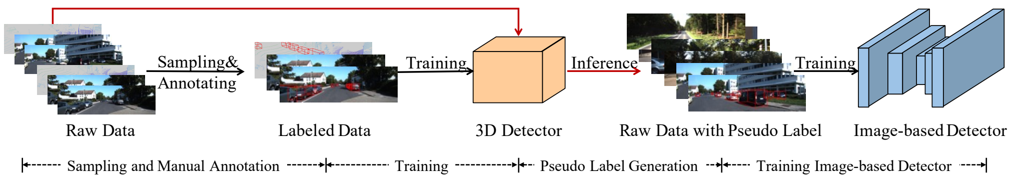


Figure 7.1: The proposed pipeline for training an image-based 3D detector. After collecting the raw data, the empirical practice is to sample some *key-frames* and annotate them to generate the training data. Based on these training samples, we further train the 3D detectors and generate pseudo-labels for the remaining unlabeled data. Finally, we use *all* frames to train our models.

are required in the annotation process to provide 3D coordinates of the objects (and applying LiDAR points in the training phase is a common practice for existing image-based models, such as [92, 155, 35, 61, 22, 86, 210, 30, 150]). By this way, we demonstrate that the pseudo-labels generated from LiDAR-based models perform well in image-based 3D detection task, and existing models can be further improved by introducing more training samples. More interestingly, based on the same training samples, the models trained with the pseudo-labels significantly outperform those trained with the manual annotations. This counter-intuitive result suggests the promising application potential of pseudo-labeling in the field of image-based 3D detection, and we provide the empirical interpretation of this phenomenon.

Besides, we apply the pseudo-labeling approach on varying settings, *e.g.* semi-supervised learning with few annotated samples, and significantly surpasses the current state of the art (SOTA) for most of them. Note that almost all the leading image-based models such as [155, 30, 35, 61] leverage LiDAR signals in their training phase, and we can build a fair environment to compare with them. The experimental results demonstrate our method is still superior to these works in performance. Furthermore, we also study whether the pseudo-label mechanism still works without LiDAR sweeps or better 3D detection models. Encouragingly, we find the models can also be effectively supervised by the knowledge they have learned before. In particular, take the monocular model GUPNet[115] as an example, we first train this model on the annotated frames and then generate the pseudo-labels for the interframe sequences. After that, our model can benefit from the enlarged training set.

In summary, we provide an empirical study of the pseudo-labeling mechanism for image-based 3D object detection. In this Chapter, we show that pseudo-labeling can significantly improve the performance of existing image-based 3D detectors under varying settings. This simple approach can cooperatively work with almost all current SOTA methods and then provide new baselines for this community. Besides, we provide extensive experiments on the KITTI-3D benchmark [52] to show the pseudo-labeling scheme in all-around, and the promising results firmly demonstrate the effectiveness of our method.

7.2 Approach

The objective of this Chapter is to study the pseudo-label mechanism [92] in image-based 3D detection. As shown in Figure 7.1, the whole pipeline can be clearly divided into two parts: generating pseudo-labels using teacher models and training the baseline models with the resulting pseudo-labels. Next, we detail the settings we used in this Chapter, including the pseudo-label generators, baseline models, datasets, etc.

Datasets and metrics. We conduct the experiments on the most commonly used KITTI-3D dataset [52], which provides 7,481 annotated frames for training and 7,518 frames for testing. Following [26, 29], we split the 7,481 training samples into a training set (3,712 frames) and a validation set (3,769 frames). Besides, we also use the KITTI raw data, which provides 155 video sequences, with about 48K unlabelled frames in total. Note that these frames include the ∼7K frames in KITTI-3D's training set, while the raw data for the testing set is not released. We further split the raw data into several sub-sets to evaluate the pseudo-labeling approach under varying settings. The summary of these splits is presented in Table 7.1. Besides, some sub-sets of these splits are used to show the performance changes *w.r.t.* the size of training data, which will be further explained in the corresponding experiment parts. As for the metrics, following [171], we evaluate the models with the $AP|_{R_{40}}$ for both 3D detection and Bird's Eye View (BEV) detection tasks. We mainly focus on the Car category, and both 0.7 and 0.5 IoU thresholds are considered. The performance of Pedestrian and Cyclist is also reported for reference.

	train	val	eigen	eigen-clean	all
# annotated	3,712	3,769	-	-	7,481
# total	13,596	10,670	23,488	14,940	47,937

Table 7.1: Summary of the data splits. We generate train/val split by collecting all the frames from the video sequences which correspond to the KITTI-3D's training/validation images. Eigen denotes Eigen's training set [42] which is commonly used in the KITTI Depth benchmark. Following [172], we generate the eigen-clean split by removing the images geometrically close to the images in the KITTI-3D's validation set.

Baseline models. To ensure generality and reproducibility, we choose some recently published methods with official codes as our baselines. In particular, we choose two monocular 3D detectors (*i.e.*, one-stage MonoDLE [122] and two-stage GUPNet [115]) which only need images and camera parameters in both training and inference phases as our baseline models. For the stereo setting, we use the LIGA-Stereo [61] in our experiments. Note that LiDAR points are required in the training phase for this baseline. Thus, we also build a pure stereo baseline by removing the relevant requirements (*i.e.*, depth loss on the cost-volume and the cross-modality knowledge distillation losses), marked as LIGA-Stereo in our experiments.

Pseudo-label generators. We first adopt two LiDAR-based models (PV R-CNN [164] and Voxel R-CNN [40]) to generate the pseudo-labels. Although this makes the LiDAR points involved in the training phase, we argue this strategy is meaningful, mainly based on the following two considerations:

- annotating objects in the 3D space requires accurate spatial information, which is usually provided by LiDAR points. Therefore, the training of the teacher models is hard to completely avoid the involvement of LiDAR points.
- almost all SOTA image-based 3D detectors adopt LiDAR points as supervision in the training phase, so we also test our method under this setting for a fair comparison.

Besides, to explore the monocular/stereo-only setting, we also use GUPNet and LIGA- Stereo as the pseudo-label generators. Note that using original LiDAR-Stereo to generate pseudo-labels is lack of practical significance, because training LIGA-Stereo also requires LiDAR points, and we can directly use LiDAR detectors to generate pseudo-labels in this setting.

7.3 Experiments and Analysis

7.3.1 Quality of the Generated Pseudo-Labels

The first problem is whether the pseudo-labels can be used as the training labels of image-based 3D detectors or not. If yes, what is the quality of the pseudo-labels compared with the official annotations? To investigate these problems, we first generate the pseudo-labels for the official training images and then compare the performances of the image-based 3D detectors trained from these pseudo-labels or official annotations.

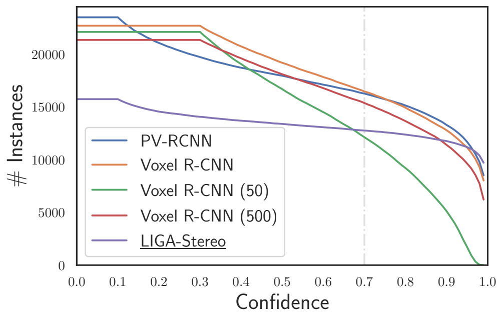
	MonoDLE			GUPNet			LIGA-Stereo			LiGA Stereo		
	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard	Easy	Mod.	Hard
PV R-CNN	19.32	14.96	13.45	23.24	17.37	15.58	-	-	-	-	-	-
Voxel R-CNN	20.34	15.78	13.82	24.99	18.10	16.22	77.21	58.67	55.84	83.85	66.40	63.23
GT	17.97	14.30	12.18	21.88	15.80	13.23	75.82	57.53	54.09	81.18	64.58	59.45
PV R-CNN	60.32	44.98	41.02	63.00	47.65	42.42	-	-	-	-	-	-
Voxel R-CNN	59.71	46.38	42.69	64.98	48.83	44.89	96.64	88.43	80.28	97.02	89.87	87.94
GT	57.88	44.03	39.40	58.99	43.85	38.94	94.80	87.58	79.92	96.77	89.59	87.60

Table 7.2: Quality of the pseudo-labels. We report the performances of four baseline models trained from the pseudo-labels generated by two LiDAR-based detectors (PV R-CNN and Voxel R-CNN). Metrics are the $AP|_{R_{40}}$ for 3D detection with 0.7 (*upper group*) and 0.5 IoU thresholds (*lower group*). We also show the performances of the models trained from the ground truth (GT) for reference. All the baselines are trained on the 3,712 *training* images and evaluated on *validation* set.

First attempt on pseudo-labeling. We first investigate whether the LiDAR-based 3D detectors, representing the best-performing 3D detectors, can generate good enough pseudo-labels. To avoid biased conclusions caused by the over-fitting of the pseudo-label generators on the training split, we train the LiDAR-based models on the KITTI-3D *validation* set, and generate the pseudo-labels for the *training* set. After that, we train our baseline models with the generated labels. The summary of the experimental results are shown in Table 7.2. From these results, we can observe that the pseudo-labels perform well in providing supervision, even better than the manually annotated labels (especially for the monocular baselines). This shows the promising potential of pseudo-labeling for image-based 3D detection.

Besides, note that the KITTI-3D dataset captures the 3D points with a 64-beam LiDAR, and sparser LiDAR signals are also well applied in other datasets (*e.g.* nuScenes [15] and Argoverse [21] adopt the 32-beam LiDAR), thus we generate the simulated 32-beam and 16-beam LiDAR sweeps for further investigation.

Removing low-quality pseudo-labels by confidence. In the above experiments, we directly use the pseudo-labels to train our models. By analyzing the detection results, we find there are some noisy samples in the results of the LiDAR-based 3D detectors. To remove the potential negative impact caused by these noises, we further filter the pseudo-labels by their confidence. Specifically, we conduct a small grid search on the confidence thresholds and show the results in Figure 7.2. We can find 0.7 threshold gives a good result, although it is not always the best, and we use this threshold for the following experiments in



method	threshold	Easy	Mod.	Hard
MonoDLE	0.5	21.42	16.26	14.87
MonoDLE	0.6	21.84	17.32	15.33
MonoDLE	0.7	22.14	16.97	15.56
MonoDLE	0.8	21.23	16.86	15.38
GUPNet	0.5	25.62	18.76	16.83
GUPNet	0.6	24.91	18.28	16.45
GUPNet	0.7	26.97	19.05	17.01
GUPNet	0.8	26.17	18.71	16.91

Figure 7.2: *Left:* the accumulated numbers (cars) of pseudo-labels with confidence. *Right:* threshold grid search for monocular detectors. We show the $AP|_{R_{40}}$ under moderate setting with 0.7 IoU threshold. The pseudo-label generators are trained on *training* set, and the data are collected on *validation*.

	$AP\|_{R_{40}}$@3D			$AP\|_{R_{40}}$@BEV		
method	Easy	Mod.	Hard	Easy	Mod.	Hard
VisualDet3D [112]	23.63	16.16	12.06	-	-	-
DLE [108]	26.43	16.72	13.02	34.06	22.59	16.96
SGM3D [225]	25.96	17.81	15.11	34.10	23.62	20.49
Baseline (mono)	21.88	15.80	13.23	-	-	-
Ours	27.72	19.38	17.11	-	-	-

Table 7.3: Comparison of the methods apply stereo images in the training phase. The data of the upper group are from their papers. All models are trained from 3,712 frames.

default. Meanwhile, we also show the accumulated instances along with confidences, which suggests about 18K car instances are kept at 0.7 confidence. Note that the annotations provide about 14K ground-truth cars for the same frames.

Can pseudo-labeling work without LiDAR points? We mainly discuss the LiDAR-based pseudo-labeling above, and the further problem is whether can pseudo-labeling work without LiDAR points. For this problem, we consider the following settings: (i) both the stereo and monocular images are available, and (ii) only the monocular images are available. For the first setting, we can use the stereo 3D detectors to generate the pseudo-labels for monocular models. This setting is also adopted by several methods [108, 112, 99, 225], here we compare the proposed method to these works in Table 7.3. For a fair comparison, we do not introduce any other frames, and use the same data with pseudo-labels generated from LIGA-Stereo to train our model. The experimental results show the stereo 3D detectors can also generate good pseudo-labels for monocular ones, and our method shows better performance than the competitors with the same setting ([112, 108] use stereo images to augment the training set while [225] use stereo 3D detector to provide guidance for their model using knowledge distillation [67]).

As for the more challenging setting (ii), we find it can also work in some specific scenarios, and we will further discuss it in Section 7.3.3.

7.3.2 Scalability of the Training Samples

After confirming the effectiveness of the pseudo-labels, this section will provide more applications and an in-depth analysis of pseudo-labeling for image-based 3D detection.

Training with more samples. We first investigate whether the 3D detectors can benefit from more training samples. Particularly, we train the pseudo-label generators on the *training* set and generate the pseudo-labels for the KITTI raw data. Then we divide the raw data into several splits (see Table 7.1 for more details of these data splits) and compare the performances of the selected baseline models trained from them. The experiments shown in Table 7.4 indicate that the monocular 3D detectors can benefit from more training samples. For instance, 10K/25K more training data (b$\rightarrow$d and b$\rightarrow$e) can bring another 4.42/6.93 AP improvements (IoU=0.7, moderate setting).

Stability. A common issue for existing monocular 3D detectors is the unstable performance, especially on the KITTI-3D benchmark. In this Chapter, we find this problem can be largely alleviated by introducing more training samples, especially the temporal sequences. In particular, in the experiments presented by Table 7.4, we conduct multiple runs and report the mean and standard deviation for unbiased comparison. From these results, we can find that (i): increasing the training samples (b$\rightarrow$d, d$\rightarrow$e) is helpful for improving the stability, along with the accuracy, of the algorithms; (ii): compared with simply extending the size of the training set, introducing the temporal sequences (video cues) can better improve the stability (b$\rightarrow$c). In summary, these two observations imply that introducing images with novel scenes tends to boost the model's performance, while more annotated samples in the same scenes can make the models more stable.

Data leakage. Lots of monocular 3D detection methods adopted depth estimators, which are generally pre-trained on the KITTI raw data with Eigen's split [42], to provide depth cues for their models. However, there is an overlap between Eigen's training set and KITTI-3D's validation set. In particular, these two data splits share eight video sequences with 2,859 frames in total. Although

split	# images	$AP\|_{R_{40}}$@IoU=0.7			$AP\|_{R_{40}}$@IoU=0.5		
		Easy	Mod.	Hard	Easy	Mod.	Hard
a. training (gt)	3,712	$22.37_{\pm 0.65}$	$16.21_{\pm 0.42}$	$13.98_{\pm 0.63}$	$60.50_{\pm 1.41}$	$44.97_{\pm 1.25}$	$39.99_{\pm 1.01}$
b. training (pl)	3,712	$25.87_{\pm 1.39}$	$19.07_{\pm 0.91}$	$16.52_{\pm 0.88}$	$64.30_{\pm 2.37}$	$48.83_{\pm 1.63}$	$44.31_{\pm 1.31}$
c. train	13,596	$26.68_{\pm 0.34}$	$19.92_{\pm 0.52}$	$17.50_{\pm 0.20}$	$66.36_{\pm 0.74}$	$50.90_{\pm 0.64}$	$46.58_{\pm 0.65}$
d. eigen-clean	14,940	$31.82_{\pm 0.71}$	$23.49_{\pm 0.85}$	$20.97_{\pm 0.49}$	$67.65_{\pm 0.72}$	$51.58_{\pm 0.58}$	$48.04_{\pm 0.20}$
e. train $\cup$ eigen-clean	28,536	$35.77_{\pm 0.63}$	$26.00_{\pm 0.54}$	$22.69_{\pm 0.16}$	$73.56_{\pm 0.69}$	$57.55_{\pm 0.33}$	$53.47_{\pm 0.27}$
f. eigen	23,488	50.94	40.05	35.32	80.35	68.10	62.25
g. eigen-sampling	14,940	50.85	38.56	34.75	79.92	65.94	60.12

Table 7.4: More training samples. Performances on KITTI-3D *validation* set of GUPNet trained from different data splits. $\pm$ captures the standard deviation over 5 runs. Pseudo-labels are generated by Voxel R-CNN trained from KITTI-3D's *training* split.

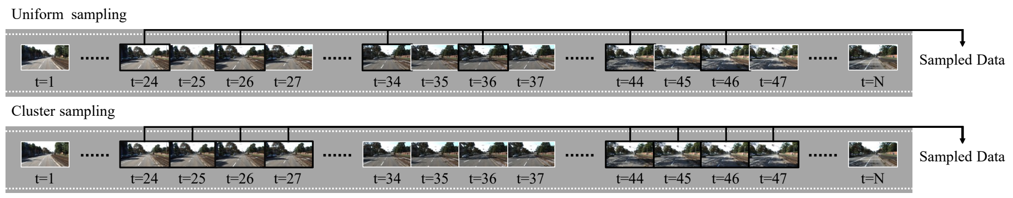


Figure 7.3: Illustration of the sampling strategies. *Top*: Uniform sampling, we use this sampling scheme to simulate the different frame rates during data collection. *Bottom*: Cluster sampling, we use this sampling scheme to divide images into several clips.

the KITTI-3D validation set only include some key-frames among them, there are still 1,258 validation images that overlap with Eigen's training set. Here we quantitatively analyze the bias in performance caused by the data leakage. In particular, following [172], we remove the images, which are geographically close to any images in the validation set, and compare the performances of the models trained from this split (eigen-clean) and Eigen's training set (eigen). Furthermore, we also randomly sample the images from eigen split to generate a new training set (eigen-sampling), which has the same size as the eigen-clean split, for a fairer comparison. As shown in Table 7.4, both the models trained on eigen and eigen-sampling splits show anomalous high performances, compared with that trained from eigen-clean, which suggests the data leakage will cause seriously unfair performance comparison on *validation* set.

Sampling rate. We use all frames in the video sequences as training data in Table 7.4. However, a reasonable conjecture is we may not need all the frames because the contents captured by adjacent frames are similar, thus providing limited information. Although previous experiments show that a denser sampling rate can make the model more stable, presenting the trade-off between the sampling rate and final performance is still meaningful. For this purpose, we uniformly sample the frames from 28,536 training images (train + eigen-clean) to generate several sub-sets and evaluate the models trained from these sub-sets on *validation* set. As shown in Figure 7.4, we can see that about 62.5% of the data can meet the training needs, and more training data does not bring obviously performance improvement. Note that this conclusion can not be directly applied to other datasets due to the different frame rates and moving speed at the data collecting phase. For reference, the KITTI team collected data at 10 Hz with varying driving speeds.

Scenes diversity. Compared with denser sampling from the same image set, a more effective way to expand the dataset scale, *e.g.* collecting more video clips

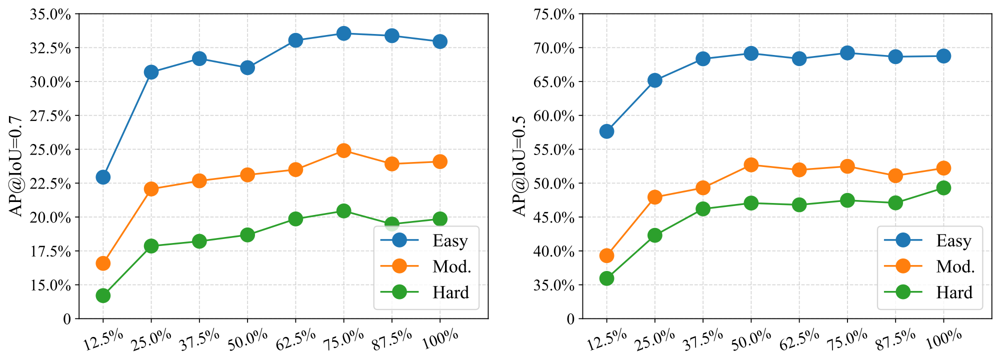


Figure 7.4: Sampling rate. The performance curve of the GUPNet trained with varying sampling rates. Metrics are the AP$|_{R_{40}}$ with 0.5 IoU (*left*) and 0.7 IoU (*right*) thresholds on the KITTI-3D *validation* set.

at different scenes. To study the effect of scene diversity on the performance of monocular 3D detectors, we divide the raw data (train + eigen-clean) into several clips (see Figure 7.3) and use them to train our models. In particular, each clip contains 200 images (about 20 seconds) and at least 200 images are skipped between adjacent clips. Figure 7.5 shows the performance changes of the baseline model *w.r.t* the increase of the scenes. We can see that the baseline model continues to improve as the increase of scenes, which indicates the importance of the scenes' diversity. We hope these data can provide useful knowledge in data collecting and (pseudo) labeling for future work.

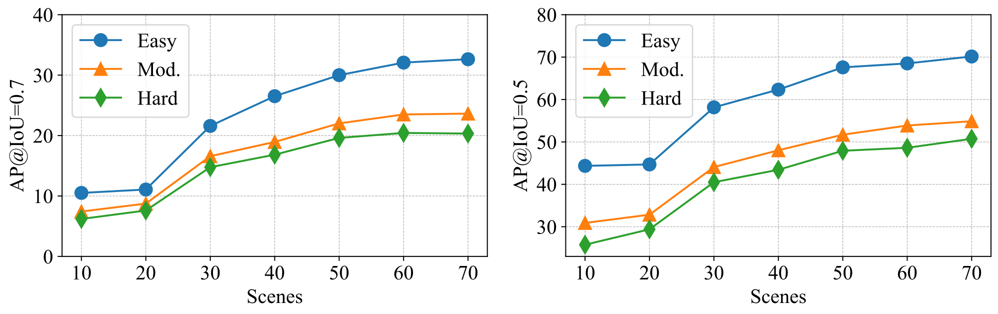
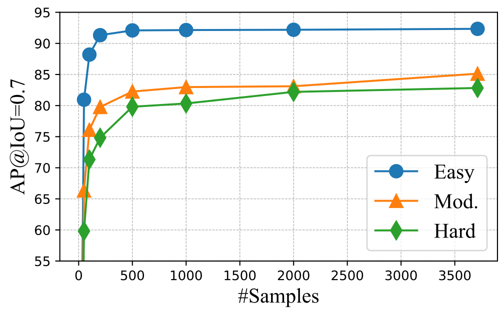


Figure 7.5: Scenes diversity. The performance curve of the GUPNet train with varying scene diversity. Metrics are the AP$|_{R_{40}}$ with 0.5 IoU (*left*) and 0.7 IoU (*right*) thresholds on the KITTI-3D *validation* set.

Training with less manual annotations. Except for introducing unlabeled data, another dimension to evaluate the semi-supervised method is reducing the number of labeled samples. Our method also performs well in this setting when the LiDAR signals are available. The results summarized in Figure 7.6 (*left*) demonstrate the effectiveness of our method in a semi-supervised setting, where our



methods	# imgs	Easy	Mod.	Hard
baseline	3,712	22.37	16.21	13.98
ours	50	21.13	16.25	13.66
ours	100	26.86	18.84	15.96
ours	500	26.91	19.38	16.59
ours	3,712	25.87	19.07	16.52

Figure 7.6: *Left:* the performance curve for Voxel R-CNN with the increase of labeled samples. *Right:* the performance of our method under the semi-supervised setting. # imgs denotes the numbers of *labeled* data, and all models are trained with 3,712 training images. All models are evaluated on KITTI-3D *validation* set with $\mathrm{AP}|_{\mathrm{R}_{40}}$.

method obtains comparable performance to the fully supervised baseline/final model with only 50/100 labeled samples. The performance curve shown in Figure 7.6 (*right*) reveals why it works. For the LiDAR-based 3D detectors, the rich and accurate spatial features make the CNN models can learn the mapping function from LiDAR data to 3D results easily, and we can apply lots of augmentations, *e.g.* scaling, rotation, shifting, and copy-paste, to effectively extend the training samples for this kind of data. Both of them make the LiDAR-based 3D detectors still work with a few annotations (*e.g.* Voxel R-CNN gets 66.35 $\mathrm{AP}|_{\mathrm{R}_{40}}$ with only 50 labeled samples, and reaches 'saturation' at about 500 samples). The pseudo-labeling scheme builds a bridge between LiDAR-based models and image-based models, and then the latter can also work well under a few-shot setting by leveraging LiDAR-based models' features.

Performance of PL generators. Here we show the performance of the pseudo-label (PL) generators used in this Chapter in Table 7.5. In particular, model (a) is the default PL generator in the experiments part, and model (h) is used to investigate the impact on the student models caused by a different PL generator and show the generalization of the proposed method (Section 7.3.1). Besides, model (h) is also used to generate the cyclist/pedestrian pseudo-labels. We train models (d, i) to avoid the biased conclusion caused by the over-fitting (Section 7.3.1), and the models (b, c) are used to show our method when the resolution of LiDAR signals is low. Besides, models (e, f, g) are used to show that our method still performs well when the training samples are limited, and models (j, k, l) are serve for the scenarios when LiDAR signals are not available.

	models	data	setting	Easy	Mod.	Hard
a.	Voxel R-CNN	LiDAR	train → val	92.34	85.13	82.83
b.	Voxel R-CNN	LiDAR (32-beam)	train → val	92.06	80.45	77.89
c.	Voxel R-CNN	LiDAR (16-beam)	train → val	88.51	71.96	69.22
d.	Voxel R-CNN	LiDAR	val → train	91.85	82.77	77.67
e.	Voxel R-CNN	LiDAR (500 samples)	train → val	92.09	82.26	79.81
f.	Voxel R-CNN	LiDAR (100 samples)	train → val	88.20	76.10	71.33
g.	Voxel R-CNN	LiDAR (50 samples)	train → val	80.94	66.35	59.82
h.	PV R-CNN	LiDAR	train → val	92.17	84.53	82.38
i.	PV R-CNN	LiDAR	val → train	92.32	82.84	77.66
j.	GUPNet	Mono	train → val	23.43	17.06	14.84
k.	LIGA Stereo	Stereo	train → val	77.05	58.26	51.85
l.	LIGA Stereo	Stereo	train → val	82.32	64.29	59.34

Table 7.5: Performances of the PL generators used in this Chapter. Metrics are $AP|_{R_{40}}$ for 3D detection under 0.7 threshold. Train and val denote the KITTI-3D training and validation set.

7.3.3 More Discussions

Why pseudo-labeling works? Here we give an empirical interpretation of the success of the pseudo-labeling scheme. First, MonoDLE [122] reports that removing some hard (far) instances can boost the model's performance because these samples are hard to detect for monocular detectors and affect CNN's optimization. The pseudo-labeling also works in a similar way: the teacher model plays the role of the filter to remove the hard instances (MonoDLE uses human-designed rules for this purpose), and let the images-based models focus on the remaining samples, which can make the CNN learns the mapping function easier. This suppose is supported by the experiments in Figure 7.2: removing the low-confidence labels can improve the model's accuracy. Because the removed samples are not only noisy but also hard to detect. Second, the pseudo-labeling corrects some error cases in the annotations (see Figure 7.7). In particular, annotating 3D bounding boxes is a complicated task that involves multi-modal data. For some reason, some objects was not annotated or fully annotated (such as the 'DontCare' objects in KITTI-3D, only the 2D bounding boxes are provides, while the 3D bounding boxes are missed). For the image data, these objects have a similar texture to the annotated ones but are regarded as negative samples in the training phase, which misleads the image-based 3D detectors. For our pseudo-labeling scheme, these samples are re-labeled and then can provide effective supervision to our models. Third, pseudo-labeling does increase the size of the training set and leverage the unlabeled data. Lastly, the pseudo-labeling is a kind of label-smoothing scheme, which is also beneficial to CNN's optimization. Based on the above reasons, pseudo-labeling significantly boosts

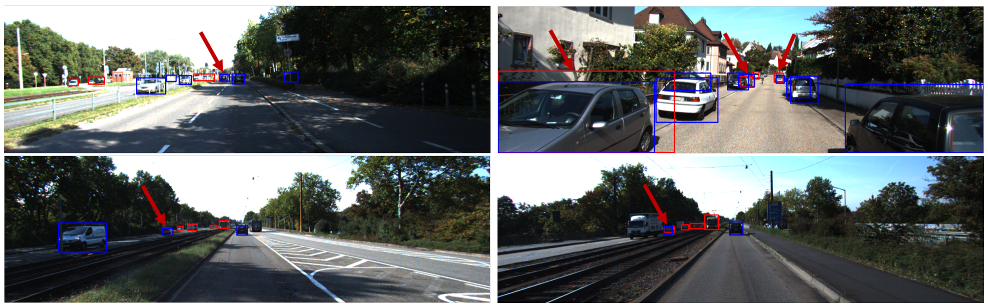


Figure 7.7: Comparison of the ground truth and pseudo-labels. We show the 'DontCare' cases in the ground truths in red and the pseudo-labels in blue. These cases suggest the proposed method can provide more effective supervision than ground truth in some cases. We only show the 2D bounding boxes for clearer presentation.

the performances of image-based 3D detectors.

Self-pseudo-labeling. In Section 7.3.1 and 7.3.2, we show that the pseudo-labels generated from LiDAR/stereo-based methods can provide reliable supervision for monocular 3D detectors (or the LiDAR/stereo combination). This is reasonable because the pseudo-label generators are generally superior to the baseline methods in performance. However, it is questioning whether pseudo-labeling still works without better models. To study this problem, we generate the pseudo-labels using the GUPNet trained from *training* split, and then train the baseline models using the resulting labels. Interestingly, as shown in Table 7.6, the models trained on train split and eigen-clean split show contrary results. Combined with the fact that the *train* split shares the same scenes with the *training* set of pseudo-label generators, we can get the following conclusion: the self-pseudo-labeling scheme still works if the target data share a similar distribution with the labeled data. In other words, it is still a feasible scheme that manually annotates the key-frames of the raw data and then extends the training set by pseudo-labeling the interframe sequences. .

split	# images	Easy	Mod.	Hard	Easy	Mod.	Hard
baseline	3,712	22.37	16.21	13.98	60.50	44.97	39.99
train	13,596	24.49	17.36	15.31	61.28	45.43	40.75
eigen-clean	14,940	21.57	16.24	13.92	59.05	44.69	38.92

Table 7.6: Pseudo-labeling without LiDAR sweeps. Metrics are the $AP_{R_{40}}$ with 0.7/0.5 IoU and 0.5 IoU thresholds. The baseline model also serves as the pseudo-label generator.

	Method	Venue	3D			BEV		
			Easy	Mod.	Hard	Easy	Mod.	Hard
a	MonoFlex[217]	CVPR'21	19.94	13.89	12.07	-	-	-
	AutoShape[114]	ICCV'21	22.47	14.17	11.36	30.66	20.08	15.59
	GUPNet*[115]	ICCV'21	20.11	14.20	11.77	30.29	21.19	18.20
	MonoCon [111]	ICCV'21	22.50	16.46	13.95	31.12	22.10	19.00
b	RTM3D[99]	ECCV'20	14.41	10.34	8.77	19.17	14.20	11.99
	VisualDet3D[112]	RA-L'21	21.65	13.25	9.91	29.81	17.98	13.08
	DLE[108]	BMVC'21	24.23	14.33	10.30	31.09	19.05	14.13
	Ours	-	23.93	14.87	12.45	33.17	20.47	17.31
c	MonoPSR[86]	CVPR'19	10.76	7.25	5.85	18.33	12.58	9.91
	MonoRUn [22]	CVPR'21	19.65	12.30	10.58	27.94	17.34	15.24
	CaDDN [155]	CVPR'21	19.17	13.41	11.46	27.94	18.91	17.19
	MonoDistill [171]	ICLR'22	22.97	16.03	13.60	31.87	22.59	19.72
	Ours		24.43	17.08	15.25	33.38	24.78	22.00
d	Demystifying [172]	ICCV'21	22.40	12.53	10.64	-	-	-
	DDMP-3D [187]	CVPR'21	19.71	12.78	9.80	28.08	17.89	13.44
	PCT[188]	NeurIPS'21	21.00	13.37	11.31	29.65	19.03	15.92
	DFR-Net[229]	ICCV'21	19.40	13.63	10.35	28.17	19.17	14.84
	Ours		28.29	20.23	17.55	37.81	27.70	24.61

Table 7.7: Comparing with SOTA methods for monocular setting on KTTI test set. We show the performances of the proposed method and best-performing counterparts under the following settings: (a) only the monocular images provided by the KITTI-3D are available in the training phase; (b) both monocular images and stereo images provided by the KITTI-3D are available in the training phase; (c) both monocular images and LiDAR signals provided by the KITTI-3D are available in the training phase; (d) both images and LiDAR signals provided by the KITTI-3D and KITTI raw are available in the training phase; Methods are ranked by the $AP|_{R_{40}}$ under moderate setting on *testing* set in each group. *: our baseline model.

7.3.4 Comparing with SOTA Methods

Monocular version. We select the results of some representative settings and evaluate them on the KITTI-3D *testing* server. In particular, the proposed method surpasses DLE [108] by 0.54% for 3D detection, and 1.42% for BEV under (b) setting; surpasses MonoDistill [35] by 1.05% for 3D detection, and 2.19% for BEV under (c) setting; and surpasses DFR-Net [229] 6.6% for 3D detection, and 8.53% for BEV, respectively. These new SOTA performances firmly demonstrate the effectiveness of the proposed methods. See Table 7.7 for more details.

Stereo version. As shown in Table 7.8, to verify the effectiveness of the pseudo label, we also submitted our stereo-based results trained based on trainval dataset with the pseudo label from Voxel R-CNN [40] to the official evaluation benchmark for evaluating our performance on the test set of the KITTI-3D dataset.

	Method	Venue	3D			BEV		
			Easy	Mod.	Hard	Easy	Mod.	Hard
a	Stereo R-CNN [97]	CVPR'19	47.58	30.23	23.72	61.92	41.31	33.42
	SIDE	WACV'19	47.69	30.82	25.68	-	-	-
	Stereo CenterNet	Neurc'22	49.94	31.30	25.62	-	-	-
	RTS3D	AAAI'21	58.51	37.38	31.12	72.17	51.79	43.19
	LIGA-Stereo (Ours)		77.81	58.57	52.13	86.67	71.23	64.08
b	YOLOStereo3D	ICRA'21	65.68	41.25	30.42	-	-	-
	DSGN	CVPR'20	73.50	52.18	45.14	82.90	65.05	56.60
	CDN	NeurIPS'20	74.52	54.22	46.36	83.32	66.24	57.65
	LIGA Stereo [61]	ICCV'21	81.39	64.66	57.22	88.15	76.78	67.40
	LIGA-Stereo (Ours)		83.77	66.97	58.41	90.76	77.40	70.00

Table 7.8: Comparing with SOTA methods for stereo setting on KTTI test set. We show the performances of the proposed method and best-performing counterparts under the following settings: (a) only the stereo images provided by the KITTI-3D in the training phase; (b) both stereo images and LiDAR signals provided by the KITTI-3D are available in the training phase. Methods are ranked by the $AP|_{R_{40}}$ under moderate setting on *testing* set in each group.

Under the pure stereo-based setting, we provide LIGA-Stereo results by removing the depth loss and imitation loss in LIGA-Stereo [61]. For BEV performance, we surpass RTS3D by 19.44% mAP. For 3D detection performance, we surpass RTS3D by 21.19% mAP. Compared with using both stereo images and LiDAR signals in the training phase, ours surpass LIGA-Stereo 2.31% mAP for 3D detection, and 0.62% for BEV, respectively. Besides, the results of stereo models in the KITTI-3D validation set are shown in Table 7.9 and 7.10.

Cyclist and Pedestrian. Due to the high variances in cyclist/pedestrian detection, previous works mainly focus on the car category and attribute this to the limited training samples of cyclist and pedestrian categories. In this Chapter, we introduce more samples and show the performance changes of cyclist/pedestrian detection. In particular, we use the PV R-CNN (Voxel R-CNN is designed for car detection) to generate pseudo-labels and report the mean and standard deviation over the last 10 epochs (140 epochs in total) in Table 7.12. First, overall, we can see that the stereo models trained with pseudo-labels achieve better performances than the baselines (a$\rightarrow$b, e$\rightarrow$f). However, different from the car category, the pseudo-labels are not always better than ground-truth labels, and this may be caused by the following two reasons: (i) LiDAR-based models are not good at detecting small or holed objects, which makes the quality of pseudo-labels of cyclists/pedestrians worse than those of cars (see Figure **??** for an error case of cyclist/pedestrian detection); (ii) we directly adopt the hyper-parameter (*i.e.* confidence threshold) tuned from car detection, and the

		$AP\|_{R_{40}}$@IoU=0.7			$AP\|_{R_{40}}$@IoU=0.5		
split	# images	Easy	Mod.	Hard	Easy	Mod.	Hard
a. training (gt)	3,712	$77.05_{\pm 0.26}$	$58.26_{\pm 0.48}$	$51.85_{\pm 0.13}$	$97.63_{\pm 1.00}$	$89.46_{\pm 0.61}$	$82.36_{\pm 0.05}$
b. training (pl)	3,712	$79.19_{\pm 1.83}$	$59.61_{\pm 1.23}$	$54.19_{\pm 1.57}$	$98.04_{\pm 0.75}$	$89.62_{\pm 0.86}$	$84.40_{\pm 0.85}$
c. train	13,596	$79.31_{\pm 0.90}$	$61.20_{\pm 0.88}$	$55.90_{\pm 1.21}$	$96.45_{\pm 0.71}$	$89.91_{\pm 0.17}$	$84.75_{\pm 0.16}$
d. train ∪ eigen-clean	28,536	$84.55_{\pm 1.00}$	$68.72_{\pm 1.26}$	$63.48_{\pm 0.97}$	$97.40_{\pm 0.82}$	$91.96_{\pm 0.22}$	$88.12_{\pm 0.89}$

Table 7.9: Performances ($AP|_{R_{40}}$) on KITTI-3D *validation* set of LIGA Stereo trained from different data splits. $\pm$ captures the standard deviation over the last 10 epochs. Pseudo-labels are generated by Voxel R-CNN trained from KITTI-3D's *training* split.

		$AP\|_{R_{40}}$@IoU=0.7			$AP\|_{R_{40}}$@IoU=0.5		
split	# images	Easy	Mod.	Hard	Easy	Mod.	Hard
a. training (gt)	3,712	$82.32_{\pm 1.15}$	$64.29_{\pm 1.75}$	$59.34_{\pm 1.61}$	$98.76_{\pm 0.78}$	$92.81_{\pm 0.29}$	$87.72_{\pm 0.25}$
b. training (pl)	3,712	$85.04_{\pm 0.86}$	$66.77_{\pm 1.03}$	$61.97_{\pm 0.94}$	$98.65_{\pm 0.91}$	$92.78_{\pm 0.29}$	$89.02_{\pm 0.96}$
c. train	13,596	$85.66_{\pm 1.03}$	$67.22_{\pm 0.51}$	$62.44_{\pm 0.42}$	$98.63_{\pm 0.10}$	$92.70_{\pm 0.13}$	$89.84_{\pm 0.17}$
d. train ∪ eigen-clean	28,536	$87.41_{\pm 0.55}$	$72.98_{\pm 0.56}$	$68.56_{\pm 0.77}$	$98.60_{\pm 0.23}$	$94.63_{\pm 0.15}$	$91.76_{\pm 0.53}$

Table 7.10: Performances ($AP|_{R_{40}}$) on KITTI-3D *validation* set of LIGA Stereo trained from different data splits. $\pm$ captures the standard deviation over the last 10 epochs. Pseudo-labels are generated by Voxel R-CNN trained from KITTI-3D's *training* split.

	training (gt)	training (pl)	train (pl)	eigen-clean (pl)
# images	955 of 3,712	1,045 of 3,712	5,173 of 13,596	1,451 of 14,940
# instances	2,207	2,209	10,189	1,767
# images	514 of 3,712	566 of 3,712	2,019 of 13,596	1,205 of 14,940
# instances	734	826	3,048	1,270

Table 7.11: The numbers of images and instances of pedestrians (*upper group)* and cyclists (*lower group*). For pseudo-labels, we only consider the instances with confidence over 0.7.

models may be enhanced by further fine-tuning. Second, increasing the training samples also improves the accuracy of cyclist/pedestrian detection (b$\rightarrow$c, f$\rightarrow$g). Third, we can find the proportion of corresponding samples among training sets by comparing experiments in Table 7.12 and the statistical information shown in Table 7.11.

7.4 Conclusion

In this Chapter, we present the pseudo-labeling scheme for image-based 3D detection. In this approach, we leverage the side-products in the data collecting and annotating phases and use these data to generate the pseudo-labels, and then augment the training set of image-based models. Surprisingly, except for the increased size of training data, the pseudo-labels themselves have a significantly positive impact on the monocular/stereo models, and we provide an empirical explanation for it. Besides, we also conduct extensive experiments under varying settings to explore the potential scenarios for our method, and our method gets impressive performances for all of them, *i.e.* our method achieves new SOTA for multiple settings on the KITTI-3D *testing* benchmark.

		$AP\|_{R_{40}}$@IoU=0.5			$AP\|_{R_{40}}$@IoU=0.25		
split	# images	Easy	Mod.	Hard	Easy	Mod.	Hard
a. training (gt)	3,712	$8.75_{\pm 0.33}$	$7.00_{\pm 0.46}$	$5.42_{\pm 0.39}$	$33.96_{\pm 0.33}$	$26.51_{\pm 0.23}$	$22.15_{\pm 0.19}$
b. training (pl)	3,712	$10.59_{\pm 0.22}$	$8.29_{\pm 0.16}$	$6.38_{\pm 0.12}$	$30.08_{\pm 0.58}$	$26.79_{\pm 0.20}$	$21.98_{\pm 0.73}$
c. train	13,596	$20.08_{\pm 0.32}$	$15.27_{\pm 0.22}$	$12.12_{\pm 0.15}$	$41.53_{\pm 0.66}$	$33.15_{\pm 0.18}$	$27.04_{\pm 0.13}$
d. eigen-clean	14,940	$1.21_{\pm 0.19}$	$0.98_{\pm 0.15}$	$0.66_{\pm 0.05}$	$9.00_{\pm 0.52}$	$7.25_{\pm 0.22}$	$6.15_{\pm 0.19}$
e. training (gt)	3,712	$8.68_{\pm 0.64}$	$4.48_{\pm 0.18}$	$4.08_{\pm 0.36}$	$24.36_{\pm 0.51}$	$13.86_{\pm 0.26}$	$12.62_{\pm 0.20}$
f. training (pl)	3,712	$8.93_{\pm 0.69}$	$4.31_{\pm 0.39}$	$3.90_{\pm 0.18}$	$28.05_{\pm 0.74}$	$15.44_{\pm 0.42}$	$14.23_{\pm 0.35}$
g. train	13,596	$10.24_{\pm 0.32}$	$5.58_{\pm 0.24}$	$4.95_{\pm 0.13}$	$30.01_{\pm 0.81}$	$17.96_{\pm 0.40}$	$16.33_{\pm 0.34}$
h. eigen-clean	14,940	$5.92_{\pm 0.40}$	$2.84_{\pm 0.34}$	$2.61_{\pm 0.15}$	$26.87_{\pm 0.88}$	$14.07_{\pm 0.46}$	$12.38_{\pm 0.46}$

Table 7.12: Performances of GUPNet on KITTI-3D *validation* set for Pedestrian (*upper group*) and Cyclist (*lower group*). $\pm$ captures the standard deviation over the last 10 epochs. Pseudo-labels are generated by PV R-CNN trained from KITTI-3D's *training* split.

Chapter 8

More Discussions and Future Directions

8.1 Discussions

In this section, we provide additional discussions about image-based 3D detection, including the metrics it uses, the trade-off between accuracy and speed, and the comparisons with LiDAR-based methods.

8.1.1 Metrics and Applications

The metrics used in 3D detection are mainly derived from AP. In particular, KITTI 3D and Waymo Open use 3D/BEV IoU as criteria to distinguish a prediction among TP or FP, and then compute the precision-recall curve. However, the 3D/BEV IoU is sensitive to 3D position and goes from 1 to 0 quickly, which makes it difficult to detect far-away objects, especially for image-based methods [122, 162, 15]. However, these far-away objects are less important than the near ones in the autonomous driving scenarios, and the metric should not be so sensitive to these samples. In contrast, nuScenes use center distance as the criterion to distinguish between TP and FP, and evaluate each item of the objects separately. Under this setting, image-based methods achieve higher accuracy, even surpassing LiDAR-based methods in some cases [15]. However, this metric still applies unified standards for all samples, instead of treating different samples differently [48, 39].

Furthermore, we should note that different types of detection errors bring different potential hazards to practical applications. For example, it is more important to provide a prediction instead of missing it, even if the localization accuracy is not so accurate. However, for existing AP-based metrics, the penalty of giving a FP is greater than that of missing a TP (both cases have the same recall, but the former has a lower precision, see Equation 2.2 for the definition of recall and precision).

In summary, in addition to meeting the basic requirements, we believe the ideal metric for the 3D detection task in autonomous driving applications should have the following two features: (i) treat the objects at different distances differently and focus more on the near objects (*e.g.* reduce the weights or the criterion for the far-away objects in evaluation); (ii) treat different types of errors differently (*e.g.* higher penalty for the missing than mislocalization).

8.1.2 Accuracy and Speed

The inference speed in the 3D detection task is equally important to the accuracy for autonomous driving applications. However, so far, most of the research works only focused on the accuracy of predictions. For instance, pseudo-LiDAR-based methods achieve some gains in performance, but they also introduce extra computational overhead because they use an auxiliary network to estimate the depth maps. In particular, the most commonly used depth estimators [47, 19] in pseudo-LiDAR models take around 400 ms to compute the depth map for a standard KITTI 3D image. This latency will cause about 4 meters shift for a vehicle with a speed of 36 km/h. Although the actual situation can not be modeled so simply, this is just an example to illustrate the importance of algorithms' speed.

Additionally, the research community around 3D object detection is not so well established yet as compared to those studying some fundamental CV tasks such as 2D detection or semantic segmentation. Consequently, standard evaluation protocols are less mature and it is difficult to force all methods to use the same backbones or frameworks for a fair comparison. In this respect, evaluating both accuracy and speed represent a necessary step forward for comparing different methods, similar to what the 2D detection community [**tradeoff**, 156, 110] did a few years ago.

8.1.3 Image-based Methods and LiDAR-based Methods

Another main branch of the 3D detection task is the LiDAR-based methods. Here we discuss the strengths and weaknesses of the image-based methods and LiDAR-based methods.

	KITTI			nuScenes	
	Easy	Moderate	Hard	mAP	NDS
PointPillars [90]	79.05	74.99	68.30	30.5	45.3
MonoDIS [171]	16.54	12.97	11.04	30.4	38.4

Table 8.1: Performances of PointPillars [90] and MonoDIS [171]. For KITTI, we show the $\mathrm{AP}|_{R40}$ of the Car category on the *test* set. For nuScenes, we show the mAP and NDS on the *test* set.

Overall Performances on KITTI 3D and nuScenes. In this section, we choose two representative 3D detectors[4] to deeply analyze the features of LiDAR-based methods and image-based methods. As shown in Table 8.1, there is a huge gap between the performances of PointPillars [90] and MonoDIS [171] on KITTI 3D, while they have a similar accuracy on nuScenes. This is a common phenomenon for existing algorithms, which is mainly caused by the following three reasons: (i) The resolutions of LiDAR signals are different. KITTI 3D uses a 64-beam LiDAR to capture the objects, while a 32-beam LiDAR is adopted in nuScenes when data collecting. (ii) The objects of interest are different. KITTI 3D mainly focuses on the Car category, while nuScenes averages the performance of ten classes, including some small or holed objects that are not easily captured by LiDAR signals. (iii) The metrics are different. The metric used in nuScenes disengages the objects and reduces the criterion of localization accuracy, which is the primary error type for image-based methods. In particular, nuScenes averages the AP under different distance thresholds, *i.e.* $\mathbb{D} = \{0.5, 1, 2, 4\}$ meters, and the average distance error (ATE in Table 8.2) of MonoDIS is about 0.7m which is okay for 1.0m, 2.0m, and 4.0m thresholds.

Detailed Performances Analysis on nuScenes. nuScenes decouples detection and reports the accuracy for each item individually. This design allows us to analyse the strengths and drawbacks of image-based methods in detail. The following observations can be made according to the results in Table 8.2: (i) Although the mAP of the two algorithms is very close (30.5 *v.s.* 30.4), they show different patterns in class-wise evaluation. MonoDIS is good at holed objects (*e.g.* Barrier and Bicycle) and thin objects (*e.g.* Traffic Cone), while PointPillars shows a higher accuracy on large objects (*e.g.* Bus and Car). (ii) In most cases, PointPillars has a lower ATE, which means it can estimate the location of objects more accurately. (iii) The size of objects estimated from MonoDIS

[4]We choose MonoDIS and PointPillar as examples because they are con-current and also adopted in the nuScenes's official report [15]. More recent methods with better performance can be found in our website.

	Barrier	Bicycle	Bus	Car	Constr. Veh.	Motorcycle	Pedestrian	Traffic Cone	Trailer	Truck	Mean
AP	38.9/51.1	1.1/24.5	28.2/18.8	68.4/47.8	4.1/7.4	27.4/29.0	59.7/37.0	30.8/48.7	23.4/17.6	23.0/22.0	30.5/30.4
ATE	0.71/0.53	0.31/0.71	0.56/0.84	0.28/0.61	0.89/1.03	0.36/0.66	0.28/0.70	0.40/0.50	0.89/1.03	0.49/0.78	0.52/0.74
ASE	0.30/0.29	0.32/0.30	0.20/0.19	0.16/0.15	0.49/0.39	0.29/0.24	0.31/0.31	0.39/0.36	0.20/0.20	0.23/0.20	0.29/0.26
AOE	0.08/0.15	0.54/1.04	0.25/0.12	0.20/0.07	1.26/0.89	0.79/0.51	0.37/1.27	-/-	0.83/0.78	0.18/0.08	0.50/0.55
AVE	-/-	0.43/0.93	0.42/2.86	0.24/1.78	0.11/0.38	0.63/3.15	0.25/0.89	-/-	0.20/0.64	0.25/1.80	0.32/1.55
AAE	-/-	0.68/0.01	0.34/0.30	0.36/0.12	0.15/0.15	0.64/0.02	0.16/0.18	-/-	0.21/0.15	0.41/0.14	0.37/0.13

Table 8.2: Detailed performances of PointPillars/MonoDIS on the nuScenes *test* set. We use red/blue to highlight the PointPillars/MonoDIS if it achieves at least 20% improvements at the corresponding item to its counterpart. Data is collected from nuScenes [15].

	Singapore	Rain	Night
PointPillars [90]	- 1%	- 6%	-36%
MonoDIS [171]	- 8%	+ 3%	-58%

Table 8.3: The relative mAP changes of MonoDIS and PointPillars on the subsets of the nuScenes *validation* set.

is slightly better than that from PointPillars. (iv) Benefiting from the accurate spatial information provided by LiDAR signals, PointPillars can accurately estimate the instantaneous velocity of the objects. (v) MonoDIS shows a better ability to recognize the attributes of objects (*e.g.* whether a car is stopped or moving), which is an important feature for autonomous driving.

Generalization. nuScenes provides the performance changes of PointPillars and MonoDIS on subsets of validation set (shown in Table 8.3), which can be used to analyze the robustness of these methods. We can observe that a small performance drop on the Singapore split for MonoDIS. This indicates that the change in data distribution will affect the accuracy of monocular-based methods (mainly caused by the biased depth estimation). Besides, although the performance of PointPillars in the rainy split has only slightly decreased, it may be worse in practice because some frames in this split are not ongoing rainfall [15]. The biggest challenge is the night data, and we can see both MonoDIS and PointPillars experience a significant performance drop. Furthermore, MonoDIS has a more significant performance decrease than PointPillars, which may indicate that image-based methods are more sensitive to poor lighting.

8.2 Future Directions

Image-based 3D object detection is a relatively new field. Performances have been rapidly and constantly improving but there are still many limitations and directions which need to be further analyzed and explored. In this section we highlighted some of the most relevant ones, hoping to provide relevant cues for impactful future work.

8.2.1 Depth Estimation

The performance of image-based 3D object detection methods heavily relies on the capability of estimating the precise distance of the objects. A relevant future direction is therefore to analyze and improve the depth estimation capabilities of 3D object detectors. Many recent works, such as [155, 168, 115, 50, 191, 190], try to address this, proposing alternative definitions for the regression targets and loss formulations and demonstrating that there is still a lot of room for improvement.

Another interesting future direction comes by observing that, quite surprisingly, the depth estimation and 3D object detection communities have been almost completely independent from one another. A first attempt to join these communities has been made with the introduction of Pseudo-LiDAR methods [192, 121, 195], where 3D object detectors have been paired with pre-trained depth estimators and demonstrated to achieve better overall performance. While this is a promising initial step, the depth and detection methods were still completely independent. To overcome this, [150, 137] proposed to join the 3D detection and depth estimation into a single multi-task network. They demonstrated that, when these two tasks are trained together and have the possibility to benefit from one another, the 3D detection performance increases even more. We believe these results to show and validate the potential of the union of depth and detection, highlighting that this will constitute a relevant future direction.

8.2.2 Pre-Training

Some work pre-train certain components of the model. However, the vast majority of the methods still train their models from scratch or use ImageNet pre-trained weights. We expect approaches that adopt pre-training to be further investigated and become more popular, especially taking into account their potential in challenging scenarios, *e.g.* unsupervised settings. Particularly, related techniques, such as BERT [81] or MoCo [62], achieve great success in NLP and 2D vision, while have not been introduced in to image-based 3D detection field. Such technologies may be able to leverage the massive un-labeled data in the autonomous driving scenario and further boost the accuracy of 3D detectors.

8.2.3 Beyond Fully Supervised Learning

The creation of 3D detection datasets is an extremely expensive and time consuming operation. It generally involves the synergy between different technologies (*e.g.* LiDAR, GPS, cameras) as well as a substantial amount of workforce. The annotation process is highly demanding and, even in the presence of many quality checks, it is inevitably affected by errors, especially for long-range objects. In light of this, it is concerning to see that the almost totality of the 3D object detection methods is fully supervised, *i.e.* requires the 3D bounding box annotations to be trained. Contrarily to other related communities where the full supervision requirement has been relaxed *e.g.* depth estimation [57, 58] or LiDAR-based 3D detection [160, 218], very little effort has been devoted to exploring semi- or self-supervised approaches [6, 98, 141]. In this regard, it is worth to highlight the method in [6], which introduces a differentiable rendering module that enables to exploit input RGB image as the only source of supervision. Also in light of the recent advancements in the field of differentiable rendering on generic scenes (*e.g.* NeRF [127]) and real objects (*e.g.* [132], [134]), we believe this particular direction to be extremely valuable and able to potentially relax the requirements of 3D box annotations. Besides, to address possible errors in data annotations, *e.g.* missing annotation for long-range objects, the geometry consistency between temporal sequences or multi-frame is also encouraged for both full-, semi-, or self-supervised learning.

8.2.4 Multi-Modality

As discussed in Section 7.3.3, both image data and LiDAR data have their advantages, and some methods, such as [29, 85, 146, 104, 102], have recently started to integrate these two types of data into a single model. However, the research in this field is still on its infancy. Additionally, other modalities of data could be considered to further improve the accuracy and robustness of algorithms. For example, compared with LiDAR, RADAR equipment has a longer sensing distance, which may be used to boost the accuracy of far-away objects. Besides, RADAR is more stable in some extreme weather conditions, such as rainy day and foggy day. However, although the synchronized RADAR data are already provided in some datasets [15, 5, 10], there are only a few methods [10, 149, 130] which investigate how to use them. Another example is the data from thermal cameras [34], which provides new opportunities to advance detection accuracy by tackling adverse illumination conditions. In summary, the ideal detection algorithms should integrate a variety of data, to cover heterogeneous and extreme conditions.

8.2.5 Temporal Sequences

In the real world, human drivers rely on *continuous* visual perception to obtain information about the surrounding environment. However, most of the works in this field solve the 3D detection problem from the perspective of a single frame, which is obviously sub-optimal, and only one recently work [14] has started to consider temporal cues and constraints. On the other hand, lots of works had proved the effectiveness of using video data in many tasks, including 2D detection [76, 227], depth estimation [213, 221], and LiDAR-based 3D detection [117, 209]. The successes in these related fields demonstrate the potential of leveraging video data in the 3D detection task, and new breakthroughs can be achieved by introducing the temporal data and building new constraints in the spatio-temporal space.

A particularly interesting future direction regarding the use of sequences is that they can be used to relax the requirement of full supervision. If combined with the already available input RGB images in fact, they are demonstrated to enable self-supervised depth estimation [60]. In light of this, it is reasonable to think that if the same supervision would be used to also recover the shape and appearance of objects, the same approach could be used to perform 3D object detection as suggested by [134, 6].

The last relevant direction is represented by velocity estimation. Some datasets, *e.g.* nuScenes [15], are in fact required to estimate not only the 3D boxes of objects but also their velocities w.r.t. the global coordinate system. This introduces another extremely challenging task which requires to be solved through the use of multiple images.

8.2.6 Generalization

Generalization plays an important role in the security of self-driving cars. In this regard, it is unfortunately quite well known that image-based 3D object detection methods experience a quite significant drop in performance when tested on unseen datasets, objects, or challenging weather conditions. An example can be found in Table 8.3, where we show the performance of an image-based baseline (along with a LiDAR baseline) on subsets of the popular nuScenes dataset which contain images captured with rain or at night. On the many factors that cause this performance drop, there is certainly the issue that almost the totality of image-based 3D detectors is camera dependent *i.e.* they expect the camera intrinsic parameters to be unchanged between the training and testing phase. Initial attempts to overcome this limitation have been developed in [66] but we believe that this direction should be further explored. Another crucial factor comes from the fact that many image-based 3D object detection methods rely on dataset-specific object priors *i.e.* average object 3D extents in order to make their predictions. If tested on different datasets where the objects, *e.g.* cars, are significantly deviating from these average extents then the 3D detector is likely to fail. Since the effort towards solving this issue have very limited [193, 207, 203, 118] and uniquely focused on LiDAR-based approaches, we believe that this also constitutes a relevant future direction.

Bibliography

[1] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*. 1999.

[2] W. Bao, B. Xu, and Z. Chen. "Monofenet: Monocular 3d object detection with feature enhancement networks". In: *IEEE Transactions on Image Processing (T-IP)* (2019).

[3] W. Bao, Q. Yu, and Y. Kong. "Object-Aware Centroid Voting for Monocular 3D Object Detection". In: *arXiv preprint arXiv:2007.09836* (2020).

[4] I. Barabanau, A. Artemov, E. Burnaev, and V. Murashkin. "Monocular 3d object detection via geometric reasoning on keypoints". In: *arXiv preprint arXiv:1905.05618* (2019).

[5] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner. "The Oxford Radar RobotCar Dataset: A Radar Extension to the Oxford RobotCar Dataset". In: *International Conference on Robotics and Automation(ICRA)*. 2020.

[6] D. Beker, H. Kato, M. A. Morariu, T. Ando, T. Matsuoka, W. Kehl, and A. Gaidon. "Monocular differentiable rendering for self-supervised 3D object detection". In: *European Conference on Computer Vision (ECCV)*. 2020.

[7] I. Bello, B. Zoph, A. Vaswani, J. Shlens, and Q. V. Le. "Attention augmented convolutional networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[8] J. Beltrán, C. Guindel, F. M. Moreno, D. Cruzado, F. Garcia, and A. De La Escalera. "Birdnet: a 3d object detection framework from lidar information". In: *International Conference on Intelligent Transportation Systems (ITSC)*. 2018.

[9] L. Bertoni, S. Kreiss, and A. Alahi. "MonoLoco: Monocular 3D Pedestrian Localization and Uncertainty Estimation". In: *International Conference on Computer Vision (ICCV)*. 2019.

[10] M. Bijelic, T. Gruber, F. Mannan, F. Kraus, W. Ritter, K. Dietmayer, and F. Heide. "Seeing Through Fog Without Seeing Fog: Deep Multimodal Sensor Fusion in Unseen Adverse Weather". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[11] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. “Soft-NMS–improving object detection with one line of code”. In: *International Conference on Computer Vision (ICCV)*. 2017.

[12] D. Bolya, S. Foley, J. Hays, and J. Hoffman. “TIDE: A General Toolbox for Identifying Object Detection Errors”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[13] G. Brazil and X. Liu. “M3d-rpn: Monocular 3d region proposal network for object detection”. In: *International Conference on Computer Vision (ICCV)*. 2019.

[14] G. Brazil, G. Pons-Moll, X. Liu, and B. Schiele. “Kinematic 3D Object Detection in Monocular Video”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[15] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. “nuscenes: A multimodal dataset for autonomous driving”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[16] Y. Cai, B. Li, Z. Jiao, H. Li, X. Zeng, and X. Wang. “Monocular 3D Object Detection with Decoupled Structured Polygon Estimation and Height-Guided Depth Estimation.” In: *Association for the Advancement of Artificial Intelligence (AAAI)*. 2020.

[17] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. “A unified multi-scale deep convolutional neural network for fast object detection”. In: *European Conference on Computer Vision (ECCV)*. 2016.

[18] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau. “Deep MANTA: A Coarse-to-fine Many-Task Network for joint 2D and 3D vehicle analysis from monocular image”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[19] J.-R. Chang and Y.-S. Chen. “Pyramid stereo matching network”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[20] J.-R. Chang and Y.-S. Chen. “Pyramid stereo matching network”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[21] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan, and J. Hays. “Argoverse: 3D Tracking and Forecasting With Rich Maps”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[22] H. Chen, Y. Huang, W. Tian, Z. Gao, and L. Xiong. “MonoRUn: Monocular 3D Object Detection by Reconstruction and Uncertainty Propagation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[23] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2017).

[24] L. Chen, J. Sun, Y. Xie, S. Zhang, Q. Shuai, Q. Jiang, G. Zhang, H. Bao, and X. Zhou. "Shape Prior Guided Instance Disparity Estimation for 3D Object Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)*. 2021.

[25] P. Chen, S. Liu, H. Zhao, and J. Jia. "Distilling Knowledge via Knowledge Review". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[26] X. Chen, K. Kundu, Z. Zhang, H. Ma, S. Fidler, and R. Urtasun. "Monocular 3d object detection for autonomous driving". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[27] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. "3d object proposals for accurate object class detection". In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2015.

[28] X. Chen, K. Kundu, Y. Zhu, H. Ma, S. Fidler, and R. Urtasun. "3d object proposals using stereo imagery for accurate object class detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2017).

[29] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. "Multi-view 3d object detection network for autonomous driving". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[30] Y. Chen, S. Liu, X. Shen, and J. Jia. "Dsgn: Deep stereo geometry network for 3d object detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[31] Y. Chen, L. Tai, K. Sun, and M. Li. "MonoPair: Monocular 3D Object Detection Using Pairwise Spatial Relationships". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[32] Z. Chen, V. Badrinarayanan, C.-Y. Lee, and A. Rabinovich. "Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks". In: *International Conference on Machine Learning (ICML)*. 2018.

[33] J. Choi, D. Chun, H. Kim, and H.-J. Lee. "Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving". In: *International Conference on Computer Vision (ICCV)*. 2019.

[34] Y. Choi, N. Kim, S. Hwang, K. Park, J. S. Yoon, K. An, and I. S. Kweon. "KAIST multi-spectral day/night data set for autonomous and assisted

driving". In: *IEEE Transactions on Intelligent Transportation Systems (T-ITS)* (2018).

[35] Z. Chong, X. Ma, H. Zhang, Y. Yue, H. Li, Z. Wang, and W. Ouyang. "MonoDistill: Learning Spatial Features for Monocular 3D Object Detection". In: *International Conference on Learning Representations (ICLR)*. 2022.

[36] X. Chu, J. Deng, Y. Li, Z. Yuan, Y. Zhang, J. Ji, and Y. Zhang. "Neighbor-vote: Improving monocular 3d object detection through neighbor distance voting". In: *ACM Multimedia (ACM-MM)*. 2021.

[37] R. T. Collins. "A space-sweep approach to true multi-image matching". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 1996.

[38] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. "Deformable convolutional networks". In: *International Conference on Computer Vision (ICCV)*. 2017.

[39] B. Deng, C. R. Qi, M. Najibi, T. Funkhouser, Y. Zhou, and D. Anguelov. "Revisiting 3D Object Detection From an Egocentric Perspective". In: 2021.

[40] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li. "Voxel R-CNN: Towards High Performance Voxel-based 3D Object Detection". In: *Association for the Advancement of Artificial Intelligence (AAAI)*. 2021.

[41] M. Ding, Y. Huo, H. Yi, Z. Wang, J. Shi, Z. Lu, and P. Luo. "Learning Depth-Guided Convolutions for Monocular 3D Object Detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[42] D. Eigen, C. Puhrsch, and R. Fergus. "Depth map prediction from a single image using a multi-scale deep network". In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2014.

[43] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. "The pascal visual object classes (voc) challenge". In: *International Journal of Computer Vision (IJCV)* (2010).

[44] J. Feng, E. Ding, and S. Wen. "Monocular 3D Object Detection via Feature Domain Adaptation". In: *European Conference on Computer Vision (ECCV)*. 2020.

[45] J. Flynn, I. Neulander, J. Philbin, and N. Snavely. "Deepstereo: Learning to predict new views from the world's imagery". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[46] F. D. Foresee and M. T. Hagan. "Gauss-Newton Approximation to Bayesian Learning". In: *International Conference on Neural Networks (ICNN)*. 1997.

[47] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. "Deep Ordinal Regression Network for Monocular Depth Estimation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[48] N. Gählert, N. Jourdan, M. Cordts, U. Franke, and J. Denzler. "Cityscapes 3D: Dataset and Benchmark for 9 DoF Vehicle Detection". In: *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020.

[49] N. Gählert, J.-J. Wan, N. Jourdan, J. Finkbeiner, U. Franke, and J. Denzler. "Single-Shot 3D Detection of Vehicles from Monocular RGB Images via Geometrically Constrained Keypoints in Real-Time". In: *IEEE Symposium on Intelligent Vehicle (IV)*. 2020.

[50] D. Garg, Y. Wang, B. Hariharan, M. Campbell, K. Weinberger, and W.-L. Chao. "Wasserstein Distances for Stereo Disparity Estimation". In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2020.

[51] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. "Vision meets Robotics: The KITTI Dataset". In: *International Journal of Robotics Research (IJRR)* (2013).

[52] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for autonomous driving? the kitti vision benchmark suite". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012.

[53] Q. Geng, H. Zhang, F. Lu, X. Huang, S. Wang, Z. Zhou, and R. Yang. "Part-level car parsing and reconstruction in single street view images". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2021).

[54] J. Geyer, Y. Kassahun, M. Mahmudi, X. Ricou, R. Durgesh, A. S. Chung, L. Hauswald, V. H. Pham, M. Mühlegg, S. Dorn, et al. "A2D2: Audi autonomous driving dataset". In: *arXiv preprint arXiv:2004.06320* (2020).

[55] R. Girshick. "Fast r-cnn". In: *International Conference on Computer Vision (ICCV)*. 2015.

[56] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014.

[57] C. Godard, O. Mac Aodha, and G. J. Brostow. "Unsupervised monocular depth estimation with left-right consistency". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[58] C. Godard, O. Mac Aodha, M. Firman, and G. J. Brostow. "Digging into self-supervised monocular depth estimation". In: *International Conference on Computer Vision (ICCV)*. 2019.

[59] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. "Generative Adversarial Nets". In: *NeurIPS*. 2014.

[60] V. Guizilini, R. Ambrus, S. Pillai, A. Raventos, and A. Gaidon. "3D Packing for Self-Supervised Monocular Depth Estimation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[61] X. Guo, S. Shi, X. Wang, and H. Li. “LIGA-Stereo: Learning LiDAR Geometry Aware Representations for Stereo-based 3D Detector”. In: *International Conference on Computer Vision (ICCV)*. 2021.

[62] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. “Momentum contrast for unsupervised visual representation learning”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[63] K. He, G. Gkioxari, P. Dollár, and R. Girshick. “Mask r-cnn”. In: *International Conference on Computer Vision (ICCV)*. 2017.

[64] K. He, X. Zhang, S. Ren, and J. Sun. “Deep residual learning for image recognition”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[65] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang. “Bounding box regression with uncertainty for accurate object detection”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[66] J. Heylen, M. De Wolf, B. Dawagne, M. Proesmans, L. Van Gool, W. Abbeloos, H. Abdelkawy, and D. Olmeda Reino. “MonoCInIS: Camera Independent Monocular 3D Object Detection using Instance Segmentation”. In: *International Conference on Computer Vision Workshops (ICCVW)*. 2021.

[67] G. Hinton, O. Vinyals, and J. Dean. “Distilling the knowledge in a neural network”. In: *arXiv preprint arXiv:1503.02531* (2015).

[68] D. Hoiem, Y. Chodpathumwan, and Q. Dai. “Diagnosing error in object detectors”. In: *European Conference on Computer Vision (ECCV)*. 2012.

[69] Y. Hou, Z. Ma, C. Liu, T. Hui, and C. C. Loy. “Inter-Region Affinity Distillation for Road Marking Segmentation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[70] J. Hu, L. Shen, and G. Sun. “Squeeze-and-Excitation Networks”. In: *Computer Vision and Pattern Recognition (CVPR)*. 2018.

[71] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger. “Densely connected convolutional networks”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[72] J. Huang, G. Huang, Z. Zhu, and D. Du. “BEVDet: High-performance Multi-camera 3D Object Detection in Bird-Eye-View”. In: *arXiv preprint arXiv:2112.11790* (2021).

[73] Q. Huang, W. Wang, and U. Neumann. “Recurrent Slice Networks for 3D Segmentation of Point Clouds”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[74] S. Ji, W. Xu, M. Yang, and K. Yu. "3D convolutional neural networks for human action recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2012).

[75] B. Jiang, R. Luo, J. Mao, T. Xiao, and Y. Jiang. "Acquisition of localization confidence for accurate object detection". In: *European Conference on Computer Vision (ECCV)*. 2018.

[76] K. Kang, W. Ouyang, H. Li, and X. Wang. "Object detection from video tubelets with convolutional neural networks". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[77] A. Kendall. "Geometry and Uncertainty in Deep Learning for Computer Vision". PhD thesis. University of Cambridge, 2018.

[78] A. Kendall and Y. Gal. "What uncertainties do we need in bayesian deep learning for computer vision?" In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2017.

[79] A. Kendall, Y. Gal, and R. Cipolla. "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[80] A. G. Kendall. "Geometry and uncertainty in deep learning for computer vision". PhD thesis. University of Cambridge, 2019.

[81] J. D. M.-W. C. Kenton and L. K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*. 2019.

[82] R. Kesten, M. Usman, J. Houston, T. Pandya, K. Nadhamuni, A. Ferreira, M. Yuan, B. Low, A. Jain, P. Ondruska, S. Omari, S. Shah, A. Kulkarni, A. Kazakova, C. Tao, L. Platinsky, W. Jiang, and V. Shet. *Level 5 Perception Dataset 2020*. `https://level-5.global/level5/data/`. 2019.

[83] Y. Kim and K. Dongsuk. "Deep Learning based Vehicle Position and Orientation Estimation via Inverse Perspective Mapping Image". In: *IEEE Symposium on Intelligent Vehicle (IV)*. 2019.

[84] J. Ku, A. Harakeh, and S. L. Waslander. "In defense of classical image processing: Fast depth completion on the cpu". In: *Conference on Computer and Robot Vision (CRV)*. 2018.

[85] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander. "Joint 3d proposal generation and object detection from view aggregation". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2018.

[86] J. Ku, A. D. Pon, and S. L. Waslander. "Monocular 3d object detection leveraging accurate proposals and shape reconstruction". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[87] A. Kumar, G. Brazil, E. Corona, A. Parchami, and X. Liu. “Deviant: Depth equivariant network for monocular 3d object detection”. In: *European Conference on Computer Vision (ECCV)*. 2022.

[88] A. Kumar, G. Brazil, and X. Liu. “GrooMeD-NMS: Grouped Mathematically Differentiable NMS for Monocular 3D Object Detection”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[89] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard. “g2o: A General Framework for Graph Optimization”. In: *International Conference on Robotics and Automation (ICRA)*. 2011.

[90] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. “Pointpillars: Fast encoders for object detection from point clouds”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[91] H. Law and J. Deng. “CornerNet: Detecting Objects as Paired Keypoints”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[92] D.-H. Lee et al. “Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks”. In: *International Conference on Machine Learning Workshop (ICMLW)*. 2013.

[93] J. H. Lee, M.-K. Han, D. W. Ko, and I. H. Suh. “From big to small: Multi-scale local planar guidance for monocular depth estimation”. In: *arXiv preprint arXiv:1907.10326* (2019).

[94] B. Li. “3d fully convolutional network for vehicle detection in point cloud”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2017.

[95] B. Li, W. Ouyang, L. Sheng, X. Zeng, and X. Wang. “GS3D: An Efficient 3D Object Detection Framework for Autonomous Driving”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[96] C. Li, J. Ku, and S. Waslander. “Confidence Guided Stereo 3D Object Detection with Split Depth Estimation”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2020.

[97] P. Li, X. Chen, and S. Shen. “Stereo R-CNN Based 3D Object Detection for Autonomous Driving”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[98] P. Li and H. Zhao. “Monocular 3D Detection With Geometric Constraint Embedding and Semi-Supervised Training”. In: *IEEE Robotics and Automation Letters (RA-L)* (2021).

[99] P. Li, H. Zhao, P. Liu, and F. Cao. “RTM3D: Real-Time Monocular 3D Detection from Object Keypoints for Autonomous Driving”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[100] S. Li, Z. Yan, H. Li, and K.-T. Cheng. “Exploring intermediate representation for monocular vehicle pose estimation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[101] Q. Lian, B. Ye, R. Xu, W. Yao, and T. Zhang. “Geometry-aware data augmentation for monocular 3D object detection”. In: *arXiv preprint arXiv:2104.05858* (2021).

[102] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun. “Multi-task multi-sensor fusion for 3d object detection”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[103] M. Liang, B. Yang, S. Wang, and R. Urtasun. “Deep Continuous Fusion for Multi-Sensor 3D Object Detection”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[104] M. Liang, B. Yang, S. Wang, and R. Urtasun. “Deep continuous fusion for multi-sensor 3d object detection”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[105] Y. Liao, J. Xie, and A. Geiger. “KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D”. In: *arXiv preprint arXiv:2109.13410* (2021).

[106] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. “Feature pyramid networks for object detection”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[107] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. “Focal loss for dense object detection”. In: *International Conference on Computer Vision (ICCV)*. 2017.

[108] C. Liu, S. Gu, L. Van Gool, and R. Timofte. “Deep Line Encoding for Monocular 3D Object Detection and Depth Prediction”. In: *British Machine Vision Conference (BMVC)*. 2021.

[109] S. Liu, D. Huang, and Y. Wang. “Adaptive nms: Refining pedestrian detection in a crowd”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[110] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. “Ssd: Single shot multibox detector”. In: *European Conference on Computer Vision (ECCV)*. 2016.

[111] X. Liu, N. Xue, and T. Wu. “Learning Auxiliary Monocular Contexts Helps Monocular 3D Object Detection”. In: *Association for the Advancement of Artificial Intelligence (AAAI)*. 2022.

[112] Y. Liu, Y. Yixuan, and M. Liu. “Ground-aware monocular 3d object detection for autonomous driving”. In: *IEEE Robotics and Automation Letters (RA-L)* (2021).

[113] Z. Liu, Z. Wu, and R. Toth. “SMOKE: Single-Stage Monocular 3D Object Detection via Keypoint Estimation”. In: *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020.

[114] Z. Liu, D. Zhou, F. Lu, J. Fang, and L. Zhang. “AutoShape: Real-Time Shape-Aware Monocular 3D Object Detection”. In: *International Conference on Computer Vision (ICCV)*. 2021.

[115] Y. Lu, X. Ma, L. Yang, T. Zhang, Y. Liu, Q. Chu, J. Yan, and W. Ouyang. “Geometry Uncertainty Projection Network for Monocular 3D Object Detection”. In: *International Conference on Computer Vision (ICCV)*. 2019.

[116] S. Luo, H. Dai, L. Shao, and Y. Ding. “M3DSSD: Monocular 3D single stage object detector”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[117] W. Luo, B. Yang, and R. Urtasun. “Fast and Furious: Real Time End-to-End 3D Detection, Tracking and Motion Forecasting With a Single Convolutional Net”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[118] Z. Luo, Z. Cai, C. Zhou, G. Zhang, H. Zhao, S. Yi, S. Lu, H. Li, S. Zhang, and Z. Liu. “Unsupervised Domain Adaptive 3D Detection With Multi-Level Consistency”. In: *International Conference on Computer Vision (ICCV)*. 2021.

[119] X. Ma, S. Liu, Z. Xia, H. Zhang, X. Zeng, and W. Ouyang. “Rethinking Pseudo-LiDAR Representation”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[120] X. Ma, W. Ouyang, A. Simonelli, and E. Ricci. “3D object detection from images for autonomous driving: A survey”. In: *arXiv preprint arXiv:2202.02980* (2022).

[121] X. Ma, Z. Wang, H. Li, P. Zhang, W. Ouyang, and X. Fan. “Accurate Monocular 3D Object Detection via Color-Embedded 3D Reconstruction for Autonomous Driving”. In: *International Conference on Computer Vision (ICCV)*. 2019.

[122] X. Ma, Y. Zhang, D. Xu, D. Zhou, S. Yi, H. Li, and W. Ouyang. “Delving Into Localization Errors for Monocular 3D Object Detection”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[123] Y. Ma, T. Wang, X. Bai, H. Yang, Y. Hou, Y. Wang, Y. Qiao, R. Yang, D. Manocha, and X. Zhu. “Vision-centric bev perception: A survey”. In: *arXiv preprint arXiv:2208.02797* (2022).

[124] F. Manhardt, W. Kehl, and A. Gaidon. “ROI-10D: Monocular Lifting of 2D Detection to 6D Pose and Metric Shape”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[125] D. Maturana and S. Scherer. “Voxnet: A 3d convolutional neural network for real-time object recognition”. In: *International Conference on Intelligent Robots and Systems (IROS)*. 2015.

[126] N. Mayer, E. Ilg, P. Hausser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. “A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[127] B. Mildenhall, P. Srinivasan, M. Tancik, J. Barron, R. Ramamoorthi, and R. Ng. “NeRF: Representing Scenes as Neural Radiance Fields for View Synthesis”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[128] J. J. Moré. “The Levenberg-Marquardt algorithm: implementation and theory”. In: *Numerical analysis*. 1978.

[129] A. Mousavian, D. Anguelov, J. Flynn, and J. Košecká. “3d bounding box estimation using deep learning and geometry”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[130] R. Nabati and H. Qi. “CenterFusion: Center-Based Radar and Camera Fusion for 3D Object Detection”. In: *Winter Conference on Applications of Computer Vision (WACV)*. 2021.

[131] A. Naiden, V. Paunescu, G. Kim, B. Jeon, and M. Leordeanu. “Shift r-cnn: Deep monocular 3d object detection with closed-form geometric constraints”. In: *International Conference on Image Processing (ICIP)*. 2019.

[132] M. Niemeyer and A. Geiger. “GIRAFFE: Representing Scenes as Compositional Generative Neural Feature Fields”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[133] C. Ning, H. Zhou, Y. Song, and J. Tang. “Inception single shot multibox detector for object detection”. In: *International Conference on Multimedia and Expo Workshops (ICMEW)*. 2017.

[134] J. Ost, F. Mannan, N. Thuerey, J. Knodt, and F. Heide. “Neural Scene Graphs for Dynamic Scenes”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[135] W. Ouyang, X. Zeng, X. Wang, S. Qiu, P. Luo, Y. Tian, H. Li, S. Yang, Z. Wang, H. Li, et al. “DeepID-Net: Object detection with deformable part based convolutional neural networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2016).

[136] D. Park, R. Ambrus, V. Guizilini, J. Li, and A. Gaidon. “Is Pseudo-Lidar needed for Monocular 3D Object detection?” In: *International Conference on Computer Vision (ICCV)*. 2021.

[137] D. Park, R. Ambrus, V. Guizilini, J. Li, and A. Gaidon. “Is Pseudo-Lidar needed for Monocular 3D Object detection?” In: *International Conference on Computer Vision (ICCV)*. 2021.

[138] A. Patil, S. Malla, H. Gang, and Y.-T. Chen. “The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes”. In: *International Conference on Robotics and Automation (ICRA)*. 2019.

[139] L. Peng, F. Liu, Z. Yu, S. Yan, D. Deng, Z. Yang, H. Liu, and D. Cai. “Lidar point cloud guided monocular 3d object detection”. In: *European Conference on Computer Vision (ECCV)*. 2022.

[140] L. Peng, X. Wu, Z. Yang, H. Liu, and D. Cai. “DID-M3D: Decoupling Instance Depth for Monocular 3D Object Detection”. In: *European Conference on Computer Vision (ECCV)*. 2022.

[141] L. Peng, S. Yan, B. Wu, Z. Yang, X. He, and D. Cai. “WeakM3D: Towards Weakly Supervised Monocular 3D Object Detection”. In: *International Conference on Learning Representations (ICLR)*. 2022.

[142] W. Peng, H. Pan, H. Liu, and Y. Sun. “Ida-3d: Instance-depth-aware 3d object detection from stereo vision for autonomous driving”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[143] Q.-H. Pham, P. Sevestre, R. S. Pahwa, H. Zhan, C. H. Pang, Y. Chen, A. Mustafa, V. Chandrasekhar, and J. Lin. “A*3D dataset: Towards autonomous driving in challenging environments”. In: *International Conference on Robotics and Automation (ICRA)*. 2020.

[144] J. Philion and S. Fidler. “Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d”. In: *European Conference on Computer Vision (ECCV)*. 2020.

[145] A. D. Pon, J. Ku, C. Li, and S. L. Waslander. “Object-centric stereo matching for 3d object detection”. In: *International Conference on Robotics and Automation(ICRA)*. 2020.

[146] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas. “Frustum pointnets for 3d object detection from rgb-d data”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[147] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[148] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. “Pointnet++: Deep hierarchical feature learning on point sets in a metric space”. In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2017.

[149] K. Qian, S. Zhu, X. Zhang, and L. E. Li. "Robust Multimodal Vehicle Detection in Foggy Weather Using Complementary Lidar and Radar Signals". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[150] R. Qian, D. Garg, Y. Wang, Y. You, S. Belongie, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao. "End-to-End Pseudo-LiDAR for Image-Based 3D Object Detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[151] R. Qian, X. Lai, and X. Li. "3d object detection for autonomous driving: a survey". In: *PR* (2022).

[152] Z. Qin, J. Wang, and Y. Lu. "Monogrnet: A geometric reasoning network for monocular 3d object localization". In: *Association for the Advancement of Artificial Intelligence (AAAI)*. 2019.

[153] Z. Qin, J. Wang, and Y. Lu. "Triangulation Learning Network: From Monocular to Stereo 3D Object Detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[154] A. Radford, L. Metz, and S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks". In: *arXiv preprint arXiv:1511.06434* (2015).

[155] C. Reading, A. Harakeh, J. Chae, and S. L. Waslander. "Categorical Depth Distribution Network for Monocular 3D Object Detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[156] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[157] J. Ren, X. Chen, J. Liu, W. Sun, J. Pang, Q. Yan, Y.-W. Tai, and L. Xu. "Accurate Single Stage Detector Using Recurrent Rolling Convolution". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[158] S. Ren, K. He, R. Girshick, and J. Sun. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2015.

[159] Z. Ren and E. B. Sudderth. "Three-dimensional object detection and layout prediction using clouds of oriented gradients". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[160] Z. Ren, I. Misra, A. G. Schwing, and R. Girdhar. "3D Spatial Recognition without Spatially Labeled 3D". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[161] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese. "Generalized intersection over union: A metric and a loss for bounding

box regression". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[162] T. Roddick, A. Kendall, and R. Cipolla. "Orthographic feature transform for monocular 3d object detection". In: *arXiv preprint arXiv:1811.08188* (2018).

[163] H. Schütze, C. D. Manning, and P. Raghavan. *Introduction to information retrieval*. Cambridge University Press Cambridge, 2008.

[164] S. Shi, C. Guo, L. Jiang, Z. Wang, J. Shi, X. Wang, and H. Li. "PV-RCNN: Point-Voxel Feature Set Abstraction for 3D Object Detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[165] S. Shi, X. Wang, and H. Li. "Pointrcnn: 3d object proposal generation and detection from point cloud". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[166] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li. "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2020).

[167] X. Shi, Z. Chen, and T.-K. Kim. "Distance-Normalized Unified Representation for Monocular 3D Object Detection". In: *European Conference on Computer Vision (ECCV)*. 2020.

[168] X. Shi, Q. Ye, X. Chen, C. Chen, Z. Chen, and T. K. Kim. "Geometry-based Distance Decomposition for Monocular 3D Object Detection". In: *International Conference on Computer Vision (ICCV)*. 2021.

[169] A. Shrivastava, A. Gupta, and R. Girshick. "Training Region-Based Object Detectors With Online Hard Example Mining". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[170] A. Simonelli, S. R. Bulo, L. Porzi, M. L. Antequera, and P. Kontschieder. "Disentangling monocular 3d object detection: From single to multi-class recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2020).

[171] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder. "Disentangling monocular 3d object detection". In: *International Conference on Computer Vision (ICCV)*. 2019.

[172] A. Simonelli, S. R. Bulò, L. Porzi, P. Kontschieder, and E. Ricci. "Are We Missing Confidence in Pseudo-LiDAR Methods for Monocular 3D Object Detection?" In: *International Conference on Computer Vision (ICCV)*. 2021.

[173] A. Simonelli, S. R. Bulò, L. Porzi, P. Kontschieder, and E. Ricci. "Are we Missing Confidence in Pseudo-LiDAR Methods for Monocular 3D Object Detection?" In: *International Conference on Computer Vision (ICCV)*. 2021.

[174] A. Simonelli, S. R. Bulò, L. Porzi, E. Ricci, and P. Kontschieder. "Towards Generalization Across Depth for Monocular 3D Object Detection". In: *European Conference on Computer Vision (ECCV)*. 2020.

[175] K. Simonyan and A. Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[176] S. Song and J. Xiao. "Sliding shapes for 3d object detection in depth images". In: *European Conference on Computer Vision (ECCV)*. 2014.

[177] X. Song, P. Wang, D. Zhou, R. Zhu, C. Guan, Y. Dai, H. Su, H. Li, and R. Yang. "ApolloCar3D: A Large 3D Car Instance Understanding Benchmark for Autonomous Driving". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[178] S. Srivastava, F. Jurie, and G. Sharma. "Learning 2D to 3D Lifting for Object Detection in 3D for Autonomous Vehicles". In: *International Conference on Intelligent Robots and Systems (IROS)*. 2019.

[179] J. Sun, L. Chen, Y. Xie, S. Zhang, Q. Jiang, X. Zhou, and H. Bao. "Disp r-cnn: Stereo 3d object detection via shape prior guided instance disparity estimation". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[180] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, et al. "Scalability in perception for autonomous driving: Waymo open dataset". In: *International Conference on Computer Vision (ICCV)*. 2020.

[181] *Tesla AI Day 2021*. 2021.

[182] *Tesla AI Day 2022*. 2022.

[183] Z. Tian, C. Shen, H. Chen, and T. He. "Fcos: Fully convolutional one-stage object detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[184] H. Tu, S. Peng, V. Leung, and R. Gao. "SoK: Vehicle Orientation Representations for Deep Rotation Estimation". In: *arXiv preprint arXiv:2112.04421* (2021).

[185] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. "Selective search for object recognition". In: *International Journal of Computer Vision (IJCV)* (2013).

[186] L. Wang, L. Du, X. Ye, Y. Fu, G. Guo, X. Xue, J. Feng, and L. Zhang. "Depth-Conditioned Dynamic Message Propagation for Monocular 3D

Object Detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[187] L. Wang, L. Du, X. Ye, Y. Fu, G. Guo, X. Xue, J. Feng, and L. Zhang. "Depth-conditioned Dynamic Message Propagation for Monocular 3D Object Detection". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[188] L. Wang, L. Zhang, Y. Zhu, Z. Zhang, T. He, M. Li, and X. Xue. "Progressive Coordinate Transforms for Monocular 3D Object Detection". In: *Annual Conference on Neural Information Processing Systems (NeurIPS)*. 2021.

[189] T. Wang, X. Zhu, J. Pang, and D. Lin. "FCOS3D: Fully Convolutional One-Stage Monocular 3D Object Detection". In: *International Conference on Computer Vision Workshops (ICCVW)*. 2021.

[190] T. Wang, X. Zhu, J. Pang, and D. Lin. "Probabilistic and Geometric Depth: Detecting Objects in Perspective". In: *Conference on Robot Learning (CoRL)*. 2021.

[191] X. Wang, W. Yin, T. Kong, Y. Jiang, L. Li, and C. Shen. "Task-aware monocular depth estimation for 3d object detection". In: *Association for the Advancement of Artificial Intelligence (AAAI)*. 2020.

[192] Y. Wang, W.-L. Chao, D. Garg, B. Hariharan, M. Campbell, and K. Q. Weinberger. "Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[193] Y. Wang, X. Chen, Y. You, L. E. Li, B. Hariharan, M. Campbell, K. Q. Weinberger, and W.-L. Chao. "Train in Germany, Test in the USA: Making 3D Object Detectors Generalize". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[194] Y. Wang, V. Guizilini, T. Zhang, Y. Wang, H. Zhao, and J. Solomon. "DETR 3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries". In: *Conference on Robot Learning (CoRL)*. 2021.

[195] X. Weng and K. Kitani. "Monocular 3D Object Detection with Pseudo-LiDAR Point Cloud". In: *International Conference on Computer Vision Workshops (ICCVW)*. 2019.

[196] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon. "Cbam: Convolutional block attention module". In: *European Conference on Computer Vision (ECCV)*. 2018.

[197] Y. Xiang, W. Choi, Y. Lin, and S. Savarese. "Data-driven 3d voxel patterns for object category recognition". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015.

[198] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. “Aggregated residual transformations for deep neural networks”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[199] B. Xu and Z. Chen. “Multi-Level Fusion Based 3D Object Detection From Monocular Images”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[200] D. Xu, W. Ouyang, X. Wang, and N. Sebe. “PAD-Net: Multi-Tasks Guided Prediction-and-Distillation Network for Simultaneous Depth Estimation and Scene Parsing”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[201] D. Xu, E. Ricci, W. Ouyang, X. Wang, and N. Sebe. “Monocular depth estimation using multi-scale continuous crfs as sequential deep networks”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2018).

[202] J. Xu, Y. Ma, S. He, and J. Zhu. “3D-GIoU: 3D Generalized Intersection over Union for Object Detection in Point Cloud”. In: *Sensors* (2019).

[203] Q. Xu, Y. Zhou, W. Wang, C. R. Qi, and D. Anguelov. “Grid-gcn for fast and scalable point cloud learning”. In: *International Conference on Computer Vision (ICCV)*. 2021.

[204] Z. Xu, W. Zhang, X. Ye, X. Tan, W. Yang, S. Wen, E. Ding, A. Meng, and L. Huang. “ZoomNet: Part-Aware Adaptive Zooming Neural Network for 3D Object Detection”. In: *Association for the Advancement of Artificial Intelligence (AAAI)*. 2020.

[205] Y. Yan, Y. Mao, and B. Li. “Second: Sparsely embedded convolutional detection”. In: *Sensors* (2018).

[206] B. Yang, W. Luo, and R. Urtasun. “PIXOR: Real-Time 3D Object Detection From Point Clouds”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[207] J. Yang, S. Shi, Z. Wang, H. Li, and X. Qi. “ST3D: Self-Training for Unsupervised Domain Adaptation on 3D Object Detection”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*.

[208] Y. Yao, Z. Luo, S. Li, T. Fang, and L. Quan. “Mvsnet: Depth inference for unstructured multi-view stereo”. In: *European Conference on Computer Vision (ECCV)*. 2018.

[209] Y. You, K. Z. Luo, X. Chen, J. Chen, W.-L. Chao, W. Sun, B. Hariharan, M. Campbell, and K. Q. Weinberger. “Hindsight is 20/20: Leveraging Past Traversals to Aid 3D Perception”. In: *International Conference on Learning Representations (ICLR)*. 2022.

[210] Y. You, Y. Wang, W.-L. Chao, D. Garg, G. Pleiss, B. Hariharan, M. Campbell, and K. Q. Weinberger. “Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving”. In: 2020.

[211] F. Yu, D. Wang, E. Shelhamer, and T. Darrell. “Deep Layer Aggregation”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[212] A. R. Zamir, A. Sax, W. Shen, L. J. Guibas, J. Malik, and S. Savarese. “Taskonomy: Disentangling task transfer learning”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[213] G. Zhang, J. Jia, T.-T. Wong, and H. Bao. “Consistent depth maps recovery from a video sequence”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (T-PAMI)* (2009).

[214] R. Zhang, H. Qiu, T. Wang, X. Xu, Z. Guo, Y. Qiao, P. Gao, and H. Li. “Monodetr: Depth-aware transformer for monocular 3d object detection”. In: *arXiv preprint arXiv:2203.13310* (2022).

[215] Y. Zhang, X. Ma, S. Yi, J. Hou, Z. Wang, W. Ouyang, and D. Xu. “Learning Geometry-Guided Depth via Projective Modeling for Monocular 3D Object Detection”. In: *arXiv preprint arXiv:2107.13931* (2021).

[216] Y. Zhang and Q. Yang. “A survey on multi-task learning”. In: *IEEE Transactions on Knowledge and Data Engineering (T-KDE)* (2021).

[217] Y. Zhang, J. Lu, and J. Zhou. “Objects Are Different: Flexible Monocular 3D Object Detection”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[218] N. Zhao, T.-S. Chua, and G. H. Lee. “Sess: Self-ensembling semi-supervised 3d object detection”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[219] D. Zhou, J. Fang, X. Song, C. Guan, J. Yin, Y. Dai, and R. Yang. “Iou loss for 2d/3d object detection”. In: *International Conference on 3D Vision (3DV)*. 2019.

[220] D. Zhou, X. Song, Y. Dai, J. Yin, F. Lu, M. Liao, J. Fang, and L. Zhang. “IAFA: Instance-Aware Feature Aggregation for 3D Object Detection from a Single Image”. In: *Asian Conference on Computer Vision (ACCV)*. 2020.

[221] T. Zhou, M. Brown, N. Snavely, and D. G. Lowe. “Unsupervised learning of depth and ego-motion from video”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017.

[222] X. Zhou, D. Wang, and P. Krähenbühl. “Objects as Points”. In: *arXiv preprint arXiv:1904.07850*. 2019.

[223] Y. Zhou and O. Tuzel. “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2018.

[224] Y. Zhou, Y. He, H. Zhu, C. Wang, H. Li, and Q. Jiang. “Monocular 3D Object Detection: An Extrinsic Parameter Free Approach”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021.

[225] Z. Zhou, L. Du, X. Ye, Z. Zou, X. Tan, E. Ding, L. Zhang, X. Xue, and J. Feng. “SGM3D: Stereo Guided Monocular 3D Object Detection”. In: *arXiv preprint arXiv:2112.01914* (2021).

[226] X. Zhu, H. Hu, S. Lin, and J. Dai. “Deformable ConvNets V2: More Deformable, Better Results”. In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[227] X. Zhu, Y. Wang, J. Dai, L. Yuan, and Y. Wei. “Flow-guided feature aggregation for video object detection”. In: *International Conference on Computer Vision (ICCV)*. 2017.

[228] C. L. Zitnick and P. Dollár. “Edge boxes: Locating object proposals from edges”. In: *European Conference on Computer Vision (ECCV)*. 2014.

[229] Z. Zou, X. Ye, L. Du, X. Cheng, X. Tan, L. Zhang, J. Feng, X. Xue, and E. Ding. “The devil is in the task: Exploiting reciprocal appearance-localization features for monocular 3d object detection”. In: *International Conference on Computer Vision (ICCV)*. 2021.