國立臺灣大學電機資訊學院電信工程學研究所

碩士論文

Graduate Institute of Communication Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

駕駛場景中的影像辨識：三維物件辨識與影像分割

Image Recognition in Driving Scene: Monocular 3D Object Detection and Image Segmentation

游家權

Jia-Quan Yu

指導教授: 貝蘇章 博士

Advisor: Soo-Chang Pei, Ph.D.

中華民國 112 年 5 月

May, 2023

doi:10.6342/NTU202301439

國立臺灣大學碩士學位論文
口試委員會審定書

MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

駕駛場景中的影像辨識：三維物件辨識與影像分割

Image Recognition in Driving Scene:
Monocular 3D Object Detection and Image Segmentation

本論文係游家權君(R10942152)在國立臺灣大學電信工程研究所之碩士學位論文，於民國 112 年 5 月 20 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Institute of Communication Engineering on 20 May 2023 have examined a Master's thesis entitled above presented by Jia-Quan, Yu (R10942152) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

貝蘇章	羅國富	曾定詩
（指導教授 Advisor）		
杭學鳴	丁建均	

系主任/所長 Director: 陳錫坤

Acknowledgements

This thesis has been written under the guidance of Prof. Soo-Chang Pei. His expertise, valuable feedback in weekly meeting, and enthusiastic research approach have played a crucial role in shaping the innovative ideas presented in this work. I would also like to thank my labmates, Yi-Hsuan Wu and Chih-Yao Chang, for their constant support and assistance during the writing process. The research papers and online resources they shared have helped greatly on implementing the new method present in this thesis. Additionally, I would like to thank our lab assistant, Yi-Zi Zou, for her continuous support throughout the two-year master's degree journey, allowing me to focus on other essential aspects of my research. Lastly, I want to express my gratitude to OpenAI and all the developers involved in creating ChatGPT, an incredibly useful language generative model. ChatGPT has been used in this thesis to refine grammar, correct typos, adjust typesetting, and rephrase sentences to enhance clarity and readability. The time saved by the ChatGPT has allowed me to allocate more time towards research activities, resulting in a more comprehensive and polished thesis.

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

摘要

影像辨識演算法在自駕車領域中扮演著關鍵的角色，它使得計算機能夠感知周圍環境並確保駕駛安全。本研究聚焦於自動駕駛場景中的物件檢測和影像分割任務的相關深度學習技術，針對三維物件辨識，本研究提出基於場景的數據擴增方法、錨點採樣技巧以及能夠改進特徵提取能力的相機視角卷積模組，本研究於 KITTI3D 數據集的測試中取得了顯著的 23.9% 平均準確度。此外，針對影像分割任務，我們提出了整合深度估計和全景分割的新任務並開發了 Panoptic-DepthLab 網絡，以實現分割和深度估計任務的共同訓練。本研究旨在提供自動駕駛領域中的影像識別演算法可能的改進方向，並設法提高物件檢測和影像分割的準確性和效率。

關鍵字：三維物件辨識，深度估計，駕駛場景，實例分割，物件辨識，全景分割，安全性指標，語意分割

doi:10.6342/NTU202301439

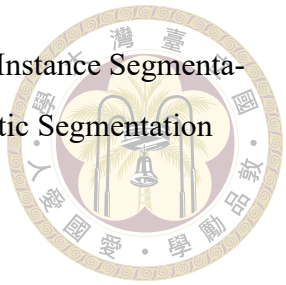
doi:10.6342/NTU202301439

Abstract

Image recognition plays a crucial role in autonomous driving, enabling computers to perceive the surrounding environment and ensure driving safety. In this thesis, we focus on both detection and segmentation tasks in driving scenarios, leveraging deep-learning techniques. Our work introduces novel approaches to enhance the performance of 3D object detection, including a scene-aware data augmentation method, a depth-aware anchor sampling technique, and a perspective-aware convolutional module that improves feature extraction capabilities. By combining these methods, we achieve a significant AP of 23.9% on the easy benchmark of the KITTI3D dataset. Furthermore, we propose a novel integration of depth estimation and panoptic segmentation on a color map. We also introduce the Panoptic-DepthLab network, which enables joint training of segmentation and depth estimation tasks. Our research aims to advance the field of image recognition in autonomous driving, improving the accuracy and efficiency of object detection and segmentation algorithms.

doi:10.6342/NTU202301439

Keywords: 3D Object Detection, Depth Estimation, Driving Scene, Instance Segmentation, Object Detection, Panoptic Segmentation, Safety Metric, Semantic Segmentation



doi:10.6342/NTU202301439

Contents

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

List of Figures

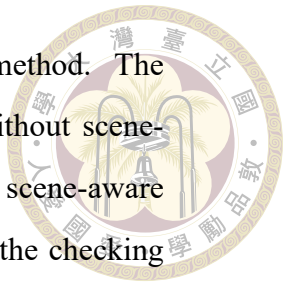
doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439



doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

List of Tables

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

Chapter 1 Introduction

Image recognition has become increasingly important in many fields, such as robotics, autonomous driving vehicle, and augmented reality. In recent years, the emergence of deep learning techniques has taken image recognition to the next level, achieving state-of-the-art performance in many tasks and revolutionizing the research paradigm of Computer Vision algorithms. Therefore, in this thesis, we focus on deep-learning-based image recognition algorithms for driving scene applications, with a specific focus on two main tasks: object detection and image segmentation.

Object detection is the process of identifying and locating objects of interest within an image. For example, cameras mounted on autonomous driving cars need to detect and recognize obstacles such as pedestrians, cyclists, and other cars. This typically involves using a rectangular bounding box to mark the boundary of the object and classifying it into the correct category. Furthermore, self-driving cars require information about the distance, orientation, and location of on-road obstacles, which falls under the domain of 3D object detection. However, monocular object detection is often considered an ill-posed problem due to the challenges of obtaining reliable estimations with limited input information. Despite this, accurate object detection algorithms offer immense potential benefits, ranging from improving public safety to optimizing industrial manufacturing processes.

doi:10.6342/NTU202301439

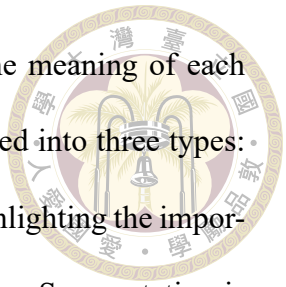


Image segmentation, on the other hand, involves identifying the meaning of each pixel in an image in a fine-grained manner. It can be broadly classified into three types: semantic, instance, and panoptic segmentation. This task involves highlighting the important pixels in the image and classifying them correctly into categories. Segmentation is a fundamental image recognition task with wide-ranging applications, such as in medical image recognition.

In this thesis, our focus is on researching object detection, monocular 3D object detection, and image segmentation algorithms and their applications in autonomous driving scenarios. Our main contributions can be summarized as follows:

- We propose a novel perspective-aware convolution layer that can extract long-range dependency features from the image. By leveraging prior knowledge of the scene structure, we are able to improve the performance of anchor-based 3D object detectors on the KITTI dataset.

- We propose a scene-aware copy-paste data augmentation method customized for 3D object detection. We use a pretrained depth estimation model to obtain scene information, which we then use to paste additional objects into appropriate locations. We emphasize the importance of considering the scene when pasting objects for data augmentation.

- We propose an innovative network that integrates depth estimation and segmentation tasks, allowing both tasks to be trained together. We evaluate the performance of our approach on the Cityscape dataset. Our research provides a foundation for further advancements in image segmentation algorithms for driving scenes.

- We propose a safety metric for 3D object detection in driving scenes, emphasizing

doi:10.6342/NTU202301439

the importance of taking closer objects into consideration. This metric can help improve the safety of autonomous driving systems.

The reminder of the thesis is structured as follows. In Chapter 2, we introduce object detection algorithms. Chapter 3 discusses data augmentation techniques and ours scene-aware copy-paste method. Chapter 4 introduces 3D object detection algorithms and our proposed perspective-aware convolution. In Chapter 5, we explore image segmentation tasks and introduce Panoptic-DepthLab, our proposed network for panoptic segmentation with depth. In Chapter 6, we propose our safety AP metric, which is suitable for use in driving scenes. Finally, we conclude our research in Chapter 7.

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

Chapter 2 Object Detection

Object detection is a fundamental task in Computer Vision that has been extensively researched. The goal of object detection algorithms is to take a single image as input and output a set of bounding boxes that accurately localize and classify objects in the image. A bounding box is a rectangle box and is defined by four variables (x, y, w, h), where (x, y) is the rectangle center in image coordinate and (w, h) is the box width and box height in pixel unit. In addition, a confidence score s is assigned to each bounding box to reflect the algorithm's certainty of the detection and class prediction c to show the categorize result. To sum up, the object detection algorithm outputs an arbitrary number of bounding boxes to capture objects present in the image, and each detection is defined by six variables: (x, y, w, h, c, s). The bounding box definition is shown in Figure 2.1.

2.1 Evaluation Metric - Average Precision

To evaluate the performance of an object detection algorithm, we use average precision (AP) to evaluate how well the detection results match the ground truth. Typically, we use the term "ground truth" to address boxes that is provided in label file and use " predicted box" to address boxes that generate by the algorithm. In the section, we explain how to calculate average precision given a set of ground truth and predicted boxes.

doi:10.6342/NTU202301439

Figure 2.1: Object detection bounding box definition. The goal of object detection algorithms is to output an arbitrary number of rectangular bounding boxes to capture objects present in the image. Each detection is defined by six variables: (x, y, w, h, c, s), where (x, y) represents the center of the rectangle, (w, h) represents the width and height of the rectangle in pixel units, s is the confidence score reflecting the detection's confidence, and c represents the predicted category.

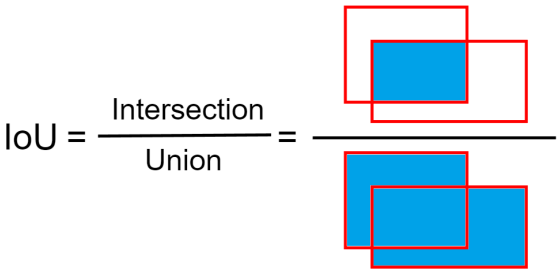


Figure 2.2: Intersection over Union (IoU) evaluates the similarity of two bounding boxes by dividing the intersection of the boxes by their union.

Firstly, we calculate the intersection over union (IoU) between all the predicted boxes and ground truth. Then, we assign a ground truth to the predicted box with the highest IoU with it and define it as a positive example if its IoU with the ground truth is higher than a threshold (usually 0.5), and vice versa. The assignment process and IoU calculation is depicted in Figure 2.3 and Figure 2.2. In this assignment result, we can find three statistical figures:

- True Positive (TP): A successful detection that matches a ground truth.
- False Positive (FP): A failed detection that does not match any ground truth.
- False Negative (FN): A ground truth that the detector fails to capture.

doi:10.6342/NTU202301439

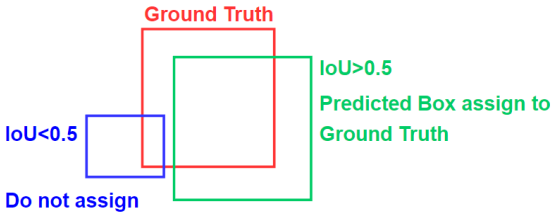


Figure 2.3: Predicted box assignment. Predicted boxes are assigned as to the ground truth if their Intersection over Union (IoU) with it is greater than 0.5.

Based on TP, FP, and FN, we calculate the precision and recall rate of the detection result as the following:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.1}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

Precision represents the percentage of correct detections, while recall represents the percentage of successfully captured ground truth objects. It is important to note that both precision and recall depend on the specified Intersection over Union (IoU) threshold and confidence scores. To create a metric that is independent of the confidence score criterion, we enumerate all possible confidence scores and calculate the precision and recall rates for each score threshold. We can then plot the results on a precision-recall curve (PR-curve) and calculate the average precision as the area under the curve. An example of a PR-curve is depicted in Figure 2.4.

Enumerating all possible confident thresholds to generate a complete precision-recall(PR) curve is computationally expensive, so most papers and popular datasets, such as PASCAL VOC, use 11-point Interpolated Average Precision (AP-11) metric. This method involves evenly sampling 11 points on the PR curve and using their average to represent the aver-

doi:10.6342/NTU202301439

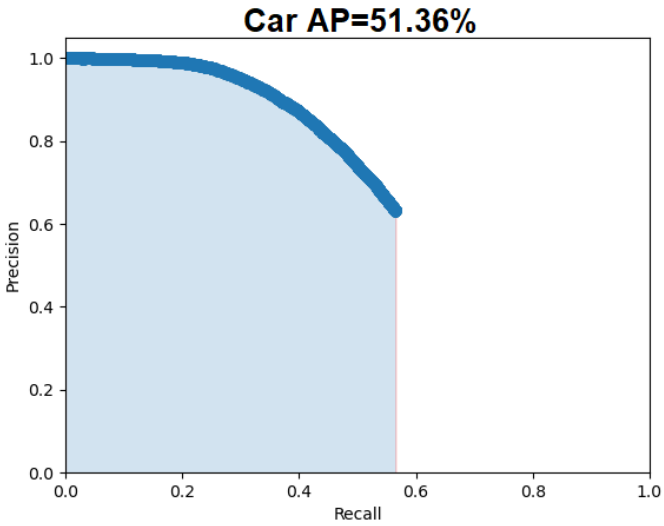


Figure 2.4: Precision-Recall Curve: This curve is obtained by enumerating all confidence score thresholds for the predictions. The light blue area under the curve represents 0.5136, which corresponds to an average precision of 51.36%. This metric is commonly used to evaluate the performance of object detectors.

age precision. AP-11 was used in the PASCAL VOC challenges between 2007 and 2010. However, recent studies [56] have suggested that AP-11 may not accurately represent the entire curve, and instead propose AP-41 as a more precise metric. The KITTI3D dataset has adopted AP-41 as its evaluation metric. In this thesis, we use AP-41 for all evaluations unless stated otherwise.

2.2 Related Work

2.2.1 Two-stage Object Detector

Object detection is a challenging task as it requires predicting an arbitrary number of objects in an image. To address this issue, a two-stage detector is a natural choice. In the first stage, the algorithm generates class-agnostic object proposals in the image, and in the second stage, it classifies the objects within each proposal. This localize-and-classify scheme was first introduced in R-CNN [24], the first two-stage object detector that uses

doi:10.6342/NTU202301439

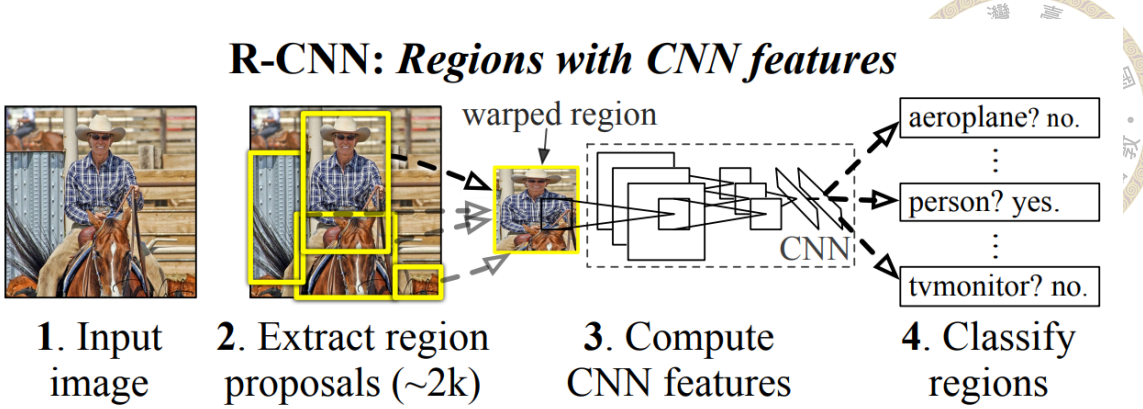


Figure 2.5: R-CNN[24] pipeline. It follows the detect-then-classify scheme, where the region of interest (RoI) is identified in the first stage, and the object is classified using a CNN network combined with SVM.

a convolutional neural network (CNN) as a feature extractor. R-CNN utilizes selective search [60] to produce 2k+ object proposals in each image, and each proposal is warped to a fixed size of 227x227 pixels, which is then processed by five convolutional layers and two fully connected layers to extract 4096-dimensional features. Finally, R-CNN uses a support vector machine (SVM) to classify the features into 1000 classes for the PASCAL VOC challenge. While R-CNN demonstrated the potential of CNNs to extract image features, it suffered from slow training and testing times.

Fast R-CNN [23], proposed shortly after R-CNN, aimed to address the slow training and testing time issues. Fast R-CNN used a Deep ConvNet to extract features from the image, and for each object proposal, it used an RoI-Pooling layer to extract fixed-length features from the shared feature map. This approach prevented the network from extracting duplicate features from nearby region proposals. Additionally, Fast R-CNN used a fully connected layer to replace the SVM classifier and used truncated SVD to speed up computation. Overall, Fast R-CNN integrated the network feature better and created an end-to-end architecture, making Fast R-CNN training nine times faster and testing 200 times faster than R-CNN.

doi:10.6342/NTU202301439

Despite many efforts to improve the network architecture, Fast R-CNN[23] still relied on selective search to produce possible object regions. Faster R-CNN[51] took a step further and used a fully convolutional network to build its own region proposal network (RPN). The RPN finds proposals by sliding a 3x3 window through the entire feature map and extracting 256-dimensional features from each location, which are then fed into a regressor to refine the bounding box and a classifier to predict the objectness score.

To address multi-scale objects, Faster R-CNN introduced anchors to the RPN. Anchors are pre-defined rectangular boxes with different sizes and aspect ratios, and each location that the sliding window passes through has the same set of anchors. These anchors serve as initial guesses of the size and aspect ratio of the object at that location. This design guarantees that the object detector is translation invariant, meaning that the detection head will extract the same feature regardless of the location of that pattern. With all these improvements, Faster R-CNN not only achieved higher accuracy than Fast R-CNN but also ran 10 times faster.

2.2.2 Single-stage Object Detector

In contrast to two-stage detectors, single-stage detectors consider object detection as a single regression problem, using a single and unified detector to achieve its goal. One of the classic networks in this line of work is YOLO[49], which achieves real-time object detection with high speed. YOLO divides the image into an S x S grid, with each grid responsible for detecting objects whose center lies within it and classifying it to the correct category. Each grid generates two bounding boxes and each bounding box needs to regress the dimension, location, and confidence score of it. YOLO uses 24 layers of a convolutional network to extract features and two layers of fully connected layers to

doi:10.6342/NTU202301439

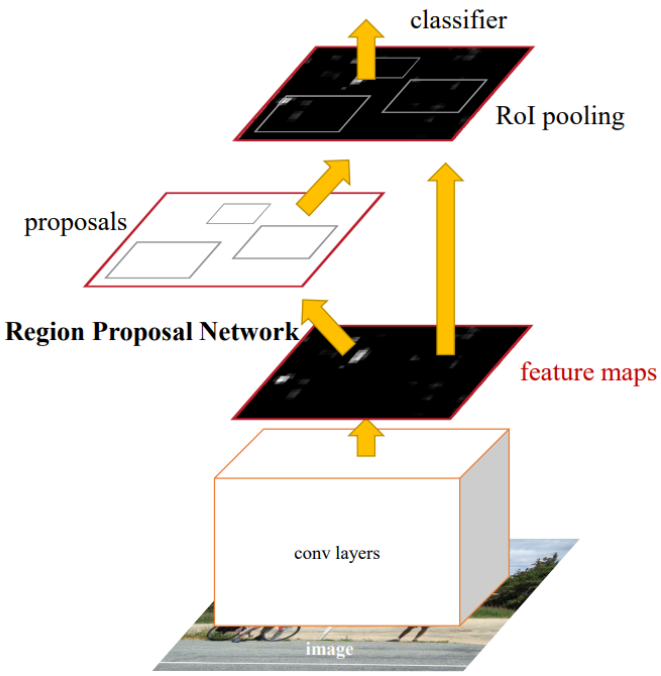


Figure 2.6: Faster R-CNN[51] architecture. The most innovative aspect of Faster R-CNN is the replacement of selective search with a Region Proposal Network (RPN), which uses predefined bounding box anchors to simplify the bounding box regression task.

regress and classify objects. However, despite its fast speed, YOLO has a low recall rate and produces many localization mistakes, particularly in crowded scenes, and is weak at detecting small objects because every predefined grid can only belong to a single category in its architecture.

Based on the observations of YOLO's limitations, YOLOv2[54] made several improvements. YOLOv2 introduced anchor boxes in its bounding box regression design and used k-means clustering to define the dimensions of the anchor boxes. This improvement greatly increased its recall rate. Secondly, YOLOv2 removed the fully connected layers to reduce the number of parameters and introduced its own backbone, called Darknet-19, which consists of 19 convolution layers and 5 max pooling layers. Lastly, YOLOv2 trained the network with different resolution images to force the network to generalize its ability to recognize objects of different sizes.

doi:10.6342/NTU202301439

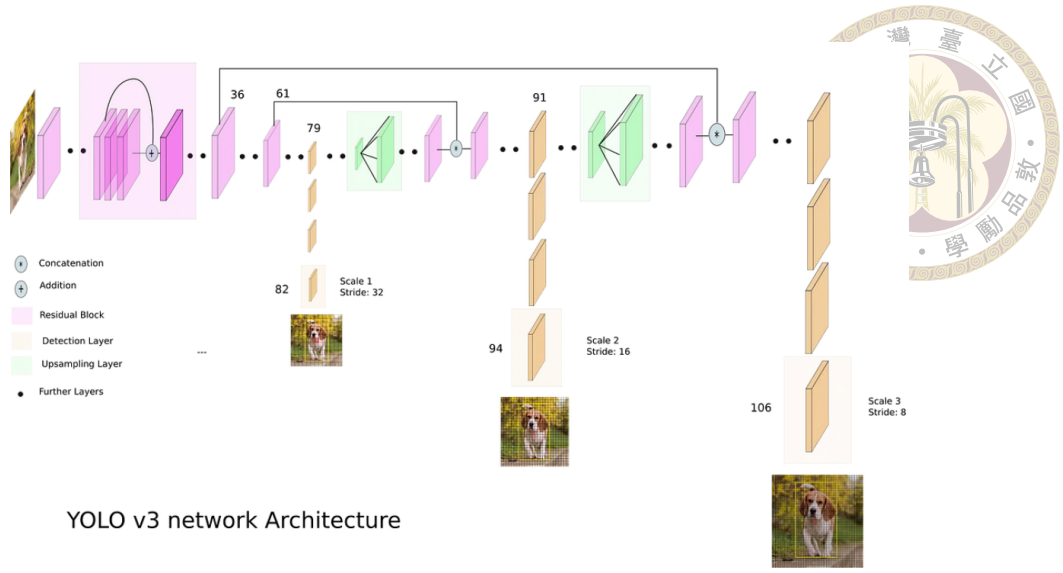


Figure 2.7: YOLOv3[50] network architecture. To address the limitation of its predecessors in detecting small objects, YOLOv3 uses three different scales of feature maps to detect objects of different sizes.

YOLOv3[50], as the next improved version of YOLOv2, developed a deeper feature extractor, Darknet-53, was introduced to replace Darknet-19. In addition, YOLOv3 takes multi-scale feature maps to predict bounding boxes at three different scales, which enables it to excel at small object detection. The network architecture of YOLOv3 also includes skip connections, allowing features at different scales to be combined and improving the model accuracy. These improvements result in significantly better detection performance, with YOLOv3 outperforming previous versions and achieving state-of-the-art results.

As for other line of work, SSD [40] is a popular one-stage object detection network that uses a modified VGG16 backbone to extract features from the input image. It utilizes multiple feature maps from the backbone, each of which has a different scale, and defines anchor boxes of various sizes on each of them. This allows SSD to detect objects of different sizes with high accuracy. To address the issue of imbalance between positive and negative samples, SSD employs hard negative mining, which involves selecting only the top negative examples for optimization to ensure a balance of positive and negative samples in the training process.

doi:10.6342/NTU202301439

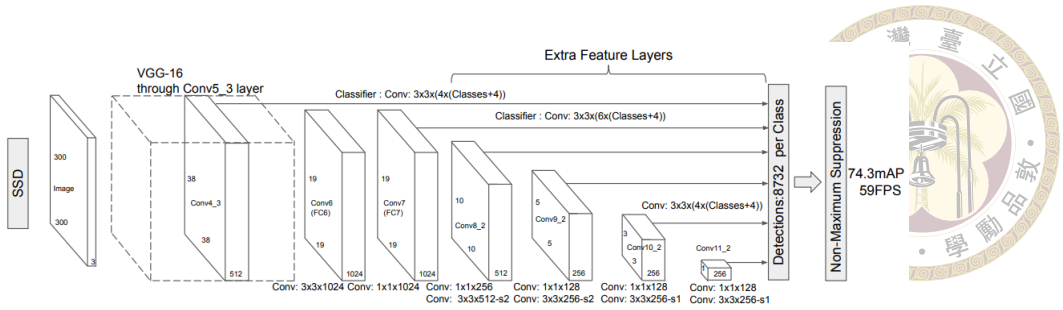


Figure 2.8: SSD[40] network architecture. SSD adopts VGG16 as the feature extractor and utilizes anchor techniques and hard negative example mining to improve training effectiveness.

SSD also employs data augmentation techniques such as random cropping to improve the robustness of the model. This approach has been shown to be effective, with SSD achieving high performance on benchmark datasets such as COCO and PASCAL VOC. Despite its success, SSD has some limitations, such as its dependence on anchor boxes, which can lead to inaccurate detections when the objects are of different scales or aspect ratios than the predefined anchor boxes. The network architecture of SSD is depicted in Figure 2.8.

DSSD [21] improves upon the SSD architecture by replacing the VGG16 backbone with ResNet101, which allows for better feature extraction. It also adds prediction modules to different scale feature maps and incorporates deconvolution layers at the end of the network to enhance the early layer feature maps' ability to capture small objects. The prediction modules generate class scores and bounding box offsets for each anchor, which are used to refine the detection. DSSD further improves upon SSD's hard negative mining technique by introducing a technique called bootstrapped hard negative mining, which uses the most confident negatives during training to reduce the false positive rate. Finally, DSSD uses random cropping as a data augmentation technique to further improve performance.

Receptive field block (RFB)[38] aims to enhance the receptive field of convolutional

doi:10.6342/NTU202301439

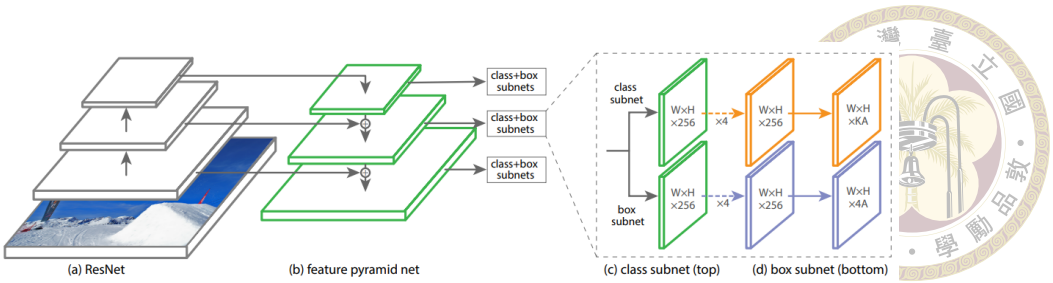


Figure 2.9: RetinaNet[35] network architecture. It uses ResNet101 and FPN as the feature extractor to detect objects of different sizes. The main contribution of RetinaNet is the proposal of focal loss, which down-weights the easy negative samples in the training data, allowing the training process to focus more on the hard samples.

neural networks (CNNs) and proposes a block consisting of three parallel branches with different dilation rates and convolutional kernel sizes. These parallel branches are concatenated to mimic the ability of the human visual system to capture features of different scales simultaneously. The RFB is applied to the SSD architecture, resulting in a 3% mean-AP improvement without significant speed loss.

RetinaNet[35] is another one-stage object detector that aims to tackle the issue of class imbalance during training. RetinaNet proposed a focal loss function that down-weights the influence of easy negative examples and make the training process focuses on the hard ones. It is built upon ResNet101 and FPN[34] architectures and it has been shown to outperform DSSD in detecting small and medium-sized objects by a large margin on the MSCOCO dataset. The overall network architecture is shown in Figure 2.9.

Through extensive research on previous work of object detection, we have observed that the anchor technique is widely used in detectors, and the proper configuration of anchors is crucial for successful training. However, most existing work adopts a general anchor setting that may not be suitable for autonomous driving scenes. Therefore, in the next section, we propose a depth-aware anchor sampling technique that takes into account the distribution of ground truth objects, aiming to fully utilize the training data. We believe that this approach will improve the performance of the object detector and make it more

doi:10.6342/NTU202301439

effective in autonomous driving applications.

2.3 Proposed Methods

In this section, we explore an approach to improve the anchor distribution in object detectors to increase their ability to detect small objects in driving scenes. Our approach is motivated by the observation that the default uniform anchor distribution may not cover all ground truth in the training dataset. In anchor-based detectors, ground truth boxes must be assigned to at least one anchor during training; otherwise, they will be ignored during training. An anchor will be assigned to a ground truth if its IoU is greater than 0.5, meaning that the anchor have similar dimensions and location to the ground truth box. To investigate this process further, we conducted an experiment on KITTI3D dataset to determine the number of ground truth boxes successfully assign to at least one anchor. The experiment results is shown in Figure 2.10. To our surprise, the default anchors cannot cover all ground truth boxes, indicating that uniformly-distributed anchors are not suitable for the KITTI3D dataset, especially at the scene horizon where many boxes are missed, resulting in a loss of 5.4% of the total training labels.

This abnormality occurs because objects in driving scenes are distributed differently compared to other general datasets. In driving scenes, most objects stay on the ground plane, and many distant objects are concentrated near the horizon. This special data distribution makes default anchors hard not suitable to use anymore. While it may seem like a simple solution to sample denser anchors with smaller strides on feature maps, we argue that it is ineffective because only a small part of the image requires denser anchors than other areas. This solution would generate too many useless anchors and introduce

doi:10.6342/NTU202301439

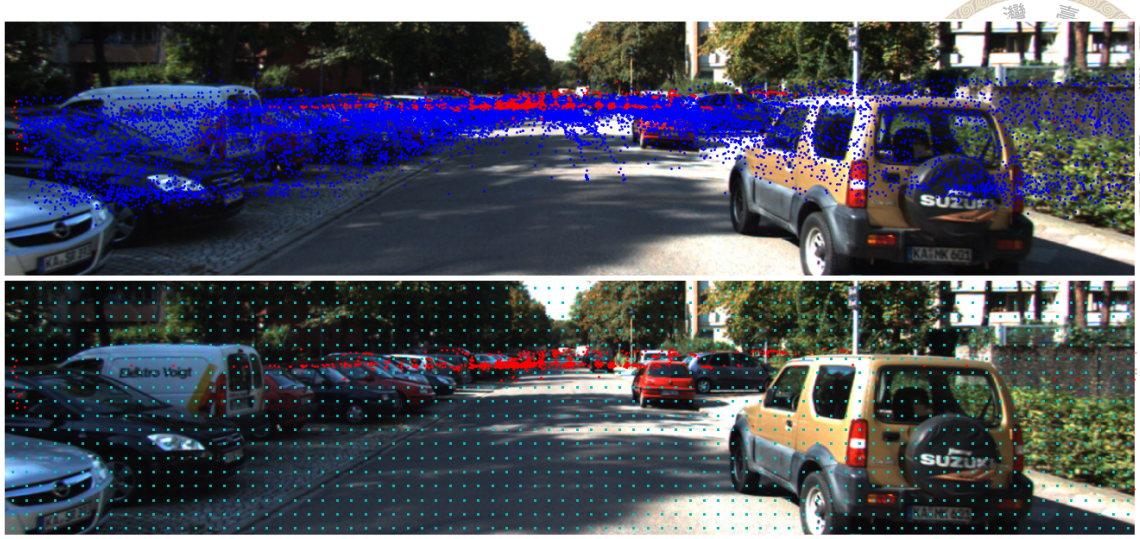


Figure 2.10: Anchor assignment results with default anchor settings. Each labeled box center is represented by a dot, with blue dots indicating covered ground truth boxes and red dots representing missed ground truth boxes. Cyan-blue dots represent the anchor's central locations. As shown in the figure, many ground truth boxes located near the horizon are missed, resulting in a total loss of 5.4% of the training data.

redundant parameters to the network, making it prone to overfitting and more difficult to converge.

To address this issue, we propose a depth-aware anchor sampling (DAS) method that customizes the pre-defined anchor distribution based on the training data. Specifically, we adjust the sampling stride of the anchors based on depth bins. Our method samples densely near the horizon to capture far objects while reducing the number of anchors in image parts that are closer to the camera and have sparser label distributions. This approach leverages our prior knowledge of the object distribution in driving scenes and the perspective projection of the image. In Figure 2.11, we demonstrate the effectiveness of the proposed DAS method, as it greatly reduces the number of missed ground truth boxes and aligns the anchor distribution more closely with the ground truth distribution. By utilizing anchors more effectively and tailoring them to the driving scene, our proposed method is able to capture 99.23% of the ground truth boxes, making better use of the available training data compared to the uniform-distributed anchor, which can only capture 94.6%

doi:10.6342/NTU202301439

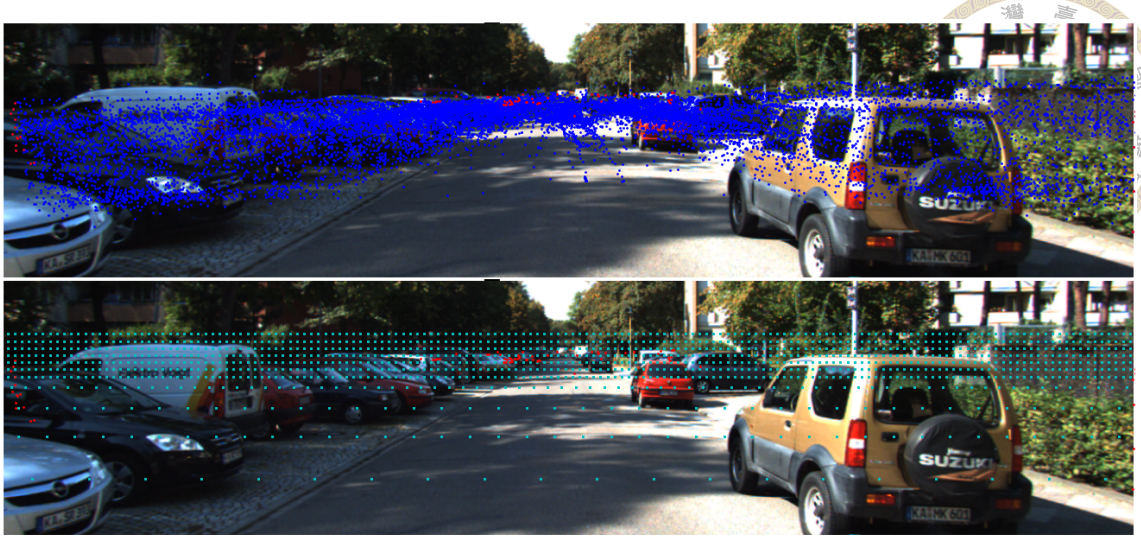


Figure 2.11: Anchor assignment results of the DAS-generated anchors. It can be observed that most of the missed ground truth boxes are now covered, and the distribution of anchors is more similar to the distribution of the training data.

of the label boxes.

2.3.1 Depth-aware Anchor Sampling

Our proposed depth-aware anchor sampling (DAS) method is inspired by the depth-aware convolution approach used in M3D-RPN[8]. In that work, the feature map is horizontally split into multiple bins, and non-shared kernels are used to perform convolution in each bin. This is based on the observation that different rows in the feature map may correspond to objects at different depths and scales, making separate convolutions for different depth scenes beneficial for the network to learn. We extend this concept to anchor generation by horizontally splitting the camera image and corresponding ground truth box centers into bins. In each bin, we calculate the IoU between anchors and ground truth boxes only within that bin. We also attempt to merge two adjacent bins into a larger bin with a bigger sampling stride to reduce the number of anchors. However, we only merge bins if the number of missed ground truth boxes after merging is smaller than a threshold, to avoid sacrificing too many training labels. The merging process is illustrated in Figure

doi:10.6342/NTU202301439

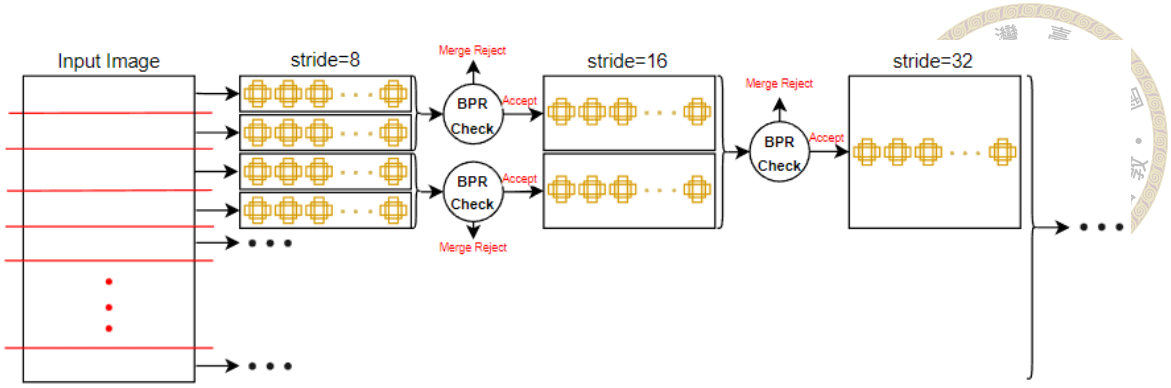


Figure 2.12: The merging process of the Depth-Aware Sampling (DAS) method. The image is divided into multiple depth bins, and the ground truth boxes and anchors are placed in their respective bins. The algorithm attempts to merge adjacent bins without sacrificing too many ground truths. If a merge is successful, anchors with a larger stride can be sampled to remove redundant anchors.

2.12. This design is based on the assumption that the ground truth should be captured by anchors in the same depth bin.

The anchor merging process can be done recursively until no bin is mergeable. In our experiment, we start at stride 8 and attempt to merge to stride 16, 32, 64, and 128. The intermediate results of the merging process are shown in Figure 2.13. Our method gradually replaces unnecessary dense anchors in the lower part of the image, and the total merging process stops at stride 128, since there are no more mergeable bins. In Figure 2.14, we show the number of anchors and the best possible recall rate, where our proposed method finds the optimal set of anchor distribution with 41,600 anchors and the best possible recall (BPR) rate of 99.2%. We emphasize that compared to the naive dense anchor solution, our method only sacrifices 0.05% of the ground truth boxes and reduces more than 60,000 unnecessary anchors. This demonstrates the effectiveness of our proposed DAS method.

2.3.2 K-means Anchor Dimension

In addition to anchor distribution, the dimensions of anchors are also important factors influence the performance. In some previous work, such as in Ground-aware[41],

doi:10.6342/NTU202301439

Figure 2.13: Intermediate result of the Depth-Aware Sampling (DAS) anchor merging process. The process starts with a stride of 8, where the anchors are densely sampled. With each attempt to merge adjacent anchors, the total number of anchors is reduced until a stride of 128 is reached.

doi:10.6342/NTU202301439

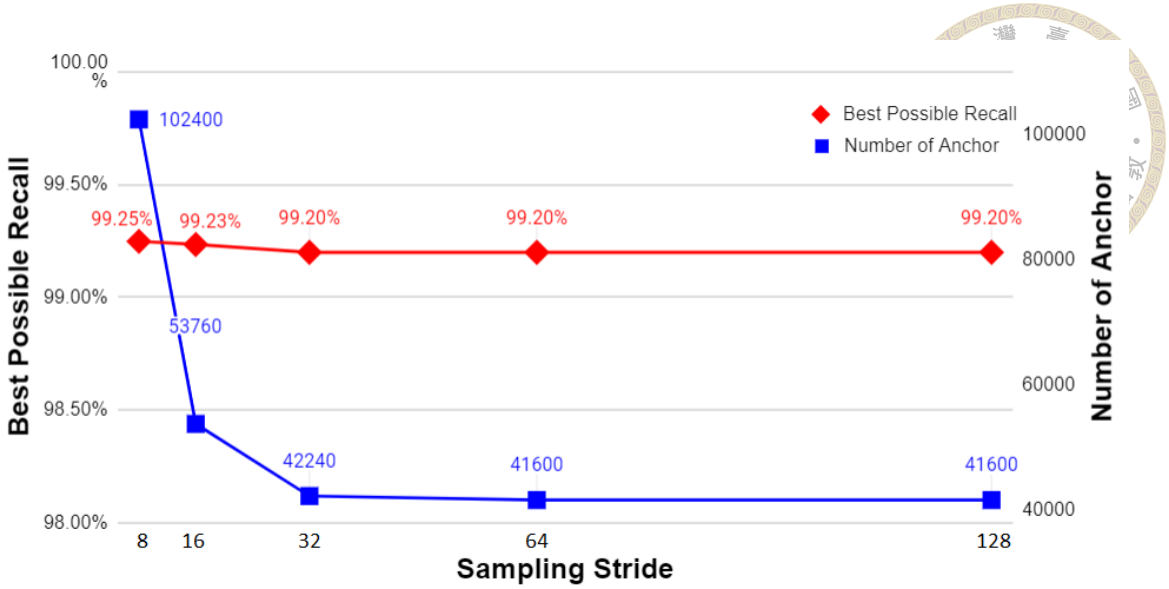


Figure 2.14: Best Possible Recall (BPR) and number of anchors during the anchor merging process. The red line represents the BPR of all depth bins, while the blue line indicates the number of anchors. With the DAS method, we achieve a training labels loss of only 0.05% , while reducing the number of anchors by over 60,000.

hand-crafted anchor aspect ratios and sizes were used without considering the training data. To improve upon this, we propose using a k-means clustering algorithm to determine the anchor width and height based on the dimensions of the label boxes. This approach was first proposed in YOLOv2[54]. We report our k-means clustering results in Figure 2.15 and Figure 2.16.

As we can see in Fig2.15, the anchor dimensions generated by the k-means clustering algorithm are more relevant to the label boxes. Furthermore, we demonstrate the importance of the DAS method by comparing the performance of the detector with and without DAS, for different numbers of anchors. In Figure 2.17, we show that no matter how many anchors we add to the detector, the best possible recall is capped at roughly 94%. Only with the help of DAS, the BPR is able to reach a higher level. This result highlights the effectiveness of the proposed DAS method for improving the performance of object detectors in driving scenes.

doi:10.6342/NTU202301439

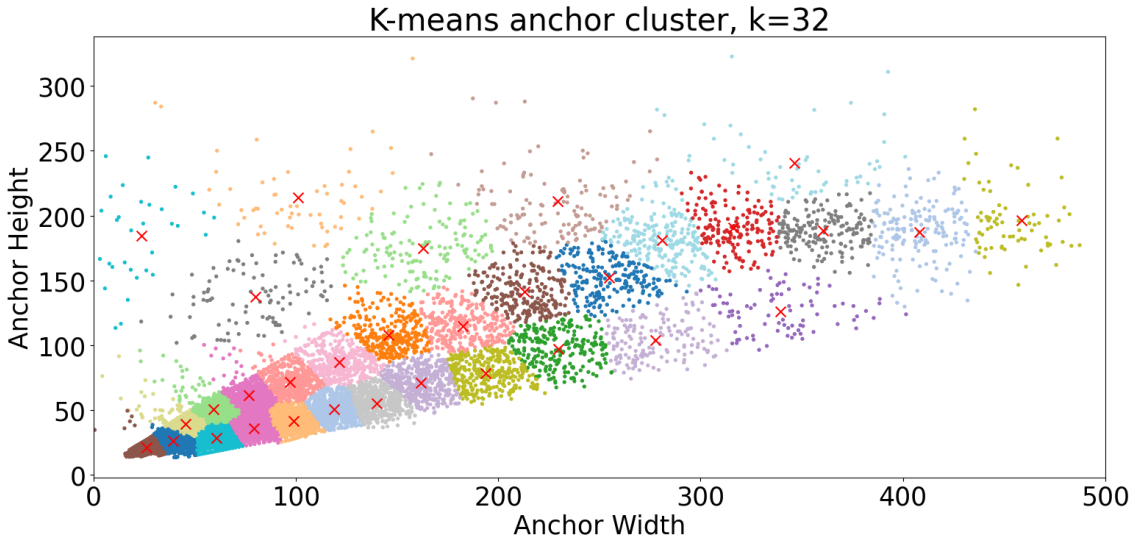


Figure 2.15: K-means clustering results for training label width and height. Each point represents a label box in the training dataset, and the red cross points indicate the centroid of each cluster. The anchor dimensions we will use are the centroids of each cluster group.

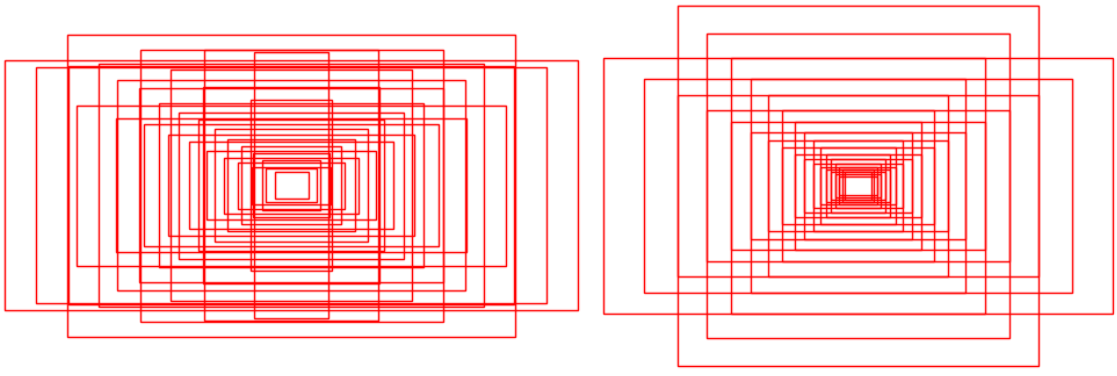


Figure 2.16: Anchor dimensions calculated by K-means. The right-hand figure shows the 32 hand-crafted anchors defined in Ground-aware[41], and the left-hand figure shows the clustered anchors calculated by K-means. By utilizing the K-means anchors, we can make the anchors more closely aligned with the training dataset.

doi:10.6342/NTU202301439

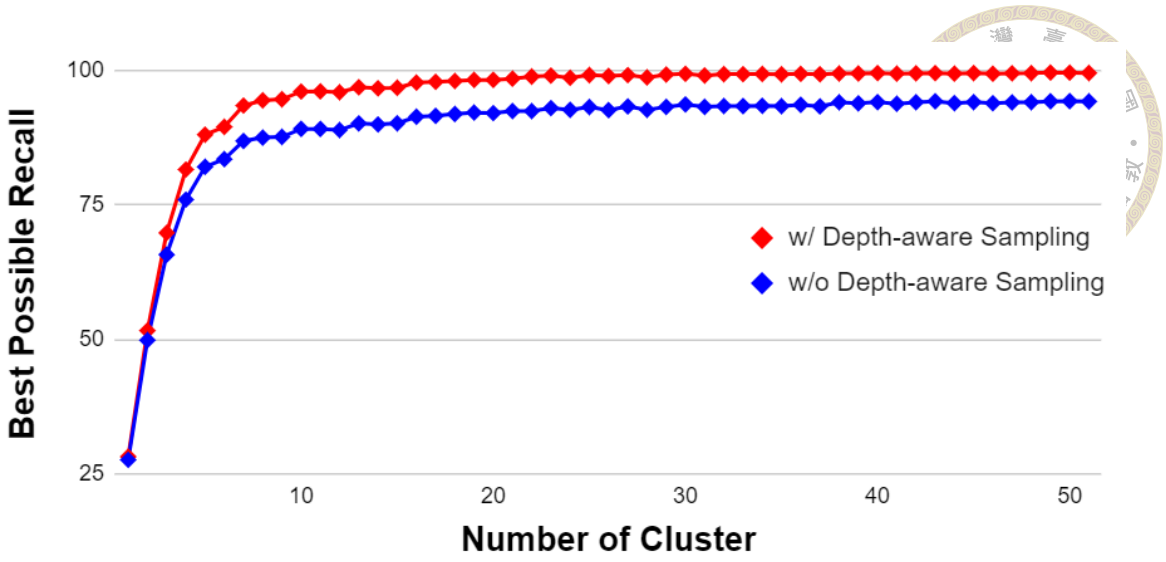


Figure 2.17: The best possible recall increases as the number of anchors increases. However, even with a large number of anchors, we still need DAS to achieve a recall above 95%.

Table 2.1: Experimental results of the 3D object detection algorithm on the KITTI3D validation dataset. The best performance in each column is indicated in bold font.

K-means	DAS	Car AP (IoU=0.7)		
		Easy	Moderate	Hard
		95.91	89.11	81.92
✓		97.38	88.36	81.26
	✓	96.08	**90.19**	**82.48**
✓	✓	**98.46**	90.07	82.43

2.4 Experiment Result

We experimented with our depth-aware anchor sampling(DAS) method on KITTI3D dataset and found that it greatly benefits the detector in capturing objects. We observed that it outperforms the baseline's accuracy. Specifically, it excels in detecting moderate and hard objects, which are considered farther objects in the scene.

doi:10.6342/NTU202301439

2.5 Summary

In this chapter, we have discussed object detection, a fundamental task in image recognition that involves accurately localizing and classifying objects by generating bounding boxes. Object detectors can be categorized into two types: two-stage detectors and single-stage detectors. Two-stage detectors propose regions of interest in the first stage and classify the objects within each bounding box in the second stage. On the other hand, single-stage detectors treat object detection as a regression problem, using a unified detector to achieve their goal. Notable two-stage detectors include R-CNN, Fast R-CNN, and Faster R-CNN, while popular single-stage detectors include YOLO, YOLOv2, YOLOv3, SSD, DSSD, RFB, and RetinaNet.

During our experiments with object detection in the context of autonomous driving scenes, we observed that default anchor settings may not be suitable, particularly for ground truth objects near the horizon. To address this issue, we proposed two approaches: depth-aware anchor sampling and k-means anchor clustering. These methods aim to align the anchor distribution with the characteristics of the training dataset. By implementing these techniques, we were able to improve the performance of the detectors on the KITTI3D validation set by 2.5% in the easy benchmark. This experiment highlights the importance of considering data distribution when utilizing anchor techniques in deep learning networks.

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

Chapter 3 Data Augmentation

Data augmentation is a widely used deep learning technique to increase the size and diversity of a training dataset by generating new variations of existing data. The primary objective of data augmentation is to create a robust and generalizable model by exposing it to a wider range of data variations. For example, altering the image brightness in the dataset can make the network more resilient to changes in lighting conditions and resizing the image can make the network robust to the object's scale. In the field of object detection, data augmentation has been demonstrated to significantly enhance model performance.

In this chapter, we first introduce commonly used data augmentation techniques for 2D and 3D object detection. Secondly, we implemented several data augmentation methods for 3D object detection and tested their performance. Additionally, we proposed a scene-aware copy-paste method tailored for 3D object detection and reported the experimental results on the KITTI3D dataset. Through these experiments, we hope to provide insights into how data augmentation can be utilized to enhance the performance of 3D object detection models in driving scenarios.

doi:10.6342/NTU202301439

3.1 Related Work

3.1.1 Data Augmentation for Object Detection

We categorize data augmentation methods into three types: image transformation, image erasing, and image mixture. The first type, image transformation, applies basic image processing operations, including scaling, rotation, translation, horizontal flipping, and vertical flipping of the input image. These operations can generate new images with different object locations and object orientations with minimal computational cost, making the network more robust to different sizes of objects occurring in different locations. Another popular method is to adjust the hue, contrast, and saturation of the image. This can make the network more robust to changes in lighting conditions in the scene. Overall, image transformation methods can significantly increase the diversity of the training dataset with simple operations. We show an example of image transformation method in Figure 3.1.

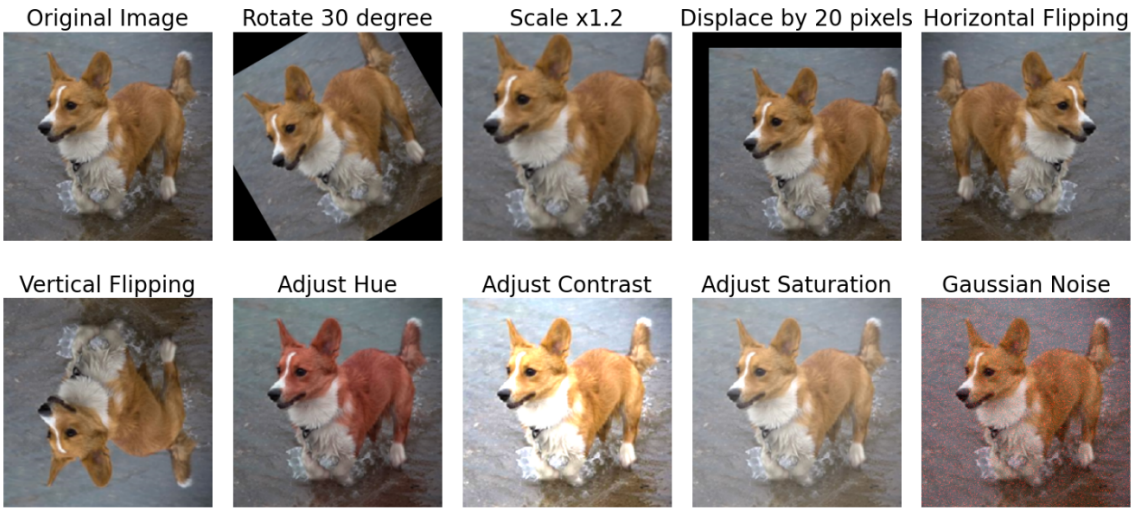


Figure 3.1: Data augmentation methods based on image transformation. These methods involve minimal computational cost and provide robustness against variations in lighting conditions and object sizes.

The second type of data augmentation is image erasing method, which involves mask-

doi:10.6342/NTU202301439

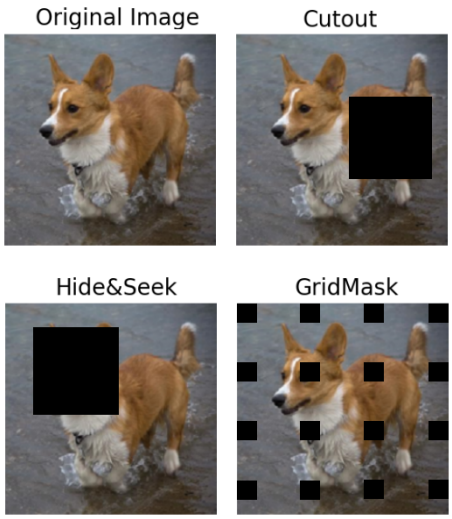


Figure 3.2: Data augmentation method based on image erasing. These methods all erase part of the input image during training, It focus network to predict the same outcome using other image feature. This can make network no easy to overfitt to training data.

ing out parts of the image and requiring the network to still predict the same result even when the input image is incomplete. For example, the Cutout method randomly removes cubic patches from the image, while the Hide and Seek[57] method tries to remove the most prominent and recognizable part of the image. The GridMask[14] method removes image patches on a grid to erase pixels in a uniform way. These methods have all shown positive results by preventing the detector from relying too heavily on specific regions of the input and forcing it to recognize objects using more subtle information. An example of image erasing methods is shown in Fig3.2.

The third type of data augmentation is the image mixture method, which combines multiple images and labels together to increase the complexity of the dataset. Mixup[55] directly blends two images together and requires the network to produce both labels with equal probability in image classification. CutMix[72] involves copying and pasting a cropped image patch onto another image and asking the network to predict the probability based on the ratio of cropped region. Mosaic[6], proposed by YOLOv4, combines four images into one and asks the detector to find all objects present in all four different

doi:10.6342/NTU202301439

scenes. Overall, image mixture methods blend labels and pixels from different sources into a single image and ask the network to process them all together. This helps the training process by increasing the difficulty of the task and preventing the model from overfitting. The example of image mixture method is shown in Figure 3.3.

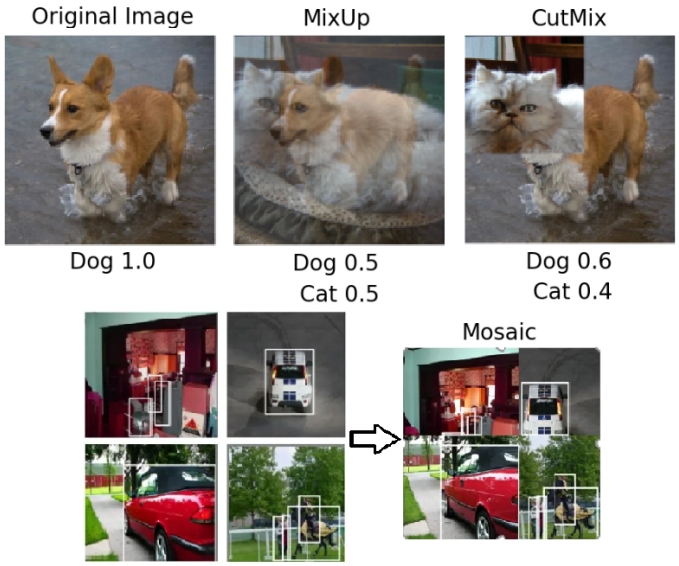


Figure 3.3: Data augmentation method based on image mixture. These methods involve combining different images together to increase the complexity of the task and the diversity of the dataset.

3.1.2 Data Augmentation for 3D Object Detection

Although there are a wide range of data augmentation(DA) methods, only a few of them can be applied to 3D object detection. It is important to maintain the reasonable image-label pair and the geometric consistency of the 3D scene. For instance, random rotation is definitely not suit to apply to 3D object detection network since the camera orientation is parallel to the ground in most driving scene datasets, and a rotated image could mislead the network. Therefore, it is necessary to adopt data augmentation methods with great care and make sure the image-label pair is reasonable after the augmentation. In this section, we introduce some related work that focuses on exploring data augmentation

doi:10.6342/NTU202301439

methods for 3D object detection.

In [55], the authors adapt the CutOut method with the setting of masking out two or four cubic patches in the image. They also adopt the Box-MixUp and Copy-paste methods, which copy objects and labels to another image. Additionally, they adopt the Mosaic method, which tiles four images together into one, making the scene more diverse. In their experiments, the Box-MixUp method performed the best, with over 4% improvement across all difficulty levels in KITTI3D. This work demonstrates the potential of adapting existing image mix techniques to 3D object detection tasks. Its prospoed method is depicted in Figure 3.4.

Figure 3.4: Data augmentation methods presented by Sugirtha T et al.[55]. They utilize the Cutout method, which erases some cubic regions from the image. Additionally, they employ Box-Mixup and Box-Cut-Paste techniques, which involve pasting image-label pairs from other image sources. Lastly, they adopt the Mosaic Tile method, which combines four different scenes together.

Focusing on preserving geometric consistency when augmenting datasets, Qing Lian et al.[33] proposes four methods for 3D object detection. The first method is random scaling, which zooms in or out of the image while adjusting the camera's focal length to main-

doi:10.6342/NTU202301439

tain the scene's perspective. The second method is random cropping, which changes the receptive field of the network by cropping and padding the image. The third method alters the camera's location in the depth direction to generate different viewpoints of the same scene. Finally, the paper proposes a copy-paste method where objects from one image are copied to another, and their depth and vertical position are adjusted based on perspective projection. In their experiments using M3D-RPN and CenterNet, both random scaling and copy-paste methods showed significant improvement, achieving over 3% improvement on moderate difficulty objects in the KITTI test dataset. Their proposed method is depicted in Figure 3.5

Figure 3.5: Data augmentation methods proposed by Qing Lian et al.[33]. They suggest zooming in the image and changing the camera focal length to simulate the image that would be obtained by moving the camera forward or backward. They also propose warping the image to a different camera perspective; however, this method requires a depth estimation network to determine the depth of pixels.

Focusing on copy-paste augmentation method, [1] incorporates the concept of augmented reality to enrich the KITTI dataset by rendering realistic car models on the image. This approach involves using 28 high-quality 3D car models and 360-degree panoramas of the scene to render the car models with authentic lighting conditions. The researchers manually decided where to place the objects in the image and found that adding more than five objects in a single image could have a detrimental effect on the detector's performance since the original scene is majorly covered by the added objects. Its augmented example

doi:10.6342/NTU202301439

is shown in Figure 3.6

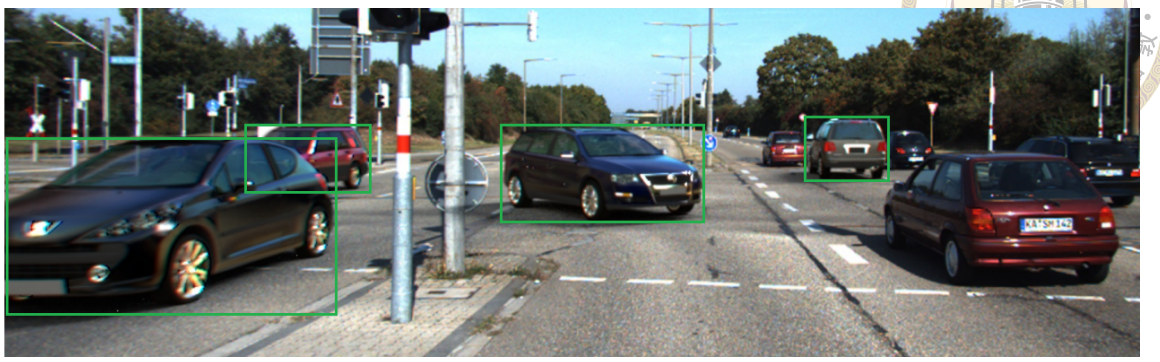


Figure 3.6: Data augmentation results using Augmented Reality[1]. High-quality car models are rendered onto the image, taking into consideration the lighting conditions of the scene. The cars marked by green boxes represent the objects pasted using this method.

In addition to research that focuses on augmentation method, we also investigate the data augmentation method used by previous 3D object detectors and we report our findings in Table3.1. The most commonly used methods are random horizontal flipping and color jittering, which are simple and have been shown to improve performance. Some detectors also use 2D box jittering to introduce uncertainty about the object's location. Random cropping and scaling can be applied to parts of the network where inconsistent 3D information won't have a negative impact on performance, such as the classification branch. One work that focuses specifically on data augmentation is ROI10D[44], which developed a copy-paste method similar to that of [1]. However, to prevent domain shift when pasting car models, ROI10D recovered 3D car models for the KITTI dataset and used car models from the same domain to enrich the training dataset. Overall, the data augmentation method application in 3D object detectors is still limited to a few methods.

Based on the related work surveyed, it is evident that image mixture type of data augmentation methods are effective and have great potential in complementing smaller datasets like KITTI. These methods, such as Copy-Paste or Box-MixUp, can generate objects in new locations, thereby resulting in significant improvements in 3D object detec-

doi:10.6342/NTU202301439

Table 3.1: Data augmentation methods adopted by previous work on 3D object detectors. Horizontal flipping and color jittering are the most commonly used techniques, while other methods are less frequently adopted by networks and may be limited to specific branches or under certain conditions.

Data Augmentation	**Network**
Horizontal Flip	DD3D[46], FQNet[36], Ground-Aware[41], MonoFlex[74], MonoDETR[73], MonoDIS[56], MonoPair[16], OFTNet[52], RTM3D[32], SMOKE[42]
Color Jittering	FQNet, Ground-Aware, MonoDETR, MonoFENet[4]
2D Box Jitter	FQNet, MonoPSR[30]
Random Scaling	MonoDETR, MonoPair, OFTNet, SMOKE
MixUp	ROI10D[44]

tion. However, one issue that remains unaddressed in the previous studies is determining the appropriate location to paste the added objects.

For example, in the study by [1], the annotators need to manually select the location to place the new object on the Bird-eye-view plane. In contrast, studies like [33] and [55] randomly spawned objects based on geometric constraints. However, this approach can lead to unrealistic placements of objects, such as on a wall or behind a tree truck, resulting in misleading augmented result, as we shown in Figure 3.7. To address this issue, we propose a scene-aware object placement algorithm that utilizes a predicted depth map to identify vacant areas in the scene and place the object in these areas.

3.2 Proposed Methods

3.2.1 Scene-aware Copy-paste Data Augmentation

We propose a scene-aware data augmentation approach that employs depth maps generated by a depth estimation model. This allows us to consider scene structure and estimate the feasibility of pasting an object in a given area. Before pasting the object onto the image, we first check the pasted area on the depth map. If there are many pixels

doi:10.6342/NTU202301439

Figure 3.7: Example of a copy-paste data augmented image without scene-aware check. The pasted objects can appear in unrealistic locations, such as on walls or behind other obstacles. The objects inside the red boxes represent the pasted objects.

doi:10.6342/NTU202301439

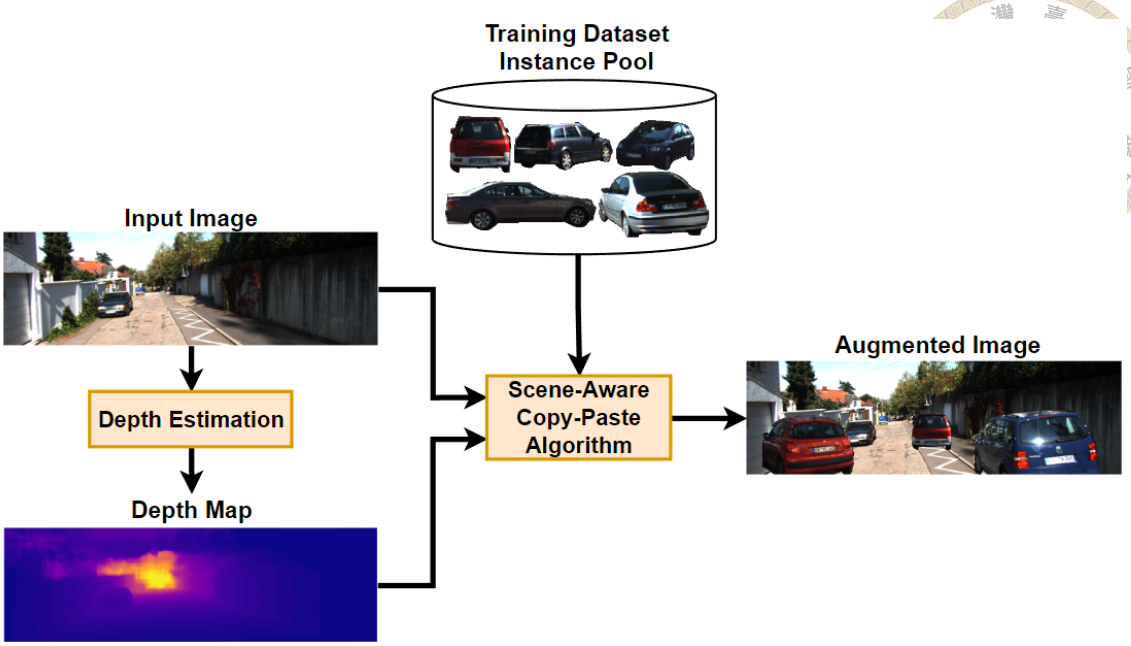


Figure 3.8: Pipeline of scene-aware copy-paste data augmentation. We utilize a pre-trained depth estimator to predict a depth map and use it to assess the scene structure before pasting an instance onto the image. If the percentage of pixels in the target region with a smaller depth than the pasted object is below a certain threshold, we consider it a bad proposal and randomly search for another location to paste the object.

within the area whose depth is smaller than the added instance depth, we consider that it is inappropriate to paste it and we search for another suitable location until a good match is found. This process is shown in Figure 3.8.

Since the depth estimation can be performed offline, the overhead of this method is almost the same as the original methods, resulting in only a minor increase during the training process. Furthermore, our scene-aware viability check is a simple yet effective plugin that can be seamlessly integrated into all 3D object detectors. It is important to note that the pasted instances are derived from the same training dataset, specifically the KITTI3D training set, and we only utilize instances that are fully visible and larger than 3000 pixels in size.

To show the effectiveness of our proposed method, we have employed our method to the Box-MixUp and Copy-Paste data augmentation methods. The MixUp approach

doi:10.6342/NTU202301439

involves cropping the objects' 2D bounding box and pasting it onto another image to generate artificial objects in a new scene. The pasted objects are set to be half-transparent to better blend in the scene. In addition to our scene-aware check, we also prevent the pasted object to block other object in the original scene. In addition, the instance can be scaled by changing its 3D location altogether. We use this random instance scaling method to generate more unseen object scales to further enhance data diversity.

3.3 Experiment Result

In this section, we present the experimental results of our evaluation of the introduced data augmentation method. We conduct experiments on the 3D object detector proposed by GAC[41], which is currently one of the state-of-the-art anchor-based 3D object detection methods. All experiments are conducted on the KITTI3D dataset, and the model is trained for 30 epochs on the training set consisting of 3711 images, while testing is performed on the 3768 images of the validation set. This data split was proposed by Chen et al.[15]. During training, the batch size is set to eight, and we use the Adam optimizer with a learning rate of 10^{-4} to train the network. Regarding data pre-processing, we cut the top-100 pixels out of the image to speed up the network and resize the image to 288x1280, as suggested by Ground-aware paper.

3.3.1 Quantitative Result

We report the performance of each data augmentation method by comparing their improvement to the baseline. The result is shown in Figure 3.9. In this experiment, we set the horizontal flipping probability to be 50%. The CutOut mask width is set to 32 pixels,

doi:10.6342/NTU202301439

Method	AP 3D(IoU=0.7)			AP BEV(IoU=0.7)		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Baseline(w/o DA)	17.97	13.05	10.25	24.43	17.64	13.43
Horizontal Flipping	+1.57	+1.06	+0.80	+2.42	+2.32	+1.96
Photometric Distortion	+1.00	+0.59	+0.56	+0.70	+0.31	+0.33
CutOut(2 holes)	+0.73	+0.29	-0.47	+0.90	+0.30	+0.25
CutOut(4 holes)	-0.80	-0.10	-0.80	-1.16	-0.30	-0.15
2D Box Jitter	-0.65	-0.50	-0.88	-0.49	-0.51	-0.22
Randan Zooming	-2.69	-2.21	-1.89	-3.07	-1.68	-1.32
Box-MixUp	+0.21	-0.50	-0.53	+1.09	+0.52	+0.47
Copy-Paste	+0.54	+0.78	+0.04	+0.58	+0.75	+0.74
Scene-Aware-Box-MixUp	+1.11	+0.94	+0.03	+1.12	+0.91	+0.69
Scene-Aware-Copy-Paste	+2.35	+0.99	+0.71	+3.93	+1.82	+2.21
Horizontal Flipping + Photometric Distortion + Scene-Aware Copy-Paste	**+4.72**	**+2.85**	**+1.70**	**+6.12**	**+3.61**	**+3.04**

Figure 3.9: Improvements observed after applying various data augmentation methods. The cells highlighted in green indicate positive performance enhancements, while the cells in red represent worse performance compared to the baseline. The horizontal flipping probability is set to 50%. The CutOut mask width is fixed at 32 pixels, and the 2D box jitter range is limited to a maximum of 3% of its width. Furthermore, the random zooming is performed within a scaling range of 0.8 to 1.2. Notably, our scene-aware copy-paste method achieves a significant improvement of +2.35%, making it the top-performing technique in our experimental survey.

while 2D Box jitter range is limited to be at most 3% of its width. Additionally, the scaling range of random zooming is set to 0.8 to 1.2.

For the Box-MixUp method, we adopted the settings proposed by [55] and paste the object according to the bounding box while averaging out pixel values between the original pixel and the added instance. Regarding the copy-paste method, since KITTI3D does not provide any instance segmentation label, we manually segmented 500 instances from the training set for the experiment. In both methods, we paste two additional objects to each image for the experiment since in our ablation study, two additional objects can yield the best result. For both, Box-MixUp and Copy-Paste, The instance can be scaled by changing their 3D location label. The random scaling of instances is range between 0.8 and 1.2, to avoid any artifact led by over-resizing.

doi:10.6342/NTU202301439

For our scene-aware methods, we set the threshold of scene-aware to be 25%. This implies that if 25% of the pixels in the paste region has a depth smaller than the instance's depth, the scene-aware copy-paste will reject the augmentation and find another location. Additionally, we use DORN[22] to derive the depth map, which is a depth estimation model pre-trained on the KITTI-depth dataset.

In Figure 3.9, we observe that horizontal flipping and photometric distortion have a positive effect on the performance across all difficulty metrics. Specifically, horizontal flipping results in a +1.57% AP improvement in easy objects, while photometric distortion yields a +1.00% AP. CutOut also shows a mild improvement with a +0.73% AP gain. However, when cutting four holes out of the image, the effect becomes detrimental with -0.8% AP. We believe that the impact of 2D box jitter is negative because the detector still relies on accurate 2D bounding box geometry information to infer the object's depth. Therefore, jittering the center of boxes makes it harder for the network to infer depth accurately. Similarly, random zooming results in a worse performance since it introduces inconsistent viewpoints to the network, thereby making it difficult to infer depth.

Most of the image mixture methods result in a positive outcome, indicating the benefits of placing instances in different scenes to enhance the diversity of the training set. The Copy-Paste method results in a +0.54% AP improvement, and the use of the scene-aware method further improves the performance to +2.35% AP in easy objects. This demonstrates the effectiveness of our proposed method and the importance of eliminating unrealistic locations when pasting objects. Box-MixUp also shows a small improvement with a +0.21% AP gain. However, there are slight drawbacks in moderate and hard objects. We believe this is because the crop region of the added object in Box-MixUp sometimes includes more than one object, but the label cannot reflect this. This might confuse the

doi:10.6342/NTU202301439

network, and it is especially problematic for occluded objects in moderate and hard difficulty.

We also tested the data augmentation method by combining all the positive methods together, resulting in a +4.72% AP improvement in easy objects. This highlights the potential of different data augmentation methods from various branches working well together to complement each other.

3.3.2 Ablation Study

In our ablation study, we investigate the impact of scene-aware and random scaling methods on data augmentation for object detection. The results shown in Figure 3.10 demonstrate that Box-MixUp, while effective in some cases, can produce unrealistic augmented images that hinder the network's ability to learn the relationship between scene and object depth. However, after applying a scene-aware method to prevent such issues, both Box-MixUp and Copy-Paste exhibit significant improvements in performance. Specifically, Box-MixUp shows a 1.12% increase in AP for easy objects, while Copy-Paste achieves an impressive 3.93% improvement in the same metric.

Additionally, we conducted an experiment to determine the optimal number of pasted objects in the image. As shown in Figure 3.11, the best number of pasted objects is one or two, as pasting too many objects can have a detrimental effect on the model. The scene could deviates from the original dataset, if its majorly covered by pasted objects. This effect can be seen in Figure 3.12 where we paste seven objects on the scene.

doi:10.6342/NTU202301439

Method	Scene-Aware	Instance Random Scaling	3D(IoU=0.7)			BEV(IoU=0.7)		
			Easy	Moderate	Hard	Easy	Moderate	Hard
Baseline(w/o DA)	-	-	17.97	13.05	10.25	24.43	17.64	13.43
Box-MixUp			+0.21	-0.50	-0.53	+1.09	+0.52	+0.47
Box-MixUp		✓	+0.73	-0.26	-0.38	-0.14	-0.54	+0.16
Box-MixUp	✓		+1.13	+0.72	**+0.52**	+0.95	**+0.93**	**+0.81**
Box-MixUp	✓	✓	**+1.11**	**+0.94**	+0.03	**+1.12**	+0.91	+0.69
Copy-Paste			+0.54	+0.78	+0.04	+0.58	+0.75	+0.74
Copy-Paste		✓	+1.08	+0.64	+0.44	+1.27	+0.75	+0.56
Copy-Paste	✓		+0.70	+0.68	-0.14	+1.04	+0.80	+0.61
Copy-Paste	✓	✓	**+2.35**	**+0.99**	**+0.71**	**+3.93**	**+1.82**	**+2.21**

Figure 3.10: Ablation study of Copy-paste and Box-MixUp method. Green cells indicate improvements in performance compared to the baseline, while red cells indicate worse performance. The method with the best performance in each column is highlighted in bold font.

Method	Number of pasted objects	AP 3D(IoU=0.7)			AP BEV(IoU=0.7)		
		Easy	Moderate	Hard	Easy	Moderate	Hard
Scene-Aware Copy Paste	0	17.97	13.05	10.25	24.43	17.64	13.43
	1	**20.32**	14.8	**11.62**	27.05	19.37	14.8
	2	**20.32**	14.04	10.96	**28.36**	**19.46**	**15.64**
	3	20.09	14.17	10.36	26.45	18.73	14.19
	4	19.99	**14.4**	11.28	25.91	18.81	14.43
	5	19.33	14.13	11.05	26.62	19.16	15.35
	6	17.2	13.08	9.87	25.74	18.49	14.31

Figure 3.11: Number of paste objects in the scene can influence the performance. We find that pasting one or two objects is optimal for our settings.

doi:10.6342/NTU202301439

Figure 3.12: Overcrowded car objects paste in the scene. The left column is original images and the second column is the augmented images where seven objects are pasted on them.

3.3.3 Qualitative Evaluation

In this section, we present the augmented image results obtained using our implemented data augmentation methods to demonstrate the effectiveness of each technique. Firstly, in Figure 3.13, we display the augmented images produced by the Box-Mixup method, where three objects are pasted onto each image. Next, in Figure 3.14, we showcase the Cutout augmentation results, where four cubic patches are erased from each image. Furthermore, we exhibit the results of our randomly scaling technique in Figure 3.15, which simulates the effect of moving the camera forward or backward by zooming in or out on the entire scene.

For the random jitter augmentation, as shown in Figure 3.16, we apply jittering to the location and dimensions of the 2D bounding boxes to enhance the robustness of box regression. Figure 3.17 displays the augmented results of the copy-paste method, where

doi:10.6342/NTU202301439

Figure 3.13: Implementation result of Box-Mixup. The pasted objects are marked by red boxes.

Figure 3.14: Implementation of Cutout(4 holes)

doi:10.6342/NTU202301439

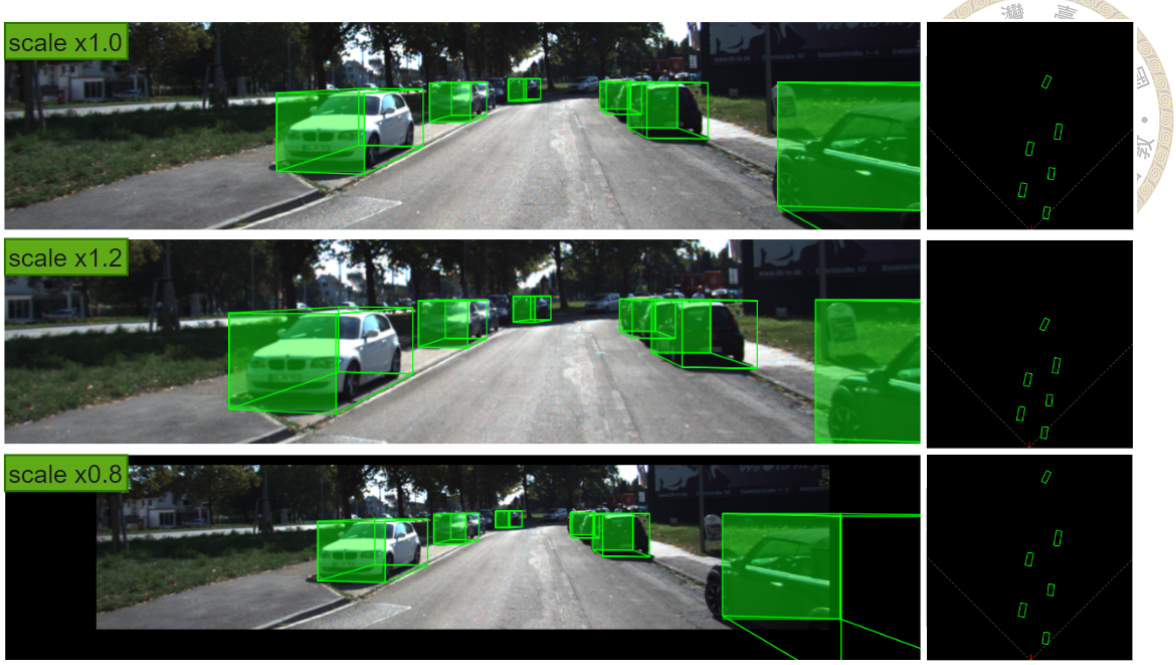


Figure 3.15: Implementation of image scaling

objects are pasted onto the images. In Figure 3.18, we present the results of the random instance scaling technique, where the pasted objects are randomly zoomed in or out, and the corresponding labels are adjusted accordingly.

Moreover, we showcase the predictions of our depth estimation model in Figure 3.19, illustrating the estimated depth maps. Lastly, in Figure 3.20, we demonstrate the augmented results of our proposed scene-aware copy-paste method, which checks the corresponding depth maps to ensure that the object is suitable for pasting in that region. Through this simple check, we can prevent objects from appearing in unrealistic locations.

3.4 Summary

In this chapter, we have explored a wide range of data augmentation techniques for 2D and 3D object detection. We have found that data augmentation can effectively increase the size and diversity of training data, and thus improve the generalization perfor-

doi:10.6342/NTU202301439

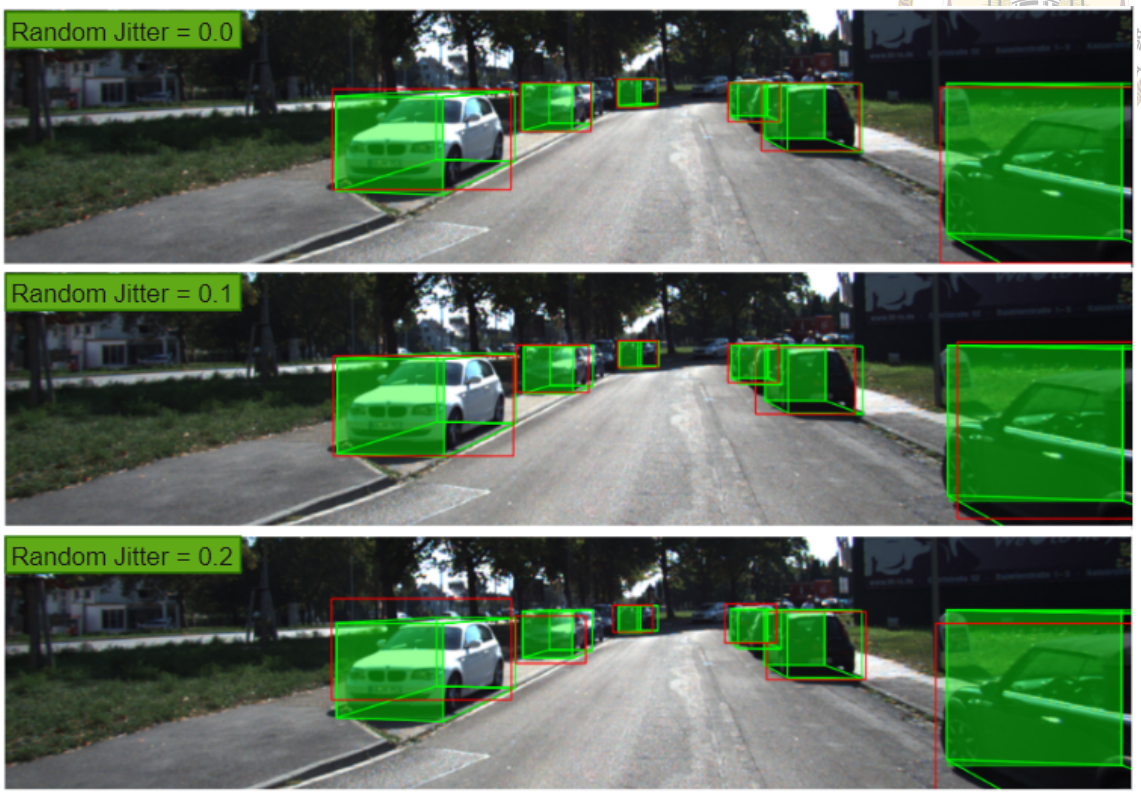


Figure 3.16: Implementation of 2D box jitter

Figure 3.17: Implementation of copy-paste method. The pasted objects are marked by red boxes

doi:10.6342/NTU202301439

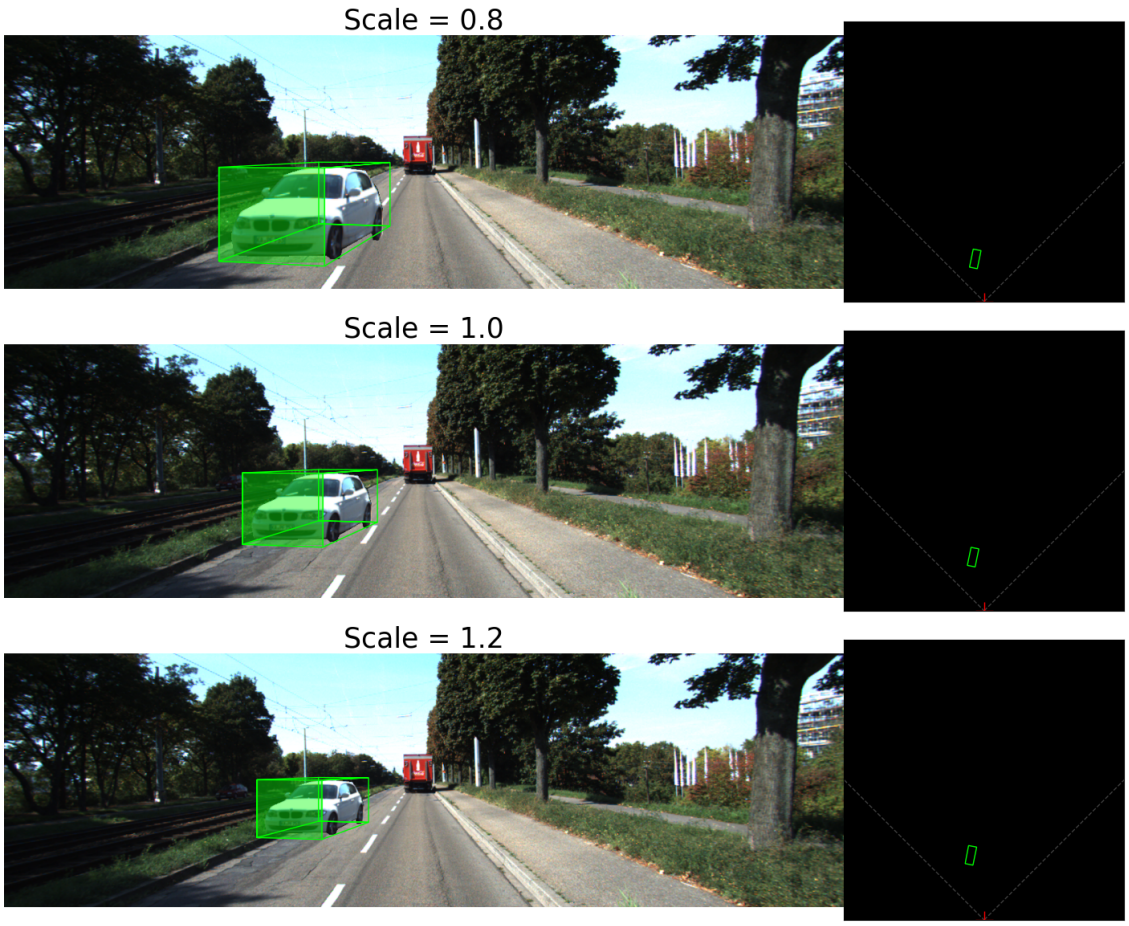


Figure 3.18: Implementation of instance random scaling. The objects are scaled before pasted on the images and its label is scaled accordingly.

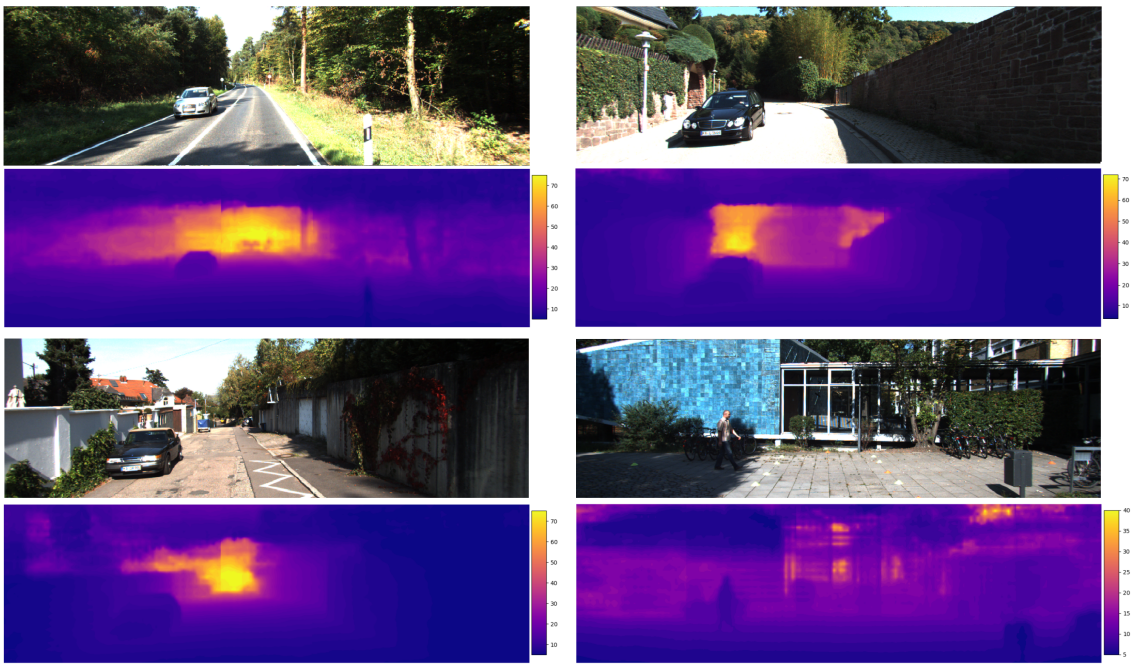


Figure 3.19: Implementation of scene-aware copy-paste method. The depth map is produced by DORN[22] indicating the scene structure.

doi:10.6342/NTU202301439

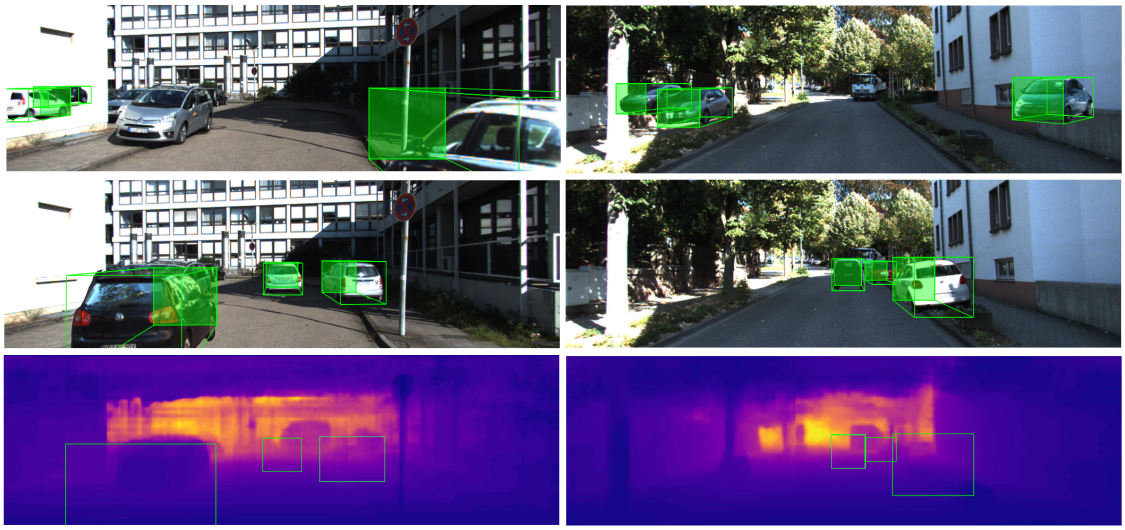


Figure 3.20: Implementation of our proposed scene-aware copy-paste method. The first row shows a possible unrealistic augmented result without scene-aware. The second row shows the augmented image after scene-aware check, and the third row represent the scene structure and the checking region before pasting the objects.

mance and robustness of the models. We have reviewed many popular data augmentation methods, including image transformations, color jittering, image erasing, and image mixture. Additionally, we have proposed a new scene-aware data augmentation method that can take into account the relationships between objects and scenes by utilizing depth map information. Our experimental results show that scene-aware data augmentation can produce more realistic and informative augmented images and improve the performance of object detection models on challenging datasets. We believe that data augmentation will continue to play an important role in deep learning research and applications.

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

Chapter 4 3D Object Detection in Driving Scene

In recent years, deep convolutional neural networks(DCNN) have demonstrated great performance in image classification and object detection tasks. Many researchers have shifted their focus towards more sophisticated image tasks, such as 3D object detection. This advanced image task can be used in autonomous driving scenes, where the distance to the on-road obstacle can be critical to self-driving safety. In this chapter, we survey recent 3D object detection algorithms and evaluate their performance on the KITTI3D dataset. We also propose ours improved anchor-based 3D object detector by introducing a novel perspective-aware convolution(PAC) module. By combining the PAC module with 3D object detectors, we achieve state-of-the-art performance in the KITTI3D validation set.

4.1 Preliminary

In the field of 3D object detection, a 3D object is typically represented by a cuboid in camera coordinates. This cuboid is characterized by seven key parameters: the centroid (x, y, z), which specifies the position of the cuboid center relative to the camera center, and the dimensions (w, h, l), which correspond to the width, height, and length of the

doi:10.6342/NTU202301439

Figure 4.1: 3D object detection. The cuboids in defined in camera coordinate and we need to find the location (x, y, z), dimension (w, h, l), and orientation θ of each cuboid.

cuboid, respectively. Additionally, the yaw angle θ denotes the orientation of the cuboid. Our objective in 3D object detection is to identify objects within images and accurately localize them in camera coordinates by predicting these seven variables $(x, y, z, w, h, l, \theta)$ for each object. This concept is illustrated in Figure 4.1.

4.1.1 Pinhole Camera Model

The pinhole camera model is a widely used perspective projection model that explains how objects in the 3D world are projected onto a 2D image plane through cameras. In the pinhole model, the camera's shutter is represented as a pinhole, and every beam of light that comes from the scene passes through the pinhole. The camera center is defined as the pinhole location. While in reality the image is produced behind the pinhole, for simplicity of calculations, the image plane is placed in front of the pinhole. This pinhole camera model is depicted in Figure 4.2.

Using the pinhole camera model, we can project 3D objects onto a 2D image plane by using the similar triangles relationship, which is given by:

doi:10.6342/NTU202301439

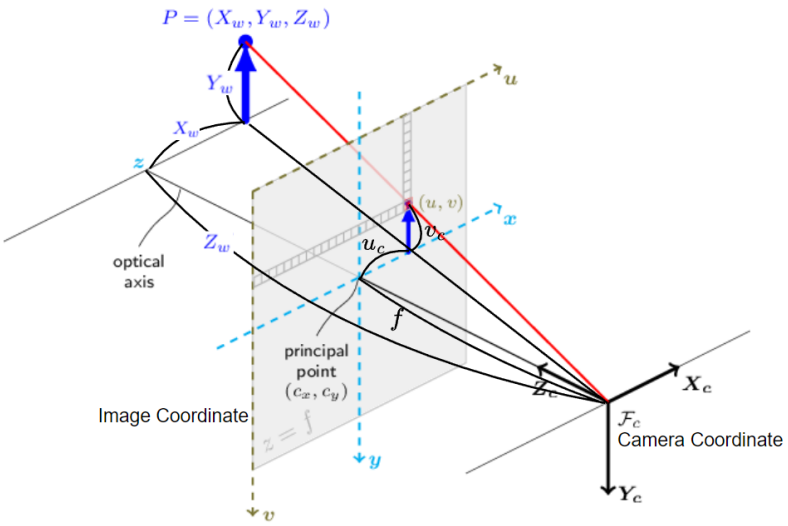


Figure 4.2: Pinhole camera model illustrating the projection relationship between the 3D world and the 2D image plane.

$$\begin{cases} \frac{u_c}{f} = \frac{X_w}{Z_w} \\ \frac{v_c}{f} = \frac{Y_w}{Z_w} \end{cases} \tag{4.1}$$

Here, (u_c, v_c) represents the coordinates of the projected point with respect to the camera coordinate system, f represents the focal length of the camera, and (X_w, Y_w, Z_w) represents a 3D point in the camera coordinate system.

However, we want to express (u_c, v_c) in the image coordinate system. We can use the following formula:

$$\begin{cases} u_c = \frac{u - c_x}{k} \\ v_c = \frac{v - c_y}{l} \end{cases} \tag{4.2}$$

where (u, v) is a pixel on the image coordinate and (c_x, c_y) is the principal point where the optical axis intersects with the image plane, and (k, l) are the length ratios that convert pixels to meters, namely $\frac{pixel}{m}$, at u and v axis respectively.

After combining Equation 4.1 and Equation 4.2 we can eliminate u_c, v_c, and use ma-

doi:10.6342/NTU202301439

trix form to express the result.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{f \cdot k}{Z_w} & 0 \\ 0 & \frac{f \cdot l}{Z_w} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{4.3}$$

To further simplify the equation, we use f_x, f_y to simplify the equation by $f_x = f \cdot k$ and $f_y = f \cdot l$, where f_x, f_y representing the focal length in pixel unit respectively to u and v axis. This results in the following:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{1}{Z_w} \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \end{bmatrix} + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{4.4}$$

Here we can use homogeneous coordinates to represent the projection as a linear system:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z_w} \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \tag{4.5}$$

We can simplify Equation4.5 as following:

$$p = \frac{1}{Z_w} W P_w \tag{4.6}$$

where p is a pixel coordinate in the image coordinate system, W is the camera intrinsic matrix, and P_w is the projected 3D point. Using these notations, we can rewrite Equation 4.5 as:

Equation 4.6 expresses the projection from the camera coordinate system to the image plane. Note that the intrinsic matrix is typically provided by the camera manufacturer or

doi:10.6342/NTU202301439

dataset provider. In our case, we use the intrinsic matrix provided by the KITTI dataset.

Furthermore, we note that in Equation 4.5, the depth information Z_w is lost when projecting to the camera image plane. As a result, monocular 3D object detection is an ill-posed problem, since the input image does not directly provide any depth information. Therefore, estimating an object's depth is a challenging task, and we propose to model the relationship between the scene and object depth using deep learning techniques. We will use this perspective relationship throughout this chapter.

4.2 Related Work

One type of 3D object detection involves predicting all nine variables of the cuboid, including roll and pitch. However, determining the pitch and yaw angle of an object with only visual cues is a challenging task. As a result, this type of work typically limits the detection environment to controlled indoor scenes with a limited range of object types. For instance, the SSD6D [27] aims to detect 3D objects using only 15 specific 3D CAD models. To address the challenge of regressing orientation, SSD6D classifies all possible camera viewpoints into a set of discretized camera points distributed on a dome that surrounds the object. This approach reframes the regression problem as a classification problem, which is easier for the neural network to solve. The inference result of SSD6D is shown in Figure 4.3

The other type of 3D object detection, on the other hand, does not require to predict roll and pitch because it assumes that the object's heading is parallel to the ground, without. This task typically focuses on the autonomous driving scenario. Despite not needing to handle the two tricky angle variables, 3D object detection in driving scenes remains

doi:10.6342/NTU202301439

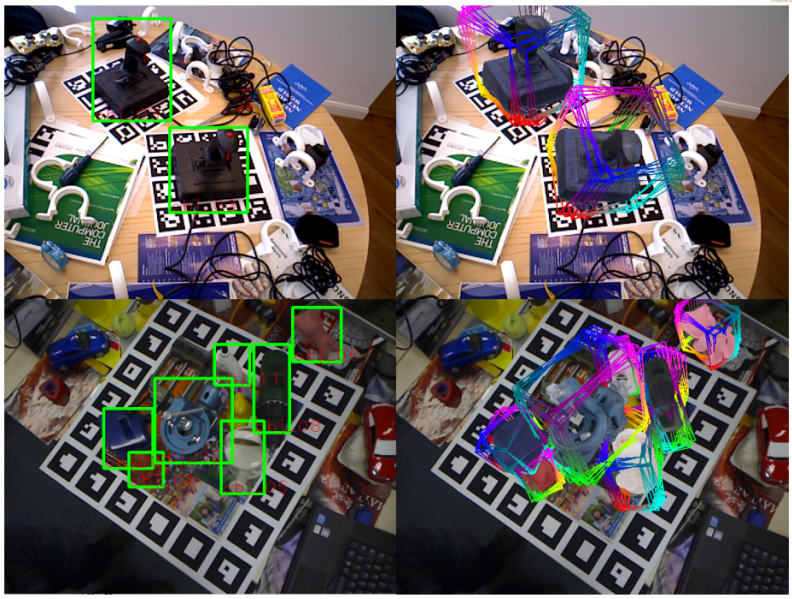


Figure 4.3: Prediction result produced by SSD6D[27]. Their work focuses on in-door scene and predicts objects with various camera viewpoint.

challenging due to the varying object sizes distributed along the depth axis. To tackle this problem, there are two main approaches that differ in their input data types. LiDAR-based methods take point cloud data as input and leverage the accuracy and abundance of such data, while image-based methods use monocular images from cameras and need to deal with an ill-posed issue. In this thesis, we focus on monocular 3D object detection in driving scenes and will discuss related works in this field in the following section.

4.2.1 LiDAR-based 3D Object Detectors

A frame of LiDAR point cloud typically contains over 10,000 points, with each point represented by its location (displacement from the origin in the x, y, and z-axis) and reflectance r. PointNet [47] was the first work to use a neural network to process point cloud data. PointNet utilizes a multi-layer perceptron (MLP), Batch Normalization Layer, and ReLU to transform raw point cloud data into features and applies max pooling to each channel to ensure the permutation invariance of the network. However, despite its simple

doi:10.6342/NTU202301439

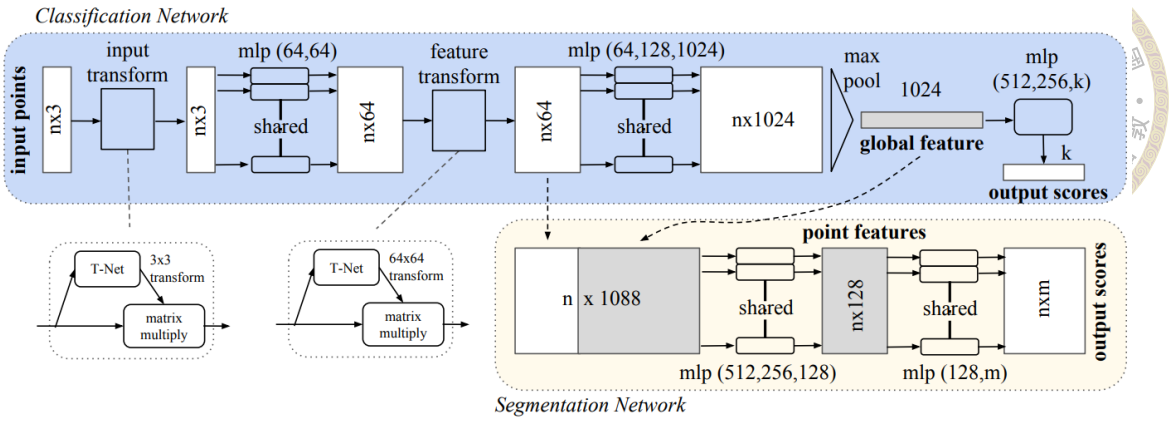


Figure 4.4: PointNet network architecture[47]. It utilizes multiple layer perceptrons (MLPs) to learn features from point clouds. The max pooling operation is applied to ensure the network's permutation invariance.

architecture, PointNet is computationally expensive and slow due to the need to process each point independently.

VoxelNet [77] takes a different approach by dividing the 3D space into 3D voxel cells and placing all LiDAR points in them. After randomly sampling points contained within each voxel, a fixed-length vector is produced to represent each voxel based on the properties of the contained points. Next, a voxel vector extractor transforms the vector into a feature, and the resulting 4D tensor is fed to a 3D convolutional layer to extract spatial features. Lastly, based on the encoded features, a standard anchor-based region-proposal network is used to predict 3D boxes on the Bird's Eye View (BEV) plane. As a result, VoxelNet significantly reduces computational costs by grouping the point cloud into voxels instead of processing points directly, and its end-to-end architecture is more efficient and effective for 3D object detection.

SECOND [68], which followed VoxelNet, argues that the 3D convolutional layer is too slow and uses a sparse convolutional layer to replace it, resulting in a 4x speedup during inference. In contrast, PointPillars [31] uses "pillars" instead of "voxels" to divide the point cloud, where pillars can be considered as voxels with infinite height, PointPillars

doi:10.6342/NTU202301439

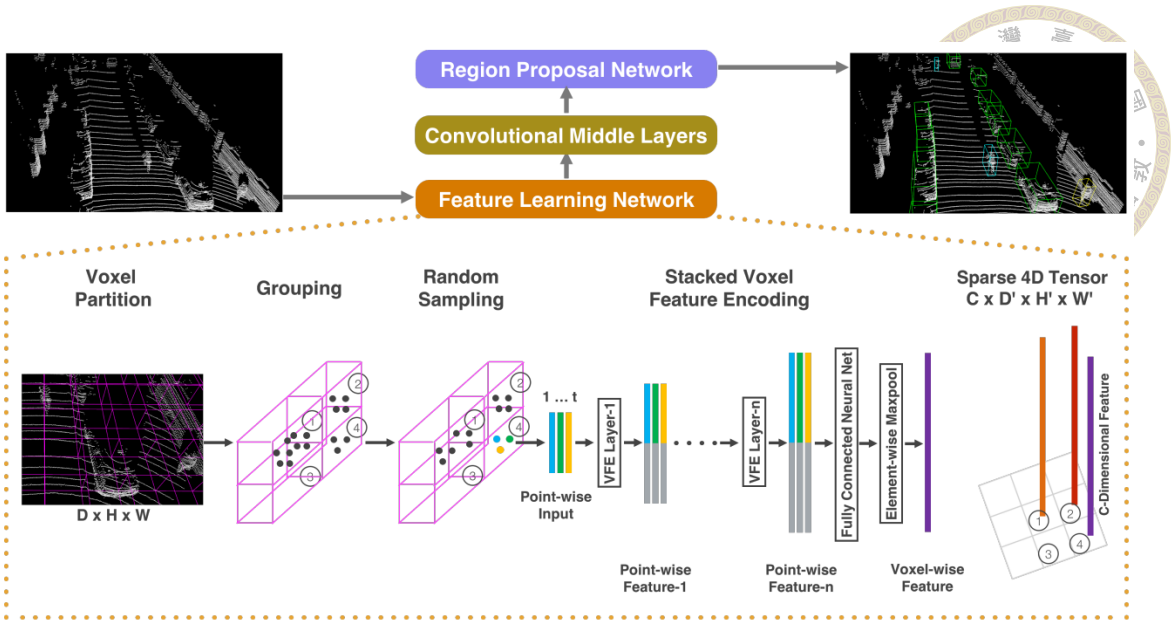


Figure 4.5: VoxelNet network architecture[77]. VoxelNet divides the point cloud into a set of 3D cuboid spaces called voxels. It then extracts features from each voxel based on the points contained within it. Finally, it utilizes 3D convolutional layers to extract features among neighboring voxels.

avoids the most computationally expensive part of the network, which is the 3D convolutional layer, resulting in an even faster network.

LiDAR-based methods for 3D object detection can capture precise 3D bounding boxes due to their accurate sensors, which provide depth information by actively emitting lasers to hit objects. However, the sparsity of point clouds can make processing point clouds a challenging task. Although VoxelNet and PointPillars have addressed this issue by grouping spare points into voxels or pillars, the uneven point distribution still poses problems for the network, with VoxelNet reporting that approximately 97% of the voxels in 3D space is empty. Another drawback of using LiDAR is cost, as the sensor itself is expensive and requires an extended computational platform to support it. Installing LiDAR is also difficult, as it needs to be placed on a block-free platform, typically on top of the car, to obtain a 360-degree view. This can make the mechanical design of the car very hard. In contrast, cameras are cheaper, easier to install, and widely used sensors that have been deployed in cars for years. Consequently, researchers have recently shifted their fo-

doi:10.6342/NTU202301439

cus towards monocular image-based methods for 3D object detection, hoping to address these challenges.

4.2.2 Monocular 3D Object Detectors

Monocular 3D object detection takes a single camera image as input and finds the 3D bounding box in the image. Because the image can be stored as a 2D array with RGB channels and can be processed by the CNN layer directly, monocular 3D object detection is a fast and practical solution for self-driving vehicle. Despite its great potential, inferencing depth from a monocular image has been considered as an ill-posed problem that is tricky to tackle. Still, this new research topic gain a lot of attention for many researchers due to its potential; in this section, we put the related works into five categories and introduce them in the following section.

4.2.2.1 Two-stage detectors

Due to the well-developed research done in the 2D object detection task, it's straightforward to use the 2D detector to localize object on the image and estimate its 3D bounding box based on the 2D box prior. Therefore, many of the early works follow this paradigm and treat 3D object detection as a two-stage task.

Deep3DBox[45] is one of the classical networks that exploit the geometrical constraint given by 2D object detectors. Deep3DBox assumes that all 3D bounding box projected corners must inscribe its 2D bounding box, i.e., each side of the 2D bounding box must be coincided with one of the 3D bounding box corners. With the 2D-3D geometric tight constraint, Deep3DBox is able to determine 3D boxes given their predicted orienta-

doi:10.6342/NTU202301439

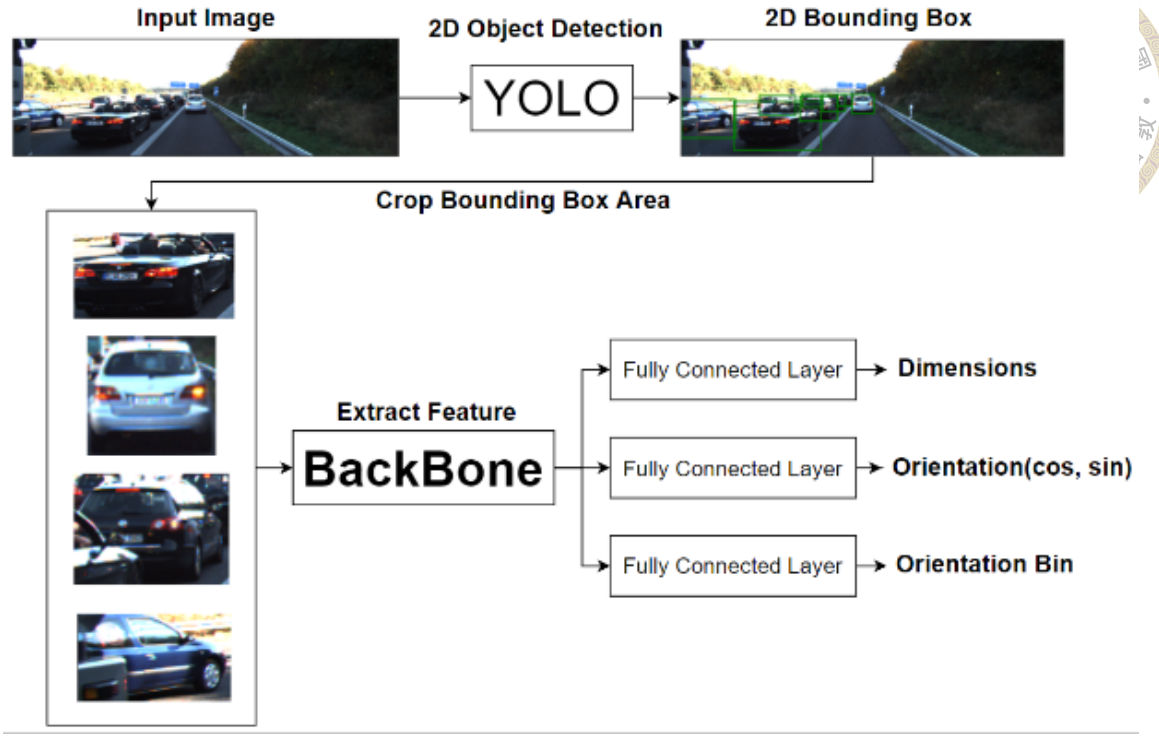


Figure 4.6: Deep3DBox[45] network architecture.

tion and dimension. Deep3Dbox also replaces the orientation regression task with Multi-Bin classification task in the hope of having a better orientation estimation. We show the network architecture of Deep3DBox in Figure 4.6.

Despite its simple architecture, Deep3DBox suffers greatly from performance drop when the 2D bounding boxes are not accurate and the prediction quality heavily relies on 2D box prediction. FQNet[36] tries to solve this issue by sampling dense 3d box candidates around the tight 2D-3D bounding constraint with Gaussian function and uses an additional network to evaluate the fitting quality between candidates and objects.

An alternative approach pool the feature from the region of the 2D box on the feature map and predict the 3D box geometry according to it. This helps the network to narrow down the possible location of objects and filter out most of the background pixel on image. This approach has shown promising results in improving the performance of object detection networks. MonoDIS[56], ROI10D[44], MonoGRNet[48], and MonoPSR[30]

doi:10.6342/NTU202301439

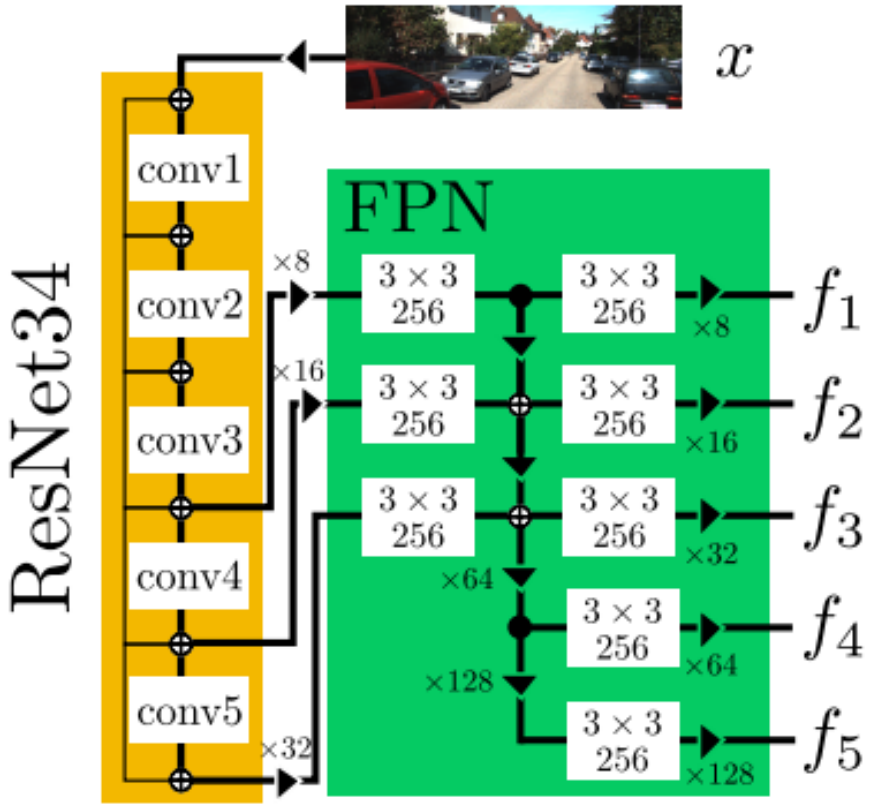


Figure 4.7: MonoDIS network architecture[56]. It uses a simple ResNet34 as a feature extractor and FPN as a neck to extract multi-scale features. It is an anchor-based architecture, and the paper mainly focuses on a disentangled loss function that enhances training efficiency.

are all use this apporach. MonoDIS[56] uses a single-stage 2D detector as the first stage with an additional 3D detection head, which uses 2D box proposals and RoIAlign to pool features from the feature map. MonoDIS further proposes an innovative disentangle loss to prevent interference between each loss term and help it to converge faster. We show the MonoDis network architecture in Figure 4.7.

Similarly, ROI10D[44] uses a pre-trained monocular depth estimator to obtain the depth map from the image and fuse it with the image feature map. ROI10D uses the fused features to predict eight projected corners of a 3D box on the image to get a better estimation of the object's orientation. Additionally, ROI10D proposes a method to generate 3D mesh car models from training images and LiDar points to paste synthetic car models on

doi:10.6342/NTU202301439

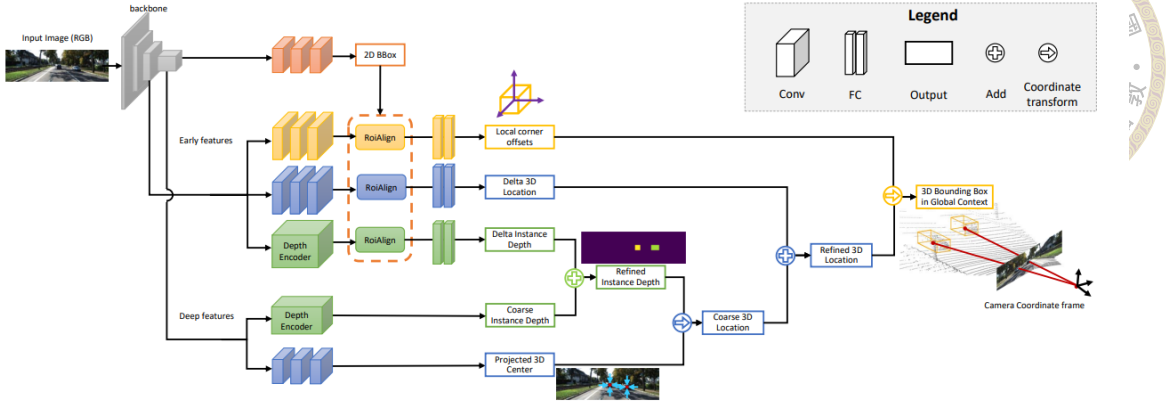


Figure 4.8: MonoGRNet[48] network architecture. It uses four subnetworks to predict the 2D box, 3D box location, 3D box projected corners, and instance depth.

the image to enrich the dataset by showing unseen object poses in new locations.

MonoPSR[30] uses 2D bounding boxes to crop regions of interest (ROI) from both the image and feature map and uses a geometric constraint to generate 3D object proposals similar to Deep3DBox's method. The special part of the network is its instance reconstruction module, which generates point cloud data from the image feature and uses LiDAR data to train the network to learn depth estimation for every instance. MonoGRNet[48] uses four subnetworks to predict the 2D box, 3D box location, 3D box projected corners, and instance depth. Additionally, it utilizes early features in the backbone to capture small objects. We show the MonoGRNet architecture in Figure 4.8.

In conclusion, using a 2D detector in the first stage is a common and effective approach for monocular 3D object detection. The approach has the advantage of reducing the imbalance problem between positive and negative examples and focusing on recognizing visual cues in the RoI. However, the accuracy of 3D box predictions is heavily reliant on the quality of 2D box proposals. If the network predicts an inaccurate 2D window, the 3D box prediction may be incorrect due to the wrong geometric constraint or misaligned features pooled from the RoI. Additionally, predicting object depth solely based on RoI

doi:10.6342/NTU202301439

features may cause the network to ignore valuable information in another part of the scene. This limitation is reflected in the evaluation of the KITTI performance metric, where most methods have less than 20% AP on easy objects in the validation set.

4.2.2.2 Single-stage detectors

The other approach views a 3D object as the extension of a 2D object. It uses a single-stage, unified architecture to predict 2D and 3D boxes in paralleled branches. It can be viewed as adding a few more variables in the 2D bounding box regression branch to get the additional 3D box geometry.

For example, M3D-RPN[8] reformulates 2D detector network to capture the 3D proposal and use a shared network for both tasks. It also uses statistical data from the training set to determine its 3D anchor box prior, which serves as an initial guess. M3D-RPN also proposes a depth-aware convolution, which uses separate kernels to extract features from different rows of images. This depth-aware local feature will be fused with a global feature map with an attention map. M3D-RPN also uses a 2D-3D box post-estimation algorithm to optimize the estimated orientation. Ground-aware[41] is very similar to M3D-RPN, but it uses a ground-aware feature extraction network to infuse object and ground contact point features together. This allows the network to take more parts of the scene into consideration and slightly improve its depth estimation ability. The architecture of Ground-aware network is shown in Figure 4.9

Anchor-free methods are other popular architecture in object detection and are typically derived from the keypoint-based detector proposed in CenterNet[76]. CenterNet introduces a novel approach to object detection by formulating it as a keypoint detection

doi:10.6342/NTU202301439

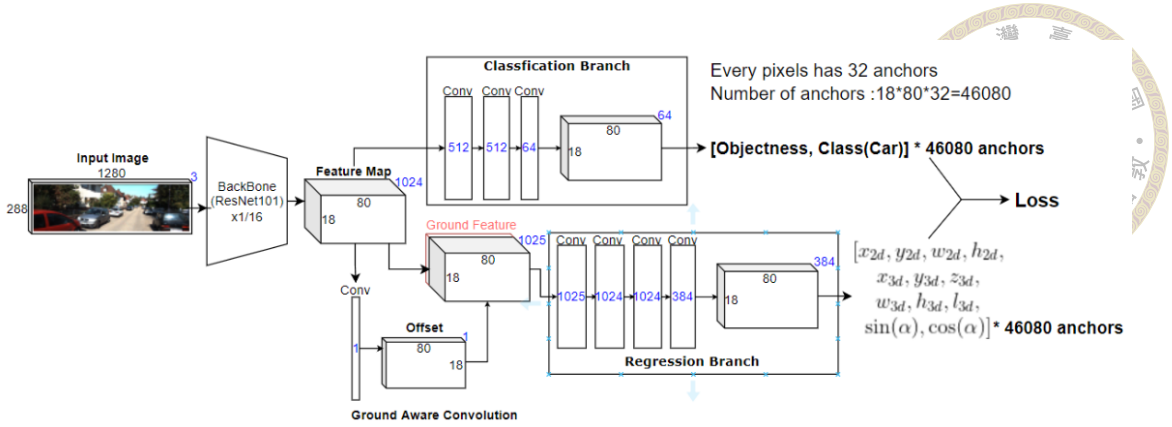


Figure 4.9: Ground-aware network architecture[41]. It utilizes two parallel branches to predict the object category and 3D box geometry. The network leverages the ground contact point to aid in predicting object depth, and thus includes a dedicated subnetwork to locate the contact point and extract features from that specific location.

task where the center of the bounding box represents the object. It employs DLA-34 as backbone and predicts a heatmap to identify the top 100 potential object centers. With additional regression branches, CenterNet can also perform 3D object detection and pose estimation. Since it defines only one anchor at each location, CenterNet can be viewed as a one-stage detector that eliminates the need for non-maximum suppression(NMS) and anchor IoU calculation. This simplifies the architecture and allows CenterNet to achieve remarkable speed and simplicity.

The simplicity of CenterNet's proposed architecture has attracted many other researchers to follow this paradigm. RTM3D [32] finds objects by predicting nine key points, which consist of eight corners of 3D box and one object center. RTM3D also proposes a novel feature pyramid network for key point estimation to capture multi-scale keypoint on image. SMOKE [42] further simplifies the network by eliminating the 2D box regression branch and improves convergence by importing MonoDIS's disentangle loss. We show the SMOKE network architecture in Figure 4.10. MonoPair[16] focuses on exploiting the relationship between adjacent objects by adding a key point at the middle point between each adjacent object pair. Furthermore, MonoPair introduces uncertainty

doi:10.6342/NTU202301439

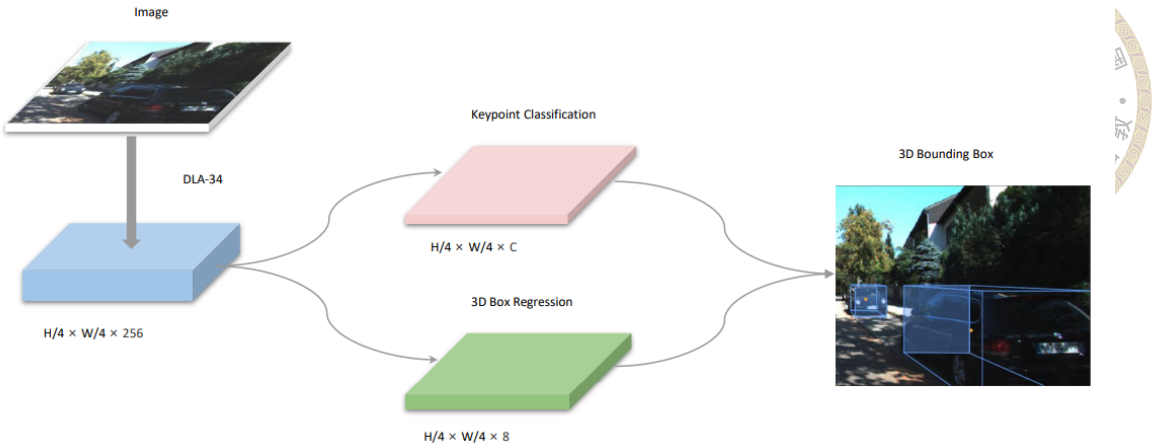


Figure 4.10: SMOKE network architecture[42]. Instead of using anchors, this keypoint-based network identifies the center of the 3D box in the image to locate objects, while another branch regresses the 3D box geometry. It can be seen as a single-anchor network that operates efficiently but may sacrifice some accuracy compared to anchor-based methods.

in the regression branch and uses them to optimize detection results in a post-processing step. MonoFlex[74] concentrates on dealing with truncated objects whose center lay outside of the image and proposes an edge fusion module to decouple feature learning and truncated object prediction. MonoFlex uses an ensemble to predict depth, which considers depth uncertainty and multiple pixel height of the predicted 3D bounding box to enhance depth prediction.

Overall, Keypoint estimation-based methods are very fast compared to other methods. Due to their end-to-end and simple architecture, they can achieve real-time detection on limited equipment. Despite their impressive speed, keypoint methods have limited accuracy on truncated objects and need to use an additional module to tackle them. Compared to anchor-based methods, the anchor-free method only generates a single detection at each feature location, resulting in worse performance.

doi:10.6342/NTU202301439

4.2.2.3 Representation Transformation

The limitation of monocular images to predict depth has been widely discussed, and some works have shown that this difficulty is not just due to the lack of direct depth information in the image, but also because of the suboptimal representation of the data on the image plane. To overcome this performance bottleneck, a line of work has tried to convert the image pixel data to the bird's-eye view (BEV) plane or 3D space. These methods typically utilize pre-trained models to assist in data conversion and require LiDAR-point or stereo-image data during training.

For instance, BirdGAN[58] utilizes an image-to-image GAN network to convert monocular images to BEV images, and then applies a detection head to find 3D objects within the BEV plane. Pseudo-Lidar[20] proposes a two-stage scheme where a pre-trained depth estimation model is used to convert the image to a pseudo-Lidar point cloud in 3D space, and then a standard LiDAR-based 3D detector is used to predict 3D boxes with the pseudo point cloud. To improve [20], Pseudo-Lidar++[70] uses a stereo depth estimation network to replace the monocular one and adopts sparse LiDAR data to guide the depth map and align it with 3D space landmarks. We show the pipepline of Pseudo-Lidar in Figure 4.11. In contrast, OFTNet[52] converts the image features to the BEV plane using the perspective projection of voxels in 3D space, without requiring any additional training data. It can then directly find 3D bounding boxes within the BEV plane. These approaches leverage the use of pre-trained models and alternative data representations to address the limitations of monocular images for 3D object detection.

After transforming image features into a 3D representation that explicitly includes depth, the accuracy of 3D box depth regression can be significantly improved, particularly

doi:10.6342/NTU202301439

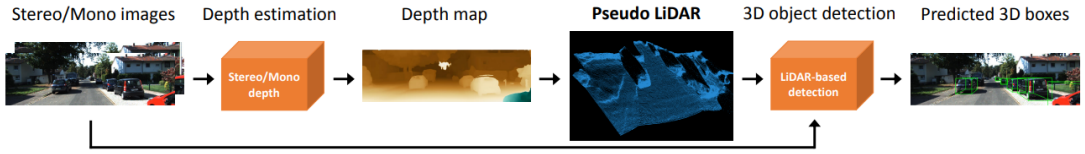


Figure 4.11: Pseudo-Lidar network architecture[20]. It proposes a 3D object detection pipeline that leverages a pre-trained depth estimation model to obtain a depth map. The image pixels are then mapped onto 3D space based on the depth values. Finally, a pointcloud-based 3D object detector is applied. By utilizing depth information, Pseudo-Lidar typically achieves higher accuracy compared to other methods.

for objects located at greater distances. However, these methods often involve multiple stages and require additional information during training. Due to their complex architectures, there is still a need for further research to improve the speed and efficiency of these methods to enable real-time applications.

4.3 Proposed Methods

4.3.1 Backbone Improvement

The feature extractor, or backbone, of a neural network processes the input image and extracts features or patterns from it. The quality of the backbone is crucial for network performance, as it determines the upper bound of the network's accuracy. ResNet[26] is one of the most popular backbones due to its ability to alleviate the gradient vanishing problem in deep neural networks using residual connections. These connections allow gradients to propagate through the network without being significantly diminished during backpropagation.

Based on ResNet, ResNext[66] introduced a new dimension, cardinality, in designing neural networks. The cardinality of the network is defined by the number of parallel branches it contains. By splitting a single propagation path into multiple parallel ones,

doi:10.6342/NTU202301439

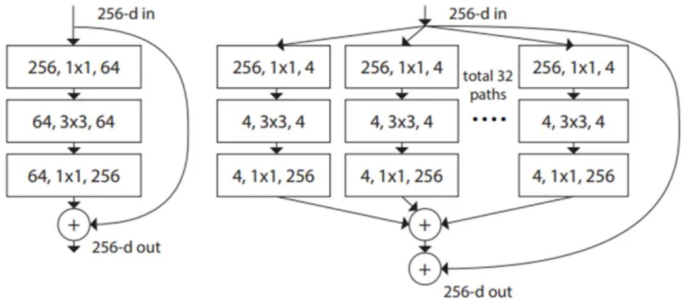


Figure 4.12: ResNext architecture. The left graph shows the ResNet skip connection and the right graph shows the improved module of ResNext.

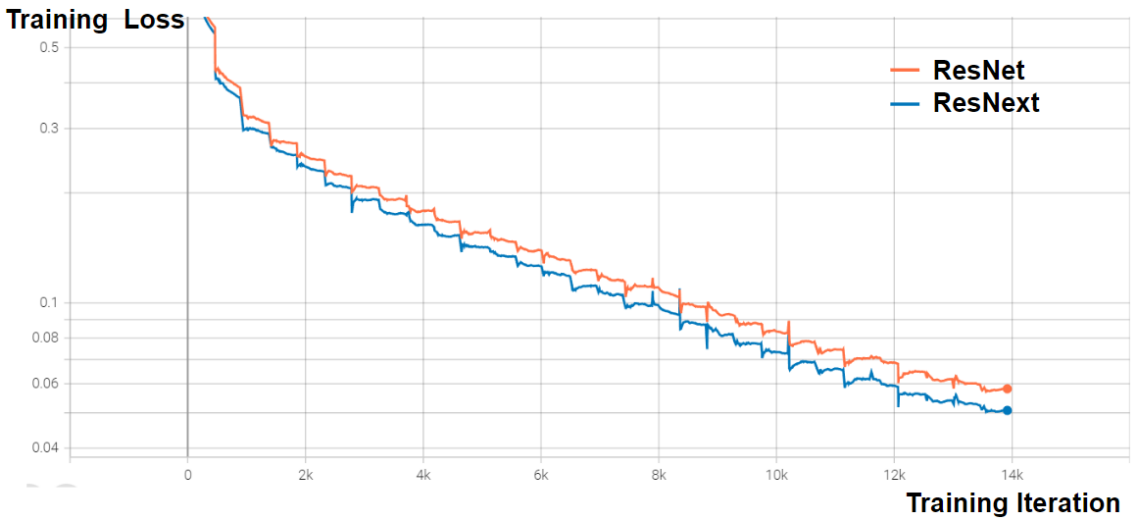


Figure 4.13: Training loss of ResNet and ResNext.

ResNext achieves high cardinality and shows significant improvements compared to its predecessors on ImageNet, CIFAR, and COCO datasets. The multiple paths of ResNext is shown in Figure 4.12.

We upgrade the backbone of our 3D object detector model from ResNet-101 to ResNext-101, and the performance improvement is shown in Figure 4.14. As can be seen, using ResNext can produce an increase of approximately 1% mAP in almost all difficulty AP metrics in KITTI dataset. Furthermore, throughout the training process, the training loss is consistently lower than that of the original backbone, as shown in Figure 4.13. This indicates that ResNext is easier to train and converges faster, thanks to the multiple parallel branches in its design.

doi:10.6342/NTU202301439

	AP 3D			AP BEV		
	Easy	Moderate	Hard	Easy	Moderate	Hard
ResNet-101	**22.23**	15.82	13.06	28.44	20.65	17.5
ResNext-101(32x8d)	21.99	**16.85**	**14.23**	**28.74**	**21.73**	**18.69**

Figure 4.14: Experiment result of ResNext[66] backbone

4.3.2 Perspective-aware Convolution

In this section, we explore ways to improve convolution layers' feature extraction ability. We begin by reviewing the related work in this field and then propose our perspective-aware convolution module, which is tailored to 3D object detection and can enhance the model's ability to estimate depth. Our module is designed to extract features that are highly relevant to depth estimation and can capture long-range dependencies between different parts of the image.

4.3.2.1 Convolutional Modules

Convolutional layers are a fundamental component of deep learning models, responsible for extracting features from the input data by convolving a learnable kernel with the input in a sliding-window fashion. In most image-based tasks, such as object detection, a 3x3 convolution kernel is commonly used to capture local features from the image. The convolution operation produces a strong response when the pattern within the 3x3 region of the image aligns with the kernel parameters, allowing the convolution layer to extract important patterns from the input image. By stacking multiple layers of convolutional layers, the convolution network can learn to extract increasingly higher-level features, such as human faces or cars.

Convolutional layers are designed to capture local patterns in the image, which limits

doi:10.6342/NTU202301439

ability to model long-range dependencies and extract global information. To overcome this limitation, deep convolutional networks often resort to increasing the network depth and receptive field size. The receptive field refers to the region size that a deep layer feature pixel considers when convolving with the input image. However, with the growth of image research, the ability of convolution layers to sufficiently capture the receptive field has been put to the test, particularly in segmentation tasks.

In the field of image segmentation, it was found that simply increasing the depth of a convolutional neural network for modeling long-range dependencies in an image is inefficient. To address this issue, [71] proposed the use of atrous or dilated convolutions, which increase the receptive field of the convolutional layer without compromising the feature map resolution. Dilation convolution works by skipping a few image pixels when convolving the kernel with the image. The number of skipping pixels is controlled by a dilation rate parameter. Varying the dilation rate of the convolutional kernel enables features to interact with farther locations and gradually increase the receptive field. They experimented with dilation rates of 1, 2, 4, 8, and 16, resulting in a +5.4% improvement in their test results for the VOC-2012 semantic segmentation task. This is because pooling layers that were previously used to gain greater receptive fields in convolutional networks reduced the resolution of the image, making it difficult to output high-resolution masks for dense prediction tasks. By contrast, dilation convolution allows the network to enlarge the receptive field, making it particularly effective in segmentation tasks. The dilation convolution is depicted in Figure 4.15.

In DeepLabv2[11], the authors proposed the use of atrous spatial pyramid pooling (ASPP) to extract multi-scale features from the same feature map using parallel dilated convolutions. Unlike [71] which uses dilated convolutions in consecutive layers, ASPP

doi:10.6342/NTU202301439

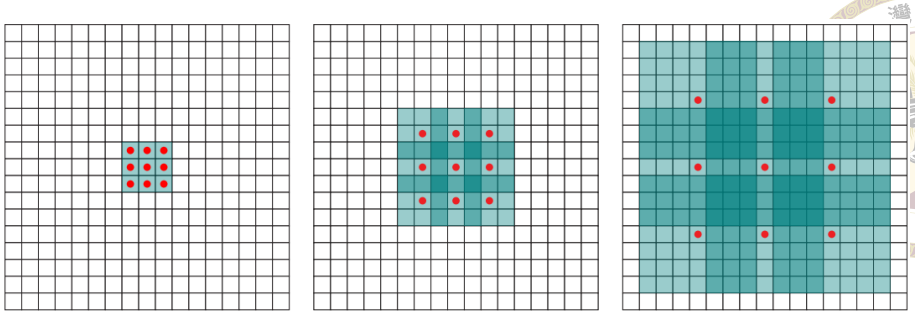


Figure 4.15: Receptive field of dilated convolution with different dilation rates. The red dots represent the pixels convolved with the kernel, and the green region indicates the receptive field. As the dilation rate increases, the receptive field expands.

uses four branches with different dilation rates to extract multi-scale features simultaneously. The output feature maps from these branches are then concatenated and passed through a 1×1 convolution layer to reduce the number of channels. According to their experiments, ASPP produced roughly 1.5 mean intersection-over-union (mIoU) improvement on the PASCAL VOC 2012 validation set.

In RFB[38], the authors proposed a receptive field block that combines the ideas of InceptionNet and ASPP. It uses dilation convolution with different scale kernel sizes, with a design that includes a 3x3 convolution layer with a 3x3 kernel size and other branches that use a 5x5 kernel size with a dilation rate of 5. This design is argued to be similar to how the human perceptual system works. In their experiments, they incorporated the receptive field block into the SSD network and achieved a 2.4% improvement in object detection performance on the PASCAL VOC 2007 test set. RFB's network architecture is shown in Figure 4.16.

To address the limitations of regular convolutional layers in effectively increasing the size of the receptive field, the authors of DCN[19] proposed the use of deformable convolutional networks (DCN). Instead of using pre-defined convolution locations, DCN uses a simple convolutional network to learn pixel locations for convolution. For example, if the kernel size is 3x3, the network takes the feature map as input and outputs a 3x3

doi:10.6342/NTU202301439

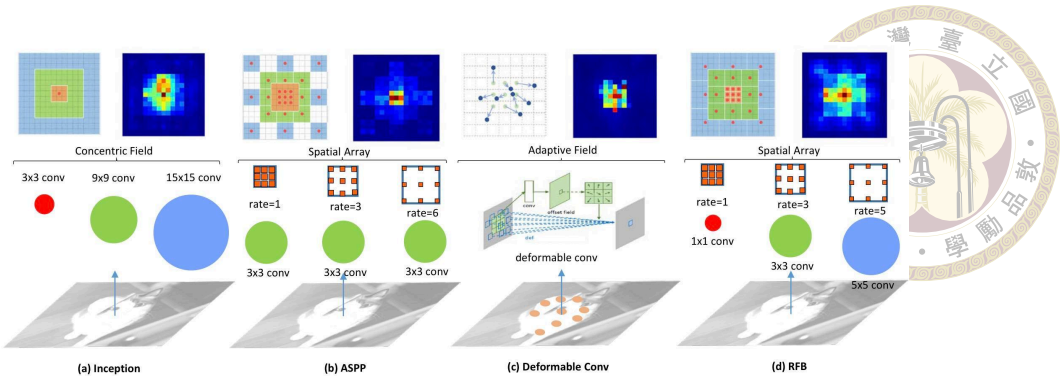


Figure 4.16: Comparsion of InceptionNet[59], ASPP[11], Deformanble Convolution[19], and Receptive Field Block[38]

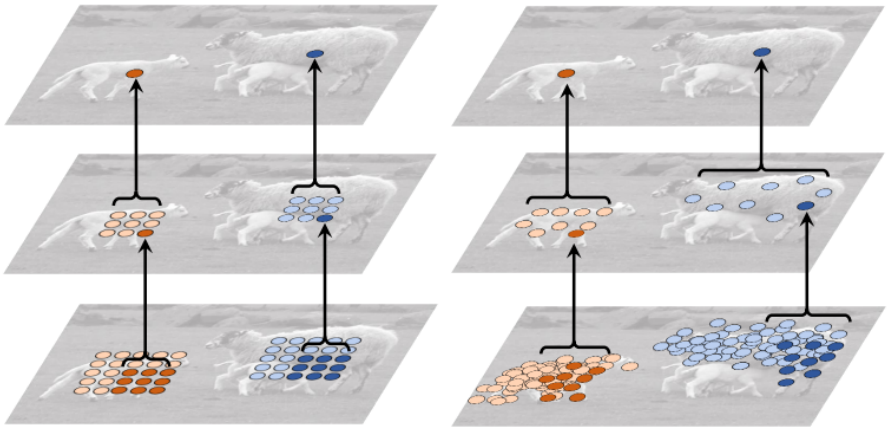


Figure 4.17: Deformable convolution[19]. Instead of using a fixed kernel shape, it incorporates a learnable subnetwork to determine the convolved feature locations. This allows the convolutional layer to flexibly extract image features and adapt to different object sizes.

and x and y offset to indicate how far the location needs to be offset from the kernel center. The DCN approach is shown in Figure 4.17. Although this method introduces more flexibility to the training network, it also causes a non-negligible overhead during training. Therefore, in this paper, the authur propose that stacking three DCN layers is a good compromise, and it produces a +5.5% mIoU improvement in the PASCAL VOC 2007 test set. To further enhance the flexibility of the network, DCNv2[78] proposes the use of an additional network to learn a mask that adjusts feature amplitudes at each location. This allows the network to enhance features at specific locations or to neglect features at other locations, resulting in improved performance on various tasks.

doi:10.6342/NTU202301439

After surveying many research studies on designing convolution layers, none of them incorporate the image scene structure or perspective projection into their design. We believe that depth estimation is a unique process that requires additional design considerations. In the next section, we will introduce our proposed perspective-aware convolution module based on ASPP. We aim to address this challenge and enhance the ability of 3D object detectors to estimate the depth of objects accurately. Our module can extract features that are highly related to depth estimation and model long-range dependencies to improve overall performance.

4.3.2.2 Perspective-aware Convolution Module

We propose a perspective-aware convolution module that takes into account the perspective projection and extracts features along the projection line. Our work is closely related to ASPP[11] and DCN[78]. While the DCN module uses a subnetwork to learn the extracted feature location during training, our PAC directly uses perspective projection to determine the feature location, reducing the number of parameters. Therefore, our proposed method is faster to train and has little overhead on performance.

Despite the absence of depth information in camera images, we posit that the human perceptual system is capable of inferring depth from limited visual cues by leveraging other scene information. For instance, as depicted in Figure 4.18, humans can infer the location of objects based on the relative positions of other nearby objects in the scene, exploiting their understanding of scene structure. By comparing the distances between adjacent objects, depth can be estimated even in cases where occlusion occurs. Hence, we believe understanding scene structure plays a pivotal role in accurate depth estimation. Our objective is to integrate this perspective information into our convolutional neural

doi:10.6342/NTU202301439

Figure 4.18: Illustration of perspective lines parallel to the depth-axis in camera coordinate and projected onto the image plane. The long-range dependency along the perspective line is crucial for accurate object depth prediction.

network.

We aim to capture features along the perspective lines, which are straight lines parallel to the depth axis in 3D space. To achieve this, we project these straight lines onto the image plane. In order to simplify the representation, we introduce the concept of perspective angle, which refers to the angle between each projected line and the u-axis on the image coordinate system. Each pixel on the image has its own perspective angle, which is determined by the pinhole camera model. In the subsequent section, we will provide an explanation of how we derive the perspective angle.

To get the perspective angle, we begin with the formulation of the pinhole camera model:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{Z_w} \begin{bmatrix} f_x & 0 & C_x \\ 0 & f_y & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} \tag{4.7}$$

In Equation 4.7, (X_w, Y_w, Z_w) represents the coordinates of a point in the camera coordinate system, while (u, v) represents the projected pixel coordinates on the image plane. The parameters f_x and f_y denote the focal lengths expressed in pixel units for

doi:10.6342/NTU202301439

the u and v axes. The values C_x and C_y correspond to the principal point, which is the intersection point of the optical axis with the image plane.

To determine the amount of pixel displacement caused by a change in depth, we differentiate Equation 4.7 with respect to Z_w and obtain:

$$\begin{cases} \dfrac{du}{dZ_w} = -\dfrac{X_w f_x}{Z_w^2} \\ \dfrac{dv}{dZ_w} = -\dfrac{Y_w f_y}{Z_w^2} \end{cases} \tag{4.8}$$

where (X_w, Y_w, Z_w) represents a 3D point in camera coordinates, which is inversely projected from a pixel (u, v). Since the depth of each pixel is unknown, we assume that all image pixels lie on the ground plane and set Y_w equal to the height of the ground plane, denoted as Y_0. By applying Equation 4.7, we can inversely project (u, v) back to the camera coordinate system and the inversely projection equation is as following:

$$\begin{cases} X_0 = \dfrac{(u_0 - C_x) Y_0 f_y}{(v_0 - C_y) f_x} \\ Y_0 = Y_0 \\ Z_0 = \dfrac{Y_0 f_y}{v_0 - C_y} \end{cases} \tag{4.9}$$

Here, (u_0, v_0) represents a specific pixel on the image, and (X_0, Y_0, Z_0) represents the inversely projected 3D point in camera coordinates. By substituting (X_0, Y_0, Z_0) into (X_w, Y_w, Z_w) in Equation 4.8, we can calculate the derivatives and the perspective angle ϕ is determined by the following equation:

$$\phi = \text{atan2}\left(\frac{dv}{dZ_w}, \frac{du}{dZ_w}\right) \tag{4.10}$$

doi:10.6342/NTU202301439

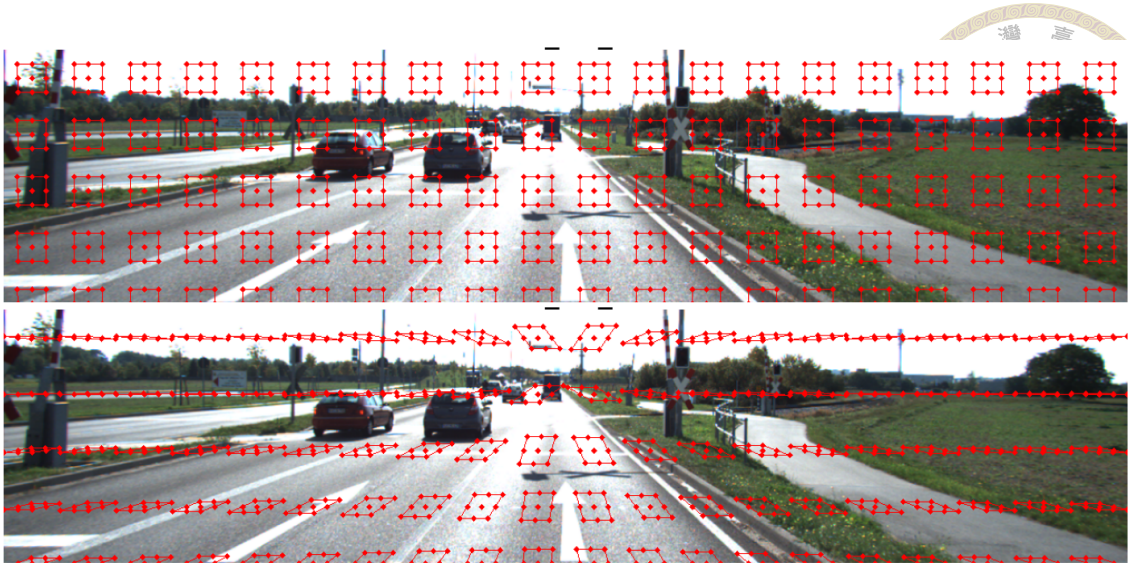


Figure 4.19: Perspective-aware convolution. The top image shows the dilation convolution and the bottom image shows our perspective-aware convolution. Our PACs' dilated kernel will convolute with different feature locations depending on the perspective angle.

Using Equations 4.8 and 4.10 , we can calculate the perspective angle for each image pixel (u, v). This perspective angle allows us to guide kernel to change their shape according to its pixel coordinate and perspective angle, as illustrated in Figure 4.19.

In addition to a single PAC convolution layer, we adopt the multiple-branch design proposed in ASPP[11] and use different dilation rates for each branch. A comparison between ASPP and our proposed PAC module is depicted in Figure 4.20.

4.4 Experiment Result

We evaluated the performance of our proposed perspective-aware convolutional (PAC) layer on the Ground-aware [41] network, an anchor-based 3D object detector, using the KITTI3D dataset. We utilized ResNet-101 as the network backbone and trained the model for 30 epochs. In addition to the PAC layer, we experimented with several introduced convolution layers, including dilation convolution, deformable convolution [19], ASPP [11], and RFB [38]. We applied the dilation convolution setting in the last three convolutional

doi:10.6342/NTU202301439

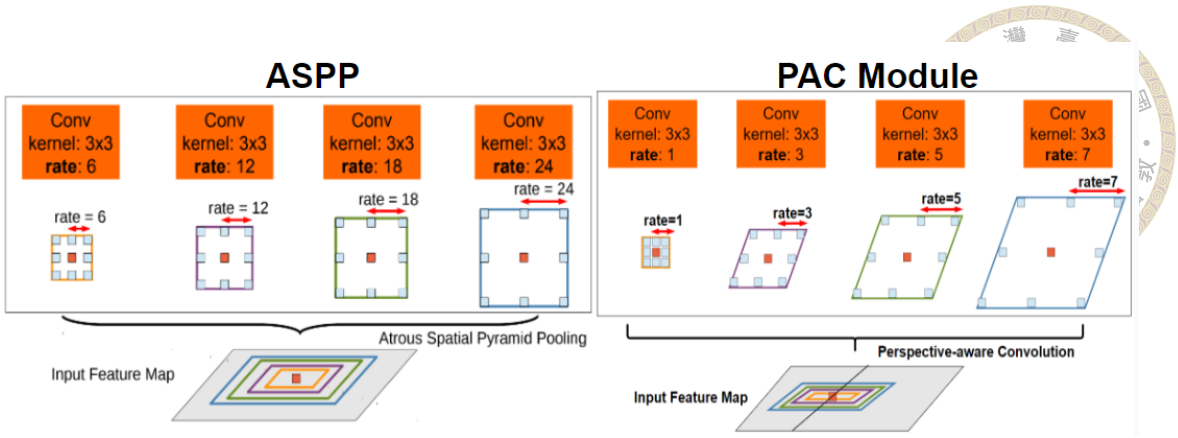


Figure 4.20: Comparison of PAC module and ASPP module. Both modules utilize parallel branches to capture multi-scale features. However, the key distinction lies in the kernel shape employed for feature extraction. While the ASPP module utilizes a regular kernel shape, the PAC module incorporates a tilted kernel shape to guide feature extraction along the perspective line.

layers of the backbone and set the dilation rate to (2,2,2) and (3,3,3) in our experiment. We also add DCN, RFB, ASPP, and PAC after the backbone, following their respective recommended settings. We conducted experiments with a single layer of PAC, where the dilation rate is set to two. As for the PAC module, we set the dilation rate to 2, 4, 6, and 8 in each parallel branch. We reported our experiment result in Figure 4.21.

As shown in our experimental results in Figure 4.21, our proposed PAC module outperforms all other methods and achieves an improvement of +1.59% AP compare to baseline in the 3D metric with moderate difficulty. In contrast, dilation convolution showed no improvement compared to the baseline. Deformable convolution had a slight improvement with a single layer, but its performance dropped after using more than one DCN layer. We suspect this is because DCN introduces too many parameters to train, making it easier to overfit the training data, especially since KITTI's training set is small.

RFB showed roughly the same performance as the baseline, while ASPP showed a fair improvement, especially with hard-difficulty objects. This indicates that far or truncated objects require long-range information to predict their depth accurately. We suspect

doi:10.6342/NTU202301439

Method	3D			BEV		
	Easy	Moderate	Hard	Easy	Moderate	Hard
Baseline	22.08	15.64	13	28.56	21.08	17.48
Dilation Convolution(2,2,2)	20.32	14.07	12.06	27.23	19.7	16.72
Dilation Convolution(3,3,3)	17.93	12.93	10.71	24.93	17.93	15.68
DCNv2(layer1)	22.13	16.2	13.44	29.19	21.58	18.57
DCNv2(layer2)	20.21	15.52	13.23	29.03	21.78	18.93
DCNv2(layer3)	20.21	14.58	12.24	28.12	19.79	17.11
RFB	21.32	15.62	12.94	28.53	21.26	18.35
ASPP	22.44	16.96	14.23	29.69	22.2	19.03
PAC(Ours)	22.71	15.73	13.05	30.55	21.85	18.56
PAC Module(Ours)	**23.53**	**17.23**	**14.33**	**30.57**	**22.44**	**19.07**

Figure 4.21: Experimental results of different convolutional modules. DCNv2 shows a mild improvement with the one-layer setting. ASPP also demonstrates a good improvement, indicating that a larger receptive field is beneficial for the 3D object detection task. Our proposed PAC and PAC module both perform well compared to other methods, highlighting the importance of injecting perspective information into the network.

that the performance difference between RFB and ASPP is due to the dilation rate, as a larger dilation rate is needed to capture distant features.

We also tested different settings for the PAC layer and showed the results in Figure 4.22. We found that the best dilation rate for the PAC layer is five. If the dilation rate is too large, many pixels near the boundary will end up sampling the output of the image and getting a trivial feature from the zero-padding region. Conversely, a dilation rate of only one is too small to capture long-range information.

Addtionally, we compare of our proposed method with other 3D object detectors in the same experiment settings. The results are shown in Figure 4.23.

As we can see from the results in Figure 4.23, our proposed method outperforms the other 3D object detector, including the Ground-aware network which our method is based on. We split the methods into two categories based on whether they require depth maps during inference or not. For methods that do not require depth maps, our method performs

doi:10.6342/NTU202301439

Dilation Rate	3D			BEV		
	Easy	Moderate	Hard	Easy	Moderate	Hard
1	20.39	14.7	12.31	27.29	19.53	16.73
2	22.37	15.55	12.81	29.78	20.54	17.29
3	**22.59**	15.82	12.97	30.03	21.61	18.24
4	21.49	15.39	12.84	29.17	20.9	17.66
5	22.14	**15.88**	**13.08**	**30.09**	**21.79**	**18.57**
6	20.15	14.73	12.11	28.45	21	17.15
7	21	15.39	12.69	29.21	21.25	18.12
8	20.52	15.3	12.57	29.03	20.72	17.48
9	19.54	14.91	12.5	27.07	20.88	18
10	19.81	14.83	12.27	28.27	20.87	17.7

Figure 4.22: Experiment showing the influence of different dilation rates on the performance of a single PAC layer. The best dilation rate is five.

Methods	AP 3D(IoU=0.7)			AP BEV(IoU=0.7)			AP 2D		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
w/o depth map:									
SMOKE	6.96	4.30	3.98	12.73	7.93	6.94	82.91	74.20	67.11
DD3D	19.16	15.27	13.37	25.72	20.78	18.38	80.06	81.50	76.58
MonoFlex	22.14	16.19	14.18	29.30	21.91	18.82	97.95	92.57	85.05
Ground-aware	21.90	16.06	13.17	28.29	20.98	17.59	96.36	87.67	80.27
Ours	**23.90**	**17.51**	**14.76**	**31.61**	**22.89**	**19.58**	**98.40**	**89.32**	**82.17**
w/ depth map:									
Pseudo-LiDAR	61.51	43.33	36.85	75.58	55.58	48.25	92.45	79.87	69.91

Figure 4.23: Experiment result of 3D object detectors on the KITTI3D validation set. Our proposed method is an improved version of the Ground-aware network, where we replace the backbone with ResNext101 and incorporate the PAC module. As a result, our method achieves the best performance compared to other methods.

doi:10.6342/NTU202301439

the best, outperforming SMOKE, DD3D, and MonoFlex. Although DD3D and MonoFlex are keypoint-based methods, they demonstrate promising results in 3D object detection, with performance close to the anchor-based network.

For methods that require depth maps, the pseudo-LiDAR method easily surpasses all the methods that do not use depth maps. This is because it utilized depth map information and can get object depth information explicitly from it. In addition, pseudo-LiDAR converts the image pixel to point cloud data, which is a better representation for 3D object detection.

To illustrate the improvement of our proposed method in 3D object detection, we show some detection examples in Figure 4.24 and Figure 4.25. As we can see, after applying our improvement, the network is able to regress more accurate 3D bounding boxes.

Figure 4.24: Comparison between our method and Ground-aware[41]. The left image shows the predicted bounding boxes, while the right image in the Bird's Eye View (BEV) corresponds to it. With the PAC module, our network is able to more accurately regress 3D boxes. This is evident in the BEV, where our method's predicted boxes are closer to the ground truth.

To facilitate a detail comparison of different 3D object detectors, we present visualizations of their inference example on the KITTI3D validation set, as shown in Figures

doi:10.6342/NTU202301439

Figure 4.25: Comparison between our method and Ground-aware[41]. The left image shows the predicted bounding boxes, while the right image in the Bird's Eye View (BEV) corresponds to it. With the PAC module, our network is able to more accurately regress 3D boxes. This is evident in the BEV, where our method's predicted boxes are closer to the ground truth.

4.26, Figures 4.27, Figures 4.28, Figures 4.29, Figures 4.30, Figures 4.31. In these figures, the left column shows the predicted 3D bounding box projected onto the image plane. However, to accurately evaluate the 3D box location, we recommend that readers use the bird's-eye-view (BEV) provided in the right column. In the BEV figures, the yellow box represents the ground truth and the green box represents the predicted result.

Based on these results, we can observe some interesting traits of each detector. Firstly, despite the impressive accuracy of Pseudo-LiDAR in 3D box estimation, it can sometimes incorrectly predict the object orientation in pretty obvious cases. This is because Pseudo-LiDAR converts the image to a point cloud, sacrificing some advantages that are only available when the data is in image form. As for MonoFlex and Ground-aware, they perform roughly the same in this experiment, showing keypoint-based and anchor-based method both has potential in 3D object detection. DD3D, on the other hand, tends to generate too many false positives, although their confidence scores are low. This also highlights the advantage of anchor-based methods, where non-maximum suppression is

doi:10.6342/NTU202301439

applied to avoid similar issues.

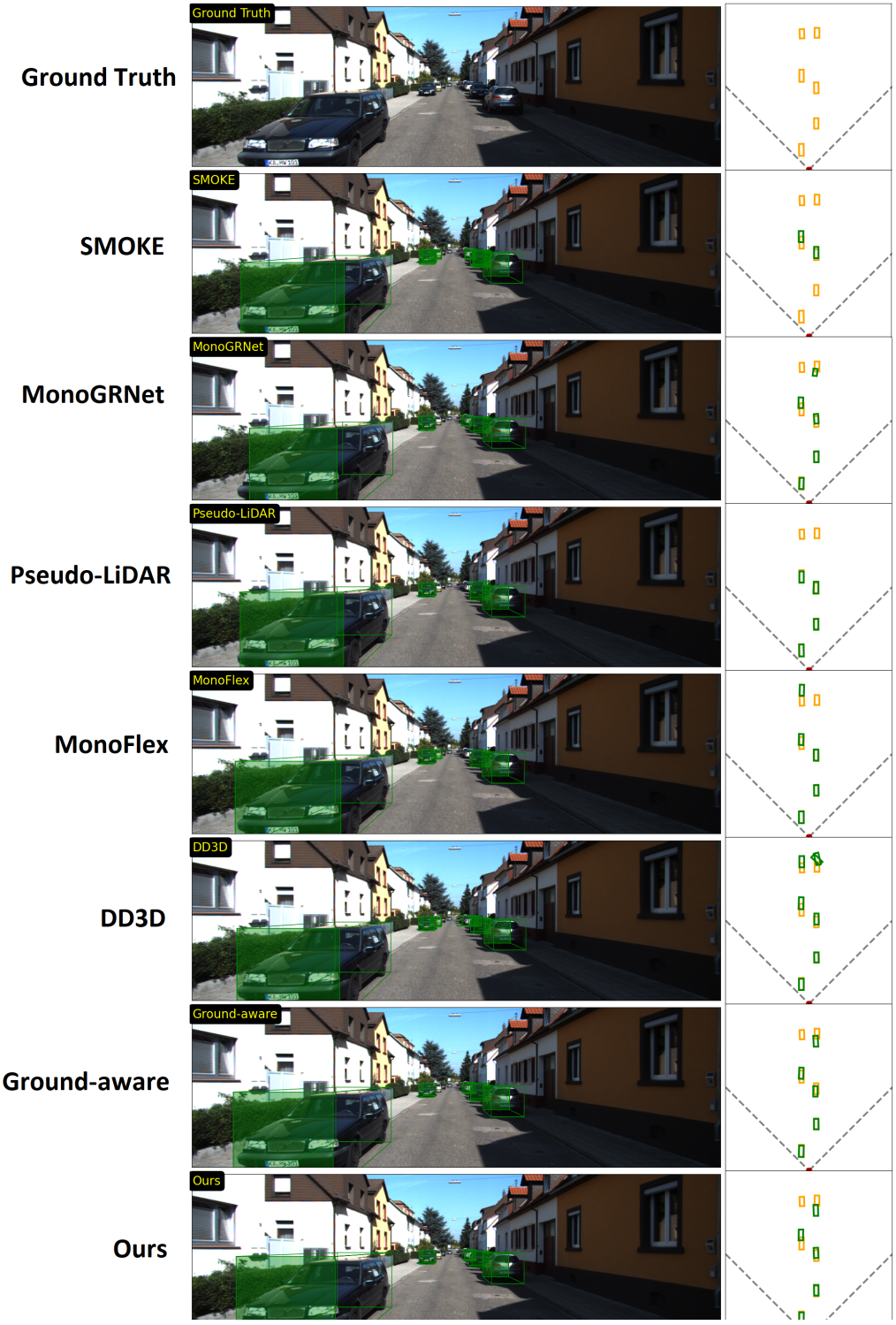


Figure 4.26: Inference example of 3D object detectors. The left column shows the predicted 3D bounding boxes in the image plane, while the right column displays the bounding boxes projected onto the bird's-eye-view (BEV) plane. The yellow boxes on the BEV represent the ground truth, while the green boxes indicate the predictions. In this experiment, we observe that most methods exhibit inaccuracies in predicting object depth, whereas our proposed method demonstrates more accurate depth estimation.

doi:10.6342/NTU202301439

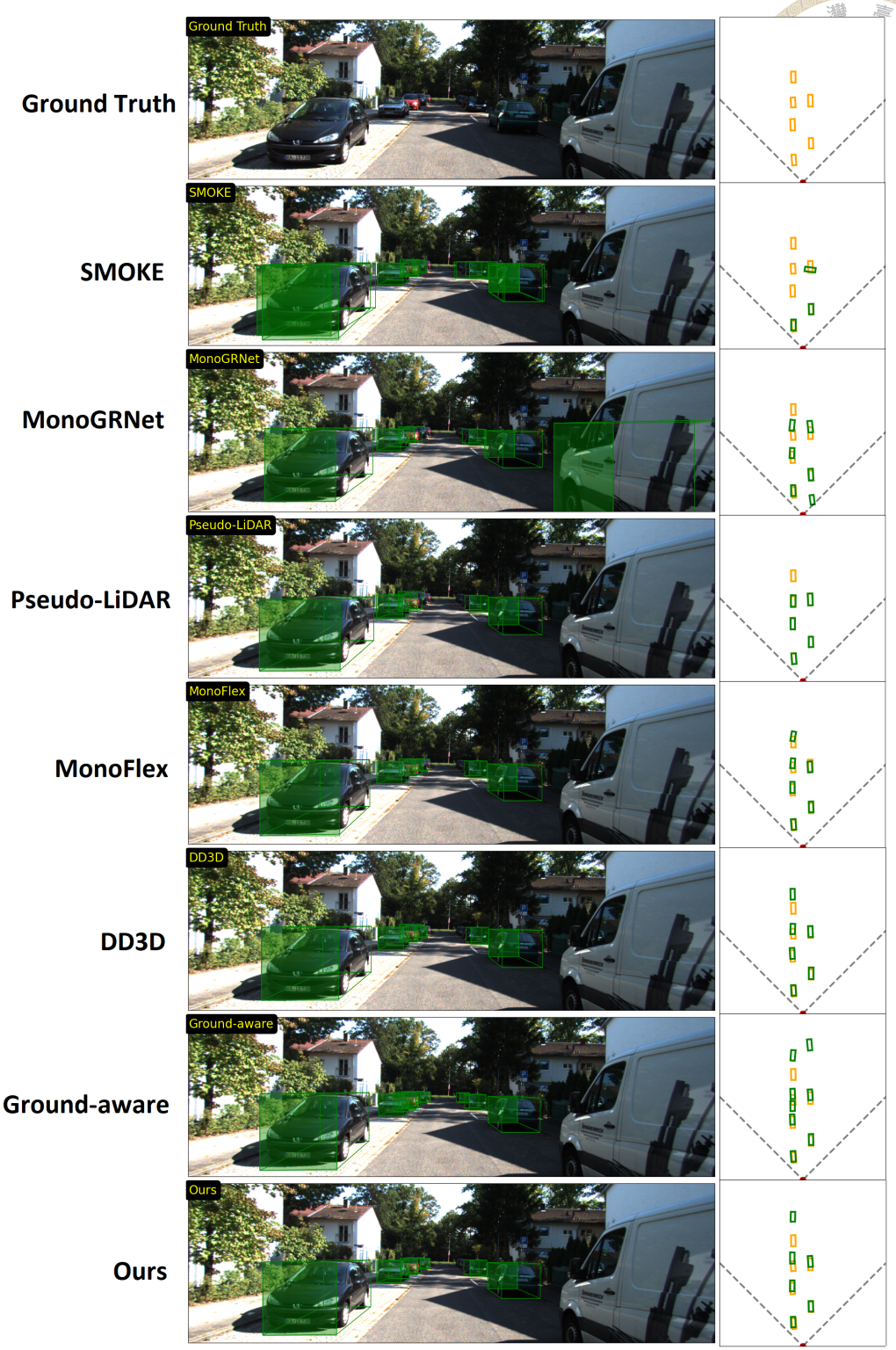


Figure 4.27: Inference example of 3D object detectors. The left column shows the predicted 3D bounding boxes in the image plane, while the right column displays the bounding boxes projected onto the bird's-eye-view (BEV) plane. The yellow boxes on the BEV represent the ground truth, while the green boxes indicate the predictions. In this experiment, we observe that most methods exhibit inaccuracies in predicting object depth, whereas our proposed method demonstrates more accurate depth estimation.

doi:10.6342/NTU202301439

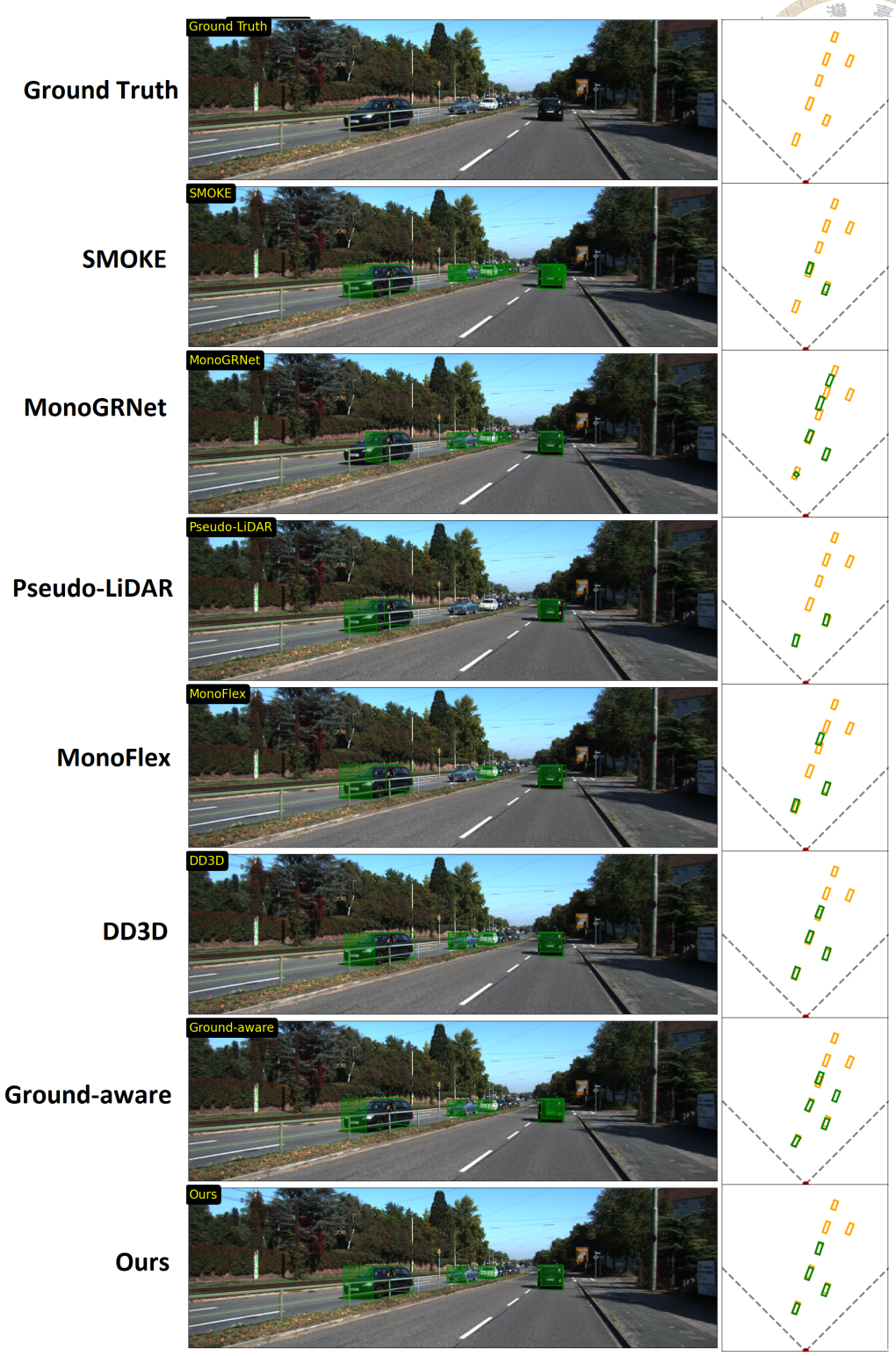


Figure 4.28: Inference example of 3D object detectors. The left column shows the predicted 3D bounding boxes in the image plane, while the right column displays the bounding boxes projected onto the bird's-eye-view (BEV) plane. The yellow boxes on the BEV represent the ground truth, while the green boxes indicate the predictions. In this experiment, we observe that most methods exhibit inaccuracies in predicting object depth, whereas our proposed method demonstrates more accurate depth estimation.

doi:10.6342/NTU202301439

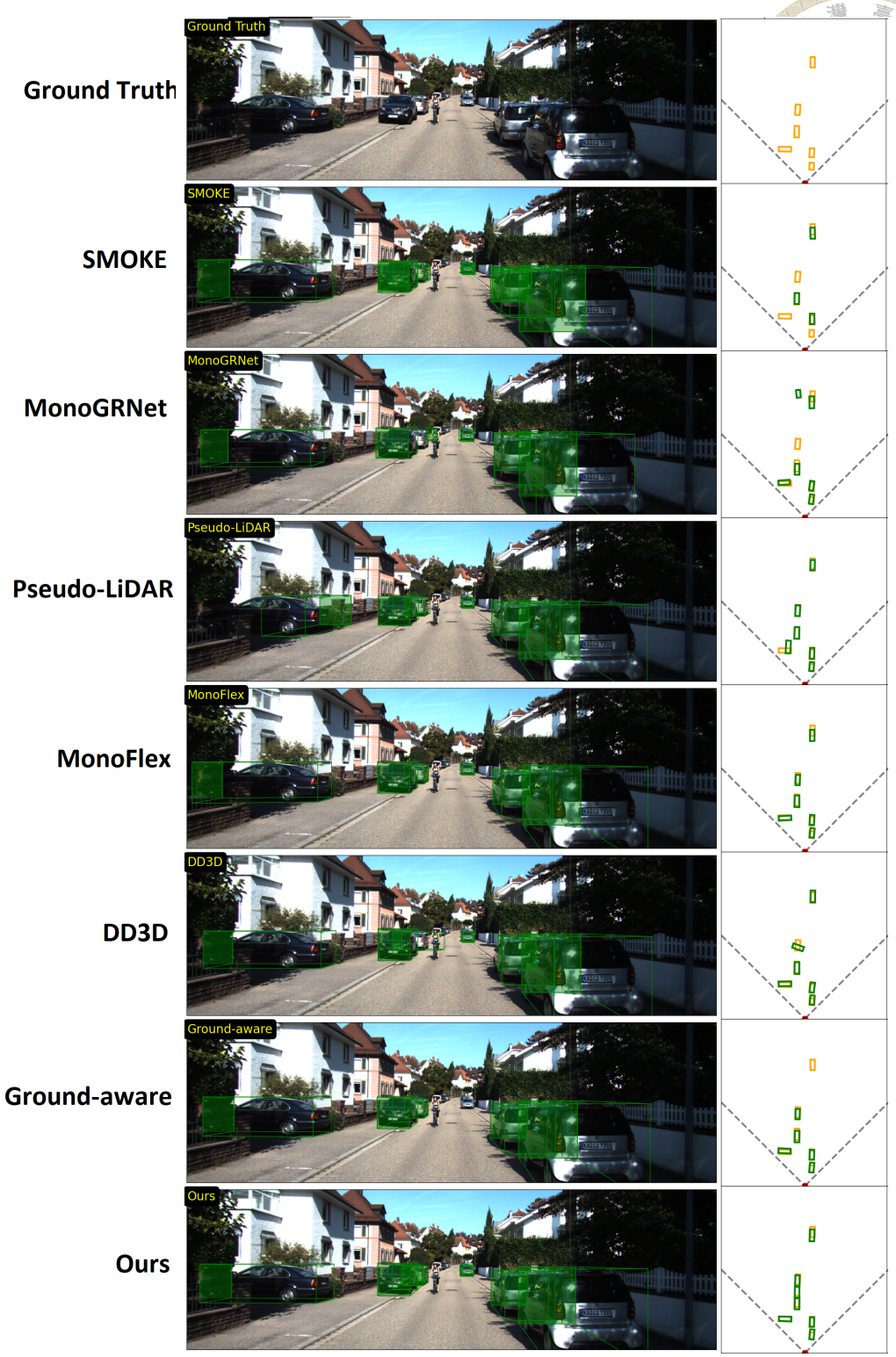


Figure 4.29: Inference example of 3D object detectors. The left column shows the predicted 3D bounding boxes in the image plane, while the right column displays the bounding boxes projected onto the bird's-eye-view (BEV) plane. The yellow boxes on the BEV represent the ground truth, while the green boxes indicate the predictions. In this experiment, we observe that most methods exhibit inaccuracies in predicting object depth, whereas our proposed method demonstrates more accurate depth estimation.

doi:10.6342/NTU202301439

4.5 Summary

In conclusion, our exploration of the monocular 3D object detection task shows the potential of this emerging image task in computer vision. Through our comprehensive survey of previous work, we identified five categories of approaches and discussed their respective advantages and disadvantages. We also proposed three methods to improve the anchor-based 3D object detector, namely upgrading the ResNet backbone to ResNext, using depth-aware anchor sampling, and incorporating perspective-aware convolution. By combining these methods, we developed a more accurate and efficient method for 3D object detection, which achieved a 23.9% AP in the easy difficulty of the KITTI3D validation set. Overall, our work contributes to the advancement of monocular 3D object detection, and we believe that our proposed methods can be further improved upon and applied to real-world scenarios, such as autonomous driving, to enhance the safety and accuracy of object detection.

doi:10.6342/NTU202301439

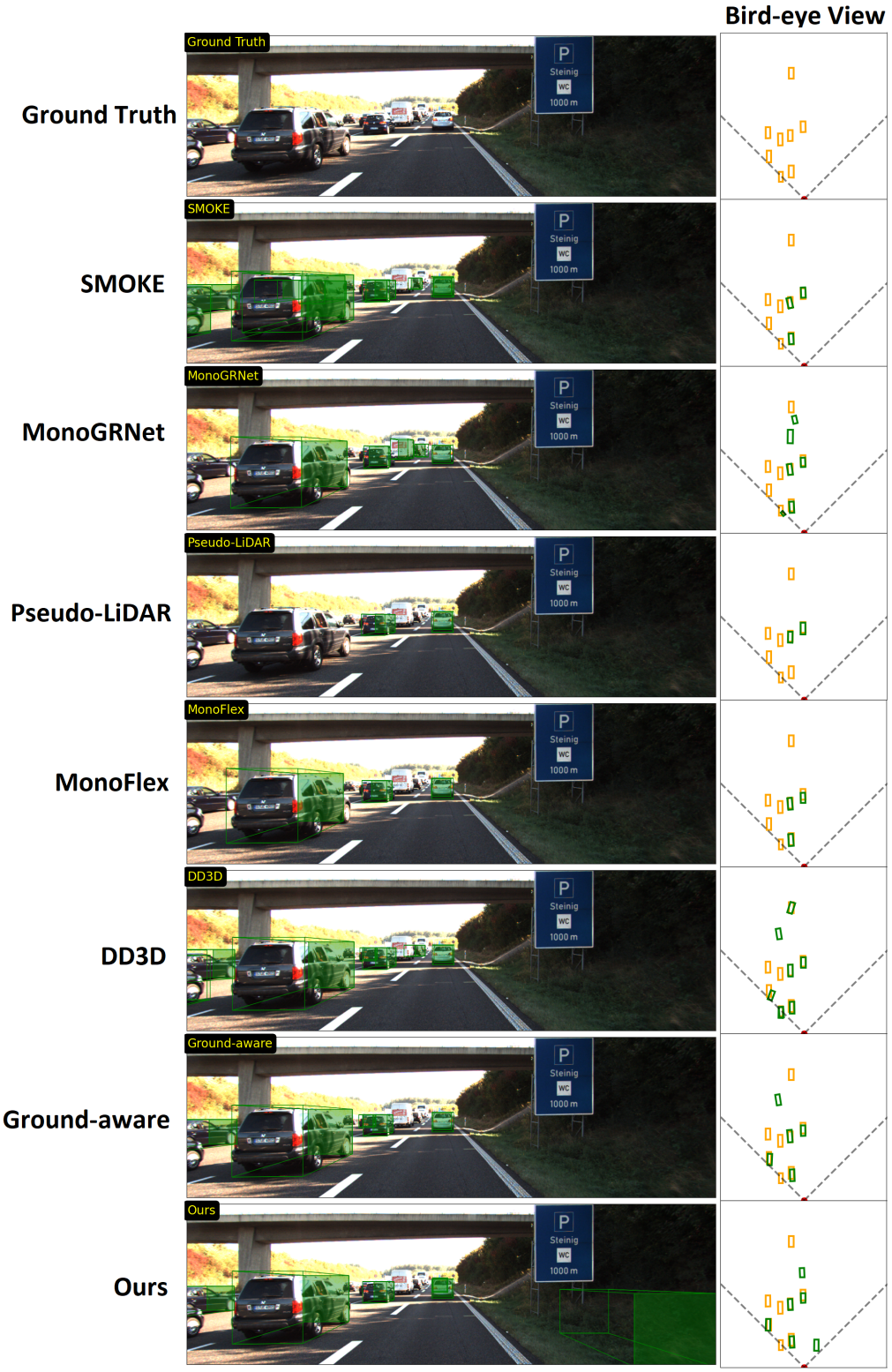


Figure 4.30: Inference example of 3D object detectors. The left column shows the predicted 3D bounding boxes in the image plane, while the right column displays the bounding boxes projected onto the bird's-eye-view (BEV) plane. The yellow boxes on the BEV represent the ground truth, while the green boxes indicate the predictions. In this experiment, we observe that most methods exhibit inaccuracies in predicting object depth, whereas our proposed method demonstrates more accurate depth estimation.

doi:10.6342/NTU202301439

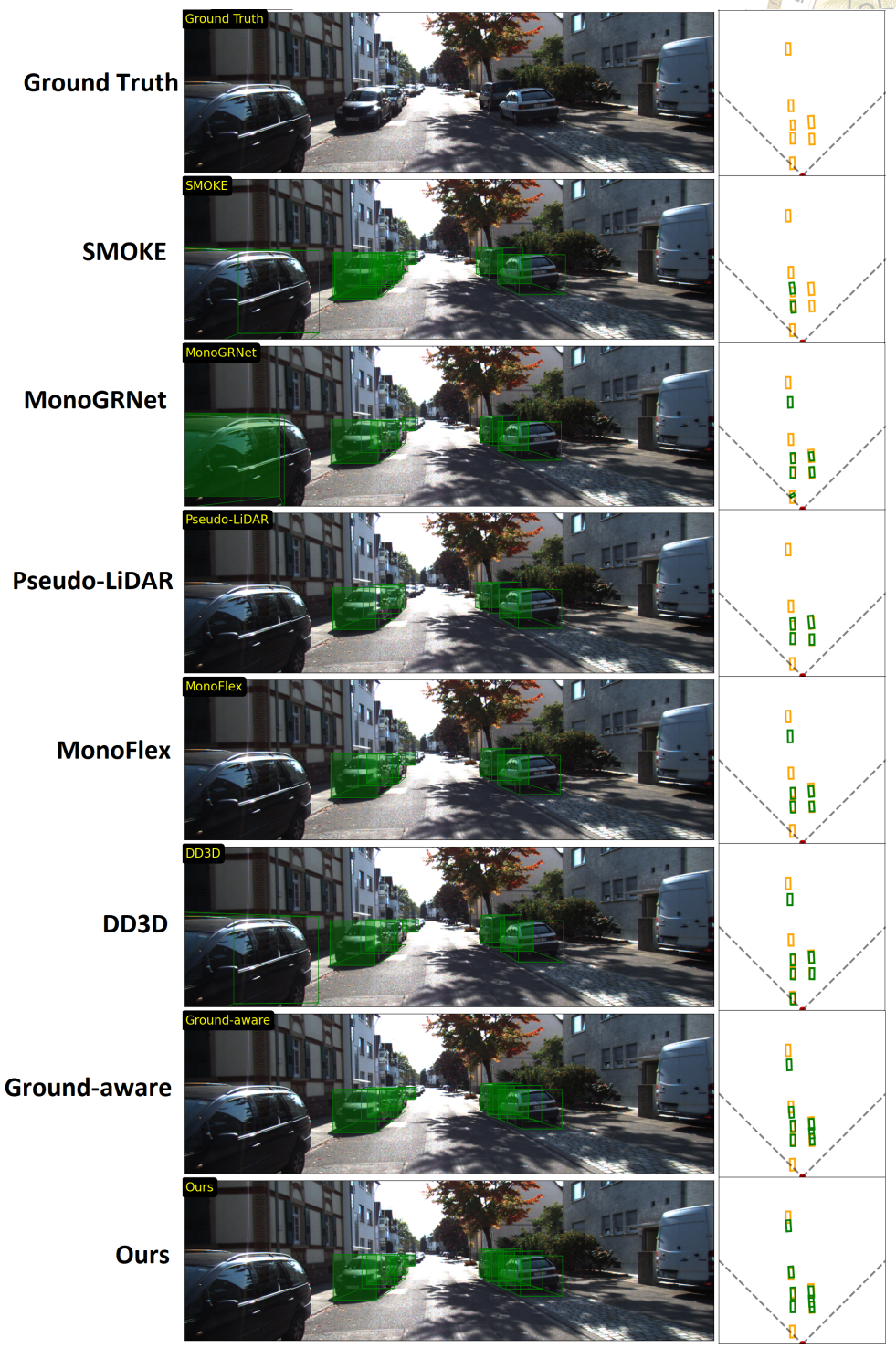


Figure 4.31: Inference example of 3D object detectors. The left column shows the predicted 3D bounding boxes in the image plane, while the right column displays the bounding boxes projected onto the bird's-eye-view (BEV) plane. The yellow boxes on the BEV represent the ground truth, while the green boxes indicate the predictions. In this experiment, we observe that most methods exhibit inaccuracies in predicting object depth, whereas our proposed method demonstrates more accurate depth estimation.

doi:10.6342/NTU202301439

Chapter 5 Image Segmentation

Image segmentation is a critical Computer Vision task that involves partitioning an input image into different segments or instances. Unlike humans, who are able to visually segment most images without encountering difficulty, segmentation computer algorithms are not easy or straightforward to design at all. However, segmentation is a fundamental tool for a wide range of applications, including medical image processing, autonomous vehicles, video surveillance, augmented reality, and image processing.

Furthermore, the success of Deep Convolutional Neural Networks (DCNN) in image classification has shown its great potential. One of the major challenges in image segmentation is to capture the fine-grained details of the image and accurately identify the boundaries of each segment. The powerful feature learning capability of DCNN, along with its ability to capture complex spatial relationships, makes it an ideal candidate for this task. Therefore, many researchers have developed and refined deep convolutional neural networks for image segmentation, resulting in significant improvements in segmentation accuracy and efficiency.

In recent years, research in image segmentation has broadly focused on three types of segmentation tasks: semantic, instance, and panoptic segmentation. Semantic segmentation involves assigning each pixel in an image to a particular class, such as ”car,” ”

doi:10.6342/NTU202301439

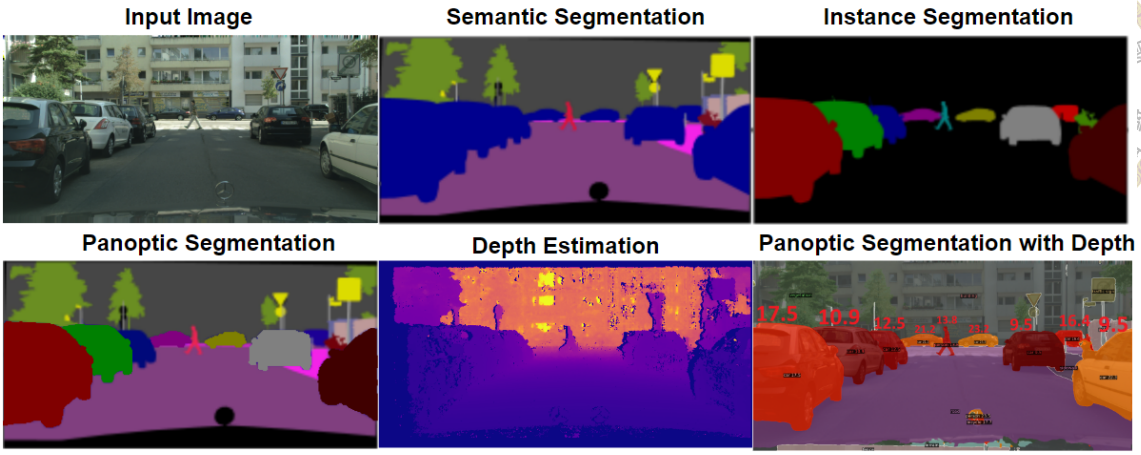


Figure 5.1: Image task taxonomy. Semantic segmentation classifies each image pixel into semantic categories. Instance segmentation identifies different instances within the foreground pixels. Panoptic segmentation combines both instance and semantic segmentation tasks. Depth estimation evaluates the depth value of each pixel. Lastly, our proposed task aims to integrate depth estimation and panoptic segmentation, producing a color map that represents the depth value for each instance.

person," or "tree.". Instance segmentation, on the other hand, focuses on segmenting out the foreground object in the scene and predicting the correct category for each object. Instance segmentation is useful when the goal is to identify and track individual objects in an image. Panoptic segmentation aims to unify semantic and instance segmentation into a single task by conducting semantic segmentation in background pixels and conducting instance segmentation in the foreground pixels. This task is still in its early stages of development, but it has the potential to provide a more comprehensive understanding of the visual world. The comparison of these three kind of segmentation tasks is shown in Figure 5.1.

In this chapter, we introduce a novel approach that combines panoptic segmentation with depth estimation to assign depth values to each instance in the scene. We propose a co-training network that simultaneously learns segmentation and depth estimation tasks, enabling the generation of a color map to visualize our results. By integrating these two tasks, we aim to provide a more comprehensive understanding of the scene, incorporating

doi:10.6342/NTU202301439

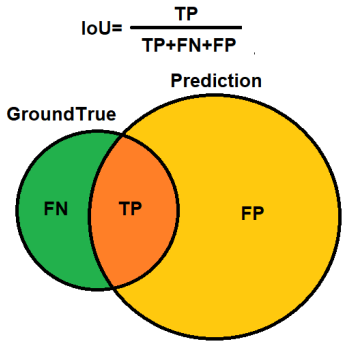


Figure 5.2: Intersection-over-union(IoU) calculation used in semantic segmentation

both semantic segmentation and depth information for each instance.

5.1 Evaluation Metric

As we mentioned in the previous section, there are three types of image segmentation tasks, and each has its own evaluation metric for determining the quality of predicted outcomes. In this section, we will introduce these metrics in detail.

For semantic segmentation, mean intersection-over-union (mIoU) is commonly used to evaluate the accuracy of a network in classifying pixels to each class. The mIoU is the average IoU of all categories, and the IoU for a single class i is defined as follows:

$$IoU_i = \frac{|TP|}{|TP| + |FP| + |FN|} \tag{5.1}$$

where $|TP|$ (true positive) is the number of correctly classified pixels belonging to class i, $|FP|$ (false positive) is the number of pixels wrongly classified to class i, and $|FN|$ (false negative) is the number of pixels that belong to class i but were not classified as such. The calculation of IoU is depicted in Figure 5.2 and mean-IoU is simply the average of IoU in all classes.

doi:10.6342/NTU202301439

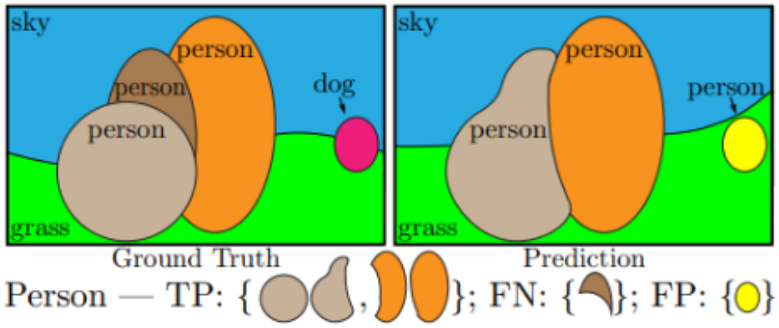


Figure 5.3: TP, FP, FN classification in panoptic quality metric calculation proposed by panoptic segmentation[29].

For instance segmentation, we inherit the Average Precision (AP) metric that is widely used in object detection. Instance segmentation can be seen as object detection with masks instead of bounding boxes, so it makes sense to directly calculate the IoU between the ground truth mask and the predicted mask and then calculate precision and recall to find the AP. We won't go into too much detail on how AP is calculated here, as it was already addressed in Chapter 2.

Panoptic segmentation requires a more complicated evaluation metric because it needs to evaluate both the pixel-wise classification result and the instance masks quality. The Panoptic Quality (PQ) metric was proposed by [29] specifically designed for this task. The PQ of a class is defined as follows:

$$PQ = \overbrace{\frac{\sum_{(p,g)TP} IoU(p,g)}{|TP|}}^{\text{Segmentation Quality(SQ)}} \times \overbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}^{\text{Recognition Quality(RQ)}} \tag{5.2}$$

The first term is segmentation quality (SQ), the average IoU of True Positives (TP). The second term is recognition quality (RQ), the F1 score of classification. The TP, FP, and FN in this metric are defined similarly to those in instance segmentation.

In the next sections, we will explore some of the most prominent deep learning meth-

doi:10.6342/NTU202301439

ods dealing with image segmentation tasks. Specifically, we will focus on their applications in the driving scene.

To assess the quality of panoptic segmentation results, Kirillov et al. introduced a novel metric called Panoptic Quality (PQ) [29]. PQ evaluates the accuracy of both semantic classification and the quality of predicted masks simultaneously. The PQ for a class is defined as follows:

$$PQ = \overbrace{\frac{\sum_{(p,g)TP} IoU(p,g)}{|TP|}}^{\text{Segmentation Quality(SQ)}} \times \overbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}^{\text{Recognition Quality(RQ)}} \tag{5.3}$$

where $|TP|$, $|FP|$, and $|FN|$ represent the number of true positive, false positive, and false negative respectively. The first term of the equation 5.3 is segmentation quality (SQ), which is the average IoU(Intersection over Union) of the True Positives masks. The second term is recognition quality (RQ), which is the F1 score of classification result.

To evaluate the depth estimation result, we adopt the following metrics:

Relative squared error:

$$\text{sqErr} = \frac{1}{N}\sum_{i=1}^{N}(\frac{d_i^* - d_i}{d_i})^2 \tag{5.4}$$

where d_i and d_i^* represent ground truth depth value and predicted depth value. And the N represent the number of pixel in image.

Relative absolute error:

$$\text{absErr} = \frac{1}{N}\sum_{i=1}^{N}|\frac{d_i^* - d_i}{d_i}| \tag{5.5}$$

doi:10.6342/NTU202301439

Inverse Root Mean Square Error:

$$\text{IRMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{d_i^*} - \frac{1}{d_i}\right)^2} \tag{5.6}$$

Scale invariant logarithmic error:

$$\text{SILog} = \frac{1}{N}\sum_{i=1}^{N} x_i^2 - \frac{1}{N^2}(\sum_{i=1}^{N} x_i)^2 \tag{5.7}$$

where $x_i = logd_i - logd_i^*$

Accuracy with threshold:

$$\delta_i = max(\frac{d_i}{d_i^*}, \frac{d_i^*}{d_i}) < t \tag{5.8}$$

where $t \in [1.25, 1.25^2, 1.25^3]$, It represents the percentage of image pixels whose depth value ratio with the ground truth is smaller than the given threshold t.

5.2 Related Works

5.2.1 Semantic Segmentation

Semantic segmentation is a fundamental task in computer vision that involves doing pixel-wise classification, where each pixel in an image is classified into predefined categories. Unlike image classification, which outputs a single class for an input image, semantic segmentation requires the network to output dense predictions, which is a challenging task. Most related research focuses on improving the network's ability to accurately capture segment boundaries.

doi:10.6342/NTU202301439

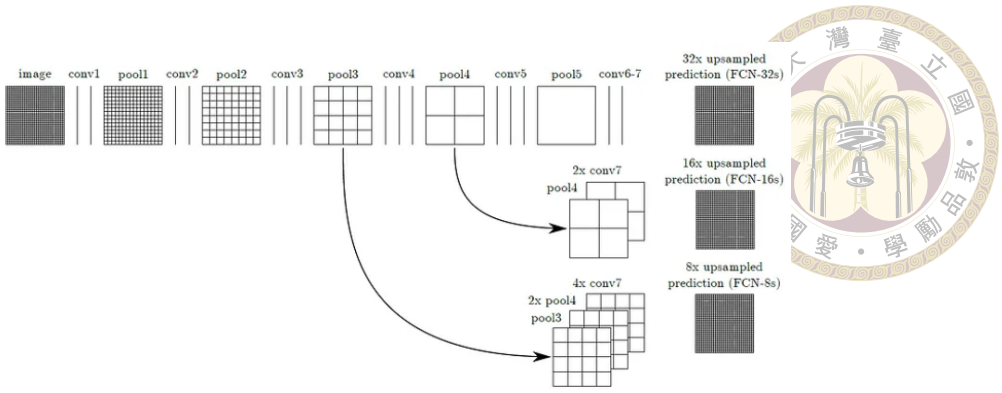


Figure 5.4: The architecture of the Fully Convolutional Network (FCN)[43]. FCN combines feature maps extracted from different hierarchical levels to capture fine-grained information and generate dense pixel-wise predictions.

Fully Convolution Network (FCN) [43] was the first deep learning network capable of producing dense predictions for image segmentation tasks. It uses image classification backbone such as AlexNet, VGG-16, and GoogleLeNet as encoders but replaces the fully connected layer with convolutional layers to build an decoder and output coarse feature maps. FCN then combines high-level and coarse feature maps with low-level and fine feature maps on the feature hierarchy using deconvolution layers to upsample the coarser feature maps to denser output. FCN achieved state-of-the-art performance on PASCAL VOC at that time and was a milestone in DCNN architecture development, demonstrating its potential for handling segmentation tasks. FCN's network architecture is depicted in Figure 5.4.

Building upon the success of FCN, SegNet[2] proposed a novel upsampling method that directly uses corresponding max-pooling downsampling indices to propagate location information to the decoder to get precious segment boundary without introducing any additional parameter at the upsampling stage. This simple yet effective architecture has made SegNet a popular choice for segmentation tasks. Another notable work that addresses the information loss caused by downsampling is U-Net[53]. U-Net combines the encoder's feature with the decoder's feature during the upsampling step to ensure fine-grained infor-

doi:10.6342/NTU202301439

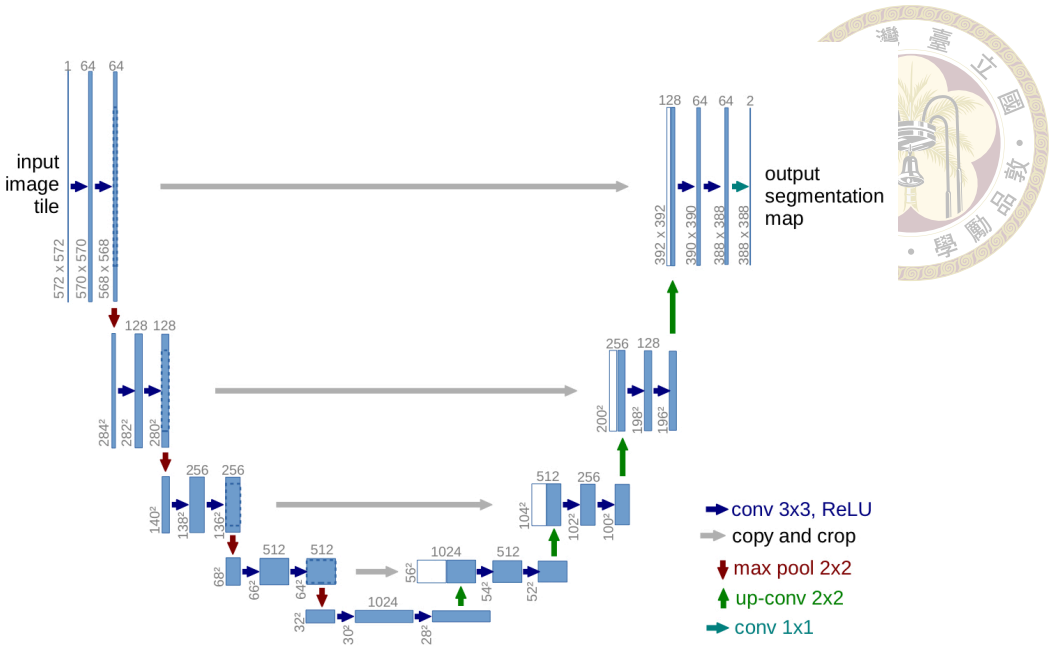


Figure 5.5: U-Net[53] architechture. The encoder's feature map is able to be reused during the upsampling stage.

mation is able to probagate to the decoder during the process. This architecture is widely used in medical image recognition and is particularly effective at detecting small tumors or anomalies. The network architecture of UNet is depicted in Figure 5.5.

Another line of work focuses on the application of atrous convolution or dilation convolution. The DeepLab[11] network, proposed by Google, uses dilation convolution to replace the last two pooling layers in the encoder to enlarge the receptive field without compromising the resolution of the image. The network also addresses the translation-invariant properties that make it hard to produce precise segmentation boundaries by introducing a fully connected Conditional Random Field (CRF) at the output score map of the deep convolutional neural network. DeepLabv2[11] further improves upon this approach by using multiple parallel dilated convolutions with different dilation rates to capture multi-scale features. After applying these techniques, DeepLabv2 achieved a mIoU of 79.7 on the PASCAL VOC 2012 test set.

Google has continued to improve its segmentation methods by combining the benefits

doi:10.6342/NTU202301439

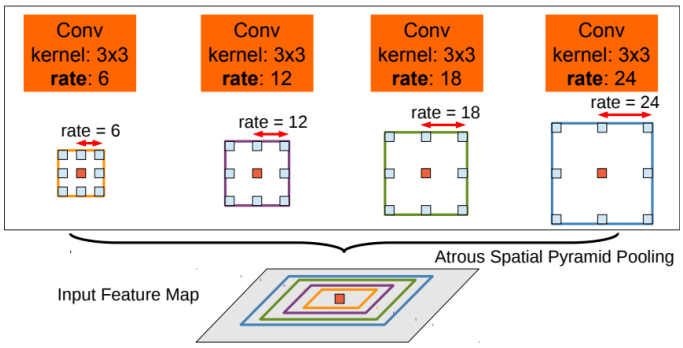


Figure 5.6: Atrous spatial pyramid pooling(ASPP) module proposed by DeepLabv3[12].

of both the encoder-decoder architecture and dilation convolution. Their latest approach, Deeplabv3+, removes the CRF refinement process and uses lateral connections from the downsampling layer during the upsampling process, which is similar to the U-Net architecture. This ensures that fine-grained information can be passed to the upsampling network, resulting in high-quality segmentation results. As a result, Deeplabv3+ achieves an impressive 89% mIoU on the PASCAL VOC 2012 test set, making it one of the most powerful and effective networks for semantic segmentation. In Figure 5.6, we depicted the atrous spatial pyramid pooling(ASPP) module used in DeepLabv3. We also shown the Deeplabv3 architecture in Figure 5.7.

We conducted experiments on semantic segmentation networks to compare the performance of various deep-learning models. We utilized the MMSegmentation[18] platform, which offers several useful pre-trained models and architectures for semantic segmentation tasks. We use models that are pre-trained on the CityScape training dataset and test on its validation dataset. The experiment was conducted on a TITAN-RTX GPU, and the input images were not resized and had a size of 512x1024 pixels. The experiment results are shown in Figure 5.8.

As seen in Figure 5.8, DeepLabv3+ achieved the best mIoU, overall accuracy, and mean accuracy, indicating that its predicted results are the most accurate. The DeepLabv3

doi:10.6342/NTU202301439

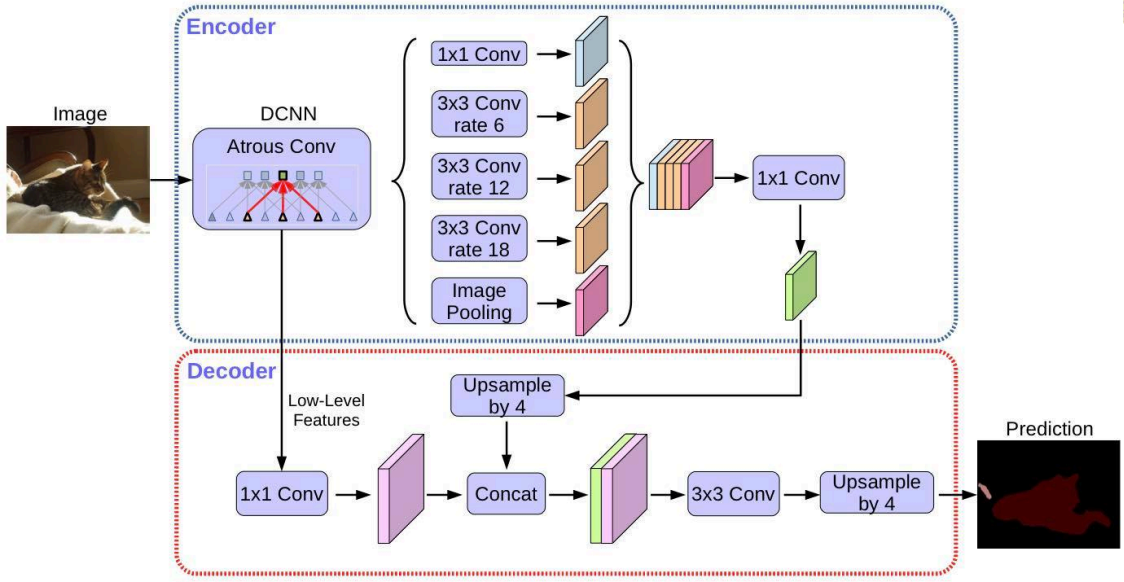


Figure 5.7: Deeplabv3[12] Architecture. Deeplabv3 uses multiple dilation convolution layers in the encoder to capture multi-scale feature; furthermore, Deeplabv3 also adopt from encoder-decoder scheme to use the encoder's information to help it recover fine-grain boundary during upsampling.

Method	BackBone	mIoU	Overall Accuracy	Mean Accuracy	Memory(GB)	FPS
FCN	ResNet101-D8	75.23	96.03	82.26	**9.20**	1.12
Unet	UNet-S5-D16	69.10	94.91	76.76	17.91	1.10
DeepLabv3	ResNet101-D8	78.45	96.29	85.91	9.60	**0.99**
DeepLabv3+	ResNet101-D8	**80.49**	**96.47**	**87.69**	11.00	1.09

Figure 5.8: Experimental results of different semantic segmentation models. mIoU represents the average Intersection over Union (IoU) across all categories. Overall Accuracy indicates the percentage of correctly classified image pixels. Memory denotes the total memory consumption during inference, and Frame per Second (FPS) indicates the speed of each model. In our experiment, DeepLabv3+ demonstrates the best performance.

doi:10.6342/NTU202301439

also performed similarly to the DeepLabv3+ but with a small performance drop. Despite being 5% behind DeepLabv3+ in mIoU, FCN occupied less memory space during inference due to its simpler architecture. On the other hand, U-Net performed the worst in our experiment and consumed the most GPU memory of 17.91 GB. We speculate that U-Net requires too many parameters for its lateral connection between the encoder and decoder, which becomes a significant issue when the input image size is large, as was the case in our experiment.

We show some of our inference example in Figure 5.9. It can be observed that DeepLab3+ generally produces the best segmentation results, while U-Net produces the worst masks in most cases.

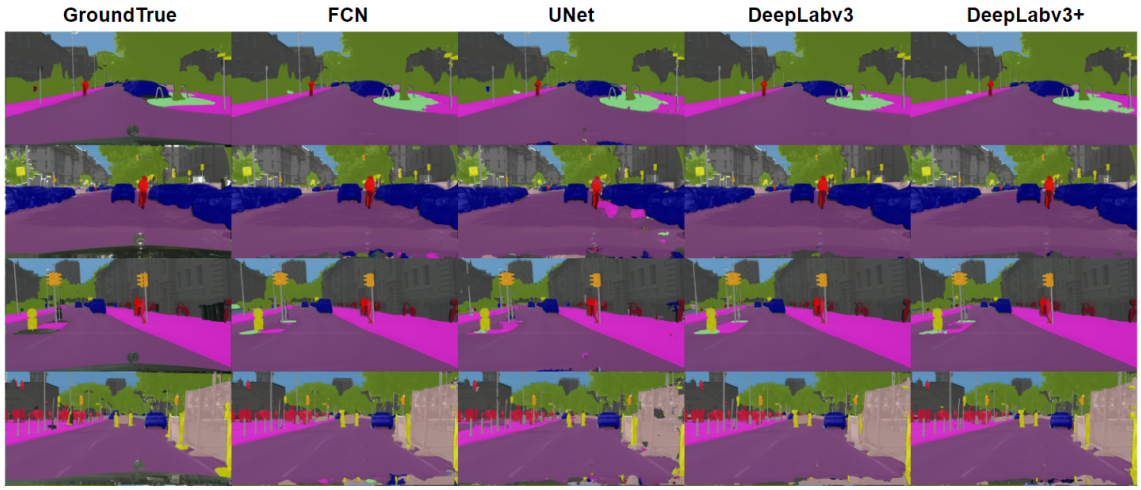


Figure 5.9: Inference example of semantic segmentation models.

5.2.2 Instance Segmentation

Instance segmentation is a more challenging task than semantic segmentation, as it requires differentiating between different instances of the same class. One way to think of instance segmentation is as object detection but uses an irregular mask to bound objects' contours instead of using bounding boxes. As a result, one intuitive approach for

doi:10.6342/NTU202301439

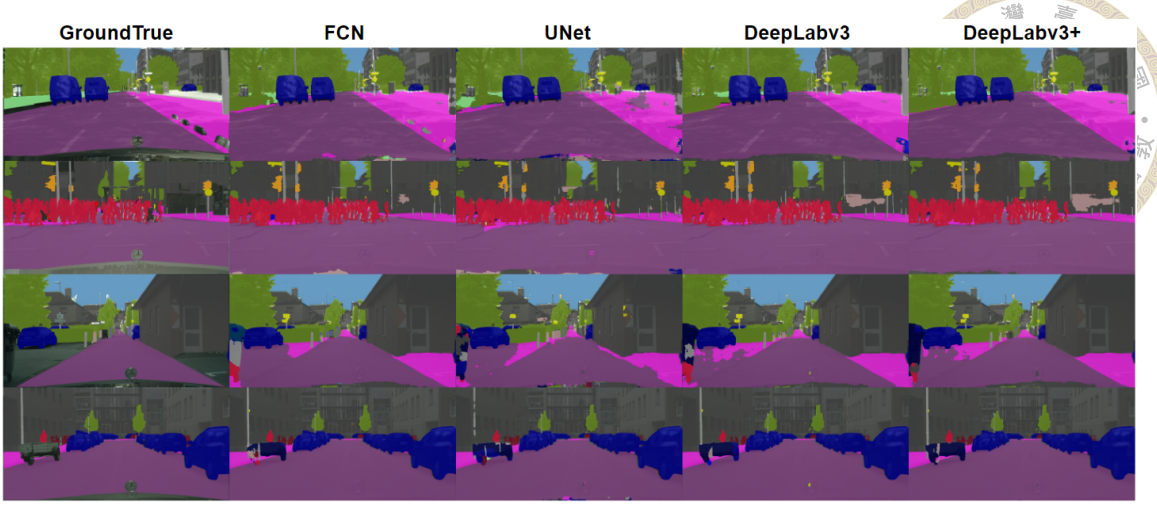


Figure 5.10: Inference example of semantic segmentation models.

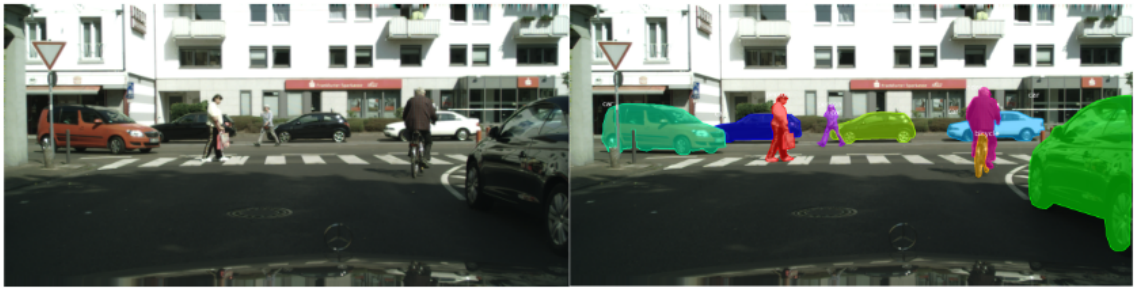


Figure 5.11: Example of Instance Segmentation task on the Cityscape dataset. The task involves detecting and segmenting all objects in the image. It is a challenging task due to the need for accurate object delineation.

instance segmentation is to detect bounding boxes and then predict image masks within those boxes, known as the detect-and-segment approach.

One of the most notable detect-and-segment approaches is Mask R-CNN [25]. Based on the Faster R-CNN architecture, Mask R-CNN adds an additional branch to predict the segmentation mask within the bounding box. It proposes a novel pooling method, called RoIAlign, to pool features inside the proposal region, ensuring that the same size of feature map is always sampled from the RoI, regardless of the RoI size and dimensions. With its simple architecture, Mask R-CNN performs exceptionally well, achieving 35.7% mAP on the COCO test set. The architecture of Mask-RCNN is depicted in Figure 5.12

PANet[39] is an extension of Mask-RCNN that seeks to enhance feature representa-

doi:10.6342/NTU202301439

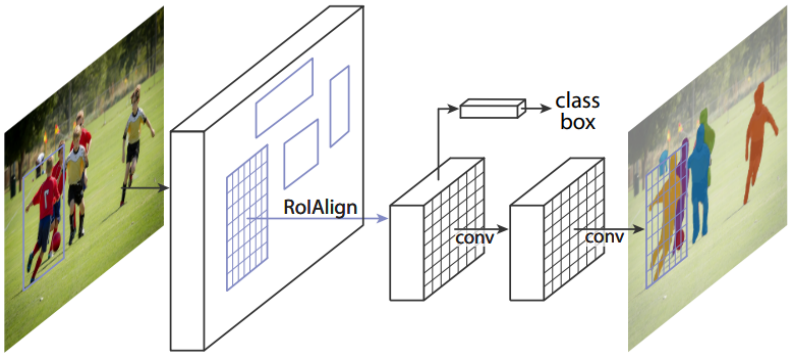


Figure 5.12: Mask-RCNN[25] network architecture. It adopts a detect-and-segment scheme where an object detector is used to crop the region of interest from the image, and the features within that RoI are then used to predict the object segmentation.

tion by introducing bottom-up path augmentation and adaptively pooling the features from the feature hierarchy. With a stronger multi-scale feature, PANet is able to improve the performance on COCO test set to 40.0% mAP. Masklab[10], which is also based on Mask-RCNN, uses a different approach by replacing the direct mask prediction scheme with a semantic segmentation branch and a directional prediction branch. The semantic segmentation branch is responsible for classifying foreground and background pixels, while the directional prediction branch predicts the pixel offset to its instance center. This approach gains a slight boost for its quality mask results and achieves 38.1% mAP in COCO testset.

While most methods focus on improving the quality of instance segmentation, YOLACT[7] has mainly focused on speeding up the segmentation process. To achieve this goal, YOLACT predicts a fixed number of full-image-size semantic segmentation logits, called prototypes, and linearly combines prototypes to get different instances and use the predicted bounding box to crop mask from it, where the weight of the linear combination is learnable during training. This design significantly reduces the number of dense predictions and allows YOLACT to run in real-time with a frame rate of 33.5 FPS, compared to Mask-RCNN's frame rate of 8.6 FPS. However, the speed-up comes at the cost of lower performance, as its performance on the COCO test set is only 29.8 mAP. The YOLACT architecture is

doi:10.6342/NTU202301439

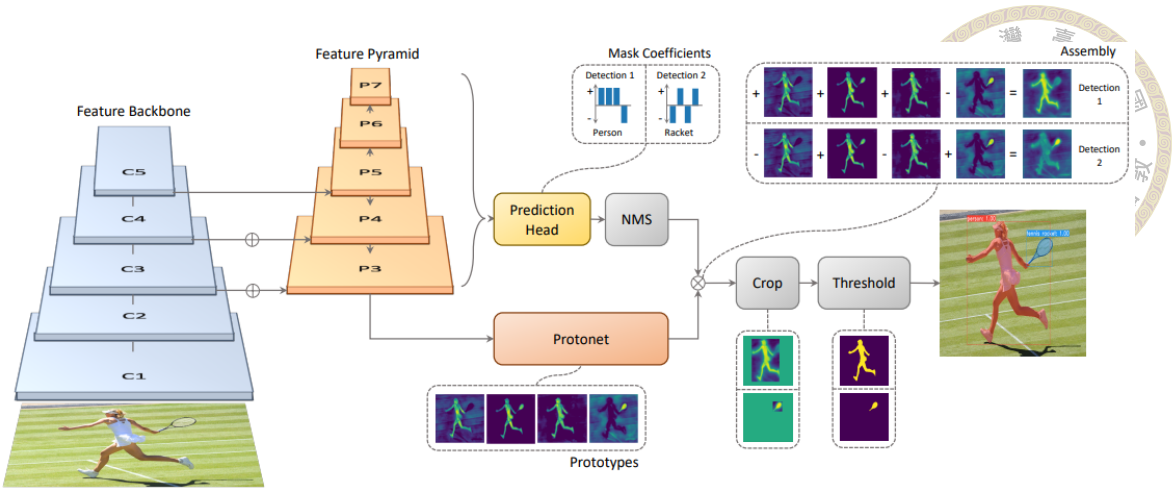


Figure 5.13: YOLACT[7] network architecture. It predicts multiple prototypes to represent instances. Each instance is a linear combination of the predicted prototypes, and the weights are determined by another prediction head. YOLACT is known for its high speed but may have lower mask quality compared to other methods.

shown in Figure 5.13.

YOLACT++[75] aimed to improve the low AP of YOLACT by introducing deformable convolution layers in its backbone and using an additional mask rescoring network to re-rank the predicted masks. With these improvements, YOLACT++ was able to maintain roughly the same inference speed as YOLACT while improving performance to 34.6 AP on the COCO dataset. The deformable convolution layer allows the network to adapt to object deformations and leads to better feature representation, while the mask rescoring network can further refine the mask predictions. YOLACT++ has achieved impressive results in real-time instance segmentation, making it suitable for applications where detection speed is essential.

Recently, a new approach to instance segmentation has emerged. SOLO[62] and its improved version, SOLOv2[63], unlike the traditional detection-then-segmentation approach, directly predicts the instance mask for each instance location. In SOLO, the input image is divided into an $S \times S$ grid, and each grid is responsible for predicting the mask and logits for the instance whose center is located in the corresponding grid cell. SOLO

doi:10.6342/NTU202301439

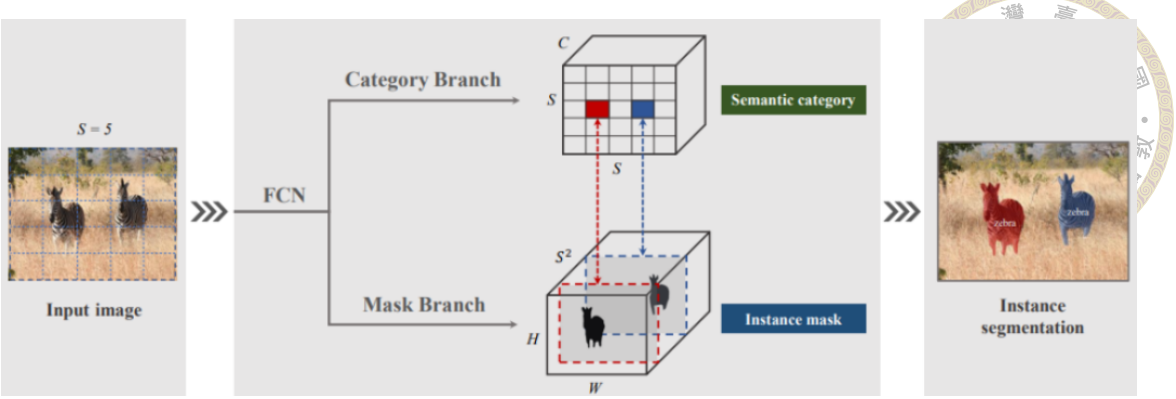


Figure 5.14: SOLO[62] network architecture. SOLO treats instances at different locations as different categories, enabling it to generate high-quality masks. However, this approach leads to a larger network size and slower training compared to other methods.

treats instances in different locations as if they were different classes and uses different channels to predict the mask. Additionally, to address the translation-invariant properties of convolutional layers, SOLO introduces CoordConv [37] to allow the network to learn with location information. With a fully dedicated mask prediction approach, SOLO can produce high-quality masks with 41.7% AP in COCO test set. However, SOLOv2 can be slow to train and infer since it needs to output S^2 full-image-size masks for each location. SOLO's network architecture is shown in Figure 5.14

Overall, the field of instance segmentation has seen significant advancements in recent years. While methods such as Mask-RCNN and its variants have focused on improving the quality of predicted masks, YOLACT has instead aimed to increase inference speed by predicting a fixed number of prototypes. Meanwhile, SOLO and SOLOv2 have taken a unique approach by directly predicting instance masks for each location in a grid, achieving state-of-the-art performance on the COCO dataset. These developments demonstrate the continued progress in the field of instance segmentation and provide promising directions for future research.

We conducted experiments using Mask-RCNN, SOLOv2, and YOLACT++ and present

doi:10.6342/NTU202301439

Method	AP	AP(IoU=0.5)
Mask-RCNN	**36.50**	**62.20**
YOLACT	16.02	32.93
SOLOv2	18.70	34.00

Figure 5.15: Experiment result of the instance segmentation network.

the results in Figure 5.15. For training, we used the Cityscape training set and trained SOLOv2 for 30 epochs and YOLACT++ for 275 epochs. As for Mask-RCNN, we utilized the pre-trained model available on the Detectron2[65] platform. The evaluation of the models was performed on the Cityscape validation set. In our findings, Mask-RCNN achieved the best performance, which can be attributed to the utilization of a pre-trained model instead of training from scratch. This highlights the significance of using pre-trained models in segmentation tasks.

Our qualitative results are presented in Figure 5.16. As can be seen, Mask-RCNN produces the most accurate masks in these examples, demonstrating the effectiveness and efficiency of this simple yet powerful network. SOLOv2 and YOLACT++ typically produce worse mask quality compared to Mask-RCNN. However, we also noticed that all three methods were able to accurately capture car instances in this experiment. The difference is apparent when dealing with more deformable objects, such as cyclists or pedestrians.

5.2.3 Panoptic Segmentation

Panoptic segmentation, also known as scene parsing, unifies the tasks of semantic and instance segmentation. As finding instances in an image is often more challenging than pixel-wise classification, many approaches extend well-developed instance segmentation networks by adding an additional segmentation branch to achieve scene parsing.

doi:10.6342/NTU202301439

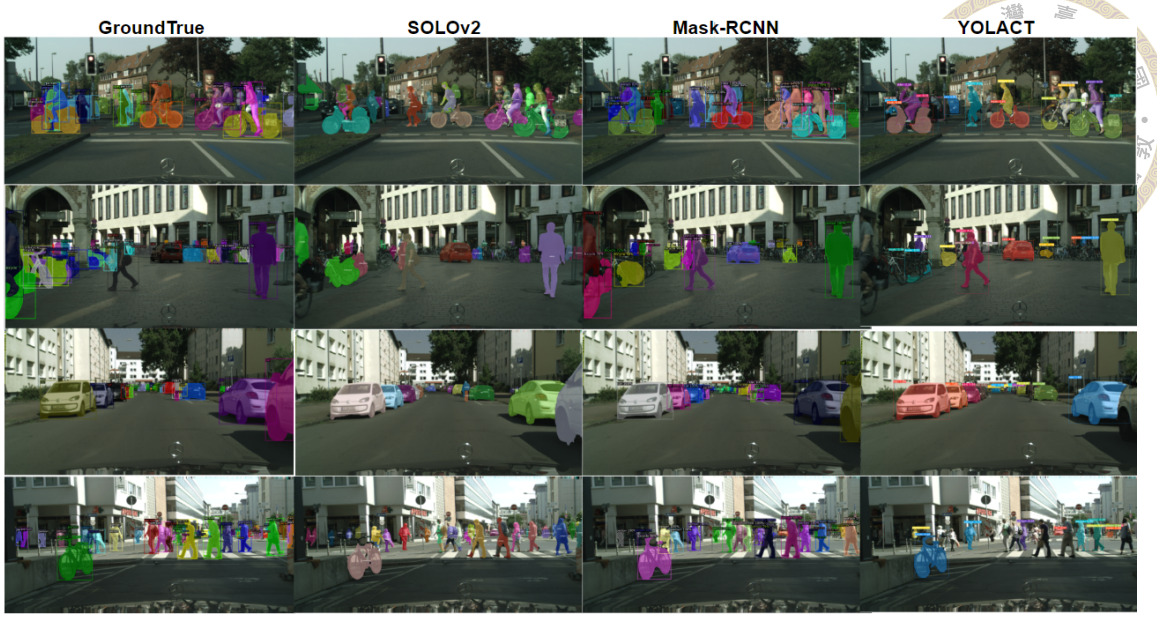


Figure 5.16: Qualitative result of instance segmentation algorithm.

For example, Panoptic-FPN [28] grafts a new segmentation branch onto Mask-RCNN by using a Feature Pyramid Network (FPN) after the backbone to provide enough pixel-wise information for segmentation. After obtaining instance and semantic segmentation results, it fuses them by eliminating overlapping predictions. On the Cityscapes validation set, Panoptic-FPN achieves 58.1% PQ.

DeeperLab[69] adopts the encoder-decoder architecture of DeepLab[13] and adds an additional prediction head after the feature map to predict semantic segmentation and four other keypoint-based detections to find different instances. The UPSNet[67] architecture is similar to Panoptic-FPN, which adds an FPN neck to the backbone and designs separate semantic and instance heads for prediction. UPSNet also adds a panoptic head at the end of the pipeline to combine the results. UPSNet achieves 61.8% PQ on the Cityscape validation dataset.

Panoptic-DeepLab[17] extends the DeepLabv3+ architecture to enable panoptic segmentation. They use the dual-ASPP and dual-decoder to output three different outputs: semantic segmentation, object center prediction, and center offset prediction. The seman-

doi:10.6342/NTU202301439

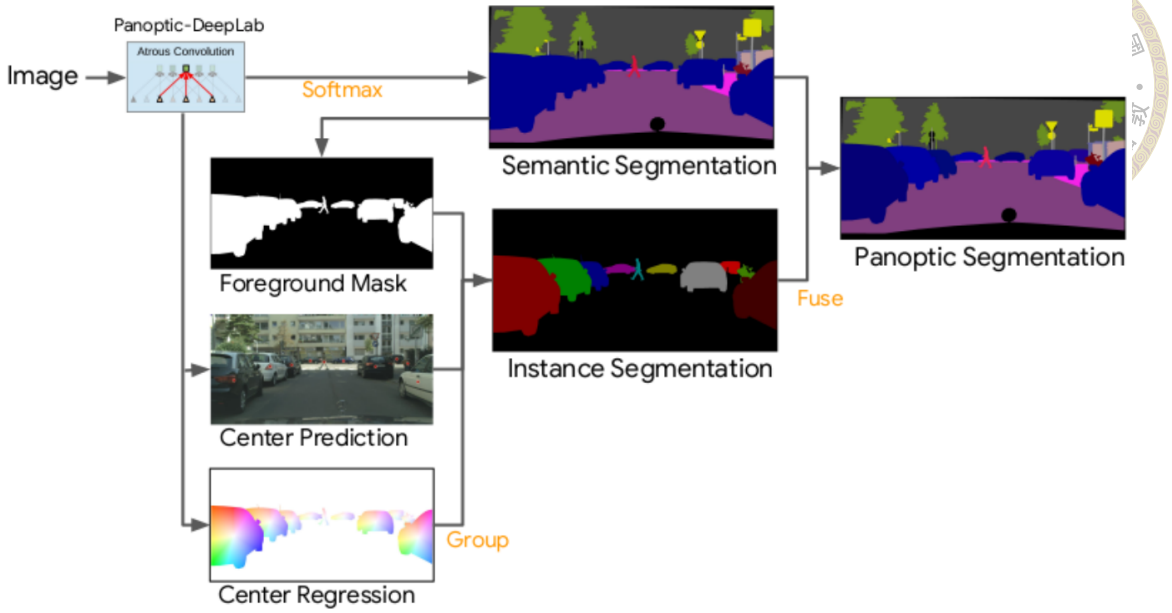


Figure 5.17: Panoptic-DeepLab[17] architecture

tic segmentation branch produces the same output as DeepLabv3+. The center prediction and offset regression result combine to obtain class-agnostic instances. They use a majority-vote algorithm to determine the class of each instance by fusing it with semantic segmentation prediction. Panoptic-DeepLab employs a bottom-up approach to handle the instance segmentation branch in this task and exploits the advantage of the DeepLab architecture. As a result, it achieves 64.1% PQ on the Cityscape validation set.

We present our experimental results with panoptic segmentation using the Detectron2[65] platform. Similar to MMSegmentation, Detectron2 offers many pre-trained models for various image tasks. Therefore, we conduct an experiment using its implemented Panoptic-DeepLab and use the provided pre-train weight, and ran inference on a TITAN RTX GPU.

In Figure 5.18, we present the Panoptic Quality (PQ) of the Panoptic-Deeplab model tested on the Cityscape validation dataset. We observe that the PQ of the ”thing” category, which represents instances, is lower than that of the ”stuff” category, which rep-

doi:10.6342/NTU202301439

Methods	PQ	SQ	RQ
Panoptic-FPN	55.4	77.9	69.3
UPSNet	60.1	80.3	**73.5**
Panoptic-DepthLab	**60.3**	**81.5**	72.9

Figure 5.18: Experimental results of panoptic segmentation models. Panoptic-DeepLab achieved the highest performance among the other models.

resents background categories. This indicates that capturing instances is indeed a more challenging task than pixel-wise classification. In Figure 5.19 and Figure 5.20, we show the inference result of Panoptic-DeepLab on Cityscape validation set.

5.3 Proposed Method - Panoptic-DepthLab

In this section, we introduce Panoptic-DepthLab, our proposed method that extends the architecture of Panoptic-DeepLab to incorporate a depth estimation decoder branch. By leveraging the strong performance of Panoptic-DeepLab in panoptic segmentation, we enhance the network's capabilities to also generate accurate depth estimation results. During training, we optimize the loss function of both the panoptic segmentation branch and the depth estimation branch, allowing the network to jointly learn these tasks. Notably, all decoder branches share the same encoder features, enabling the segmentation and depth estimation tasks to leverage the same image features. To combine the predicted depth map with the panoptic segmentation, we average the pixel depths within each instance region. The shared encoder features facilitate efficient end-to-end training. Specifically, our newly added depth estimation branch follows a similar design to the semantic segmentation branch. We utilize an Atrous Spatial Pyramid Pooling (ASPP) module to capture global context for understanding scene structure. The detailed architecture is illustrated in Figure 5.21.

doi:10.6342/NTU202301439

Figure 5.19: Inference example of Panoptic-DeepLab. The first column shows the input image, and the second column displays the predicted panoptic segmentation results. Each instance is segmented and marked with a different color, while the background pixels are assigned different semantic labels.

doi:10.6342/NTU202301439

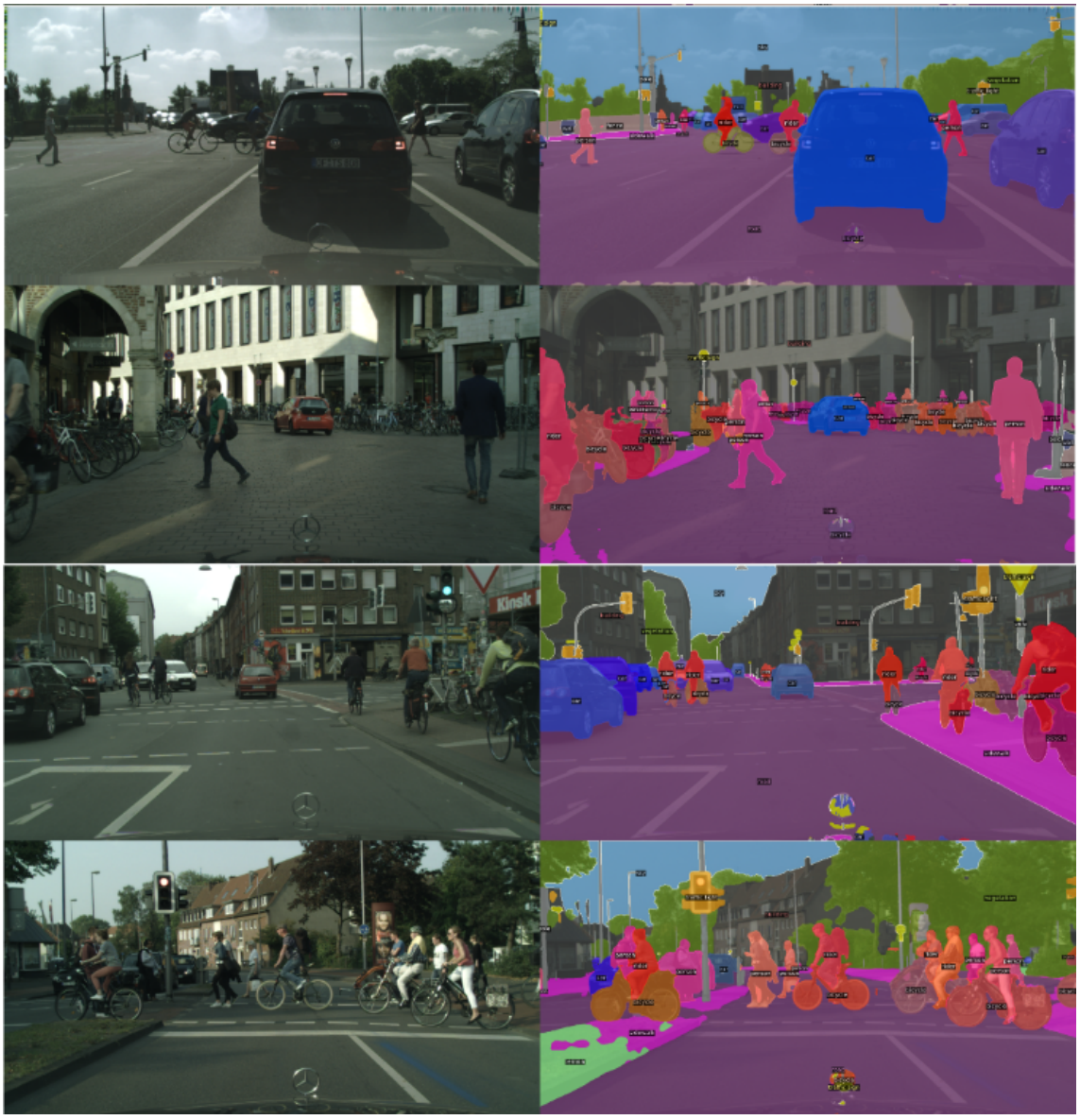


Figure 5.20: Inference example of Panoptic-DeepLab. The first column shows the input image, and the second column displays the predicted panoptic segmentation results. Each instance is segmented and marked with a different color, while the background pixels are assigned different semantic labels.

doi:10.6342/NTU202301439

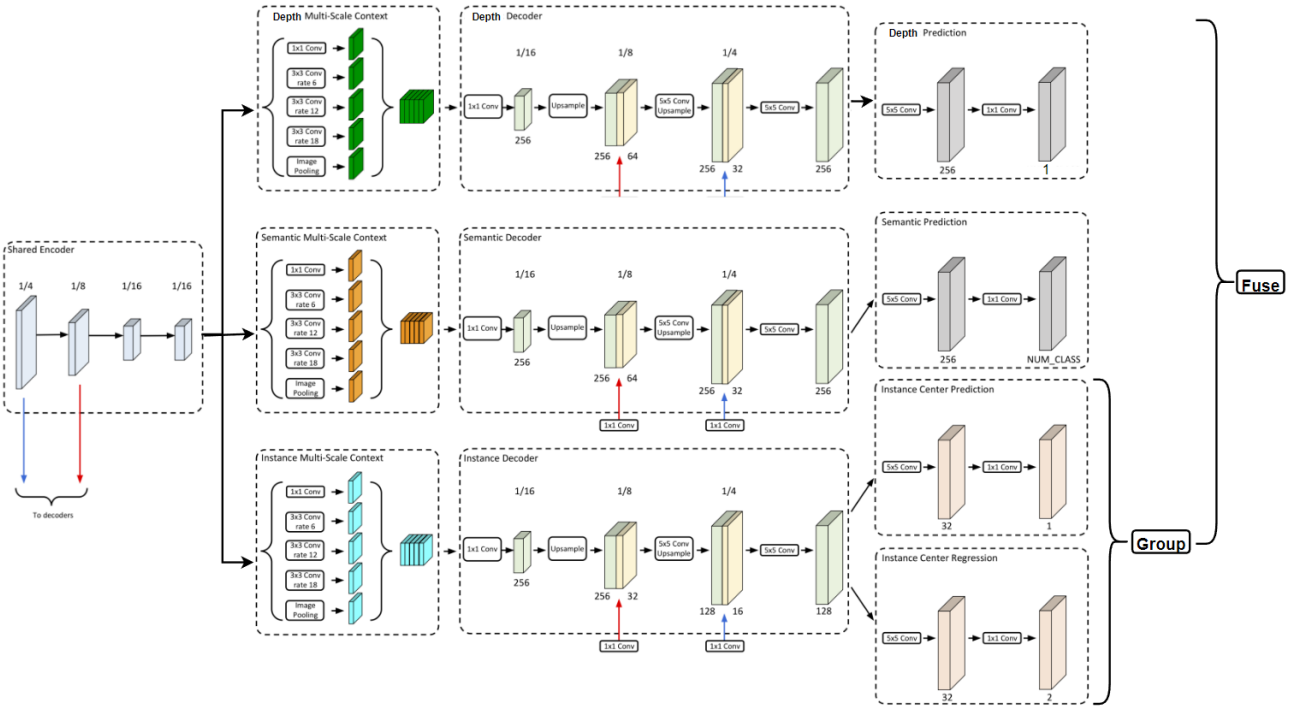


Figure 5.21: Architecture of Panoptic-DepthLab. Based on Panoptic-DeepLab, we introduce an additional decoder branch specifically designed for depth estimation. The network consists of three branches: semantic segmentation, instance segmentation, and depth estimation, all utilizing the shared feature map learned by the encoder. This unified network allows for end-to-end training, seamlessly integrating the segmentation and depth estimation tasks. During inference, a post-processing step combines the segmented regions with depth information.

doi:10.6342/NTU202301439

5.4 Experiment Result

In this section, we present the experimental results of our Panoptic-DepthLab network, which is built upon the Panoptic-DeepLab implementation in the Detectron2 platform. We conduct training and testing on the Cityscape dataset, which includes 2975 images in the training set and 500 images in the validation set. The dataset consists of eight background categories and eleven foreground categories for the panoptic segmentation challenge.

During training, we initialize our network with the pre-trained weights provided by Panoptic-DeepLab, which were trained on the Cityscape training dataset for 90K iterations. Subsequently, we fine-tune the entire network for an additional 10K iterations on the same training set after incorporating our depth estimation branch. The training process is conducted on 2 TITAN RTX GPUs, with a batch size of 14. We employ the Adam optimizer with a learning rate set to 0.001. Since Cityscape does not provide ground truth depth estimation, we generate depth maps by converting the disparity map between the left and right camera images during the training process.

5.4.1 Quantitative Result

To determine the optimal loss function for our depth estimation branch, we compared two different loss functions during training. The first one is the loss function proposed by DORN[22], which discretizes depth values into intervals based on uncertainty and formulates the regression problem as multiple binary classification subtasks. The second loss function is the Smooth L1 loss.

doi:10.6342/NTU202301439

Table 5.1: Evaluation of depth estimation results produced by Panoptic-DepthLab. Both loss function settings were trained for 10K iterations and tested on the Cityscape validation dataset. Evaluation metrics include relative squared error (sqErrorREL), relative absolute error (absErrorRel), inverse root mean square error (IRMSE), and scale invariant logarithmic error (SILog), which assess the difference between the predicted depth map and the ground truth depth map. Smaller values indicate better performance.

Loss	sqErr	absErr	IRMSE	SILog
DORN[22]	0.72	0.69	44.64	22.31
L1	**0.63**	**0.60**	**34.57**	**18.58**

Table 5.2: Evaluation of depth estimation results produced by Panoptic-DepthLab. Both loss function settings were trained for 10K iterations and tested on the Cityscape validation dataset. The evaluation metric is accuracy with threshold, representing the percentage of depth map pixels whose ratio with the ground truth pixels is smaller than the assigned threshold value. Threshold values of 1.25, 1.25^2, and 1.25^3 are used. Higher values indicate better performance.

Loss	$\delta_1 < 1.25$	$\delta_2 < 1.25^2$	$\delta_3 < 1.25^3$
DORN	0.27	0.49	0.68
L1	**0.30**	**0.61**	**0.82**

The results of our experiments are presented in Table 5.1 and Table 5.2. Surprisingly, we found that the Smooth L1 loss outperformed DORN's loss after training for 10K iterations. This may be attributed to the complexity of the DORN method, which involves a more complex network structure and a larger number of additional parameters. In contrast, the Smoothed L1 loss proved to be a more effective and straightforward method for our specific task.

5.4.2 Qualitative Result

To assess the performance of Panoptic-DepthLab, we present inference example on the Cityscape validation set, depicted in Figure 5.22 and Figure 5.23. In these examples, each predicted instance is visually distinguished by its assigned color based on their depth value. Objects in close proximity appear in vibrant red, while those farther away are represented in cooler shades of blue. Additionally, the background pixels, such as road,

doi:10.6342/NTU202301439

sky, and vegetation, are effectively segmented and labeled in the resulting output. Each instance is also annotated with its respective category and depth information, providing a detail understanding of the scene.

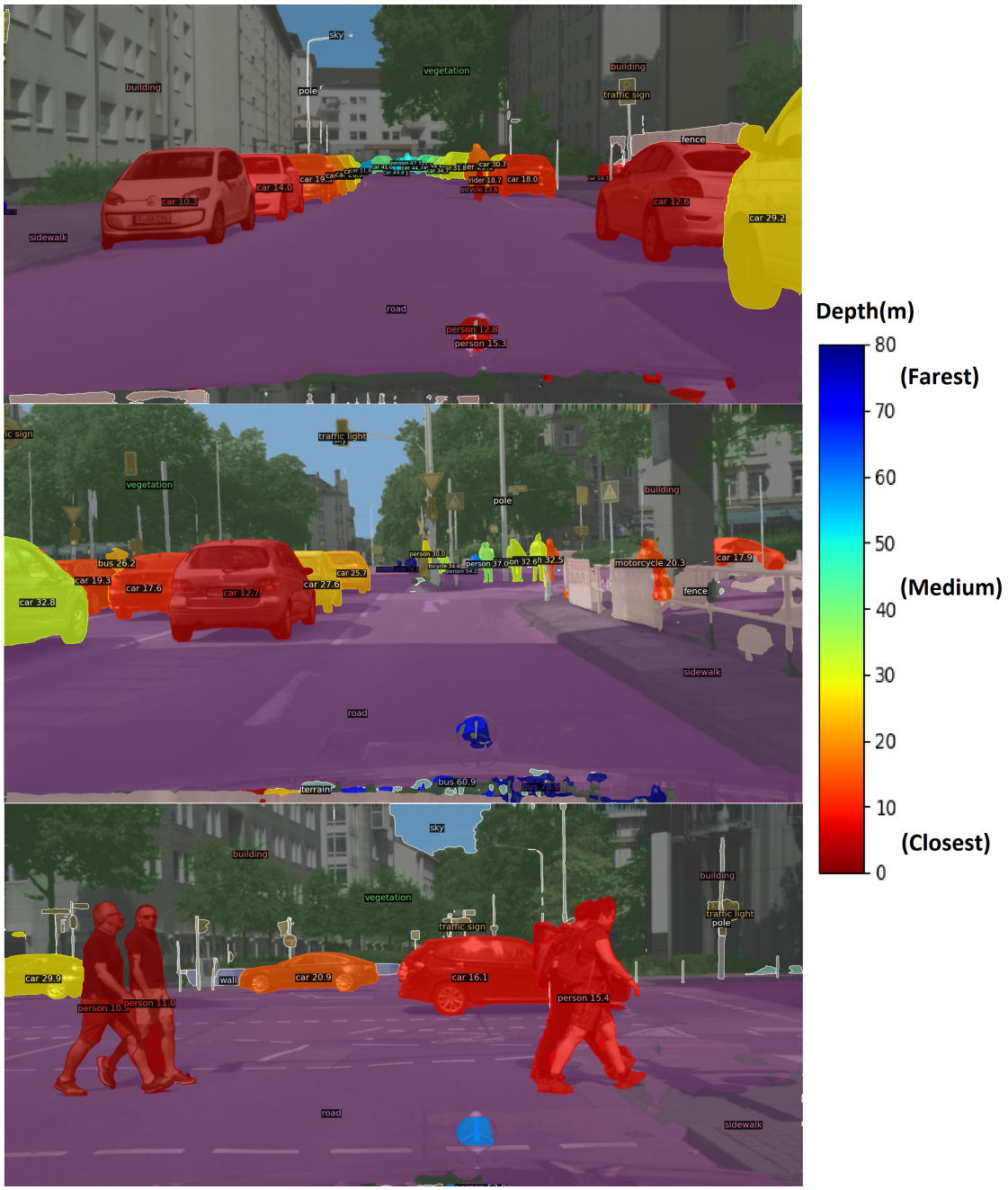


Figure 5.22: Inference example of Panoptic-DepthLab on the Cityscape validation set. Each instance is labeled with a category and a corresponding depth value. The color assigned to foreground objects is determined based on its depth value. The objects closest to the camera are colored in red, while the farthest objects are colored in blue.

doi:10.6342/NTU202301439

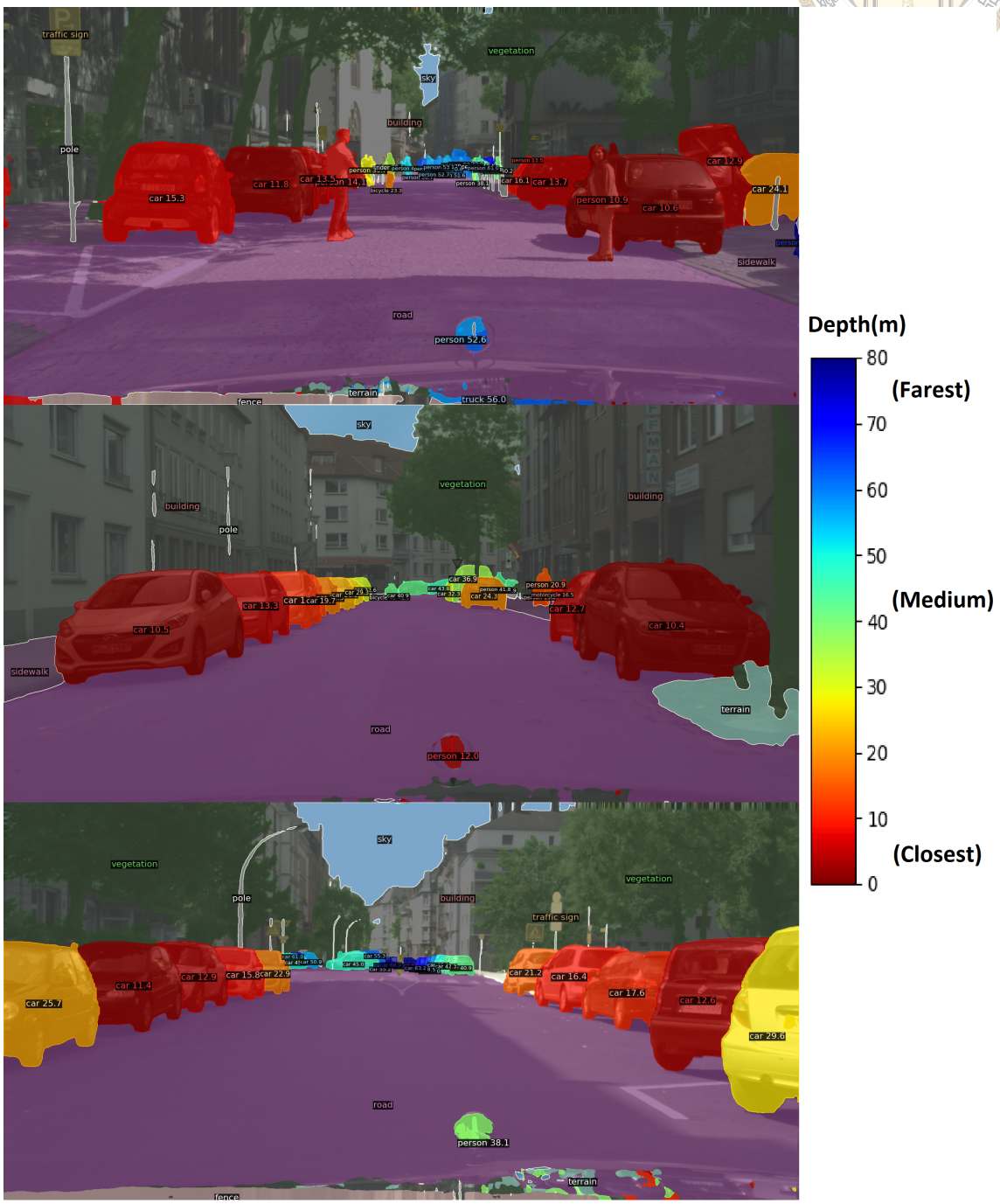


Figure 5.23: Inference example of Panoptic-DepthLab on the Cityscape validation set. Each instance is labeled with a category and a corresponding depth value. The color assigned to foreground objects is determined based on its depth value. The objects closest to the camera are colored in red, while the farthest objects are colored in blue.

doi:10.6342/NTU202301439

5.5 Summary

In this chapter, we examined three types of image segmentation tasks: semantic, instance, and panoptic segmentation. Semantic segmentation involves labeling each pixel in an image with a specific class label, while instance segmentation focuses on identifying individual objects in an image and assigning each pixel to a specific object instance. Panoptic segmentation aims to unify semantic and instance segmentation into a single task. We explored several deep-learning approaches for each task, including Mask-RCNN, YOLACT, SOLO, and Panoptic-DeepLab.

Moreover, we propose a novel approach that combines depth estimation and panoptic segmentation to achieve fine-grained image reconstruction. We introduce the Panoptic-DeepLab architecture, which extends the Panoptic-DeepLab model by incorporating an additional depth estimation branch. This allows us to obtain depth values for all instances in the image. The results are visualized using a color map, and we demonstrate significant improvements in the accuracy and quality of the reconstructed images.

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

Chapter 6 Safety Metric in Driving Scenario

In detection tasks, the commonly used evaluation metric is average precision (AP). The AP metric assesses the performance of a detector by classifying detected positive and negative boxes as true or false, based on their intersection over union (IoU) with the ground truth boxes. By calculating the number of true positives (TP), false positives (FP), and false negatives (FN), precision and recall can be determined. Average Precision is then obtained by computing the area under the Precision-Recall curve, which represents the detector's performance across various confidence threshold values, ranging from zero to one. This makes AP a robust metric that captures the detector's overall performance, regardless of the specific confidence threshold used. As we have discussed the calculation of the AP metric in Chapter 2, we will not delve into the details here.

To assess the performance of 3D object detectors in driving scenarios, the commonly used metric is AP3D, which extends the AP metric by considering the IoU of 3D bounding boxes instead of 2D bounding boxes. However, we have identified a limitation with the AP3D metric—it treats all on-road objects equally, regardless of their distance from the ego-vehicle. We argue that the metric should assign less weight to farther objects compared to closer objects, as the failure to detect a close object poses a higher risk of collision.

doi:10.6342/NTU202301439

To address this concern, we propose a safety metric for 3D object detection that incorporates factors such as object distance, orientation, and other safety considerations into the evaluation. By doing so, we create a more comprehensive metric that better captures the performance of 3D object detectors in driving scenarios. The concept of safety weighting is illustrated in Figure 6.1 and Figure 6.2.

Figure 6.1: Safety weighting of the default Average Precision metric. The number on each red box represents the weight assigned to each object during the AP evaluation. The metric treats all objects equally, regardless of their proximity to the ego-vehicle. However, we argue that this approach is inappropriate for autonomous driving scenarios, as it fails to account for the varying risks associated with different object distances.

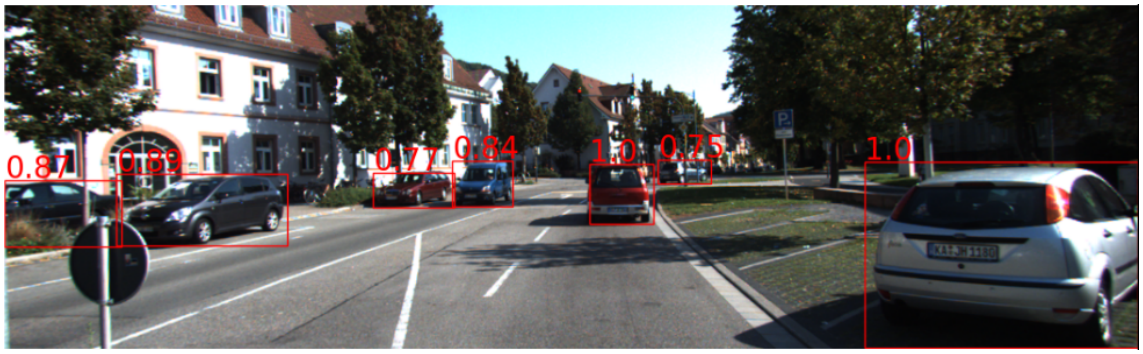
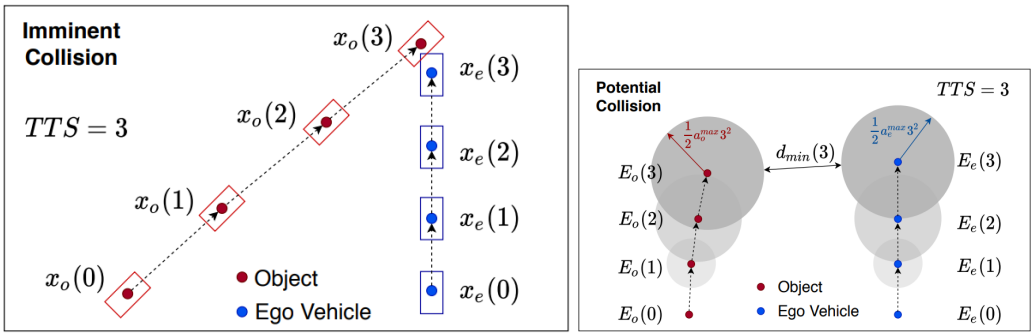


Figure 6.2: Safety weighting of our proposed safety-aware Average Precision metric. The number on each red box represents the weight assigned to each object during the AP evaluation. Our metric places higher weight on closer objects and lower weight on farther objects. This is because accurately detecting close objects is more critical in terms of safety, while missing detections of distant objects has less impact.

doi:10.6342/NTU202301439

6.1 Related Work

To establish a safety metric, previous studies have relied on kinematic information of objects such as 3D location, orientation, velocity, and maximum acceleration. Using these properties, researchers have attempted to predict potential collision points between obstacles and the ego-vehicle, which are then used to evaluate the safety weight of each object. For instance, the RRR approach ranks objects based on their likelihood of colliding with the ego vehicle, categorizing them as imminent risk, potential risk, or no risk. Imminent risk objects are those that will collide with the ego vehicle within three seconds without additional action, while potential risk objects are those that will collide if both vehicles use the worst-possible acceleration strategy. However, this criterion is quite stringent, resulting in highly skewed rankings. In the Waymo image sequence dataset, only 3.4% of objects were classified as potential risk and none were classified as imminent risk. This indicates that the criterion can only identify a small number of critical objects in the dataset and cannot provide a comprehensive safety evaluation. The calculation of RRR[3] is shown in Figure 6.3.



(a) Imminent Risk (b) Potiential Risk

Figure 6.3: Safty metric proposed in RRR[3]

For other work, Mirja Wolf et al.[64] defines safety as the distance between a pedestrian and a danger zone in front of the ego vehicle and the Time-To-Collide (TTC) with

doi:10.6342/NTU202301439

the ego vehicle based on the provided pedestrian velocity and heading. The closer the pedestrian is to the danger zone, the more weight the object is given. In contrast to RRR's approach, this work incorporates its calculated weight into the AP metric by multiplying the safety weight to each true positive (TP), false positive (FP), and false negative (FN). This results in a safety-aware precision rate, recall rate, and F1 score. The newly weighted metric is able to reflect how well the detector can capture on-road pedestrians that are critical to driving safety.

Georg Volk et al.[61] proposes a comprehensive safety metric for evaluating the performance of an object-tracking model. It considers the quality of perception, relevance of object collision, and the time used by the detector. To evaluate quality of perception, it adopts the CLEAR metric [5], which is commonly used in object tracking evaluation. To derive relevance, it fences a relevant zone around the ego vehicle and calculates a collision score for each object inside the zone. In terms of detection time, it directly scores each object in the frame based on how long the detector needed to finish the prediction for that frame. The longer the time required, the lower the safety score. This work then combines all three safety criteria into a single safety score ranging from zero to one. The comprehensive safety metric provides a more thorough evaluation of the detector's performance by taking into account both the quality of perception and the relevance of object collision, as well as the time used for detection.

Safety-aware Metric[9], which is the most related work to ours, calculates the criticality of every object based on its velocity, orientation, and distance to the ego vehicle. The criticality factor considers three aspects of the object: its distance to the ego-vehicle, its distance to a potential collision point with the ego-vehicle, and the time it needs to reach the collision point. These calculated factors are combined to form a safety score

doi:10.6342/NTU202301439

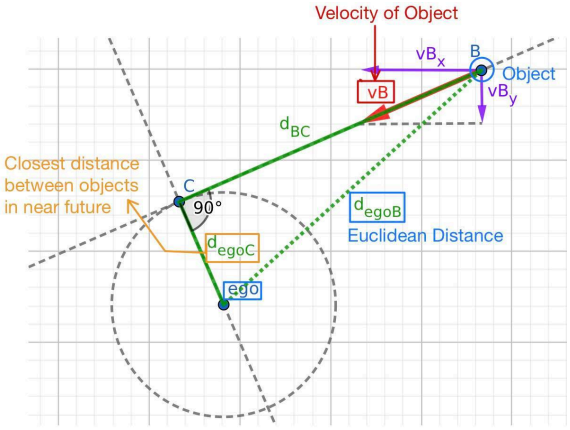


Figure 6.4: The Safety-aware Metric[9] introduces a criticality factor based on the geometric relationship between the ego-vehicle and the object. It takes into account the Euclidean distance between the ego-vehicle and the obstacle, the estimated time to collision, and the distance to the collision point. These three factors are integrated into a value ranging from 0 to 1, which is then used to assign a weight to each object for calculating the Average Precision (AP). This safety-aware approach ensures that objects closer to the ego-vehicle are given more importance in the evaluation process.

ranging from zero to one. Similar to [64], Safety-aware Metric uses the safety score as a weight during the calculation of average precision, resulting in an improved version of the AP metric that takes the criticality of objects into account. This work provides valuable insights into the importance of considering criticality in object detection for driving scenarios. It proposed criticality is depicted in Figure 6.4.

Overall, there is a limited amount of related work on safety metrics for object detection in driving scenarios, especially considering the rapid development of object detector. Furthermore, none of the works mentioned above has addressed safety metrics on a single-frame image where the object velocity is unknown. Most open datasets for object detection do not provide object velocity, making these approaches impractical in many cases. Therefore, our work aims to develop a universal safety metric for 3D object detection that does not rely on knowledge of object velocity. By doing so, we hope to provide a more practical and widely applicable approach to evaluating the performance of object detection models for driving scenarios.

doi:10.6342/NTU202301439

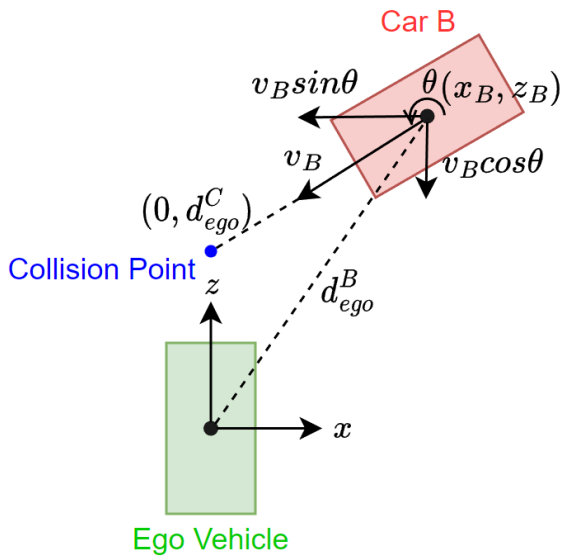


Figure 6.5: Our proposed safety weight calculation. v_B represents the velocity of Car B, θ represents the yaw angle with respect to the x-axis, and (x_B, z_B) represents the location of Car B relative to the ego vehicle. Lastly, d^C_{ego} represents the location of the collision point. We wish to find the d^C_{ego} and d^B_{ego}.

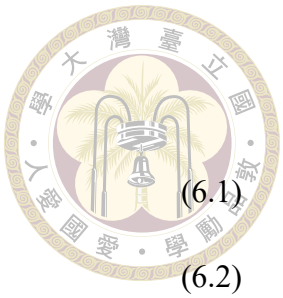
6.2 Proposed Method - Safety-aware Average Precision

Our proposed safety metric builds upon the work of [9], but we extend it to single-frame images where the velocity of the objects is not available. To achieve this, we remove the third criticality criterion, which considers the collision time between the ego-vehicle and the obstacle, as it cannot be calculated without knowing the velocity. However, we still can identify the potential collision point by leveraging basic geometric relationships between the object and the ego-vehicle. Therefore, the two remaining criticalities are the distance between the object and the ego-vehicle and the distance to the potential collision point. Our proposed method is depicted in Figure 6.5.

Based on the geometry relationship we presented in Figure 6.5, we can derive the

doi:10.6342/NTU202301439

following equations:



$$\text{x-direction: } v_B cos\theta \Delta t + x_B = 0 \tag{6.1}$$

$$\text{z-direction: } v_B sin\theta \Delta t + z_B = d^C_{ego} \tag{6.2}$$

Here, v_B represents the velocity of Car B, θ represents the yaw angle with respect to the x-axis, and (x_B, z_B) represents the location of Car B relative to the ego vehicle. Lastly, d^C_{ego} represents the location of the collision point. Although Δt, v_B, and d^C_{ego} are unknown variables in this system of equations, we can still solve for d^C_{ego} by dividing the first equation by cos θ and substituting into the second equation and the result is shown in the following:

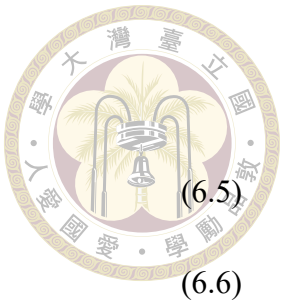
$$d^C_{ego} = -x_B tan\theta + z_B \tag{6.3}$$

Note that v_B and Δt have been eliminated in the equation; hence, there's no need to obtain object velocity. The other criticality is the distance between ego-vehicle to Car B, which are be formalized as follows:

$$d^B_{ego} = \sqrt{x_B^2 + z_B^2} \tag{6.4}$$

where d^B_{ego} represents the distance between ego-vehicle and Car B. (x_B, z_B) are Car B center location. We use the following equation to calculate criticality:

doi:10.6342/NTU202301439



$$\kappa_b = max(0, -(\frac{d^B_{ego}}{D_{max}})^2 + 1) \tag{6.5}$$

$$\kappa_c = max(0, -(\frac{d^C_{ego}}{D_{max}})^2 + 1) \tag{6.6}$$

$$\kappa = 1 - (1 - \kappa_b)(1 - \kappa_c) \tag{6.7}$$

where κ_b is the criticality factor that indicates how close is the ego-vehicle to the object, and κ_c represents criticality factors related to the distance to the collision point. And κ is the final safety weight we assign to this object. We use the safety weight to recalibrate our calculation of precision and recall to emphasize objects with higher criticality.

6.3 Experiment Result

In this section, we show our proposed AP_{crit} evaluation result in the KITTI3D validation dataset. We use it to evaluate the performance of the 3D object detector we mentioned in Chapter 4.

Table 6.1: Experiment result of AP and AP_{crit}. We try to evalute some 3D object detector and show their difference here. In our experiemtn the AP_{crit} is always higher since we weight more on closer object, which are typically easy recongnize by detectors.

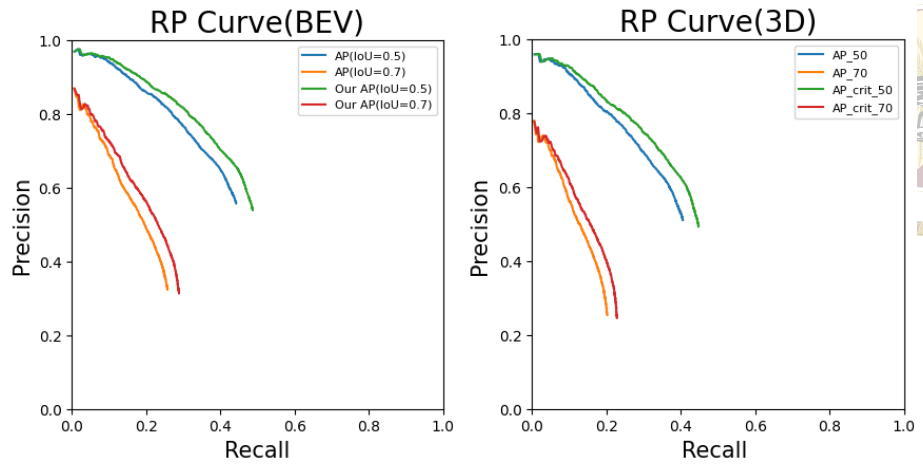
	AP 3D		AP BEV	
Method	AP	AP_{crit}	AP	AP_{crit}
GAC	31.16	34.14	34.99	39.43
MonoFlex	34.04	36.83	37.68	42.03
Pseudo-Lidar	52.38	55.00	55.26	57.84

APcrit places more emphasis on closer objects, which are typically detected more accurately by object detectors. As a result, APcrit tends to have higher values compared to the standard AP metric. At a low-recall rate, the precision may be lower for APcrit due to the significant penalty it imposes on false positives in close proximity. Conversely, at

doi:10.6342/NTU202301439

Figure 6.6: Illustration of our proposed criticality calculation. The left image displays the calculated safety weight for each car object. The right figure presents a bird's-eye view of the scene, where the rectangular boxes represent the 3D location of the objects. Each object is accompanied by a line indicating its orientation, and the dot at the end of the line represents the estimated collision point with the ego-vehicle. This criticality calculation takes into account the relative positions and orientations of the objects to assess their potential safety impact.

doi:10.6342/NTU202301439



(a) Precision-recall curve with BEV IoU (b) Precision-recall curve with 3D IoU

Figure 6.7: Comparison of our proposed AP_{crit} metric with the default AP metric. Our proposed metric assigns higher weights to closer objects, which are generally easier to detect. As a result, the AP_{crit} metric tends to be higher than the default AP metric.

a high-recall rate, the precision tends to be higher for APcrit as it assigns greater weights to easily detectable objects.

6.4 Summary

In this chapter, we discuss evaluation metrics for 3D object detection tasks. While the Average Precision (AP) metric is commonly used to measure detector performance, it may not be well-suited for driving scenarios since it treats all on-road objects equally, regardless of their distance from the ego-vehicle. To address this issue, a safety metric called Critical Average Precision (APcrit) has been proposed for 3D object detection, which weighs objects based on their distance from the ego-vehicle and potential collision point. The criticality of each object is calculated using basic geometric relationships between the object and the ego-vehicle. The results show that APcrit is generally higher than the original AP since it places more weight on closer objects, which are usually easier to detect. The proposed method has been tested on the KITTI3D validation dataset

doi:10.6342/NTU202301439

and can evaluate the performance of several 3D object detection models, identifying their strengths and weaknesses in different scenarios.

doi:10.6342/NTU202301439

doi:10.6342/NTU202301439

Chapter 7 Conclusion

In this thesis, we have explored a range of image recognition tasks and algorithms relevant to driving scenarios. Our focus has been on two fundamental tasks: object detection and segmentation, which provide a detailed understanding of the scene. Through our research, we have aimed to improve the computer vision capabilities of autonomous vehicles.

We began by reviewing the most prominent algorithms for object detection, including Faster-RCNN, YOLO, SSD, and RetinaNet. In addition, we introduced monocular 3D object detection as a novel image task that requires the localization and orientation of objects in 3D space. We proposed a perspective-aware convolution layer that can effectively incorporate scene structure information into the convolutional neural network (CNN) architecture. Our experimental results on the KITTI3D dataset showed that our approach improved depth regression and alleviated the challenges of 3D object detection.

Data augmentation is a commonly used technique in deep learning to increase the diversity of training data. We conducted a comprehensive survey of data augmentation methods used in 3D object detection and proposed a scene-aware copy-paste augmentation method. This method takes into account the scene structure before pasting objects, resulting in improved performance compared to the original copy-paste method. Our pro-

doi:10.6342/NTU202301439

posed method highlights the importance of maintaining appropriate scene structure after data augmentation.

We categorized the segmentation task into three categories: semantic, instance, and panoptic segmentation. We surveyed some popular algorithms used for these tasks, including FCN, U-Net, DeepLabv3, SOLO, YOLACT, and panoptic-DeepLab, and conducted experiments on the Cityscapes dataset. We found that the detect-and-segment approach is effective in the segmentation task and often provides impressive results.

Lastly, we proposed a new safety metric for object detection, as most papers adopt the AP metric, which treats every object equally. We argued that it is unreasonable to do so and proposed a metric that weighs closer objects more and far objects less. Our proposed metric, AP_{crit}, takes into account the distance between on-road objects and the ego-vehicle and the distance to the estimated future collision point. We showed that our safety metric generally produces better results than the original AP metric, as it gives more weight to easily detected objects.

In summary, our research has covered many tasks and algorithms related to deep learning in this thesis. We hope that it will provide valuable insights into how we can improve computer capabilities in image recognition and contribute to the development of autonomous vehicles.

doi:10.6342/NTU202301439

References

[1] H. Abu Alhaija, S. K. Mustikovela, L. Mescheder, A. Geiger, and C. Rother. Augmented reality meets computer vision: Efficient data generation for urban driving scenes. International Journal of Computer Vision, 126:961–972, 2018.

[2] V. Badrinarayanan, A. Kendall, and R. Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE transactions on pattern analysis and machine intelligence, 39(12):2481–2495, 2017.

[3] A. Bansal, J. Singh, M. Verucchi, M. Caccamo, and L. Sha. Risk ranked recall: Collision safety metric for object detection systems in autonomous vehicles. In 2021 10th Mediterranean Conference on Embedded Computing (MECO), pages 1–4. IEEE, 2021.

[4] W. Bao, B. Xu, and Z. Chen. Monofenet: Monocular 3d object detection with feature enhancement networks. IEEE Transactions on Image Processing, 29:2753–2765, 2019.

[5] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. EURASIP Journal on Image and Video Processing, 2008:1–10, 2008.

doi:10.6342/NTU202301439

[6] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. Yolov4: Optimal speed and accuracy of object detection. arXiv preprint arXiv:2004.10934, 2020.

[7] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. Yolact: Real-time instance segmentation. In Proceedings of the IEEE/CVF international conference on computer vision, pages 9157–9166, 2019.

[8] G. Brazil and X. Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 9287–9296, 2019.

[9] A. Ceccarelli and L. Montecchi. Safety-aware metrics for object detectors in autonomous driving. arXiv preprint arXiv:2203.02205, 2022.

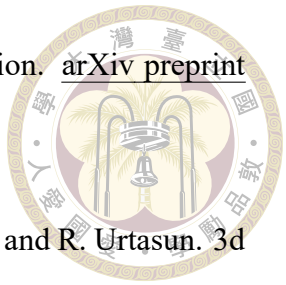
[10] L.-C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4013–4022, 2018.

[11] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence, 40(4):834–848, 2017.

[12] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam. Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:1706.05587, 2017.

[13] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In Proceedings of the European conference on computer vision (ECCV), pages 801–818, 2018.

doi:10.6342/NTU202301439



[14] P. Chen, S. Liu, H. Zhao, and J. Jia. Gridmask data augmentation. arXiv preprint arXiv:2001.04086, 2020.

[15] X. Chen, K. Kundu, Y. Zhu, A. G. Berneshawi, H. Ma, S. Fidler, and R. Urtasun. 3d object proposals for accurate object class detection. Advances in neural information processing systems, 28, 2015.

[16] Y. Chen, L. Tai, K. Sun, and M. Li. Monopair: Monocular 3d object detection using pairwise spatial relationships. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12093–12102, 2020.

[17] B. Cheng, M. D. Collins, Y. Zhu, T. Liu, T. S. Huang, H. Adam, and L.-C. Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 12475–12485, 2020.

[18] M. Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020.

[19] J. Dai, H. Qi, Y. Xiong, Y. Li, G. Zhang, H. Hu, and Y. Wei. Deformable convolutional networks. In Proceedings of the IEEE international conference on computer vision, pages 764–773, 2017.

[20] 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 Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition workshops, pages 1000–1001, 2020.

[21] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. Dssd: Deconvolutional single shot detector. arXiv preprint arXiv:1701.06659, 2017.

doi:10.6342/NTU202301439

[22] H. Fu, M. Gong, C. Wang, K. Batmanghelich, and D. Tao. Deep ordinal regression network for monocular depth estimation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2002–2011, 2018.

[23] R. Girshick. Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), December 2015.

[24] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014.

[25] K. He, G. Gkioxari, P. Dollar, and R. Girshick. Mask r-cnn. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Oct 2017.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[27] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab. Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again. In Proceedings of the IEEE international conference on computer vision, pages 1521–1529, 2017.

[28] A. Kirillov, R. Girshick, K. He, and P. Dollár. Panoptic feature pyramid networks. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 6399–6408, 2019.

[29] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 9404–9413, 2019.

doi:10.6342/NTU202301439

[30] J. Ku, A. D. Pon, and S. L. Waslander. Monocular 3d object detection leveraging accurate proposals and shape reconstruction. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 11867–11876, 2019.

[31] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 12697–12705, 2019.

[32] P. Li, H. Zhao, P. Liu, and F. Cao. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16, pages 644–660. Springer, 2020.

[33] Q. Lian, B. Ye, R. Xu, W. Yao, and T. Zhang. Exploring geometric consistency for monocular 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1685–1694, 2022.

[34] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 2117–2125, 2017.

[35] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. Focal loss for dense object detection. In Proceedings of the IEEE international conference on computer vision, pages 2980–2988, 2017.

[36] L. Liu, J. Lu, C. Xu, Q. Tian, and J. Zhou. Deep fitting degree scoring network for monocular 3d object detection. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 1057–1066, 2019.

doi:10.6342/NTU202301439

[37] R. Liu, J. Lehman, P. Molino, F. Petroski Such, E. Frank, A. Sergeev, and J. Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. Advances in neural information processing systems, 31, 2018.

[38] S. Liu, D. Huang, et al. Receptive field block net for accurate and fast object detection. In Proceedings of the European conference on computer vision (ECCV), pages 385–400, 2018.

[39] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia. Path aggregation network for instance segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 8759–8768, 2018.

[40] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. In B. Leibe, J. Matas, N. Sebe, and M. Welling, editors, Computer Vision – ECCV 2016, pages 21–37, Cham, 2016. Springer International Publishing.

[41] Y. Liu, Y. Yixuan, and M. Liu. Ground-aware monocular 3d object detection for autonomous driving. IEEE Robotics and Automation Letters, 6(2):919–926, 2021.

[42] Z. Liu, Z. Wu, and R. Tóth. Smoke: Single-stage monocular 3d object detection via keypoint estimation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pages 996–997, 2020.

[43] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 3431–3440, 2015.

[44] F. Manhardt, W. Kehl, and A. Gaidon. Roi-10d: Monocular lifting of 2d detection to

doi:10.6342/NTU202301439

6d pose and metric shape. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2069–2078, 2019.

[45] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka. 3d bounding box estimation using deep learning and geometry. In Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, pages 7074–7082, 2017.

[46] D. Park, R. Ambrus, V. Guizilini, J. Li, and A. Gaidon. Is pseudo-lidar needed for monocular 3d object detection? In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 3142–3152, 2021.

[47] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 652–660, 2017.

[48] Z. Qin, J. Wang, and Y. Lu. Monogrnet: A geometric reasoning network for monocular 3d object localization. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 33, pages 8851–8858, 2019.

[49] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

[50] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767, 2018.

[51] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc., 2015.

doi:10.6342/NTU202301439

[52] T. Roddick, A. Kendall, and R. Cipolla. Orthographic feature transform for monocular 3d object detection. arXiv preprint arXiv:1811.08188, 2018.

[53] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18, pages 234–241. Springer, 2015.

[54] J. Sang, Z. Wu, P. Guo, H. Hu, H. Xiang, Q. Zhang, and B. Cai. An improved yolov2 for vehicle detection. Sensors, 18(12):4272, 2018.

[55] K. Santhakumar, B. R. Kiran, T. Gauthier, S. Yogamani, et al. Exploring 2d data augmentation for 3d monocular object detection. arXiv preprint arXiv:2104.10786, 2021.

[56] A. Simonelli, S. R. Bulo, L. Porzi, M. López-Antequera, and P. Kontschieder. Disentangling monocular 3d object detection. In Proceedings of the IEEE/CVF International Conference on Computer Vision, pages 1991–1999, 2019.

[57] K. K. Singh, H. Yu, A. Sarmasi, G. Pradeep, and Y. J. Lee. Hide-and-seek: A data augmentation technique for weakly-supervised localization and beyond. arXiv preprint arXiv:1811.02545, 2018.

[58] S. Srivastava, F. Jurie, and G. Sharma. Learning 2d to 3d lifting for object detection in 3d for autonomous vehicles. In 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 4504–4511. IEEE, 2019.

[59] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1–9, 2015.

doi:10.6342/NTU202301439

[60] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. International journal of computer vision, 104:154–171, 2013.

[61] G. Volk, J. Gamerdinger, A. von Bernuth, and O. Bringmann. A comprehensive safety metric to evaluate perception in autonomous systems. In 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC), pages 1–8. IEEE, 2020.

[62] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li. Solo: Segmenting objects by locations. In Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16, pages 649–665. Springer, 2020.

[63] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen. Solov2: Dynamic and fast instance segmentation. Advances in Neural information processing systems, 33:17721–17732, 2020.

[64] M. Wolf, L. R. Douat, and M. Erz. Safety-aware metric for people detection. In 2021 IEEE International Intelligent Transportation Systems Conference (ITSC), pages 2759–2765. IEEE, 2021.

[65] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019.

[66] S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 1492–1500, 2017.

[67] Y. Xiong, R. Liao, H. Zhao, R. Hu, M. Bai, E. Yumer, and R. Urtasun. Upsnet: A unified panoptic segmentation network. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 8818–8826, 2019.

doi:10.6342/NTU202301439

[68] Y. Yan, Y. Mao, and B. Li. Second: Sparsely embedded convolutional detection. Sensors, 18(10):3337, 2018.

[69] T.-J. Yang, M. D. Collins, Y. Zhu, J.-J. Hwang, T. Liu, X. Zhang, V. Sze, G. Papandreou, and L.-C. Chen. Deeperlab: Single-shot image parser. arXiv preprint arXiv:1902.05093, 2019.

[70] 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. arXiv preprint arXiv:1906.06310, 2019.

[71] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122, 2015.

[72] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In Proceedings of the IEEE/CVF international conference on computer vision, pages 6023–6032, 2019.

[73] 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. arXiv preprint arXiv:2203.13310, 2022.

[74] Y. Zhang, J. Lu, and J. Zhou. Objects are different: Flexible monocular 3d object detection. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 3289–3298, 2021.

[75] C. Zhou. Yolact++ Better Real-Time Instance Segmentation. University of California, Davis, 2020.

doi:10.6342/NTU202301439

[76] X. Zhou, D. Wang, and P. Krähenbühl. Objects as points. arXiv preprint arXiv:1904.07850, 2019.

[77] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 4490–4499, 2018.

[78] X. Zhu, H. Hu, S. Lin, and J. Dai. Deformable convnets v2: More deformable, better results. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 9308–9316, 2019.

doi:10.6342/NTU202301439