Object 3D Perception via Camera-Radar Cross-Modality Learning for Autonomous Driving

Yizhou Wang

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2022

Reading Committee:
Jenq-Neng Hwang, Chair
Hui Liu
Eli Shlizerman

Program Authorized to Offer Degree:
Electrical Engineering

University of Washington

Abstract

Object 3D Perception via Camera-Radar Cross-Modality Learning for Autonomous Driving

Yizhou Wang

Chair of the Supervisory Committee:
Professor Jenq-Neng Hwang
Electrical and Computer Engineering

Autonomous or assisted driving is increasingly feasible through the recent development of machine learning and deep learning community. As the autonomous vehicle is usually a multi-sensor platform, it is crucial to not only exploit the information from the different sensors but also effectively fuse the information to compensate for individual limitations under different driving scenarios. In this dissertation, we focus on two common and cost-efficient sensors on an autonomous vehicle, i.e., camera and radar, and manage to achieve an accurate and robust object perception strategy for autonomous or assisted driving purposes. Specifically, the camera can capture rich semantic information, while radar has reliable ranging and speed estimation capability. Thus, with a camera and radar, we can potentially achieve accurate and reliable object perception results, including 3D object detection and 3D object tracking. On the other hand, however, the camera itself is not a robust sensor under severe conditions, such as weak/strong lighting or bad weather. Whereas radar is relatively more reliable in most harsh environments, even though its capability of semantic understanding of scene contents is quite limited. Therefore, it is critical to propose a system that can also rely purely on the radar for semantic object detection, in the format of radio frequency (RF) images, under scenarios when the camera cannot provide reliable information.

To achieve the aforementioned goals, we start by collecting a new camera-radar dataset with various driving scenarios, named CRUW dataset, including camera, radar, and LiDAR

sensors. We adopt the RF image as our radar data representation for better radar data exploitation. We make our CRUW dataset public and set up a benchmark, named ROD2021, to help the community further develop the related algorithms.

As for the algorithms, we first develop the camera-radar cross-modality supervision algorithms for object 3D perception, including a camera-only (CO) object detection and 3D localization system to perform 3D localization of detected objects in the camera coordinates, and a camera-radar fusion (CRF) framework that takes advantage of the accurate ranging results from the radar to obtain more reliable 3D object detection. After that, to achieve radar-only object detection, we propose a radar object detection network (ROD-Net) that only takes RF image sequences as the input and estimates object confidence maps (ConfMaps). Moreover, to accomplish radar-based multi-object tracking (RadarMOT), we further propose the RadarMOT framework, which jointly predicts object detection and radar instance features for tracking.

After exploiting radar-only object perception tasks, we propose a camera-radar cross-modality check pipeline when we have perception results from both camera and radar sensor modalities. The cross-modality check pipeline can be divided into three stages. First, we conduct detection alignment between the camera and radar through a proposed camera-radar bilateral coordinate projection (BCP). Second, for the aligned detections, we conduct alignment refinement to achieve geometrical consistency. Third, for unaligned detection, we introduce an alignment verification stage by considering temporal continuity.

Overall, the contributions of this dissertation can be concluded as follows:

- An accurate and robust object perception system via camera-radar cross-modality learning for autonomous or assisted driving applications.
- A new dataset, named CRUW, containing synchronized camera-radar frames, is collected and can serve as a valuable dataset for camera-radar cross-modality research.

- A novel and robust radar object detection network, called *RODNet*, for robust object detection in severe driving scenarios, which can be used for adverse autonomous or assisted driving without camera or LiDAR information.
- A radar multi-object tracking (RadarMOT) method that can reliably track objects with deep features from the RF images.
- A reliable camera-radar cross-modality check pipeline that can accurately detect objects when both camera and radar are present, considering geometrical consistency and temporal continuity.

In conclusion, this dissertation is aimed to explore a "camera+radar" solution for object 3D perception in autonomous driving applications. The overall system has great robustness and is also accurate and cost-efficient compared with camera or LiDAR based solutions. Potentially, our solution can become a cheap and portable autonomous or assisted driving solution in the future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

Time flies. After more than four years of Ph.D. studies, I accomplished my Ph.D. dissertation with some minor but potentially useful contributions to the autonomous driving community about object 3D perception using camera and radar. During the whole research, I learned a lot from my professors, mentors, colleagues, and friends. Here, I would express my sincere appreciation to all of them.

First, I wish to thank my Ph.D. advisor, Prof. Jenq-Neng Hwang, for his kind and valuable advice during my Ph.D. studies. Prof. Hwang can always provide insightful ideas and suggestions to me on both my research and career. With his guidance, I started to know deeper about research, critical thinking, and even presentation skills. I also learned from his meticulous personality to become a good researcher and engineer in the future.

Second, I would thank my Ph.D. committee members, Prof. Hui Liu, Prof. Eli Shlizerman, Prof. Xuegang Ban, and Prof. Yasuo Kuga for the valuable discussion and suggestions on my Ph.D. dissertation. The insightful questions raised during my final defense are also very informative and make me think of more potentials and challenges in my future research.

Third, I would also thank the sponsors during my Ph.D., including Silkwave Holdings Limited, ETRI, and CISCO. The projects helped me dive deeper into my research and explore some more realistic industrial applications or products.

Besides, I would also like all of the co-authors and collaborators during my research, especially the colleagues at the Information Processing Lab (IPL). Many IPL senior members, including Dr. Hung-Min Hsu, Dr. Gaoang Wang, Dr. Zheng Tang, Dr. Tsung-Wei Huang, Dr. Renshu Gu, and Dr. Li Chen helped me a lot during the early stage of my Ph.D. studies. Since working on similar topics about autonomous driving and 3D perception, Dr. Haotian

Zhang can always have great discussions on research projects with me and come up with some interesting ideas. Dr. Jiarui Cai is another IPL colleague who had a close collaboration with me. I learned some long-tailed and open-set knowledge during co-authoring several papers. I also collaborate with Zhongyu Jiang, Chris Yang, Andy Cheng, Yudong Li, etc. Thank them for their contributions to the projects and best wishes to their future Ph.D. studies or careers. ithout all the above brainstorming and collaboration, I cannot achieve today's progress.

Moreover, I would thank my family for their support during my Ph.D. Without their support, I cannot start my graduate studies in the United States. Whenever I met some difficulties, my family always support me and give me the courage to overcome them.

Finally, I would thank the University of Washington, where I had the opportunity to conduct cutting-edge research and education resources.

Chapter 1

INTRODUCTION

1.1 Motivations and Objectives

Object perception is crucial for autonomous or assisted driving applications. With the commonly used sensors, i.e., camera, LiDAR and radar, the objects around the autonomous vehicle can be potentially detected and tracked accurately and robustly. The camera can provide rich semantic information, e.g., object detection and tracking. But its depth/3D geometry estimation is not reliable. It is neither robust to lighting nor robust to adverse weather conditions. The LiDAR can collect dense 3D point clouds, but it is usually expensive, hard to install, and not robust to adverse weather conditions. The radar has great capability of distance and speed estimation and is robust to all-weather conditions. However, it is usually unintuitive, over-sensitive, and has low angle resolution.

In this dissertation, we consider a **"camera+radar"** object 3D perception system that is able to handle 3D information analysis and is even more robust than LiDAR under adverse weather conditions. Therefore, we think that camera+radar is potentially a more robust and cost-efficient solution, and is significantly useful with high academic and industrial potential.

The objectives of this dissertation can be concluded as solving the following two significant questions in the autonomous driving community:

- When camera and radar are both available (e.g., in normal driving scenarios), how can we take advantage of both sensors for object 3D perception?

- When the camera is not reliable (e.g., in adverse driving scenarios), can radar accomplish the object perception tasks itself?

These two questions can be considered as the core problems for our "camera+radar" object 3D perception system.

In the later part of this section, we will start with some radar background knowledge and describe the methodology for how we utilize autonomous radar data. After that, we will introduce our overall "camera+radar" system pipeline before diving into the details in the later chapters.

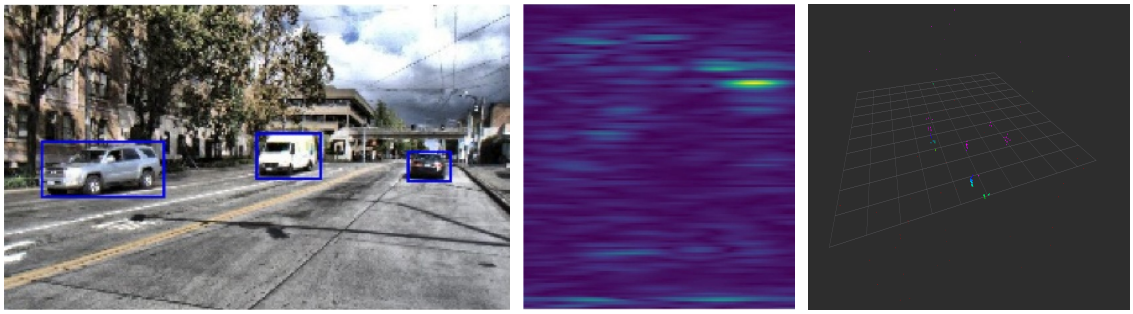
1.2 Radar Background Knowledge

Frequency modulated continuous wave (FMCW) radar, which operates in the millimeter-wave (MMW) band (30-300GHz) that is lower than visible light, has the following properties: 1) MMW has great capability to penetrate through the fog, smoke, and dust; 2) The huge bandwidth and high working frequency give FMCW radar great range detection ability.

1.2.1 Radar Data Representations

There are usually two different kinds of radar data representations, i.e., **radar points** and **radio frequency (RF) images**, as shown in Figure 1.1. Radar points are more frequently used for obstacle ranging and speed estimation in autonomous driving since the ranging and speed can be directly inferred from the raw radar data through Fast Fourier Transform (FFT) and adaptive peak thresholding and clustering [75]. However, radar points from an mmWave radar are usually very sparse with relatively low angle resolution, especially compared with LiDAR [11, 26]. Thus, a large amount of useful semantic information is missing using this kind of representation.

Actually, radar is feasible for semantic understanding, e.g., object classification, detection, and tracking, owing to the hidden ***phase*** information inside the radio frequencies. Typically, radar's signal amplitude is commonly used to estimate the distance and speed of the obstacles, while the phase information is usually not well-utilized because of its "non-intuitiveness", making it difficult to be interpreted by the classical signal processing mechanisms.



(a) RGB Image (b) RF Image (c) Radar Points

Figure 1.1: Visualization among (a) camera RGB image, (b) radar RF image, and (c) radar points. Compared with the RGB image, the semantic information in the RF image is implicit and hard to extract. Radar points are usually very sparse and do not contain semantic information.

1.2.2 Radar Data Processing

We illustrate the details of our radar data processing steps, which can be divided into two parts, i.e., RF images in range-azimuth (RA) coordinates to localize and classify the objects in the BEV, and range-azimuth-Doppler (RAD) coordinates to obtain the relative radial speed information.

RF-RA images This process is the same as the pre-processing mentioned in [94]. RF images in radar range-azimuth coordinates can be described as a bird's-eye view (BEV) representation, where the x-axis denotes azimuth (angle) and the y-axis denotes range (distance). From the raw radar data, we first implement a range FFT on the received chirp samples to estimate the range of the reflections. After that, we conduct a second angle FFT on the samples along different receiver antennas to estimate the azimuth angle of the reflections. An example RF-RA image is shown in Figure 1.3 (c). After being transformed into RF images, the radar data is represented as a complex-valued 2D format (with real and imaginary channels).

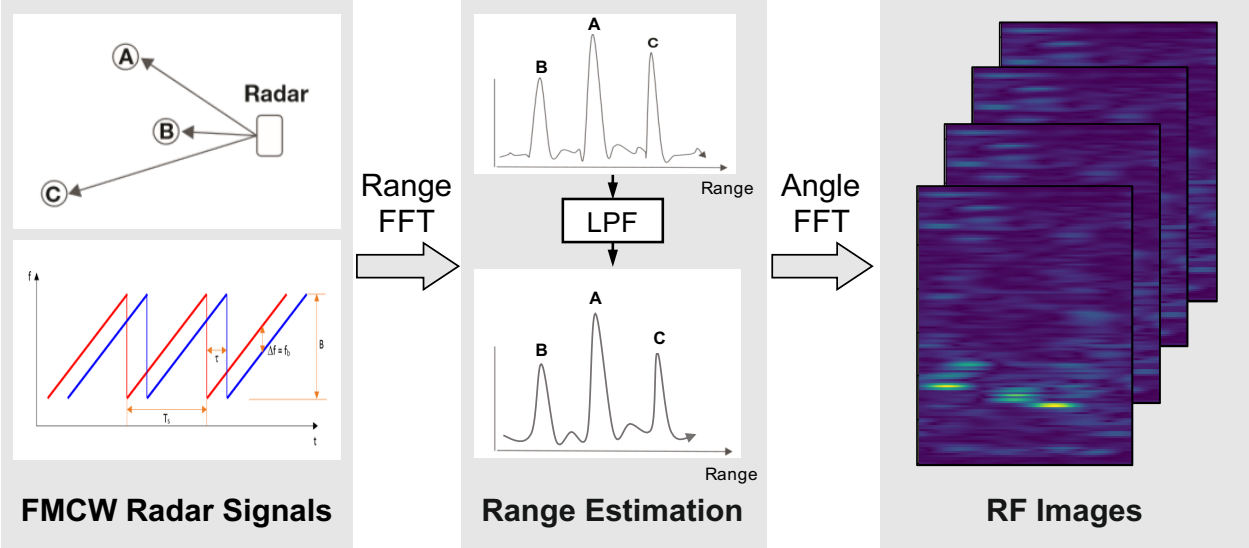
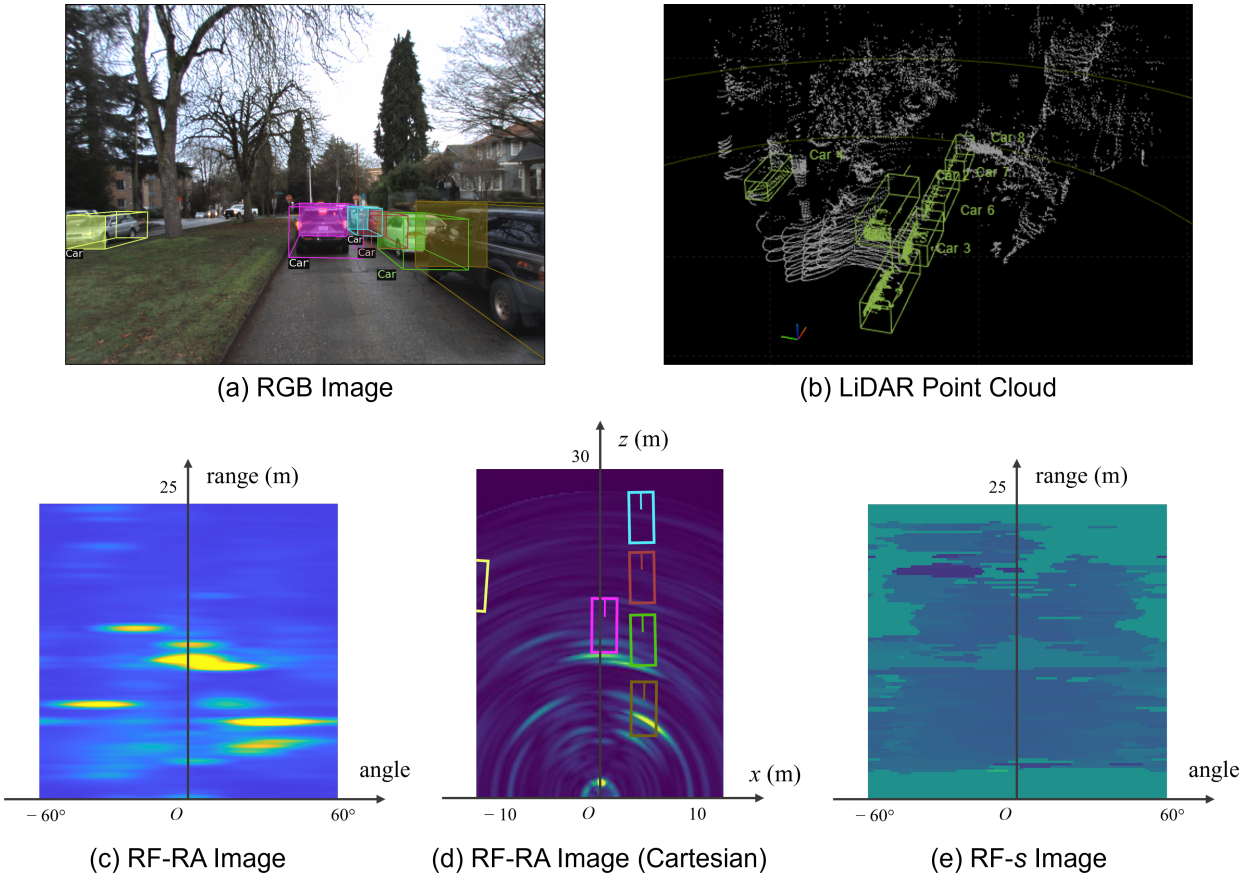


Figure 1.2: The workflow of the RF image generation from the raw radar signals.

RF-RA images in Cartesian coordinates In order to better associate the data among different sensor modalities, we also transform RF-RA images into Cartesian coordinates, as shown in Figure 1.3 (d). We first generate a Cartesian grid for the target RF image, and map each grid location to the polar coordinates. The value at a certain location is obtained by bilinear interpolation from the original RF-RA images. Note that, due to the continuity of amplitude and phase in RF-RA images, we conduct the interpolation on the amplitude and phase parts for each complex pixel value.

RF-RAD images Besides the above RF images in the range-azimuth coordinates, to obtain the speed information from radar, we further process the raw radar data into RF-RAD images. First, same as the RF-RA pre-processing procedure, we use the range FFT to estimate the range of the reflections. Then, a Doppler FFT is implemented to estimate the speed at each range grid. Afterward, the angle FFT is appended to estimate the azimuth angle. Here, we will get a 3D tensor in the RAD coordinates. In order to get the relative radial speed between the object and the ego-car, we select the speed grid with the greatest



(a) RGB Image (b) LiDAR Point Cloud

(c) RF-RA Image (d) RF-RA Image (Cartesian) (e) RF-*s* Image

Figure 1.3: Visualization for multi-modality data processing, including RGB image, LiDAR point cloud, RF-RA image, RF-RA image in Cartesian coordinates, and RF-s image for relative radial speed map.

amplitude value along the Doppler axis. We call this resulting tensor the speed map RF-s, and each element in RF-s represents the relative radial speed at a certain range and angle location. An example RF-s image is shown in Fig. 1.3(e).

1.2.3 Radar Properties

Moreover, radio frequency data have the following special properties to be handled for object detection tasks.

- **Rich motion information.** According to the Doppler principle of the radio signal,

rich motion information is included. The objects' speed and their law of variation over time are dependent on their surface texture information, size, shape, etc. For example, the motion information of a non-rigid body, like a pedestrian, is usually widely distributed, while for a rigid body, like a car, it should be more consistent due to the Doppler effect. In order to better utilize the temporal information, we need to consider multiple consecutive radar frames, instead of one single frame, as the system input.

- **Inconsistent resolution.** Radar usually has high-resolution (HR) in range but low-resolution (LR) in azimuth angle due to the limitation of radar hardware specifications, like the number of antennas and the distances among them.
- **Complex numbers.** Radio signals are usually represented as complex numbers containing frequency and phase information. This kind of data is unusual to be modeled by a typical CNN architecture.

1.3 Overview

The whole dissertation is organized as follows. We first go through some related research works on camera-based perception, radar-based perception, cross-modality and sensor fusion, and radar datasets in Chapter 2. Then, we will introduce our self-collected CRUW dataset for our follow-up research works in Chapter 3. After that, we will describe our proposed algorithms in Chapter 4 and Chapter 5. Finally, we will draw a conclusion in Chapter 6.

The overall pipeline of our proposed system is shown in Figure 1.4. The proposed object 3D perception system can be divided into two parts.

The first one is called **cross-modality supervision** (Chapter 4). It considers the camera as a teacher, and the radar as a student, and aims to let the radar handle the object perception task itself after being cross-supervised by the camera. Here, we consider two radar-based object perception tasks, i.e., object detection and multi-object tracking. For radar-based object detection, we propose the RODNet introduced in Section 4.3, and for radar-based

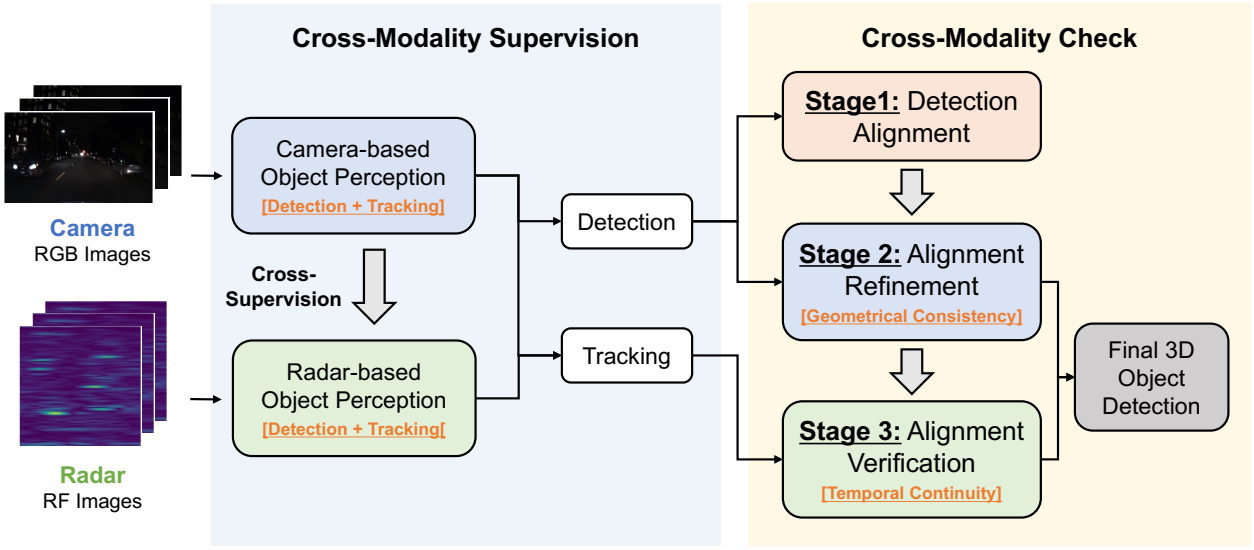


Figure 1.4: The overall pipeline of our "camera+radar" object 3D perception system. The proposed system takes the camera RGB images and the radar RF images as input. They then go through the modality-specific object perception methods, respectively, and obtain the detection and tracking results. Next, the detection and tracking results from both camera and radar pass through a three-stage cross-modality check module for detection alignment, refinement, and verification.

multi-object tracking, we propose the RadarMOT in Section 4.4.

The second section is called **cross-modality check** (Chapter 5). It jointly considers detection and tracking results from camera and radar sensors. The cross-modality check pipeline includes three main stages: 1) Detection alignment between camera and radar; 2) Alignment refinement for the aligned detections (i.e., the objects can be detected by both camera and radar); 3) Alignment verification for the non-aligned detections (i.e., the objects can be detected by only one of the sensors). After the cross-modality check, we would get refined 3D object detection with geometrical consistency and temporal continuity.

Chapter 2

RELATED WORKS

2.1 Camera-Based Perception Tasks

Image-based object detection [31, 74, 36, 12, 54, 73, 52, 24, 106], which is fundamental and crucial for many computer vision applications, is aimed to detect every object with its class and precise bounding box location from RGB images. Region-based CNN (R-CNN) is one of the fundamental series of works, including R-CNN [32], Fast R-CNN [31], Faster R-CNN [74], and etc. This kind of work is also called the two-stage method because it divides object detection into two stages. The first stage is to extract region proposals and the second stage is to classify and further refine the localization of the objects. Another stream of image-based object detection is called the one-stage method, which eliminates the region proposal estimation in the two-stage methods and makes predictions directly on a grid [54, 72, 73, 52, 24, 106]. For example, CenterNet [106] predicts heatmap drawn from the ground truth bounding box centers without considering region proposals. It also has two additional prediction heads, i.e., the bounding box dimension head and the bounding box offset head, to predict regression values for box dimensions and offsets.

Multi-object tracking (MOT) has been intensively studied in the computer vision community since the 2010s for crowd detection and tracking [27, 49, 58, 21]. Tracking-by-detection, the most common scheme for recent MOT methods [77, 25, 44, 59, 86, 88, 91, 87], which focuses on exploiting the association between the objects in consecutive frames, given the object detections. Some of the methods track objects by feature fusion [77], which considers both appearance and temporal motion features from CNNs or some classical feature extractors. Moreover, graph model based tracking-by-detection methods achieve favorable performance because of their effectiveness and robustness [59, 86, 88, 91]. Besides, joint ob-

ject detection and appearance feature generation for MOT schemes becomes more popular [8, 97, 105, 102]. More specifically, Tracktor [8] conducts bounding box regression using the detections from the previous frame as region proposals to refine the box positions. Joint detection and embedding (JDE) [97] incorporates the appearance feature prediction task into a detector to predict detections and the corresponding embeddings simultaneously. FairMOT [102] proposes an anchor-free architecture for both detection and re-id feature prediction via multi-task learning.

Object 3D localization has attracted great interest in autonomous and assisted driving communities [81, 82, 60, 61, 6]. One idea is to localize vehicles by estimating their 3D structures using a CNN, e.g., 3D bounding boxes [60] and 3D keypoints [61, 6, 46]. Then, a pre-defined 3D vehicle model is used to estimate the deformations, resulting in accurate 3D keypoints as well as the vehicle location. Another idea [81, 82], however, tries to develop a real-time monocular structure-from-motion (SfM) system, taking into account different kinds of cues, including SfM cues (3D points and ground plane) and object cues (bounding boxes and detection scores). Although these works achieve favorable performance in object 3D localization, they only work for the vehicles since only the rigid-body vehicle structure is assumed.

Object 3D detection [92, 55, 67] predicts objects as 3D bounding boxes, including 3D location, dimension, and orientation. FCOS3D [92], built on CenterNet [106], uses 2D Gaussian distribution on the projected 3D center to fit the 3D target formulation. It is a simple yet effective single-stage detector that gets rid of any 2D detections or 2D-3D correspondence priors. SMOKE [55] is a single-stage 3D object detection method also based on CenterNet [106]. Given an input image, it detects targeted objects' 3D centers projected on the image plane. DD3D [67] uses a large-scale depth dataset DDAD15M [34] to pre-train the network to obtain better depth-aware features from images, which achieves state-of-the-art among monocular 3D object detection methods.

2.2 Radar-Based Perception Tasks

Significant research in radar object classification has demonstrated its feasibility as a good alternative when cameras fail to provide good sensing performance [38, 5, 15, 48, 13, 69, 68]. With handcrafted feature extraction, Heuel, *et al.* [38] classify radar objects using a support vector machine (SVM) to distinguish cars and pedestrians. Moreover, Angelov *et al.* [5] use a neural network to extract features from the short-time Fourier transform (STFT) heatmap of radar signals. However, the above methods only focus on the *classification* tasks, which assume only one object has been appropriately identified in the scene. Recently, a radar object detection method is proposed in [28], which combines a statistical constant false alarm rate (CFAR) [75] detection algorithm with a CNN-based VGG-16 classifier [80]. All of the above approaches are not applicable to complex driving scenarios with noisy background reflections, e.g., trees, buildings, traffic signs, etc., and could easily give many *false positives*. Besides, the laborious human annotations on the radar RF images required by these methods are usually impossible to obtain.

Classical radar tracking is commonly regarded as a point-based tracking problem, which can be modeled as a dynamic system represented by states in a discrete-time domain. A Kalman filter [43, 42] assumes a linear state transition function and a linear observation model, with additive Gaussian noises, to obtain optimal state predictions and update formulations. Kalman filtering is based on restricted assumptions and can therefore achieve computational simplicity, which is a desired property for real-time online tracking systems. To achieve radar tracking for autonomous driving applications, some radar point based tracking methods have been proposed [18, 78, 35, 4]. Scheel et al. [78] propose a learnable variational radar model directly from the actual radar detection to track multiple vehicles. Akita et al. [4] propose a classification and tracking method for moving objects in the parking lot. Here, the tracking part is to simply associate the detected objects with the nearest neighbor algorithm and classify the tracked objects by a long short-term memory (LSTM) network. Haag et al. [35] introduce an interacting multiple motion model to describe nonlinear object

motions. However, these existing works either track the radar points without distinguishing background noises or object classes, or take detection results from advanced image-based object detection methods as the input.

2.3 Cross-Modality and Sensor Fusion

Recently, the concept of cross-modal learning has been proposed in the machine learning community [45, 89, 71, 41]. This concept is trying to transfer or fuse the information between two different signal modalities in order to help train the neural networks. Specifically, RF-Pose [104] introduces the cross-modal supervision idea into wireless signals to achieve human pose estimation based on WiFi range radio signals, using a computer vision technique, i.e., OpenPose [14], to systematically generate annotations of human body keypoints from the camera. However, radar object detection is more challenging: 1) Feature extraction for object detection (especially for classification) is more difficult than human joint detection which could just classify different joints by their relative locations without considering object motion and texture information; 2) The typical FMCW radars on the vehicles have much less resolution than the WiFi array sensors used in RF-Pose.

As for autonomous or assisted driving applications, Major et al. [56] propose an automotive radar based vehicle detection method using LiDAR information for cross-modal learning. However, our work is different from theirs: 1) They only consider vehicles as the target object class, while we detect pedestrians, cyclists, and cars; 2) The scenarios in their dataset are mostly highways without noisy obstacles, which is easier for radar object detection, while we are dealing with much more diverse driving scenarios. Palffy et al. [66] propose a radar based, single-frame multi-class object detection method. However, the annotation method is a semi-automatic stereo camera based algorithm, which needs significant human effort, and is not applicable for large-scale datasets. Besides, they only consider the data from a single radar frame, which is not reliable and does not involve object motion information.

Nabati et al. [63] propose an object perception called CenterFusion that exploits both radar and camera data for 3D object detection in a middle-fusion scheme. This method

first detects object centers on the image and association radar detections with the camera detections. The associated features are then concatenated with the image features to regress the object properties.

2.4 Related Datasets

Autonomous driving datasets have attracted great attention for deep learning based object perception methods. The KITTI dataset [30] is the first complete autonomous driving dataset, which includes stereo cameras and a LiDAR, with various annotations. Recently, larger-scale and more advanced datasets are available, e.g., BDD100K [99], nuScenes [11], ApolloScape [40], Waymo Open [83]. However, due to the hardware compatibility and less developed radar perception techniques, most datasets do not incorporate radar signals as a part of their sensor systems.

Among the available radar datasets, some of them, e.g., nuScenes [11], HiRes2019 [57], RadarRobotCar [7], RADIARE [79], etc., consider radar data in the format of radar points that do not contain the useful Doppler and surface texture information of objects for semantic understanding. Other researchers focus on using the RF image as the radar data format. More specifically, some manage to collect a dataset with the camera, radar, and LiDAR, and annotate the objects as 3D bounding boxes based on the dense point cloud from LiDAR [56, 23]. Others consider the camera-radar solution without a LiDAR [65, 66], whose annotation format is usually in pixel or point level. However, most of the datasets are not publicly available, as shown in Table 2.1.

After extensive research on the available datasets, we cannot find a suitable one that includes large-scale radar data in RF image format with labeled ground truth. Therefore, we collect our own CRUW dataset which will be introduced in Chapter 3.

Table 2.1: Comparison with related datasets with radar sensor by modality, data format, scenario, etc. Our datasets can fill the gap among these related datasets with RF-based radar data format.

Dataset	Year	Modality[1]	Radar[2]	Scenarios	Scale	# of Classes	Annotations	Public
nuScenes [11]	2019	C/R/L	RP	combined	5.5 hours	23	3D Box + Track	✓
HiRes2019 [57]	2019	C/R/L	RP	normal	546 frames	7	3D Box	✓
Qualcomm [56]	2019	C/R/L	RF	normal	3 hours	1	3D Box	✗
Xsense.ai [23]	2020	C/R/L	RF	normal	34.2 min	1	3D Box	✗
RADIATE [79]	2020	C/R/L	RP	combined	3 hours	7	2D Box	✓
CARRADA [65]	2020	C/R	RF	normal	21.2 min	3	Pixel	✓
RTCnet [66]	2020	C/R	RF	normal	1 hour	3	Point	✗
RADDet [101]	2021	C/R	RF	normal	10K frames	6	2D Box	✓
CRUW (Ours)	2020	C/R	RF	combined	3.5 hours	3	Point	✓
ROD2021 (Ours)	2021	C/R	RF	combined	28 min	3	Point	✓
CRUW2022 (Ours)	2022	C/R/L	RF	combined	40 min	5	3D Box + Track	✓

[1] Modalities: "C" for camera, "R" for radar, "L" for LiDAR.
[2] Radar data formats: "RP" for radar points, "RF" for radio frequency (RF) images.

Chapter 3

CRUW DATASET

3.1 The CRUW Dataset for Cross-Modality Supervision

3.1.1 Sensor System and Data Description

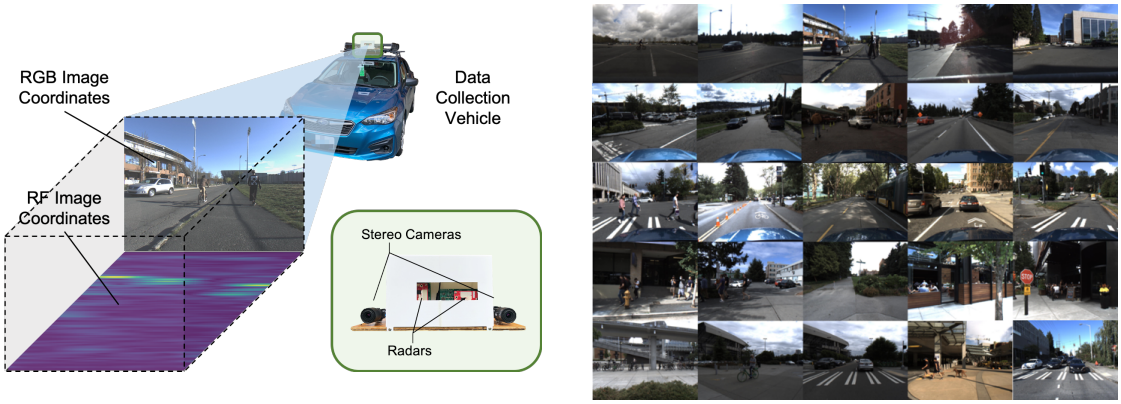
We collect a new dataset, named Camera-Radar of the University of Washington (CRUW), which uses the format of RF images for the radar data, as mentioned in Section 1.2. Our sensor platform contains a pair of stereo cameras [1] and two perpendicular 77GHz FMCW MMW radar antenna arrays [2]. The sensors, assembled and mounted together as shown in Fig. 3.1 (a), are well-calibrated and synchronized. Some configurations of our sensor platform are shown in Table 3.1.

Table 3.1: Sensor Configurations for CRUW Dataset.

Camera	Value	Radar	Value
Frame rate	30 FPS	Frame rate	30 FPS
Pixels (H×W)	1440×1080	Frequency	77 GHz
Resolution	1.6 MegaPixels	# of transmitters	2
Field of View	93.6°	# of receivers	4
Stereo Baseline	0.35 m	# of chirps per frame	255
		Range resolution	0.23 m
		Azimuth resolution	∼15°

The CRUW dataset[1] contains 3.5 hours with 30 FPS (about 400K frames) of camera-radar data in different driving scenarios, including campus road, city street, highway, and parking lot. Some sample scenarios are shown in Fig. 3.1 (b). Besides, we also collect several vision-hard sequences of poor image quality, i.e., weak/strong lighting, blur, etc.

[1]The CRUW dataset website: `http://cruwdataset.org/`

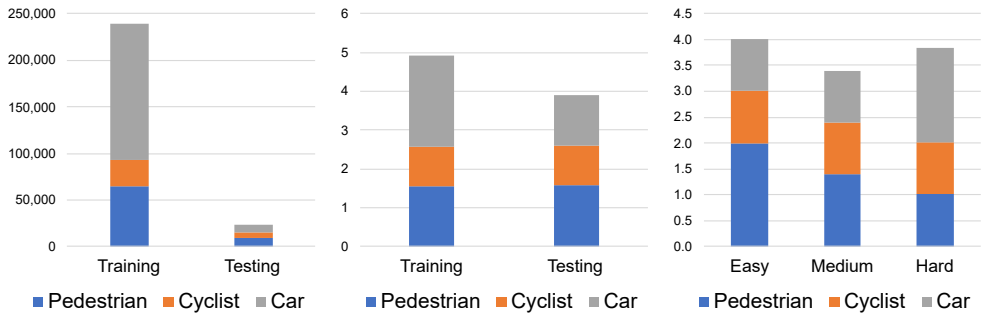


(a) CRUW dataset collection (b) Different scenes in the dataset

Figure 3.1: The sensor platform and driving scenarios for our CRUW dataset.

3.1.2 Data Distribution

The data distribution of CRUW is shown in Fig. 3.2. The object statistics in (a)-(c) only consider the objects within the radar's field of view (FoV), i.e., 0-25m, ±90°, based on the current hardware capability. There are about 260K objects in the CRUW dataset in total, including 92% for training and 8% for testing. The average number of objects in each frame is similar between training and testing data.



(a) # of objects in total (b) # of objects per frame (c) # of objects per frame in the testing set

Figure 3.2: The CRUW dataset object distribution in the radar's FoV (0-25m, ±90°).

Table 3.2: Driving scenarios statistics for CRUW dataset.

Scenarios	# of Seqs	# of Frames	Vision-Hard %
Parking Lot	124	106K	15%
Campus Road	112	94K	11%
City Street	216	175K	6%
Highway	12	20K	0%
Overall	464	396K	9%

The four different driving scenarios, i.e., parking lot, campus road, city street, and highway, are shown in Table 3.2 with the number of sequences, frames, and vision-hard percentages. From each scenario, we randomly select several complete sequences as testing sequences, which are not used for training. Thus, the training and testing sequences are captured at different locations and at different times. For the ground truth needed for evaluation purposes, 10% of the visible and 100% of the vision-hard data are human-labeled by manual refinement on the results of the annotator introduced in Section 4.2.

3.2 The ROD2021 Dataset and Challenge

In the ROD2021 Challenge, we select 50 sequences in the CRUW dataset, including 40 for training and 10 for testing, under four different driving scenarios, i.e., parking lot (PL), campus road (CR), city street (CS), and highway (HW). The data distribution of the ROD2021 dataset is shown in Table 3.3. From each scenario, we randomly split training and testing sequences, which are captured at different locations and at different times. In the testing set, we try to make the number of frames for each scenario similar to make the evaluation balanced on different scenarios. Besides, we also have several vision-hard sequences of poor image quality, i.e., weak/strong lighting, blur, etc. These data are only used for testing/evaluation purposes to illustrate the radar's robustness compared with the camera.

In this challenge, the training set contains both RGB and RF image sequences, while the testing set only contains radar RF image sequences. We provide the annotations on the RF images for all the training data, and the participants are also allowed to use both RGB and

RF images during the training stage. But the RF images are the only allowed input data in the testing stage.

Table 3.3: Driving scenarios statistics in the ROD2021 dataset.

Scenarios	Training		Testing	
	Sequences	Frames	Sequences	Frames
Parking Lot	25	22480	3	2700
Campus Road	9	8100	3	2700
City Street	3	5080	3	3354
Highway	3	5074	1	1683
Overall	40	40734	10	10437

The ROD2021 Challenge is the first radar object detection challenge based on radio frequency (RF) images for autonomous driving applications. The challenge attracts more than 260 participants from worldwide academic and industrial affiliations. The top teams achieve favorable radar object detection results using different proposed networks or advanced techniques, e.g., more efficient network architectures, data augmentation, model ensemble, scene classification, etc. The challenge appears to achieve the objectives of attracting the autonomous driving community to push the boundaries of the state-of-the-art on a practical and crucial problem in real-world autonomous driving solutions.

3.3 The CRUW2022 Dataset with 3D Box and Tracking Annotations

We introduce a new dataset, named CRUW2022, containing 66K synchronized camera, radar, and LiDAR data frames, under various driving scenarios, with object 3D bounding box and trajectory annotations (some examples are shown in Figure 3.3). Note that we also include a LiDAR in our dataset because our object 3D bounding box annotations are labeled based on LiDAR point clouds and projected to camera and radar coordinates, respectively, based on our sensor calibration results.

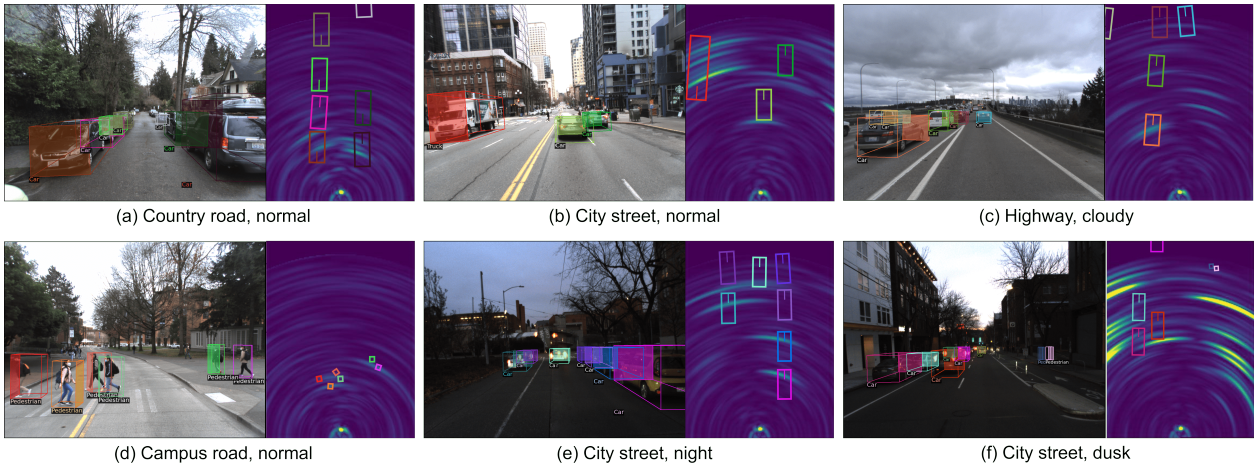


Figure 3.3: Examples of the CRUW2022 dataset. Each example contains a camera RGB image and a radar RF image pair. The RF images are transformed to Cartesian coordinates for better visualization. We include data examples under different driving scenarios and lighting conditions. The corresponding 3D bounding box annotations are projected to RGB and RF images, respectively.

3.3.1 Data Collection

We propose a dataset collection pipeline with stereo cameras, mmWave radar, and a LiDAR, including a sensor platform, a data collection software, and a sensor calibration method. With our proposed pipeline, the data collected from three sensor modalities can be temporally synchronized and spatially calibrated accurately.

Sensor Platform Our dataset collection sensor system is shown in Figure 3.4. There are two FLIR BFS-U3-16S2C-CS cameras, one TI AWR1843 radar board, and one Livox Horizon LiDAR. The detailed specifications are listed in Table 3.4.

Sensor Synchronization Our dataset collection software is based on Robot Operating System (ROS) under Ubuntu. For cameras and LiDAR, since they provide open-source API,

[2]The frame rate is after the point cloud integration to scan over LiDAR's field of view (FOV). Details introduced in Section 3.3.2.

[3]Better radar performance and resolution within $\pm 60°$.

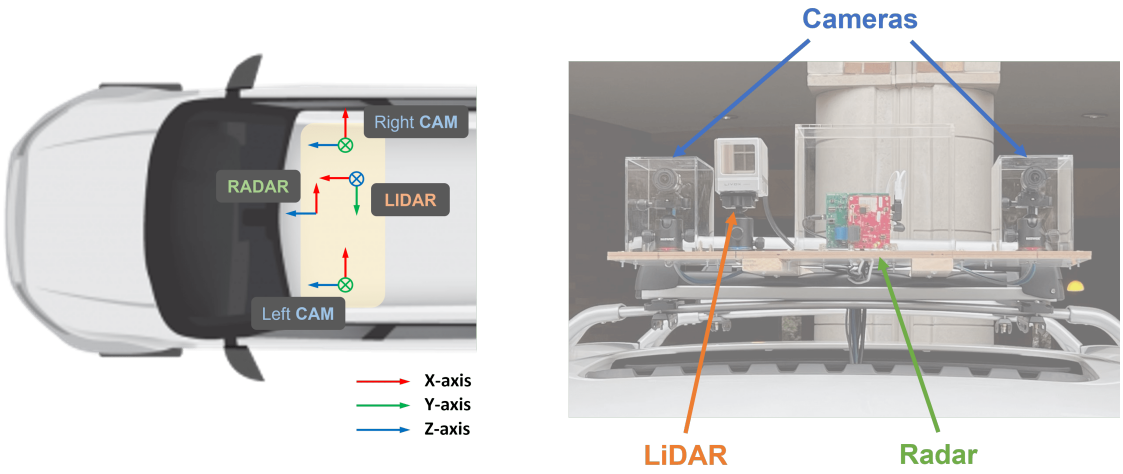


Figure 3.4: Sensor coordinates and sensor platform of our CRUW2022 dataset, including two cameras, one mmWave radar, and one LiDAR. Note that our radar does not have elevation angle resolution (i.e., y-axis), thus, it is equivalent to the BEV of the camera after applying a translation vector between the two sensors.

Table 3.4: Sensor configurations for CRUW2022 dataset.

Cameras	Value	Radar	Value	LiDAR	Value
Frame Rate	30 FPS	Frame Rate	30 FPS	Frame Rate	10 FPS[2]
Pixels (W×H)	1440×1080	Frequency	77 GHz	Point Rate	240,000 pts/s
Resolution	1.6 MP	# of Transmitters	2	Detection Range	260 m
Field of View	93.6°	# of Receivers	4	Range Precision	2 cm
Stereo Baseline	0.6 m	# of Chirps per Frame	255	Field of View	81.7° × 25.1°
		Max Range	28 m	Angular Precision	0.05°
		Range Resolution	0.23 m		
		Min & Max Angle	±90°[3]		
		Azimuth Resolution	∼15°		

we directly integrate them into the ROS system. However, TI only provides software based on Windows and MATLAB. Therefore, we create a Windows virtual box in our Ubuntu system and communicate different processes through ROS. We set up hardware time synchronization between cameras and LiDAR using Transistor-Transistor Logic (TTL) signal generated by the right camera. Both camera and LiDAR sensors support TTL signal time synchronization through their APIs. On the software level, we use the ApproximateTime synchronization policy provided by the ROS library to align three sensors' data into 30 FPS time slots. To synchronize between radar and other sensors, we use a software trigger to start a data sequence collection. A service client triggers the collection process of radar data and starts another collection process of other sensor data upon receiving the response. From our experiments, the latency of the software trigger is under a few milliseconds, which is negligible. More details of our data collection system are described in the supplementary document.

Sensor Calibration First, we calibrate the stereo cameras using Zhang's method [103], which gives us the intrinsic parameters, distortion coefficients, and extrinsic parameters of the two cameras. These results are used later for stereo rectification in Section 3.3.2. For the sensor calibration between cameras and LiDAR, we adopt the calibration algorithm proposed by Dhall et al. [22]. This will give us two transformation matrices, representing the transformation between the left camera and LiDAR, and between the right camera and LiDAR, respectively. As for radar, which is carefully mounted and aligned with cameras and LiDAR according to their pitch angle, its coordinates are parallel to the camera's bird's-eye view (BEV). The translation vectors between the sensors are also measured to form the full transformation matrices between cameras and radar.

3.3.2 Data Processing

Camera Data Processing The image sequences captured by the stereo cameras are first undistorted and rectified based on the camera calibration. Then, for the low-quality images

due to adverse lighting conditions, we conduct image enhancement to improve the quality and lighting stability of the collected videos. Here, we implement a deep learning based method, named RRDNet [107], to restore the underexposed image in zero-shot using a three-branch CNN. In order to achieve stable enhancement results for video sequences, we train the network using only the first frame of each sequence, and do inference on the rest frames.

Radar Data Processing Our radar data processing is similar to the pre-processing mentioned in [94], where RF images in radar range-azimuth coordinates are described as a bird's-eye view (BEV) representation, where the x-axis denotes azimuth (angle) and the y-axis denotes range (distance). From the raw radar data, we first implement a range fast Fourier transform (FFT) on the received chirp samples to estimate the range of the reflections. After that, we conduct a second angle FFT on the samples along different receiver antennas to estimate the azimuth angle of the reflections. Besides, we also transform the RF images into Cartesian coordinates for better alignment with the camera and more clear visualization.

LiDAR Data Processing Livox LiDAR has a special laser scanning technology called non-repetitive horizontal scanning, which is significantly different from the repetitive linear scanning offered by most traditional LiDAR sensors. It accumulates the points captured inside the FOV to get denser point clouds within an integration time window. However, based on this technology, the point cloud from LiDAR cannot cover the whole FOV within a camera frame (i.e., $1/30$ seconds). To ensure every camera/radar frame has a corresponding LiDAR frame for annotation, we accumulate the point clouds captured in the consecutive three frames (i.e., $1/10$-second time window) as a complete frame, which means the frame rate of our LiDAR is 10 FPS, as mentioned in Table 3.4.

3.3.3 Data Annotation

In the CRUW2022 dataset, we label 3D bounding boxes on LiDAR point clouds using an open-source annotation software [50]. Unlike the 3D bounding box labels in the KITTI dataset, we use three Euler angles to represent the orientation of each bounding box, since the streets in the CRUW2022 dataset are not as flat as those in the KITTI dataset. Here, we consider the following 5 object categories during the annotation: pedestrian, car, van, truck, and bus. The detailed statistics are shown in Section 3.3.4. In addition to the 3D bounding boxes, we also annotate object track IDs for later multi-object tracking (MOT) tasks. However, because different sensors have different FOVs, and point clouds for the faraway objects are usually sparse, we only annotated the object within the overlapped areas, shown in Figure 3.5. After the 3D bounding boxes are labeled on point clouds, we project all the bounding boxes to camera and radar coordinates by the transformation matrices from sensor calibration. Then, the annotations can be used to train networks for camera and radar, respectively.

Table 3.5: Statistics of the CRUW2022 dataset.

Description	Value		
Driving Time	40 min		
Scenarios	70% normal, 30% adverse city street, highway, sidewalk		
	`Overall`	`Train`	`Test`
# of Frames	66K	56K	10K
# of Seqs	74	56	18
# of Labeled 3D Bboxes	80K	57K	23K
# of Labeled 3D Tracks	576	397	179

3.3.4 Data Statistics

Our CRUW2022 dataset contains about 66K frames of synchronized camera, radar, and LiDAR data under various driving scenarios with different lighting conditions. Approximately

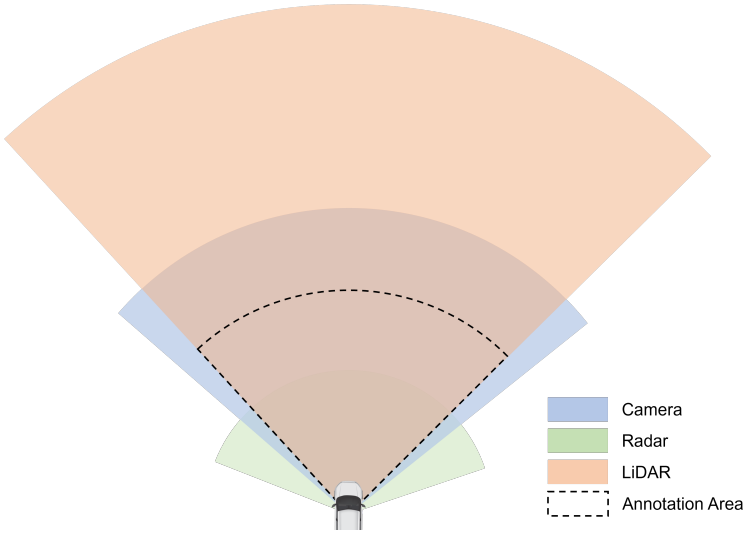


Figure 3.5: Illustration of sensors' FOVs and our annotation area.

70% of the data are captured in normal driving scenarios with good lighting conditions. The rest 30% are captured in adverse lighting conditions, e.g., nighttime, or strong lighting. Some data statistics are shown in Table 3.5. Among all the data frames, we annotate 19K frames as the training set, and 10K frames as the testing set.

As for the annotations for the CRUW2022 dataset, we analyze the different distributions of our labeled objects in Figure 3.6, including the number of 3D bounding boxes, number of 3D object trajectories, object depths, object azimuth angles, and object dimensions.

Object Class Distribution For the 5 object categories (i.e., pedestrian, car, van, truck, and bus) we are interested in, pedestrian and car are two dominant categories, as shown in Figure 3.6 (a) and (b), which reasonably reflects the actual object class distribution in real driving scenarios.

Object Location Distribution First, we analyze the depth distribution of the 3D bounding boxes in Figure 3.6 (c), where object depth represents the distance between LiDAR and the center of a 3D bounding box along LiDAR's x-axis. Here, most annotated 3D bounding

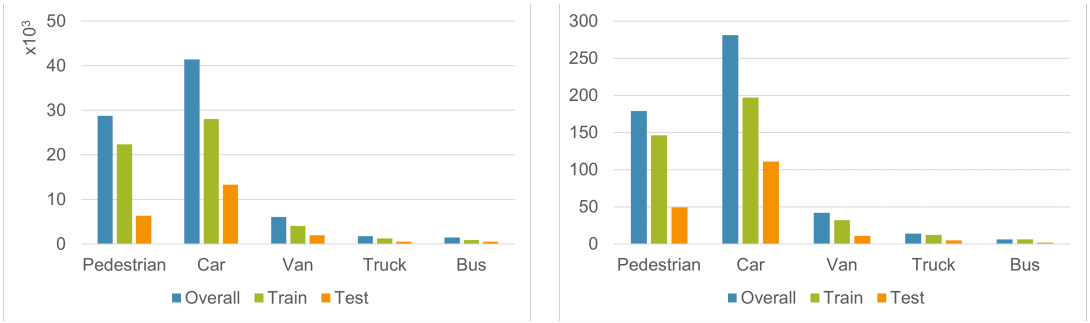
boxes are distributed within 0 – 40 meters. Besides, we also analyze object azimuth angle distribution in Figure 3.6 (d). Most labels fall into the range between $-50°$ to $50°$, which is the overlapped region for three sensor modalities.

Object Size Distribution Figure 3.6 (e) shows the distribution of object length for different object classes, including pedestrian, car, truck, and bus. The distributions for pedestrians and cars are relatively concentrated, while those of trucks and buses are more spread.

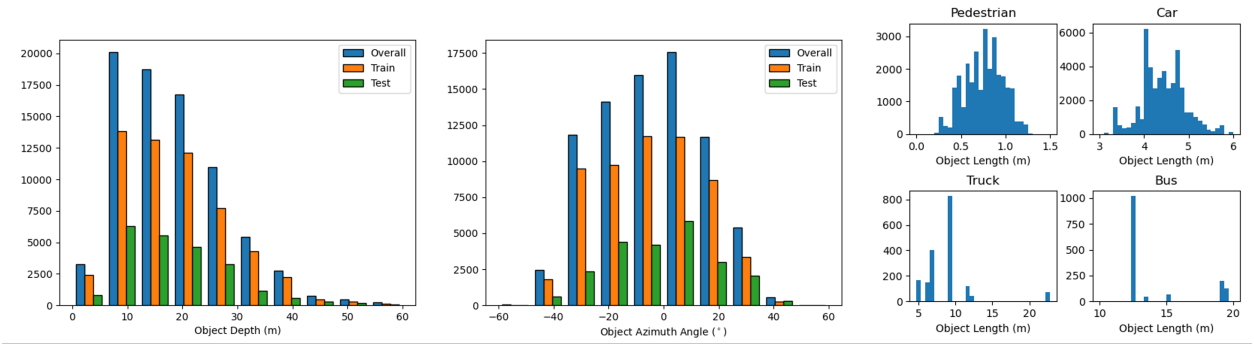
Object Trajectory We also record some statistics for that in Table3.5. Overall, there are 576 object trajectories, including 397 trajectories in the training set and 179 in the testing set. The average length of the object trajectories is 121 frames.

3.3.5 Comparison with Related Datasets

We compare the CRUW2022 dataset with some related datasets with radar sensors in Table 2.1. We discuss the dataset settings in different aspects, including sensor modalities, radar data format, driving scenarios, dataset scale, annotated object categories, annotation format, and public availability. From Table 2.1, most related datasets, whose radar data format is RF image, do not provide 3D bounding box and trajectory annotations.



(a) Number of 3D bounding boxes (b) Number of object trajectories



(c) Object depth distribution (d) Object azimuth angle distribution (e) Object length distribution

Figure 3.6: Object annotation distributions in the CRUW2022 dataset, including (a) object 3D bounding box distribution, (b) object trajectory distribution, (c) object depth distribution, (d) object azimuth angle distribution, and (e) object length distribution.

Chapter 4

OBJECT PERCEPTION VIA CROSS-MODALITY SUPERVISION

In this chapter, we introduce our proposed object perception system via cross-modality supervision. Since the camera can provide better semantic information while radar can provide more reliable range and speed information, we consider using the camera as a teacher to teach radar for object perception tasks, including object detection and multi-object tracking (MOT). Here, we first consider using a monocular camera object 3D localization algorithm as the supervision in Section 4.1. Then, we propose a camera-radar fused object 3D localization algorithm for more reliable supervision in Section 4.2. Based on these supervision algorithms, we further propose a radar-based object detection method (RODNet) in Section 4.3 and a radar-based multi-object tracking algorithm (RadarMOT) in Section 4.4.

4.1 Monocular Camera Object 3D Localization

4.1.1 Framework Overview

The object 3D information like distance and angle to the driver is crucial but challenging for cheap and light-weighted monocular camera settings. We present a framework for adaptive 3D object localization, combining information from monocular depthmap, ground plane estimation, and multi-object tracking, to achieve accurate and robust results.[1]

The proposed system's flowchart is shown in Figure 4.1, where the system takes a video sequence as the input. For each frame, we firstly use Mask R-CNN object detector [36] and monocular DepthNet CNN [33] to get object detections with masks and a dense depthmap respectively. Then, we use the object mask to crop the depthmap and derive object depth.

[1]The webpage for this work: `http://yizhouwang.net/blog/2019/07/15/object-3d-localization/`

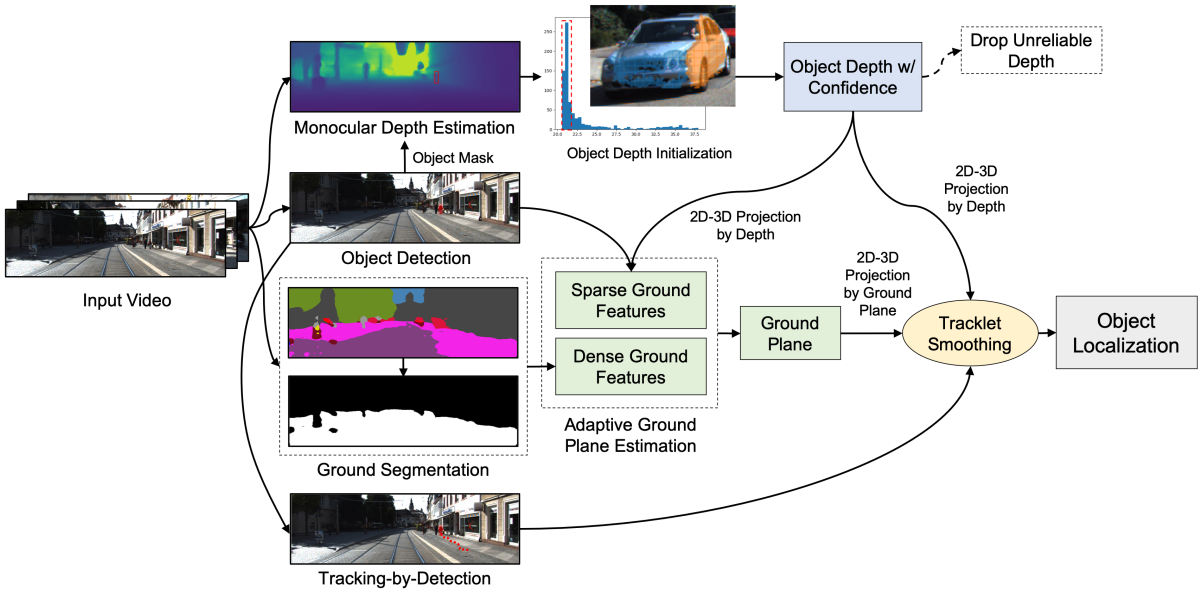


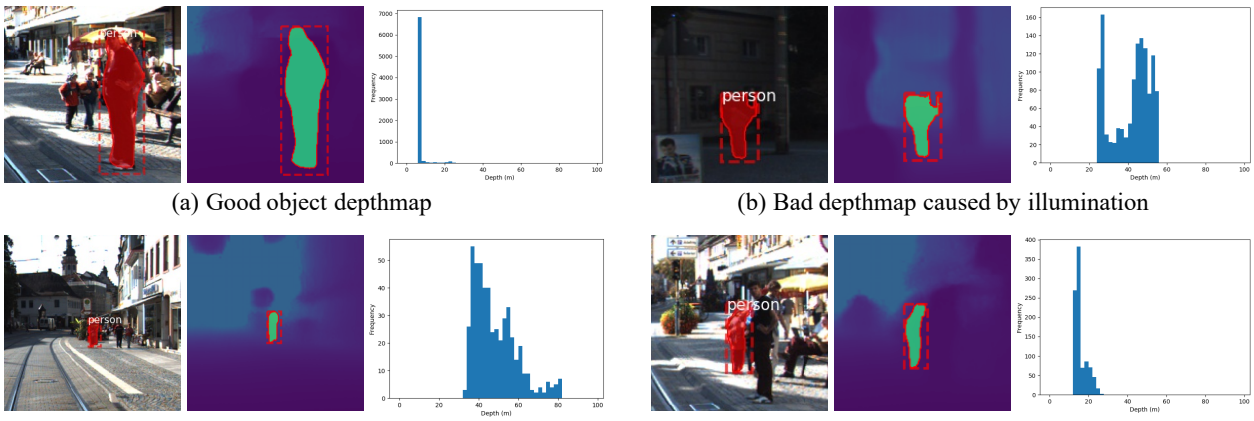
Figure 4.1: Overview of our proposed 3D localization framework for detected objects in the road scenes. There are mainly three parts in this framework: object depth initialization, adaptive ground plane estimation, and object tracklet smoothing.

On the other hand, semantic segmentation [17] is also applied to get the ground mask. With the depth of the ground points, we can project them to 3D coordinates. With object depths and ground point clouds, we can adaptively estimate the ground plane for each frame. Finally, a multi-object tracking technique is applied for object 3D tracklet smoothing.

4.1.2 Object Depth Initialization

First of all, we put each frame in the input video into a DepthNet CNN proposed in [33] to obtain an initial dense depthmap for each frame. To analyze the object depth from the CNN-generated depthmap, we take advantage of recent object detection techniques to obtain object categories and their bounding boxes. However, an object bounding box also contains some background areas. Instead, we use Mask R-CNN [36], which not only gives accurate bounding boxes for the objects but also provides the instance segmentation masks.

After the depthmap is cropped out by an object mask, we analyze the cropped depthmap



(a) Good object depthmap

(b) Bad depthmap caused by illumination

(c) Bad depthmap caused by resolution

(d) Bad depthmap caused by occlusion

Figure 4.2: Some examples for object image, masked depthmap, and depth histogram.

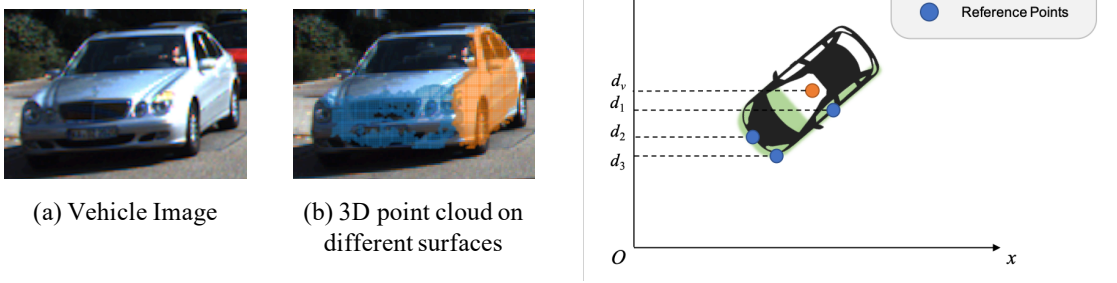
by depth histogram. Some examples of the cropped depthmaps and their corresponding depth histograms are shown in Figure 4.2. Here, we consider the histogram of range within $[0, 80]$ meters, with bin width being 2 meters. We define the proposal depth bins PB based on the neighboring w bins centered at the mode of the depth histogram $\mathcal{H}$, i.e., bin b with the largest histogram value f_b. More specifically, $PB = \{bin_l, bin_{l+1}, \cdots, bin_b, \cdots, bin_{u-1}, bin_u\} \subseteq \mathcal{H}$, where the lower bound is $l = \max(0, b - w)$ and the upper bound is $u = \min(80, b + w)$. From these proposal depth bins, we collect all the depth points and calculate the object depth d_{obj} as the average of the proposal depth points,

$$d_{obj} = \frac{1}{|PB|} \sum_{d_i \in PB} d_i. \tag{4.1}$$

The depthmap cropped by the object mask is found to be fairly uniform if the depth estimation is reliable, and the distribution of the depth histogram also reflects this kind of consistency. Thus, we define the object depth confidence using the statistics of the depth histogram,

$$c_{obj} = \left(1 - \frac{\sigma}{\mu}\right) \cdot \frac{|PB|}{|\mathcal{H}|}. \tag{4.2}$$

The first term contains the coefficient of variation (CV) of the whole histogram, where μ and



(a) Vehicle Image

(b) 3D point cloud on different surfaces

(c) Schematic diagram for 3D point cloud geometry

Figure 4.3: Illustration of vehicle depth initialization.

σ are the mean and standard deviation of the histogram respectively. When the histogram is dominant, CV will be small and the value of the first term will be big, and vice versa. The second term reflects the proportion of the proposal bins in all bins. This term is associated with the window size w. A good choice of w for a reliable depthmap can result in a high value of the second term. Because the values of the two terms are both between 0 and 1, the depth confidence c_{obj} is also between 0 and 1.

For some objects whose size cannot be neglected, like vehicles, we can also take advantage of the cropped object depthmap to estimate their depths. We firstly get the 3D point cloud by projecting the 2D points under the mask using the object depthmap, from which we can fit into different surfaces based on the simple linear iterative clustering (SLIC) strategy with superpixel segmentation [3, 39], and only two visible surfaces will be considered (front and left in the example of Figure 4.3). According to the shape constraints of the vehicles, we use a simple geometry to calculate the vehicle depth d_v shown in Figure 4.3(c), where

$$d_v = d_1 + (d_2 - d_3). \tag{4.3}$$

Here, d_1, d_2, and d_3 are three reference points whose depth can be easily obtained from the different surfaces we fitted above.

4.1.3 Adaptive Ground Plane Estimation

Typically, the ground plane can be represented as $g = [n^\top, h] = [n_1, n_2, n_3, h]$, where n is the normal vector of the plane and h is the plane offset corresponding to the origin, where h should be equal to the camera height. To fit the ground plane, we consider two kinds of ground features. One is the 3D point cloud laying on the ground, which can be named as dense ground features. The other is the object's 3D bottom points of pedestrians or the bottom center of the vehicles, which can be named as sparse ground features.

For sparse features, it is easy to obtain from the object depth in Section 4.1.2 by projection using depth information. The geometric formulation for this projection is

$$P = dK^{-1}\hat{p}, \tag{4.4}$$

where $\hat{p} \in \mathbb{R}^3$ denotes the homogeneous representation of a 2D point in pixel coordinates, P denotes the projected 3D point in the 3D camera coordinates, d is the depth of the point, and K is the camera intrinsic projection matrix. For dense features, we infer the semantic segmentation using a DeepLab [17] pre-trained model on CityScapes [19] and select the pixels with the ground label. After that, we can project the 2D points under the ground mask to 3D points using Equation 4.4.

After we have sparse and dense features, we use a novel augmented RANSAC algorithm to estimate the ground plane. The classical RANSAC algorithm iteratively fits the model from random subsets of the original data. To make the sparse features contribute more to the plane fitting, we do sparse feature augmentation by randomly adding sparse feature points around the object's bottom line of the detection bounding box with the same depth as the object depth. The number of augmented sparse feature points m_{i+} is decided by the object depth confidence c_i, the number of sparse feature points m_s, and the number of dense feature points m_d for the current frame,

$$m_{i+} = \left\lfloor \alpha \cdot \frac{c_i m_d}{m_s} \right\rfloor, \tag{4.5}$$

Algorithm 1: Adaptive Ground Plane Estimation

Input : Dense ground features DF, sparse ground features SF.
Output: Estimated ground plane parameters *planes*.

$planes \leftarrow []$, $\alpha \leftarrow 0.5$;
for each frame j **do**
  $dense \leftarrow DF[j]$;
  $sparse \leftarrow SF[j]$;
  if *sparse* **is not** `None` **then**
    $m_d \leftarrow dense.length$;
    $m_s \leftarrow sparse.length$;
    $m_+ \leftarrow []$;
    for each point i **in** *sparse* **do**
      $c_i \leftarrow sparse[i].conf$;
      $m_+ \leftarrow \left\lfloor \alpha \cdot \frac{c_i m_d}{m_s} \right\rfloor$; `// append to list` m_+
    $sparse_+ \leftarrow$ `SparseAugment(`$sparse, m_+$`)`;
    $points \leftarrow$ `Concatenate(`$dense, sparse_+$`)`;
  else `// use` *dense* `only when no` *sparse*
    $points \leftarrow dense$;
  $planes \leftarrow$ `RANSAC(`$points$`)` ; `// append to list` *planes*
return *planes*;

where α is a hyperparameter that represents the overall weight for sparse features. Finally, we concatenate the dense features and augmented sparse features together and apply RANSAC to fit a final ground plane.

After a refined ground plane is estimated, we can back-project the object from 2D to 3D using this ground plane. The geometric formulation for this projection is

$$P = -\frac{hK^{-1}\hat{p}}{n^\top K^{-1}\hat{p}}. \tag{4.6}$$

4.1.4 Object Tracklet Smoothing

Although the object depth is carefully estimated in Section 4.1.2, we might still get errors because of depthmap bias or some other frame-level errors. This kind of error cannot be solved by depth histogram or ground plane estimation because the frame-level error makes everything unreliable. In this case, one practical solution is to consider the information from

the neighboring frames.

Therefore, we involve an object tracklet smoothing method in our 3D object localization system. Firstly, we use TNT [91], which can associate consecutive bounding boxes of the same object among multiple objects to form tracklets from the input video sequences, to robustly handle occlusion and camera motion. Then, the trajectories are split into several shorter tracklets with a moving split window. After that, for each 3D location of the object in the same tracklet, the weighted Huber regression for each of the three axes of the camera coordinates is used. Huber regression is to use Huber loss (Equation 4.7) instead of squaring the residual in the ordinary linear regression for the 3D object from the same tracklet. The special property for Huber loss is that the loss will increase linearly when the absolute value of residual is greater than the pre-specified threshold η.

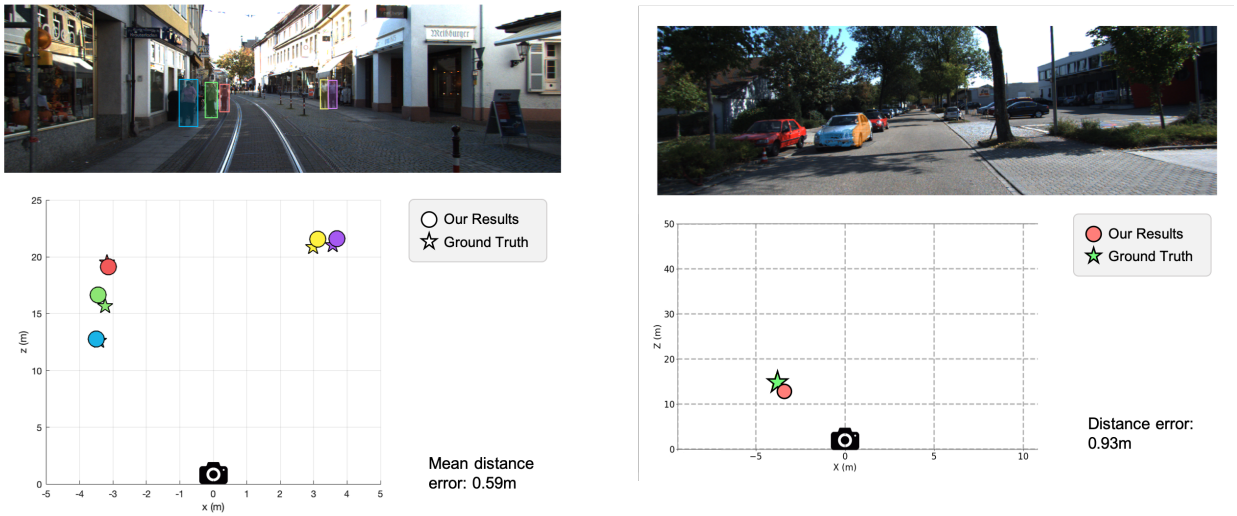
$$\rho(r_i) = \begin{cases} c_i \cdot r_i^2 & \text{if } |r_i| \leq \eta, \\ c_i \cdot \eta(2|r_i| - \eta) & \text{if } |r_i| > \eta. \end{cases} \tag{4.7}$$

Here, r_i represents the residual of object i, and c_i is the corresponding depth confidence.

4.1.5 Experiments

We use KITTI dataset [30] to evaluate the performance of our proposed object 3D localization method. Object 3D tracking annotations are provided by KITTI with the 3rd human validation stage for most of the sequences in date `2011-09-26`. Here, we choose sequences numbered 0005, 0009, 0014, 0051, 0056, 0059, 0084, 0091 for pedestrian 3D localization evaluation, and choose sequence 0009 for vehicle 3D localization.

We compare the object 3D localization results by computing the mean absolute translation error (with its standard deviation) of the pedestrians (in meters) against the available ground truth object 3D locations from KITTI. The qualitative results are shown in Figure 4.4(a), where the upper image is the object detection results and the lower figure shows the 3D localization results compared with the ground truth in a bird-view. The quantitative



(a) Pedestrian Localization (b) Vehicle Localization

Figure 4.4: Qualitative results for our object 3D localization method on KITTI.

results are shown in Table 4.1, where we use some short expressions for the different methods: `DHist`: Depth histogram; `TS`: Tracklet smoothing; `AGPE`: Adaptive ground plane estimation. According to Table 4.1, we can see that all of our pedestrian 3D localization results achieve less than 0.8 meters localization error, which outperforms the state-of-the-art methods that are mainly reported to vehicle localization.

4.2 Object 3D Localization by Camera-Radar Fusion

An intuitive way of improving the above camera-only annotation is by taking advantage of radar, which has a plausible capability of range estimation without any systematic bias. Here, we adopt the Constant False Alarm Rate (CFAR) detection algorithm [75], which is commonly used in the signal processing community to detect peaks in the RF image. As shown in Fig. 4.5 (b), the CFAR algorithm can detect a number of peaks in RF images, denoted as red dots. However, these detected peaks cannot be directly used as supervision because 1) the CFAR algorithm cannot provide the object classes for each detection; 2) the

Table 4.1: Mean localization error (standard deviation) for pedestrians compared with some vehicle localization methods on KITTI.

Methods	Overall (m)	$\leq$ 15m	$\leq$ 30m	$>$ 30m	Running speed
Murthy et al. [62]	2.61 ($\pm$2.23)	1.59 ($\pm$0.96)	2.52 ($\pm$2.16)	4.30 ($\pm$2.83)	–
Ansari et al. [6]	1.00 ($\pm$0.77)	0.67 ($\pm$0.50)	0.94 ($\pm$0.69)	2.19 ($\pm$1.18)	–
Ansari et al. `(Opt)` [6]	0.86 ($\pm$0.87)	0.55 ($\pm$0.50)	0.79 ($\pm$0.79)	2.16 ($\pm$1.18)	–
Ours `(DHist)`	0.79 ($\pm$0.75)	0.43 ($\pm$0.31)	0.76 ($\pm$0.73)	2.78 ($\pm$2.01)	6.1 FPS
Ours `(DHist+AGPE)`	0.74 ($\pm$0.64)	0.43 ($\pm$0.31)	0.71 ($\pm$0.63)	2.39 ($\pm$1.61)	3.3 FPS
Ours `(DHist+TS)`	0.73 ($\pm$0.62)	**0.40 ($\pm$0.30)**	0.71 ($\pm$0.61)	2.15 ($\pm$1.32)	2.6 FPS
Ours `(DHist+AGPE+TS)`	**0.69 ($\pm$0.51)**	0.42 ($\pm$0.33)	**0.68 ($\pm$0.53)**	**1.22 ($\pm$0.74)**	2.0 FPS

CFAR algorithm usually gives a large number of false positive detections. Thus, an object localization method by camera-radar fusion strategy is needed to address these issues.

4.2.1 Heuristic Fusion Algorithm

The basic idea of the heuristic fusion algorithm for CRF supervision is to fuse the location results from the camera and radar by the distances between each pair. During the fusion process, we trust the range from radar, and trust the azimuth from the camera. The algorithm is written step-by-step in Alg. 2. The brief pipeline of this algorithm can be concluded as follows:

1) Calculate the fused locations for each pair of the camera-radar locations.

2) Remove nearby redundant radar locations according to their distances.

3) First, find and keep the best matching pair for each radar detection. Then, find and keep the best matching pair for each camera detection.

4) Now, the above matching becomes a one-to-one mapping. Collect all the mappings as the final CRF annotations.

Algorithm 2: Heuristic Camera-Radar Fusion

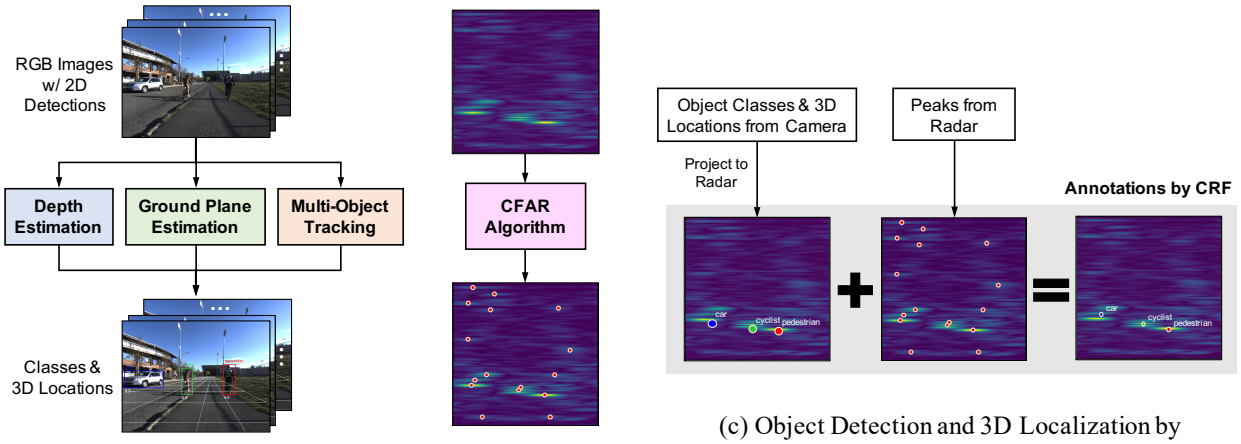
Input : Locations from camera $\Xi^c = \{(\rho_i^c, \theta_i^c)\}$, locations from radar $\Xi^r = \{(\rho_j^r, \theta_j^r)\}$.
Output: Final locations after CR fusion Ξ^{CRF}.
Initialize fused locations for all CR pairs $\Xi^{cr} = \{\xi_{i,j}^{cr}\}$;
Initialize matching flags for CR location pairs $\{\mu_{i,j}\}$;
for *every pair* (ξ_i^c, ξ_j^r) **from** (Ξ^c, Ξ^r) **do**
    Fused location for pair k, $\xi_k^\star \leftarrow (\rho_j^r, \theta_i^c)$;
    $\xi_{i,j}^{cr} \leftarrow \xi_k^\star$;
    Calculate range residual $e_\rho \leftarrow |\rho_i^c - \rho_k^\star|$;
    Calculate azimuth residual $e_\theta \leftarrow |\theta_j^r - \theta_k^\star|$;
    if $e_\rho < \phi_\rho$ **and** $e_\theta < \phi_\theta$ **then**
        $\mu_{i,j} \leftarrow$ `True`;
    else
        $\mu_{i,j} \leftarrow$ `False`;
    end
end
for *every* ξ_i^c **from** Ξ^c **do**
    $\Xi_i^r \leftarrow$ collect all ξ_j^r satisfying $\mu_{i,j}$ is `True`;
    Group Ξ^r according to euclidean distance to remove redundant radar detections;
    Update Ξ^r and Ξ^{cr};
end
for *every* ξ_j^r **from** Ξ^r **do**
    $\Xi_j^c \leftarrow$ collect all ξ_i^c satisfying $\mu_{i,j}$ is `True`;
    Find the nearest $\xi^{c\star} \in \Xi_j^c$ to ξ_j^r;
    for *every* ξ_j^c **from** Ξ_j^c **do**
        if ξ_j^c *is not* $\xi^{c\star}$ **then**
            $\mu_{i,j} \leftarrow$ `False`;
        end
    end
end
for *every* ξ_i^c **from** Ξ^c **do**
    $\Xi_i^r \leftarrow$ collect all ξ_j^r satisfying $\mu_{i,j}$ is `True`;
    Find the nearest $\xi^{r\star} \in \Xi_i^r$ to ξ_i^c;
    for *every* ξ_i^r **from** Ξ_i^r **do**
        if ξ_i^r *is not* $\xi^{r\star}$ **then**
            $\mu_{i,j} \leftarrow$ `False`;
        end
    end
end
$\Xi^{CRF} \leftarrow$ collect all $\xi_{i,j}^{cr} \in \Xi^{cr}$ that $\mu_{i,j}$ is `True`;

4.2.2 Probabilistic Fusion Algorithm

Fig. 4.5 (c) illustrates the camera-radar fusion (CRF) pipeline, where the classes and 3D locations of the detected objects from the camera are first passed through a transformation to project the detections from 3D camera coordinates to radar range-azimuth coordinates. The transformation can be formulated as

$$\begin{aligned} \rho_c &= \sqrt{(x^c - x_{or})^2 + (z^c - z_{or})^2}, \\ \theta_c &= \tan^{-1}\left(\frac{x^c - x_{or}}{z^c - z_{or}}\right), \end{aligned} \tag{4.8}$$

where (ρ_c, θ_c) denotes the projected location in radar range-azimuth coordinates; (x^c, z^c) denotes the object location in the camera BEV coordinates; and (x_{or}, z_{or}) denotes the location of radar origin in the camera BEV coordinates, aligned from the sensor system calibration. The peak detections from the CFAR algorithm are also involved in the same radar range-azimuth coordinates. Finally, the fusion algorithm is applied to estimate the final annotations on the input RF image.



(a) Object Detection and 3D Localization (Camera)

(b) Peak Detection (Radar)

(c) Object Detection and 3D Localization by Camera-Radar Fusion (CRF)

Figure 4.5: Three teacher's pipelines for cross-model supervision. (a) Camera-only method that provides object classes and 3D locations; (b) Radar-only method that only provides peak locations without object class; (c) Camera-radar fusion method that provides object classes and more accurate 3D locations.

After the coordinates between the camera and the radar are aligned, a probabilistic CRF algorithm is further developed to achieve a more reliable and systematic annotation performance. The basic idea of this algorithm is to generate two probability maps for camera and radar locations separately, and then fuse them by element-wise product. The probability map for camera locations with object class cls is generated by

$$\mathcal{P}^c_{(cls)}(\mathbf{x}) = \max_i \left\{ \mathcal{N}\left(\frac{1}{2\pi\sqrt{|\mathbf{\Sigma}^c_{i(cls)}|}} \exp\left\{ -\frac{1}{2}(\mathbf{x}-\mu^c_i)^\top (\mathbf{\Sigma}^c_{i(cls)})^{-1}(\mathbf{x}-\mu^c_i) \right\} \right) \right\},$$

$$\mu^c_i = \begin{bmatrix} \rho^c_i \\ \theta^c_i \end{bmatrix}, \mathbf{\Sigma}^c_{i(cls)} = \begin{bmatrix} \left(d_i s_{(cls)}/c_i\right)^2 & 0 \\ 0 & \delta_{(cls)} \end{bmatrix}. \tag{4.9}$$

Here, d_i is the object depth, $s_{(cls)}$ is the scale constant, c_i is the depth confidence, and $\delta_{(cls)}$ is the typical azimuth error for camera localization. $\mathcal{N}(\cdot)$ represents the normalization operation for each object's probability map. Similarly, the probability map for radar locations is generated by

$$\mathcal{P}^r(\mathbf{x}) = \max_j \left\{ \mathcal{N}\left(\frac{1}{2\pi\sqrt{|\mathbf{\Sigma}^r_j|}} \exp\left\{ -\frac{1}{2}(\mathbf{x}-\mu^r_j)^\top (\mathbf{\Sigma}^r_j)^{-1}(\mathbf{x}-\mu^r_j) \right\} \right) \right\},$$

$$\mu^r_j = \begin{bmatrix} \rho^r_j \\ \theta^r_j \end{bmatrix}, \mathbf{\Sigma}^r_j = \begin{bmatrix} \delta^r_j & 0 \\ 0 & \epsilon(\theta^r_j) \end{bmatrix}. \tag{4.10}$$

Here, δ^r_j is the radar's range resolution, and $\epsilon(\cdot)$ is the radar's azimuth resolution. Then, an element-wise product is used to obtain the fused probability map for each class,

$$P^{CRF}_{(cls)}(\mathbf{x}) = P^c_{(cls)}(\mathbf{x}) * P^r(\mathbf{x}). \tag{4.11}$$

Finally, the annotations are derived from the fused probability maps P^{CRF} by peak detection.

4.3 Radar Object Detection Network (RODNet)

4.3.1 RODNet Architecture

There are three major functional components adopted in constructing the network architecture of the RODNet, as shown in Fig. 4.6 (a)-(c), which is implemented based on a 3D CNN with an autoencoder structure. More specifically, our RODNet starts with a naïve version of a 3D CNN autoencoder network, shown in Fig. 4.6 (a). Then, built upon an hourglass based autoencoder [64], shown in Fig. 4.6 (b), where skip connections are added to transmit the features directly from bottom layers to top layers. We further add the temporal inception convolution layers to extract different lengths of temporal features from the input RF image sequence, inspired by the spatial inception convolution layer proposed in [85], shown in Fig. 4.6 (c).

The input of our network is a snippet of RF images $\boldsymbol{R}$ with dimension (C_{RF}, T, n, H, W), where C_{RF} is the number of channels in each complex-numbered RF images, referring [104], where the real and imaginary values are treated as two different channels in one RF image, i.e., $C_{RF} = 2$; T is the number of RF image frames in the snippet; n is the number of chirps in each frame; H and W are the height and width of the RF images, respectively.

After passing through the network, ConfMaps $\hat{\boldsymbol{D}}$ with dimension (C_{cls}, T, H, W) are predicted, where C_{cls} is the number of object classes. Note that RODNet predicts separate ConfMaps for each individual object class of radar RF images. With systematically derived binary annotations using the teacher's pipeline described in Section 4.2, we can train our RODNet using binary cross entropy loss,

$$\ell = -\sum_{cls}\sum_{i,j} \boldsymbol{D}_{i,j}^{cls} \log \hat{\boldsymbol{D}}_{i,j}^{cls} + \left(1 - \boldsymbol{D}_{i,j}^{cls}\right) \log\left(1 - \hat{\boldsymbol{D}}_{i,j}^{cls}\right). \tag{4.12}$$

Here, $\boldsymbol{D}$ represents the ConfMaps generated from CRF annotations, $\hat{\boldsymbol{D}}$ represents the predicted ConfMaps, (i, j) represents the pixel indices, and cls is the class label.

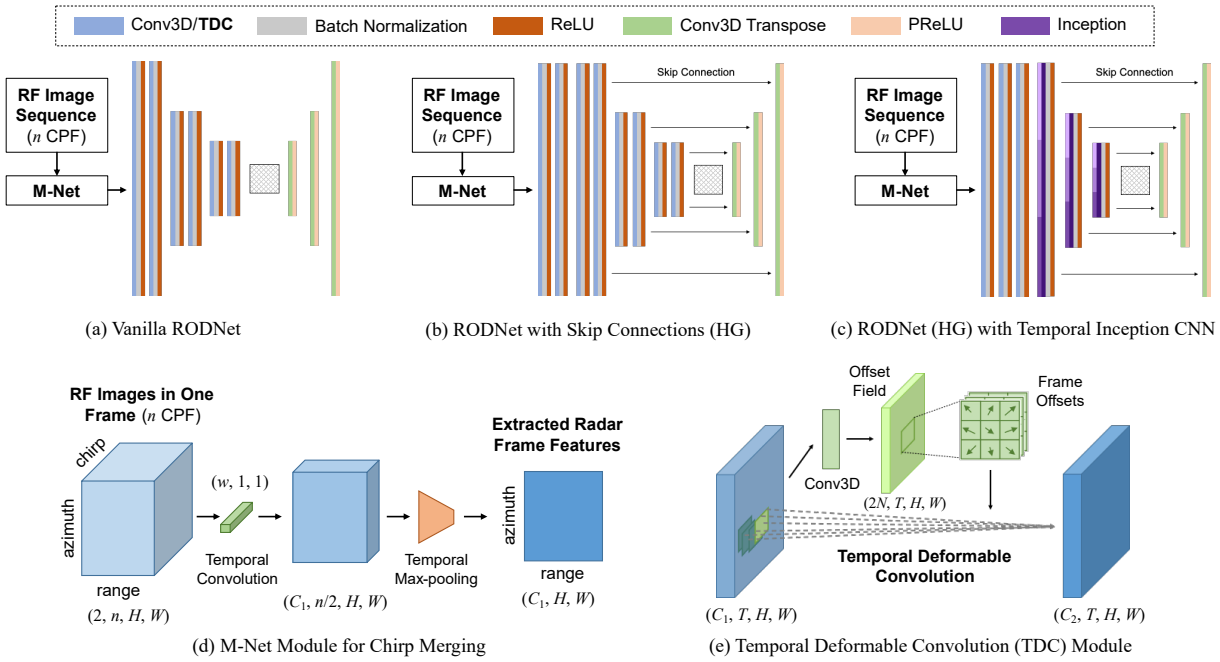


Figure 4.6: The architecture and modules of our proposed RODNet. Three different components of the RODNet are all implemented based on the 3D CNN/TDC and autoencoder network, as shown in (a), (b), and (c). The input of the RODNet is RF images with n chirps per frame (CPF). When $n = 1$, we select only one chirp's data randomly to feed into the RODNet, while for $n > 1$, the M-Net module is implemented to merge the data from different chirps in this frame. The M-Net module, described in (d), which takes as input one frame with multiple chirps of radar data and outputs this frame's merged features. Moreover, the temporal deformable convolution (TDC) module in (e) is introduced to handle the radar object dynamic motion within the input RF image sequence.

4.3.2 M-Net Module

Besides the temporal features across different frames in each RF snippet, all the information from different chirps contributes to features for radar object detection. In order to better integrate this dynamic information from different chirps, we propose a customized module, called M-Net, before the RF snippets are sent into the RODNet. As shown in Fig. 4.6 (d), the RF images of one frame with n chirps are sent into M-Net with a dimension of (C_{RF}, n, H, W), where $C_{RF} = 2$. First, a temporal convolution is applied to extract temporal features among the n chirps. This M-Net CNN operation performs like a Doppler-compensated FFT

to extract dynamic motion features but can be trained end-to-end in the deep learning architecture. Then, to merge the features from n chirps into one, a temporal max-pooling layer is applied. Finally, the output of M-Net is the extracted radar frame features with a dimension of (C_1, H, W), where C_1 is the number of filters for the temporal convolution.

4.3.3 Temporal Deformable Convolution

As mentioned in Section 4.3.1, the input of the RODNet is a snippet of RF images after features are merged by the M-Net. Thus, during this period, locations of the objects in the radar range-azimuth coordinates may be shifted due to object relative motion, which means the patterns or peaks in RF images may change their locations within the snippet. However, the classical 3D convolution can only capture the static features within a regular cuboid, therefore it is not the best feature extractor.

Recently, Dai *et al.* [20] propose a new convolution network, named deformable convolution network (DCN), for image-based object detection to handle the deformed objects within the images. Inspired by DCN, we generalize the deformed kernel to the 3D CNN and name this novel operator as temporal deformable convolution (TDC). Use the 3D CNN with kernel size of $(3, 3, 3)$ and dilation 1 as an example, the regular receptive field $\mathcal{R}$ can be defined as

$$\mathcal{R} = \{(-1, -1, -1), (-1, 0, 0), \cdots, (0, 1, 1), (1, 1, 1)\}. \tag{4.13}$$

For each location $\mathbf{p}_0$ on the output feature map $\mathbf{y}$, the classical 3D convolution can be described as

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n), \tag{4.14}$$

where $\mathbf{w}$ is the convolution kernel weight, $\mathbf{x}$ is the input feature map, and $\mathbf{p}_n$ enumerates the locations in $\mathcal{R}$.

In order to handle the object dynamic motion in the temporal domain, we propose TDC by adding an additional offset field $\{\Delta\mathbf{p}_n\}_{n=1}^{N}$, where $N = |\mathcal{R}|$ is the size of the receptive

field. So that Eq. 4.14 becomes

$$\mathbf{y}(\mathbf{p}_0) = \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n). \tag{4.15}$$

Note that the offset field $\Delta\mathbf{p}_n$ is only deformed within each temporal location, i.e., the receptive location of a certain frame will not be deformed to other frames, so that the temporal domain of the offset field is always zero. To make it easy for implementation, the offset vectors are defined as 2D vectors so that the overall offset field has a dimension of $(2N, T, H, W)$. An illustration of our proposed TDC is shown in Fig. 4.6 (e).

Similar to [20], since the offset field $\Delta\mathbf{p}_n$ is typically fractional, Eq. 4.15 is implemented via bilinear interpolation as

$$\mathbf{x}(\mathbf{p}) = \sum_{\mathbf{q}} G(\mathbf{q}, \mathbf{p}) \cdot \mathbf{x}(\mathbf{q}), \tag{4.16}$$

where $\mathbf{p} = \mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n$ is the fractional location; $\mathbf{q}$ enumerates all integer locations in the 3D feature map $\mathbf{x}$; and G is the bilinear interpolation kernel which is also two dimensional in the spatial domain. The back-propagation formulation of TDC is similar to that discussed in [20] except for adding the temporal domain.

4.3.4 Post-processing by Location-based NMS

After predicting ConfMaps from a given RF snippet, a post-processing step is still required to obtain the final detections. Here, we adopt the idea of non-maximum suppression (NMS), which is frequently used in image-based object detection to remove the redundant bounding boxes from the detection results. Traditionally, NMS uses intersection over union (IoU) as the criterion to determine if a bounding box should be removed due to its too much overlapping with the detection candidate of the highest confidence. However, there is no bounding box definition in our RF images nor the resulting output ConfMaps. Thus, inspired by object keypoint similarity (OKS) defined for human pose evaluation in the COCO dataset [53], we define a new metric, called object location similarity (OLS) that is similar to the role of IoU,

to describe the correlation between two detections considering their distance, classes and scale information on ConfMaps. More specifically,

$$\text{OLS} = \exp\left\{\frac{-d^2}{2(s\kappa_{cls})^2}\right\}, \tag{4.17}$$

where d is the distance (in meters) between the two points in an RF image; s is the object distance from the radar sensor, representing object scale information; and κ_{cls} is a per-class constant that represents the error tolerance for class cls, which can be determined by the object average size of the corresponding class. We empirically determine κ_{cls} to make OLS distributed reasonably between 0 and 1. Here, we try to interpret OLS as a Gaussian distribution, where distance d acts as the bias and $(s\kappa_{cls})^2$ acts as the variance. Therefore, OLS is a metric of similarity, which also considers object sizes and distances, so that is more reasonable than other traditional distance metrics, such as Euclidean distance, Mahalanobis distance, etc. This OLS metric is also used to *match detections and ground truth* for evaluation purposes, mentioned in Section 4.3.5.

After OLS is defined, we propose a location-based NMS (L-NMS) for post-processing. An example is shown in Fig. 4.7, and the procedure can be summarized as follows:

1) Get all the 8-neighbor peaks in all C_{cls} channels in ConfMaps within the 3×3 window as a peak set $P = \{p_n\}_{n=1}^{N}$.

2) Pick the peak $p^* \in P$ with the highest confidence, put it to the final peak set P^*, and remove it from set P. Calculate OLS with each of the rest peaks p_i ($p_i \neq p^*$).

3) If OLS between p^* and p_i is greater than a threshold, remove p_i from the peak set.

4) Repeat Steps 2 and 3 until the peak set becomes empty.

Therefore, the proposed L-NMS can be used as a point-based NMS algorithm to remove redundant detections.

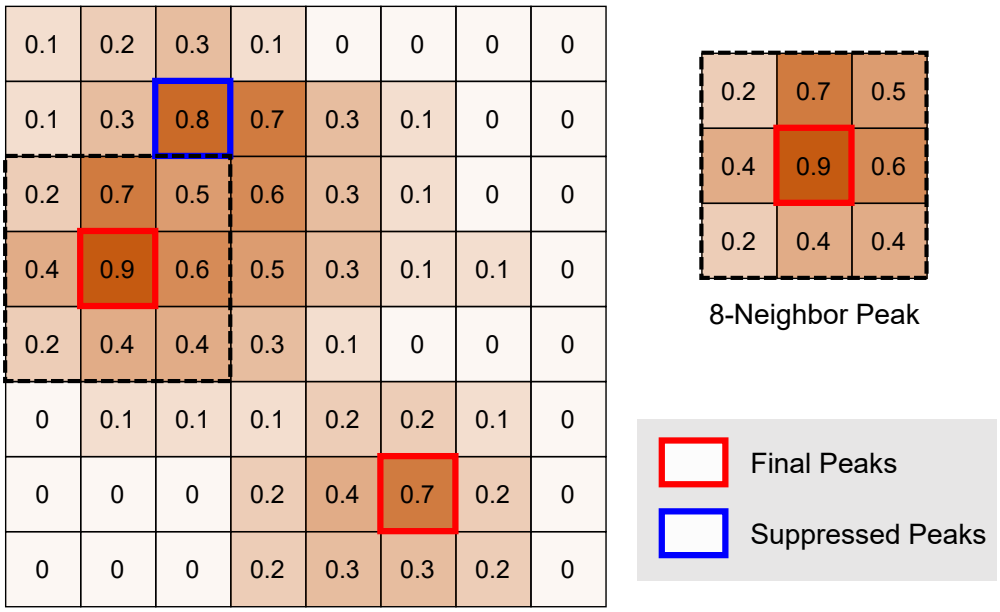


Figure 4.7: Example for L-NMS on a ConfMap. The numbers represent the confidence scores predicted by the RODNet. The 8-neighbor peaks are first detected and some peaks are then suppressed if they are nearby some other peaks with higher confidence.

4.3.5 Experiments

4.3.5.1 Evaluation Metrics

To evaluate the performance, we utilize our proposed object location similarity (OLS) (see Eq. 4.17) in Section 4.3.4, replacing the role of IoU widely used in image-based object detection, to determine how well a detection result can be matched with a ground truth. During the evaluation, we first calculate OLS between each detection result and ground truth in every frame. Then, we use different thresholds from 0.5 to 0.9 with a step of 0.05, for OLS and calculate the average precision (AP) and average recall (AR) for different OLS thresholds, which represent different localization error tolerance for the detection results. Here, we use AP and AR to represent the average values among all different OLS thresholds from 0.5 to 0.9, and use AP^{OLS} and AR^{OLS} to represent the values at a certain OLS threshold. Overall, we use AP and AR as our main evaluation metrics for the radar object detection task.

4.3.5.2 Radar Object Detection Results

We train our RODNet using the training data with CRF annotations in the CRUW dataset. For testing, we perform inference and evaluation on the human-annotated data. The quantitative results are shown in Table 4.2. We compare our RODNet results with the following baselines that also use radar-only inputs: 1) A decision tree using some handcrafted features from radar data [28]; 2) CFAR detection is first implemented and a radar object classification network with ResNet backbone [5] is appended; 3) Similar with 2), a radar object classification network with VGG-16 backbone based on CFAR detections mentioned in [28].

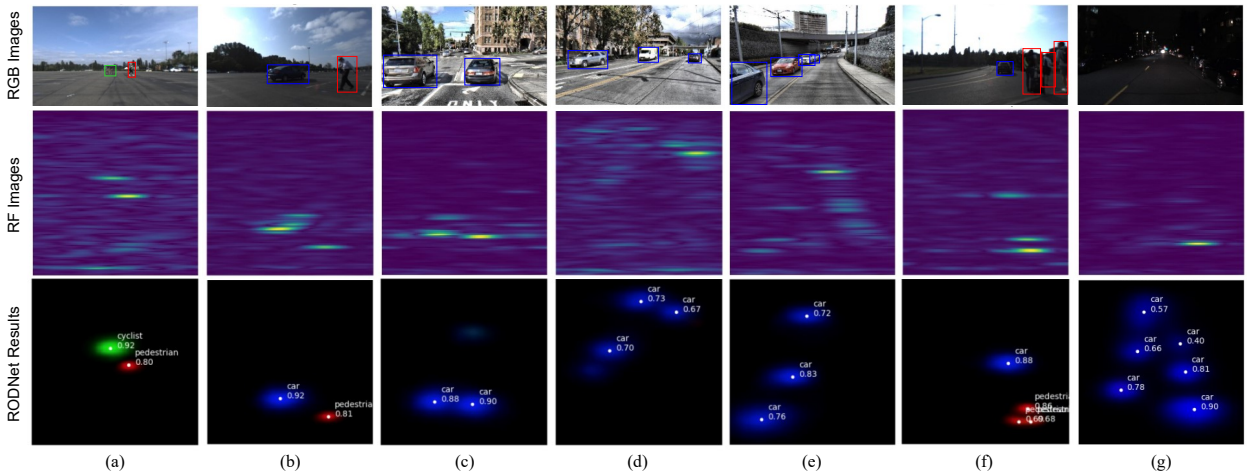


Figure 4.8: Examples of detection results from our RODNet. The first row shows the RGB images and the second row shows the corresponding RF images. The ConfMaps predicted by the RODNet are shown in the third row, where the white dots represent the final detections after post-processing. Different colors represent different detected object classes (red: pedestrian; green: cyclist; blue: car). Various driving scenarios are shown, i.e., clear parking lot, crowded city street, and strong/weak lighting conditions. More qualitative results are presented in the supplementary materials.

Compared with the above baseline methods, our RODNet outperforms significantly on both AP and AR metrics, achieving the best performance of 85.98% AP and 87.86% AR, especially the sustained performance on the medium and hard testing sets, which shows the robustness to noisy scenarios. Note that the results of RODNet shown in Table 4.2 include

Table 4.2: Radar object detection performance evaluated on CRUW dataset.

Methods	Overall		Easy		Medium		Hard	
	AP	AR	AP	AR	AP	AR	AP	AR
Decision Tree [28]	4.70	44.26	6.21	47.81	4.63	43.92	3.21	37.02
CFAR+ResNet [5]	40.49	60.56	78.92	85.26	11.00	33.02	6.84	36.65
CFAR+VGG-16 [28]	40.73	72.88	85.24	88.97	47.21	62.09	10.97	45.03
RODNet (Ours)	**85.98**	**87.86**	**96.97**	**98.02**	**76.11**	**78.57**	**67.28**	**72.60**

Table 4.3: Ablation studies on the performance improvement by several functional components in the RODNet.

Backbone	Supv.	M-Net	TDC	Incep.	AP	$AP^{0.5}$	$AP^{0.7}$	$AP^{0.9}$	AR	$AR^{0.5}$	$AR^{0.7}$	$AR^{0.9}$
Vanilla	CO				52.62	78.21	54.66	18.92	63.95	84.13	68.76	30.71
	CRF				74.29	78.42	76.06	64.58	77.85	80.05	78.93	71.72
	CRF	✓			78.36	82.73	81.03	65.82	81.54	84.51	83.39	73.53
	CRF	✓	✓		79.86	84.08	82.37	66.74	82.85	86.06	84.43	73.93
Hourglass	CO				73.86	80.34	74.94	61.16	79.87	83.94	80.73	71.39
	CO			✓	77.75	82.88	79.93	61.88	81.11	85.13	82.78	68.63
	CRF				81.10	84.71	83.08	70.21	84.26	86.54	85.42	77.44
	CRF	✓			83.37	87.51	86.04	71.11	85.64	88.55	87.19	77.37
	CRF			✓	83.76	87.99	86.00	70.88	85.62	88.79	87.37	76.26
	CRF	✓	✓		84.38	88.69	85.73	73.31	86.97	89.67	88.14	79.59
	CRF	✓	✓	✓	**85.98**	**88.77**	**87.78**	**76.34**	**87.86**	**89.93**	**89.02**	**81.26**

all the components proposed for RODNet, i.e., CRF supervision, M-Net, TDC, and temporal inception CNN.

Some qualitative results are shown in Fig. 4.8, where we can find that the RODNet can accurately localize and classify multiple objects in different scenarios. The examples in Fig. 4.8 consist of RGB and RF image pairs as well as RODNet detection results under different driving scenarios and conditions, including parking lot, campus road, and city street, with different lighting conditions. Some other examples to show the special strengths of our RODNet are shown in Section 4.3.5.

To illustrate our teacher's pipeline is qualified for this cross-supervision task, we evaluate the object 3D localization performance for both CO and CRF annotations in Table 4.4. Besides, we also compare the performance between CO/CRF supervision and our RODNet on both visible (V) and vision-hard (VH) data. The results are shown in Fig. 4.9 with

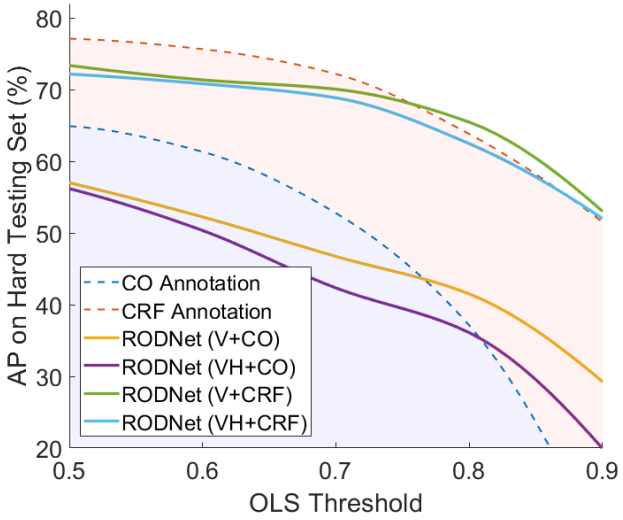


Figure 4.9: Performance of vision-based and our RODNet on hard testing set with different OLS thresholds, representing localization error tolerance. (CO: camera-only; CRF: camera-radar fusion; V: visible data; VH: vision-hard data.)

Table 4.4: The mean localization error (standard deviation) of CO/CRF annotations on CRUW dataset (in meters).

Supervision	Pedestrian	Cyclist	Car
CO	0.69 (±0.77)	0.87 (±0.89)	1.57 (±1.12)
CRF	0.67 (±0.55)	0.82 (±0.59)	1.26 (±0.64)

respect to different OLS thresholds. From Fig. 4.9, the performance of vision-based method drops significantly given a tighter OLS threshold, while our RODNet shows its superiority and robustness on its localization performance. Moreover, the RODNet can still maintain the performance on vision-fail data where the vision-based methods have a hard time to maintain the performance.

4.3.5.3 Performance-Speed Trade-off of the RODNet

Since our RODNet starts with an M-Net chirp-merged input to the vanilla autoencoder with 3D convolution layers, further added with skip connections in hourglass (HG) structure

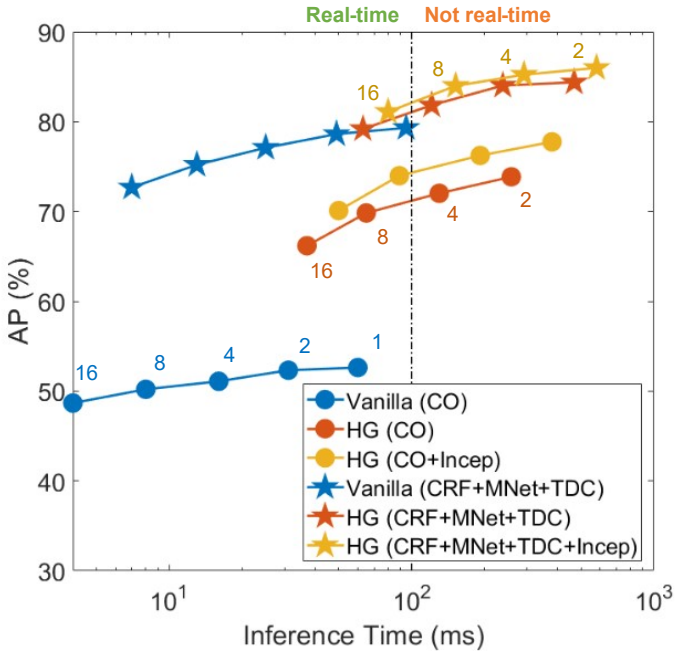


Figure 4.10: Performance-speed trade-off for real-time implementation. Here, the inference time of less than 100 ms is used as the real-time criterion. The snippet length is 16, and the numbers besides the markers are the steps between each two overlapped RF snippets.

and inception convolution layers, and eventually with TDC incorporation, it is important to know the performance influence of each functional module, as well as the computational complexities involved.

First of all, APs under different OLS thresholds are evaluated in Table 4.3. Here, we use different combinations of the backbones, supervision, and other modules in the RODNet. From the results in the table, the following conclusions can be reached: 1) The performance of the HG backbone is better than the vanilla backbone by around 5%. 2) Training the RODNet using CRF supervision can improve the performance by about 8%. 3) The customized modules, i.e., M-Net and TDC, along with temporal inception, can also each improve the detection performance by approximately $1\% - 2\%$, respectively.

Moreover, *real-time* implementation is very important for autonomous or assisted driving applications. As mentioned in Section 4.3.4, we use different overlapping lengths of RF frames

during the inference. With more overlapped frames, more robust detection results from the RODNet can be achieved, however, the inference time will also increase. The training and inference of the RODNet models are run on an NVIDIA Quadro GV100 GPU, and the time consumed is reported in Fig. 4.10. Here, we show the AP of three building architectures (Vanilla, HG, and HG with temporal inception) for the RODNet, and use 100 ms as a reasonable real-time threshold. The results illustrate that the RODNet with a relatively simpler vanilla backbone can achieve real-time. As for the HG backbone, it steps across the real-time threshold when the overlapping length increases. Moreover, HG without temporal inception layers is slightly faster than HG with all network components.

4.4 Radar Multi-Object Tracking (RadarMOT)

After radar-based object detection is accomplished and based on the substantial advantages of the mmWave radars, we are motivated to explore the potential of radars to serve as a primary sensor, or a solid complementary sensor, for object detection ***and tracking*** in autonomous driving scenarios.

Here, we introduce a new task, named radar-based multi-object tracking (**RadarMOT**), to detect, classify, and track every object in the scene captured by the mmWave radar on an autonomous vehicle at the same time.

First, we introduce a new RadarMOT benchmark for radar-based multi-object detection and tracking, based on the ROD2021 dataset [93], using the widely used radar dataset representation – radio frequency (RF) images [104, 94]. The tracking annotations are obtained using our semi-automatic labeling method, which applies simple online and real-time tracing (SORT) [10] algorithm on the original detection annotations with manual refinement. Besides, we introduce the evaluation metrics for RadarMOT, which are revised from those for vision-based MOT [9, 58, 51, 76].

Second, we propose a novel RadarMOT framework, as shown in Figure 4.11, to address this multi-object detection and tracking task for mmWave radars in autonomous driving scenarios. A deep neural network is first proposed for jointly detecting objects and extracting

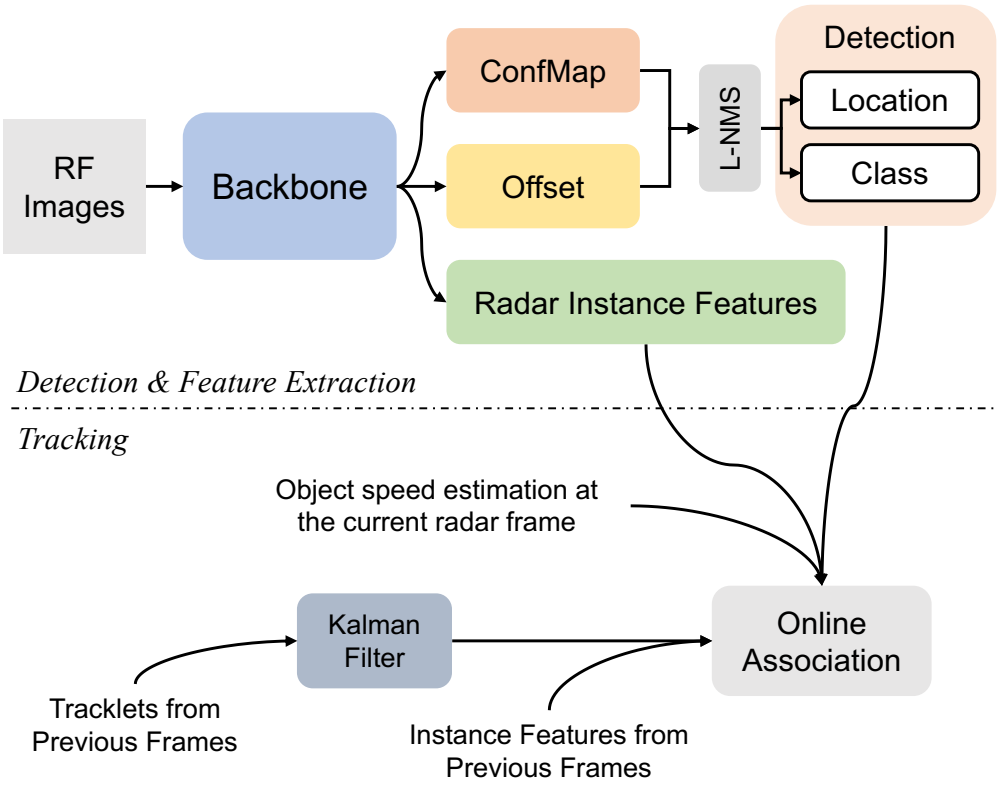


Figure 4.11: The pipeline of the proposed RadarMOT method, which can be divided into two parts. The upper part includes a unified network that accomplishes object detection and radar instance feature prediction tasks from the input RF images, while the lower part takes the detections, radar instance features, and object relative speed from the current frame and the aggregated information from existing tracklets based on detections of previous frames for online multi-object tracking.

corresponding radar instance features from a snippet of input RF images. Then, a Kalman filter based online tracking algorithm, which considers object 3D locations and object relative speeds from radar, is followed to predict the next location and speed from detections of previous frames and infer the motion distance with the actual detections and speeds of the current frame. The object association between the current frame's detections and existing tracklets, which combines the motion and radar instance feature distances, is accomplished by the Hungarian assignment algorithm [47].

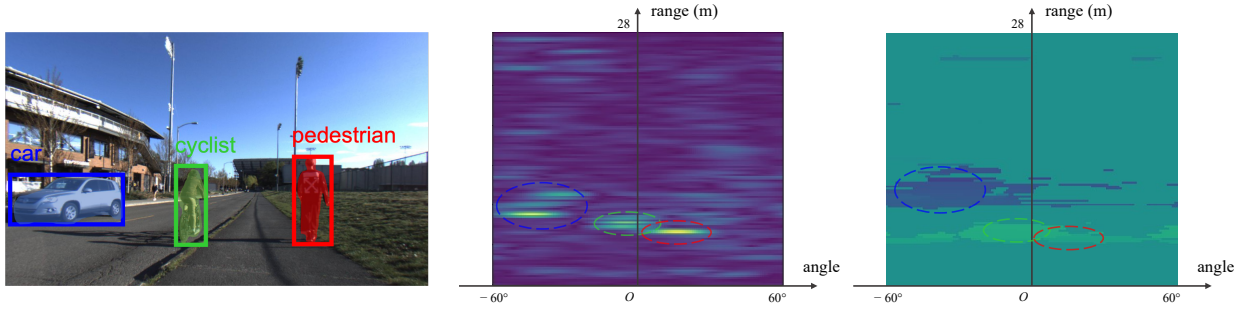
Table 4.5: Summary of some selected related MOT works with camera or radar modalities.

Modality	Method	In-Model Detector	In-Model Feature Extractor	Input for Tracking[1]	Class-Aware[2]	Identity-Aware[3]	Target Scenes
Camera	SORT [10]			`loc`	*	✓	Visible
	DeepSORT [98]		✓	`loc+fea`	*	✓	Visible
	MOANA [87]		✓	`loc+fea`	*	✓	Visible
	TNT [91]		✓	`loc+fea`	*	✓	Visible
	Tracktor [8]	✓	✓	`loc+fea`	✓	✓	Visible
	JDE [97]	✓	✓	`loc+fea`	✓	✓	Visible
	CenterTrack [105]	✓		`loc`	✓	✓	Visible
	FairMOT [102]	✓	✓	`loc+fea`	✓	✓	Visible
Radar	Scheel et al. [78]			`loc+speed`		✓	All-Weather
	Akita et al. [4]			`loc`	✓		All-Weather
	Haag et al. [35]			`loc+speed`			All-Weather
	RadarMOT (Ours)	✓	✓	`loc+fea+speed`	✓	✓	All-Weather

[1] Input for Tracking: `loc` for detection locations, `fea` for embedding features, and `speed` for radial speed derived from radar.
[2] Class-Aware: The method is able to distinguish among different object classes or between foreground and background.
[3] Identity-Aware: The method is able to track multiple identities at the same time.
* Not handle by the method itself because the object class attribute is usually included in the given detections.

Table 4.6: Statistics of the RadarMOT benchmark dataset. Each column represents the numbers of sequences, frames, detections, and tracks in the training and testing sets.

Scenarios	Training Set							Testing Set						
	Seqs	Frames	Dets	Tracks				Seqs	Frames	Dets	Tracks			
				`total`	`ped`	`cyc`	`car`				`total`	`ped`	`cyc`	`car`
Parking Lot	25	22480	39480	49	28	12	9	3	2700	5400	6	5	1	0
Campus Road	9	8100	15755	49	18	14	17	3	2700	2541	18	4	3	11
City Street	3	5080	10344	881	155	14	712	3	3354	5847	71	4	0	67
Highway	3	5074	2857	62	0	0	62	1	1683	4410	23	0	0	23
Overall	**40**	**40734**	**68447**	**1041**	**201**	**40**	**800**	**10**	**10437**	**18198**	**118**	**13**	**4**	**101**



(a) RGB image (b) RF-RA image (BEV) (c) RF-*s* image (BEV)

Figure 4.12: An example of corresponding RGB and two RF images, i.e., RF-RA and RF-*s*. A car, a pedestrian, and a cyclist are shown in all three images. But unlike the RGB image, in either of the two RF images, object class and features are almost impossible to interpret by human eyes.

4.4.1 RadarMOT Benchmark Dataset

Dataset Details In this work, we create our RadarMOT dataset based on the ROD2021 dataset [93], which is a subset of the CRUW dataset. It contains 50 sequences, including 40 for training and 10 for testing, under four different driving scenarios, i.e., parking lot (PL), campus road (CR), city street (CS), and highway (HW). Overall, there are more than 1000 object trajectories in the training set and more than 100 object trajectories in the testing set with three different object classes, i.e., pedestrian, cyclist, and car. Besides, there are also several vision-hard sequences of poor image quality, i.e., weak/strong lighting, blur, etc. Those scenarios can serve as good examples to show the advantages of radar compared with camera-based methods in adverse driving conditions.

Semi-Automatic Tracking Annotation Here, we introduce our semi-automatic tracking labeling method used for creating our RadarMOT dataset, which can significantly make data annotation work for the RadarMOT task more efficient. The annotations on the ROD2021 dataset only include object detections. Here, we incorporate the RadarMOT annotations based on the following procedures:

- Assign 2D bounding boxes for each object annotated in camera images, and 3D localized then projected into RF images. The bounding box size is defined as the real size of the default corresponding object class.

- Run the SORT algorithm [10] using the bounding boxes in RF images. Manually correct the tracking results from the SORT algorithm, i.e., removing, merging, and splitting the trajectories.

- Linearly interpolate the missing detections for each object trajectory, and smooth each object trajectory using the Savitzky–Golay filter [70]. The window size of the filter is defined as follows,

$$w^j = \max\left\{\frac{l^j}{10}, 3\right\}, \tag{4.18}$$

  where l^j is the length of trajectory j. Other parameters are empirically finetuned for better smoothing results.

4.4.2 RadarMOT Framework

In this section, we will introduce our proposed RadarMOT framework, which is able to jointly detect and track multiple objects with radar data as the only input. Specifically, the framework contains two main components:

- A unified detection and instance feature prediction network: A multi-task neural network that predicts detection confidence maps, regression offsets, and radar instance feature maps in one shot.

- Online Tracking: An online tracking algorithm, based on Kalman filter motion prediction and Hungarian assignment algorithm for detection association between frames, that jointly utilizes object 3D locations and object radar instance features.

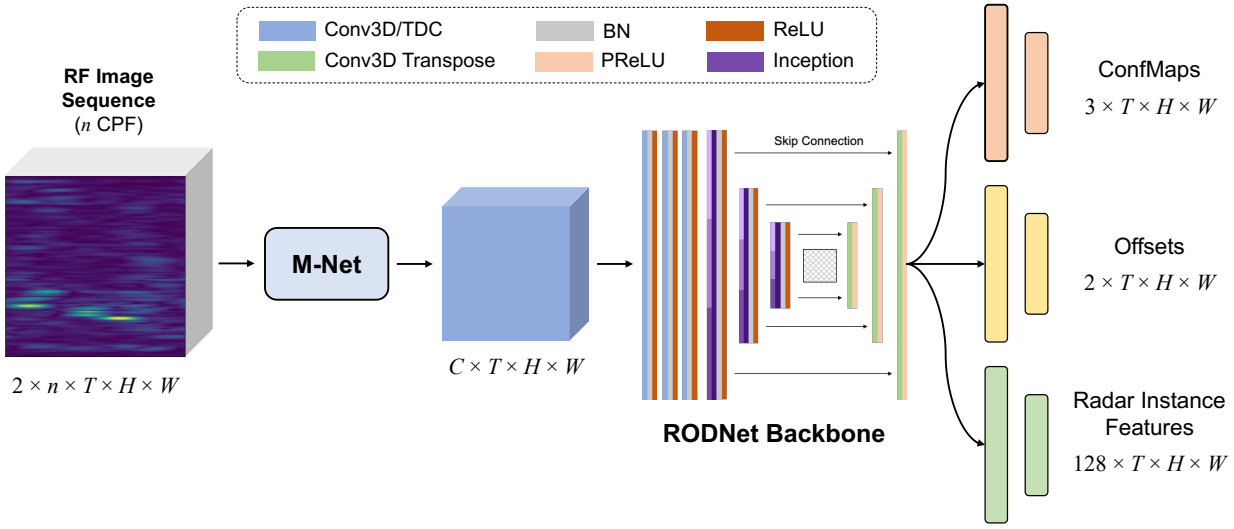


Figure 4.13: The network architecture of the proposed RadarMOT framework. The input radar data has a dimension of $2 \times n \times T \times H \times W$, where 2 represents the real and imaginary part values in the RF images, n is the number of chirps per frame (CPF) sampled from the radar data. The RODNet backbone, which is a CNN architecture with 3D convolutions, follows the definition in [95], except modifying the output channels of the last layer from the number of classes i.e., 3, to the number of instance feature channels, i.e., 32, in our experiments. Three heads, i.e., ConfMap head, offset head, and radar instance feature head, are appended to the backbone for multi-task learning. We use 2D convolution layers in these heads to reduce computational complexity, because the temporal features should be properly extracted by the 3D CNNs in the backbone.

4.4.2.1 Backbone Network

Unlike the backbone networks used for feature extraction from RGB images, e.g., ResNet [37], Deep Layer Aggregation [100], etc., the feature extraction for RF images is different. As mentioned in RODNet [94, 95], the feature extraction from the RF images should also consider the temporal domain, which conveys rich speed and texture information of the scene. Therefore, 3D convolutions are adopted for feature extraction. Since RODNet is an advanced object detection network based on RF images, in this work, we also use that as our backbone network for feature extraction. Note that to make RODNet as a backbone network for the later stages, we remove the last layer for ConfMap generation, and append several different heads to achieve multi-task learning.

4.4.2.2 Detection Branch

After discriminative feature maps are created from the backbone network, we first introduce the detection branch for the radar object detection task, including classification and 3D localization. There are two different heads in this branch, i.e., confidence map (ConfMap) head, and location offset head. After that, we utilize the location-based non-maximum suppression (L-NMS) [94] as our post-processing algorithm to get the final detection results.

Specifically, ConfMap head outputs $\mathbf{d} \in \mathbb{R}^{C\times T\times H\times W}$, where C is the number of object classes, T is the number of frames in the input RF snippet, H and W are the height and width of the ConfMaps, which are defined by the ground truth object 3D locations. Here, similar to CenterNet [106], we use Gaussian distributions to set the ConfMap values around the ground truth object location (r^i, a^i), whose mean is the integral object location $(\lfloor r^i \rfloor, \lfloor a^i \rfloor)$, and the variance is related to the object class and scale information, following the rule in [94]. The loss function is defined as binary cross entropy loss

$$\mathcal{L}_{\text{conf}} = -\sum_{c=1}^{C}\sum_{i,j} \mathbf{d}^c_{i,j} \log \hat{\mathbf{d}}^c_{i,j} + \left(1-\mathbf{d}^c_{i,j}\right) \log\left(1-\hat{\mathbf{d}}^c_{i,j}\right), \tag{4.19}$$

where $\mathbf{d}$ represents the ground truth ConfMaps, $\hat{\mathbf{d}}$ represents the predicted ConfMaps, and (i, j) represents the pixel indices of the ConfMaps.

Object location offset head, as inspired in the CenterNet [106], is aimed to localize objects more precisely. The mapping from the real bird's-eye view (BEV) locations (in meters) to integral pixel values is discrete, resulting in precision error. Thus, this offset head predicts a continuous offset relative to the object location. The output dimension of the offset head is $2 \times T \times H \times W$. The offset can be represented as $\mathbf{o}^i = (r^i - \lfloor r^i \rfloor, a^i - \lfloor a^i \rfloor)$. The loss function is defined using l_1 loss,

$$\mathcal{L}_{\text{off}} = \sum_i \left\| \mathbf{o}^i - \hat{\mathbf{o}}^i \right\|_1, \tag{4.20}$$

where $\hat{\mathbf{o}}^i$ represents the estimated offsets for detection i.

4.4.2.3 Radar Instance Feature Branch

The radar instance feature branch is to produce embedding features for the detected objects that can be used to distinguish their identities and to associate with the existing tracked identities. For example, the same object appearing in different frames should have similar radar instance features with a smaller distance, while the distance between instance features of different objects should be larger. This idea is frequently used in metric learning for the vision-based tracking methods [98, 8, 102], but seldom considered in radar-based applications.

To obtain the radar instance features, we propose an additional 2D convolution layer after the backbone network. The output feature map $\mathcal{E}$ has a dimension of $E \times T \times H \times W$, where $E = 128$ is the feature dimension.

To train this radar instance feature branch, since no ground truth is available, we treat it as a classification problem, that is shown to achieve better performance than metric learning [97], whose class IDs are assigned as object tracking IDs. For the detection at (r^i, a^i), the corresponding predicted instance features are located at $\mathcal{E}(r^i, a^i) \in \mathbb{R}^{128}$. We use a fully connected layer with softmax to map the features to a class probability distribution

$\hat{\mathbf{p}}(k) \in \mathbb{R}^K$, where K is the number of identities, i.e., track IDs, in the training dataset. The loss is defined as

$$\mathcal{L}_{\text{emb}} = -\sum_{i}\sum_{k=1}^{K} \mathbf{p}^i(k) \log(\hat{\mathbf{p}}^i(k)), \tag{4.21}$$

where $\mathbf{p}^i(k)$ is the one-hot ground truth tracking label associated with the k-th identity for detection i in the training dataset.

4.4.2.4 Network Training

We jointly train the detection branch and radar instance feature branch in one-shot by adding the above losses together as

$$\mathcal{L}_{\text{total}} = \lambda_1 \mathcal{L}_{\text{conf}} + \lambda_2 \mathcal{L}_{\text{off}} + \lambda_3 \mathcal{L}_{\text{emb}}. \tag{4.22}$$

Empirically, we use $\lambda_1 = \lambda_2 = \lambda_3 = 1$ in our experiments.

4.4.2.5 Online Tracking

Our online radar tracking method is inspired by [97], with some algorithmic modifications specifically for our radar tracking problem. The tracking algorithm is shown in Algorithm 3.

Tracklet State Definition We define a motion state ξ^j and an instance state ϵ^j for each tracklet. Here, the instance state $\epsilon^j \in \mathbb{R}^{128}$ is defined by the radar instance features predicted from the network. While, the motion state is defined as

$$\xi^j = (x, y, v, \dot{x}, \dot{y}, \dot{v}), \tag{4.23}$$

where (x, y) represents object 2D location in BEV (in meters) transformed from polar coordinates (r, a); v represents the estimated relative radial speed from the radar; $\dot{x}, \dot{y}, \dot{v}$ are the derivatives of the above variables, respectively.

The motion states of tracklets indicate the location and motion of the tracked objects in the BEV coordinates, and can be probabilistically predicted by a Kalman filter [43].

Online Association In this section, we follow [102] to conduct a *two-stage* association to assign the detections to tracklets. The association step is managed to align the detection set $\mathbf{d}_t$ in the current frame t with the existing tracklet set $\mathbf{tr}_{t-1}$ in frame $t-1$. In the first stage, we utilize the Mahalanobis distance of predicted motion and cosine distance of instance features as the association criteria in the Hungarian assignment algorithm to address the general matching problem between $\mathbf{d}_t$ and $\mathbf{tr}_{t-1}$. After that, we consider using the more straightforward object location similarity (OLS) (see Eq. 4.17) as the criterion in the second stage to handle the unmatched detections in the first stage.

More specifically, in the first stage, we calculate the pairwise Mahalanobis distance D_m between the detected object locations and speeds $\{(x, y, v)\}$ in $\mathbf{d}_t$ and the predicted object locations and speeds $\{(\hat{x}, \hat{y}, \hat{v})\}$ in $\mathbf{tr}_{t-1}$ by a Kalman filter. As for the instance state, we calculate the pairwise cosine distance D_e between the instance features of the detected radar objects in $\mathbf{d}_t$ and the continuously updated instance features in $\mathbf{tr}_{t-1}$ (see Eq. 4.25). Then, the overall pairwise distance used in the Hungarian assignment is defined as

$$D = \alpha D_e + (1 - \alpha) D_m. \tag{4.24}$$

Then, the association between $\mathbf{d}_t$ and $\mathbf{tr}_{t-1}$ is achieved using the Hungarian assignment algorithm with a matching threshold of τ_1 to remove weakly associated assignments.

In the second stage, we try to associate the unmatched detections in the first stage due to unreliable instance features or inaccurate motion state prediction. To achieve this association, we compare the detected object locations in $\mathbf{d}_t$ with the predicted object locations of unmatched trajectories from $\mathbf{tr}_{t-1}$ by Kalman filter.

Unlike the association for vision-based tracking, which uses 2D bounding box IoU as a criterion to match the detections, our RadarMOT task does not have the definition of the

bounding box in RF images. Therefore, we adopt OLS (Eq. 4.17) to replace the role of IoU as the matching criterion Then, we apply the Hungarian assignment algorithm again using OLS, with matching threshold τ_2.

For both of the above matching stages, we apply an additional radial speed validation to filter out the wrong tracklet-detection matches by comparing the radial object motion based on adjacent frames of associated detections with the relative radial speed directly from the radar sensor. Here, the radial object motion is calculated from the history of the tracklets and projects the motion to the directions from the ego-car to the target objects. Note that we only apply this speed validation for the relative radial speed within the unambiguous maximum speed for our radar sensor. The final speed (km/h) of the tracked radar objects relative to the ego-car can also be inferred based on the adjacent frames of associated detections, as shown in Fig. 4.14.

Then, for each detection matched with a tracklet, we adopt the idea of "detection-by-tracking", i.e., using the robust object class information in our tracking results to correct the object classification mistakes made in the object detection step (i.e., ConfMap prediction).

After the two-stage association, the instance state follows a simple updating strategy [102] for feature smoothing to improve its stability and robustness. Here, the momentum updating strategy can be described as

$$\epsilon_t^j = \beta \epsilon_{t-1}^j + (1-\beta) e_t^j, \tag{4.25}$$

where e_t^j is the radar instance features of the observation j in frame t, β is the momentum term.

To allow our tracking algorithm to be able to catch the reappeared objects after missing them for a short time, we keep the unmatched tracklets alive and perform associations with new detections in the following 30 frames. If no more associations afterward, the tracklets will be removed, otherwise will be revived for subsequent tracking associations.

4.4.3 Evaluation Metrics

In this section, we will introduce our modified evaluation metrics for RadarMOT in detail, which are revised based on those metrics designed for the vision-based MOT [58].

Predicted detections and ground truth matching A bijective mapping is constructed between predicted and ground truth detections in each frame. The matched predictions and ground truths are identified as true positives (TPs). The unmatched predictions are identified as false positives (FPs). The unmatched ground truths are identified as false negatives (FNs). The matching between predictions and ground truths is based on the object location similarity (OLS) as defined in [96]. The OLS between two radar detections i and j is defined in Eq. 4.17. Therefore, we can calculate the OLS between each detection and ground truth pair, and the matching step is controlled by a threshold δ, such that detections are matched when $\text{OLS}(i, j) \geq \delta$.

Identity Switch (IDSW) An IDSW occurs when object identities are swapped or when a track is lost and re-initialized with a different identity. IDSWs only measure association errors compared to the single previous TP.

Multi-Object Tracking Accuracy (MOTA) MOTA measures three types of tracking errors, i.e., the detection errors of FNs and FPs, and the association error of IDSW. The final MOTA is defined as follows,

$$\text{MOTA} = 1 - \frac{|\text{FN}| + |\text{FP}| + |\text{IDSW}|}{n_{\text{gt}}}, \tag{4.26}$$

where n_{gt} is the number of ground truth detections.

IDF1 Score IDF1 calculates a bijective mapping between predicted trajectories and ground truth trajectories based on the scores of identity true positives (IDTPs), identity false positives (IDFPs), and identity false negatives (IDFNs). IDTPs are matches on the overlapping

part of the trajectories, i.e., $\text{OLS}(i,j) \geq \delta$, IDFPs and IDFNs are the unmatched predictions and ground truths, respectively. The ID-Recall, ID-Precision, and IDF1 scores can be defined as

$$\text{ID-Recall} = \frac{|\text{IDTP}|}{|\text{IDTP}| + |\text{IDFN}|}, \tag{4.27}$$

$$\text{ID-Precision} = \frac{|\text{IDTP}|}{|\text{IDTP}| + |\text{IDFP}|}, \tag{4.28}$$

$$\text{IDF1} = \frac{|\text{IDTP}|}{|\text{IDTP}| + 0.5|\text{IDFN}| + 0.5|\text{IDFP}|}. \tag{4.29}$$

Overall, MOTA, IDF1, and IDSW are considered as three main evaluation metrics in the RadarMOT benchmark as well as the experiments section (Section 4.4.4).

4.4.4 Experiments

4.4.4.1 RadarMOT Results

We train our proposed RadarMOT framework using the training sequences in the RadarMOT benchmark dataset and evaluate the performance of the testing sequences. Some qualitative results are shown in Figure 4.14, illustrating our proposed RadarMOT framework can achieve favorable detection and tracking performance with only RF image sequences as the input.

The quantitative results of the RadarMOT framework, evaluated using the metrics introduced in Section 4.4.3, are shown in Table 4.7. Here, we compare our results with the following methods based on some popular vision-based trackers:

1) FairMOT [102]: train the original FairMOT network with DLA-34 backbone [100] using the RadarMOT benchmark dataset.

2) RODNet [95] + SORT [10]: take the detection results from RODNet, assign 2D bounding boxes in RF images to each detection, and run the SORT algorithm on the 2D bounding boxes.

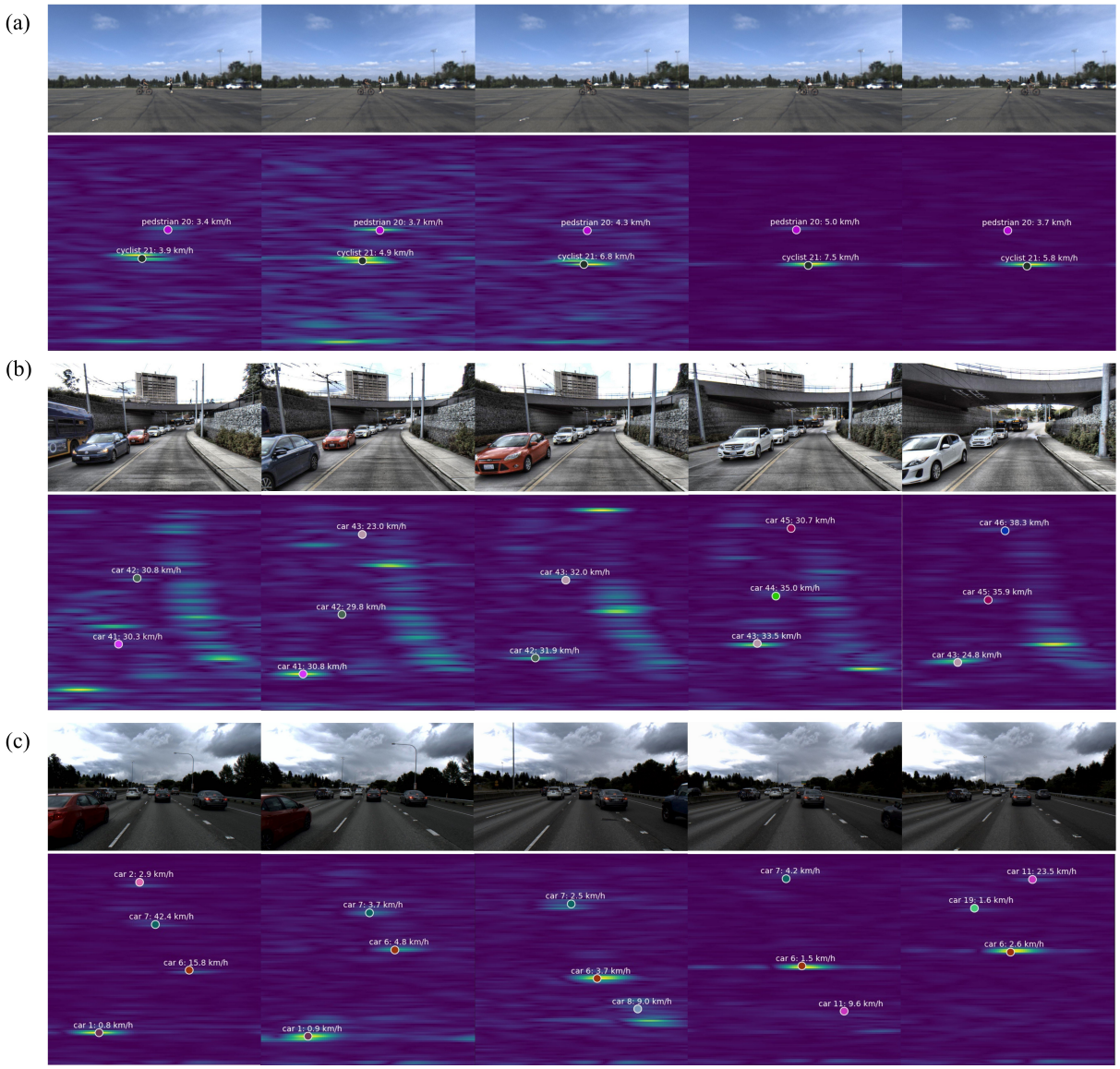


Figure 4.14: The qualitative results of the proposed RadarMOT framework. Note that radar's field of view is 0-25m, ±60°. The class and the relative speed (relative to the ego-car, unit: km/h) of each detected object are labeled on the RF-RA images. (Better view with color.)

From Table 4.7, directly using FairMOT to achieve joint detection and tracking cannot achieve a good result. The reason is that the backbone of FairMOT is based on 2D CNNs and designed for vision-based MOT. This kind of network cannot handle the feature prediction from our radar RF images, resulting in poor detection performance. Although RODNet+SORT method can achieve fair performance, especially on MOTA due to reliable radar object detection performance, our proposed RadarMOT framework achieves better MOT performance overall, as seen from the 6% IDF1 improvement.

As for the running speed of our proposed framework, all the methods can achieve more than 10 FPS, which is a reasonable criterion for real-time operation. Among them, the FairMOT baseline is much faster because of using 2D CNNs, however, sacrificing the performance by a large margin. Comparing RODNet+SORT and our proposed RadarMOT framework, the joint detection and tracking framework can be faster than the separated steps. Note that all the inferences are running on a single NVIDIA GV100 GPU.

4.4.4.2 Comparison with Vision-Based MOT

To show the strengths of radar-based MOT compared with vision-based MOT, we also implement Faster R-CNN [74] and DeepSORT [98] on the RGB images in the ROD2021 dataset. Since the RGB and RF images are synchronized, the RGB and RF sequences show exactly the same scenarios but with different sensors. To evaluate the performance of the vision-based MOT, we manually annotate the MOT ground truths for a part of the RGB image sequences in the ROD2021 dataset and evaluate them in the same field of view as radar. The comparison is shown in Table 4.8 and some examples are shown in Fig. 4.15. We separate the visible and vision-hard scenarios for evaluation, since the vision-based MOT is not working under vision-hard scenarios.

From the experimental results, our RadarMOT framework can achieve comparable performance with the vision-based MOT in visible driving scenarios and it can still work well in vision-hard scenarios, where vision-based MOT almost fails.

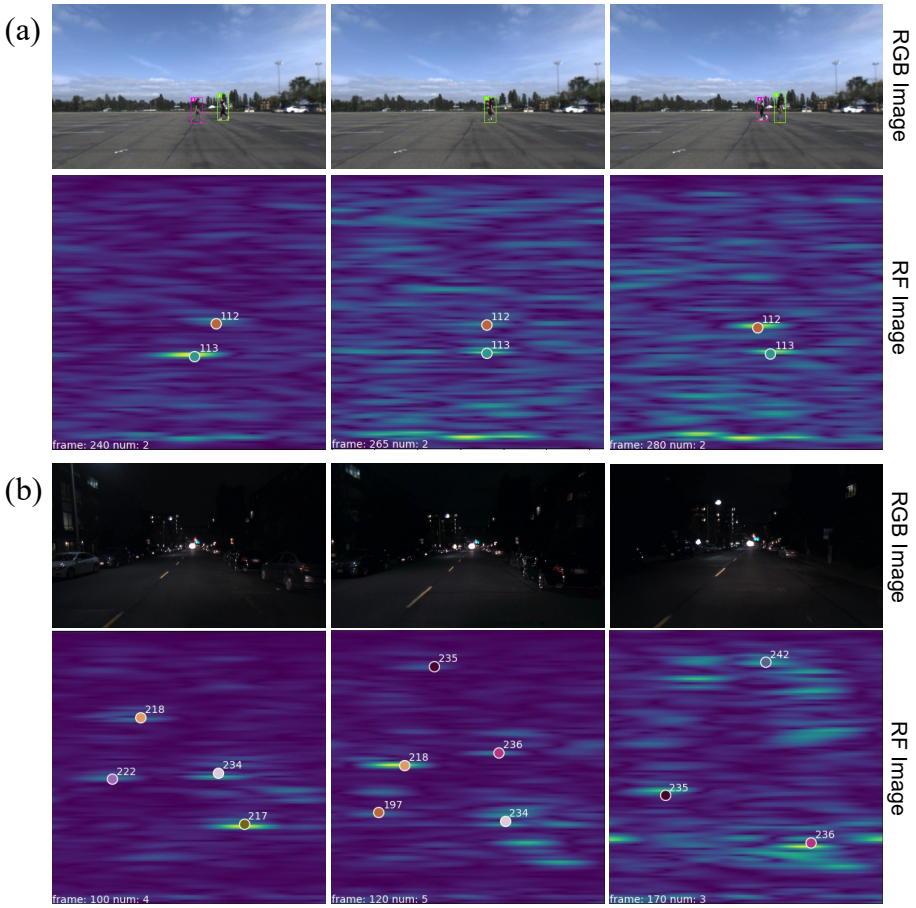


Figure 4.15: Examples to illustrate the advantages of radar-based MOT compared with vision-based MOT. The radar-based MOT can effectively reduce ID switches and work well at night. Note that radar's field of view is 0-25m, ±60°. (Better view with color.)

4.4.4.3 Ablation Studies

Two-stage tracking analysis We first analyze the contributions for each stage in our two-stage tracking mechanism in Table 4.9. We can find that with either stage only, the tracking performance decreases to some extent. This shows the effective complement between these two stages during tracking.

Performance improvement with radar's radial speed We also try to use the original Kalman filter design in vision-based methods, without the relative radial speed estimation from radar. The motion state without relative radial speed is defined as

$$\xi^j = (x, y, \dot{x}, \dot{y}) \,, \tag{4.30}$$

where x and y represent object location in BEV (in meters) transformed from polar coordinates (r, a); $\dot{x}, \dot{y}$ are the derivatives of the above variables, respectively.

The experimental results of this setting is shown in Table 4.9. It shows that with the relative radial speed obtained from radar, the performance of our RadarMOT framework can be improved to some extent.

Algorithm 3: Tracking Algorithm in RadarMOT

Input: An RF-RA sequence S_{RA}; the corresponding speed maps S_s; detection score threshold τ_{det}; tracking score threshold τ_1, τ_2.
Output: Trajectories $\mathcal{T}$ in the RF sequence.
Initialization: $\mathcal{T} \leftarrow \emptyset$;
for *each snippet* s_{RA} *in* S_{RA} **do**
  $\mathcal{D}, \mathcal{F} \leftarrow \texttt{Network}(s_{RA})$; `// get detections` $\mathcal{D}$ `and radar instance features` $\mathcal{F}$
  $\mathcal{D} \leftarrow \texttt{GetSpeed}(S_s, \mathcal{D})$;
  for d *in* $\mathcal{D}$ **do**
    if $d.score < \tau_{det}$ **then**
      $\mathcal{D} \leftarrow \mathcal{D} \setminus \{d\}$;
      $\mathcal{F} \leftarrow \mathcal{F} \setminus \{\mathcal{F}[d]\}$;
    end
  end
  for t *in* $\mathcal{T}$ **do**
    $t \leftarrow \texttt{KalmanFilter}(t)$;
  end
  `// first-stage association`
  $D_m \leftarrow \texttt{MahDist}(\mathcal{T}_D, \mathcal{D})$;
  $D_e \leftarrow \texttt{CosineDist}(\mathcal{T}_F, \mathcal{F})$;
  $D \leftarrow \alpha D_e + (1-\alpha) D_m$;
  Associate $\mathcal{T}$ and $\mathcal{D}$ using distance D with threshold τ_1 ;
  $\mathcal{D}_{remain} \leftarrow$ remaining detections from $\mathcal{D}$;
  $\mathcal{T}_{remain} \leftarrow$ remaining tracks from $\mathcal{T}$;
  Relative radial speed validation for object motion ;
  `// second-stage association`
  $D_{OLS} \leftarrow 1 - \texttt{OLS}(\mathcal{T}_D, \mathcal{D})$;
  Associate $\mathcal{T}_{remain}$ and $\mathcal{D}_{remain}$ using OLS distance D_{OLS} with threshold τ_2 ;
  $\mathcal{D}_{remain} \leftarrow$ remaining detections from $\mathcal{D}_{remain}$;
  $\mathcal{T}_{remain} \leftarrow$ remaining tracks from $\mathcal{T}_{remain}$;
  $\mathcal{T} \leftarrow \mathcal{T} \setminus \mathcal{T}_{remain}$;
  Relative radial speed validation for object motion ;
  `// updates`
  Correct the classes of the detected objects according to the matched tracklets ;
  Update Kalman filter states ;
  Update radar instance features using Eq. 4.25 ;
  `// initialize new tracks`
  for d *in* $\mathcal{D}_{remain}$ **do**
    $\mathcal{T}_D \leftarrow \mathcal{T}_D \cup \{d\}$;
    $\mathcal{T}_F \leftarrow \mathcal{T}_F \cup \{d\}$;
    $\mathcal{T} \leftarrow [\mathcal{T}_D, \mathcal{T}_F]$;
  end
end
Return: $\mathcal{T}$

Table 4.7: Evaluation results for the proposed RadarMOT framework on the RadarMOT benchmark.

Method	IDF1↑	IDP↑	IDR↑	MOTA↑	IDSW↓	Recall↑	Precision↑	FPS↑
FairMOT (Visible)	43.4	48.2	39.5	65.0	219	44.0	66.5	
FairMOT (Vision-Hard)	29.4	34.5	25.6	25.0	112	43.6	61.4	24
FairMOT (Overall)	40.9	45.8	36.9	57.4	331	44.0	66.5	
RODNet + SORT (Visible)	76.0	81.7	71.1	74.2	60	80.9	93.1	
RODNet + SORT (Vision-Hard)	71.5	80.2	64.7	59.7	18	69.0	87.0	11
RODNet + SORT (Overall)	75.1	81.2	69.7	70.9	78	78.6	**92.0**	
RadarMOT (Visible)	82.5	87.8	77.8	74.3	26	81.6	92.0	
RadarMOT (Vision-Hard)	74.8	80.6	69.8	60.3	12	73.6	85.0	13
RadarMOT (Overall)	**81.1**	**86.5**	**76.3**	**71.6**	**38**	**80.0**	90.7	

Table 4.8: Comparison between vision-based MOT and radar-based MOT on a subset of the ROD2021 dataset.

Method	IDF1↑	IDP↑	IDR↑	MOTA↑	IDSW↓	Recall↑	Precision↑
DeepSORT (Visible)	73.2	83.1	65.3	67.4	121	73.2	93.3
RadarMOT (Visible)	76.7	82.0	72.0	77.4	45	82.8	94.2
DeepSORT (Vision-Hard)	–	–	–	–	–	–	–
RadarMOT (Vision-Hard)	74.8	80.6	69.8	60.3	12	73.6	85.0

Table 4.9: Ablation studies for the RadarMOT framework.

1st stage	2nd stage	w/ speed	IDF1↑	IDP↑	IDR↑	MOTA↑	IDSW↓	Recall↑	Precision↑
✓			75.9	81.5	71.0	70.2	103	78.9	90.6
	✓		72.2	78.1	67.1	69.6	75	77.9	**90.8**
✓	✓		79.3	84.4	74.8	71.3	55	**80.1**	90.4
✓	✓	✓	**81.1**	**86.5**	**76.3**	**71.6**	**38**	80.0	90.7

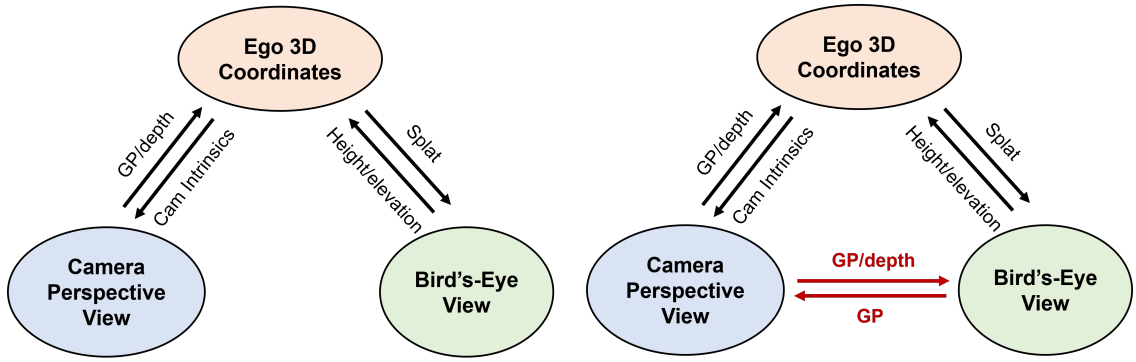
Chapter 5

OBJECT PERCEPTION VIA CROSS-MODALITY CHECK

After the adverse driving scenarios are considered in Chapter 4, we propose a novel Radar-Camera Cross-modality Check (RCCC) pipeline to take advantage of both camera and radar sensors to obtain accurate and robust 3D object perception results. In this chapter, we first introduce a radar-camera bilateral coordinate projection (BCP) to bridge the relationship between the camera's perspective view (PV) and the radar's bird's-eye view (BEV) in Section 5.1. Second, the proposed radar-camera cross-modality check pipeline is described in Section 5.2 to obtain better 3D object perception results using the 3D bounding boxes from the camera and object locations from the radar. Third, the experimental results on the CRUW2022 dataset are shown in Section 5.3

5.1 Radar-Camera Bilateral Coordinate Projection

In classical autonomous geometric projections, we usually make projections among three different coordinates, i.e., camera pixel coordinates, a.k.a., perspective view (PV) coordinates, bird's-eye view (BEV) coordinates, and ego 3D coordinates, as shown in Figure 5.1. The projection between camera PV and ego 3D can be achieved using the pinhole camera model as mentioned in Section 4.1. The difference between ego 3D and BEV is the height or elevation information so that is much easier. However, there is no projection defined between camera PV and BEV. It is necessary because most autonomous radars are just in BEV without elevation resolution so they cannot be processed in ego 3D coordinates. Therefore, in this chapter, we propose a new radar-camera bilateral coordinate projection (BCP) to address this issue.



(a) Classical autonomous geometric projections

(b) Proposed autonomous geometric projections by BCP

Figure 5.1: Geometric projection definitions. The proposed BCP can directly project between the camera's PV and the radar's BEV so that is a feasible and efficient projection for a typical autonomous radar sensor.

5.1.1 Notations

Coordinates. We first define some important coordinates used in our system: 1) Camera 2D pixel coordinates $(u, v) \in \mathcal{C}$; 2) Radar range-azimuth coordinates, i.e., bird's-eye view (BEV), $(r, \theta) \in \mathcal{R}$; 3) 3D camera coordinates with the origin at the camera center $(x^c, y^c, z^c) \in \mathcal{W}^c$; and 4) 3D radar coordinates with the origin at the radar $(x^r, y^r, z^r) \in \mathcal{W}^r$. The system takes two different kinds of detections from both camera and radar, i.e., object detection from RGB images and peak detection from RF images, as the input, and these detections are defined within $\mathcal{C}$ and $\mathcal{R}$.

Ground plane parameters. Instead of using the normal vector to define the ground plane, we define it by three ground plane parameters, i.e., two rotation angles and one offset, due to their convenience for the later discussion of Camera-Radar Bilateral Coordinate Projection (BCP). The ground plane parameters for each frame can be represented as

$$\mathbf{g} = [\varphi, \gamma, h], \tag{5.1}$$

where φ and γ respectively denote the pitch and roll angles of the ground plane w.r.t. the z-axis of $\mathcal{W}^c$; h is the camera height. The illustration of these ground plane parameters is shown in Fig. 5.2.

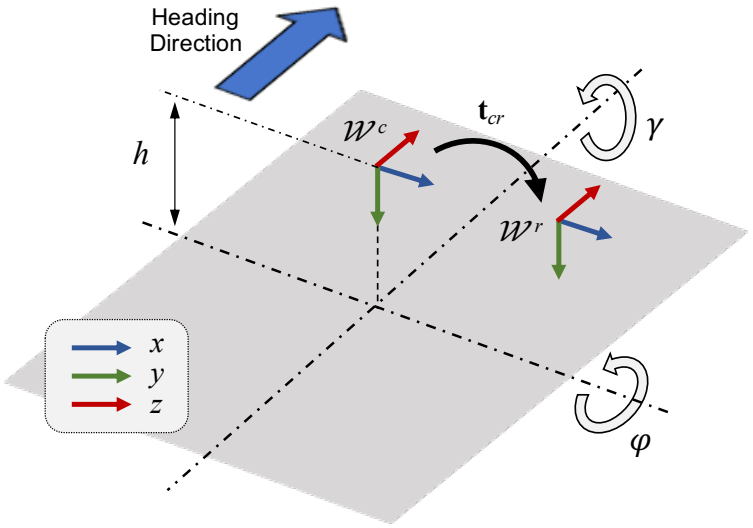


Figure 5.2: Illustration for 3D coordinates and ground plane parameters. The gray rectangle represents the ground plane. The camera and radar mounted on a vehicle are above the ground plane. $\mathbf{t}_{cr}$ is the translation vector from $\mathcal{W}^c$ to $\mathcal{W}^r$.

5.1.2 Projections

Considering the ground plane parameters defined above, we derive the camera-radar bilateral coordinate projection (BCP) for any point on the ground plane. We start from projecting a point $(r, \theta) \in \mathcal{R}$ to $\mathbf{x}^c = (x^c, y^c, z^c)$. Before that, we first do a polar to Cartesian coordinate transformation,

$$\begin{aligned} x^r &= r\sin(\theta), \\ z^r &= r\cos(\theta). \end{aligned} \tag{5.2}$$

This point can be transformed to $\mathcal{W}^c$ by sensor calibration of the translation from camera to radar $\mathbf{t}_{cr} = [t_{cr,x}, t_{cr,y}, t_{cr,z}]^\top$,

$$\begin{aligned} x^c &= x^r + t_{cr,x}, \\ z^c &= z^r + t_{cr,z}. \end{aligned} \tag{5.3}$$

Here, we ignore the rotation transformation between $\mathcal{W}^r$ and $\mathcal{W}^c$ since both sensors are well-calibrated with the same orientation.

To calculate y^c, we first consider the ground plane with only pitch rotation angle φ. Assuming a small φ, which is a valid assumption in driving scenarios, y^c_φ can be approximated by the similar triangles rule as

$$y^c_\varphi = \left(\frac{h}{\sin(\varphi)} - r\right) \cdot \tan(\varphi) \approx \left(\frac{h}{\sin(\varphi)} - r\right) \cdot \sin(\varphi) = h - r\sin(\varphi). \tag{5.4}$$

By taking the second rotation angle γ into consideration, we then have

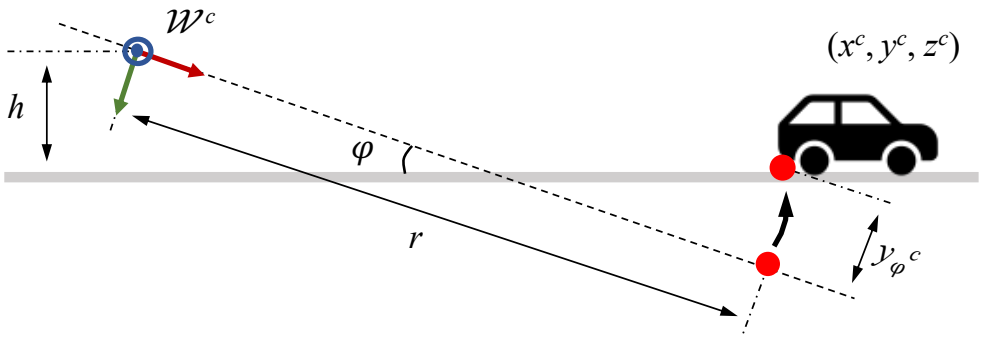
$$y^c_\gamma = x^c \tan(\gamma). \tag{5.5}$$

The illustrations for these two projections are shown in Fig. 5.3 (a) and (b). Therefore, the overall y^r can be represented as

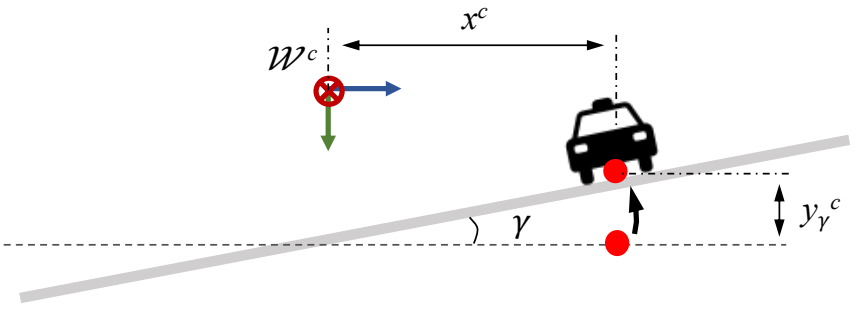
$$\begin{aligned} y^c &= y^c_\varphi - y^c_\gamma = h - r\sin(\varphi) - x^c\tan(\gamma) \\ &= h - \sqrt{(x^c)^2 + (z^c)^2} \cdot \sin(\varphi) - x^c\tan(\gamma). \end{aligned} \tag{5.6}$$

Finally, we project $\mathbf{x}^c \in \mathcal{W}^c$ to $\mathcal{C}$ by the camera intrinsic matrix $\mathbf{K}$,

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{K}\mathbf{x}^\mathbf{c} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x^c \\ y^c \\ z^c \end{bmatrix} \tag{5.7}$$



(a) Bilateral coordinate projection by ground plane parameter φ, h



(b) Bilateral coordinate projection by ground plane parameter γ

Figure 5.3: Geometric illustrations for our proposed bilateral coordinate projection. The projection is split into two steps, i.e., φ rotation and γ rotation, where the gray lines represent the ground planes. To clearly show the projection, these two steps are shown in two different perspectives, where (a) is based on the parameters φ and h; (b) is based on γ.

Thus, we have

$$\begin{aligned} u &= f_x \frac{x^c}{z^c} + c_x, \\ v &= f_y \frac{y^c}{z^c} + c_y. \end{aligned} \tag{5.8}$$

Put it together, we can project a point $(r, \theta) \in \mathcal{R}$ to $(u, v) \in \mathcal{C}$ by the following $\mathcal{R} \to \mathcal{C}$ projection, denoted as $P(\cdot)$,

$$[u, v]^\top = P(r, \theta), \tag{5.9}$$

where

$$\begin{aligned} u &= f_x \frac{r\sin(\theta) + t_{cr,x}}{r\cos(\theta) + t_{cr,z}} + c_x, \\ v &= f_y \frac{h - r\sin(\varphi) - (r\sin(\theta) + t_{cr,x})\tan(\gamma)}{r\cos(\theta) + t_{cr,z}} + c_y. \end{aligned} \tag{5.10}$$

After the $\mathcal{R} \to \mathcal{C}$ projection is properly derived, we would like to consider the other direction, i.e., $\mathcal{C} \to \mathcal{R}$ projection. Since the projection in Eq. 5.9 is a 2D-to-2D projection without losing any degrees of freedom (DoF), it can be reversed. Therefore, we define the $\mathcal{C} \to \mathcal{R}$ projection as

$$[r, \theta]^\top = P^{-1}(u, v). \tag{5.11}$$

Here, the $\mathcal{C} \to \mathcal{W}^c$ projection can be derived as

$$\begin{aligned} x^c &= \frac{h\hat{x}}{\sqrt{1 + \hat{x}^2}\sin(\varphi) + \hat{x}\tan(\gamma) + \hat{y}}, \\ z^c &= \frac{h}{\sqrt{1 + \hat{x}^2}\sin(\varphi) + \hat{x}\tan(\gamma) + \hat{y}}, \end{aligned} \tag{5.12}$$

where

$$\begin{aligned} \hat{x} &= \frac{x^c}{z^c} = \frac{u - c_x}{f_x}, \\ \hat{y} &= \frac{y^c}{z^c} = \frac{v - c_y}{f_y}. \end{aligned} \tag{5.13}$$

Note that $\mathcal{W}^c \to \mathcal{R}$ projection just contains the camera to radar translation and a Cartesian to polar coordinate transformation, which is trivial and will not be elaborated in detail.

5.2 Radar-Camera Cross-Modality Check Pipeline

In this section, we will introduce our proposed radar-camera cross-modality check (RCCC) pipeline for 3D object detection. We split the whole pipeline into three stages: detection alignment, alignment refinement, and alignment verification. First, we manage to align the detections from the two sensors by a ground plane optimization. Second, considering the aligned detections, which means the objects can be detected by both sensors, we propose an alignment refinement method to refine the object location accuracy for the aligned detections. Third, for the detections that are not aligned, which means the objects are just detected by one of the sensors, we use an alignment verification strategy to find the true positives and remove the false positives. The overall pipeline is shown in Figure 5.4.

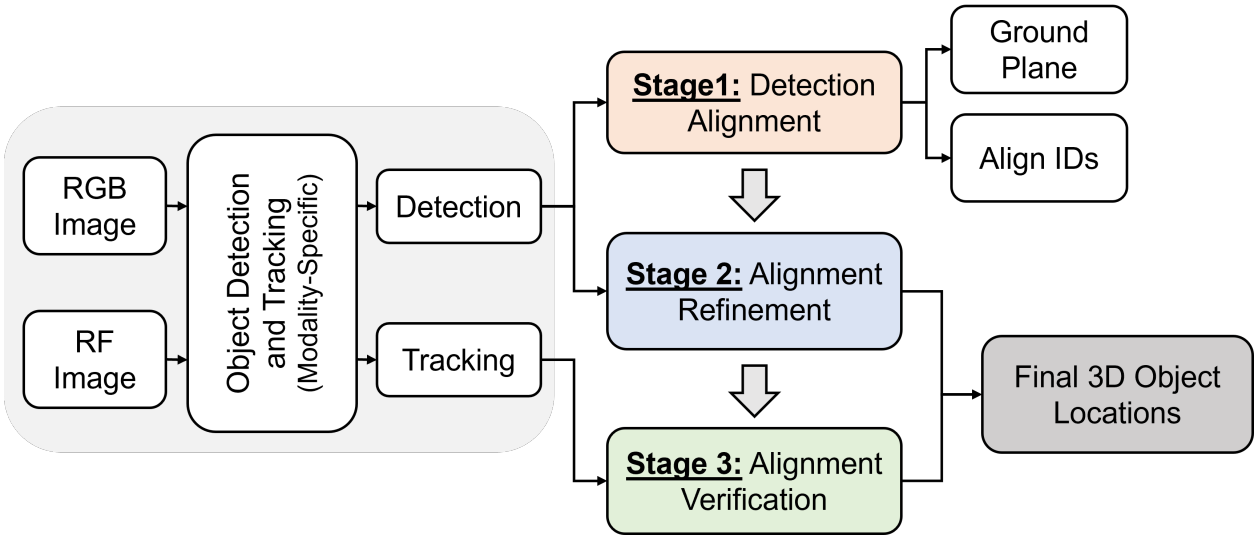


Figure 5.4: Illustration of the proposed RCCC pipeline. This is a three-stage pipeline, including detection alignment, alignment refinement, and alignment verification. The first stage takes the detection and tracking results from both camera and radar and manages to align them. The second stage considers the aligned detections and refines the detections by taking advantage of the results from both the camera and radar. The third stage focuses on the verification of those non-aligned detections.

5.2.1 Radar-Camera Detection Alignment

Based on the derived BCP, the connection between the camera and radar is established through the ground plane parameters. To align these two coordinates, we come up with a detection alignment strategy by defining detection alignment costs between 3D bounding box detections from the camera and detections from the radar. The proposed detection alignment is shown in Figure 5.5.

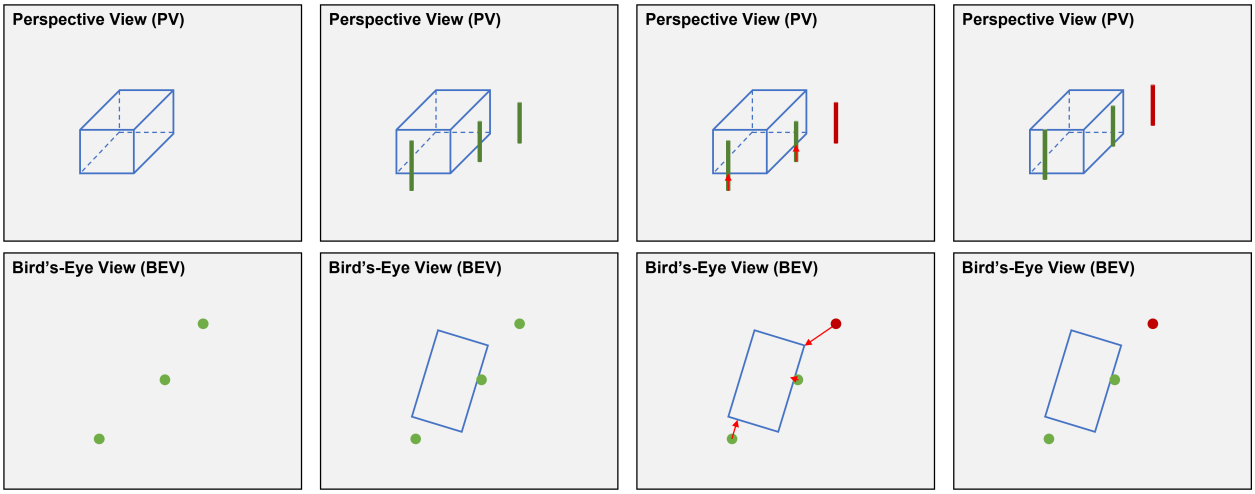


Figure 5.5: Radar-camera detection alignment illustration. (a) Detections from both camera and radar; (b) Projected detections by BCP; (c) Calculate alignment costs in both PV and BEV; (d) Optimize the ground plane and radar poles will be better aligned with camera detections.

Alignment cost in PV. The alignment between $\mathcal{R}$ and $\mathcal{C}$ is challenging because they are both 2D coordinates with significantly different perspectives. However, we find that the object height is very useful information during the alignment. The object will have very different heights at different distances in the RGB image due to the perspective. Thus, the object height is a special representation of its 3D location.

During the alignment, we first use different initial heights for different object classes.

Then, we project all the radar detections to RGB images by $\mathcal{R} \to \mathcal{C}$ projection with the object height. Each radar detection will be projected as a line segment in the RGB image, named as *radar pole*. We define the detection alignment cost for a radar detection i in PV to be

$$\ell_{\text{PV}} = \sum_{i=1}^{n_{\text{rad}}} \left(\lambda_v \left\| \frac{P_{\mathcal{R}\to\mathcal{C}}(D_i^{\text{rad}}, \mathbf{g}) - v_i^{\text{3D}}}{h_i^{\text{3D}}} \right\|_2^2 + \lambda_h \left\| \frac{P^h_{\mathcal{R}\to\mathcal{C}}(D_i^{\text{rad}}, \mathbf{g}) - h_i^{\text{3D}}}{h_i^{\text{3D}}} \right\|_2^2 \right), \tag{5.14}$$

where $P_{\mathcal{R}\to\mathcal{C}}$ is the projection defined in Section 5.1; $P^h_{\mathcal{R}\to\mathcal{C}}$ is to obtain the object 3D height after $P_{\mathcal{R}\to\mathcal{C}}$; h_i^{3D} is the height of the projected 3D box to RGB image detected by camera; λ_v and λ_h are two weights for v axis and height, respectively, where $\lambda_v + \lambda_h = 1$.

Alignment cost in BEV. On the other hand, we also consider calculating the alignment cost in BEV as

$$\ell_{\text{BEV}} = \sum_{j=1}^{n_{\text{3D}}} \left\| P_{\text{3D}\to\text{BEV}}(D_j^{\text{3D}}), D_j^{\text{rad}} \right\|_{\text{bbox}}, \tag{5.15}$$

where $P_{\text{3D}\to\text{BEV}}$ is the splat operator by ignoring y axis in the ego 3D coordinates; $\|B, p\|_{\text{bbox}}$ is the distance between a BEV box B and a BEV point p, shown in Figure 5.5 (c) as red arrows. This distance is defined as the nearest distance from the point to the bounding box boundary. If the point locates inside the bounding box, the distance is defined as 0.

Using these two detection alignment costs in both PV and BEV, we can align the detections between $\mathcal{R}$ and $\mathcal{C}$ robustly and efficiently.

Ground Plane Optimization After the detections are aligned between the camera and radar, the ground plane parameters can be optimized accordingly. Here, we define the ground plane parameters optimization as

$$\min_{\varphi,\gamma} \sum_{k=1}^{n_{\text{aligned}}} \left\| v_k^{\text{rad}} - v_k^{\text{3D}} \right\|^2 + \lambda_{\text{sm}}\ell_{\text{sm}} + \lambda_{\text{hr}}\ell_{\text{hr}}, \tag{5.16}$$

where n_{aligned} is the number of aligned detections. Besides, we add two ground plane losses, i.e., ground plane smoothing loss ℓ_{sm} and ground plane horizontal loss ℓ_{hr}. They are defined as follows,

$$\ell_{\text{sm}} = \left\| \mathbf{g}^t - \mathbf{g}^{t-1} \right\|_2^2, \tag{5.17}$$

$$\ell_{\text{hr}} = \gamma^2. \tag{5.18}$$

The smoothing loss is to make the ground plane smooth for continuous estimation, and the horizontal loss is to keep the estimated ground plane horizontal under the autonomous driving scenarios.

5.2.2 Radar-Camera Alignment Refinement

After the detections are aligned between camera PV coordinates and radar BEV coordinates, we further refine the 3D locations of the aligned 3D bounding boxes. The idea of the refinement is to achieve a geometrical consistency between the detections from the camera and radar sensors. Here, we do not directly use the 3D locations of radar detections to refine the 3D bounding boxes because radar detections usually distribute randomly around the objects due to the sensitivity, specularity, and poor angle resolution of radar signals. In contrast, we consider taking advantage of the proposed BCP to reproject the bottom vertices. This is an equivalent solution to ensure the *geometrical consistency* between the detections from the camera and radar sensors.

The procedure of alignment refinement, shown in Figure 5.6, can be described as follows:

- Project camera and radar detections to both PV and BEV with the optimal ground plane, according to Section 5.2.1.
- Select the four bottom vertices of each 3D bounding box.
- Project the four vertices to BEV using the optimal ground plane.
- Adjust object 3D locations (depths) to get refined 3D bounding boxes.

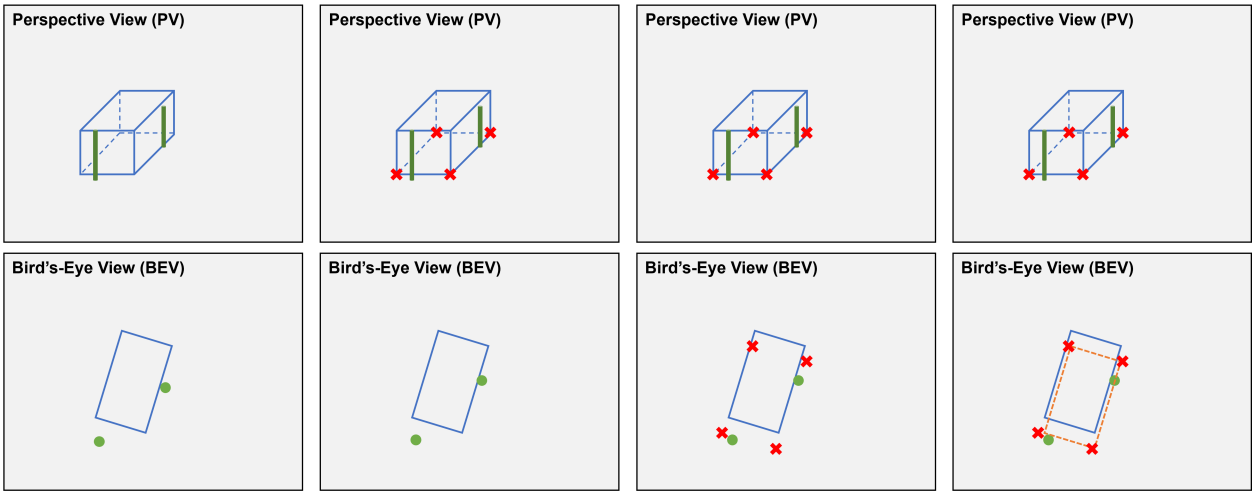


Figure 5.6: Radar-camera alignment refinement illustration. (a) Detections in both PV and BEV after detection alignment; (b) Select the four bottom vertices of the 3D bounding box; (c) Project the four vertices to BEV using BCP; (d) Adjust the location (depth) of the 3D bounding box by the projected bottom vertices.

This refinement procedure jointly considers the 3D information predicted by the camera and radar. The reason we consider utilizing the four bottom vertices of the 3D bounding box is our ground plane assumption that all the objects are located on the estimated ground plane. Thus, the four bottom vertices can be projected from PV to BEV using our proposed BCP.

Note that this reprojection procedure can be also described as two ways to project a 3D bounding box from ego 3D coordinates to BEV coordinates. The first way is to directly project the 3D bounding box to BEV by splatting its y axis. The second way is to project the 3D bounding box to PV and then project it again to BEV by BCP. This is also the reason there will be some inconsistency between the blue box and the red marks shown in Figure 5.6 (c).

5.2.3 Radar-Camera Alignment Verification

Then, we consider the detections that are not aligned successfully in Section 5.2.1, which means those objects are only detected by one of the sensors. We call this stage alignment verification.

In Section 5.2.2, we emphasize geometrical consistency between the two sensors. In this section, however, we emphasize temporal continuity instead, taking advantage of the tracking results obtained from the MOT algorithms for each sensor. The intuition behind this stage is that objects usually will not appear or disappear abruptly in autonomous driving applications. When we have multi-object tracking results from the earlier stages, we consider the tracking information as a temporal cue for more accurate and robust final object detection results.

Here, we split the detections into two categories, i.e., 1) camera detections that cannot find aligned radar detections; 2) radar detections that cannot find aligned camera detections.

- For a camera 3D bounding box that cannot be aligned with radar, we consider it as a true detection if 1) it can be aligned with a 3D box in previous frames by tracking; AND 2) it is located on the estimated ground plane; AND 3) it is within the FoV.

- For a radar detection that cannot be aligned with the camera, we consider it as true detection if it can be aligned with a detection in previous frames by tracking.

- Otherwise, we think this non-aligned detection is a *False Positive* and discard it from the final detection results.

5.3 Experiments

We conduct the experiments of RCCC on the CRUW2022 dataset introduced in Section 3.3. The camera-only method is DD3D [67] trained on our CRUW2022 training set.

The quantitative results are shown in Table 5.1. Our camera-only baseline achieves 71.5% on AP and 75.52% on AR for the overall testing set. Besides, we split the testing set into

normal driving scenarios and adverse driving scenarios. From Table 5.1, after RCCC is implemented, our 3D object detection performance is improved significantly, specifically for adverse driving scenarios by 9.4% improvement in AP and 12.1% improvement in AR.

Table 5.1: Experimental results for the proposed RCCC pipeline on the CRUW2022 dataset.

Method	Overall		Normal		Adverse	
	AP	AR	AP	AR	AP	AR
Camera-Only [67]	71.50	75.52	77.54	82.05	58.04	59.84
RCCC (Ours)	75.49	78.99	80.28	83.95	63.49	67.09
	(+5.6%)	(+4.6%)	(+3.5%)	(+2.3%)	(+9.4%)	(+12.1%)

Some qualitative results are shown in Figure 5.7. (a) and (b) Our RCCC removes a false positive detection and refines the 3D locations of the cars. (c) In this adverse scenario, there are some false negatives in the detection results from the camera-only model. Our RCCC can find those missing detections according to temporal continuity. (d) In this adverse scenario, the locations of the detected bounding boxes are not reliable due to the weak lighting condition. Our RCCC refines the locations of the bounding boxes by geometrical consistency between the camera and radar.

Moreover, we find that our RCCC specifically improves the 3D detection performance for pedestrians, even in normal driving scenarios. It is because pedestrians have smaller volumes than cars, so the radar's localization error is significantly smaller for pedestrians. Therefore, with an accurate localization performance, radar can help more with the cross-modality check for pedestrians.

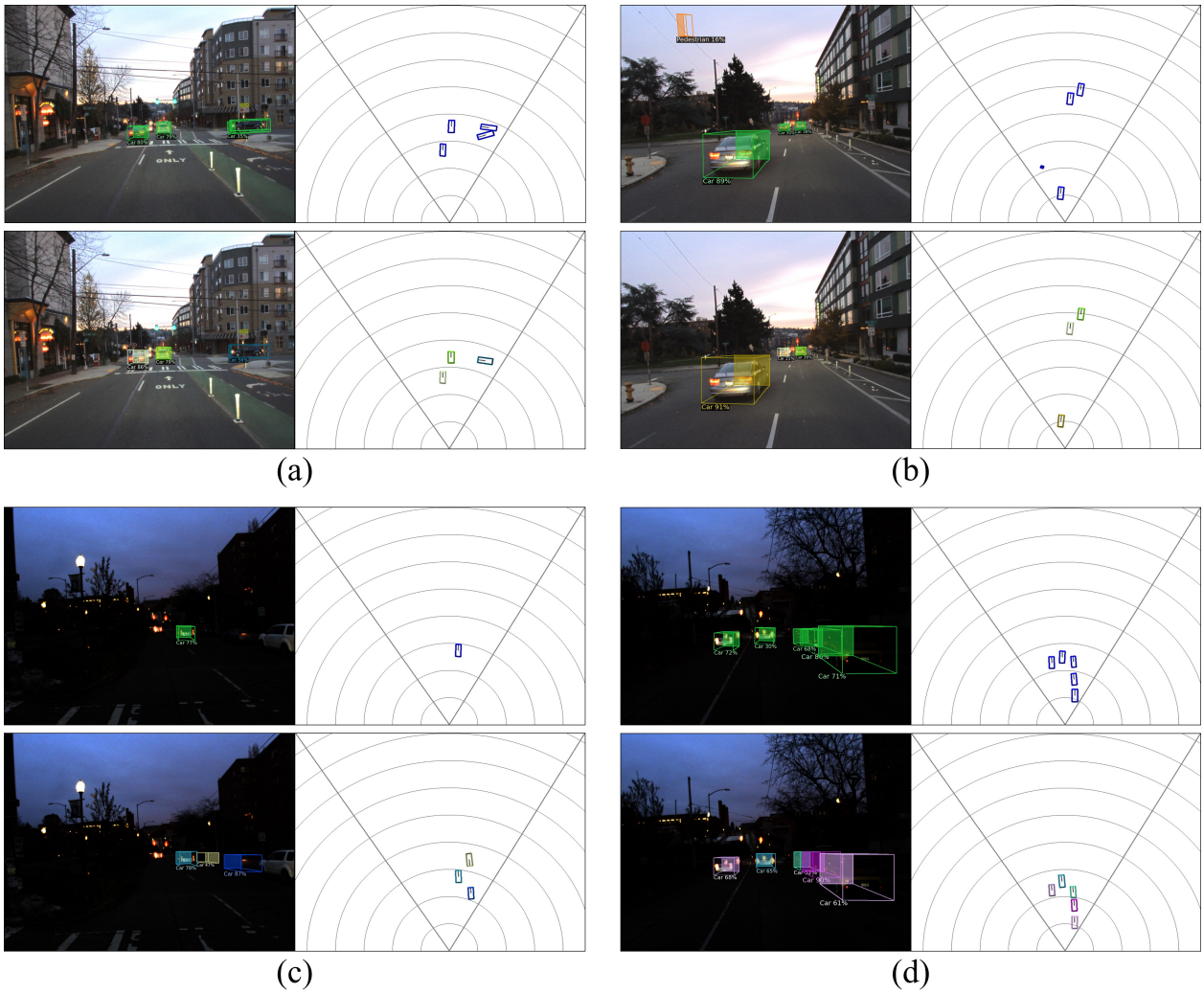


Figure 5.7: Some qualitative results for the proposed RCCC pipeline.

Chapter 6

CONCLUSION

In this dissertation, we proposed an accurate and robust "camera+radar" object 3D perception system for autonomous driving applications. The proposed system consists of two main parts, i.e., cross-modality supervision and cross-modality check. The cross-modality supervision aims to address radar-based object perception problems, including object detection and multi-object tracking, cross-supervised by the camera. On the other hand, the cross-modality check manages to obtain reliable 3D object perception results by jointly considering the detection results from camera and radar sensors. Finally, our proposed object 3D perception system can provide accurate and robust detection and tracking results using a monocular camera and an mmWave radar for normal and adverse driving scenarios.

BIBLIOGRAPHY

[1] Flir systems. `https://www.flir.com/`.

[2] Texas instruments. `http://www.ti.com/`.

[3] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[4] Tokihiko Akita and Seiichi Mita. Object tracking and classification using millimeter-wave radar based on lstm. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 1110–1115. IEEE, 2019.

[5] A. Angelov, A. Robertson, R. Murray-Smith, and F. Fioranelli. Practical classification of different moving targets using automotive radar and deep neural networks. *IET Radar, Sonar Navigation*, 12(10):1082–1089, 2018.

[6] Junaid Ahmed Ansari, Sarthak Sharma, Anshuman Majumdar, J Krishna Murthy, and K Madhava Krishna. The earth ain't flat: Monocular reconstruction of vehicles on steep and graded roads from a moving camera. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8404–8410. IEEE, 2018.

[7] Dan Barnes, Matthew Gadd, Paul Murcutt, Paul Newman, and Ingmar Posner. The oxford radar robotcar dataset: A radar extension to the oxford robotcar dataset. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Paris, 2020.

[8] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. *arXiv preprint arXiv:1903.05625*, 2019.

[9] 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.

[10] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[11] 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.

[12] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018.

[13] Peibei Cao, Weijie Xia, Ming Ye, Jutong Zhang, and Jianjiang Zhou. Radar-id: human identification based on radar micro-doppler signatures using deep convolutional neural networks. *IET Radar, Sonar & Navigation*, 12(7):729–734, 2018.

[14] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7291–7299, 2017.

[15] Samuele Capobianco, Luca Facheris, Fabrizio Cuccoli, and Simone Marinai. Vehicle classification based on convolutional networks applied to fmcw radar signals. In *Italian Conference for the Traffic Police*, pages 115–128. Springer, 2017.

[16] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[17] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE transactions on pattern analysis and machine intelligence*, 40(4):834–848, 2018.

[18] Marco Chiani, Andrea Giorgetti, and Enrico Paolini. Sensor radar for object tracking. *Proceedings of the IEEE*, 106(6):1022–1041, 2018.

[19] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[20] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[21] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. 2020.

[22] Ankit Dhall, Kunal Chelani, Vishnu Radhakrishnan, and K Madhava Krishna. Lidar-camera calibration using 3d-3d point correspondences. *arXiv preprint arXiv:1705.09785*, 2017.

[23] Xu Dong, Pengluo Wang, Pengyue Zhang, and Langechuan Liu. Probabilistic oriented object detection in automotive radar. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 102–103, 2020.

[24] Kaiwen Duan, Song Bai, Lingxi Xie, Honggang Qi, Qingming Huang, and Qi Tian. Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6569–6578, 2019.

[25] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. Detect to track and track to detect. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3038–3046, 2017.

[26] Di Feng, Christian Haase-Schütz, Lars Rosenbaum, Heinz Hertlein, Claudius Glaeser, Fabian Timm, Werner Wiesbeck, and Klaus Dietmayer. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[27] J. Ferryman and A. Ellis. Pets2010: Dataset and challenge. In *2010 7th IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 143–150, 2010.

[28] Xiangyu Gao, Guanbin Xing, Sumit Roy, and Hui Liu. Experiments with mmwave automotive radar test-bed. In *Asilomar Conference on Signals, Systems, and Computers*, 2019.

[29] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[30] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

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

[32] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[33] Clément Godard, Oisin Mac Aodha, and Gabriel J Brostow. Unsupervised monocular depth estimation with left-right consistency. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 270–279, 2017.

[34] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3d packing for self-supervised monocular depth estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[35] Stefan Haag, Bharanidhar Duraisamy, Felix Govaers, Wolfgang Koch, Martin Fritzsche, and Jürgen Dickmann. Extended object tracking assisted adaptive clustering for radar in autonomous driving applications. In *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–7. IEEE, 2019.

[36] 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.

[37] 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.

[38] S. Heuel and H. Rohling. Two-stage pedestrian classification in automotive radar systems. In *2011 12th International Radar Symposium (IRS)*, pages 477–484, Sep. 2011.

[39] Tsung-Wei Huang, Jenq-Neng Hwang, Suzanne Romain, and Farron Wallace. Fish tracking and segmentation from stereo videos on the wild sea surface for electronic monitoring of rail fishing. *IEEE Transactions on Circuits and Systems for Video Technology*, 2018.

[40] Xinyu Huang, Peng Wang, Xinjing Cheng, Dingfu Zhou, Qichuan Geng, and Ruigang Yang. The apolloscape open dataset for autonomous driving and its application. *IEEE transactions on pattern analysis and machine intelligence*, 42(10):2702–2719, 2019.

[41] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *arXiv preprint arXiv:1902.06162*, 2019.

[42] R. E. Kalman and R. S. Bucy. New Results in Linear Filtering and Prediction Theory. *Journal of Basic Engineering*, 83(1):95–108, 03 1961.

[43] Rudolph Emil Kalman. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82(1):35–45, 03 1960.

[44] Kai Kang, Hongsheng Li, Junjie Yan, Xingyu Zeng, Bin Yang, Tong Xiao, Cong Zhang, Zhe Wang, Ruohui Wang, Xiaogang Wang, et al. T-cnn: Tubelets with convolutional neural networks for object detection from videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 28(10):2896–2907, 2018.

[45] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.

[46] Hyungjin Kim, Bingbing Liu, and Hyun Myung. Road-feature extraction using point cloud and 3d lidar sensor for vehicle localization. In *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, pages 891–892. IEEE, 2017.

[47] Harold W Kuhn. The hungarian method for the assignment problem. *Naval Research Logistics (NRL)*, 52(1):7–21, 2005.

[48] Jihoon Kwon and Nojun Kwak. Human detection by neural networks using a low-cost short-range doppler radar sensor. In *2017 IEEE Radar Conference (RadarConf)*, pages 0755–0760. IEEE, 2017.

[49] Laura Leal-Taixé, Anton Milan, Ian Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. 2015.

[50] E Li, Shuaijun Wang, Chengyang Li, Dachuan Li, Xiangbin Wu, and Qi Hao. Sustech points: A portable 3d point cloud interactive annotation platform system. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1108–1115. IEEE, 2020.

[51] Yuan Li, Chang Huang, and Ram Nevatia. Learning to associate: Hybridboosted multi-target tracker for crowded scene. In *2009 IEEE conference on computer vision and pattern recognition*, pages 2953–2960. IEEE, 2009.

[52] 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.

[53] 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.

[54] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

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

[56] 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 International Conference on Computer Vision Workshops*, 2019.

[57] Michael Meyer and Georg Kuschk. Automotive radar dataset for deep learning based 3d object detection. In *2019 16th European Radar Conference (EuRAD)*, pages 129–132. IEEE, 2019.

[58] A Milan, L Leal-Taixé, I Reid, S Roth, and K Schindler. Mot16: A benchmark for multi-object tracking. 2016.

[59] Anton Milan, Konrad Schindler, and Stefan Roth. Multi-target tracking by discrete-continuous energy minimization. *IEEE transactions on pattern analysis and machine intelligence*, 38(10):2054–2068, 2016.

[60] 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.

[61] J Krishna Murthy, GV Sai Krishna, Falak Chhaya, and K Madhava Krishna. Reconstructing vehicles from a single image: Shape priors for road scene understanding. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 724–731. IEEE, 2017.

[62] J Krishna Murthy, Sarthak Sharma, and K Madhava Krishna. Shape priors for real-time monocular object localization in dynamic environments. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1768–1774. IEEE, 2017.

[63] 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.

[64] Alejandro Newell, Kaiyu Yang, and Jia Deng. Stacked hourglass networks for human pose estimation. In *European Conference on Computer Vision*, pages 483–499. Springer, 2016.

[65] Arthur Ouaknine, Alasdair Newson, Julien Rebut, Florence Tupin, and Patrick Pérez. Carrada dataset: Camera and automotive radar with range-angle-doppler annotations. *arXiv preprint arXiv:2005.01456*, 2020.

[66] Andras Palffy, Jiaao Dong, Julian FP Kooij, and Dariu M Gavrila. Cnn based road user detection using the 3d radar cube. *IEEE Robotics and Automation Letters*, 5(2):1263–1270, 2020.

[67] Dennis Park, Rares Ambrus, Vitor Guizilini, Jie Li, and Adrien Gaidon. Is pseudo-lidar needed for monocular 3d object detection? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3142–3152, 2021.

[68] Kanil Patel, Kilian Rambach, Tristan Visentin, Daniel Rusev, Michael Pfeiffer, and Bin Yang. Deep learning-based object classification on automotive radar spectra. In *2019 IEEE Radar Conference (RadarConf)*, pages 1–6. IEEE, 2019.

[69] Rodrigo Pérez, Falk Schubert, Ralph Rasshofer, and Erwin Biebl. Single-frame vulnerable road users classification with a 77 ghz fmcw radar sensor and a convolutional neural network. In *2018 19th International Radar Symposium (IRS)*, pages 1–10. IEEE, 2018.

[70] William H Press and Saul A Teukolsky. Savitzky-golay smoothing filters. *Computers in Physics*, 4(6):669–672, 1990.

[71] Yonggang Qi, Yi-Zhe Song, Honggang Zhang, and Jun Liu. Sketch-based image retrieval via siamese convolutional neural network. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 2460–2464. IEEE, 2016.

[72] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[73] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[74] 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.

[75] Mark A Richards. *Fundamentals of radar signal processing*. Tata McGraw-Hill Education, 2005.

[76] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*, pages 17–35. Springer, 2016.

[77] Ergys Ristani and Carlo Tomasi. Features for multi-target multi-camera tracking and re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6036–6046, 2018.

[78] Alexander Scheel and Klaus Dietmayer. Tracking multiple vehicles using a variational radar model. *IEEE Transactions on Intelligent Transportation Systems*, 20(10):3721–3736, 2018.

[79] Marcel Sheeny, Emanuele De Pellegrin, Saptarshi Mukherjee, Alireza Ahrabian, Sen Wang, and Andrew Wallace. Radiate: A radar dataset for automotive perception. *arXiv preprint arXiv:2010.09076*, 2020.

[80] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[81] Shiyu Song and Manmohan Chandraker. Robust scale estimation in real-time monocular sfm for autonomous driving. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1566–1573, 2014.

[82] Shiyu Song and Manmohan Chandraker. Joint sfm and detection cues for monocular 3d localization in road scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3734–3742, 2015.

[83] 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.

[84] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chenfeng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end

object detection with learnable proposals. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 14454–14463, 2021.

[85] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[86] Siyu Tang, Mykhaylo Andriluka, Bjoern Andres, and Bernt Schiele. Multiple people tracking by lifted multicut and person re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3539–3548, 2017.

[87] Zheng Tang and Jenq-Neng Hwang. Moana: An online learned adaptive appearance model for robust multiple object tracking in 3d. *IEEE Access*, 7:31934–31945, 2019.

[88] Zheng Tang, Gaoang Wang, Hao Xiao, Aotian Zheng, and Jenq-Neng Hwang. Single-camera and inter-camera vehicle tracking and 3d speed estimation based on fusion of visual and semantic features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 108–115, 2018.

[89] Subhashini Venugopalan, Marcus Rohrbach, Jeffrey Donahue, Raymond Mooney, Trevor Darrell, and Kate Saenko. Sequence to sequence-video to text. In *Proceedings of the IEEE international conference on computer vision*, pages 4534–4542, 2015.

[90] Thang Vu, Hyunjun Jang, Trung X Pham, and Chang Yoo. Cascade rpn: Delving into high-quality region proposal network with adaptive convolution. *Advances in neural information processing systems*, 32, 2019.

[91] Gaoang Wang, Yizhou Wang, Haotian Zhang, Renshu Gu, and Jenq-Neng Hwang. Exploit the connectivity: Multi-object tracking with trackletnet. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 482–490. ACM, 2019.

[92] 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.

[93] Yizhou Wang, Jenq-Neng Hwang, Gaoang Wang, Hui Liu, Kwang-Ju Kim, Hung-Min Hsu, Jiarui Cai, Haotian Zhang, Zhongyu Jiang, and Renshu Gu. Rod2021 challenge: A summary for radar object detection challenge for autonomous driving applications. In *Proceedings of the 2021 International Conference on Multimedia Retrieval*, pages 553–559, 2021.

[94] 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.

[95] 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.

[96] Yizhou Wang, Gaoang Wang, Hung-Min Hsu, Hui Liu, and Jenq-Neng Hwang. Rethinking of radar's role: A camera-radar dataset and systematic annotator via coordinate alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2815–2824, 2021.

[97] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*, pages 107–122. Springer, 2020.

[98] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[99] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020.

[100] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2403–2412, 2018.

[101] Ao Zhang, Farzan Erlik Nowruzi, and Robert Laganiere. Raddet: Range-azimuth-doppler based radar object detection for dynamic road users. In *2021 18th Conference on Robots and Vision (CRV)*, pages 95–102. IEEE, 2021.

[102] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, pages 1–19, 2021.

[103] Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11):1330–1334, 2000.

[104] Mingmin Zhao, Tianhong Li, Mohammad Abu Alsheikh, Yonglong Tian, Hang Zhao, Antonio Torralba, and Dina Katabi. Through-wall human pose estimation using radio signals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7356–7365, 2018.

[105] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer, 2020.

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

[107] Anqi Zhu, Lin Zhang, Ying Shen, Yong Ma, Shengjie Zhao, and Yicong Zhou. Zero-shot restoration of underexposed images via robust retinex decomposition. In *2020 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2020.

Appendix A

CRUW2022 DATASET DETAILS

A.1 Data Collection Pipeline

The pipelines for our CRUW2022 dataset collection software and the ROS-based time synchronizer are shown in Figure A.1.

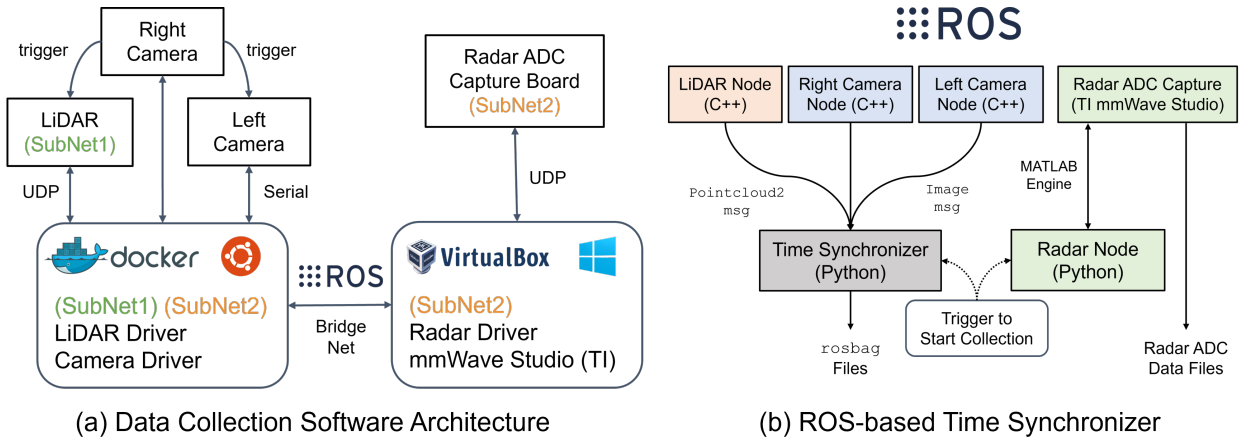


Figure A.1: Pipelines for our CRUW2022 dataset collection software and time synchronizer.

The main software is deployed on Ubuntu 20.04 operating system, which is responsible for communicating with two cameras and the LiDAR. Since TI's radar does not opensource their APIs, we create a Windows 10 virtual box on the Ubuntu system to communicate with the radar. The bridge between these two systems is built by the ROS.

As for the time synchronizer, we build the pipeline based on ROS. We use a software trigger to start time synchronizer and radar node at the same time. Then, the time synchronizer triggers the right camera, and the left camera and LiDAR are triggered through the synchronization cable (hardware trigger). On the other hand, radar node triggers radar

ADC capturing through the MALTAB engine by TI mmWave Studio.

A.2 Image Enhancement Implementation Details

As mentioned in Section 3.2, we implement the RRDNet for camera RGB image enhancement. Some qualitative enhancement results are shown in Figure A.2. Before the image enhancement, some images from the camera are relatively dark due to insufficient lighting. After the enhancement, images are brighter and easier for annotation and visualization.

Figure A.2: Qualitative results for the camera RGB image enhancement with RRDNet [107].

A.3 Additional Baseline Experimental Results

A.3.1 2D Object Detection Baselines

In addition to 3D object detection and tracking baselines, we also conduct some 2D object detection experiments on our dataset, as shown in Table A.1. We implement three recent 2D object detectors, i.e., Cascade RPN [90], YOLOX [29], and Sparse R-CNN [84], on our CRUW2022 dataset. The 2D bounding box annotations for this task are projected 3D bounding boxes from LiDAR coordinates to camera image coordinates.

The RGB images that we use for training are all enhanced images mentioned before. During training, we use eight NVIDIA GeForce RTX 3090 GPUs under Ubuntu 20.04. Our

Table A.1: Experiments for 2D object detection on the CRUW2022 dataset.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	AR	AR_S	AR_M	AR_L
Cascade RPN [90]	43.7	75.0	44.1	10.8	26.6	52.0	50.8	15.9	34.0	59.9
YOLOX [29]	32.6	53.8	35.0	15.1	22.5	39.8	43.3	26.4	35.5	48.4
Sparse R-CNN [84]	47.9	82.8	50.0	10.9	32.0	55.6	66.2	15.9	56.3	72.2

software is based on the MMDetection [16] toolbox. Specifically for each model:

- Cascade RPN [90]: The training batch size is 16 on 8 GPUs with 2 samples per GPU. We use the stochastic gradient descent (SGD) as the optimizer with a momentum of 0.9. The learning rate is set to 0.02 with the learning rate decay at epoch 8 and 11 by a factor of 10.

- YOLOX [29]: The training batch size is 64 on 8 GPUs with 8 samples per GPU. We use SGD as the optimizer with a momentum of 0.9. The learning rate is set to 0.01.

- Sparse R-CNN [84]: The training batch size is 16 on 8 GPUs with 2 samples per GPU. We use the Adam with weight decay (AdamW) as the optimizer. The learning rate is set to 2.5×10^{-5} with the learning rate decay at epoch 27 and 33 by a factor of 10.

Appendix B

RODNET IMPLEMENTATION AND EXPERIMENTS

B.1 RODNet Implementation Details

B.1.1 Network Architecture Implementation

In this section, we describe the network architecture of the RODNet. We introduce two different kinds of backbones, i.e., vanilla and hourglass (HG). The detailed parameters of the neural network are presented in Table B.1 and Table B.2, where `Conv3D` denotes 3D convolution layer; `TDC` denotes temporal deformable convolution layer; `ConvT3D` denotes transpose 3D convolution layer; and `Conv3D(Skip)` denotes 3D convolution layer for skip connection.

Table B.1: RODNet with the vanilla backbone.

Layer	Kernel	Stride	Channels
`Conv3D/TDC`	$(5, 3, 3)$	$(1, 1, 1)$	64
`Conv3D/TDC`	$(5, 3, 3)$	$(2, 2, 2)$	64
`Conv3D`	$(9, 5, 5)$	$(1, 1, 1)$	128
`Conv3D`	$(9, 5, 5)$	$(2, 2, 2)$	128
`Conv3D`	$(9, 5, 5)$	$(1, 1, 1)$	256
`Conv3D`	$(9, 5, 5)$	$(2, 2, 2)$	256
`ConvT3D`	$(4, 6, 6)$	$(2, 2, 2)$	128
`ConvT3D`	$(4, 6, 6)$	$(2, 2, 2)$	64
`ConvT3D`	$(3, 6, 6)$	$(1, 2, 2)$	3

The illustration of our proposed temporal inception convolution layer is shown in Fig. B.1. We utilize three different lengths for the temporal convolution kernels, i.e., 5, 9, 13, to extract the features with different temporal lengths. The features of the three different temporal lengths are then concatenated in the channel domain and sent to the subsequent layers. For simplicity, we use 160 as the number of both input and output channels of all temporal

Table B.2: RODNet with the HG backbone (single stack).

Layer	Kernel	Stride	Channels
`Conv3D/TDC`	$(5, 3, 3)$	$(1, 1, 1)$	32
`Conv3D/TDC`	$(5, 3, 3)$	$(1, 1, 1)$	64
`Conv3D`	$(9, 5, 5)$	$(1, 1, 1)$	64
`Conv3D`	$(9, 5, 5)$	$(2, 2, 2)$	64
`Conv3D`	$(9, 5, 5)$	$(1, 1, 1)$	128
`Conv3D`	$(9, 5, 5)$	$(2, 2, 2)$	128
`Conv3D`	$(9, 5, 5)$	$(1, 1, 1)$	256
`Conv3D`	$(9, 5, 5)$	$(2, 2, 2)$	256
`Conv3D(Skip)`	$(9, 5, 5)$	$(1, 1, 1)$	64
`Conv3D(Skip)`	$(9, 5, 5)$	$(2, 2, 2)$	64
`Conv3D(Skip)`	$(9, 5, 5)$	$(1, 1, 1)$	128
`Conv3D(Skip)`	$(9, 5, 5)$	$(2, 2, 2)$	128
`Conv3D(Skip)`	$(9, 5, 5)$	$(1, 1, 1)$	256
`Conv3D(Skip)`	$(9, 5, 5)$	$(2, 2, 2)$	256
`ConvT3D`	$(4, 6, 6)$	$(2, 2, 2)$	128
`ConvT3D`	$(4, 6, 6)$	$(2, 2, 2)$	64
`ConvT3D`	$(3, 6, 6)$	$(1, 2, 2)$	64
`Conv3D`	$(9, 5, 5)$	$(1, 1, 1)$	3

inception convolution layers.

Moreover, we train the RODNet on an NVIDIA Quadro GV100 GPU. It takes about 4 days to train the RODNet (vanilla), 8 days to train the RODNet (HG), and 10 days to train the RODNet (HG) with temporal inception convolution layers.

B.1.2 Implementation Details of Our Visual Teacher

There are 5 modules in the teacher's pipeline. The implementation details are stated as follows:

- Image-based object detection: pre-trained on the KITTI dataset and fine-tuned on the Cityscapes dataset.

- Depth estimation: a pre-trained self-supervised method that only needs stereo image

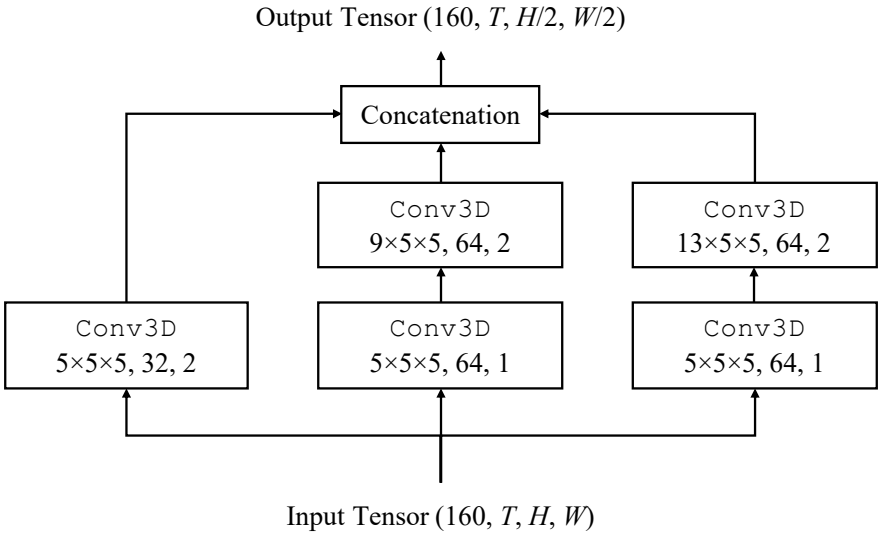


Figure B.1: Illustration of the proposed temporal inception convolution layer. The three parameters inside `Conv3D` blocks are representing kernel size, number of channels, and spatial stride, respectively.

pairs for training, and it is further retrained on our CRUW dataset.

- Semantic segmentation: pre-trained on the Cityscapes dataset.
- Multi-object tracking: pre-trained on the KITTI dataset and can be easily generalized to our CRUW dataset without much performance degradation.

B.2 Temporal Deformable Convolution Back-propagation

According to the temporal deformable convolution defined in Eq. 4 (in the original manuscript), the gradient w.r.t. the offset field $\Delta\mathbf{p}_n$ can be calculated as

$$\begin{aligned}\frac{\partial \mathbf{y}(\mathbf{p}_0)}{\partial \Delta\mathbf{p}_n} &= \sum_{\mathbf{p}_n \in \mathcal{R}} \mathbf{w}(\mathbf{p}_n) \cdot \frac{\partial \mathbf{x}(\mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n)}{\partial \Delta\mathbf{p}_n} \\ &= \sum_{\mathbf{p}_n \in \mathcal{R}} \left[\mathbf{w}(\mathbf{p}_n) \cdot \sum_{\mathbf{q}} \frac{\partial G(\mathbf{q}, \mathbf{p}_0 + \mathbf{p}_n + \Delta\mathbf{p}_n)}{\partial \Delta\mathbf{p}_n} \mathbf{x}(\mathbf{q}) \right].\end{aligned} \tag{B.1}$$

Here, $\mathbf{q}$ enumerates all integer locations in the 3D feature map $\mathbf{x}$, and $G(\cdot, \cdot)$ is the bilinear interpolation kernel on the 2D offset field defined as

$$G(\mathbf{q}, \mathbf{p}) = [g(q_u, p_u), g(q_v, p_v)], \tag{B.2}$$

where $g(a, b) = \max(0, 1 - |a - b|)$.

B.3 Additional RODNet Experimental Results

Some additional qualitative results are shown in Fig. B.2, where different colors represent different detected object classes (red: pedestrian; green: cyclist; blue: car). Based on the current hardware capability, the radar's FoV is 0-25m, $\pm 90°$. Our RODNet presents promising radar object detection performance in various driving scenarios, i.e., parking lot, campus road, city street, with different lighting conditions. More qualitative results can be found in our supplementary demo video.

Besides, we also show some failure cases of our RODNet in Fig. B.3. In Fig. B.3, (a) and (b) show some false negatives of pedestrian detection. Since pedestrians usually have much smaller volume than cars, so that cannot be easily detected separately when they are next to each other. Fig. B.3 (c) and (d) are two examples of false negatives of car detections, where (c) shows a car with back trunk opened is missing in our RODNet detection because of abnormal surface features, and (d) shows that the reflection signals from the faraway cars are suppressed by a nearby car in the static scenario. Finally, Fig. B.3 (e) and (f) show some false positives caused by traffic signs.

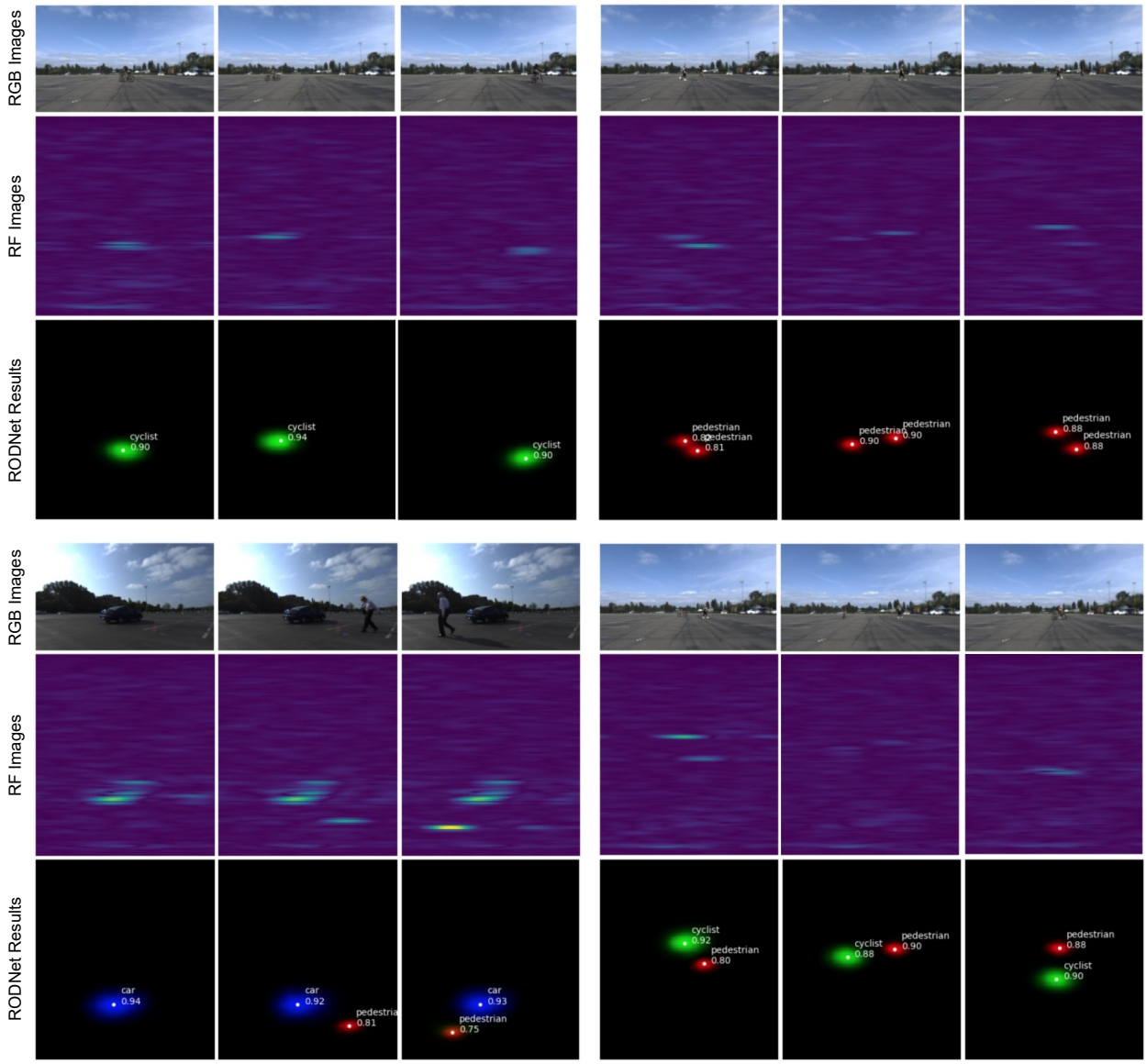


Figure B.2: The qualitative results of the RODNet in different driving scenarios (12 sequences in total). The three rows are RGB images, RF images, and resulting ConfMaps with final detections, respectively. The white dots on the ConfMaps represent the final detections after post-processing.

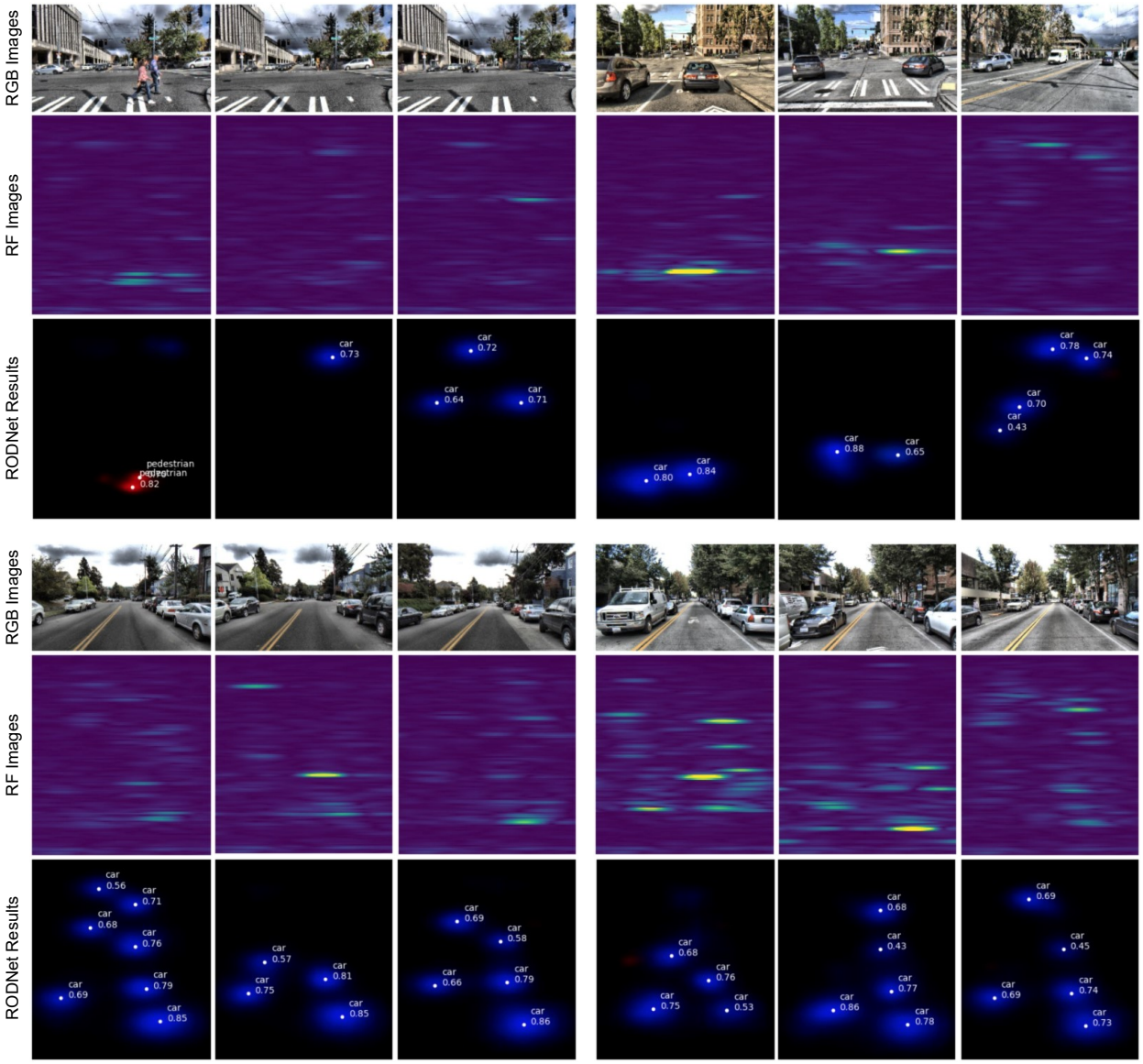


Figure B.2: The qualitative results of the RODNet in different driving scenarios (12 sequences in total). The three rows are RGB images, RF images, and resulting ConfMaps with final detections, respectively. The white dots on the ConfMaps represent the final detections after post-processing. (Continued)

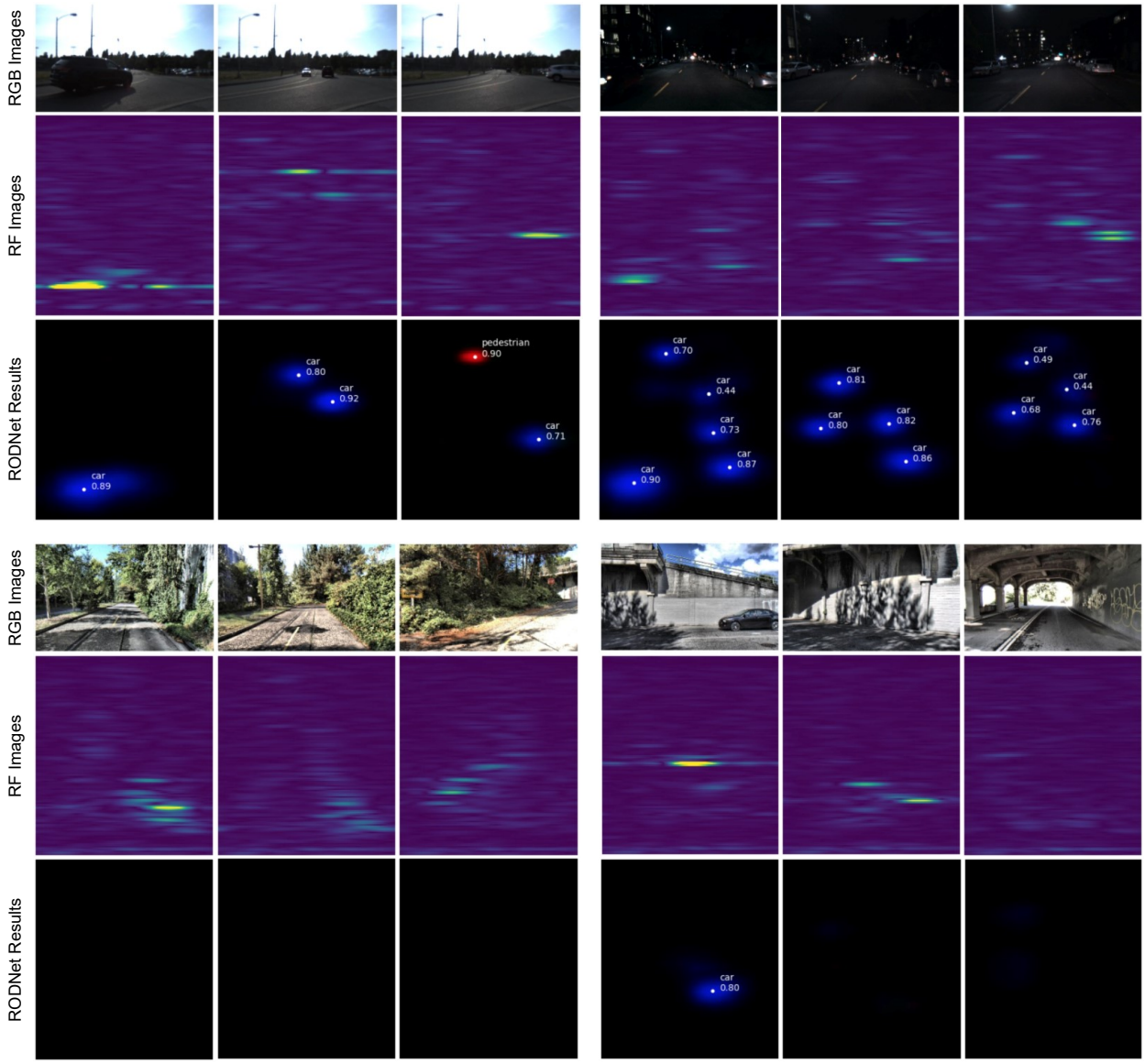


Figure B.2: The qualitative results of the RODNet in different driving scenarios (12 sequences in total). The three rows are RGB images, RF images, and resulting ConfMaps with final detections, respectively. The white dots on the ConfMaps represent the final detections after post-processing. (Continued)

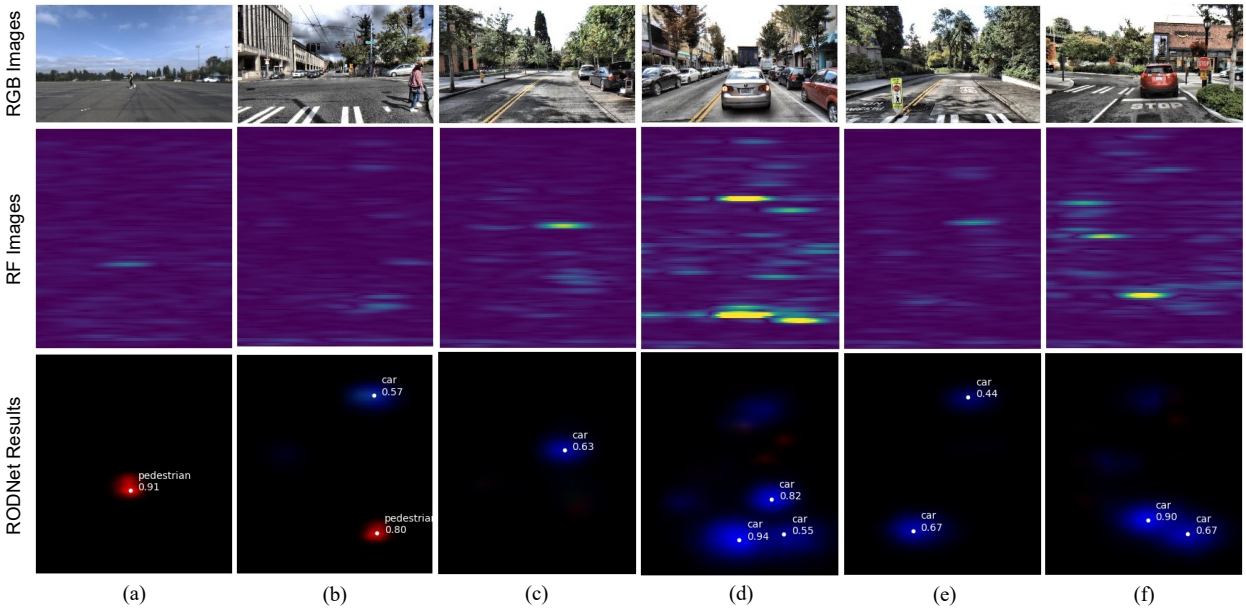


Figure B.3: The failure cases of our RODNet. (a) & (b): False negatives when multiple pedestrians are very near (two pedestrians in both scenes); (c): False negative for a car whose trunk is open (abnormal surface feature for a car); (d): False negatives due to the strong reflection of the nearby object in the static scenario; (e) & (f): False positives from traffic signs.