3D OBJECT DETECTION AND TRACKING FOR AUTONOMOUS VEHICLES

By

Su Pang

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

Electrical Engineering — Doctor of Philosophy

2022

ABSTRACT

3D OBJECT DETECTION AND TRACKING FOR AUTONOMOUS VEHICLES

By

Su Pang

Autonomous driving systems require accurate 3D object detection and tracking to achieve reliable path planning and navigation. For object detection, there have been significant advances in neural networks for single-modality approaches. However, it has been surprisingly difficult to train networks to use multiple modalities in a way that demonstrates gain over single-modality networks. In this dissertation, we first propose three networks for Camera-LiDAR and Camera-Radar fusion. For Camera-LiDAR fusion, CLOCs (Camera-LiDAR Object Candidates fusion) and Fast-CLOCs are presented. CLOCs fusion provides a multi-modal fusion framework that significantly improves the performance of single-modality detectors. CLOCs operates on the combined output candidates before Non-Maximum Suppression (NMS) of any 2D and any 3D detector, and is trained to leverage their geometric and semantic consistencies to produce more accurate 3D detection results. Fast-CLOCs can run in near real-time with less computational requirements compared to CLOCs. Fast-CLOCs eliminates the separate heavy 2D detector, and instead uses a 3D detector-cued 2D image detector (3D-Q-2D) to reduce memory and computation. For Camera-Radar fusion, we propose TransCAR, a Transformer-based Camera-And-Radar fusion solution for 3D object detection. The cross-attention layer within the transformer decoder can adaptively learn the soft-association between the radar features and vision queries instead of hard-association based on sensor calibration only. Then, we propose to solve the 3D multiple object tracking (MOT) problem for autonomous driving applications using a random finite set-based

(RFS) Multiple Measurement Models filter (RFS-M^3). In particular, we propose multiple measurement models for a Poisson multi-Bernoulli mixture (PMBM) filter in support of different application scenarios. Our RFS-M^3 filter can naturally model these uncertainties accurately and elegantly. We combine learning-based detections with our RFS-M^3 tracker by incorporating the detection confidence score into the PMBM prediction and update step. We have evaluated our CLOCs, Fast-CLOCs and TransCAR fusion-based 3D detector and RFS-M^3 3D tracker using challenging datasets including KITTI, nuScenes, Argoverse and Waymo that are released by academia and industry leaders. Superior experimental results demonstrated the effectiveness of the proposed approaches.

To my loving family for always supporting me and believing in me.

ACKNOWLEDGMENTS

I would like to thank my advisors, Dr. Hayder Radha and Dr. Daniel Morris, for offering me the opportunity to join the Connected and Autonomous Networked Vehicle for Active Safety (CANVAS) program at Michigan State University. I am very honored to have Dr. Radha as my primary advisor. Dr. Radha's mentorship and encouragement have made me accomplish goals that I thought impossible for myself. I truly learned a lot from his knowledge, experience and vision in not only research but also other perspectives. It is my great pleasure to have Dr. Morris as my advisor. The time we spent brainstorming, discussion on research ideas, polishing papers has significantly improved my skills in critical thinking, academic writing and presentation. I would like to express my gratitude to my committee members Dr. Xiaobo Tan and Dr. Ali Zockaie for their advice, insight and mentorship.

I am grateful for my labmates from WAVES lab and 3D Vision Lab, Daniel Kent, Mazin Hnewa, Xiaohu Lu, Yunfei Long and Saif Imran. I will never forget the good memories of us doing experiments on the CANVAS Lincoln MKZ and AutoDrive Shevy Bolt, and the working hours in MSU Foundation building and CANVAS Research Complex in Spartan Village.

Finally, I would like to thank my parents who always support my dreams, my friends who always have my back, and special thanks to my wife Yueyao Ren for always believing in me and supporting me throughout my time at MSU.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction and Motivation

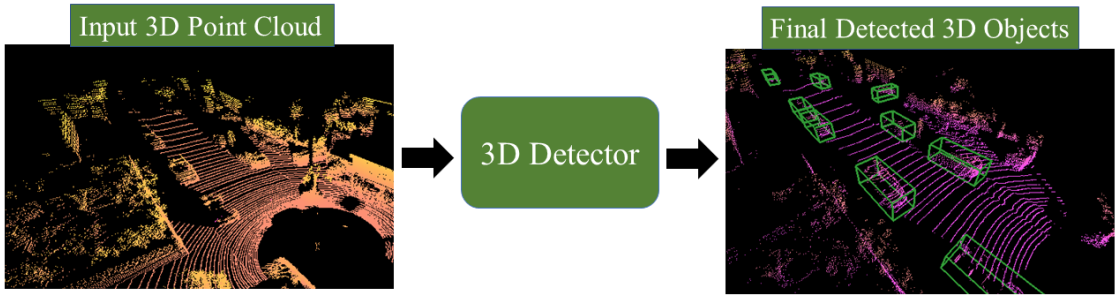
1.1 3D Object Detection

3D object detection, which is the core of 3D scene understanding and situation awareness, is to recognize and determine 3D information of physical objects from sensor data. There are mainly two tasks for 3D object detection: 3D object localization and object classification. Many applications, including autonomous driving, service robots, virtual reality, and so on, require 3D object detection.

Autonomous driving systems require accurate 3D object detection to achieve reliable path planning and navigation. There are different definitions for 3D object detection under different applications; for autonomous driving, 3D object detection is defined as determining the object category, 3D position, heading, and size via a set of 3D bounding boxes from sensor data. As shown in Figure 1.1, compared to 2D object detection, which has been well-studied [13, 14, 15, 16], 3D object detection is more challenging with more output parameters needed to specify 3D oriented bounding boxes around targets. Usually 4 parameters are used to model a 2D bounding box, including top left pixel coordinates of the bounding box x_1, y_1 and bottom right pixel coordinates x_2, y_2 [1]. While at least 7 parameters are needed to represent a 3D bounding box, including 3D center location x, y, z, height h, width w and

[1] There are other ways to represent a 2D bounding box, such as center point pixel coordinate, and height and width, it also requires 4 parameters.

(a) 2D object detection pipeline.

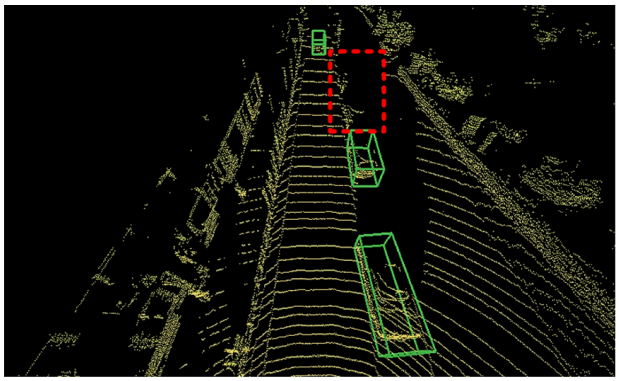


(b) 3D object detection pipeline.

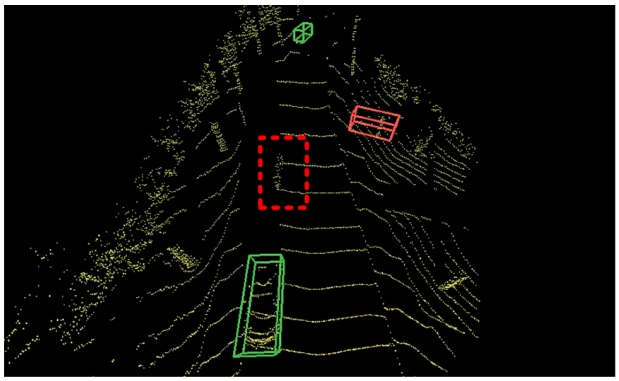
Figure 1.1: 2D and 3D object detection pipelines. (a) 2D detection system takes RGB images (or other sensor data) as input, and outputs 2D axis-aligned bounding boxes. (b) 3D detection system takes 3D point cloud (or other sensor data) and generates classified oriented 3D bounding boxes.

length l of the 3D bounding box, and heading angle θ.

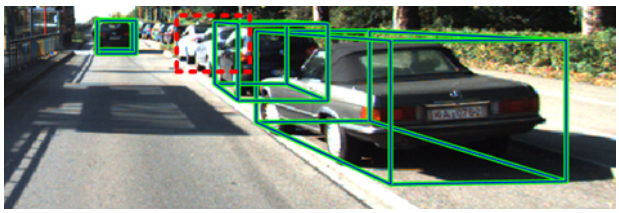
LiDAR is the most commonly used 3D range sensor for accurate 3D object detection. However, LiDAR-based methods [17, 4, 18, 9, 19] are hampered by typically lower input data resolution than video which has a large adverse impact on accuracy at longer ranges. Figure 1.2 illustrates the difficulty of detecting vehicles at long range from just a few points without texture information. The red bounding boxes in Figure 1.2 show some missed detections from a LiDAR-only detector (SECOND [4]) due to heavily occlusion and long distance. In Figure 1.2.(b), the solid red bounding box represents a false positive detection from a LiDAR-only detector. Due to the lack of texture and color information, and the low resolution of LiDAR point cloud, the points within the solid red bounding box are detected as a vehicle. Human annotators use both the camera images together with the LiDAR point clouds to create the 3D ground truth bounding boxes [11]. This motivates camera-LiDAR



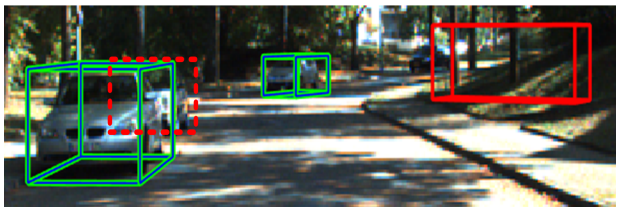
(a) 3D detection using LiDAR point cloud on scene#1.



(b) 3D detection using LiDAR point cloud on scene#2.



(c) LiDAR point cloud detection on scene#1 shown in image.



(d) LiDAR point cloud detection on scene#2 shown in image.

Figure 1.2: The challenge in detecting objects from LiDAR point cloud. Example #1 of car detection from LiDAR-only methods: (a) and (c) show LiDAR-only detector missed two vehicles due to occlusion, highlighted in dashed red bounding box. A second example, #2, is show in (b) and (d), besides the missed detection highlighted in dashed red bounding box, there is also a false positive detection represented in solid red bounding box.

fusion as a way to improve single-modal methods and achieve better 3D object detection.

Monocular cameras can also be used for 3D object detection, and are very popular due to its low cost [20, 8, 21, 22, 12]. It can classify the object accurately, and can predict heading angle and azimuth angle of the object precisely. However, the errors in depth estimation are significant because regressing depth from single image is inherently an ill-posed inverse problem. As shown in Figure 1.3, the 3D properties obtained from monocular images are often orders of magnitude worse than those obtained from a LiDAR point cloud.

Radars have been used for Advanced Driving Assistance System (ADAS) for many years. Despite radar's popularity in the automotive industry, it is challenging to directly estimate

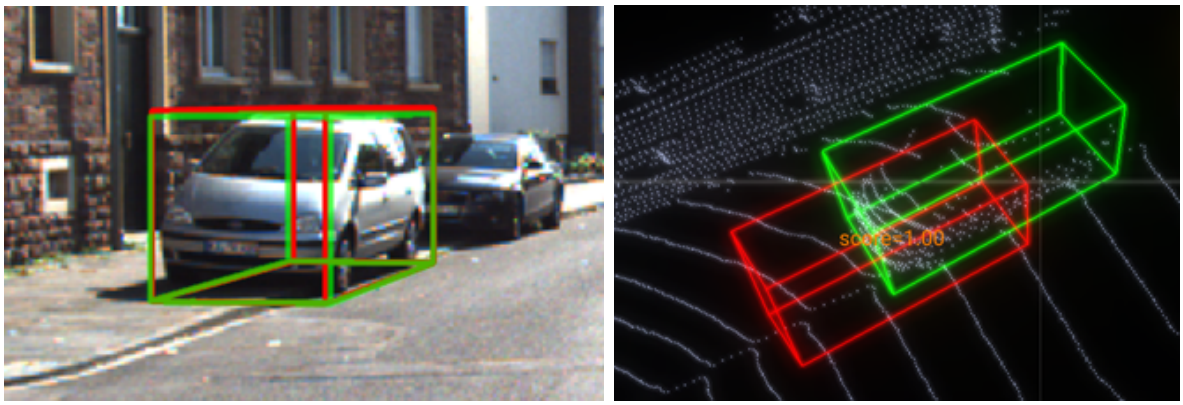


Figure 1.3: Example of M3D-RPN [8] SOTA monocular image-only 3D detections. This example highlights the error in 3D properties learned from monocular images. The red bounding boxes represent the predictions from M3D-RPN, the green bounding boxes stand for the ground truth. As shown in (a), the prediction looks perfect in image, but in 3D space (b), there is significant location error.

classified 3D bounding boxes from automotive radar signal [23, 24, 25]. Compared to LiDAR point clouds, automotive radar signals are much sparser, and they lack height information. These properties make it challenging to distinguish between returns from objects of interest and backgrounds.

While sensor fusion has potential to address the shortcomings of LiDAR-only, camera-only and radar-only detections, finding an effective approach that improves on the state-of-the-art single modality detectors has been difficult. This is illustrated in the official KITTI 3D Detection benchmark leaderboard [2], where LiDAR-only based methods outperform most of the fusion-based methods. Therefore, there is a clear need for further research in multi-modal sensor fusion.

In this dissertation, we propose sensor fusion-based 3D object detection solutions for the two most popular sensor setups in autonomous vehicles and ADAS systems: (1) camera and LiDAR; (2) camera and radar. We first introduce CLOCs (Camera-LiDAR Object

[2]KITTI 3D Detection Leaderboard: http://www.cvlibs.net/datasets/kitti/.

Candidates Fusion) and Fast-CLOCs as a way to achieve improved accuracy for 3D object detection. CLOCs uses much-reduced thresholds for each sensor and combines detection candidates before Non-Maximum Suppression (NMS). By leveraging cross-modality information, it can keep detection candidates that would be mistakenly suppressed by single-modality methods, and remove detection candidates that violate the consistency between different sensors. Fast-CLOCs can run in near real-time with less computational requirements compared to CLOCs. It eliminates the separate heavy image-based 2D detector, and instead uses a 3D detector-cued 2D image detector (3D-Q-2D) to reduce memory and computation.

We then propose TransCAR, a Transformer-based Camera-And-Radar fusion solution for 3D object detection. TransCAR learns radar features from multiple radar scans and then applies transformer decoder to learn the interactions between radar features and vision queries. The cross-attention layer within the transformer decoder can adaptively learn the soft-association between the radar features and vision queries instead of hard-association based on sensor calibration only. In the end, TransCAR estimates a bounding box per query using set-to-set Hungarian loss, which enables the method to avoid NMS. TransCAR improves the velocity estimation using the radar scans without temporal information.

1.2 3D Multiple Object Tracking

3D Multiple object tracking (MOT) is to track independent 3D detections from multiple frames and solve the association between them in 3D space. 3D MOT is a critical module for enabling an autonomous vehicle to achieve robust perception of its environment and, consequently, to achieve safe maneuvering within the environment surrounding the vehicle. There are three main problems in 3D perception: 3D object detection, multiple object

tracking (MOT) and object trajectory forecasting. In a modular perception system, MOT is a critical module that connects detection and forecasting.

For tracking-by-detection approaches, the impact of the quality of input detections that are provided by the underlying detector is of paramount importance. However, due to the complexity of cluttered environments and limitations of learning-based detectors, there are many false positives, misses and inaccurate detections among input detections, as shown in Figure 1.4. The main challenges for MOT in autonomous driving applications are threefold: (1) uncertainty in the number of objects; (2) uncertainty regarding when and where the objects may appear and disappear; (3) uncertainty in objects' states. Traditional filtering based methods, such as Kalman filtering [26, 27, 28], perform well in state update and estimation but can hardly model the unknown number of objects, and the so-called *birth and death* phenomena of objects. Meanwhile, the emergence of random finite set (RFS)[29, 30, 31] based approaches has opened the door to developing theoretically sound Bayesian frameworks that naturally model all the aforementioned uncertainties accurately and elegantly.

RFS-based MOT algorithms have been shown to be very effective for radar-based MOT applications [32, 33]. In particular, Poisson multi-Bernoulli mixture (PMBM) filtering has shown superior tracking performance and favorable computational cost [34] compared to other RFS-based approaches. Consequently, under this work, we propose a PMBM filter to solve the amodal MOT problem for autonomous driving applications. Applying RFS-based trackers for 3D LiDAR data and/or for 2D/3D amodal detections (bounding boxes) has not been well explored. Existing works in this area either underperform state-of-the-art trackers or they have been tested using a small dataset that do not reflect broad and truly challenging scenarios [35, 36, 37]. We believe that RFS-based method could provide a robust and highly effective solution for these emerging detection modalities.

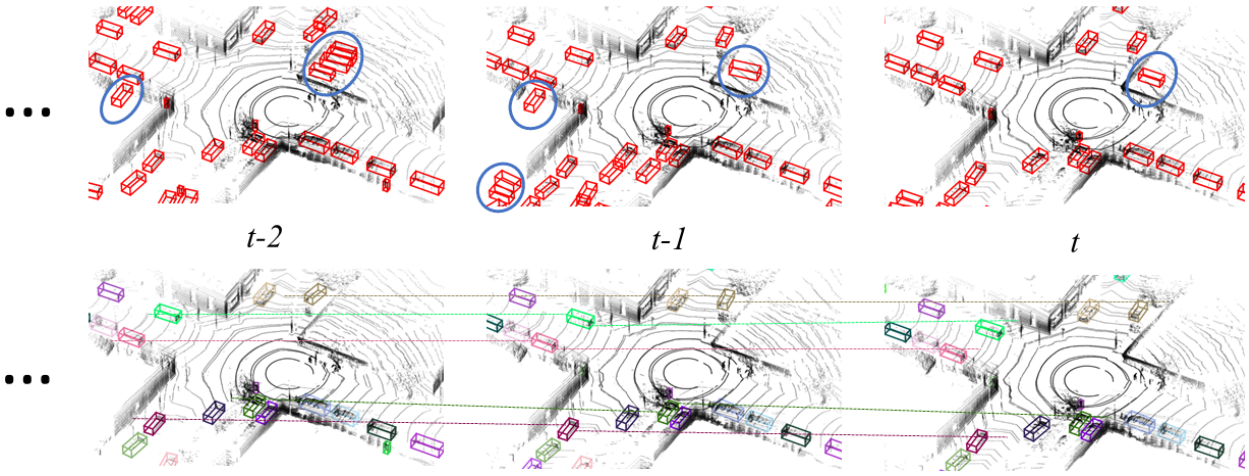


Figure 1.4: Overview of 3D multiple object tracking (MOT) system. For each frame, many 3D detections are generated, as the red bounding boxes on the top row. A 3D MOT system needs to track the targets and filters out false positives (boxes shown within the blue ellipses). For figures in the bottom row, different bounding box colors correspond to different unique tracked IDs. Some tracks with the same IDs are connected with dashed lines to help visualization.

In this dissertation, we propose to solve the 3D MOT problem for autonomous driving applications using a random finite set-based (RFS) Multiple Measurement Models filter (RFS-M^3). In particular, we propose multiple measurement models for a Poisson multi-Bernoulli mixture (PMBM) filter in support of different application scenarios. Our RFS-M^3 filter can naturally model these uncertainties accurately and elegantly. We combine learning-based detections with our RFS-M^3 tracker by incorporating the detection confidence score into the PMBM prediction and update step. To the best of our knowledge, this represents a first successful attempt to employ an RFS-based approach in conjunction with 3D learning-based detections for 3D MOT applications with comprehensive validation using challenging datasets made available by industry leaders.

1.3 Summary of Research Contributions

This dissertation studies 3D object detection and tracking for autonomous vehicle applications. In particular, we explore using neural network-based sensor fusion as a way to achieve improved accuracy for 3D object detection. Then we study applying random finite set to solve the 3D multiple object tracking problem in tracking-by-detection fashion. This dissertation delivers the following contributions:

- A novel Camera-LiDAR Object Candidates (CLOCs) fusion network is proposed to achieve improved accuracy for 3D object detection. CLOCs improves single-modality detectors, including SOTA detectors, to achieve new performance levels. As for now (July 2022), CLOCs ranks the highest among all the monocular camera and LiDAR fusion-based methods in the official KITTI 3D detection leaderboard.

- CLOCs and Fast-CLOCs are designed to exploit the geometric and semantic consistencies between 2D and 3D detections and automatically learns probabilistic dependencies from training data to perform fusion.

- Fast-CLOCs is significantly more memory and computationally efficient than SOTA fusion methods, and improves the SOTA camera-LiDAR fusion performance on the KITTI and nuScenes datasets.

- The proposed 3D-Q-2D image detector within Fast-CLOCs outperforms SOTA image-based detectors in 2D object detection.

- CLOCs uses any pair of pre-trained 2D and 3D detectors without retraining. Fast-CLOCs uses any pre-trained 3D detector without retraining. The modularity and flexibility of CLOCs and Fast-CLOCs enable them to be easily employed by any relevant

already-optimized detection approaches.

- To the best of our knowledge, this represents a first successful attempt to employ transformer for the challenging camera and radar fusion. As for now (July 2022), the proposed TransCAR ranks **1st** among camera-radar fusion-based methods on nuScenes 3D detection benchmark.

- Our studies investigate the inherent difficulties of data association for radar and camera. And we propose to apply transformer to adaptively learn the soft-association and shows superior 3D detection performance compared to hard-association depended on sensor calibration.

- An Random Finite Set-based Multiple Measurement Models filter (RFS-M^3) is proposed to solve the 3D MOT problem for autonomous driving applications. Multiple measurement models ranging from 3D bounding box model to point measurement model for a PMBM filter are studied in support of different application scenarios and optimize the usage of computing resources.

- To the best of our knowledge, this represents a first successful attempt to employ an RFS-based approach that incorporates 3D detections from a neural network for 3D MOT. We validated the performance of our RFS-M^3 tracker using three extensive open datasets provided by three industry leaders – Waymo [7], Argoverse [38] and nuSceness [2].

1.4 Thesis Organization

The rest of this dissertation is organized as follows. Chapter 2 gives more background introduction and reviews related work on 3D object detection and traking. Then, we illustrate our Camera-LiDAR Object Candidates (CLOCs) Fusion architecture and relevant details in Chapter 3. In Chapter 4, we discuss the Fast-CLOCs. We then explain our Transformer-based camera-and-radar fusion network (TransCAR) in Chapter 5. In Chapter 6, we discuss the RFS-M^3 tracker and analyze the experimental results on different open autonomous driving datasets. Chapter 7 provides conclusions of the studies and discusses the future work.

Chapter 2

Background and Related Work

In the previous chapter we introduce the pipelines of 3D object detection and tracking, including the input and output of the system. This chapter introduces the most commonly used evaluation metrics for 3D object detection and tracking tasks. Then we discuss the popular open datasets that are released by academia and industry leaders. These datasets are golden standards for evaluating the 3D perception system of autonomous vehicles. In the end, we review the related works.

2.1 Basics

2.1.1 Evaluation Metrics

There are multiple evaluation metrics for 3D object detection and tracking, we will give a brief introduction on the evaluation metrics that are used in this dissertation, and are most commonly applied in the autonomous driving applications.

2.1.1.1 Evaluation Metrics for 3D Object Detection

As discussed in the previous chapter, the output of a 3D object detection system is a set of classified 3D bounding boxes and corresponding confidence scores. Each 3D bounding box is represented as an 8-digit vector containing 3D dimension (height, width and length), 3D

location (x, y, z), rotation (yaw angle) and confidence score. The confidence score is used to model the confidence estimation of an object from the detector. 3D intersection over union (IoU) is used for most 3D detection benchmarks as the true positive metric [11, 7]. IoU is designed as a scale invariant metric, meaning that doubling the size and relative overlap of two boxes will not change its value. All 3D detections are iteratively assigned to ground truth 3D bounding boxes starting with the largest IoU. True positives are required to overlap by more than a threshold and count multiple detections of the same object as false positives [11]. The threshold would be different for different object classes. The well established average precision (AP) metric as describe in [39] is applied for evaluating the 3D object detection.

In addition to 3D IoU, 2D center distance d on the ground plane between the 3D detection and ground truth is also used as the true positive metric in some 3D detection benchmarks [2]. Compared to 3D IoU, 2D center distance can decouple detection from object size and orientation. This is better for evaluating small objects, such as pedestrians and cyclists, if detected with a small translation error, but 0.0 IoU. Using center distance would make some vision-only methods that tend to have large localization errors better evaluated. Figure 2.1

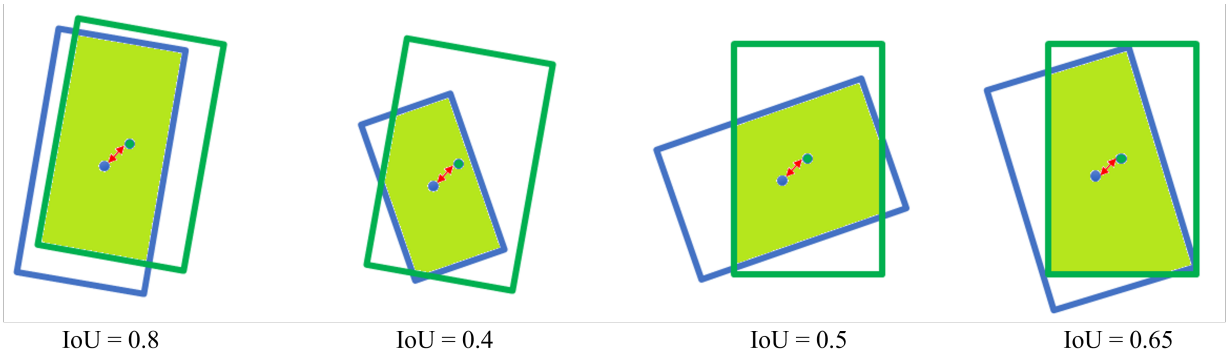


Figure 2.1: The difference between intersection over union (IoU) and center distance shown in 2D. The blue and green bounding boxes in the four examples shown above have the same center distance, while different IoU.

shows the difference between IoU and center distance.

2.1.1.2 Evaluation Metrics for 3D Multiple Object Tracking

3D MOT is to track the independent 3D detections from multiple frames and solve the association between them in 3D space. The output of a 3D MOT system is a set of tracked 3D targets, also referred as 3D tracks, modeled as classified 3D bounding boxes with tracked IDs. For each tracked 3D track, in addition to the 7 parameters $(x, y, z, h, w, l, \theta)$ used to represent the 3D bounding box, there are two more parameters attached: tracked ID and tracked confidence. Tracked ID is used to specify the identity of an object, the same object across multiple frames possesses the same tracked ID. Tracked confidence is the confidence estimation of a track from the tracker.

We use standard evaluation metrics commonly used for MOT [40, 41, 42, 2, 38, 7]. The primary metrics are as follows:

MOTA: Multiple object tracking accuracy, it computes the ratio of all object configuration errors made by the tracker, including false positives, misses, mismatches, over all frames [40]. Usually represented as a percentage or a floating number smaller than 1.0, when there are more mistakes than the number of ground truths, MOTA could be negative.

AMOTA: Average multiple object tracking accuracy, proposed in [6]. Similar to the calculation of average precision (AP), AMOTA averages MOTA across all recall thresholds.

IDF1: $F1$ score, the harmonic mean of precision and recall, denotes as $2(precision * recall)/(precision + recall)$, where $precision$ is the number of true positives over sum of true positives and false positives, $recall$ is the number of true positives over number of total ground truth labels.

There are some other secondary metrics:

MOTP: Multiple object tracking precision. The total error in estimated position for matched object-hypothesis pairs over all frames, averaged by the total number of matches made. It shows the ability of the tracker to estimate precise object positions, independent of its skill at recognizing object configurations, keeping consistent trajectories [40].

AMOTP: Average multiple object tracking precision, proposed in [6]. AMOTP averages MOTP across all recall thresholds.

MOTP-I: Multiple object tracking precision based on intersection over union error. The amodal shape estimation error, computed by the $1 - IoU$ of 3D bounding box projections on xy plane after aligning orientation and centroid.

FP and **FP%**: The number of false positives and the false positive ratio. FP% is defined as the number of false positives divided by the total number of objects [40].

TP and **TP%**: The number of true positives and the true positive ratio. TP% is calculated as the number of true positives divided by the total number of objects [40].

IDS: The number of identity ID switches.

Similar to 3D object detection metrics, the true positive metrics for 3D MOT includes 3D IoU-based [7] and center distance-based [38, 2].

2.1.2 Datasets

There are some open datasets released by academia and industry leaders that are treated as golden standards for evaluating 3D object detection and tracking.

KITTI: The KITTI dataset [11] released by Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago is one of the pioneering works for benchmarking perception systems for autonomous driving applications. The data are collected in the midsize city of Karlsruhe, Germany. KITTI dataset has multiple benchmarks including 2D and 3D ob-

ject detection, 2D MOT, road/lane detection, semantic segmentation, depth completion and so on. Since KITTI doesn't provide 3D tracking benchmark, we only use this dataset for evaluating 3D object detection. The 3D object detection benchmark consists of 7481 training images and 7518 test images as well as the corresponding point clouds, comprising a total of 80,256 labeled objects. Three classes including car, pedestrian and cyclist are well labeled and are counted in the final evaluation. Although the field of view (FOV) of LiDAR is 360 degree, KITTI only labeled the shared FOV between LiDAR and camera which is 90 degree. Objects that are visible for both LiDAR and camera are counted in evaluation. The data collection frequency and annotation frequency are both 10Hz. 3D IoU is used as the true positive metric.

nuScenes: nuScenes dataset is a large-scale autonomous driving dataset with 3D object annotations released in 2018 by Motional (formerly nuTonomy) [2]. It carries the full autonomous vehicle sensor suite: 6 cameras, 5 radars and 1 32-line LiDAR, all with 360 degree field of view (FOV). nuScenes consists of 1000 data sequences collected in Boston and Singapore, each 20 seconds long and fully annotated with 3D bounding boxes for 10 major classes. Among the 1000 data sequences, 850 sequences are for training and validation, the rest 150 sequences are for testing. It has 7× as many annotations and 100× as many images as the pioneering KITTI dataset. nuScenes has both 3D object detection and 3D multi-object tracking benchmarks. The data collection frequency for LiDAR, camera and radar are 20Hz, 12Hz and 13Hz respectively. The annotation frequency is 2Hz. Center distance is used as the true positive metric.

Argoverse: Argoverse was collected in Pittsburgh and Miami and released in 2019 [38] by Argo AI. The 3D tracking dataset includes 360 degree images from 7 cameras with overlapping FOV, 3D point clouds from long range LiDAR. Argoverse 3D tracking dataset consists

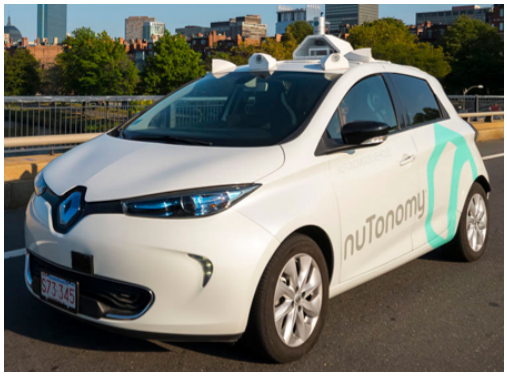
	KITTI	nuScenes	Argoverse	Waymo
Number of data sequences	22	1000	113	1150
Annotated frequency	10Hz	2Hz	10Hz	10Hz
Number of annotated Frames	15K	40K	22K	230K
Hours	1.5	5.5	1	6.4
Number of 3D annotations	80K	1.4M	993K	12M
Number of LiDAR	1	1	2	5
Number of camera	4	6	9	5
Visited area (km^2)	-	5	1.6	76

Table 2.1: Basic information comparison between KITTI, nuScenes, Argoverse and Waymo Datasets.

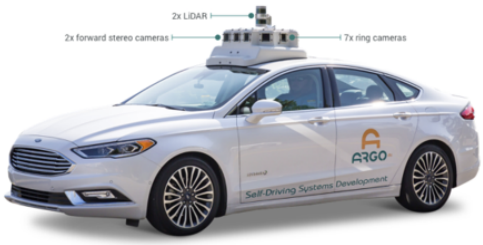
of 113 total number of data sequences and 15-30 seconds for each sequence. The data are divided into three sets: 65 sequences for training ($\sim$ 13000 frames), 24 sequences for validation ($\sim$ 5000 frames) and 24 sequences for test ($\sim$ 4200 frames). The data collection frequency for LiDAR and camera are 10Hz and 30Hz respectively. The annotation frequency is 10Hz. Center distance is applied as the true positive metric.

Waymo: Waymo dataset [7] is released in 2019 and so far is the largest dataset for autonomous driving perception systems. The dataset consists of 1150 data sequences that each span 20 seconds. Among the 1140 data sequences, 1000 sequences are for training and validation, 150 sequences are for testing. The data collection was conducted using 5 LiDARs and 5 high-resolution cameras in multiple cities including San Francisco, Phoenix, and Mountain View, with large geographic coverage within each city. The data collection frequency and annotation frequency are both 10Hz. 3D IoU is applied as the true positive metric.

(a) KITTI data collection vehicle [11].



(b) nuScenes data collection vehicle [2].



(c) Argoverse data collection vehicle [38].

(d) Waymo data collection vehicle [7].

Figure 2.2: Data collection vehicles for different datasets.

2.2 Existing Works in 3D Object Detection

We can classify 3D object detection methods according to the sensor modalities applied. Therefore, the four main categories of 3D object detection are based on (1) cameras, (2) LiDARs, (3) Radars, and (4) Multi-Modal Fusion. Although camera-based methods are attractive for not requiring LiDAR, there is a large gap in 3D performance between these methods and those that utilize LiDARs. Camera-LiDAR Fusion-based methods are supposed to have the best performance, however, the fact is that the overall performance of camera-LiDAR fusion-based methods is still worse than LiDAR-only based methods. According to the KITTI 3D detection leaderboard, the average precision (AP) for the best monocular camera-based 3D detector is 13.87% (MonoEF [43]), for the best stereo camera-based 3D

detector is 64.66% (LIGA-Stereo [44]), while for the best LiDAR-based 3D detector is 82.54% (SE-SSD [45]). Our proposed camera-LiDAR fusion-based method CLOCs [1] is currently the best camera-LiDAR fusion-based method, the AP is 82.28%. CLOCs utilizes image-based 2D detections to boost the performance of LiDAR-only detectors. The current best result of CLOCs (82.28%) is achieved by improving CT3D [46] whose original AP is 81.77%. Ideally, CLOCs is able to boost the best LiDAR-only detector SE-SSD (82.54%) to achieve the best overall performance, but we do not have the source code for reproducing SE-SSD. CLOCs creates a new baseline for camera-LiDAR fusion approaches.

Despite radar's popularity in the automobile industry, few studies focus on fusing radar signals with other sensor data. Compared to LiDAR point clouds, automotive radar signals are much sparser, and they lack height information. However, as will be illustrated in the later chapters, radar has its strengths. Radar can provide accurate depth measurement that monocular camera-based solutions lack of. Our TransCAR uses transformer to fuse radar and vision features, and ranks **1st** among camera-radar fusion-based methods on nuScenes 3D detection benchmark. TransCAR even outperforms some LiDAR-based solutions released in the early stages, such as PointPillars [19].

2.2.1 3D Detection using Camera

Monocular camera-based approaches: Estimating objects' 3D dimensions and locations from monocular 2D images without any references is impractical for traditional computer vision approaches. The development of Convolutional Neural Networks (CNN) and massive annotated training data make it possible. Mousavian *et al.* [20] use geometric constraints between 2D and 3D bounding boxes to recover 3D information. [47, 48] estimate 3D object information by calculating the similarity between 3D objects and CAD models. [8] proposes

to reformulate the monocular 3D detection problem as a standalone 3D region proposal network rather than relying on external networks, data or models. They leverage the geometric relationship of 2D and 3D perspectives, allowing 3D boxes to utilize convolutional features from the image plane. It is always challenging to handle objects that are occluded; to this end, [21] proposes to consider the relationship of paired samples. This allows them to encode spatial constraints for partially occluded objects from their adjacent neighbors. RTM3D [49] studies to predict the nine perspective keypoints of a 3D bounding box in image space, and then utilizes the geometric relationship of 3D and 2D perspectives to recover 3D properties in 3D space. RTM3D is fast and the inference speed is faster than 24Hz. [50] quantifies the impact introduced by each subtask in monocular 3D detection by intensive diagnosis experiments, and find the 'localization error' is the vital factor. They propose three strategies to improve the network. Another category is based on "Pseudo-LiDAR" techniques [51, 52] originally designed for stereo cameras. These methods leverage advances in monocular depth estimation and generate pseudo point clouds first, then training LiDAR-based detectors on the resulting "Pseudo-LiDAR" point clouds to perform 3D object detection [53, 54, 55, 56]. Per-pixel depth estimation is the key for Pseudo-LiDAR-based approaches. The community has made great progress in dense depth prediction [57, 58, 59, 60, 61, 62]. Unlike above methods, DETR3D [12] proposes a top-down transformer-based approach which utilizes initial 3D queries to index 2D image features to refine 3D queries and achieves new SOTA performance.

Stereo camera-based approaches: Compared to monocular camera systems, stereo camera systems can calculate depth with much more accuracy based on disparity maps. With the help of CNNs and large number of annotated data, stereo camera system could achieve much better performance. Pseudo-LiDAR [52] and Pseudo-LiDAR++ [51] explore using

stereo images to generate dense point cloud (with disparity map) and conduct 3D object detection using that cloud. Disp RCNN [63] thinks that the disparity map computed for the entire image in Pseudo-LiDAR is costly and fails to leverage category-specific prior. Therefore, they design an instance disparity estimation network (iDispNet) that predicts disparity only for pixels on objects of interests and learns a category-specific shape prior for more accurate disparity estimation. LIGA-Stereo [44] points out the high-level features learned by stereo-based detectors are easily affected by the erroneous depth estimation due to the limitation of stereo matching, so they utilize superior geometric-aware features from LiDAR-based 3D detection models to guide the training of stereo-based 3D detectors.

These image-based methods are promising, but compared to LiDAR-based techniques, they generate much less accurate 3D bounding boxes.

2.2.2 3D Detection using LiDAR

Point-cloud techniques currently lead in popularity for 3D object detection. Compared to multi-modal fusion based methods, single sensor setup avoids multi-sensor calibration and synchronization issues. However, object detection performance at longer distance is still relatively poor. Methods vary by how they encode and learn features from raw point clouds. PointNet [18] is the seminal work that takes raw point clouds as input and designs a neural network to learn point-wise features directly from point clouds for 3D object detection and instance segmentation. [17] uses voxels to encode the raw point clouds, and 3D Convolutional Neural Networks (CNN) are applied to learn voxel features for classification and bounding box regression. SECOND [4] is the upgrade version of [17], since raw LiDAR point cloud has very sparse data structure, it uses sparse 3D CNNs which reduces the inference time significantly. PointPillars [19] uses PointNets [18] in an encoder that represents point clouds

organized in vertical columns (pillars) followed by a 2D CNN detection head to perform 3D object detection; it enables inference at 62 Hz.

Compared with one-stage methods discussed above, PointRCNN [9], Fast PointRCNN [64] and STD [65] apply a two-stage architecture that first generates 3D proposals in a bottom-up manner and then refines these proposals in a second stage. PV-RCNN [10] leverages the advantages of both 3D voxel CNN and PointNet-based [18] set abstraction to learn more discriminative features. Additionally, Part-A^2 in [66] explores predicting intra-object part locations (lower left, upper right, etc.) in the first stage, and such part locations can assist accurate 3D bounding box refinement in the second stage. In order to achieve better performance in long distance, Pyramid RCNN [67] is proposed to utilize a second-stage module named pyramid RoI head, to adaptively learn the features from the long range sparse points of interests.

Mao *et al.* and Sheng *et al.* introduce transformer into 3D object detection using point clouds and propose Voxel Transformer (VoTr) [68] and Channel-wise Transformer (CT3D) [46]. VoTr enables long-range relationships between voxels by self-attention to capture large context information. CT3D simultaneously performs proposal-aware embedding and channel-wise context aggregation for the point features with each proposal to achieve improved accurate 3D object predictions. SE-SSD [45] utilizes both soft and hard targets with constraints to jointly optimize the SSD (single-stage object detector) [69] through a pair of teacher and student model [70].

2.2.3 3D Detection using Radar

Due to the sparsity nature of radar measurements, it is challenging to use radar as the primary sensor for full 3D object detection. Existing studies mainly focus on 2D and bird's

eye view (BEV) object detection tasks [23, 24, 25]. Major *et al.* [23] propose to encode radar signal as Range-Azimuth-Doppler tensors, then applies 3D convolutions to detect vehicles in the BEV space. Schumann *et al.* [24] use the popular PointNet [18] to extract features from radar point cloud and perform semantic segmentation. Graph Convolution Network (GNN) is also introduced to process the raw radar tensor for 3D object detection [25]. Existing works [23, 24, 24] in this area are mainly for low level autonomy such as collision avoidance instead of advanced self-driving and ADAS system. And they are tested using small datasets that do not reflect broad and truly challenging scenarios.

2.2.4 3D Detection using Multi-Modal Fusion

Camera-LiDAR Fusion-based approaches: Frustum PointNet [71], Pointfusion [72] and Frustum ConvNet [73] are the representatives of 2D driven 3D detectors, which exploit mature 2D detectors to generate 2D proposals and narrow down the 3D processing domain to the corresponding cropped region in the image. But the 2D image-based proposal generation might fail in some cases that could only be observed from 3D space.

MV3D [74] and AVOD [75] project the raw point clouds into bird's eye view (BEV) to form a multi-channel BEV image. A deep fusion based 2D CNN is used to extract features from this BEV image as well as the front camera image for 3D bounding box regression. The overall performance of these fusion-based methods is worse than LiDAR-only-based methods. Possible reasons include: First, transforming raw point clouds into BEV images loses spatial information. Second, the crop and resize operation used in these algorithms in order to fuse feature vectors from different sensor modalities may destroy the feature structure from each sensor. Camera images are high-resolution dense data, while LiDAR point clouds are low-resolution sparse data, fusing these two different types of data structure

is not trivial. Forcing feature vectors from 2D images and 3D LiDAR point clouds to have the same size or equal-length, then concatenating, aggregating or averaging them could result in inaccurate correspondence between these feature vectors and therefore is not the optimal way for fusing features. In order to fuse features from different sensor modalities with better correspondence, MMF [76] adopts continuous convolution [77] to build dense LiDAR BEV feature maps and performs point-wise feature fusion with dense image feature maps. 3D-CVF [78] employs auto-calibrated projection to transform the 2D camera features to a smooth spatial feature map with the highest correspondence to the LiDAR features in the BEV domain, then a gated feature fusion network is applied to fuse the features. PointPainting [79] works by projecting LiDAR points into the semantic segmented image and appending the classification scores to each point, this 'painted' point cloud can then be fed to any LiDAR-based detectors. EPNet [80] presents a fusion module to augment the point features with semantic image features in a point-wise manner. The overall performance of these fusion-based methods is worse than LiDAR-only-based methods. Our proposed CLOCs [1] is currently the best camera-LiDAR fusion-based 3D detector in the KITTI 3D detection leaderboard.

Camera-Radar Fusion-based approaches: Few studies have focused on radar and camera fusion-based 3D object detection. Most existing works either cannot predict 3D bounding boxes or underperform monocular camera-based solutions. Some studies project radar points onto the image plane to assist 2D object detection, especially for distant objects [81, 82]. RODNet [83, 84] proposes a camera-radar fusion cross-model supervision framework for training the radar object detection network. It also proposes a different evaluation metric to evaluate point-based radar detections in range-azimuth coordinates. Nabati *et al.* [85] introduce a radar object proposal network to generate 3D proposals. These proposals are

mapped to the image and fed into a radar proposal refinement network for 3D bounding box refinement. CenterFusion [86] uses a frustum-based method to associate radar features and object center points in BEV. The associated radar features are used to complement image features for better depth estimation in 3D object detection.

2.3 Existing Works for 3D Multi-Object Tracking

In this section, we focus on multiple object tracking (MOT) methods for autonomous driving applications.

2.3.1 Multi-Object Tracking using Traditional Filtering

Kalman filter [26] and its variants are the most popular approaches in this category. Weng *et al* [6] uses a combination of 3D Kalman filter and Hungarian algorithm for state estimation and data association. This method can achieve reasonable performance with very low computational cost and became the baseline methods for many 3D tracking competitions [2, 38, 7]. Chiu *et al* [28] modified [6] by using stochastic information from the Kalman filter in the data association step by measuring the Mahalanobis distance between predicted object states and detections. But the simple birth-and-death management of targets and single distance-based association make these methods struggle in cluttered environments.

2.3.2 Multi-Object Tracking using Neural Networks

Compared to traditional filtering-based approaches, recently developed neural network-based methods can capture the descriptive features and temporal motion features from raw sensor data for MOT. Frossard and Urtasun apply a convolutional neural network (CNN)-based

Match Network to compute a matching cost score to formulate the data association problem as a linear program [87]. Similarly, FANTrack [88] uses a CNN to learn a similarity function that combines cues from both image and spatial features of objects. Weng *et al* proposes GNN3DMOT to use a graph neural network to improve the discriminative feature learning for MOT [89]. Some other methods [90, 91, 92, 93] combine MOT with detection or forecasting to reduce system complexity. These learning-based approaches use complicated networks and require a great deal of training. They have great potential, but tracking performance is so far similar or slightly worse than many filtering-based methods according to popular open 3D tracking benchmark leaderboards.

2.3.3 Multi-Object Tracking using Random Finite Set

A recent family of MOT algorithms is based on RFS [29, 30, 31], including probability hypothesis density (PHD) filter [94], cardinalized PHD filter [95], generalized labeled multi-Bernoulli (GLMB) [96] and PMBM [97, 98]. PHD filter and CPHD filter are two examples of moment approximations of the multi-object density. GLMB and PMBM are examples of using multi-object conjugate priors. Among these RFS-based filtering methods, PMBM filtering has shown superior performance and favourable computational cost [34] when compared to other RFS-based approaches. RFS-based MOT algorithms have been shown to be very effective for point target and extended shape target measurement models MOT applications [32, 33, 37]. However, the 3D input detection format (classified 3D/2D bounding boxes) for modern autonomous driving systems is significantly different from point/extended target models. Applying RFS for 2D/3D amodal detections (bounding boxes) from learning-based detections has not been well explored. Existing works in this area either underperform state-of-the-art trackers or they have been tested using a small dataset that do not reflect

broad and truly challenging scenarios [35, 36, 37].

2.4 Summary of the Chapter

In this chapter we first review some basics about 3D object detection and tracking. For 3D object detection, there are mainly two true positive metrics: 3D IoU-based and center-distance-based. Average precision is widely used to evaluate the performance for 3D object detection. For 3D MOT, there are multiple evaluation metrics from different perspectives, the primary metrics are MOTA and F1 score. Then we discuss the popular open datasets that are released by academia and industry leaders: KITTI, nuScenes, Argoverse and Waymo datasets. These datasets are the golden standards for evaluating 3D perception systems of autonomous vehicles. In the end, we review the existing published works.

From the existing works we can summarize the issues in 3D object detection and tracking. For 3D object detection, camera-based approaches suffer from low accuracy depth estimation, which result in high 'localization error'. Even for stereo camera-based approaches, the depth estimation for long distance is still problematic. The main issues for LiDAR-based approaches are originated from the low resolution of LiDAR scans and the lack of appearance information. These issues make LiDAR-based approaches struggled in handling long-distance and occluded objects. Due to the sparsity nature of radar measurements, it is challenging to use radar as the primary sensor for full 3D object detection. Fusion-based approaches are supposed to have the best performance, but due to the issues in multi-sensor data alignment and feature association, they perform worse than LiDAR-only approaches. Our proposed CLOCs, Fast-CLOCs and TransCAR resolve some of the issues in fusion and achieve new performance levels.

For 3D MOT, traditional Kalman filter-based approaches cannot properly model the birth and death of tracks and other uncertainties in MOT, which makes them struggled in cluttered environments. Neural network-based approaches open a new window for solving 3D MOT, but these methods are much heavier than traditional methods. This is problematic for limited computing resources on the vehicle that are also requested by other heavy perception tasks, such as object detection. The similarity matching network is also not stable for various tracking scenarios. RFS-based approaches can naturally model the uncertainties in MOT applications, but applying RFS with 3D detections (bounding boxes) and LiDAR has not been well explored. Existing works in this area either underperform state-of-the-art trackers or have been tested using small unrealistic datasets. Our proposed RFS-M^3 addresses these issues and outperforms previous methods.

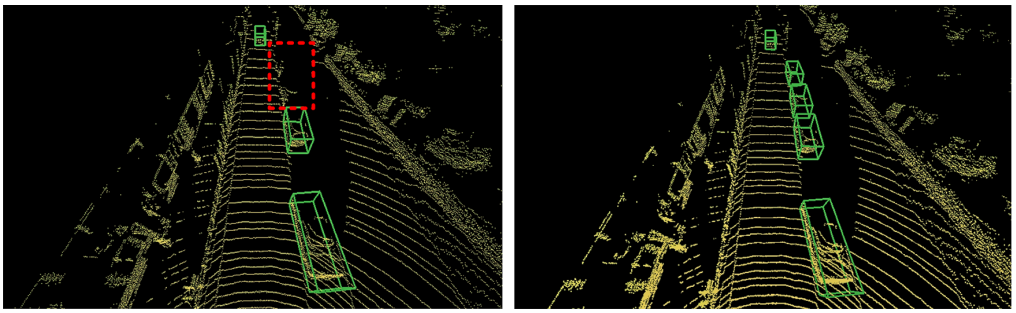
Chapter 3

CLOCs: Camera-LiDAR Object Candidates Fusion for 3D Object Detection

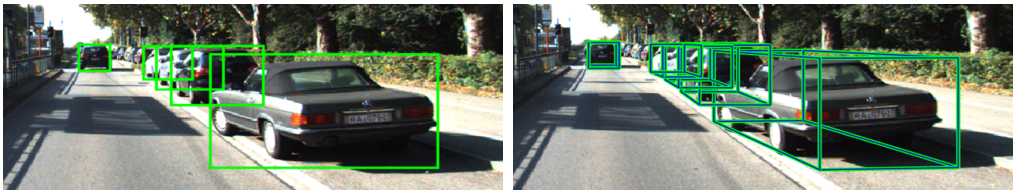
3.1 Introduction

Autonomous driving systems need accurate 3D perception of vehicles and other objects in their environment. Unlike 2D visual detection, 3D-based object detection enables spatial path planning for object avoidance and navigation. Compared to 2D object detection, which has been well-studied [13, 14, 15, 16], 3D object detection is more challenging, with more output parameters needed to specify 3D oriented bounding boxes around targets. In addition, LiDAR methods [17, 4, 18, 9, 19] are hampered by typically lower input data resolution than video, which has a significant adverse impact on accuracy at longer ranges. Figure 3.1 illustrates the difficulty in detecting vehicles from just a few points with no texture at long range. Human annotators use both camera images together with the LiDAR point clouds to create ground truth bounding boxes [11]. This motivates multi-modal sensor fusion as a way to improve single-modal methods.

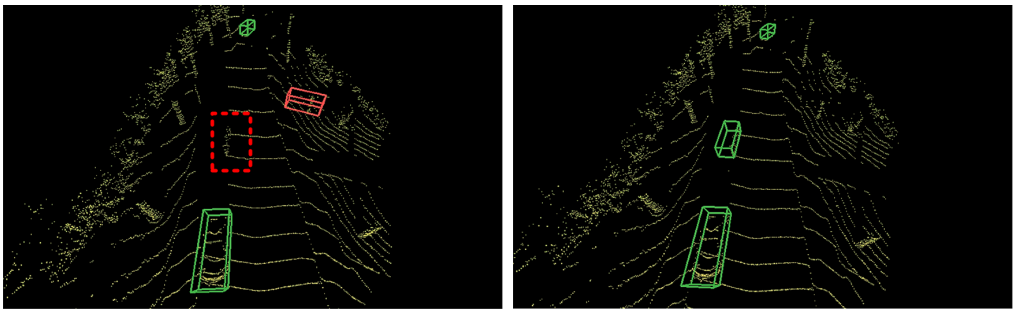
While sensor fusion has the potential to address the shortcomings of video-only and



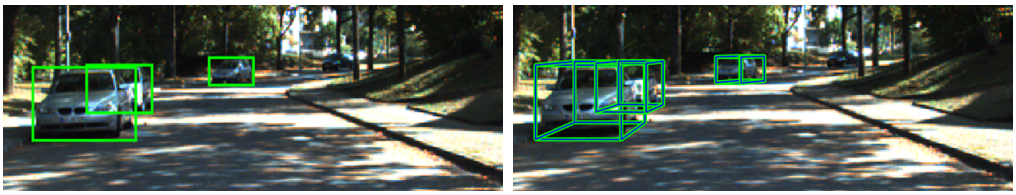
(a) LiDAR-only detector on #1. (b) Our CLOCs fusion on #1.



(c) Image-only detector on #1. (d) CLOCs on #1 shown in img.



(e) LiDAR-only detector on #2. (f) Our CLOCs fusion on #2.



(g) Image-only detector on #2. (h) CLOCs on #2 shown in img.

Figure 3.1: Example #1 of car detection from single modality methods: (a) LiDAR-only detector, and (c) image-only detector, with our CLOCs fusion shown in (b) and (d). A second example, #2, is shown in the bottom 4 sub-figures. Dashed red box shows missed object and solid red bounding box shows false positive detection. Our proposed CLOCs fusion can correct both of these errors.

LiDAR-only detections, finding an effective approach that improves on the state-of-the-art single modality detectors has been difficult. This is illustrated in the official KITTI 3D object detection benchmark leaderboard, where LiDAR-only based methods outperform

most of the fusion based methods. Fusion methods can be divided into three broad classes: early fusion, deep fusion and late fusion, each with their own pros and cons. Although early and deep fusion have greatest potential to leverage cross modality information, they suffer from sensitivity to data alignment, often involve complicated architectures [74, 75, 72, 77], and typically require pixel-level correspondences of sensor data. On the other hand, late fusion systems are much simpler to build as they incorporate pre-trained, single-modality detectors without change, an only need association at the detection level. Our late fusion approach uses much-reduced thresholds for each sensor and combines detection candidates before Non-Maximum Suppression (NMS). By leveraging cross-modality information, it can keep detection candidates that would be mistakenly suppressed by single-modality methods.

We propose Camera-LiDAR Object Candidates Fusion (CLOCs) as a way to achieve improved accuracy for 3D object detection. The proposed architecture delivers the following contributions:

- **Versatility & Modularity**: CLOCs uses any pair of pre-trained 2D and 3D detectors without requiring re-training, and hence, can be readily employed by any relevant already-optimized detection approaches.

- **Probabilistic-driven Learning-based Fusion**: CLOCs is designed to exploit the geometric and semantic consistencies between 2D and 3D detections and automatically learns probabilistic dependencies from training data to perform fusion.

- **Speed and Memory**: CLOCs is fast, leveraging sparse tensors with low memory footprint, which only adds less than 3ms latency for processing each frame of data on a desktop-level GPU.

- **Detection Performance**: CLOCs improves single-modality detectors, including state-

of-the-art detectors, to achieve new performance levels. At the time of submission, CLOCs ranks the highest among all the fusion-based methods in the official KITTI leaderboard.

3.2 2D and 3D Object Detection

We first introduce the basic representations of 2D and 3D object detection used in this dissertation. 2D detection systems discussed in this dissertation take RGB images as input, and output classified 2D axis-aligned bounding boxes with confidence scores, as shown in Figure 3.2. 3D detection systems generate classified oriented 3D bounding boxes with confidence scores, as shown in Figure 3.2. In the KITTI dataset [11] only rotation in z axis is considered (yaw angle), while rotations in x and y axis is set to zero for simplicity. Using calibration parameters of the camera and LiDAR, the 3D bounding box in the LiDAR coordinate can be accurately projected into the image plane, as shown in Figure 3.2. As we will discuss in section 3.5.1, this projection motivates us to design an IoU-based metric in the image plane to quantify the geometric consistency between 3D detections and 2D detections.

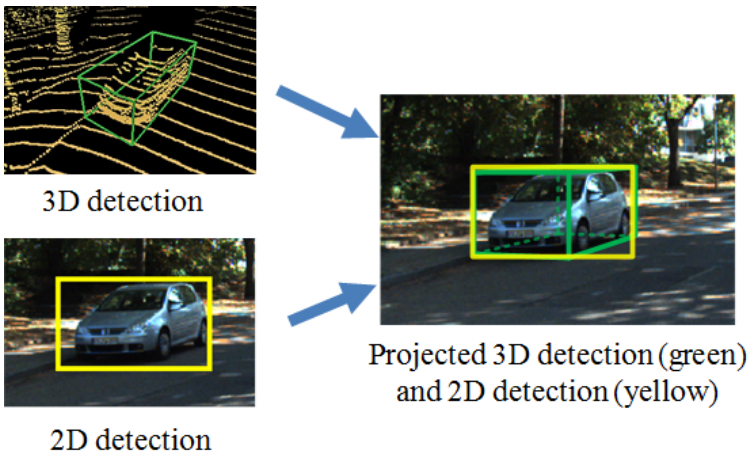


Figure 3.2: 2D and 3D object detection. An object that is correctly detected by both a 2D and 3D detector will have highly overlapped bounding boxes in the image plane.

3.3 Existing Issues for Camera-LiDAR Fusion

Data association between different sensor modalities is arguably the most challenging issue for multi-modal sensor fusion. While 3D LiDAR points can be projected into a corresponding image to obtain a 2D–3D association, the LiDAR points are typically sparse, resulting in incomplete pixel association, and moreover, association typically has errors. There are multiple reasons for erroneous associations, including differing visibility, occlusions due to displaced sensor viewpoints, as well as scan-time differences from a moving platform, as illustrated in Figure. 3.3.

These mismatches are particularly problematic for early and deep fusion. These methods rely on fused data structures including image-augmented LiDAR point cloud [79, 80], LiDAR-augmented depth image [99] and BEV grid-map that stores visual information and LiDAR points from a pillar space into a grid [74, 75]. The mismatch error in the original pixel association can harm the downstream learning due to difficulty in distinguishing backgrounds and foregrounds.

The mismatch exists in feature domains as well. In early and deep fusion methods, some fusion operations are performed in the intermediate stage; this includes the concatenation,

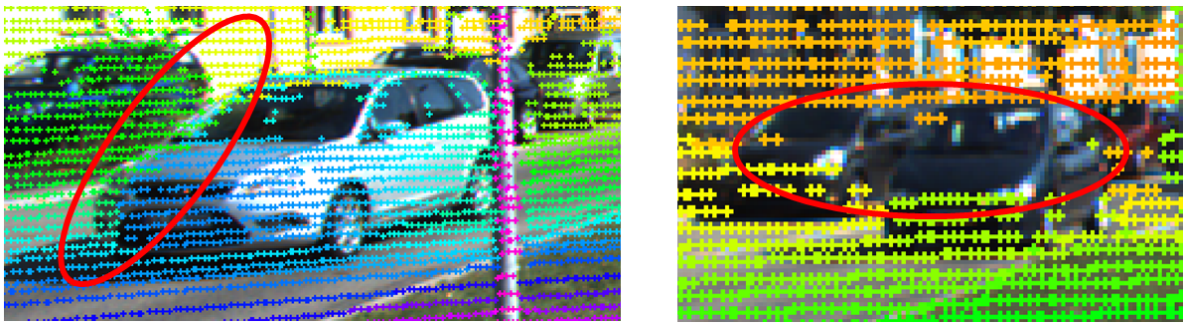


Figure 3.3: Examples of mismatch in projected LiDAR points and camera image pixels. Left ellipse shows many points from the background projected on the vehicle. The right ellipse shows missing LiDAR points and so no association for camera pixels.

aggregation and averaging of visual and point cloud features. Cropping, resizing and pooling operations are needed to convert them into the same shape. These steps can result in inaccurate correspondence between these feature vectors and therefore likely degrading the final fusion performance.

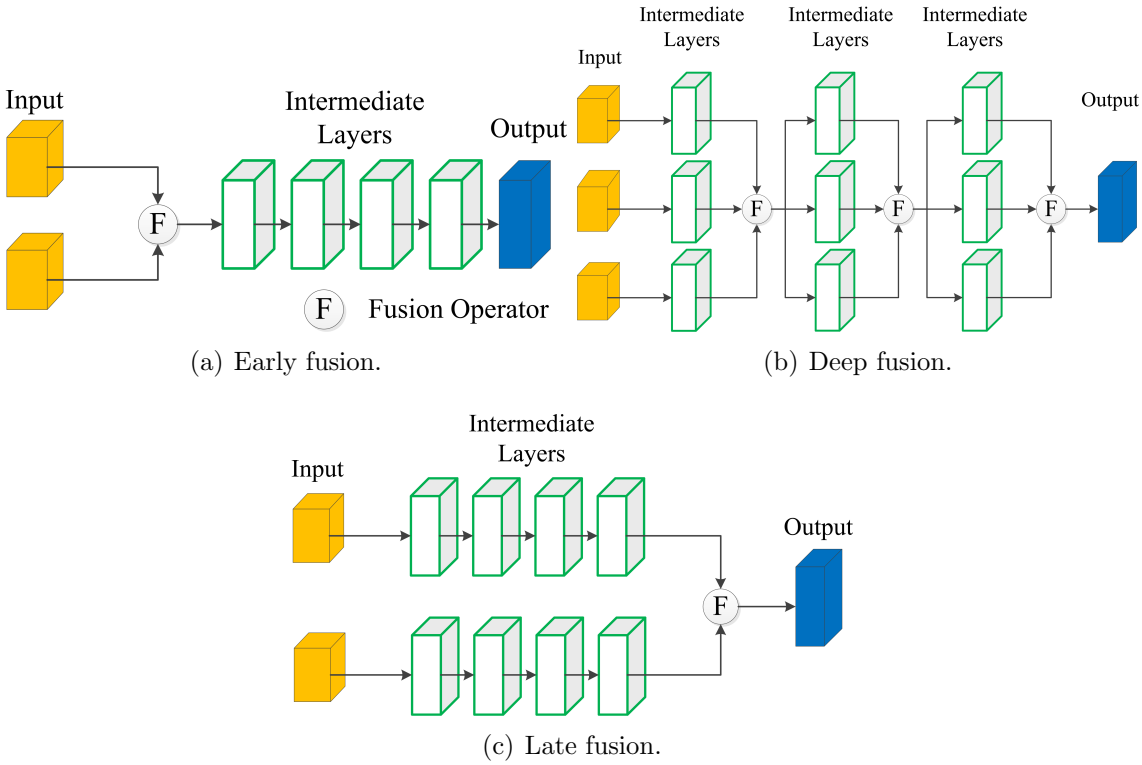
3.4 Why Fusion of Detection Candidates

Fusion architectures can be categorized according to the point at which features from different modalities are combined. As shown in Figure 3.4, three general categories are (1) *early fusion* that combines data at input, (2) *deep fusion* that has different networks for different modalities while simultaneously combining intermediate features, and (3) *late fusion* that processes each modality on a separate path and fuses the outputs at the decision level.

Early fusion has the greatest opportunity for cross-modal interaction, but at the same time inherent data differences between modalities including alignment, representation, and sparsity are not necessarily well-addressed by passing them all through the same network.

Deep fusion addresses this issue by including separate channels for different modalities, while still combining features during processing. This is the most complicated approach, and it is not easy to determine whether complexity actually leads to real improvements.

Late fusion has a significant advantage in training; single-mode algorithms can be trained using their own sensor data. Hence, the multi-modal data do not need to be synchronized or aligned with other modalities. Only the final fusion step requires jointly aligned and labeled data. Additionally, the detection level data on which late fusion operates are compact and simple to encode for a network. Since late fusion prunes rather than creates new detections, it is important that the input detectors be tuned to maximize their recall rate rather than



(a) Early fusion.

(b) Deep fusion.

(c) Late fusion.

Figure 3.4: Different fusion architectures. (a) Early fusion combines the input sensor data at the input stage. (b) Deep fusion has different channels for each sensor modalities and fuses the features at the intermediate stage. (c) Late fusion processes each modality on a separate path and fuses the outputs in the decision level.

their precision. In practice, this implies that individual modalities (a) avoid the NMS stage, which may mistakenly suppress true detections, and (b) keep thresholds as low as possible.

In our late fusion framework, we incorporate all detection candidates before NMS in the fusion step to maximize the probability of extracting all potential correct detections. Our approach is data-driven; we train a discriminative network that receives as input the output scores and classifications of individual detection candidates, as well as spatial descriptions of the detection candidates. It learns from the data how best to combine the input detection candidates for the final output detection.

3.5 Camera-LiDAR Object Candidates Fusion

3.5.1 Geometric and Semantic Consistencies

For a given frame of image and LiDAR data, there may be many detection candidates of with various confidences in each modality from which we seek a single set of 3D detections and scores. Fusing these detection candidates requires an association between the different modalities (even if the association is not unique). For this, we build a geometric association score and apply semantic consistency. These are described in more detail as follows.

3.5.1.1 Geometric Consistency

As discussed in Section 3.2 and shown in Figure 3.2, an object that is correctly detected by both a 2D and a 3D detector will have highly overlapped bounding boxes in the image plane, where false positives are less likely to have highly overlapped bounding boxes. Small errors in pose will result in a reduction of overlap. This motivates an image-based Intersection over Union (IoU) of the 2D bounding box and the bounding box of the projected corners of the 3D detection, to quantify geometric consistency between a 2D and a 3D detection.

There are 8 corner points and 12 edges in a 3D bounding box, after projecting them on the image plane, the contour would be an irregular polygon, see Figure 3.5. It would be slightly complicated to calculate the IoU between an irregular polygon and a 2D bounding box (from the image detector). Therefore, in this dissertation, we use the maximum and minimum xy pixel coordinates from the 8 corner points to form an axis-aligned 2D bounding box as the projected 3D bounding box. So, in the following sections, we refer to this axis-aligned 2D bounding box as the projected 3D detection.

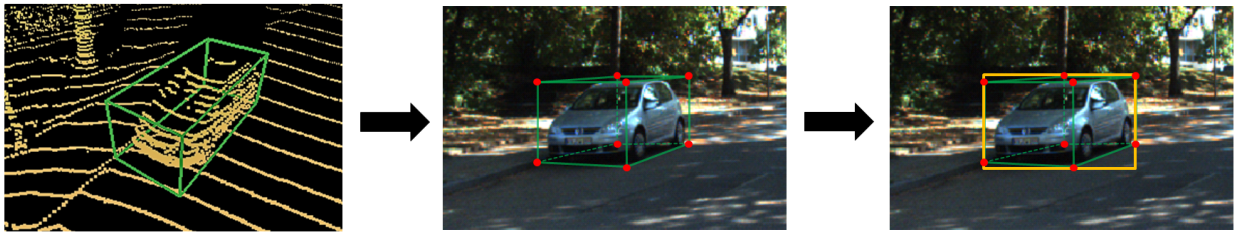


Figure 3.5: The process of projecting 3D bounding box in the LiDAR coordinate onto the image plane, and leveraging the 8 corner points (highlighted in red) to build axis-aligned 2D bounding box (the orange box in the third figure). This 2D bounding box is referred as the projected 3D detection, and used to quantify the geometric consistency.

3.5.1.2 Semantic Consistency

Detectors may output multiple categories of objects, but we only associate detections of the same category during fusion. We avoid thresholding detections at this stage (or use very low thresholds) and leave thresholding to the final output based on the final fused score.

The two types of consistencies illustrated above is the fundamental concept used in our fusion network.

3.5.2 Network Architecture

In this section we explain the preprocessing/encoding of fused data, the fusion network architecture and the loss function used for training.

3.5.2.1 Sparse Input Tensor Representation

The goal of our encoding step is to convert all individual 2D and 3D detection candidates into a set of all consistent joint detection candidates which can be fed into our fusion network. The general output of a 2D object detector is a set of 2D bounding boxes in the image plane and the corresponding confidence scores. For k 2D detection candidates in one image can

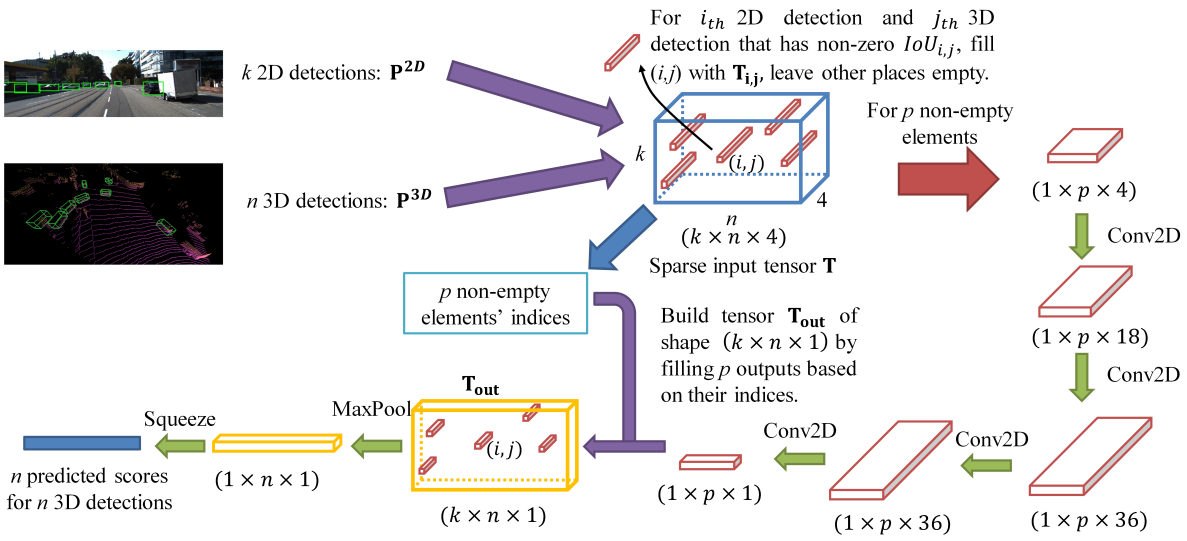


Figure 3.6: CLOCs Fusion network architecture. First, individual 2D and 3D detection candidates are converted into a set of consistent joint detection candidates (a sparse tensor, the blue box); Then a 2D CNN is used to process the non-empty elements in the sparse input tensor; Finally, this processed tensor is mapped to the desired learning targets, a probability score map, through MaxPooling.

be defined as follows:

$$\begin{aligned} \mathbf{P^{2D}} &= \{\mathbf{p_1^{2D}}, \mathbf{p_2^{2D}}, ...\mathbf{p_k^{2D}}\}, \\ \mathbf{p_i^{2D}} &= \{[x_{i1}, y_{i1}, x_{i2}, y_{i2}], s_i^{2D}\} \end{aligned} \tag{3.1}$$

$\mathbf{P^{2D}}$ is the set of all k detection candidates in one image, for i_{th} detection $\mathbf{p_i^{2D}}$, x_{i1}, y_{i1} and x_{i2}, y_{i2} are the pixel coordinates of the top left and bottom right corner points from the 2D bounding box. s_i^{2D} is the confidence score.

The output of 3D object detectors are 3D oriented bounding boxes in LiDAR coordinate and confidence scores. There are multiple ways to encode the 3D bounding boxes, in KITTI dataset [11], a 7-digit vector containing 3D dimension (height, width and length), 3D location (x,y,z) and rotation (yaw angle) is used. For n 3D detection candidates in one LiDAR scan

can be defined as follows:

$$\begin{aligned}\mathbf{P^{3D}} &= \{\mathbf{p_1^{3D}}, \mathbf{p_2^{3D}}, ...\mathbf{p_n^{3D}}\}, \\ \mathbf{p_i^{3D}} &= \{[h_i, w_i, l_i, x_i, y_i, z_i, \theta_i], s_i^{3D}\}\end{aligned} \tag{3.2}$$

where $\mathbf{P^{3D}}$ is the set of all n detection candidates in one LiDAR scan, for i_{th} detection $\mathbf{p_i^{3D}}$, $[h_i, w_i, l_i, x_i, y_i, z_i, \theta_i]$ is the 7-digit vector for 3D bounding box. s_i^{3D} is the 3D confidence score. Note that we take 2D and 3D detections without doing NMS, as discussed in the previous section, some correct detections may be suppressed because of limited information from single sensor modality. Our proposed fusion network would re-evaluate all detection candidates from both sensor modalities to make better predictions. For k 2D detections and n 3D detections, there are $k \times n$ combinations of associations; therefore, we build a $k \times n \times 4$ tensor $\mathbf{T}$, as shown in Figure 3.6. For each element $\mathbf{T_{i,j}}$, there are 4 channels denoted as follows:

$$\mathbf{T_{i,j}} = \{IoU_{i,j}, s_i^{2D}, s_j^{3D}, d_j\} \tag{3.3}$$

where $IoU_{i,j}$ is the IoU between i_{th} 2D detection and j_{th} projected 3D detection, s_i^{2D} and s_j^{3D} are the confidence scores for i_{th} 2D detection and j_{th} 3D detection, respectively. d_j represents the normalized distance between the j_{th} 3D bounding box and the LiDAR in the xy plane. Elements $\mathbf{T_{i,j}}$ with zero IoU are eliminated as they are geometrically inconsistent.

The input tensor $\mathbf{T}$ is sparse because for each projected 3D detection, only few 2D detections intersect with it and so most of the elements are empty. The fusion network only needs to learn from these intersected examples. Because we take the raw predictions before NMS, k and n are large numbers, for SECOND [4], there are 70400 ($200 \times 176 \times 2$) predictions

in each frame. It would be time consuming to do 1×1 convolution on a dense tensor with this shape. We propose an implementation architecture to utilize the sparsity of tensor $\mathbf{T}$ and make the calculations much faster and feasible for large k and n values. Only non-empty elements are delivered to the fusion network for processing, shown in Figure 3.6. As we shall discuss later, the indices of the non-empty elements (i, j) are important for further calculations; therefore, the indices of these non-empty elements are saved in the cache, as shown in the blue box in Figure 3.6. Here, note that for projected 3D detection $\mathbf{p_j}$ that does not have 2D detection intersected, we still fill the last element in the j_{th} column $\mathbf{T_{k,j}}$ in $\mathbf{T}$ with the available 3D detection information and set $IoU_{k,j}$ and s_k^{2D} as -1. Because sometimes 3D detector could detect some objects that 2D detector could not, and we do not want to discard these 3D detections. Setting the IoU and s^{2D} to -1 rather than 0 enables our network to distinguish this case from other examples with very small IoU and s^{2D}.

3.5.2.2 Network Details

The fusion network is a set of 1×1 2D convolution layers. We use Conv2D($c_{in}, c_{out}, \mathbf{k}, \mathbf{s}$) to represent an 2 dimensional convolution operator where c_{in} and c_{out} are the number of input and output channels, $\mathbf{k}$ and $\mathbf{s}$ are the kernel size vector and stride respectively. We employ four convolution layers sequentially as Conv2D(4, 18, (1,1), 1), Conv2D(18, 36, (1,1), 1), Conv2D(36, 36, (1,1), 1) and Conv2D(36, 1, (1,1), 1), which yields a tensor of size $1 \times p \times 1$ shown in Figure 3.6, where p is the number of non-empty elements in the input tensor $\mathbf{T}$. Note that for the first three convolution layers, after each convolution layer applied, ReLU [100] is used. Since we have saved the indices of these non-empty elements (i, j), as shown in Figure 3.6 now we could build a tensor $\mathbf{T_{out}}$ of shape $k \times n \times 1$ by filling p outputs based on the indices (i, j) and putting negative infinity elsewhere. Because one projected

3D detection candidate could intersect with more than one 2D detection candidate, in other words, p is always larger than the number of 3D detection candidates n. For some columns in $\mathbf{T_{out}}$, there could be multiple non-empty elements. Therefore, 2D max-pooling is used along each column of $\mathbf{T_{out}}$ to obtain the final predicted fused confidence scores for n 3D detection candidates.

3.5.2.3 Loss Function

We use a cross entropy entropy loss for target classification, modified by the focal loss in [16] with parameters $\alpha = 0.25$ and $\gamma = 2$ to address the large class imbalance between targets and background.

3.5.3 Training

The fusion network is trained using stochastic gradient descent (SGD). We use the Adam optimizer with an initial learning rate of $3 * 10^{-3}$ and decay the learning rate by a factor of 0.8 for 15 epochs.

3.6 Experimental Results

In this section we present our experimental setup and results, including dataset, platform, performance results and analyses. For all experiments, we focus on the car class since it has the most training and testing samples in the KITTI [11] dataset.

3.6.1 Dataset

Our fusion system is evaluated on the challenging 3D object detection benchmark KITTI dataset [11] which contains both LiDAR point clouds and camera images. There are 7481 training samples and 7518 testing samples. Ground truth labels are only available for training samples. For the evaluation of testing samples, one needs to submit the detection results to the KITTI server. Since there are submission time limits for the KITTI evaluation server, for experimental studies, we follow the convention in [101] to split the original training samples into 3712 training samples and 3769 validation samples. We compare our method with sate-of-the-art multi-modal fusion methods of 3D object detection on official test split of KITTI as well as validation split.

3.6.2 2D/3D Detector Setup

We apply our CLOCs fusion network for a combination of different 2D and 3D detectors to demonstrate the flexibility of our proposed pipeline. The 2D detectors we used include: RRC [102], MS-CNN [103] and Cascade R-CNN [3]. The 3D detectors we incorporated are: SECOND [4], PointPillars [19], PointRCNN [9], PV-RCNN [10] and CT3D [46]. Although not all are the top performers within the KITTI leaderboard, we have selected these methods because they are the best currently-available open-source detectors. Our experiments show that CLOCs improves the performance of these detectors significantly. At the time of submission, CLOCs fusion of CT3D with Cascade R-CNN, is ranked number 2 on the KITTI 3D detection leaderboard, number 1 on the 2D detection leaderboard, and outperforms all other fusion methods.

3.6.3 Evaluation Results

We evaluate the detection results on the KITTI test server. The IoU threshold for car is 0.7. All the instances are classified into three difficulty levels: easy, moderate and hard, based on their 2D bounding boxes' height in the image plane, occlusion level and truncation level. All methods in the KITTI leaderboards are ranked according to the moderate difficulty level. For hard level, $\sim 2\%$ of the ground truth bounding boxes have not been recognized by humans, thereby upper bounding recall at 98%. Since KITTI has some restrictions on the number of submissions, we only show the results evaluated on the official KITTI test server from four fusion combinations of 2D and 3D detectors, which are SECOND [4] and Cascade R-CNN [3], written as CLOCs_SecCas, PointRCNN [9] and Cascade R-CNN, as CLOCs_PointCas, PV-RCNN [10] and Cascade R-CNN, as CLOCs_ PVCas, CT3D [46] and Cascade R-CNN, as CLOCs_CTCas. All other combinations are evaluated on the validation set.

Table 3.1 shows the performance of our fusion method on the KITTI test set through server submission. Our methods outperform all multi-modal fusion-based works in moderate and hard level at the time of submission. Note that the official open-source code of PV-RCNN performs slightly worse than the private one owned by the PV-RCNN authors shown on the KITTI leaderboard, and our CLOCs_PVCas result is based on the open-source PV-RCNN. The baseline PV-RCNN in Table 3.1 refers to the open-source PV-RCNN. As shown in Table 3.1, compared to baseline methods SECOND, PointRCNN and PV-RCNN, fusion with Cascade R-CNN through our fusion network increases the performance in 3D and BEV object detection by a large margin.

We evaluate the performance of all combinations of 2D and 3D detectors on car class of

Detector	Input Data	3D AP (%)			Bird's Eye View AP (%)		
		easy	moderate	hard	easy	moderate	hard
SECOND (baseline) [4]	LiDAR	83.34	72.55	65.82	89.39	83.77	78.59
CLOCs_SecCas (SECOND+Cascade R-CNN)	LiDAR+Img	86.38	78.45	72.45	91.16	88.23	82.63
Improvement (CLOCs_SecCas over SECOND)	-	+3.04	+5.90	+6.63	+1.77	+4.46	+4.04
PointRCNN (baseline) [9]	LiDAR	86.23	75.81	68.99	92.51	86.52	81.39
CLOCs_PointCas (PointRCNN+Cascade R-CNN)	LiDAR+Img	87.50	76.68	71.20	92.60	88.99	81.74
Improvement (CLOCs_PointCas over PointRCNN)	-	+1.27	+1.04	+2.21	+0.09	+2.47	+0.35
PV-RCNN (baseline) [10]	LiDAR	87.45	80.28	76.21	91.91	88.13	85.41
CLOCs_PVCas (PV-RCNN+Cascade R-CNN)	LiDAR+Img	88.94	**80.67**	77.15	93.05	**89.80**	**86.57**
Improvement (CLOCs_PVCas over PV-RCNN)	-	+1.49	+0.39	+0.94	+1.14	+1.67	+1.17
CT3D (baseline) [46]	LiDAR	88.50	80.19	**77.16**	92.62	88.95	84.18
CLOCs_CTCas (CT3D+Cascade R-CNN)	LiDAR+Img	89.16	**82.28**	**77.23**	92.91	89.48	**86.42**
Improvement (CLOCs_CTCas over CT3D)	-	+0.66	+2.09	+0.07	+0.29	+0.53	+2.24
F-PointNet [71]	LiDAR+Img	82.19	69.79	60.59	91.17	84.67	74.77
AVOD-FPN [75]	LiDAR+Img	83.07	71.76	65.73	90.99	84.82	79.62
F-ConvNet [73]	LiDAR+Img	87.36	76.39	66.69	91.51	85.84	76.11
UberATG-MMF [76]	LiDAR+Img	88.40	77.43	70.22	93.67	88.21	81.99
UberATG-ContFuse [77]	LiDAR+Img	83.68	68.78	61.67	**94.07**	85.35	75.88
3D-CVF [78]	LiDAR+Img	**89.20**	80.05	73.11	93.52	**89.56**	82.45
PI-RCNN [104]	LiDAR+Img	84.37	74.82	70.03	91.44	85.81	81.00
EPNet [80]	LiDAR+Img	**89.81**	79.28	74.59	**94.22**	88.47	83.69

Table 3.1: Performance comparison of object detection with state-of-the-art camera-LiDAR fusion methods on car class of KITTI test set (new 40 recall positions metric). CLOCs fusion improves the baselines and outperforms other state-of-the-art fusion-based detectors. 3D and bird's eye view detection are evaluated by Average Precision (AP) with the best in green and second-best in blue.

KITTI validation set, the results are shown in Table 3.2. Compared to the corresponding baseline 3D detectors, our fusion methods have better performance in 3D and BEV detection benchmarks. These results demonstrate the effectiveness and flexibility of our fusion approach.

Table 3.3 and Table 3.4 show the 3D and BEV evaluation results of pedestrian and cyclist on KITTI validation set. The IoU threshold for pedestrian and cyclist is 0.5. Here, for 3D detectors, only SECOND [4] and PointPillars [19] publish their training configurations for

Detector	3D AP (%)			Bird's Eye View AP (%)		
	easy	moderate	hard	easy	moderate	hard
SECOND (baseline)	90.97	79.94	77.09	95.61	89.54	86.96
SECOND+RRC	92.69	82.69	78.20	96.53	92.78	87.74
SECOND+MSCNN	92.37	82.36	78.23	96.34	92.59	87.81
SECOND+C-RCNN*	92.35	82.73	78.10	96.34	92.57	89.36
PointPillars (baseline)	87.37	76.17	72.88	92.40	87.79	86.39
PointPillars+RRC	88.48	78.50	75.13	93.53	88.87	87.09
PointPillars+MSCNN	89.22	77.05	73.16	92.80	88.46	87.26
PointPillars+C-RCNN*	89.95	78.99	73.27	93.77	88.27	87.34
PointRCNN (baseline)	92.54	82.16	77.88	95.58	88.78	86.34
PointRCNN+RRC	92.67	84.75	81.82	95.98	90.80	87.96
PointRCNN+MSCNN	92.64	83.26	79.88	95.60	90.05	87.05
PointRCNN+C-RCNN*	93.09	84.09	80.73	96.13	90.19	87.26
PV-RCNN (baseline)	92.10	84.36	82.48	93.02	90.33	88.53
PV-RCNN+RRC	92.82	85.59	83.00	93.65	92.40	90.19
PV-RCNN+MSCNN	92.66	83.89	83.29	93.50	91.63	89.42
PV-RCNN+C-RCNN*	92.78	85.94	83.25	93.48	91.98	89.48

Table 3.2: 3D and bird's eye view performance of fusion with different combinations of 2D/3D detectors through CLOCs on car class of KITTI validation set (new 40 recall positions metric). *C-RCNN is Cascade R-CNN. Our CLOCs fusion methods outperform the baseline methods.

class pedestrian and cyclist; for 2D detectors, only MS-CNN [103] does. Therefore, we only show the evaluation results based on SECOND, PointPillars and MS-CNN. As shown in Table 3.3 and Table 3.4, our fusion method improves the detection performance by a large margin.

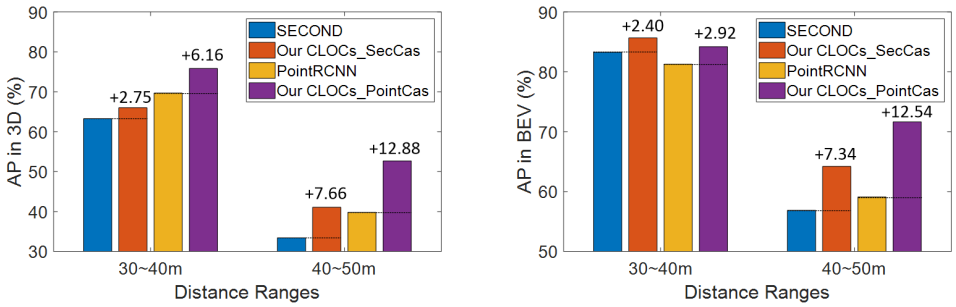
Detector	3D AP (%)			Bird's Eye View AP (%)		
	easy	moderate	hard	easy	moderate	hard
SECOND (baseline)	58.01	51.88	47.05	61.97	56.77	51.27
SECOND+MSCNN	62.54	56.76	52.26	69.35	63.47	58.93
PointPillars (baseline)	58.38	51.42	45.20	66.97	59.45	53.42
PointPillars+MSCNN	60.33	54.17	46.42	69.29	63.00	54.80

Table 3.3: 3D and bird's eye view performance of fusion on pedestrian class of KITTI validation set (using new 40 recall positions). Our CLOCs fusion methods outperform the corresponding baseline methods.

Detector	3D AP (%)			Bird's Eye View AP (%)		
	easy	moderate	hard	easy	moderate	hard
SECOND (baseline)	78.50	56.74	52.83	81.91	59.36	55.53
SECOND+MSCNN	85.47	59.47	55.00	88.96	63.40	59.81
PointPillars (baseline)	82.31	59.33	55.25	84.65	61.39	57.28
PointPillars+MSCNN	90.26	64.84	59.59	92.64	67.97	62.31

Table 3.4: Performance of fusion on cyclist class of KITTI validation set (new 40 recall positions). Our CLOCs fusion methods outperform the corresponding baseline methods.

Figure 3.7 shows the average precision (AP) on KITTI validation set in different distance ranges. The distance is defined as the Euclidean distance in xy plane between objects and LiDAR. The blue bars are the APs for SECOND detector, the orange bars represent APs for our CLOCs_SecCas. The yellow and purple bars show the APs of PointRCNN and CLOCs_PointCas respectively. As shown in Figure 3.7, APs for CLOCs is higher than the corresponding baselines in all distance ranges on both 3D and BEV detection benchmarks. The largest improvement is in $40 \sim 50m$. This is because the point clouds in long distance are too sparse for LiDAR-only detectors such as SECOND and PointRCNN, while CLOCs could utilize 2D detections to improve the performance.



(a) 3D detection. (b) Bird's eye view detection.

Figure 3.7: Average Precision (AP) based on distance. Our CLOCs outperforms the baseline by a large margin especially in long distance ($40 \sim 50m$).

Figure 4.6 shows some qualitative results of our proposed fusion method on the KITTI [11]

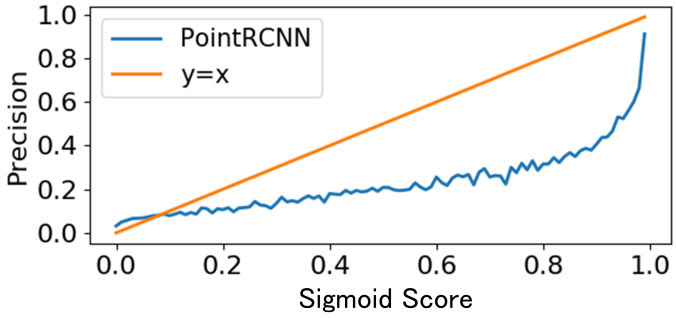


Figure 3.8: Precision and sigmoid score (predicted probability) from PointRCNN [9]. Blue curve represents the predicted probability from PointRCNN, the yellow line is the ideal situation in which the output probability equals precision. The sigmoid score does not reflect real precision.

test set. Red bounding boxes represent wrong detections (false positives) from SECOND that are deleted by our CLOCs, blue bounding boxes stand for missed detections from SECOND that are corrected by our CLOCs, green bounding boxes are correct detections.

3.6.4 Score Scales

There are two common output scores for detectors: the first is a real number approximating the log-likelihood ratio between target and clutter, and the second is a sigmoid transformation of this onto the range 0 to 1, so approximating a probability of target. We compare the use of these in CLOCs in Table 3.5 and find improved performance using the logarithmic likelihood score. The primary reason for the poor performance of the normalized score is that it poorly approximates a probability of the target (or precision), see Figure 3.8. Using this score forces the fusion network to learn a non-linear correction, whereas the equivalent log likelihood score discrepancy is a simple offset that can easily be corrected by the fusion layer. If we instead use a fitted sigmoid to obtain better probabilistic output from the PointRCNN, then fusion works equally well with either input. In general, we believe that it is simpler to use a log-likelihood output for each single-modality detector and fuse these.

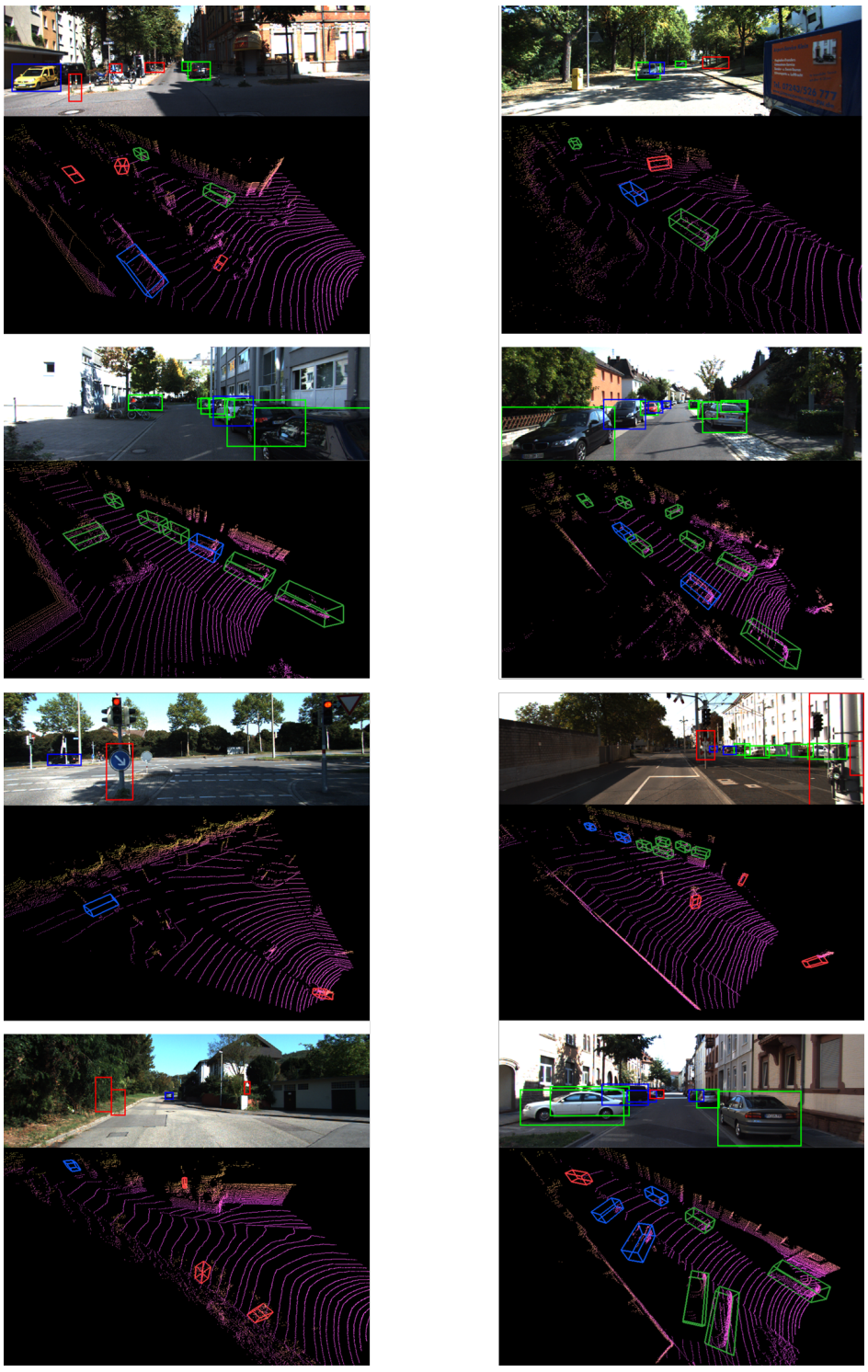


Figure 3.9: Qualitative results of our CLOCs on KITTI test set compared to SECOND [4]. Red and blue bounding boxes are false and missed detections from SECOND respectively that are corrected by our CLOCs. Green bounding boxes are correct detections. The upper row in each image is the 3D detection projected to the image; the others are 3D detections in LiDAR point clouds.

Type of Scores	3D AP (%)			Bird's Eye View AP (%)		
	easy	moderate	hard	easy	moderate	hard
log score	93.09	84.09	80.73	96.13	90.19	87.26
sigmoid score	91.64	82.96	79.13	95.33	89.70	86.36
corrected sigmoid score	92.83	83.73	80.12	95.88	90.19	87.08
corrected log score	92.88	83.92	80.22	96.07	89.93	87.21

Table 3.5: Performance of CLOCs with PointRCNN using different score scales on car class of KITTI validation set. Because the sigmoid score from PointRCNN poorly approximates probability of a target (or precision), using it for fusion could result in worse performance.

3.6.5 Ablation Study

We evaluate the contribution of each channel and focal loss in our fusion pipeline. The four channels include: IoU between 2D detections and projected 3D detections (IoU), 2D confidence score (s^{2D}), 3D confidence score (s^{3D}) and normalized distance (d) between the 3D bounding box and the LiDAR in the xy plane. The results are shown in Table 3.6.

IoU, as the measure of geometric consistency, is crucial to the fusion network. Without IoU, the association between 2D and 3D detections would be ambiguous and further lead to deterioration of performance. 2D confidence score indicates the certainty of 2D detections, which could provide useful clues for the fusion. 3D confidence score (s^{3D}) plays the most important role among the four channels, because CLOCs generates new confidence scores for all 3D detection candidates through fusion in which the original 3D scores are highly important evidences. Closer objects are usually easier to detect because there are more hits from LiDAR, the normalized distance (d) could be a useful indicator for this. Because there is a large imbalance between positives and negatives among detection candidates, focal loss could address this issue and improve detection accuracy.

IoU	s^{2D}	s^{3D}	d	focal loss	3D AP	BEV AP
					79.94	89.54
	✓	✓	✓	✓	78.95	88.43
✓		✓	✓	✓	80.96	90.32
✓	✓		✓	✓	38.64	47.16
✓	✓	✓		✓	81.96	91.90
✓	✓	✓	✓		81.01	92.17
✓	✓	✓	✓	✓	**82.73**	**92.57**

Table 3.6: The contribution of each channel and focal loss in our CLOCs fusion pipeline. The results are on the moderate level car class of KITTI *val* split with AP calculated by 40 recall positions. SECOND and Cascade R-CNN are fused in this experiment, so the baseline model is SECOND.

3.7 Summary of the Chapter

In this chapter, we propose Camera-LiDAR Object Candidates Fusion (CLOCs), as a fast and simple way to improve performance of just about any 2D and 3D object detector when both LiDAR and camera data are available. CLOCs exploits the geometric and semantic consistencies between 2D and 3D detections and automatically learns fusion parameters. The experiments show that our fusion method outperforms previous state-of-the-art methods by a large margin on the challenging 3D detection benchmark of KITTI dataset, especially in long-distance detection. As such, CLOCs provides a baseline for other types of fusion, including early and deep fusion.

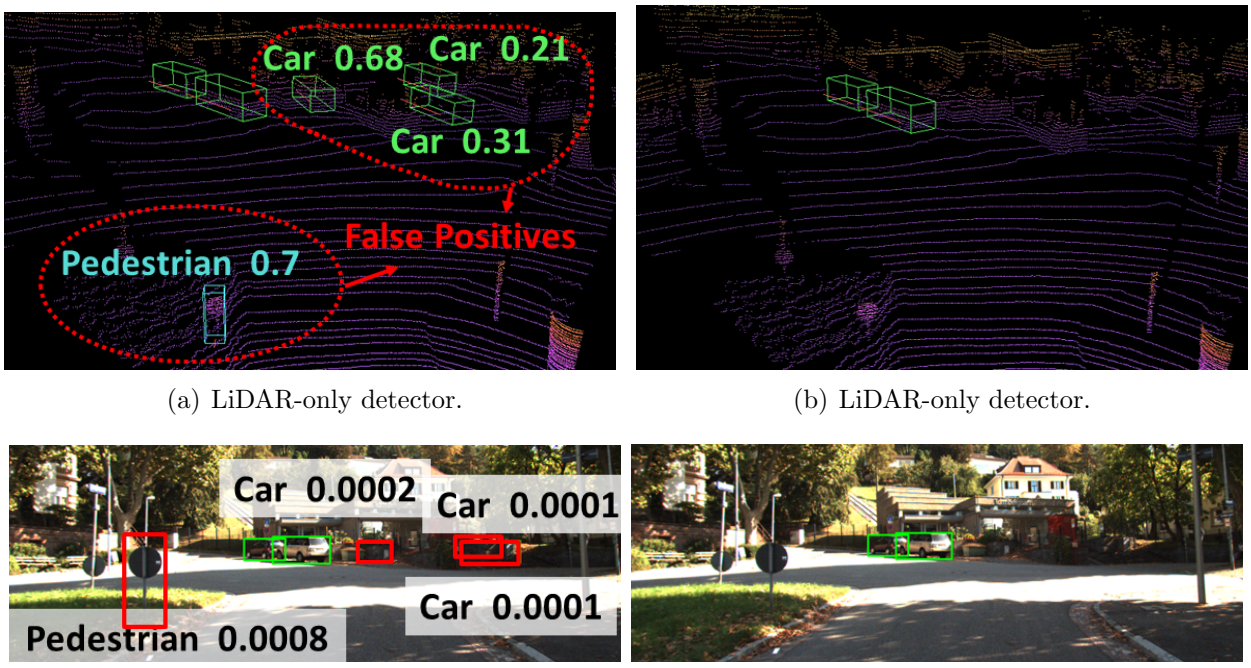
Chapter 4

Fast-CLOCs: Fast Camera-LiDAR Object Candidates Fusion for 3D Object Detection

4.1 Introduction

When compared to single modality approaches, fusion-based object detection methods often require more complex models to integrate heterogeneous sensor data and use more GPU memory and computational resources. This is particularly true for camera-LiDAR-based multimodal fusion, which may require three separate deep learning networks and/or processing pipelines that are designated for the visual data, LiDAR data, and for some form of a fusion framework.

CLOCs performs detection-level fusion using any pair of pre-trained 2D and 3D detectors to generate better 2D/3D detections. The CLOCs fusion step is fast and adds negligible delay to the perception system. More importantly, CLOCs provides significant improvements to the performance of the underlying 3D and 2D detection methods used. However, by requiring a 2D detector and a 3D detector to run simultaneously, CLOCS uses high GPU memory and compute power. We propose Fast-CLOCs: Fast Camera-LiDAR Object Can-



(a) LiDAR-only detector.

(b) LiDAR-only detector.

(c) LiDAR-only detector.

(d) LiDAR-only detector.

Figure 4.1: Fast-CLOCs can leverage the proposed 3D-Q-2D detector to remove false positive detections. (a) The LiDAR-only detector has four false positives. The numbers shown in the figure are LiDAR confidence scores. (c) The 3D-Q-2D detector suppresses these false positives by allocating them very small visual confidence scores. These LiDAR false positives are more easily removed with image appearance information. (b) and (d) show the proposed Fast-CLOCs fusion result with false positives removed.

didates fusion framework to achieve improved accuracy for 3D object detection with near real-time performance (Figure. 4.1). Unlike the original CLOCs, Fast-CLOCs eliminates a separate heavy 2D detector; and instead, uses a 3D detector-cued 2D image detector (3D-Q-2D) to reduce memory and computation. This proposed architecture delivers the following contributions:

- Fast-CLOCs uses any 3D detector without re-training.
- The proposed 3D-Q-2D image detector within Fast-CLOCs outperforms state-of-the-art (SOTA) image-based detectors in 2D object detection.
- Fast-CLOCs is significantly more memory and computationally efficient than SOTA

fusion methods, and can run in near real-time on a single desktop-level GPU.

- Fast-CLOCs improves the SOTA LiDAR-camera fusion performance on the KITTI and nuScenes datasets.

4.2 Proposed Method

A high-level diagram of the proposed Fast-CLOCs architecture is shown in Figure 4.2. This illustrates the three primary components of the system: (1) 3D object detector, (2) 3D-Q-2D image detector, and (3) CLOCs fusion. The details of the system will be described and illustrated in this section.

The 3D detector processes the 3D point cloud to generate 3D *candidates*. Here detection candidates are used *before* Non-Maximum Suppression (NMS) because many correct detections are mistakenly suppressed during NMS [1]. A lightweight 3D-Q-2D image detector is proposed to generate high-accuracy 2D detections for fusion. This proposed detector uses the projected 3D candidates as its region proposals and refines them to predict its own 2D detections. Then, CLOCs [1] fusion is applied to fuse the 3D and 2D candidates and produce more accurate 3D detection results.

4.2.1 3D-Q-2D Image Detector

4.2.1.1 The Input Data

We propose to incorporate a 3D-Q-2D image detector instead of a separate complete 2D image detector used by the original CLOCs to significantly reduce GPU memory and computational cost. As shown in Figure 4.3, for traditional RCNN-style multi-stage 2D image

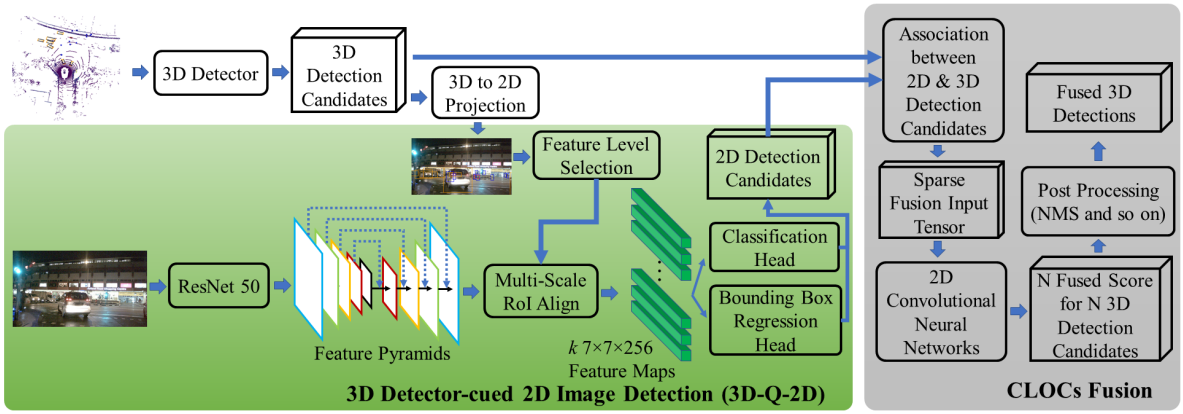


Figure 4.2: Fast-CLOCs system architecture. There are three primary components of the system: (1) 3D object detector; (2) 3D detector-cued 2D image detector (3D-Q-2D); (3) the original CLOCs fusion [1]. Compared to original CLOCs, we propose to leverage a lightweight 3D-Q-2D detector (green block), instead of a separate complete 2D image detector, to reduce the GPU memory and computational cost.

detectors [13, 105, 3], the first stage is designed to generate foreground region proposals with high recall rate, then the following stages are applied for further classification and bounding box refinement. The quality of the proposals is of paramount importance for the final detection output. The 3D detection candidates from the 3D detectors are of high quality and high recall rate; therefore, the projected 3D detection candidates can be leveraged as the 2D proposals for the 2D image detector with no extra costs. We project all 3D detection candidates into the image plane using the calibration parameters between LiDAR and camera, and these constitute *cues* for our 3D-Q-2D image detector.

Note that there are 8 corner points for each 3D bounding box. After projecting them into the image plane, we use the maximum and minimum xy pixel coordinates to build an axis-aligned 2D bounding box. In this paper, the "projected 3D detection candidate" refers to the corresponding axis-aligned 2D bounding box in the image plane.

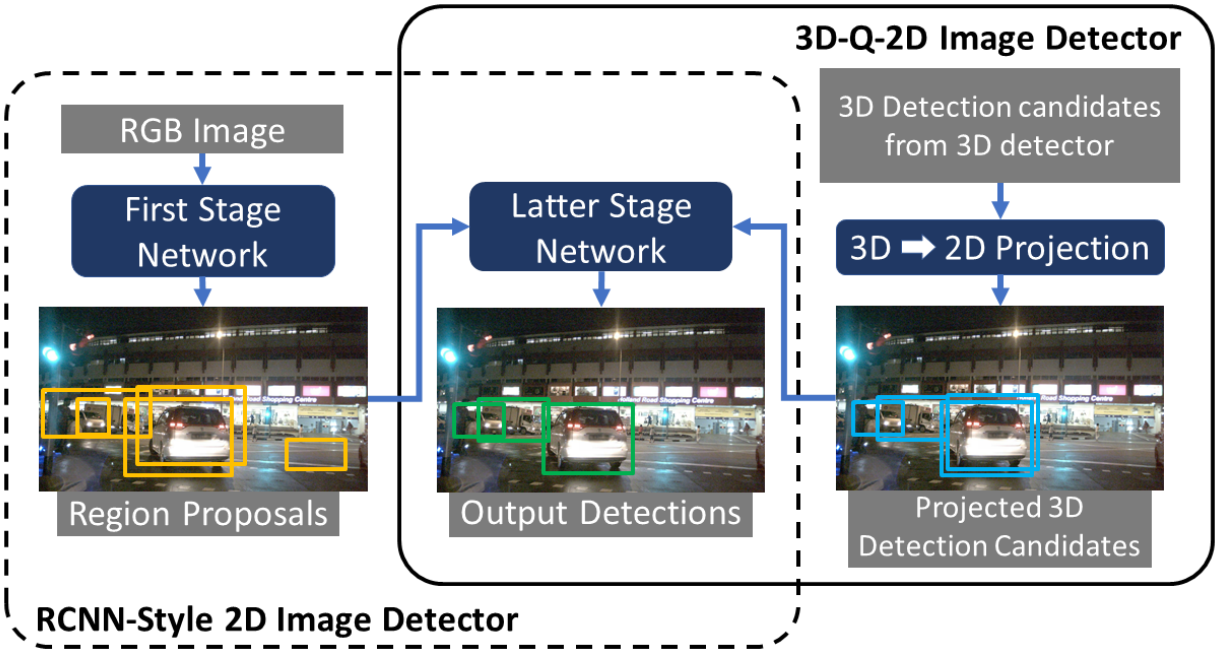


Figure 4.3: Architecture comparison of our 3D-Q-2D image detector and RCNN-style 2D image detector. The 3D detection candidates from the 3D detectors can be leveraged as the 2D proposals for the 2D detector at no extra cost. Therefore, there is no need to have a computationally expensive first-stage network in our proposed 2D detector.

4.2.1.2 The Backbone Feature Extraction Network

We use ResNet-50 with Feature Pyramid Network (FPN) [106, 15] pretrained in COCO [107] as the backbone. The FPN produces a multi-scale feature pyramid in which all levels are semantically strong, improving performance compared to producing the single-scale output feature map from the backbone. The feature level selection module assigns the level of a proposal based on its size in the original image. Small-sized proposals will be assigned to low-level, high-resolution feature maps. The corresponding regions of interest (RoI) in the feature map are calculated based on the downsample rate of that feature map (*e.g.,* 4, 8, 16). Multi-scale RoI align [5] is applied to extract features within the RoI in the feature maps. Then, the extracted feature maps are converted into a fixed spatial extent of $H \times W$ (*e.g.,* 7×7). Bilinear interpolation is applied to avoid quantization of the RoI boundaries or bins.

4.2.1.3 Detection Head

A bounding box regression head and a classification head are attached to the k output RoI feature maps, where k is the number of projected 3D detection candidates. The 2D bounding box is parameterized using the top left and bottom right pixel coordinates. The bounding box prediction and training parameterizations are defined as follows:

$$w^{pro} = x_2^{pro} - x_1^{pro}, \quad h^{pro} = y_2^{pro} - y_1^{pro} \tag{4.1}$$

$$t_{x1} = (x_1 - x_1^{pro})/w^{pro}, \quad t_{x2} = (x_2 - x_2^{pro})/w^{pro} \tag{4.2}$$

$$t_{y1} = (y_1 - y_1^{pro})/h^{pro}, \quad t_{y2} = (y_2 - y_2^{pro})/h^{pro} \tag{4.3}$$

$$t_{x1}^* = (x_1^* - x_1^{pro})/w^{pro}, \quad t_{x2}^* = (x_2^* - x_2^{pro})/w^{pro} \tag{4.4}$$

$$t_{y1}^* = (y_1^* - y_1^{pro})/h^{pro}, \quad t_{y2}^* = (y_2^* - y_2^{pro})/h^{pro} \tag{4.5}$$

where variables (x_1, y_1) and (x_2, y_2) are the top left and bottom right pixel coordinates of the 2D boundary box, with height h, and width w. Variables x, x^{pro} and x^* are for the predicted 2D box, the projected 2D box (from 3D detector) and the ground-truth 2D box, respectively.

The classification head maps the RoI features into softmax probabilities using $(n + 1)$ classes, with n for objects and 1 for background. Here note that although we have softmax probabilities for each class, the output class of the 2D image detector is still determined by the 3D detector. This is because for the fusion of a 3D detection with class #A, we ask from the image detecor "how likely is it that there is an object with class #A in the projected region?", rather than "what is the class of the object within the projected region?". The latter is more challenging, requires a more complicated network, and is not needed for our

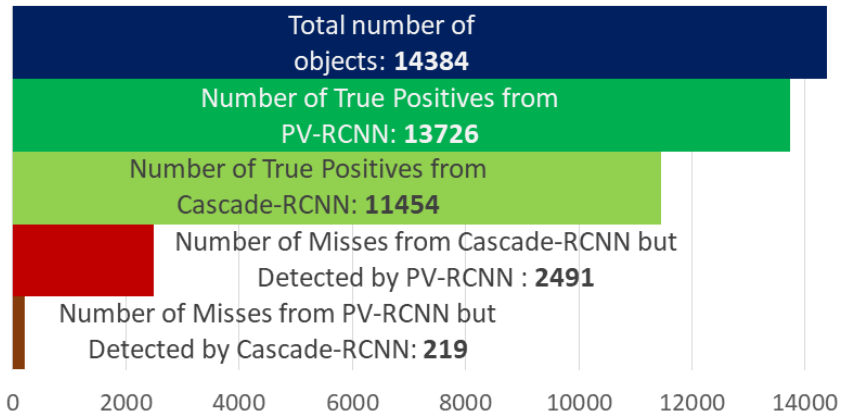


Figure 4.4: KITTI validation car detection comparison of true positives and misses from a 2D detector (Cascade-RCNN [3]) and 3D detector (PV-RCNN [10]). Only 219 objects are missed by 3D detector but detected by 2D detector, while 2491 are missed by 2D detector but detected by 3D detector. This shows that 3D detector has higher recall rate than 2D detector, and validates using 3D-Q-2D replacing an independent 2D image detector in fusion.

fusion task.

4.2.1.4 Multi-Task Loss and Training

We have two loss functions for the proposed 2D detector, namely smooth L1 loss for bounding box regression, and focal loss [16] for classification to address class imbalances between targets and backgrounds. During training, the intersection of union (IoU) between projected 3D detection candidates (2D proposals) and the 2D ground-truths are calculated. Proposals with IoU greater than θ_{up} are labeled positive, and those with IoU less than θ_{low} are labeled negative examples. Different classes have different IoU thresholds. Only positive examples are included in the bounding box regression loss, while both positive and negative examples contribute to the classification loss. The final loss is the weighted sum of the two losses, following [108] to use uncertainty to weigh the two losses during training.

The proposed network is trained using stochastic gradient descent (SGD). We use the Adam optimizer with a maximum learning rate of $1 * 10^{-4}$, weight decay 0.01, and momen-

Method	Barrier IoU=0.5	T.C. IoU=0.5	Bicycle IoU=0.5	Motor IoU=0.5	Ped IoU=0.5	Car IoU=0.7	Bus IoU=0.7	C.V. IoU=0.7	Trailer IoU=0.7	Truck IoU=0.7
CenterPoint [109]	71.06	56.05	35.78	55.89	**74.76**	63.64	61.58	4.49	26.67	55.43
CenterPoint +3D-Q-2D	**72.60**	**59.63**	**56.22**	**65.25**	72.99	**76.25**	**73.08**	**15.94**	**34.26**	**56.84**

Table 4.1: 2D detection performance on nuScenes validation set. Since nuScenes does not provide 2D evaluation metrics, we apply the KITTI 2D evaluation metrics here. All 2D detections are evaluated by Average Precision (AP) with different 2D IoU thresholds for different classes. 'T.C.', 'Motor', 'Ped' and 'C.V.' are short for traffic cone, motorcycle, pedestrian and construction vechile, respectively. CenterPoint is a 3D detector that here we evaluate on its 2D performance. By using CeterPoint to cue a 2D detector, we can obtain much improved 2D performance, which is better for fusion.

tum 0.85 to 0.95 for 20 epochs.

4.2.1.5 CLOCs Fusion Network

CLOCs fusion is taken mainly from Chapter 3. We add another $flag$ channel to highlight whether a 3D detection overlaps with at least one 2D detection. There are two reasons for this. First, we want to keep the 3D detection candidate that has no 2D detections overlap with it. Because, as shown in Section 4.3, the recall rate for the 3D LiDAR-based detector is higher than the 2D image-based detector. Second, the addition of this new channel helps the network distinguish this case from other examples with very small IoU and 2D confidence scores s^{2D}.

4.2.2 The Scalability of Fast-CLOCs

Fast-CLOCs is a detection level fusion approach with a 3D-Q-2D block that leverages 3D detections to cue the 2D detector. Thus, it is crucial that the 3D detector has a high recall rate, as detections it misses will also be missed by the 2D detector. Based on our statistics,

we believe this is not a problem because SOTA 3D-based detectors have far fewer misses than SOTA 2D detectors. Figure 4.4 shows the statistics of the KITTI dataset, where we compare the number of true positives and misses by the SOTA 2D and 3D detectors. The 3D detector misses only 219 objects that are detected by the 2D image detector, whereas the 2D detector misses 2491 objects detected by the 3D LiDAR detector. This high effective recall rate of the 3D detector validates the use of the 3D-Q-2D detector to replace an independent 2D image detector in fusion.

4.3 Experimental Results

We evaluate our Fast-CLOCs on the challenging KITTI [11] and nuScenes [2] object detection benchmarks. We implement Fast-CLOCs using three 3D detectors: SECOND [4], PV-RCNN [10] and CenterPoint [109], termed Fast-CLOCs-SEC, Fast-CLOCs-PV and Fast-CLOCs-CP respectively.

4.3.1 Evaluation Results

4.3.1.1 3D-Q-2D Image Detector:

Before evaluating our Fast-CLOCs in 3D object detection, we first evaluate the proposed 3D-Q-2D image detector. For Fast-CLOCs, the quality of the input 2D detection candidates is of paramount importance. We use SECOND [4] and PV-RCNN [10] as cueing 3D detectors for KITTI dataset, termed as SECOND+3D-Q-2D and PV-RCNN+3D-Q-2D respectively. CenterPoint is implemented for the nuScenes dataset and is termed CenterPoint+3D-Q-2D.

Table 4.1 shows the 2D detection performance of our CenterPoint+3D-Q-2D in nuScenes validation set. Since nuScenes does not provide 2D evaluation metrics, we apply the KITTI

Detector	Input Data	2D AP (%)		
		easy	moderate	hard
MS-CNN [103]	Img	90.83	89.88	79.16
Cascade-RCNN [3]	Img	91.35	90.59	80.64
SECOND [4]	LiDAR	97.87	92.37	89.87
SECOND+3D-Q-2D	LiDAR+Img	98.75	95.56	90.14
PV-RCNN [10]	LiDAR	98.26	94.42	89.24
PV-RCNN+3D-Q-2D	LiDAR+Img	98.52	95.08	89.48

Table 4.2: Comparison of 2D detection performance with SOTA 2D image-based detectors and corresponding 3D detectors on car class in KITTI validation set.

2D evaluation metrics here. 3D detections from CenterPoint [109] are projected onto the image plane and evaluated using 2D detection metrics for comparison. All 2D detections are evaluated by Average Precision (AP) with different IoU thresholds for different classes. As shown in Table 4.1, compared to CenterPoint, our CenterPoint+3D-Q-2D has much better performance for all classes except for pedestrian. The main reason for this we think is that for nuScenes dataset, there are some driving sequences from rainy weather and night time. Detecting pedestrians in these scenarios from camera images is more challenging than using LiDAR. Table 4.2 compares 2D detection performance with SOTA 2D image detectors and corresponding 3D detectors on car class in KITTI validation set. This shows that our 3D-Q-2D image detector outperforms the SOTA 2D image detectors by a large margin.

4.3.1.2 Main Results

We present our 3D detection results on the KITTI test set in Table 4.3, and results on the nuScenes test set in Table 4.4. In the KITTI test set, our Fast-CLOCs-PV outperforms most of the SOTA fusion-based methods in all three classes. Note that the official open-source code of PV-RCNN performs slightly worse than the private one owned by the PV-RCNN authors shown on the KITTI leaderboard, and our Fast-CLOCs-PV result is based on the

Detector	S*	3D AP (%) Car / Pedestrian / Cyclist		
		easy	moderate	hard
PV-RCNN (baseline) [10]	L	87.45 / 47.30 / **77.33**	80.28 / 39.42 / **62.02**	76.21 / 36.97 / **55.52**
Fast-CLOCs-PV (Ours)	F	89.11 / **52.10** / **82.83**	**80.34** / **42.72** / **65.31**	**76.98** / **39.08** / **57.43**
Improvement (Fast-CLOCs-PV over PV-RCNN)	-	+1.66 / +4.8 / +5.5	+0.06 / +3.3 / +3.29	+0.77 / +2.11 / +1.91
CLOCs_PVCas [1]	F	88.94 / — / —	**80.67** / — / —	**77.15** / — / —
CLOCs_SecCas [1]	F	86.38 / — / —	78.45 / — / —	72.45 / — / —
F-PointNet [71]	F	82.19 / **50.53** / 72.27	69.79 / **42.15** / 56.12	60.59 / **38.08** / 49.01
AVOD [75]	F	76.39 / 36.10 / 57.19	66.47 / 27.86 / 42.08	60.23 / 25.76 / 38.29
EPNet [80]	F	**89.81** / — / —	79.28 / — / —	74.59 / — / —
UberATG-MMF [76]	F	88.40 / — / —	77.43 / — / —	70.22 / — / —
UberATG-ContFuse [77]	F	83.68 / — / —	68.78 / — / —	61.67 / — / —
PI-RCNN [104]	F	84.37 / — / —	74.82 / — / —	70.03 / — / —
3D-CVF [78]	F	**89.20** / — / —	80.05 / — / —	73.11 / — / —

Table 4.3: Performance comparison of 3D object detection with SOTA camera-LiDAR fusion methods on all classes of KITTI test set. *In S column, L represents LiDAR-only. F stands for camera-LiDAR fusion-based approach. Average Precision (AP) is the evaluation metric, with the best in green and the second-best in blue. Fast-CLOCs fusion outperforms other SOTA fusion-based detectors in most measures, and has the same level of performance compared to CLOCs_PVCas [1]. While CLOCs_PVCas requires significantly more computation (shown in Table. 4.5).

open-source PV-RCNN. The baseline PV-RCNN in Table 4.3 refers to the open-source PV-RCNN. Compared to baseline PV-RCNN, our Fast-CLOCs-PV increases the performance in 3D object detection by a large margin. Only CLOCs_PVCas has slightly better performance than ours (less than 0.4% AP) in car class. But CLOCs_PVCas requires running PV-RCNN and Cascade-RCNN simultaneously, which cannot be deployed on a single desktop-level GPU platform, while Fast-CLOCs-PV can. 3D-CVF's performance is close to ours in easy and moderate level for car class, but our Fast-CLOCs-PV outperforms 3D-CVF in hard level by a large margin (nearly 4% AP better). In nuScenes test set, our Fast-CLOCs-CP outperforms other published SOTA methods in both mean Average Precision (mAP) and nuScenes Detection Score (NDS). Our Fast-CLOCs-CP outperforms baseline method

Method	Setting*	mAP ↑	NDS ↑
CenterPoint [109]	L	58.0	65.5
Fast-CLOCs-CP (Ours)	F	**63.1**	**68.7**
Improvement (Fast-CLOCs-CP over CenterPoint)	-	+5.1	+3.2
PointPainting [79]	F	46.4	58.1
3D-CVF [78]	F	52.7	62.3
WYSIWYG [111]	L	35.0	41.9
PointPillars [19]	L	40.1	55.0
CVCNet [112]	L	55.3	64.4
PMPNet [113]	L	45.4	53.1
SSN [114]	L	46.3	56.9
CBGS [115]	L	52.8	63.3

Table 4.4: Performance comparison of 3D object detection with SOTA methods on nuScenes test set. *In Setting column, L represents LiDAR-only. F stands for camera-LiDAR fusion-based approach. We show the primary evaluation metrics nuScenes Detection Score (NDS) [2] and mean Average Precision (mAP).

CenterPoint by 5.1% and 3.2% in mAP and NDS.

Speed and Memory: For self-driving vehicles, computing resources are limited. It is challenging to install server-level multi-GPU system on the vehicle. Compared to other heavy fusion-based approaches, Fast-CLOCs can run in near real-time (8-13Hz) on a single standard desktop-level GPU. All our experiments are performed using NVIDIA RTX 3080 with 10 GB GPU memory. The running speed and GPU memory comparison results are shown in Table 4.5. The original CLOCs [1] and some other fusion-based approaches [79, 110] require running Cascade-RCNN [3] or Mask-RCNN [5] in addition to a 3D detector simultaneously. These methods above are slow and cannot be deployed on a single desktop-level GPU system, while Fast-CLOCs can and has better or same level of detection performance with faster speed.

Method	3D Detector			2D Detector			Fusion GPU Mem	Total	
	Name	Mem	Speed	Name	Mem	Speed		Mem	Speed
CLOCs_PVCas [1]	PV-RCNN	3-4GB	9.5Hz	C-RCNN	5+GB	4Hz	∼1GB	10+GB	<4Hz
CLOCs_SecCas [1]	SECOND	2.5-3GB	28Hz				∼1GB	10+GB	<4Hz
Fast-CLOCs-PV	PV-RCNN	3-4GB	9.5Hz	3D-Q-2D	1-1.5GB	18Hz	∼1GB	6.5GB	∼8Hz
Fast-CLOCs-CP	CenterPoint	3.5-4GB	10Hz				∼1GB	6.5GB	∼8Hz
Fast-CLOCs-SEC	SECOND	2.5-3GB	28Hz				∼1GB	5.5GB	∼13Hz

Table 4.5: Comparison of running speed and GPU memory usage between CLOCs [1] and Fast-CLOCs on RTX 3080 GPU. The 'Mem' stands for GPU memory. The original CLOCs [1] requires running Cascade-RCNN [3] in addition to a 3D detector simultaneously, so it cannot run all together on a single desktop-level GPU system, while Fast-CLOCs can and has better or the same level of detection performance. We cannot test running CLOCs_PVCas and CLOCs_SecCas on our RTX 3080 due to out of GPU memory issue, but since Cascade-RCNN can only run with around 4Hz, their final inference speeds would be definitely slower than 4Hz. While Fast-CLOCs is much faster than that.

4.3.1.3 Ablation Study

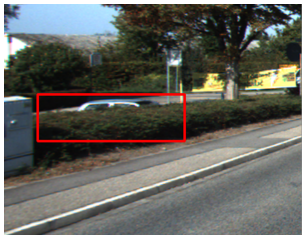
We evaluate the contributions of each component within the proposed 3D-Q-2D image detector, the results are shown in Table 4.6. RoI Align [5] applies bilinear interpolation to avoid quantization of RoI boundaries and further improve detection performance. FPN provides a multi-scale feature pyramids which are helpful for detecting small objects (more than 2% AP improvement in hard level). The box regression module refines the projected 3D detection candidates and generates better 2D bounding boxes. Ablation study on the modifications of the CLOCs fusion network is provided in the supplementary materials.

4.3.1.4 Qualitative Results

Figure 4.6 shows some qualitative results of our proposed fusion method on KITTI test set. Red bounding boxes represent false positive detections from PV-RCNN [10] that are deleted by our Fast-CLOCs, green bounding boxes are true positive detections that are confirmed by Fast-CLOCs.

focal loss	RoI Align	FPN	box regression	2D AP (%)		
				easy	moderate	hard
				95.97	90.04	86.52
✓				97.03	90.67	87.20
✓	✓			97.23	91.17	87.26
✓	✓	✓		97.83	92.66	89.54
✓	✓	✓	✓	**98.75**	**95.56**	**90.14**

Table 4.6: Ablation studies of different components in the proposed 3D-Q-2D image detector on KITTI validation set. SECOND [4] is applied to cue the 2D detector. RoI Align [5] applies bilinear interpolation to avoid quantization of the RoI boundaries or bins. Box regression represents the 2D bounding box regression head, without it the proposed 2D detector would output projected 3D boxes directly with visual confidence score.



(a) Fail Case#1: Fast-CLOCs removes a true positive from LiDAR 3D detector.

(b) Fail Case#2: Fast-CLOCs removes a true positive from LiDAR 3D detector.

(c) Fail Case#3: Fast-CLOCs fails to remove a false positive from LiDAR 3D detector.

(d) Fail Case#4: Fast-CLOCs fails to remove a false positive from LiDAR 3D detector.

Figure 4.5: Some failure cases. (a) and (b): LiDAR-only detector detects the true positives (car). But the cars are suppressed by the 3D-Q-2D detector due to high level of occlusion (case#1) and poor lighting conditions (case#2) in image plane. Therefore, Fast-CLOCs fusion removes these true positives. (c) and (d): LiDAR-only detector detects the cars but with wrong poses, so they are false positives. But these false positives are not rejected by the 3D-Q-2D detector because parts of the cars are visible in the image plane. Therefore, Fast-CLOCs fusion keeps these false positives.

Incorporating visual information from the image can help remove LiDAR false positive detections and confirm LiDAR true positive detections, as shown in Figure 4.6. The image projection region of a false positive detection in the LiDAR point cloud will be classified and usually rejected as a detection through our 3D-Q-2D detector. Then Fast-CLOCs can leverage this inconsistency to remove the false positive. Objects that are double confirmed by LiDAR detector and 3D-Q-2D image detector will be kept by Fast-CLOCs.

4.3.1.5 Failure Cases Analysis

There are mainly two types of failure cases for Fast-CLOCs. One is mistakenly removing the true positives; the other is failing to delete false positives. These failure cases occur in scenarios in which objects are heavily occluded, at long distance, or under poor lighting conditions. Figure 4.5 shows some failure examples. In fail case#1 and case#2, LiDAR-only detector detects the true positive cars. But the true positives are suppressed by our 3D-Q-2D detector due to high level of occlusion (case#1) and poor lighting condition (case#2) in the image. So Fast-CLOCs fusion removes these true positives. In fail case#3 and case#4, LiDAR-only detector detects the cars but with wrong poses, so they are false positive detections. But these false positive detections are not rejected by our 3D-Q-2D detector. Because parts of the cars are visible in the image plane, the 3D-Q-2D confirms them as cars but fails to provide the full sizes of the cars in the image plane. Fast-CLOCs fusion therefore keeps these false positive.

Detecting objects under occlusion is an open problem in the computer vision community. Designing better detection networks and collecting more training data with these corner cases could be one direction to resolve this issue. But it would require more computing resources. We believe that adding temporal information from multiple frames would be a

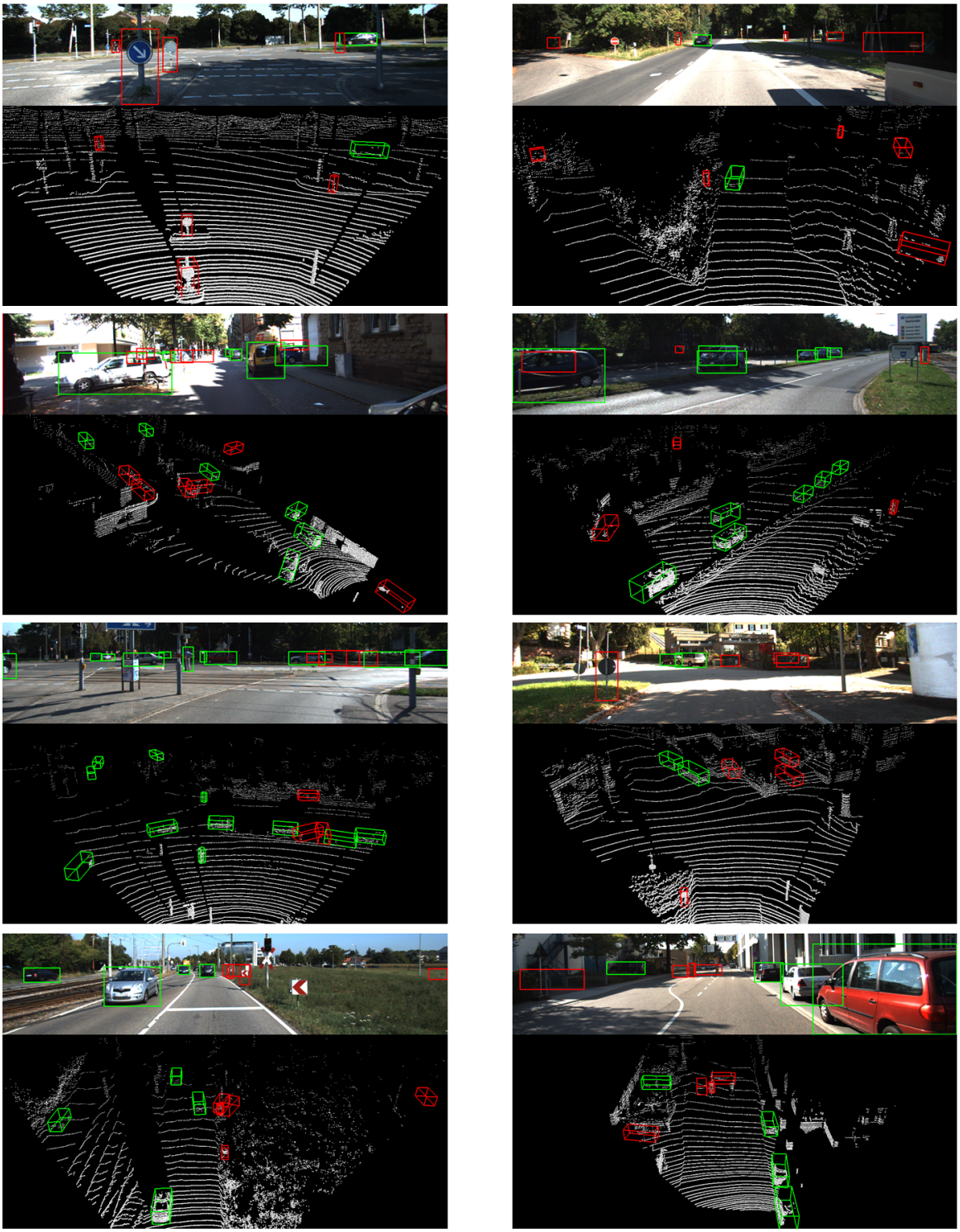


Figure 4.6: Qualitative results of our Fast-CLOCs on KITTI [11] test set compared to PV-RCNN [10]. Red bounding boxes are false positive detections from PV-RCNN that are removed by our Fast-CLOCs. Green bounding boxes are confirmed true positive detections. The upper row in each image is the 3D detection projected to the image; the others are 3D detections in LiDAR point clouds.

simpler direction.

4.4 Summary of the Chapter

In this chapter, we present Fast Camera-LiDAR Object Candidates (Fast-CLOCs) fusion that can run high-accuracy 3D object detection in near real-time. Fast-CLOCs introduces a lightweight 3D-Q-2D image detector to extract visual features from the image domain to improve 3D detections significantly. Compared to other separate fusion-based approaches that run independent 2D and 3D detectors simultaneously, Fast-CLOCs requires much less GPU memory and operates in real time on a single desktop-level GPU. At the same time, Fast-CLOCs achieves top or second-to-top performance in most categories compared to other fusion methods on KITTI and nuScenes datasets.

Chapter 5

TransCAR: Transformer-based Camera-And-Radar Fusion for 3D Object Detection

5.1 Introduction

Radars have been used for Advanced Driving Assistance System (ADAS) for many years. However, despite radar's popularity in the automotive industry, when considering 3D object detection most existing works focus on LiDAR [17, 4, 9, 19, 71, 65, 10, 109], camera [8, 12, 22, 21] and LiDAR-camera fusion [74, 75, 71, 77, 73, 80, 78, 104, 76, 1, 116]. One reason for this is that there are not as many open datasets annotated with 3D bounding boxes that include radar data [2, 38, 11, 7]. Another reason is that, compared to LiDAR point clouds, automotive radar signals are much sparser and lack height information. These properties make it challenging to distinguish between returns from objects of interest and backgrounds. However, radar has its strengths compared to LiDAR: (1) radar is robust under adverse weather and light conditions; (2) radar can accurately measure object's radial velocity through the Doppler effect without requiring temporal information from multiple frames; (3) radar has much lower cost compared to LiDAR. Therefore, we believe there is a

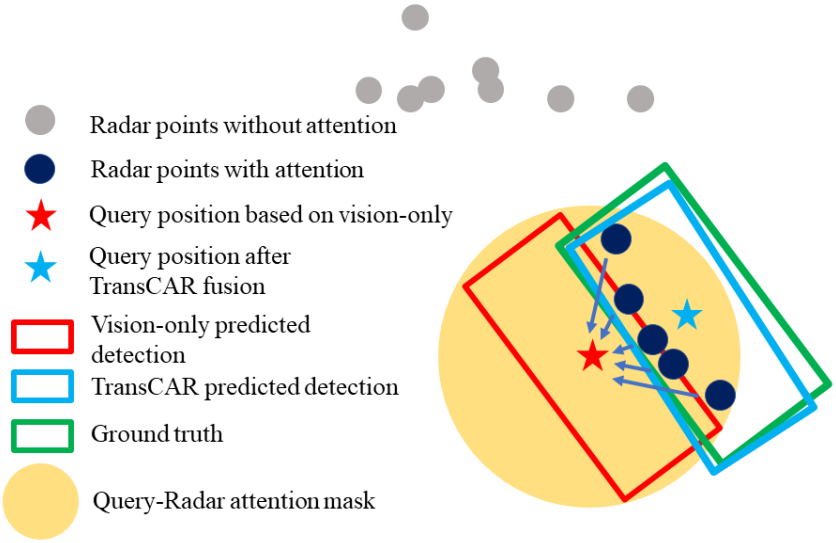


Figure 5.1: An illustrative example showing how TransCAR fusion works. Vision-only detection has significant range error. Our TransCAR fusion can learn the interactions between vision-based query and related radar signals and predict improved detection. Unrelated radar points are prevented from attention by Query-Radar attention mask.

gap and possibility for gain in radar-camera fusion research.

3D object detection is essential for self-driving and ADAS systems The goal of 3D object detection is to predict a set of 3D bounding boxes and category labels for objects of interest. It is challenging to directly estimate and classify 3D bounding boxes from automotive radar data alone due to its sparsity and lack of height information. Monocular camera-based 3D detectors [20, 8, 21, 22, 12] can classify objects, predict heading angles and azimuth angles of objects accurately. However, the errors in depth estimation are significant because regressing depth from a single image is inherently an ill-posed inverse problem. Radar can provide accurate depth measurement, which monocular camera-based solutions cannot. Camera can produce classification and 3D bounding box estimation that radar-based solutions cannot. Therefore, it is a natural idea to fuse radar and camera for better 3D object detection performance.

Data association between different sensor modalities is the main challenge for sensor

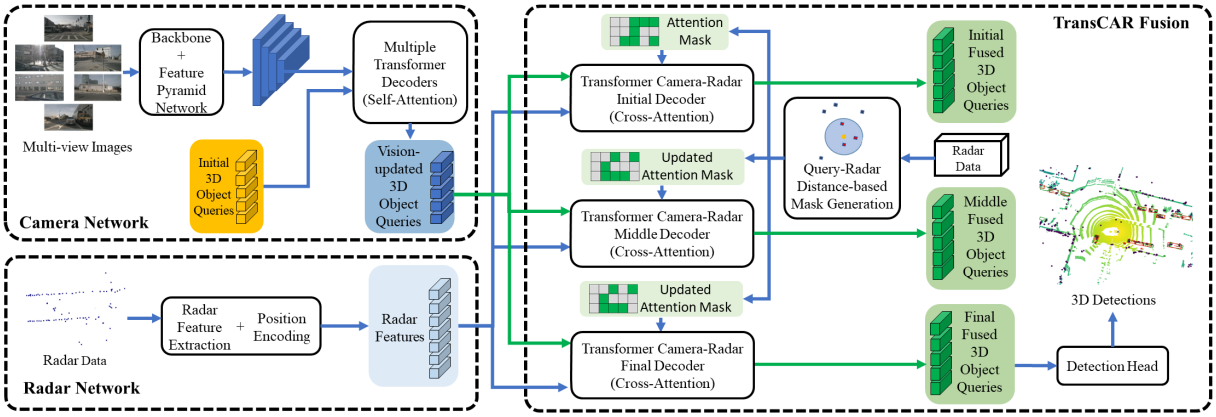


Figure 5.2: TransCAR system architecture. There are three primary components in the system: (1) A camera network (DETR3D[12]) based on transformer decoders to generate image-based 3D object queries. The initial object queries are generated randomly; (2) A radar network that encodes radar point locations and extracts radar features; (3) The TransCAR fusion module based on three transformer cross-attention decoders. We propose to use transformer to learn the interactions between radar features and vision-updated object queries for adaptive camera-radar association.

fusion technologies. Existing works mainly rely on multi-sensor calibration to do pixel-level [79], feature level [76, 77, 74, 75, 72] or detection level [1, 116] association. However, this is challenging for radar and camera association. First, the lack of height measurement in radar makes the radar-camera projection incorporate large uncertainties along the height direction. Second, radar beams are much wider than a typical image pixel and can bounce around. This can result in some hits visible to the radar but are occluded from the camera. Third, radar measurements are sparse and have low resolution. Many objects visible to the camera do not have radar hits. For these reasons, the hard-coded data association based on sensor calibration performs poorly for radar and camera fusion.

The seminal *Transformer* framework was initially proposed as a revolutionary technology for natural language processing (NLP) [117], and subsequently has shown its versatility in computer vision applications including object classification [118] and detection [119, 120]. The self-attention and cross-attention mechanism within the transformer can learn the inter-

actions between multiple sets of information [117, 119, 121]. And we believe that this makes transformer a viable fit to solve the data association in camera-radar fusion. In this dissertation, we propose a novel Transformer-based Radar and Camera fusion network termed TransCAR to address the problems mentioned above. Our TransCAR first uses DETR3D [12] to generate image-based object queries. Then TransCAR learns radar features from multiple accumulated radar scans and applies a transformer decoder to learn the interactions between radar features and vision-updated queries. The cross-attention within the transformer decoder can adaptively learn the soft-association between the radar features and vision-updated queries instead of hard-association based on sensor calibration only. Finally, our model predicts a bounding box per query using a set-to-set Hungarian loss. Figure 5.1 illustrates the main idea of TransCAR. We also add the velocity discrepancy as a metric for the Hungarian bipartite matching because radar can provide accurate radial velocity measurements. Although our focus is on fusing multiple monocular cameras and radars, the proposed TransCAR framework is applicable to stereo camera systems as well. We demonstrate our TransCAR using the challenging nuScenes dataset [2]. TransCAR outperforms all other state-of-the-art (SOTA) camera-radar fusion-based methods by a large margin. The proposed architecture delivers the following contributions:

- To the best of our knowledge, this represents a first successful attempt to employ transformer for the challenging task of camera and radar fusion.
- We propose a novel camera-radar fusion network that adaptively learns the soft-association, and we show superior 3D detection performance compared to hard-association based on radar-camera calibration.
- TransCAR improves the velocity estimation using radar without requiring temporal

information.

- As for now (July 2022). The proposed TransCAR ranks **1st** among camera-radar fusion-based methods on the nuScenes 3D detection benchmark.

5.2 TransCAR

A high-level diagram of the proposed TransCAR architecture is shown in Figure 5.2. The camera network first utilizes multi-view images to generate vision-updated object queries. The radar network encodes radar point locations and extract radar features. Then the TransCAR fusion module performs camera-radar fusion by attentively fusing vision-updated object queries with useful radar features. In the following, we present the details of each module in TransCAR.

5.2.1 Camera Network

Our camera network takes multi-view images collected by 6 cameras covering the full 360 degrees around the ego-vehicle and initial 3D object queries as input, and outputs a set of vision-updated 3D object queries in the 3D space. We apply DETR3D [12] to the camera network and follow the iterative top-down design. It utilizes initial 3D queries to index 2D features for refining 3D queries. The output 3D vision-updated queries are the input for the TransCAR fusion module.

5.2.1.1 Why Start from Camera

We use multi-view images to generate 3D object queries for fusion. Radar is not suitable for this task because many objects of interest do not have radar returns. There are mainly

Class	#Total	#Radar Misses	Radar Miss Rate	#LiDAR Misses	LiDAR Miss Rate
Car	318157	114697	36.05%	14196	4.46%
Truck	47687	12779	26.80%	1134	2.38%
Bus	7451	1521	20.41%	42	0.56%
Trailer	12604	2413	19.14%	413	3.28%
C.V.*	8655	2611	30.17%	166	1.92%
Ped.*	139493	109026	78.16%	473	0.34%
Motor.*	9209	5197	56.43%	196	2.13%
Bike	8171	5208	63.74%	111	1.36%
T.C.*	78532	54622	69.55%	1140	1.45%
Barrier	115105	81464	70.77%	1828	1.59%

Table 5.1: Statistics of objects in different classes in nuScenes training set within 50 meters of the ego vehicle. * 'C.V.', 'Ped', 'Motor' and 'T.C' represent construction vehicle, pedestrian, motorcycle and traffic cone, respectively. An object that is missed by radar or LiDAR is defined as having no hit/return from that object. Radar misses more objects. For the two most common classes in autonomous driving applications, car and pedestrian, radar misses 36.05% cars and 78.16% pedestrians. Although nuScenes does not provide detailed visibilities of objects in the image, we believe that it is much higher than radar. Therefore, we use camera instead of radar to generate 3D object queries for fusion.

two reasons behind this. First, automotive radar has a very limited vertical field of view compared to camera and LiDAR and is usually installed at a lower position. Therefore, any object that is located out of the radar's small vertical field of view will be missed. Second, unlike LiDAR, the radar beams are wider and the azimuth resolution is limited, making it difficult to detect small objects. According to our statistics in Table 5.1, in the nuScenes training set, radar has a high miss rate, especially for small objects. For the two most common classes on the road, car and pedestrian, radar misses 36.05% of cars and 78.16% of pedestrians. Cameras have much better object visibilities. Therefore, we utilize images to predict 3D object queries for fusion.

5.2.1.2 Methodology

The camera network uses ResNet-101 [106] with Feature Pyramid Network (FPN) [15] to learn a multi-scale feature pyramid. These multi-scale feature maps provide rich information

for detecting objects in different sizes. Following [120, 122], our camera network (DETR3D [12]) is iterative. It has 6 transformer decoder layers to produce vision-updated 3D object queries; each layer takes the output queries from the previous layer as input. The steps within each layer are explained below.

For the first decoder layer, a set of N ($N = 900$ for nuScenes) learnable 3D object queries $\boldsymbol{Q}^0 = \{\mathbf{q}_1^0, \mathbf{q}_2^0, ..., \mathbf{q}_N^0\} \in \mathbb{R}^C$ are initialized randomly within the 3D surveillance area. The superscript 0 represents this is the input query to the first layer, and the subscript is the index of the query. The network learns the distribution of these 3D query positions from the training data. For the following layers, the input queries are the output queries from the previous layer. Each 3D object query encodes a 3D center location $\mathbf{p}_i \in \mathbb{R}^3$ of a potential object. These 3D center points are projected to the image feature pyramid based on the camera extrinsic and intrinsic parameters to sample image features via bilinear interpolation. Assuming there are k layers in the image feature pyramid, the sampled image feature $\boldsymbol{f}_i \in \mathbb{R}^C$ for a 3D point $\mathbf{p}_i$ is the sum of sampled features across all k levels, C is the number of feature dimensions. A given 3D center point $\mathbf{p}_i$ may not be visible in any camera image. We pad the sampled image features corresponding to these out-of-view points with zeros.

A Transformer self-attention layer is used to learn the interactions among N 3D object queries and generate attention scores. The object queries are then combined with the sampled image features weighted by the attention scores to form the updated object queries $\boldsymbol{Q}^l = \{\mathbf{q}_1^l, \mathbf{q}_2^l, ..., \mathbf{q}_N^l\} \in \mathbb{R}^C$, where l is the current layer. $\boldsymbol{Q}^l$ is the input set of queries for the $(l+1)$-th layer.

For each updated object query $\mathbf{q}_i^l$, a 3D bounding box and a class label are predicted using two neural networks. The details of bounding box encoding and loss function are described in Section 5.2.4. A loss is computed after each layer during training. In inference

mode, only the vision-updated queries output from the last layer are used for fusion.

5.2.2 Radar Network

The radar network is designed to learn useful radar features and encode their 3D positions for fusion. We first filter radar points according to x and y range, since only objects within $+/-50$ meters box area in BEV are evaluated in nuScenes [2]. As radar is sparse, we accumulate radar from the previous 5 frames and transform them into the current frame. The nuScenes dataset provides 18 channels for each radar point, including the 3D location x, y, z in ego vehicle frame, radial velocities v_x and v_y, ego vehicle motion compensated velocities v_{xc} and v_{yc}, false alarm probability $pdh0$, a dynamic property channel $dynProp$ indicating whether or not the cluster is moving or stationary, and other state channels [1]. To make the state channels feasible for the network to learn, we transform them into one-hot vectors. Since we use 5 accumulated frames, the time offset of each frame with regard to the current timestamp is useful to indicate the position offset, so we also add a time offset channel for each point. With these pre-processing operations, each input radar point has 36 channels.

Multilayer perceptron (MLP) networks are used to learn radar features $\boldsymbol{F}_r \in \mathbb{R}^{M \times C}$ and radar point position encodings $\boldsymbol{P}_r \in \mathbb{R}^{M \times C}$, where M and C are the number of radar points and the number of feature dimensions, respectively. In this paper, we set $M = 1500$ and $C = 256$ for nuScenes dataset. Note that there are less than 1500 radar points for each timestep even after accumulation in nuScenes dataset. Therefore, we pad the empty spots with out-of-scope positions and zero features for dimension compatibility. Figure 5.3 shows

[1] A detailed explanation of each channel can be found at: https://github.com/nutonomy/nuscenes-devkit/blob/master/python-sdk/nuscenes/utils/data_classes.py.

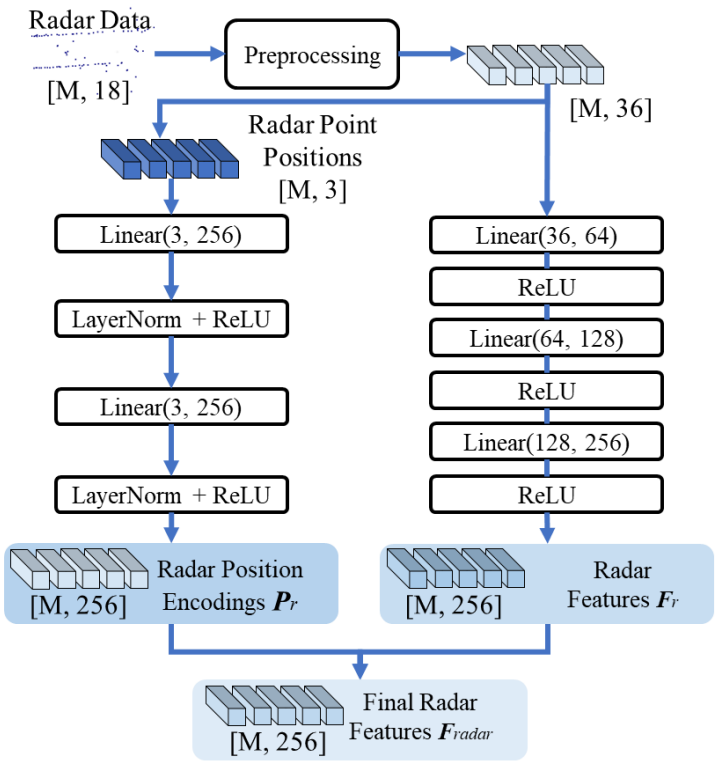


Figure 5.3: Details of radar network. The position encoding network (left) takes radar point positions (xyz) as input. The radar data after preprocessing (Section 5.2.2) are sent to the radar feature extraction network (right) to learn useful radar features. Since radar signal is very sparse, each radar point is treated independently. The numbers within the square brackets represent the shape of the data.

the details of the radar network. We combine the learned features and position encodings as the final radar features $\boldsymbol{F}_{radar} = (\boldsymbol{F}_r + \boldsymbol{P}_r) \in \mathbb{R}^{M \times C}$. These final radar features together with the vision-updated queries from the camera network are used for TransCAR fusion in the next step.

5.2.3 TransCAR Fusion

TransCAR fusion module takes vision-updated queries and radar features from previous steps as input, and outputs fused queries for 3D bounding box prediction. Three transformer decoders work in an iterative fashion in the TransCAR fusion module. The query-radar atten-

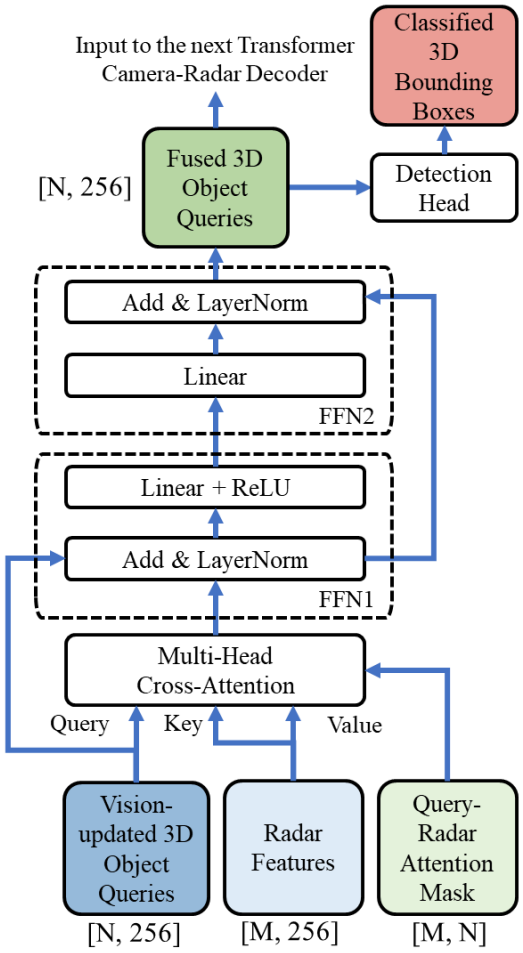


Figure 5.4: Details of transformer camera-radar decoder layer. The vision-updated 3D object queries are the queries to the multi-head cross attention module. The radar features are keys and values. See Section 5.2.3.2 for details. The numbers within the square brackets represent the shape of the data.

tion mask is proposed to assist the cross-attention layer in better learning the interactions and associations between vision-updated queries and radar features.

5.2.3.1 Query-Radar Attention Mask

It is challenging and time consuming to train a transformer if the number of input queries, keys and values are large [118, 119]. For our transformer decoder, there are N 3D object queries $\boldsymbol{Q} \in \mathbb{R}^{N \times C}$ and M radar features $\boldsymbol{F}_{rad} \in \mathbb{R}^{M \times C}$ as keys and values, where $N =$

900 and $M = 1500$ for nuScenes. It is not necessary to learn every pairwise interaction (900×1500) between them. For a query $\mathbf{q_i} \in \boldsymbol{Q}$, only the nearby radar features are useful. There is no need to interact $\mathbf{q}_i$ with other radar features that are far away. Therefore, we define a binary $N \times M$ Query-Radar attention mask $\boldsymbol{M} \in \{0,1\}^{N \times M}$ to prevent attention for certain positions, where 0 indicates no attention and 1 represents allowed attention. A position (i, j) in $\boldsymbol{M}$ is allowed for attention only when the xy Euclidean distance between the i-th query $\mathbf{q}_i$ and the j-th radar feature $\mathbf{f}_j$ is less than a threshold. There are three Query-Radar attention masks in TransCAR fusion corresponding to the three transformer decoders. The radius for these three masks are $2m$, $2m$ and $1m$, respectively.

5.2.3.2 Transformer Camera and Radar Cross-Attention

Three transformer cross-attention decoders are cascaded to learn the associations between vision-updated queries and radar features in our TransCAR fusion. Figure 5.4 shows the details of one transformer cross-attention decoder. For the initial decoder, the vision-updated queries $\boldsymbol{Q}_{img} \in \mathbb{R}^{N \times C}$ output from the camera network are the input queries. The radar features $\boldsymbol{F}_{rad} \in \mathbb{R}^{M \times C}$ are the input keys and values. The Query-Radar attention mask $\boldsymbol{M}_1$ is used to prevent attentions to certain unnecessary pairs. The cross-attention layer within the decoder will output an attention score matrix $\mathbf{A}_1 \in [0,1]^{N \times M}$. For the M elements in the i-th row of $\mathbf{A}_1$, they represent the attention scores between the i-th vision-updated query and all M radar features, and their sum is 1. Note that for each query, only radar features close to it are allowed for attention, so for each row in $\mathbf{A}_1$, most of them are zeros. These attention scores are indicators of associations between vision-updated queries and radar features. Then, the attention-weighted radar features for vision-updated queries are calculated as $\boldsymbol{F}^*_{rad1} = (\mathbf{A}_1 \cdot \boldsymbol{F}_{rad}) \in \mathbb{R}^{N \times C}$. $\boldsymbol{F}^*_{rad1}$. These weighted radar

features combined with the original vision-updated queries are then augmented by a feed-forward network (FFN) Φ_{FFN1}. This forms the fused queries for initial-stage: $\boldsymbol{Q}_{f1} = \Phi_{FFN1}(\boldsymbol{Q}_{img} + \boldsymbol{F}^*_{rad1}) \in \mathbb{R}^{N\times C}$.

The middle and final transformer decoders work similarly to the initial one. But they take the previous fused queries $\boldsymbol{Q}_{f1}$ instead of the vision-updated queries as input. Taking the middle query as an examole, the new Query-Radar attention mask $\boldsymbol{M}_2$ is calculated based on the distance between $\boldsymbol{Q}_{f1}$ and radar point positions. We also re-sample image features $\boldsymbol{f}_{f2}$ using encoded query positions in $\boldsymbol{Q}_{f1}$ as the query positions are updated in the initial decoder. Similarly to the initial decoder, the attention-weighted radar features for $\boldsymbol{Q}_{f1}$ are defined as $\boldsymbol{F}^*_{rad2} = (\mathbf{A}_2 \cdot \boldsymbol{F}_{rad}) \in \mathbb{R}^{N\times C}$, where $\mathbf{A}_2$ includes the attention scores for intial-stage fused query $\boldsymbol{Q}_{f1}$ and radar features $\boldsymbol{F}_{rad}$. The output fused queries are learned via $\boldsymbol{Q}_{f2} = \Phi_{FFN2}(\boldsymbol{Q}_{f1} + \boldsymbol{F}^*_{rad2} + \boldsymbol{f}_{f2}) \in \mathbb{R}^{N\times C}$. We apply two sets of FFNs after the two decoders to perform bounding box predictions. We compute losses from the two decoders during training, and only the bounding boxes output from the last decoder are used during inference.

Due to the sparsity of radar signals, for certain queries, there are no radar signals around. Therefore, these queries will not interact with any radar signals, and their attention scores are all zeros. Detections from these queries will be vision-based only.

5.2.4 Box Encoding and Loss Function

Box Encoding: We encode a 3D bounding box $\boldsymbol{b}_{3D}$ as an 11-digit vector:

$$\boldsymbol{b}_{3D} = [\boldsymbol{cls}, x, y, z, h, w, l, sin(\theta), cos(\theta), v_x, v_y] \tag{5.1}$$

where $\boldsymbol{cls} = \{c_1, ..., c_n\}$ is the class label, x, y and z are the 3D center location, h, w and l are the 3D dimension, θ is the heading angle, v_x and v_y are the velocities along the x and y axes. For each output object query $\mathbf{q}$, the network predicts its class scores $\boldsymbol{c} \in [0, 1]^n$ (n is the number of classes, $n = 10$ for nuScenes) and 3D bounding box parameters $\boldsymbol{b} \in \mathbb{R}^{10}$:

$$\boldsymbol{b} = [\Delta x, \Delta y, \Delta z, \log h, \log w, \log l, sin(\theta), cos(\theta), v_x, v_y] \tag{5.2}$$

where $\Delta x, \Delta y$ and Δz are the offsets between predictions and query positions from the previous layer.

Loss: We use a set-to-set Hungarian loss to guide training and measure the difference between network predictions and ground truths following [123, 119, 12]. There are two components in the loss function, one for classification and the other for bounding box regression. We apply focal loss [16] for classification to address the class imbalance, and $L1$ loss for bounding box regression. Assuming that N and K represent the number of predictions and ground truths in one frame, we pad ϕ (no object) with ground truths set since N is significantly larger than K. Following [123, 119, 12], we use Hungarian algorithm [124] to solve the bipartite matching problem between the predictions and ground truths:

$$\hat{\sigma} = \underset{\sigma \in \Theta}{arg\,min} \sum_{i}^{N} [-\mathbb{1}_{\{c_i \neq \phi\}} \hat{p}_{\sigma(i)}(c_i) + \mathbb{1}_{\{c_i \neq \phi\}} \mathcal{L}_{box}(\mathbf{b}_i, \hat{\mathbf{b}}_{\sigma(i)})] \tag{5.3}$$

where Θ denotes the set of permutations, $\hat{p}_{\sigma(i)}(c_i)$ represents the probability of class c_i with permutation index $\sigma(i)$, and $\mathcal{L}_{box}$ is the $L1$ difference for bounding boxes, $\mathbf{b}_i$ and $\hat{\mathbf{b}}_{\sigma(i)}$ are the ground truth box and predicted box respectively. Here, note that we also incorporate the velocity estimation v_x and v_y into $\mathcal{L}_{box}$ for a better match and velocity estimation. With

the optimal permutation $\hat{\sigma}$, the final Hungarian loss can be represented as follows:

$$\mathcal{L}_{Hungarian} = \sum_{i}^{N} [-\alpha(1 - \hat{p}_{\hat{\sigma}(i)}(c_i))^{\gamma} \log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \phi\}} \mathcal{L}_{box}(\mathbf{b}_i, \hat{\mathbf{b}}_{\hat{\sigma}(i)})] \tag{5.4}$$

where α and γ are the parameters of focal loss.

5.3 Experimental Results

We evaluate our TransCAR on the challenging nuScenes 3D detection benchmark [2] as it is the only open large-scale annotated dataset that includes radar.

5.3.1 Dataset

There are 6 cameras, 5 radars and 1 LiDAR installed on the nuScenes data collection vehicle. The nuScenes 3D detection dataset contains 1000 driving segments (scenes) of 20 seconds each, with 700, 150 and 150 segments for training, validation and testing, respectively. The annotation rate is 2Hz, so there are 28k, 6k and 6k annotated frames for training, validation and testing, respectively. There are 10 classes of objects. The true positive metric is based on BEV center distance.

5.3.2 Evaluation Results

We present our 3D detection results on the nuScenes test set in Table 5.2. Our TransCAR outperforms all other camera-radar fusion methods at the time of submission. Compared to the baseline camera-only method, DETR3D [12], TransCAR has higher mAP and NDS (nuScenes Detection Score [2]). As shown in Table 5.2, among the 10 classes, the car class

has the largest improvement (+1.8%). Cars and pedestrians are the main objects of interest in driving scenarios. In the nuScenes dataset, class Car has the highest proportion in the training set, it accounts for 43.74% of the total instances, and 63.95% of these car instances have radar hits. Therefore, these car examples provide sufficient training examples for our TransCAR to learn the fusion. Class Pedestrian has the second highest proportion in the training set, it accounts for 19.67% of the total instances, but only 21.84% have radar returns. TransCAR can still improve pedestrian mAP by 1.1%. This demonstrates that, for objects with radar hits, TransCAR can leverage the radar hits to improve the detection performance, and for objects without radar hits, TransCAR can preserve the baseline performance.

Table 5.3 shows the quantitative comparison with baseline DETR3D [12] in Car class with different center distance evaluation metrics. In nuScenes dataset, the true positive metric is based on the center distance, which means the center distance between a true positive and the ground truth should be smaller than the threshold. nuScenes defines four distance thresholds ranging from 0.5 to 4.0 meters. As shown in Table 5.3, TransCAR improves the AP for all 4 metrics. In particular, for the more strict and important metrics 0.5 and 1.0 meters thresholds, the improvement is 5.63% and 6.20% respectively. Figure 5.5 shows the qualitative results on the nuScenes dataset.

5.3.3 Ablation and Analysis

Contribution of each component: We evaluate the contribution of each component within our TransCAR network. The ablation study results on nuScenes validation set are shown in Table 5.4. The vision-only baseline is DETR3D [12]. Radial velocity is one of the unique measurements that radar can provide; although it is not true velocity, it can still guide the network to predict the object's velocity without temporal information. As shown

Method	S*	mAP↑	NDS↑	Car	Truck	Bus	Trailer	C.V.	Ped.	Motor.	Bike	T.C.	Barrier
MonoDIS[125]	C	30.4	38.4	47.8	22.0	18.8	17.6	7.4	37.0	29.0	24.5	48.7	51.1
CenterNet[126]	C	33.8	40.0	53.6	27.0	24.8	25.1	8.6	37.5	29.1	20.7	58.3	53.3
FCOS3D[127]	C	35.8	42.8	52.4	27.0	27.7	25.5	11.7	39.7	34.5	29.8	55.7	53.8
PGD[128]	C	38.6	44.8	56.1	29.9	28.5	26.6	13.4	44.1	39.7	31.4	60.5	56.1
DETR3D[12] (baseline)	C	41.2	47.9	60.3	33.3	29.0	35.8	17.0	45.5	41.3	30.8	62.7	56.5
PointPillar [126]	L	30.5	45.3	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9
infoFocus[129]	L	39.5	39.5	77.9	31.4	44.8	37.3	10.7	63.4	29.0	6.1	46.5	47.8
CenterFusion[86]	CR	32.6	44.9	50.9	25.8	23.4	23.5	7.7	37.0	31.4	20.1	57.5	48.4
TransCAR(Ours)	CR	**42.0**	**50.1**	**62.1**	**33.3**	**29.4**	**36.0**	**17.9**	**46.6**	**42.5**	**32.0**	**63.1**	**57.2**

Table 5.2: Quantitative comparison with SOTA methods on nuScenes test set. 'C.V.', 'Ped','Motor' and 'T.C' are short for construction vehicle, pedestrian, motorcycle, and traffic cone, respectively. TransCAR is currently the best camera-radar fusion-based method with the highest mAP and NDS, and it even outperforms early-released LiDAR-based approaches. The best performers are highlighted in bold green, excluding LiDAR-only solutions.

in the second row of Table 5.4, without radar radial velocity, the network can only use the location of radar points for fusion, and the mAVE (m/s) is significantly higher (0.906 vs. 0.546). The Query-Radar attention mask can prevent attentions for certain pairs of queries and radar features based on their distances. Without it, each query has to interact with all the radar features (1500 in our work) within the scene. This is challenging for the network to fuse useful radar features with the query, resulting in poorer performance.

The iterative refinement: There are three transformer cross-attention decoders that work iteratively in TransCAR. We study the effectiveness of the iterative design in TransCAR

Methods	AP Car@0.5m	AP Car@1.0m	AP Car@2.0m	AP Car@4.0m
Baseline(DETR3D)	16.72	46.28	71.55	83.96
TransCAR	22.35	52.48	74.52	84.53
Improvement	5.63	6.20	2.97	0.57

Table 5.3: Average Precision (AP) comparison with baseline DETR3D in Car class with different center-distance evaluation metrics on nuScenes validation set. Our TransCAR improves the AP by a large margin in all evaluation metrics.

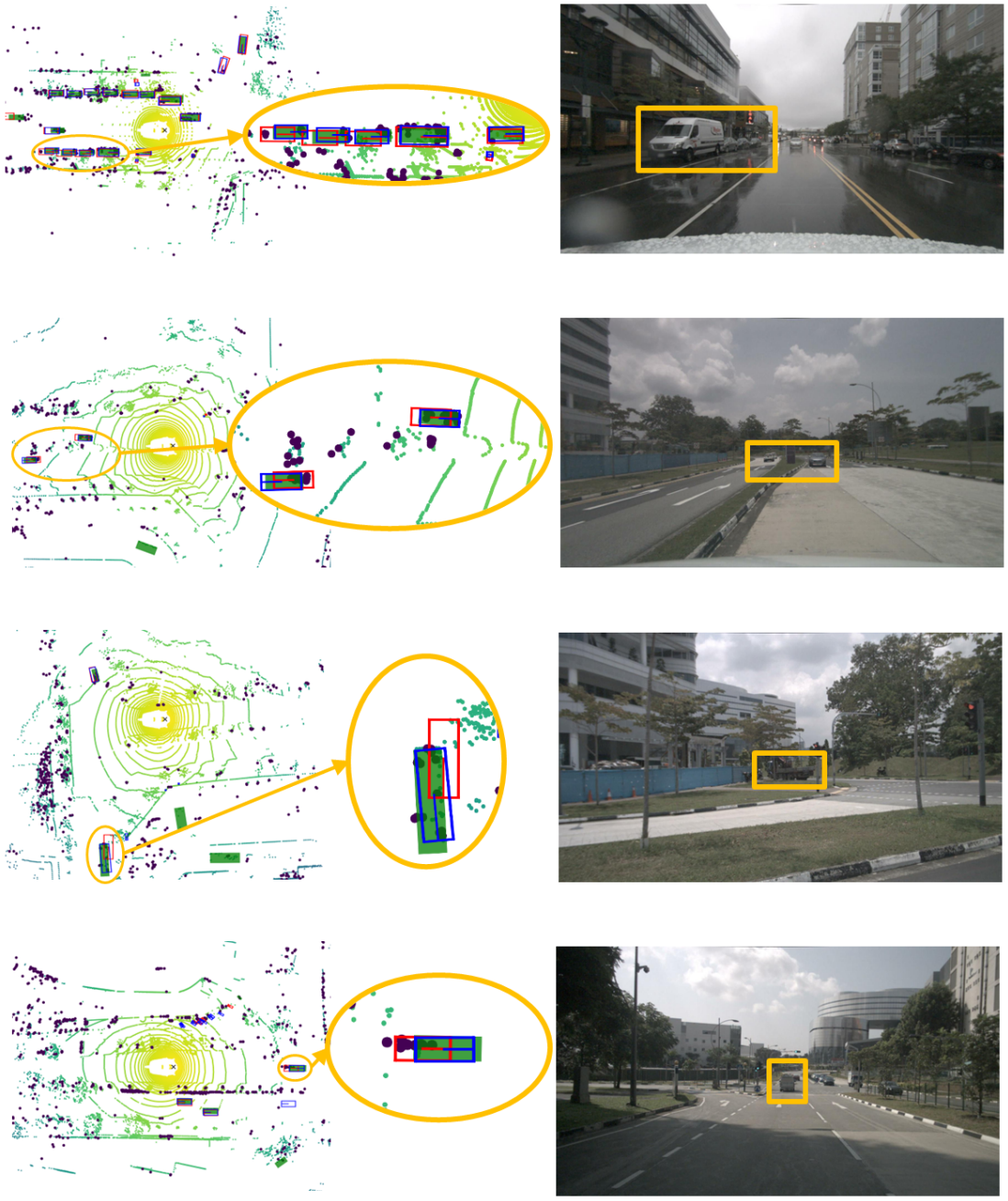


Figure 5.5: Qualitative comparison between TransCAR and baseline DETR3D on the nuScenes dataset. Blue and red boxes are the predictions from TransCAR and DETR3D respectively, green filled rectangles are ground truths. The larger dark points are radar points, smaller color points are LiDAR points for reference (color yallow to green indicates the increasing distance). The oval regions on the left column highlight the improvements made by TransCAR, the orange boxes on the image highlight the corresponding oval region in the top-down view. Best viewed with zoom-in and color.

Method	mAP↑	NDS↑	mATE↓	mAVE↓
Vision-only Baseline [12]	34.6	42.2	0.823	0.876
w/o radar feature extraction	34.7	41.7	0.766	0.906
w/o Query-Radar attention mask	34.4	39.5	0.765	1.125
TransCAR(Ours)	**35.5**	**46.2**	**0.756**	**0.546**

Table 5.4: Ablation of the proposed TransCAR components on nuScenes val set.

Number of Transformer Decoders in TransCAR	mAP↑	NDS↑	mATE↓	mAVE↓
0 (Baseline [12], without fusion)	34.6	42.2	0.823	0.876
1	34.9	43.4	0.763	0.768
2	35.4	45.4	0.758	0.625
3	**35.5**	**46.2**	**0.756**	**0.546**

Table 5.5: Evaluation on detection results from different number of transformer decoders in TransCAR.

fusion and present the results in Table 5.5. The quantitative results in Table 5.5 suggests that the iterative refinement in TransCAR fusion can improve the detection performance and is beneficial to fully leverage our proposed fusion architecture.

Performance in different distance ranges: Table 5.6 and Table 5.7 show the detection performance on nuScenes dataset in different distance ranges, Table 5.6 shows the average results for all the 10 classes, and Table 5.7 is for Car class only. The results from these

Method	All ≤20m			All in 20 - 30m			All in 30 - 40m			All in 40 - 50m		
	mAP↑	NDS↑	mAVE↓	mAP↑	NDS↑	mAVE↓	mAP↑	NDS↑	mAVE↓	mAP↑	NDS↑	mAVE↓
DETR3D [12]	47.9	48.9	0.996	25.8	38.4	0.722	11.0	23.6	0.997	0.9	10.0	1.098
Ours	48.6	54.0	0.537	26.6	41.9	0.450	11.9	27.4	0.705	1.0	10.8	0.938
Improvement	+0.7	+5.1	-0.459	+0.8	+3.5	-0.272	+0.9	+3.8	-0.292	+0.1	+0.8	-0.160

Table 5.6: Mean Average Precision (mAP, %), nuScenes Detection Score (NDS) and mean Average Velocity Error (AVE, m/s) for all classes of different distance ranges on nuScenes validation set. Our TransCAR outperforms the baseline (DETR3D) in all distance ranges.

two Tables suggest that the vision-only baseline method (DETR3D) and our TransCAR perform better in shorter distances. The improvements of TransCAR are more significant in the range of $20 - 40$ meters. This is mainly because for objects within 20 meters, the position errors are smaller, there are limited space for leveraging radar for improvement. And for objects beyond 40 meters, the baseline performs poorly, therefore TransCAR can only provided limited improvement. Note that the mean average precision (mAP) and corresponding improvements for all 10 classes in Table 5.6 are smaller than the ones for Car class in Table 5.7. There are mainly two reasons for this. First, mAP is the mean of APs of all classes, in nuScenes dataset, radar sensor has a higher miss rate for small-sized object classes (ped, cyclist, traffic cone, etc.). For example, 78.16% of pedestrians and 63.74% of cyclists do not have radar returns. Therefore, the performances for these classes are worse compared to large-sized objects (car, bus, etc.). Therefore, the improvements brought by TransCAR for these classes are limited, for those classes of objects that have radar returns, TransCAR can leverage the radar signal to improve the detection performance, for the ones that do not have radar returns, TransCAR can only preserve the baseline performance. Second, there is a significant class imbalance in the nuScenes dataset, class Car accounts for 43.74% of the training instances, while for some other classes, such as class Cyclist and class Motorcycle only accounts for 1.00% and 1.07% respectively. The training examples for these rare classes are not sufficient. As for the major class Car, which is also the most common objects in the driving scenarios, TransCAR can improve the detection performance and velocity estimation by a large margin (Table 5.7).

Different weather and lighting conditions: Radar is more robust under different weather and light conditions compared to cameras. We evaluate the detection performance under rainy conditions and during the night, the results are shown in Figure 5.6 and Figure

Method	Cars ≤20m Radar Miss 31.53%		Cars in 20 - 30m Radar Miss 73.29%		Cars in 30 - 40m Radar Miss 48.16%		Cars in 40 - 50m Radar Miss 50.45%	
	AP↑	AVE↓	AP↑	AVE↓	AP↑	AVE↓	AP↑	AVE↓
DETR3D [12]	76.4	0.917	48.7	0.810	28.8	0.934	5.0	1.015
TransCAR(Ours)	79.2	0.487	53.8	0.398	33.9	0.588	6.0	0.698
Improvement	+2.8	-0.430	+5.1	-0.412	+5.1	-0.346	+1.0	-0.317

Table 5.7: Average Precision (AP, %) and Average Velocity Error (AVE, m/s) for Car class of different distance ranges on nuScenes validation set. We also present the miss rate of radar sensor for different distance ranges. A car missed by radar is defined as a car that does not radar return. Our TransCAR improves the AP and reduces the velocity estimation error by a large margin in all distance ranges.

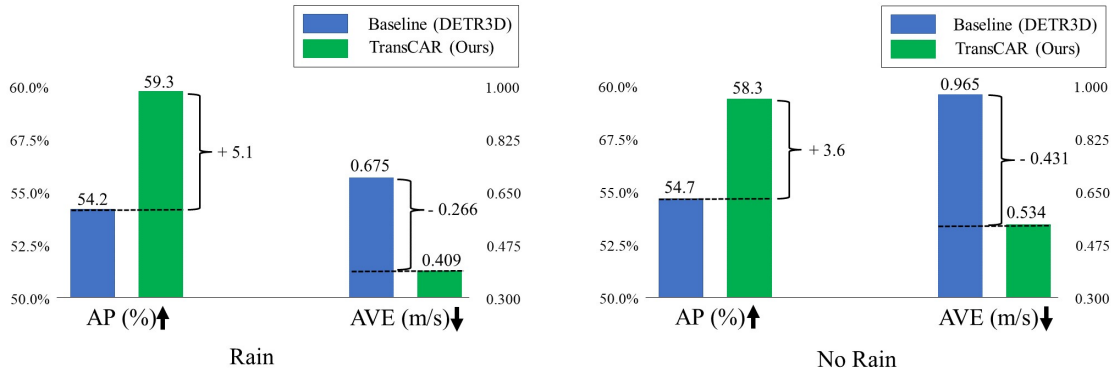


Figure 5.6: Comparison of Average Precision (AP) and Average Velocity Error (AVE, m/s) for class Car under the rain and no-rain scenes on nuScenes validation set. There are 1088 frames (27 scenes) among 6019 frames (150 scenes) in nuScenes validation set are annotated as rain. TransCAR can significant improve the detection performance and reduce the velocity estimation error under rainy conditions.

5.7. Note that nuScenes does not provide the weather label for each annotated frame, the weather information is provided in the scene description section (a scene is a 20-second data segment [2]). After manual check of some of the annotated frames, we found that not all the frames under 'rain' were captured when the rain was falling, some of them were collected right before or after the rain. However, these images are of lower quality compared to the ones collected during sunny weather. Therefore, they are suitable for our evaluation experiments. Figure 5.6 shows the AP and AVE for class Car under rain and no-rain scenes. TransCAR has a higher AP improvement (+5.1% vs. +3.6%) for the rain scenes compared to no-rain scenes. The AVE for rain scenes is smaller than in the no-rain scenes; this is because there are biases in the rain frames, and the cars within these rain scenes are closer to the ego vehicle, which makes them easier to be detected. Figure 5.7 shows the comparison of the detection performance of night and daytime scenes. Poor lighting conditions at night make the baseline method perform worse than daytime (52.2% vs. 54.8% in AP, 1.691m/w vs. 0.866m/w in AVE), TransCAR can leverage the radar data and boost the AP by 6.9% and reduce AVE by 1.012m/s. Although there are limited night scenes (15 scenes, 602 frames), this result can still demonstrate the effectiveness of TransCAR in night scenarios.

5.4 Summary of the Chapter

In this chapter, we proposed TransCAR, an effective and robust Transformer-based Camera-And-Radar 3D detection framework that can learn a soft-association between radar features and vision queries instead of a hard-association based on sensor calibration. The associated radar features improve range and velocity estimation. Our TransCAR sets the new state-of-the-art camera-radar detection performance on the challenging nuScenes detection

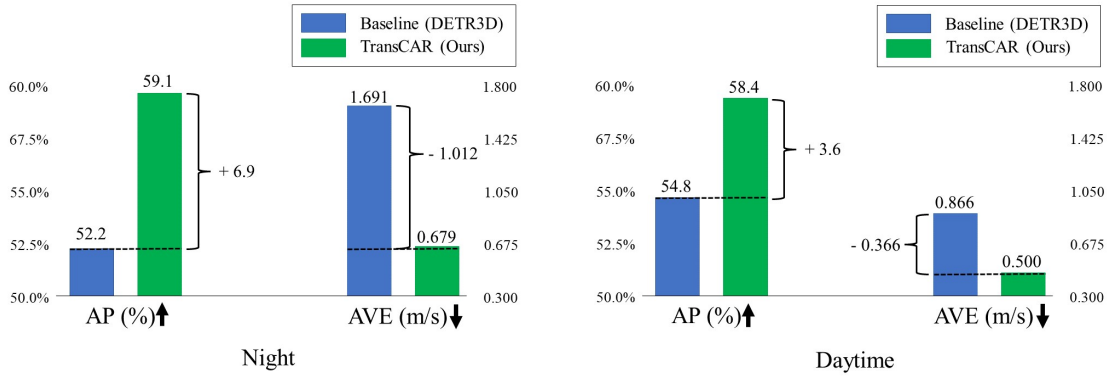


Figure 5.7: Comparison of Average Precision (AP) and Average Velocity Error (AVE, m/s) for class Car during the night and daytime scenes on nuScenes validation set. There are 602 frames among 6019 frames in nuScenes validation set are collected during the night. TransCAR can leverage the radar data to significantly improve the performance and reduce the velocity estimation error during the night when the camera is affected.

benchmark. We hope that our work can inspire further research on radar-camera fusion and motivate using transformers for sensor fusion.

Chapter 6

RFS-M^3: 3D Multi-Object Tracking using Random Finite Set based Multiple Measurement Models Filtering

6.1 Introduction

Autonomous vehicles need robust and accurate 3D perception to achieve safe maneuvering within a cluttered environment. There are three main problems in 3D perception: 3D object detection, multiple object tracking (MOT) and object trajectory forecasting. In a modular perception system, MOT is a critical module that connects detection and forecasting.

For tracking-by-detection approaches, the impact of the quality of input detections provided by the underlying detector is of paramount importance. However, due to the complexity of cluttered environments and limitations of learning-based detectors, there are many false positives, misses and inaccurate detections among input detections. Therefore, the main challenges for MOT in autonomous driving applications are threefold: (1) uncertainty in the number of objects; (2) uncertainty regarding when and where the objects may appear

and disappear; (3) uncertainty in objects' states.

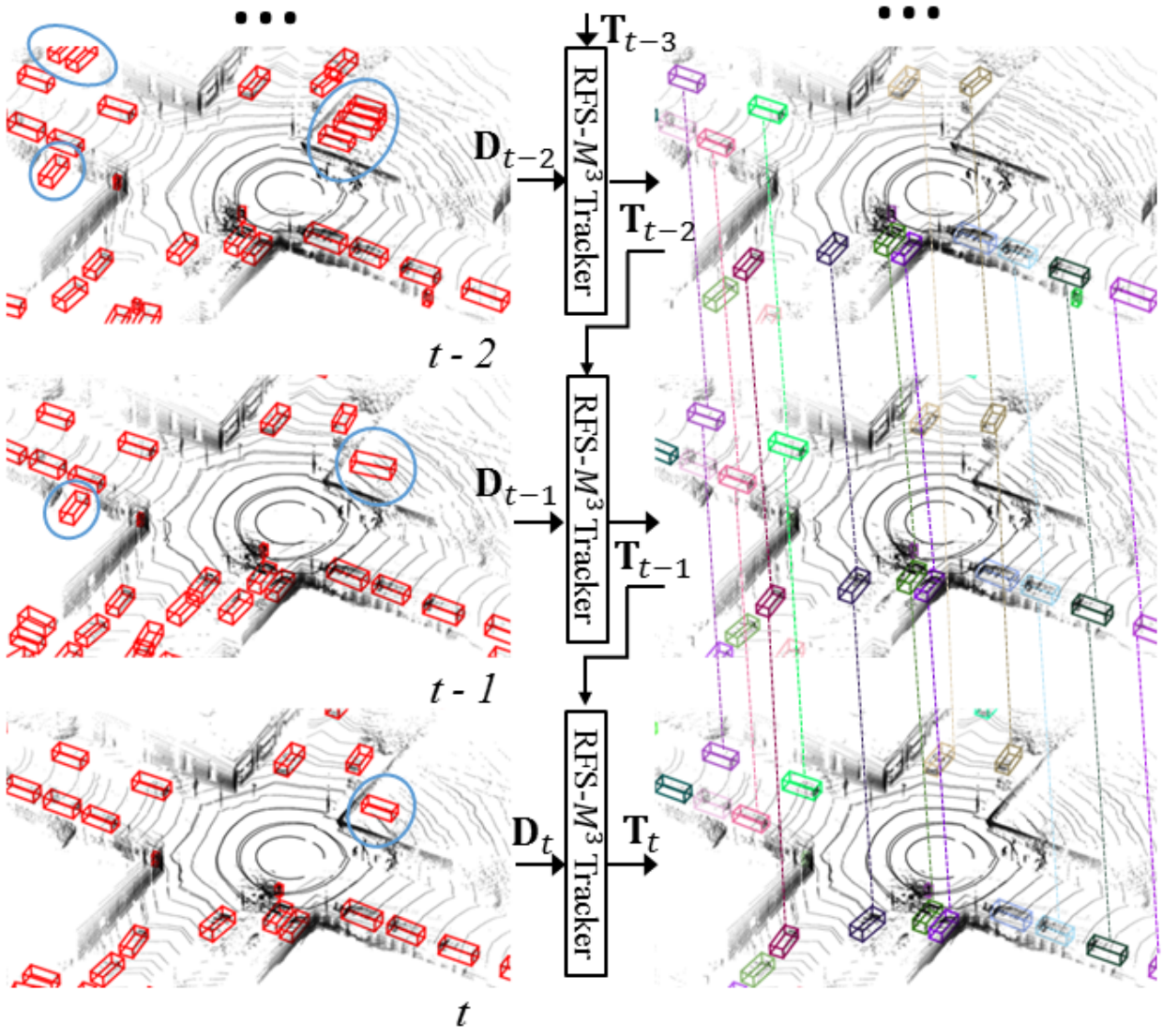


Figure 6.1: Overview of the proposed RFS-M^3 tracker pipeline. For each frame, many 3D detections are generated by a neural-network-based 3D detector, as the red bounding boxes on the left column. Our RFS-M^3 tracker successfully tracks targets and filters out false positives (boxes shown within the blue ellipses). For figures in the right column, different bounding box colors correspond to different unique tracked IDs. Some tracks with the same IDs are connected with dashed lines to help visualization. Best viewed in color.

The family of Random Finite Set (RFS) [29, 30, 31] based approaches are theoretically sound Bayesian frameworks that naturally model the uncertainties mentioned above accurately and elegantly. Although traditional filtering-based methods, such as Kalman filtering

[26, 6, 28], perform well in state estimation, they are not designed to model the unknown number of objects, or the so-called *birth and death* phenomena of objects. RFS-based MOT algorithms address these problems from a Bayesian perspective and have been shown to be very effective in radar-based applications [32, 33]. Among RFS-based approaches, Poisson multi-Bernoulli mixture (PMBM) filtering has shown superior performance and favorable computational cost [34]. However, the 3D input detection format (3D bounding boxes) for modern autonomous driving systems is significantly different from raw radar signals (points and Doppler velocity). The application of RFS for 2D/3D amodal detections (bounding boxes) from learning-based detections has not been well explored. Existing works in this area either underperform state-of-the-art trackers or they have been tested using a small dataset that do not reflect broad and truly challenging scenarios [35, 36, 37].

As discussed in Chapter 2, there are multiple evaluation metrics for 3D MOT corresponding to different application scenarios. The main difference lies in the definition of *True Positive metrics* (TP metrics). Waymo [7] uses the strictest TP metric: 3D intersection over union (IoU). This requires trackers to not only perform well in data association, but also perform well in estimating tracks' 3D locations, dimensions and orientations. The TP metrics for Argoverse [38] and nuScenes [2] are calculated only using a center distance threshold at 2 meters during matching. These different metrics and application scenarios require different system designs to optimize on-board computing resources.

We propose an RFS-based Multiple Measurement Models filter (RFS-M^3) to solve the 3D MOT problem with amodal detections for autonomous driving applications. In particular, we propose multiple measurement models ranging from the 3D bounding box model to the point measurement model for a PMBM filter in support of different TP metrics and optimize the usage of computing resources. Furthermore, our framework supports *amodal*

detections, which implies the ability to track objects that are only partially visible due to partial occlusions by other objects. This partial occlusion phenomenon represents a challenging and realistic condition that should be addressed by any viable tracking solution. The contributions of our paper are as follows:

- To the best of our knowledge, this represents a first successful attempt to employ an RFS-based approach that incorporates amodal 3D detections from a neural network for 3D MOT.
- Multiple measurement models including 3D bounding box model and point model are incorporated in our RFS-M^3 to support different MOT application scenarios.
- Our RFS-M^3 is an online real-time tracker with multiple global hypotheses maintained, and can run at an average rate of 20 Hz on a standard desktop.
- We validate the performance of our RFS-M^3 tracker using three extensive open datasets provided by three industry leaders – Waymo [7], Argoverse [38] and nuSceness [2]. It is worth noting that among entries that use the organizer provided detections, our RFS-M^3 ranks **No.2** in all-class MOTA[1] on the Waymo dataset.

6.2 Problem Formulation and System Overview

Our problem formulation and system pipeline are summarized in Figure 6.2. This includes an RFS-M^3 tracker designed to solve the 3D MOT problem given a set of noisy 3D detections, as well as estimate the number of objects, and maintain identity and state of each object. The details of the RFS-M^3 tracker are further illustrated in Section 4.

[1]Multi-object tracking accuracy, MOTA in short, is the primary metric for ranking in most MOT benchmarks.

Our work is an online tracking-by-detection system with 3D detections as input for each step. The standard format for n 3D detections in one frame can be defined as follows:

$$\begin{aligned} \mathbf{D} =& \{\mathbf{d_1}, \mathbf{d_2}, ...\mathbf{d_n}\}, \\ \mathbf{d_i} =& [x_i, y_i, z_i, h_i, w_i, l_i, \theta_i, s_i, cls_i] \end{aligned} \tag{6.1}$$

where $\mathbf{D}$ is the set of all n detections in one frame, for i_{th} detection $\mathbf{d_i}$, x_i, y_i, z_i is the center location in 3D space, h_i, w_i, l_i denote the height, width and length of the 3D bounding box, θ_i is the orientation around z axis, s_i and cls_i are the confidence score and class name (vehicle, pedestrian and so on) respectively. Note that our RFS-M^3 tracker module can work with any 3D detector that outputs standard 3D detections. The 3D detectors can be image-based [8, 56], LiDAR-based [10, 4] or multi-sensor fusion-based [1, 76]. But the impact of the quality of input detections is of paramount importance.

The output of our RFS-M^3 tracker for each frame is a set of m tracked 3D targets $\mathbf{T}$ defined as follows:

$$\begin{aligned} \mathbf{T} =& \{\mathbf{t_1}, \mathbf{t_2}, ...\mathbf{t_m}\}, \\ \mathbf{t_j} =& \left[x_j, y_j, z_j, h_j, w_j, l_j, \theta_j, weight_j, cls_j, id_j\right] \end{aligned} \tag{6.2}$$

where $weight_j$ is the weight of the target, it models the probability of the corresponding data association hypothesis. id_j denotes the unique global tracked ID of the target.

6.3 Proposed Method

The high-level system of the proposed RFS-M^3 MOT tracker architecture is shown in Figure 6.2. This illustrates the four primary components of the tracker: (1) PMBM Predictions;

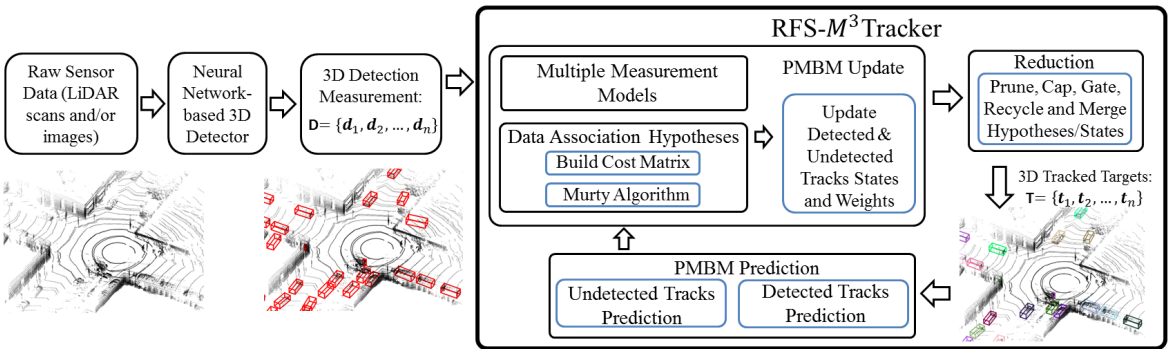


Figure 6.2: RFS-M^3 system architecture. The system works in a recursive way. Started at tracks from previous timestep, PMBM prediction is done first to predict each track's state, then according to the detection measurements for the current time step, we form different global association hypotheses from possible combinations of single target hypotheses (track-detection pairs). Then the update of tracks is done based on the data association. Finally the update outputs are sent to reduction module to remove unlikely results and output tracks for the current timestep.

(2) Data Association; (3) PMBM Update; and (4) Reduction.

6.3.1 Random Finite Set and MOT

A random finite set (RFS) is a set where both the cardinality and the variables within the set are random [95, 130]. In particular, a random finite set can be modeled using two layers of probability distributions, one for the number of elements (cardinality) of the set and the other for the elements themselves. The concept of random finite set fits the nature of MOT very well, where the number of tracks and the track's state is unknown. Combining Bayesian recursion with RFS has shown to be a promising direction to solve the MOT problem. RFS could model both the single track state as a stochastic variable as well as the number of tracks.

6.3.2 Detected and Undetected Tracks

Under the PMBM model [97, 98], the set of tracks $\mathbf{x_t}$ at timestamp t is the union of detected tracks $\mathbf{x}_t^d$ and undetected tracks $\mathbf{x}_t^u$. Detected tracks $\mathbf{x}_t^d$ are tracked objects that have been detected at least once. Undetected tracks $\mathbf{x}_t^u$ are potential objects that have not been detected. Note that we are not explicitly tracking the undetected tracks, which is impossible under a tracking-by-detection framework. Instead, we have a representation of their possible existences.

6.3.3 Object States with Different Measurement Models

There are two versions of object states corresponding to two measurement models: a 3D bounding box version and a point version. For 3D bounding boxes, the object state, $\mathbf{x^{3D}}$, and measurement, $\mathbf{z^{3D}}$, are defined as:

$$\begin{aligned} \mathbf{x^{3D}} &= \left[x, y, z, h, w, l, \theta, v_x, v_y, v_z, v_\theta\right] \\ \mathbf{z^{3D}} &= \left[x, y, z, h, w, l, \theta\right] \end{aligned} \tag{6.3}$$

where v_x, v_y, v_z represent the velocity of objects in 3D space, v_θ denotes the yaw angle velocity. x, y, z, h, w, l, θ are measurable states, while v_x, v_y, v_z, v_θ are unmeasureable states. Similarly, for point objects, the state, $\mathbf{x^{pt}}$, and measurement, $\mathbf{z^{pt}}$, are defined as:

$$\begin{aligned} \mathbf{x^{pt}} &= \left[x, y, v_x, v_y\right] \\ \mathbf{z^{pt}} &= \left[x, y\right]. \end{aligned} \tag{6.4}$$

As discussed in the Experiment section, each version has its own unique attributes to handle specific tracking TP metrics. Note that in addition to the 3D bounding box version, the

outputs from the point model version are also tracked 3D bounding boxes for 3D targets; however, the missing 3D information in the 2D point-target state is added from the associated 3D detection measurement.

6.3.4 Data Association Hypotheses

For each timestamp, there are multiple hypotheses for data association. In our measurement-driven framework, each measurement is one of three options: a newly detected target, a previously detected target, or a false positive detection. We form different global association hypotheses from possible combinations of the single target hypothesis (STH). For example, in Figure 6.3, for the time step t, T_1, T_2 and T_3 are tracked targets, D_1 and D_2 are detections. The dashed circles are gated regions for each track; detections that are out of gated regions will not be considered for association; for example, D_2 is not a valid association candidate for T_1 since it is out of the gated region of T_1. D_1 could be associated with T_1, D_2 could be associated with T_3, D_1 could be a new track or false positive detection, there are also other possibilities; these data association hypotheses for each valid target-detection pair are single target hypotheses.

In our framework, within a global association hypothesis, one detection can only be associated with one track in one global association hypothesis. For example, in Figure 6.3, $\{D_1$ is associated with T_1, T_2 miss-detected, D_2 is associated with $T_3\}$, $\{T_1$ miss-detected, D_1 is associated with T_2, D_2 is associated with $T_3\}$ and $\{T_1$ miss-detected, T_2 miss-detected, T_3 miss-detected, D_1 is a false positive detection, D_2 is a new track$\}$ are all valid global association hypotheses. While $\{D_1$ associates to T_1, D_1 associates to T_2, D_2 associates to $T_3\}$ is not a valid global association because D_1 is associated with multiple tracks T_1 and T_2 at the same time.

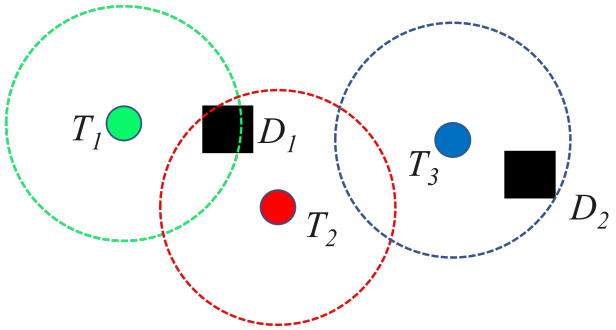


Figure 6.3: Example of single target hypothesis and global association hypotheses. T_1, T_2 and T_3 are tracked targets, D_1 and D_2 are detections. The dashed solid circles are gated regions for each track, detections that are out of gated regions will not be considered for association.

One of the advantages of our RFS-M^3 tracker is that we maintain multiple global hypotheses instead of only one. For the k_{th} global hypothesis in one frame, assuming that there are n tracked targets (detected tracks) given m detection measurements, the m by $(n+m)$ cost matrix is built as follows:

$$L^k = \begin{bmatrix} -l_{1,1,k} & -l_{1,2,k} & \dots & -l_{1,n,k} & -l_{1,0} & \infty & \dots & \infty \\ -l_{2,1,k} & -l_{2,2,k} & \dots & -l_{2,n,k} & \infty & -l_{2,0} & \dots & \infty \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ -l_{m,1,k} & -l_{m,2,k} & \dots & -l_{m,n,k} & \infty & \infty & \dots & -l_{m,0} \end{bmatrix} \tag{6.5}$$

The left $m \times n$ block contains the negative logarithmic costs of associating each of the m detection measurements with each of the n previously tracked targets (detected tracks). For the element $-l_{i,j,k}$ in the i_{th} row and the j_{th} column, it represents the negative logarithmic cost of associating the i_{th} detection measurement to the j_{th} tracked target and is calculated as $-(log(w^d_{i,j}) - log(w^d_{0,j}))$. Here, $w^d_{i,j}$ is the track-detection single target hypothesis weight, and $w^d_{0,j}$ is the missed detection single target hypothesis weight. The calculation details of these weights are given in Section 6.3.7. The right $m \times m$ square block contains negative

logarithmic costs for generating new targets from detection measurements.

Murty's algorithm [131], an extension of the Hungarian algorithm [124] is used to generate K best global hypotheses instead of only one. The weight for each global hypothesis is based on the product of its detected tracks' weights.

6.3.5 PMBM Density

Under the PMBM model, Poisson RFS, also named Poisson point process (PPP), is used to represent undetected tracks, and a multi-Bernoulli mixture (MBM) RFS is used to represent detected tracks [97, 98]. The PMBM density can be expressed as:

$$\mathcal{PMBM}_t(\mathbf{x}) = \sum_{\mathbf{x}^u \uplus \mathbf{x}^d = \mathbf{x}} \mathcal{P}_t(\mathbf{x}^u)\mathcal{MBM}_t(\mathbf{x}^d) \tag{6.6}$$

where $\mathbf{x}$ represents all objects in the surveillance area and $\mathbf{x}$ is the disjoint union set of undetected tracks $\mathbf{x}^u$ and detected tracks $\mathbf{x}^d$. Symbols $\mathcal{P}(\cdot)$ and $\mathcal{MBM}(\cdot)$ are the Poisson point process density and the multi-Bernoulli mixture density, respectively. We assume that the PPP mixture intensity and Bernoulli state probability density functions are Gaussian. Therefore, the parameters for N^u undetected tracks PPP densities are as follows:

$$\left\{w_i^u, \boldsymbol{\mu}_i^u, \boldsymbol{\Sigma}_i^u\right\}_{i=1}^{N^u} \tag{6.7}$$

where w_i^u is the weight of i_{th} undetected track, $\boldsymbol{\mu}_i^u$ and $\boldsymbol{\Sigma}_i^u$ are the state mean and covariance, respectively. The parameters for N^d detected tracks MBM densities are:

$$\left\{w_j^d, r_j^d, \boldsymbol{\mu}_j^d, \boldsymbol{\Sigma}_j^d\right\}_{j=1}^{N^d} \tag{6.8}$$

where w_j^d and r_j^d are the STH weight of the j_{th} detected track and the probability of existence, and $\boldsymbol{\mu}_j^d$ and $\boldsymbol{\Sigma}_j^d$ are the Gaussian state variables. The PMBM density parameters are the combination of PPP density parameters and MBM density parameters. The PMBM prediction and update are the processes of predicting and updating the PMBM parameters.

6.3.6 PMBM Prediction

A crucial aspect of the PMBM filter is its *conjugacy* property, which was proved in [98]. The notion of conjugacy is critical for robust and accurate Bayesian-based MOT. In summary, the conjugacy of the PMBM filter inplies that if the prior is in a PMBM form, then the distribution after the Bayesian prediction and update steps will be of the same distribution form. Therefore, the prediction stage of a PMBM filter can be written as follows.

$$\mathcal{PMBM}_{t+1|t}(\mathbf{x}_{t+1}) = \int p(\mathbf{x}_{t+1}|\mathbf{x}_t)\mathcal{PMBM}_{t|t}(\mathbf{x}_t)\delta\mathbf{x}_t \tag{6.9}$$

where $p(\mathbf{x}_{t+1}|\mathbf{x}_t)$ represents the transition density. A constant velocity model is used as the motion model in this work for simplicity. The undetected and detected tracks can be predicted independently. P_s is defined as the probability of survival, namely the probability that an object survives from one time step to the next. Standard Bayesian prediction is applied for both detected and undetected track states. The weight of each undetected track is scaled by P_s in the prediction step to account for the higher uncertainty in the prediction step:

$$w_{t+1|t}^u = P_s w_{t|t}^u \tag{6.10}$$

For detected tracks, which are modeled as multi-Bernoulli mixture (MBM) RFSs, each MB process can also be predicted independently of the others. The single target hypothesis weight w_j^d and probability of existence r_j^d of each detected track can be predicted as follows:

$$\begin{aligned} w_{t+1|t}^d &= w_{t|t}^u \\ r_{t+1|t}^d &= P_s r_{t|t}^d \end{aligned} \tag{6.11}$$

The single target hypothesis weight remains unchanged because no new information about detection measurement is incorporated in the prediction stage. The probability of existence for each MB-modeled object is decreased by a factor P_s in order to account for the higher uncertainty of existence within the Prediction stage.

Unlike the standard PMBM filter that uses a constant for P_s, we incorporate confidence scores from previously associated measurements in P_s for detected tracks. This is because detections with higher confidence scores indicate a higher chance of existence and survival across frames.

6.3.7 PMBM Update

By adding information from the measurement model $p(\mathbf{z}_t|\mathbf{x}_t)$, the PMBM density can be updated with:

$$\mathcal{PMBM}_{t+1|t+1}(\mathbf{x}_{t+1}) = \frac{p(\mathbf{z}_{t+1}|\mathbf{x}_{t+1})\mathcal{PMBM}_{t+1|t}(\mathbf{x}_{t+1})}{\int p(\mathbf{x}_{t+1}|\mathbf{x}'_{t+1})\mathcal{PMBM}_{t+1|t}(\mathbf{x}'_{t+1})\delta\mathbf{x}'_t} \tag{6.12}$$

The update steps for different hypotheses are different, including detected/undetected tracks, being associated with a detection measurement or not. The association metrics vary depending on the measurement models. For the 3D bounding box measurement model, the

association metric uses the Mahalanobis distance and 3D IoU between predicted 3D measurements and real 3D measurements. The point measurement model only uses the Mahalanobis distance between the center points.

Update of Undetected Tracks without Associated Measurement: Undetected tracks that do not have any measurement associated with them remain undetected. The Bayesian update will thus not change the states or variances of the Poisson distributions since no new information is added. Therefore, the weight of each undetected track is reduced by a factor of $(1 - P_d)$ to account for the decreased probability of existing [97, 98]:

$$w^u_{t+1|t+1} = (1 - P_d) w^u_{t+1|t} \tag{6.13}$$

P_d is the probability of detection, that is, the probability that it should be detected.

Update of Potential New Tracks Detected for the First Time: Our RFS-M^3 tracker is a measurement-driven framework: an object must be connected to a measurement in order to be classified as a new track (detected for the first time). All undetected tracks and corresponding gated measurements are considered to generate the new tracks. Note that the detections from a neural network have confidence scores attached to them. This confidence score is an invaluable indicator of the object probability of existence. Therefore, unlike a standard PMBM filter, we incorporate the detection confidence score into the update step of objects detected for the first time. For detections with confidence scores larger than a threshold (dependent on different learning-based 3D detector), we generate a potential new target by adding a new Bernoulli process and plugging the negative logarithmic weight in the right $m \times m$ blocks diagonal in the cost matrix L discussed in Section 6.3.4. For detections with lower confidence score, since we are not certain about their existences and require more

evidences from the future, an undetected track with PPP density is generated for each of them.

Update of Detected Tracks without Associated Measurement: If there is no measurement associated with the detected tracks, which was detected from a previous frame, we maintain the object predicted state unchanged. Furthermore, the probability of existence and weight are updated with:

$$\begin{aligned} r^d_{t+1|t+1} =& \frac{r^d_{t+1|t}(1-P_d)}{1-r^d_{t+1|t}+r^d_{t+1|t}(1-P_d)} \\ w^d_{t+1|t+1} =& w^d_{t+1|t}(1-r^d_{t+1|t}+r^d_{t+1|t}(1-P_d)) \end{aligned} \tag{6.14}$$

This weight is the missed detection weight mentioned in Section 6.3.4. Here P_d is the confidence score of the detection that is associated with this detected track in the previous frame. Unlike standard Kalman filter-based trackers, the survival time of detected tracks without measurement varies based on the tracking status from the previous time period.

Update of Detected Tracks with Associated Measurement: For a detected track with associated measurements, the predicted state is updated by weighting in the information contained in the measurement, and a standard Kalman filter [26] is used to update the state vector. Also, the updated probability of existence is set to 1 because one cannot associate a measurement to an object that does not exist. The updated weight can be calculated as:

$$w^d_{t+1|t+1} = w^d_{t+1|t} r^d_{t+1|t} P_d \mathcal{N}(\boldsymbol{z_t}; \boldsymbol{H}\boldsymbol{x}^d_{t+1|t+1}, \hat{\boldsymbol{S}}) \tag{6.15}$$

where $w^d_{t+1|t}$ and $r^d_{t+1|t}$ are the predicted weight and probability of existence from the prediction stage. $\hat{\boldsymbol{S}}$ and $\mathcal{N}(\boldsymbol{z_t}; \boldsymbol{H}\boldsymbol{x}^d_{t+1|t+1}, \hat{\boldsymbol{S}})$ are the innovation covariance and measurement

likelihood, and $\boldsymbol{H}$ is the observation matrix. Here we set probability of detection P_d equal to the associated detection confidence score. This weight is the target hypothesis-measurement pair weight $w_{i,j}^{d}$ mentioned in the Section 6.3.4.

6.3.8 Reduction

The general assignment problem in MOT is NP-hard [132], and hence reducing the number of hypotheses is necessary for managing computational complexity and maintaining real-time performance. Five reduction techniques are used in this work: pruning, capping, gating, recycling and merging. *Pruning* is used to remove objects and global hypothesis with low weights. Note that the weight of a global hypothesis is the sum of detected tracks' weights. *Capping* is used to set an upper bound for the number of global hypotheses and the number of detected tracks within one global hypothesis. *Gating* limits the search distance for data association using Mahalanobis distance instead of Euclidean distance. *Recycling* moves detected tracks with lower weights to undetected track set rather than discarding them. There may be non-unique global hypotheses, and *merging* combines these identical global hypotheses into one.

6.4 Experimental Results

6.4.1 Settings and Datasets

We evaluate our method on three popular open datasets provided by three industry leaders: Waymo [7], Argoverse [38] and nuScenes [2]. The basic information for each dataset is shown in Table 6.1.

Since our method is not a learning-based method, we do not use the training set; but the validation set is used for parameter tuning. The field of view of these datasets is 360 degrees. Note that for these open datasets, ground truth labels are only available for training and validation sets. For fairness of evaluation of testing samples, one needs to submit the tracking results to the relevant server.

Standard evaluation metrics [41] for MOT are used, including MOTA, MOTP (multi-object tracking precision), FP (False Positives), IDS (ID switches) and so on. The details of the metrics for each benchmark have been explained in Chapter 2 and can be found in [7, 38, 2].

6.4.2 Evaluation Results

The overall results for all classes on Waymo, Argoverse and nuScenes dataset are shown in Table 6.2, Table 6.6 and Table 6.9 respectively. RFS-M^3-Point represents our RFS-M^3 with point measurement model and RFS-M^3-3D denotes RFS-M^3 with 3D bounding box measurement model. For Waymo dataset, the results for each class (vehicle, pedestrian and cyclist) are shown in Table 6.3, Table 6.4 and Table 6.5 respectively. Note that Waymo dataset breakdowns all the ground truth labels into two difficulty levels: level 1 (L1 shown in the result Tables) considers ground truth labels that have at least 5 LiDAR points, and level2 (L2 shown in the result Tables) considers ground truth labels that have at least 1

Dataset	#Scenes/Segments	#LiDAR Frames	#Annotated Frames	#Evaluation Class
Waymo	1152	∼230.4K	∼230.04K	3
Argoverse	113	∼22.4K	∼22.4K	2
nuScenes	1000	∼400K	∼40.04K	7

Table 6.1: Basic information of Waymo, Argoverse and nuScenes Datasets.

Method	Class	MOTA/L1 (%) ↑	MOTP/L1 ↓	FP/L1(%) ↓	Misses/L1 (%) ↓	MOTA/L2 (%) ↑	MOTP/L2 ↓	FP/L2(%) ↓	Misses/L2 (%) ↓
Waymo Baseline [7]	ALL	27.13	0.1753	9.78	62.88	25.92	0.263	13.98	64.55
AB3DMOT-style KF [38]	ALL	30.20	0.2696	17.80	51.74	29.14	0.270	17.14	53.47
2-Stage Data Association Tracker [135]	ALL	37.91	0.2626	9.18	51.53	36.53	0.2626	8.85	53.27
Probabilistic KF [28]	ALL	37.97	0.2696	8.62	52.28	36.57	0.270	8.32	54.02
Our RFS-M^3-Point	ALL	39.95	0.2698	8.03	51.11	38.51	0.270	7.74	52.86
Our RFS-M^3-3D	ALL	**43.21**	0.2707	9.30	47.21	**41.73**	0.270	8.98	49.01

Table 6.2: Quantitative comparison of 3D MOT evaluation results for ALL CLASSES on Waymo test set.

LiDAR points. For Argoverse dataset, the results for each class (car and pedestrian) are shown in Table 6.7 and Table 6.8 respectively. There are 10 classes in nuScenes 3D tracking benchmark, so we don't show the results for each class, but the results are available at [**?**].

As shown in these tables, our method outperforms other state-of-the-art tracker significantly. By Oct 2020, among all entries that use organizer-provided detections, our RFS-M^3 tracker ranked **No.2** on the Waymo 3D tracking leaderboard; For Argoverse leaderboard, we ranked **No.1** in Vehicle MOTA, **No.2** in all-class MOTA and **No.3** in averaged ranking (primary metric). For tracking-by-detection, the quality of input detections is of paramount importance. Although some methods used better self-generated detections than provided by the organizer, nevertheless, our performance is very competitive. In Table 6.10, we show the results of RFS-M^3 with different input detections. The APH for CenterPoint[133] is 14.8% higher than PPBA (organizer provided detections)[134], and this results in a 10.5% improvement in MOTA. We believe that our RFS-M^3 would perform better in these test benchmarks with better input detections than organizer precomputed ones.

Our RFS-M^3-3D performs significantly better than RFS-M^3-Point on the Waymo dataset (Table 6.2), while they have similar performance on the Argoverse and nuScenes datasets (Table 6.6). This is because Waymo uses the strictest *True Positive metrics* – 3D IoU,

Method	Class	MOTA/L1 (%) ↑	MOTP/L1 ↓	FP/L1(%) ↓	Misses/L1 (%) ↓	MOTA/L2 (%) ↑	MOTP/L2 ↓	FP/L2(%) ↓	Misses/L2 (%) ↓
Waymo Baseline [7]	vehicle	42.49	0.1856	17.33	40.04	40.08	0.1856	16.35	43.44
AB3DMOT-style KF [38]	vehicle	38.20	0.1812	25.90	35.79	36.00	0.1812	24.41	39.48
2-Stage Data Association Tracker [135]	vehicle	45.59	0.1751	10.43	43.53	42.75	0.1751	9.78	47.06
Probabilistic KF [28]	vehicle	48.49	0.1783	9.25	41.88	45.48	0.1783	8.68	45.49
Our RFS-M^3-Point	vehicle	48.71	0.1787	9.85	41.26	45.70	0.1787	9.24	44.88
Our RFS-M^3-3D	vehicle	**50.11**	0.1774	9.75	40.08	**47.06**	0.1774	9.16	43.73

Table 6.3: Quantitative comparison of 3D MOT evaluation results for VEHICLE class on Waymo test set.

Method	Class	MOTA/L1 (%) ↑	MOTP/L1 ↓	FP/L1(%) ↓	Misses/L1 (%) ↓	MOTA/L2 (%) ↑	MOTP/L2 ↓	FP/L2(%) ↓	Misses/L2 (%) ↓
Waymo Baseline [7]	pedestrian	38.91	0.3403	12.00	48.60	37.69	0.3403	11.62	50.21
AB3DMOT-style KF [38]	pedestrian	28.24	0.3431	14.35	57.00	27.30	0.3431	13.87	58.43
2-Stage Data Association Tracker [135]	pedestrian	39.83	0.3456	10.14	48.71	38.56	0.3456	9.82	50.34
Probabilistic KF [28]	pedestrian	35.95	0.3444	9.68	51.91	34.79	0.3444	9.37	53.47
Our RFS-M^3-Point	pedestrian	**42.28**	0.3453	11.81	45.48	**40.97**	0.3453	11.44	47.18
Our RFS-M^3-3D	pedestrian	39.10	0.3432	7.78	52.16	37.81	0.3432	7.53	53.73

Table 6.4: Quantitative comparison of 3D MOT evaluation results for PEDESTRIAN class on Waymo test set.

Method	Class	MOTA/L1 (%) ↑	MOTP/L1 ↓	FP/L1(%) ↓	Misses/L1 (%) ↓	MOTA/L2 (%) ↑	MOTP/L2 ↓	FP/L2(%) ↓	Misses/L2 (%) ↓
AB3DMOT-style KF [38]	cyclist	24.16	0.2844	13.17	62.44	24.13	0.2844	13.15	62.49
2-Stage Data Association Tracker [135]	cyclist	28.30	0.2670	6.96	62.35	28.26	0.2670	6.95	62.40
Probabilistic KF [28]	cyclist	29.48	0.2861	6.92	63.04	29.44	0.2861	6.91	63.09
Our RFS-M^3-Point	cyclist	32.05	0.2875	6.46	59.92	32.01	0.2875	6.45	59.97
Our RFS-M^3-3D	cyclist	**37.22**	0.2893	6.35	56.06	**37.17**	0.2893	6.34	56.12

Table 6.5: Quantitative comparison of 3D MOT evaluation results for CYCLIST class on Waymo test set.

Method	Class	MOTA (%) ↑	IDF1 ↑	MOTP ↓	MOTP-I* ↓	#False Positive ↓	#Misses ↓	#IDS ↓
Argoverse Baseline [38]	All	57.11	0.685	0.355	0.190	20626	49374	624
Our RFS-M^3-Point	All	60.12	0.705	0.370	0.195	14202	48443	1145
Our RFS-M^3-3D	All	58.55	0.705	0.345	0.180	11536	52939	1520

Table 6.6: Quantitative comparison of 3D MOT evaluation results for ALL CLASSES on Argoverse test set. *MOTP-I: amodal shape estimation error, computed by the 1-IoU of 3D bounding box projections on xy plane after aligning orientation and center point.

Method	Class	MOTA (%) ↑	IDF1 ↑	MOTP ↓	MOTP-I* ↓	#False Possitive ↓	#Misses ↓	#IDS ↓
Argoverse Baseline [38]	Car	65.90	0.79	0.34	0.20	15693	23594	200
Our RFS-M^3-Point	Car	71.67	0.81	0.34	0.19	8278	24165	362
Our RFS-M^3-3D	Car	71.14	0.81	0.32	0.17	6996	26039	382

Table 6.7: Quantitative comparison of 3D MOT evaluation results for CAR on Argoverse test set.

Method	Class	MOTA (%) ↑	IDF1 ↑	MOTP ↓	MOTP-I* ↓	#False Possitive ↓	#Misses ↓	#IDS ↓
Argoverse Baseline [38]	All	48.31	0.58	0.37	0.18	4933	25780	424
Our RFS-M^3-Point	All	48.56	0.60	0.40	0.20	5924	24278	783
Our RFS-M^3-3D	All	45.92	0.60	0.37	0.19	4540	26900	1138

Table 6.8: Quantitative comparison of 3D MOT evaluation results for PEDESTRIAN on Argoverse test set.

Method	Class	AMOTA* ↑ (%)	AMOTP* ↓	MOTA (%) ↑	MOTP ↓	#False Positive ↓	#Misses ↓	#IDS ↓
nuScenes Baseline [2]	All	15.1	1.501	15.40	0.402	15088	75730	9027
Probabilistic KF [28]	All	55.0	0.798	45.9	0.353	17533	33216	950
Our RFS-M^3-Point	All	61.9	0.752	52.4	0.387	16728	27168	1525
Our RFS-M^3-3D	All	61.4	0.716	51.1	0.363	19265	26921	1892

Table 6.9: Quantitative comparison on nuScenes test set. *AMOTA and AMOTP: average MOTA and MOTP across different thresholds [6, 2].

while Argoverse and nuScenes use ground plane center distance. The temporal 3D bounding box information contained in RFS-M^3-3D provides limited assistance in reducing center distance but is crucial to estimating 3D bounding boxes with higher accuracy. This is also supported by the result that compared to RFS-M^3-Point, RFS-M^3-3D has smaller MOTP-I on Argoverse dataset.

Method (Tracker+Detector)	Class	Detection APH* ↑	MOTA↑	MOTP ↓
RFS-M^3-3D+PPBA[134]	Car	49.4%	40.4%	0.182
RFS-M^3-3D+CenterPoint[133]	Car	64.2%	50.9%	0.171

Table 6.10: Comparison of RFS-M^3 with different input detections on Waymo val set, all metrics in LEVEL_2 difficulty. *APH: average precision weighted by heading [7].

6.4.3 Ablation Study

We evaluate the contribution of each important component in our RFS-M^3. The results are shown in Table 6.11. One advantage of our RFS-M^3 tracker is that we maintain multiple global hypotheses instead of only one to achieve improved data association across frames. This is because ambiguities within input detections could lead to wrong data associations having higher weights than correct associations. Keeping the correct associations even with lower weights could help correct the associations in future frames. P_d is the probability of detection. Incorporating the detection confidence score into P_d, rather than using a constant as in standard PMBM, can better model the uncertainties within the prediction and update. Gating is applied to limit the search distance for data association, and compared to Euclidean distance, Mahalanobis distance performs better because it takes the estimated state uncertainties into consideration.

Method	Class	MOTA ↑ (Primary) (%)	MOTP ↓	FP(%) ↓	Misses (%) ↓
Single Global Hypo	Vehicle	38.82	0.183	9.63	51.29
Constant P_d	Vehicle	27.83	0.183	7.13	63.74
Gating with Euclidean dis	Vehicle	38.57	0.183	9.51	51.24
RFS-M^3	Vehicle	40.40	0.182	9.90	49.60

Table 6.11: Ablation study of RFS-M^3-3D on Waymo val set, all metrics in LEVEL_2 difficulty.

6.5 Summary of the Chapter

In this chapter, we present an RFS-M^3 tracker to solve the 3D amodal MOT problem with multiple measurement models in support of different autonomous driving scenarios. Our framework can naturally model the uncertainties in the MOT problem. This represents a first successful attempt to employ an RFS-based approach in conjunction with 3D neural network-based detectors and with comprehensive testing using large-scale datasets. The experimental results on Waymo, Argoverse and nuScenes datasets demonstrate that our approach outperforms previous state-of-the-art methods by a large margin.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Autonomous driving systems require accurate 3D object detection and tracking to achieve reliable path planning and navigation. Different types of sensors including camera, LiDAR and radar, are required in the self-driving perception system due to their complimentary characteristics. Sensor fusion is crucial for combining data from different sensors and producing robust and improved results.

Throughout this dissertation, we have proposed and demonstrated the effectiveness of multiple sensor fusion-based 3D object detection and multi-object tracking techniques. We first present CLOCs (Chapter 3) and Fast-CLOCs (Chapter 4) for camera-LiDAR fusion-based 3D object detection. Our approach uses much-reduced thresholds for each sensor and combines detection candidates before NMS. By leveraging cross-modality information, it can keep detection candidates that would be mistakenly suppressed by single-modality methods and remove detection candidates that violate the consistency between different sensors. In Chapter 5, we apply Transformer for camera-radar fusion and propose TransCAR. TransCAR can learn the soft-association between radar features and vision queries via transformer decoder instead of hard-association depended on sensor calibration. The associated radar information can improve the range and velocity estimation for vision-based

detections. Then we present an RFS-M^3 tracker (Chapter 6) to solve the 3D MOT problem with multiple measurement models in support of different autonomous driving scenarios. Our framework can naturally model the uncertainties in the MOT problem. The experimental results on Waymo, Argoverse and nuScenes datasets demonstrate that our approaches outperform previous state-of-the-art methods.

7.2 Future Work

Maps for 3D Object Detection: High resolution maps are necessary for autonomous driving applications including self-localization and path planning. Autonomous driving companies either build HD maps themselves or cooperate with mapping companies to access the maps [1]. Most open detection and tracking benchmarks including Waymo, Argoverse and nuScenes [7, 2, 38], also provide HD maps. These HD maps consist of accurate road topology information, which could be beneficial to object detection and tracking. In particular, the lane and parking information within the maps can provide critical constraints to where objects may realistically appear and head to in the future. The question of how to incorporate map information into detection and tracking networks remains an open problem; apparently, more research efforts are needed on this topic.

Joint Object Detection and Tracking: Through this dissertation, we have studied 3D object detection and tracking in cascaded fashion, which is known as tracking-by-detection framework. The tracking module takes the output of a detection module as input, and the detection network processes each data frame independently. We believe that the tracking results can be used as a priori for the detection network for the next time step. Because

[1] WAYMO website: https://waymo.com/, CRUISE website: https://www.getcruise.com/, HERE map weibsite: https://www.here.com/.

the tracking module takes previous temporal information into consideration, and tracked targets are more likely to appear at the close neighborhoods for the next time step. Within a joint detection and tracking framework, the tracking module can assist detection, and better detection can benefit tracking in return.

BIBLIOGRAPHY

BIBLIOGRAPHY

[1] Su Pang, Daniel Morris, and Hayder Radha. Clocs: Camera-lidar object candidates fusion for 3d object detection. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[2] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *arXiv preprint arXiv:1906.09756*, 2019.

[4] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[5] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[6] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[7] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020.

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

[9] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019.

[10] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *CVPR*, 2020.

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

[12] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, pages 180–191. PMLR, 2022.

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[14] Roberto Calandra, André Seyfarth, Jan Peters, and Marc P. Deisenroth. Bayesian optimization for learning gaits under uncertainty. *Annals of Mathematics and Artificial Intelligence (AMAI)*, 76(1):5–23, 2016.

[15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2117–2125, 2017.

[16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[17] Yin Zhou and Oncel 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.

[18] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[19] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

[20] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana 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.

[21] Yongjian Chen, Lei Tai, Kai Sun, and Mingyang 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.

[22] Cody Reading, Ali Harakeh, Julia Chae, and Steven L. Waslander. Categorical depth distributionnetwork for monocular 3d object detection. *CVPR*, 2021.

[23] Bence Major, Daniel Fontijne, Amin Ansari, Ravi Teja Sukhavasi, Radhika Gowaikar, Michael Hamilton, Sean Lee, Slawomir Grzechnik, and Sundar Subramanian. Vehicle detection with automotive radar using deep learning on range-azimuth-doppler tensors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[24] Ole Schumann, Markus Hahn, Jürgen Dickmann, and Christian Wöhler. Semantic segmentation on radar point clouds. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2179–2186. IEEE, 2018.

[25] Michael Meyer, Georg Kuschk, and Sven Tomforde. Graph convolutional networks for 3d object detection on radar data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3060–3069, 2021.

[26] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

[27] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020.

[28] Hsu-kuang Chiu, Antonio Prioletti, Jie Li, and Jeannette Bohg. Probabilistic 3d multi-object tracking for autonomous driving. *arXiv preprint arXiv:2001.05673*, 2020.

[29] B-N Vo, Sumeetpal Singh, and Arnaud Doucet. Sequential monte carlo methods for multitarget filtering with random finite sets. *IEEE Transactions on Aerospace and electronic systems*, 41(4):1224–1245, 2005.

[30] Ronald PS Mahler. *Statistical multisource-multitarget information fusion*, volume 685. Artech House Norwood, MA, 2007.

[31] Ba-Tuong Vo, Ba-Ngu Vo, and Antonio Cantoni. Bayesian filtering with random finite set observations. *IEEE Transactions on signal processing*, 56(4):1313–1326, 2008.

[32] Ba-Tuong Vo and Ba-Ngu Vo. A random finite set conjugate prior and application to multi-target tracking. In *2011 Seventh International Conference on Intelligent Sensors, Sensor Networks and Information Processing*, pages 431–436. IEEE, 2011.

[33] Francesco Papi, Ba-Tuong Vo, Mélanie Bocquel, and Ba-Ngu Vo. Multi-target track-before-detect using labeled random finite set. In *2013 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 116–121. IEEE, 2013.

[34] Yuxuan Xia, Karl Granstrcom, Lennart Svensson, and Ángel F García-Fernández. Performance evaluation of multi-bernoulli conjugate priors for multi-target filtering. In *2017 20th International Conference on Information Fusion (Fusion)*, pages 1–8. IEEE, 2017.

[35] Bharath Kalyan, KW Lee, S Wijesoma, D Moratuwage, and Nicholas M Patrikalakis. A random finite set based detection and tracking using 3d lidar in dynamic environments. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 2288–2292. IEEE, 2010.

[36] Kwang Wee Lee, Bharath Kalyan, Sardha Wijesoma, Martin Adams, Franz S Hover, and Nicholas M Patrikalakis. Tracking random finite objects using 3d-lidar in marine environments. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1282–1287, 2010.

[37] Karl Granström, Stephan Renter, Maryam Fatemi, and Lennart Svensson. Pedestrian tracking using velodyne data—stochastic optimization for extended object tracking. In *2017 ieee intelligent vehicles symposium (iv)*, pages 39–46. IEEE, 2017.

[38] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[39] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2011 (VOC2011) Results. http://www.pascal-network.org/challenges/VOC/voc2011/workshop/index.html.

[40] Keni Bernardin, Alexander Elbs, and Rainer Stiefelhagen. Multiple object tracking performance metrics and evaluation in a smart room environment. In *Sixth IEEE International Workshop on Visual Surveillance, in conjunction with ECCV*, volume 90. Citeseer, 2006.

[41] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

[42] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016.

[43] Yunsong Zhou, Yuan He, Hongzi Zhu, Cheng Wang, Hongyang Li, and Qinhong Jiang. Monocular 3d object detection: An extrinsic parameter free approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7556–7566, 2021.

[44] Xiaoyang Guo, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Liga-stereo: Learning lidar geometry aware representations for stereo-based 3d detector. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3153–3163, 2021.

[45] Wu Zheng, Weiliang Tang, Li Jiang, and Chi-Wing Fu. Se-ssd: Self-ensembling single-stage object detector from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14494–14503, 2021.

[46] Hualian Sheng, Sijia Cai, Yuan Liu, Bing Deng, Jianqiang Huang, Xian-Sheng Hua, and Min-Jian Zhao. Improving 3d object detection with channel-wise transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2743–2752, 2021.

[47] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Celine Teuliere, and Thierry Chateau. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2040–2049, 2017.

[48] Roozbeh Mottaghi, Yu Xiang, and Silvio Savarese. A coarse-to-fine model for 3d pose estimation and sub-category recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 418–426, 2015.

[49] Peixuan Li, Huaici Zhao, Pengfei Liu, and Feidao 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.

[50] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang. Delving into localization errors for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4721–4730, 2021.

[51] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. In *International Conference on Learning Representations*, 2019.

[52] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8445–8453, 2019.

[53] Xinshuo Weng and Kris Kitani. Monocular 3d object detection with pseudo-lidar point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[54] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End-to-end pseudo-lidar for image-based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5881–5890, 2020.

[55] Yan Wang, Xiangyu Chen, Yurong You, Li Erran Li, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. Train in germany, test in the usa: Making 3d object detectors generalize. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11713–11723, 2020.

[56] Xinzhu Ma, Zhihui Wang, Haojie Li, Pengbo Zhang, Wanli Ouyang, and Xin Fan. Accurate monocular 3d object detection via color-embedded 3d reconstruction for autonomous driving. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6851–6860, 2019.

[57] Bo Li, Chunhua Shen, Yuchao Dai, Anton Van Den Hengel, and Mingyi He. Depth and surface normal estimation from monocular images using regression on deep features and hierarchical crfs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1119–1127, 2015.

[58] Vitor Guizilini, Jie Li, Rares Ambrus, Sudeep Pillai, and Adrien Gaidon. Robust semi-supervised monocular depth estimation with reprojected distances. In *Conference on robot learning*, pages 503–512. PMLR, 2020.

[59] Clément Godard, Oisin Mac Aodha, Michael Firman, and Gabriel J Brostow. Digging into self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3828–3838, 2019.

[60] Jae-Han Lee, Minhyeok Heo, Kyung-Rae Kim, and Chang-Su Kim. Single-image depth estimation based on fourier domain analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 330–339, 2018.

[61] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2485–2494, 2020.

[62] Benjamin Ummenhofer, Huizhong Zhou, Jonas Uhrig, Nikolaus Mayer, Eddy Ilg, Alexey Dosovitskiy, and Thomas Brox. Demon: Depth and motion network for learning monocular stereo. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5038–5047, 2017.

[63] Jiaming Sun, Linghao Chen, Yiming Xie, Siyu Zhang, Qinhong Jiang, Xiaowei Zhou, and Hujun Bao. Disp r-cnn: Stereo 3d object detection via shape prior guided instance disparity estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10548–10557, 2020.

[64] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE international conference on computer vision (ICCV)*, 2019.

[65] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: sparse-to-dense 3d object detector for point cloud. *ICCV*, 2019.

[66] Shaoshuai Shi, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[67] Jiageng Mao, Minzhe Niu, Haoyue Bai, Xiaodan Liang, Hang Xu, and Chunjing Xu. Pyramid r-cnn: Towards better performance and adaptability for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2723–2732, 2021.

[68] Jiageng Mao, Yujing Xue, Minzhe Niu, Haoyue Bai, Jiashi Feng, Xiaodan Liang, Hang Xu, and Chunjing Xu. Voxel transformer for 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3164–3173, 2021.

[69] Wu Zheng, Weiliang Tang, Sijin Chen, Li Jiang, and Chi-Wing Fu. Cia-ssd: Confident iou-aware single-stage object detector from point cloud. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 3555–3562, 2021.

[70] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 1195–1204, 2017.

[71] Charles R Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 918–927, 2018.

[72] Danfei Xu, Dragomir Anguelov, and Ashesh Jain. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253, 2018.

[73] Zhixin Wang and Kui Jia. Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection. In *IROS*. IEEE, 2019.

[74] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1907–1915, 2017.

[75] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven L Waslander. Joint 3d proposal generation and object detection from view aggregation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–8. IEEE, 2018.

[76] Ming Liang, Bin Yang, Yun Chen, Rui Hu, and Raquel Urtasun. Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7345–7353, 2019.

[77] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun. Deep continuous fusion for multi-sensor 3d object detection. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 641–656, 2018.

[78] Jin Hyeok Yoo, Yecheol Kim, and Ji Song Kim. 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection.

[79] Sourabh Vora, Alex H. Lang, Bassam Helou, and Oscar Beijbom. Pointpainting: Sequential fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[80] Tengteng Huang, Zhe Liu, Xiwu Chen, and Xiang Bai. Epnet: Enhancing point features with image semantics for 3d object detection. In *European Conference on Computer Vision*, pages 35–52. Springer, 2020.

[81] Felix Nobis, Maximilian Geisslinger, Markus Weber, Johannes Betz, and Markus Lienkamp. A deep learning-based radar and camera sensor fusion architecture for object detection. In *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–7. IEEE, 2019.

[82] Simon Chadwick, Will Maddern, and Paul Newman. Distant vehicle detection using radar and vision. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8311–8317. IEEE, 2019.

[83] Yizhou Wang, Zhongyu Jiang, Xiangyu Gao, Jenq-Neng Hwang, Guanbin Xing, and Hui Liu. Rodnet: Radar object detection using cross-modal supervision. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 504–513, 2021.

[84] Yizhou Wang, Zhongyu Jiang, Yudong Li, Jenq-Neng Hwang, Guanbin Xing, and Hui Liu. Rodnet: A real-time radar object detection network cross-supervised by

camera-radar fused object 3d localization. *IEEE Journal of Selected Topics in Signal Processing*, 15(4):954–967, 2021.

[85] Ramin Nabati and Hairong Qi. Radar-camera sensor fusion for joint object detection and distance estimation in autonomous vehicles. *arXiv preprint arXiv:2009.08428*, 2020.

[86] Ramin Nabati and Hairong Qi. Centerfusion: Center-based radar and camera fusion for 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1527–1536, 2021.

[87] Davi Frossard and Raquel Urtasun. End-to-end learning of multi-sensor 3d tracking by detection. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 635–642. IEEE, 2018.

[88] Erkan Baser, Venkateshwaran Balasubramanian, Prarthana Bhattacharyya, and Krzysztof Czarnecki. Fantrack: 3d multi-object tracking with feature association network. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1426–1433. IEEE, 2019.

[89] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris M Kitani. Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6499–6508, 2020.

[90] Hou-Ning Hu, Qi-Zhi Cai, Dequan Wang, Ji Lin, Min Sun, Philipp Krahenbuhl, Trevor Darrell, and Fisher Yu. Joint monocular 3d vehicle detection and tracking. In *Proceedings of the IEEE international conference on computer vision*, pages 5390–5399, 2019.

[91] Xinshuo Weng, Ye Yuan, and Kris Kitani. Joint 3d tracking and forecasting with graph neural network and diversity sampling. *arXiv preprint arXiv:2003.07847*, 2020.

[92] Ming Liang, Bin Yang, Wenyuan Zeng, Yun Chen, Rui Hu, Sergio Casas, and Raquel Urtasun. Pnpnet: End-to-end perception and prediction with tracking in the loop. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11553–11562, 2020.

[93] Wenjie Luo, Bin Yang, and Raquel Urtasun. Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 3569–3577, 2018.

[94] Ronald PS Mahler. Multitarget bayes filtering via first-order multitarget moments. *IEEE Transactions on Aerospace and Electronic systems*, 39(4):1152–1178, 2003.

[95] Ronald Mahler. Phd filters of higher order in target number. *IEEE Transactions on Aerospace and Electronic systems*, 43(4):1523–1543, 2007.

[96] Ba-Tuong Vo and Ba-Ngu Vo. Labeled random finite sets and multi-object conjugate priors. *IEEE Transactions on Signal Processing*, 61(13):3460–3475, 2013.

[97] Jason L Williams. Marginal multi-bernoulli filters: Rfs derivation of mht, jipda, and association-based member. *IEEE Transactions on Aerospace and Electronic Systems*, 51(3):1664–1687, 2015.

[98] Ángel F García-Fernández, Jason L Williams, Karl Granström, and Lennart Svensson. Poisson multi-bernoulli mixture filter: direct derivation and implementation. *IEEE Transactions on Aerospace and Electronic Systems*, 54(4):1883–1901, 2018.

[99] Luca Caltagirone, Mauro Bellone, Lennart Svensson, and Mattias Wahde. Lidar–camera fusion for road detection using fully convolutional neural networks. *Robotics and Autonomous Systems*, 111:125–131, 2019.

[100] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

[101] Xiaozhi Chen, Kaustav Kundu, Yukun Zhu, Andrew G Berneshawi, Huimin Ma, Sanja Fidler, and Raquel Urtasun. 3d object proposals for accurate object class detection. In *Advances in Neural Information Processing Systems*, pages 424–432, 2015.

[102] Jimmy Ren, Xiaohao Chen, Jianbo Liu, Wenxiu Sun, Jiahao Pang, Qiong Yan, Yu-Wing Tai, and Li Xu. Accurate single stage detector using recurrent rolling convolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5420–5428, 2017.

[103] Zhaowei Cai, Quanfu Fan, Rogerio S Feris, and Nuno Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. In *european conference on computer vision*, pages 354–370. Springer, 2016.

[104] Liang Xie, Chao Xiang, Zhengxu Yu, Guodong Xu, Zheng Yang, Deng Cai, and Xiaofei He. Pi-rcnn: An efficient multi-sensor 3d object detector with point-based attentive cont-conv fusion module. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 12460–12467, 2020.

[105] Ross Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[106] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[107] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[108] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[109] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *CVPR*, 2021.

[110] Wenwei Zhang, Zhe Wang, and Chen Change Loy. Multi-modality cut and paste for 3d object detection. *arXiv preprint arXiv:2012.12741*, 2020.

[111] Peiyun Hu, Jason Ziglar, David Held, and Deva Ramanan. What you see is what you get: Exploiting visibility for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11001–11009, 2020.

[112] Qi Chen, Lin Sun, Ernest Cheung, and Alan L Yuille. Every view counts: Cross-view consistency in 3d object detection with hybrid-cylindrical-spherical voxelization. *Advances in Neural Information Processing Systems*, 2020.

[113] Junbo Yin, Jianbing Shen, Chenye Guan, Dingfu Zhou, and Ruigang Yang. Lidar-based online 3d video object detection with graph-based message passing and spatiotemporal transformer attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11495–11504, 2020.

[114] Xinge Zhu, Yuexin Ma, Tai Wang, Yan Xu, Jianping Shi, and Dahua Lin. Ssn: Shape signature networks for multi-class object detection from point clouds. *ECCV*, 2020.

[115] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019.

[116] Su Pang, Daniel Morris, and Hayder Radha. Fast-clocs: Fast camera-lidar object candidates fusion for 3d object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 187–196, 2022.

[117] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[118] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[119] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[120] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2020.

[121] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. Transfusion: Robust lidar-camera fusion for 3d object detection with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1090–1099, 2022.

[122] Yue Wang and Justin M Solomon. Object dgcnn: 3d object detection using dynamic graphs. *Advances in Neural Information Processing Systems*, 34, 2021.

[123] Russell Stewart, Mykhaylo Andriluka, and Andrew Y Ng. End-to-end people detection in crowded scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2325–2333, 2016.

[124] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[125] Andrea Simonelli, Samuel Rota Bulo, Lorenzo Porzi, Manuel López-Antequera, and Peter Kontschieder. Disentangling monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1991–1999, 2019.

[126] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.

[127] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 913–922, 2021.

[128] Tai Wang, ZHU Xinge, Jiangmiao Pang, and Dahua Lin. Probabilistic and geometric depth: Detecting objects in perspective. In *Conference on Robot Learning*, pages 1475–1485. PMLR, 2022.

[129] Jun Wang, Shiyi Lan, Mingfei Gao, and Larry S Davis. Infofocus: 3d object detection for autonomous driving with dynamic information modeling. In *European Conference on Computer Vision*, pages 405–420. Springer, 2020.

[130] Ronald PS Mahler. *Advances in statistical multisource-multitarget information fusion*. Artech House, 2014.

[131] Katta G Murthy. An algorithm for ranking all the assignments in order of increasing costs. *Operations research*, 16(3):682–687, 1968.

[132] Patrick Emami, Panos M Pardalos, Lily Elefteriadou, and Sanjay Ranka. Machine learning methods for solving assignment problems in multi-target tracking. *arXiv preprint arXiv:1802.06897*, 2018.

[133] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3d object detection and tracking. *arXiv:2006.11275*, 2020.

[134] Shuyang Cheng, Zhaoqi Leng, Ekin Dogus Cubuk, Barret Zoph, Chunyan Bai, Jiquan Ngiam, Yang Song, Benjamin Caine, Vijay Vasudevan, Congcong Li, et al. Improving 3d object detection through progressive population based augmentation. *arXiv preprint arXiv:2004.00831*, 2020.

[135] Minh-Quan Dao and Vincent Frémont. A two-stage data association approach for 3d multi-object tracking. *Sensors*, 21(9):2894, 2021.