PERCEPTION VIA RADAR-CAMERA FUSION FOR AUTONOMOUS DRIVING

By

Yunfei Long

A DISSERTATION

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

Electrical Engineering—Doctor of Philosophy

2025

ABSTRACT

Reliable sensing of environments is a key bottleneck in autonomous driving. Among a variety of sensors, automotive radar stands out for its low cost, robustness to adverse weather, and ability to capture motion. Nevertheless, radar is not widely recognized in the computer vision community, facing challenges in sparse, low-dimensional, and inaccurate measurements. This work sheds a different light on the traditional view of radar on perception, exploring how radar can be used to enhance monocular perception in multiple vision tasks including depth completion, velocity estimation, and 3D object detection. Depth completion with radar-camera fusion aims to predict dense depths for image pixels given sparse radar points and images. To handle ambiguous geometric associations between raw radar pixels and image pixels, we propose radar-camera pixel depth association (RC-PDA), which maps radar pixels to nearby image pixels with the same depths. We train a model to predict RC-PDA, which is used to enhance and densify radar returns for depth completion. Full velocity estimation for radar points focuses on predicting the tangential velocity, which is absent in radar measurements. We present a closed-form solution to compute point-wise full velocity for radar returns by combining radar Doppler velocity with corresponding optical flows on images. 3D object detection aims to estimate object categories and 3D bounding boxes. We focus on using radar to improve monocular detections in position estimation. To address discrepancy between radar hits and object centers, we build a model to predict point-wise 3D object centers, which are subsequently matched with monocular estimated centers for depth fusion. To deal with the complexity of possible locations of radar points reflected by targets, we build a model to estimate radar hit distributions conditioned on object properties predicted by a monocular detector, and spatially match the distributions with actual radar hits in the neighborhood of monocular detections. This method reveals radar distributions under different conditions and achieves interpretable position estimation via radar-camera fusion. Experiments show that the proposed methods achieve state-of-the-art performance on the individual vision tasks via radar-camera fusion. We believe this work will contribute new practical solutions for perception with radar for autonomous driving.

ACKNOWLEDGEMENTS

First, I am grateful to my advisor Daniel Morris for assiduously guiding me through various interesting projects in computer vision. Second, I would like to acknowledge Ford-MSU Alliance for sponsoring my research on radar-camera fusion and thank all collaborators for fruitful discussions. Finally, I wish to express my gratitude to my parents, who always support me in pursuing advanced degrees.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

1.1 Autonomous Driving

Autonomous driving exemplifies the latest engineering efforts in artificial intelligence. The evolution of human civilization is characterized by consistent technical efforts extending intrinsic human capabilities: our hands are extended by mechanics, and our intelligence by computers and algorithms. Autonomous driving [152] is the latest example of these attempts regarding intelligence: autonomous cars sense their environments and drive themselves based on the sensed information and maps.

Autonomous cars have long been vigorously pursued by engineers and the general public for a number of obvious benefits. First, it frees people from the wheel. Typically, people spend over 300 hours per year behind the wheel [85]. Autonomous driving offers the freedom to invest the time elsewhere. Thus, we argue that this technology is revolutionary for modern people in the same sense that standing upright frees the hands of ancient apes [153]. Second, autonomous driving makes it easier for those who cannot drive, e.g., people without a driver's license, to move more freely. Third, we believe that, compared with a human driver, self-driving cars, equipped with optimized algorithms and a perception ability better than human eyes, can drive safer and use energy more efficiently.

Typically, the procedures of autonomous driving consist of perception, path planning, and maneuver control [152], where technically perception has long been the bottleneck of practical deployment. It is challenging to grasp a complete and accurate picture of complex environments, such as typical traffic scenes. Historically, only passenger airplanes and drones have successfully achieved autopilot [130] in our daily lives, as the environment is much simpler to perceive in the sky than on the road. A new wave of enthusiasm for autonomous driving is fueled by the progress of supervised learning with deep neural networks [50] and big data [162]. The progress pushes the performance of perception to a level where it outperforms human in individual tasks. Industrial companies see the opportunities, invest heavily in research and development, and put on the agenda the day when self-driving cars hit the road. Nevertheless, it is admitted that current perception performance is still insufficient for autonomous driving to take over [152]. Thus, some companies focus on specialized autonomous vehicles working in simpler environments, e.g., vehicles delivering packages; self-driving trucks [1], which are mostly on freeways; and advanced driver assistance systems (ADAS), where a human driver is still a necessity.

In a word, we are optimistic with reservations about the future of autonomous driving: we see its great potentials but when it comes to achieving reliability in complex environments, there is still a long way to go. Autonomous vehicles are complex systems requiring interdisciplinary efforts from an engineering point of view, not to mention the dilemmas they face regarding law and ethics [4]. Technically, perception is a key factor that will decide the fate of autonomous driving. In this dissertation, we focus on solving the perception problems, which will be introduced in the following section.

1.2 Perception Tasks Related to Autonomous Driving

To make decisions, self-driving cars need to obtain a number of attributes of objects appearing in the neighboring environments. We ask where the objects are, what kind of objects they are, what their sizes and orientations are, and how they are moving. In other words, those attributes include class, positions (e.g., depth), sizes plus orientations (modeled by 2D/3D bounding boxes), and dynamic status (e.g., velocity). Object detection [157] is a comprehensive task extracting the attributes for objects captured by sensors.

Although these attributes belong to objects, they can be defined and extracted at multiple granularities or levels according to sensors. Typically there are three levels from small to large, i.e., pixel-level [136], object-level, and image-level [110]. The pixel-level attributes are defined for objects intersecting with the ray of a pixel/point for camera/depth sensors. The object-level attributes are naturally defined. And image-level attributes are for objects in the image. Estimating attributes at low levels is helpful for extracting the same attributes at higher levels but is not always necessary. For example, image segmentation [135] predicts classes at pixel level and may contribute to object classification in detection (object-level task), but the object classification does not always require pixel-wise object segmentation.

In this dissertation, we estimate attributes including depth and velocity at the pixel level, which may lay the groundwork for better detection at the object level. All the vision tasks depend on information collected by perception sensors, which are introduced in the following section.

1.3 Perception Sensors for Autonomous Driving

Perception sensors [132] are bridges linking the real physical world with a digital one where perception algorithms are running. Characteristics of sensors, e.g., measured physical properties and precision, underlie sensor selection and algorithm design for autonomous vehicles. Camera, radar, and lidar are the most widely-used sensors for autonomous vehicles. Lidar and radar are depth sensors, which measure object depth directly, while camera is not in the sense that it only captures colors and intensities. Depth still can be acquired from a stereo camera through triangulation [32], but accuracy is low at long range. In summary, camera captures high-resolution textures with poor or no depth, while depth sensors measure depth with poor texture. Thus, the combination of cameras and depth sensors is a natural fusion strategy to enhance monocular perception.

On the other hand, the choice of depth sensors (one or both) is not so obvious, typically a result of a trade-off of a variety of factors such as cost and compactness. Both radar [159] and lidar [59, 74] are time-of-flight sensors, which infer distance according to the time electromagnetic waves travel to and back from objects, but work in different spectrums, lidar in infrared and radar in microwave. Lower frequencies allow radar to measure longer ranges and are more robust to adverse weather. However, lidar measurements have higher resolution both in azimuth and elevation, and typically achieve better positional estimates. With regard to cost and compactness, radar is the winner: radar is more compact in size and more flexible in where to install, while lidar, more bulky in size, scans 360 degrees and is typically installed on top of vehicles. Automative radar has been widely deployed in driving assistance systems, while the usage of lidar is rare for consumer cars on the market. Different from lidar, radar can measure dynamic status of objects with Doppler velocity.

FMCW (frequency modulated continuous wave) radar [124] is widely used as automotive radar for its low cost, which typically includes a transmitter and receivers. The transmitter emits chirp signals, which are reflected by targets and received by receivers. A mixer combines transmitted and received signals, obtains their frequency difference (i.e., beat frequency) and phase difference, and produce intermediate signals. For further processing, the intermediate signals are arranged as 2D maps, with one dimension representing signals in a chirp period and the second representing different chirps. As the frequency difference is proportional to time of flight during a single chirp period, a Fast Fourier Transform (FFT) is performed on intermediate signals in a single chirp period to extract a beat frequency, from which a range can be computed. When a target is moving, the radial motion results in small changes in the object range and leads to phase shifts between neighboring received chirps. Doppler velocity can be inferred from the phase shifts over chirps. Therefore, a second FFT is additionally applied along the chirp index dimension to generate a range-Doppler map. To measure azimuths of targets, an array of horizontally aligned receivers receive echoed chirps from the same targets. There is a small difference in target ranges observed by neighboring receivers, and the range difference is a known function of the target azimuth. Similarly, the range differences lead to phase shifts in chirps received by neighboring receivers. Thus, a final FFT is applied to frames of range-Doppler maps along the receiver index dimension to compute a range-Doppler-azimuth map.

In this dissertation, we select radar to fuse with camera in order to enhance monocular perception as a cost-friendly solution. There are other fusion options that are also promising and worth more investigation as future work, e.g., radar-lidar and radar-lidar-camera combinations, as radar is complementary to lidar in scenarios of long ranges, adverse weather, and velocity estimation. A variety of sensor data are publicly available in autonomous driving datasets, which will be introduced in the following section.

1.4 Perception Dataset for Autonomous Driving

The emergence of large-scale datasets for perception tasks for autonomous driving fuels the research on perception algorithms, with an abundance of sensor data and ground truth (GT) for supervised machine learning and evaluation. The sensor data are collected by driving a vehicle with perception sensors in real traffic, and the GT data are typically annotated manually with a significant amount of time and labor. In addition, a dataset usually provides benchmark, a criterion to evaluate predictions with GT data. This evaluation makes it convenient to fairly compare the performance of different algorithms and guide the improvement of methods to further their performance. The release of datasets frees researchers from the tedious work of collecting and labeling data, allowing them to focus on the design of perception algorithms. Additionally, a dataset may not cover all scenarios of driving, and more data and testing are required for real-world deployment. Nevertheless, it is valuable data for the first step in algorithm design and gives us a glimpse of autonomous driving in the real world.

Among a few mainstream large-scale datasets, e.g., KITTI [27] and nuScenes [10], nuScenes stands out with radar data collection and GT object velocity. Using multiple commercial radars and a lidar, nuScenes allows the ego vehicle to perform a 360-degree scan of the surrounding environment. The dataset includes measurements collected in city traffic under a variety of weather and lighting conditions. As a 3D object detection dataset, nuScenes provides GT bounding boxes (i.e., center location, size, yaw, velocity, and class) of objects in ten classes related to city traffic, e.g., cars, motorcycles, pedestrians, and traffic cones.

There are two representative formats of radar data, i.e., the azimuth-range-Doppler format and point clouds. The azimuth-range-Doppler [95] is a raw format directly from the Fourier transform of received signals, while point clouds are obtained with additional processing such as clustering. We focus on dealing with the point cloud format as it is the format of radar data provided by nuScenes dataset [10]. Besides datasets, another engine pushing forward autonomous driving is neural networks, which will be introduced in the following section.

1.5 Neural Networks as Mathmatical Model in Perception Tasks

Big data and big computational power are two major factors driving the successful application of deep neural networks in different fields, including vision, audio [101] and natural language processing [94]. It is the flexibility of neural networks that makes them outperform and finally replace traditional hand-crafted features in perception tasks: from large training data, neural networks automatically learn optimized features, largely superior to hand-crafted features in quality and quantity.

Convolutional neural networks (ConvNet) [60] trained with supervised learning are among the most widely used and successful models for visual tasks. ConvNet can model a complex nonlinear function with tens of millions of parameters, while it only consists of a cascade of basic operations, i.e., convolutions and activation functions. Although ConvNet, as a black box model, achieves great performance in practice, the understanding of what is happening in the black box is still far from satisfactory [49]. Nevertheless, some properties of the ConvNet are clear and may guide us in building a proper model. First, ConvNet processes grid data formats and thus is a good choice for image or quantized data in bird's-eye view (BEV). ConvNet keeps the correspondences between inputs and outputs at the same locations. This is because the output at a pixel is a function of inputs at the same pixel and surrounding ones, i.e., receptive field. Correspondences can be obtained via upsampling or downsampling if the output resolution is different from the input. Third, convolution is location invariant. In other words, the processing is the same at every location of the input.

Transformer has been an emerging network architecture in vision tasks [67]. Featuring self-attention modules, it enables long-distance interactions while losing translation equivariance and locality existing in ConvNet. As a result, it requires large training dataset to achieve good performance: typically it is trained with large external dataset before being fine-tuned with smaller dataset specific to tasks.

The key to designing a good ConvNet model is building good loss functions that converge to small values via gradient descent. The geometric shape of a loss is typically not convex, and thus convergence to global minima is not guaranteed [156]. Sometimes local minima are satisfactory for some applications. The geometry of a loss is primarily determined by three factors, i.e., network structure, definition of predictions/labels as well as loss criteria. Other factors influencing convergence are initial parameters and learning steps. Thus, the focus of model design is on adjusting the three components through trial and error so that the geometry of the resultant loss easily converges to small values. For example, L2 loss and cross-entropy loss are two loss criteria:

L2 loss is more suitable for regression, while cross-entropy works better for classification. We typically use existing successful network structures and only need to control the model size to achieve a balance between efficiency and capacity, since a good model is flexible and can adapt to new tasks well with new training data. On the other hand, deciding what to predict is another important dimension of freedom in design that determines the geometry of loss. For example, there are typically two output options for the task of depth estimation: one is directly predicting depths, and the other is indirectly predicting the coefficients of a number of pre-defined depths [34]. It turns out that the second option leads to a better loss, which prevents the smearing of different depths among neighboring pixels.

1.6 Dissertation Contributions

The contributions of this dissertation are listed as follows:

- We propose radar-camera pixel depth association that upgrades the projection of radar onto images and prepares a densified depth layer. Using the enhanced radar depth improves radar-camera depth completion.

- We identify the problem of estimating point-wise full velocity of radar returns by fusing radar and camera. We propose a novel closed-form solution to infer full radar-return velocity by leveraging the radial velocity of radar points, optical flow of images, and the learned association between radar points and image pixels.

- We enhance radar returns to obtain 3D object center detections from each radar return. We achieve camera-radar association at the detection level using the enhanced radar locations. We improve monocular object depth estimates by fusing enhanced radar depths

- We build a model to predict radar hit distributions relative to reflecting objects in BEV. We propose position estimation by convolving predicted radar distributions with actual radar measurements.

1.7 Dissertation Organization

In light of the advantages of radar over lidar (as mentioned in Section 1.3), e.g., low cost, compactness, and capability of Doppler measurement, in this work, we use radar to enhance camera in vision tasks including estimation of depth and velocity.

In Chapter 2, we fuse radar and camera for depth estimation or depth completion. Our method improves monocular depth with radar as a low-cost depth sensor. The difficulty of fusing radar and camera for depth completion lies in inaccurate projection of radar points on images and thus misaligned radar and image features. To solve this problem, we estimate radar-camera pixel association to generate augmented radar depth features, which are compatible with image features.

In Chapter 3, we estimate the full velocity of radar returns with the help of video. This is significant as estimation of full velocity enables us to capture actual dynamic status of objects. It is known that radar only measures radial velocity, which by itself does not provide complete information to infer full velocity. We derive a closed-form solution to full velocity by using Doppler measurements and corresponding optical flows. A model is built to estimate the radar-flow association. Our method solves the ambiguities from radial to full velocity and paves the way for more accurate perception of object motion and accumulation of moving radar points over time.

In Chapters 4 and 5, we use radar-camera fusion to improve 3D object detection over monocular approaches, particularly for better position estimation. The key is to infer object centers from sparse radar hits on object surfaces. In Chapters 4, we explicitly predicts pixel and depth offsets from radar hits to their corresponding object centers in image space. In Chapter 5, we study radar hit distributions relative to object centers in BEV and use them to match actual radar measurements for more accurate center prediction. We list conclusions and future work in Chapter 6.

CHAPTER 2

RADAR-CAMERA PIXEL DEPTH ASSOCIATION FOR DEPTH COMPLETION

While radar and video data can be readily fused at the detection level, fusing them at the pixel level is potentially more beneficial. This is also more challenging in part due to the sparsity of radar, but also because automotive radar beams are much wider than a typical pixel combined with a large baseline between camera and radar, which results in poor association between radar pixels and color pixel. A consequence is that depth completion methods designed for lidar and video fare poorly for radar and video. Here we propose a radar-to-pixel association stage which learns a mapping from radar returns to pixels. This mapping also serves to densify radar returns. Using this as a first stage, followed by a more traditional depth completion method, we are able to achieve image-guided depth completion with radar and video. We demonstrate performance superior to camera and radar alone on the nuScenes dataset. This chapter was previously published as [76].

2.1 Introduction

We seek to incorporate automotive radar as a contributing sensor to 3D scene estimation. While recent work fuses radar with video for the objective of achieving improved object detection [12, 62, 87, 90, 93], here we aim for pixel-level fusion of depth estimates, and ask if *fusing video with radar can lead to improved dense depth estimation of a scene*.

Up to the present, outdoor depth estimation has been dominated by lidar, stereo, and monocular techniques. The fusion of lidar and video has led to increasingly accurate dense depth completion [34]. At the same time, radar has been relegated to the task of object detection in vehicle's ADAS [84]. However, phased array automotive radar technologies have been advancing in accuracy and discrimination [30]. Here we investigate the suitability of using radar instead of lidar for the task of dense depth estimation. Unlike lidar, automotive radars are already ubiquitous, being integrated in most vehicles for collision warning and similar tasks. If successfully fused with video, radar could provide an inexpensive alternative to lidar for 3D scene modeling and perception. However, to achieve this, attentive algorithm design is required in order to overcome some of the limitations of radar, including coarser, lower resolution, and sparser depth measurements than typical lidars.

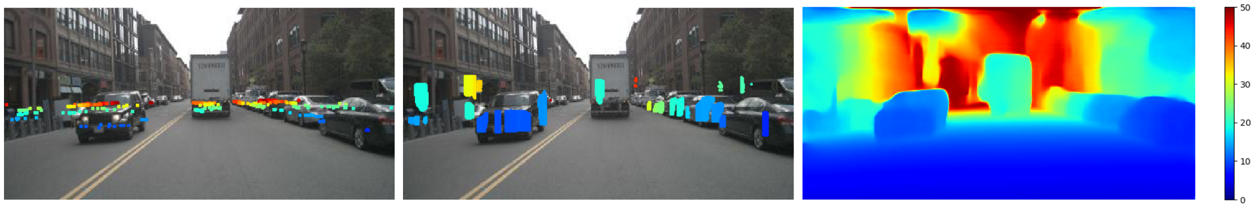


Figure 2.1 Radar-camera depth completion: (a) an image with 0.3 seconds (5 sweeps) of radar hits projected onto it, (b) enhanced radar depths at confidence level 0.9 eliminate occluded pixels and expand visible hits, and (c) final predicted depth through depth completion.

This chapter proposes a method to fuse radar returns with image data and achieve depth completion; namely a dense depth map over pixels in a camera. We develop a two-stage algorithm. The first stage builds an association between radar returns and image pixels, during which we resolve some of the uncertainty in projecting radar returns into a camera. In addition, this stage is able to filter occluded radar returns and "densify" the projected radar depth map along with a confidence measure for these associations (see Fig. 2.1 (a,b)). Once a faithful association between radar hits and camera pixels is achieved, the second stage uses a more standard depth completion approach to combine radar and image data and estimate a dense depth map, as in Fig. 2.1(c).

A practical challenge to our fusion goal is the lack of public datasets with radar. KITTI [28], the dataset used most extensively for lidar depth completion, does not include radar and nor do the Waymo [126] or ArgoVerse [13] datasets. The main exception is nuScenes [10] and the small Astyx [86] which have radar, but unfortunately do not include a dense, pixel-aligned depth map as created by Uhrig et al. [131]. Similarly, the Oxford Radar Robot Car dataset [3] includes camera, lidar, and raw radar data, but no annotations are available for scene understanding. As a result, all experiments of this work will use the nuScenes dataset along with its annotations. However, we find single lidar scans insufficient to train depth completion, and so accumulate scans to build semi-dense depth maps for training and evaluating depth completion.

The main contributions of this work include:

- Radar-camera pixel depth association that upgrades the projection of radar onto images and prepares a densified depth layer.

- Enhanced radar depth that improves radar-camera depth completion over raw radar depth.
- Lidar ground truth accumulation that leverages optical flow for occluded pixel elimination, leading to higher quality dense depth images.

2.2 Related Work

Radar for ADAS. Frequency Modulated Continuous Wave radars are inexpensive and all-weather, and have served as the key sensor for modern ADAS. Ongoing advances are improving radar resolution and target discrimination [30], while convolutional networks have been used to add discriminative power to radar data, moving beyond target detection and tracking to include classifying road environments [52, 119], and seeing beyond-line-of-sight targets [113]. Nevertheless, the low spatial resolution of radar means that the 3D environment, including object shape and classification, are only coarsely obtained. A key path to upgrading the capabilities of radar is through integration with additional sensor modalities [84].

Radar-camera fusion. Early fusion of video with radar, such as [38], relied on radar for cueing image regions for object detection or road boundary estimation [36], or used optical flow to improve radar tracks [26]. With the advent of deep learning, much more extensive multi-modal fusion has become possible [23]. However, to the best of our knowledge, no prior work has conducted pixel-level dense depth fusion between radar and video.

Radar-camera object detection. Object detection is a key task in 3D perception [8]. There has been significant recent interest in combining radar with video for improved object detection. In [12], ResNet blocks [31] are used to combine both color images and image-projected radar returns to improve longer-range vehicle detection. In [62], an FFT applied to raw radar data generates a polar detection array which is merged with a bird's-eye projection of the camera image, and targets are estimated with a single shot detector [69]. In [87], features from both images and a bird's-eye representation of radar enter a region proposal network that outputs bounding boxes [122]. An alternative model for radar hits is a 3m vertical line on the ground plane which is projected into the image plane by [93], and combined with VGG blocks to classify vehicle detections at multiple scales. Our work differs fundamentally from these methods in that our goal is dense depth estimation, rather than object classification. But we do share similarity in radar representation: we project radar hits into an image plane. However, the key novelty in our work is that we learn a neighborhood pixel association model for radar hits, rather than relying on projected circles [12] or lines [93].

Lidar-camera depth completion. Our task of depth estimation has the same goal as lidar-camera depth completion [33, 34, 37, 103, 147]. However, radar is far sparser than lidar and has lower accuracy, which makes these methods unsuitable for this task. Our radar enhancement stage densifies the projected radar depths, followed by a more traditional depth completion architecture.

Monocular depth estimation. Monocular depth inference may be supervised by lidar or self-supervised. Self-supervised methods learn depth by minimizing photometric error between images captured by cameras with known relative positions. Additional constrains such as semantics segmentation [104, 163], optical flow [35, 133], surface normal [150], and proxy disparity labels [142] improve performance. Recently, self-supervised PackNet [29] has achieved competitive results. Supervised methods [19, 25] include continuous depth regression and discrete depth classification. BTS [51] achieves the state of the art (SOTA) by improving upsampling via additional plane constraints, and more recently [107] combines supervised and self-supervised methods. Our goal is not monocular depth estimation, but rather to improve what is achievable from monocular depth estimation through fusion with radar.

2.3 Method

While there are a variety of data-spaces in which radar can be fused with video, the most natural, given our objective of estimating a high resolution depth map, is in the image space. But this immediately presents a problem: to which pixel in an image does a *radar pixel* belong? By radar pixel we mean a simple point projection of the estimated 3D radar hit into the camera. The nuScenes dataset [10] provides extrinsic and intrinsic calibration parameters needed to map the radar point clouds from the radar coordinates system to the egocentric and camera coordinate systems.

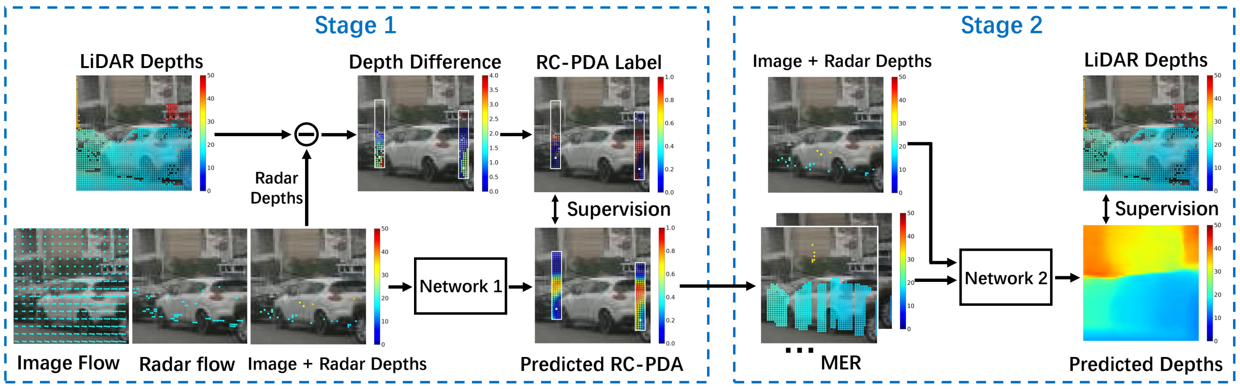


Figure 2.2 Our two-stage architecture. Network 1 learns N-channel *radar-camera pixel depth association* (RC-PDA), here illustrated for two radar pixels (marked with white squares) on their neighboring pixels (white boxes). The RC-PDA is converted into a *multi-channel enhanced radar* (MER), and input to Network 2 which performs image-guided depth completion.

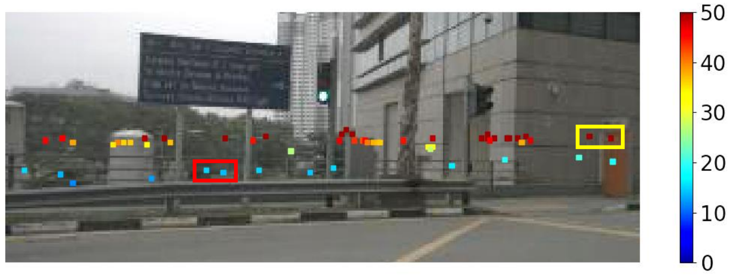


Figure 2.3 Examples of radar hits projected into a camera. While the hits project into the vicinity of the target that they hit, their image position can be quite different from their actual location. For example, radar depths in the yellow/red box are larger/smaller than corresponding image depths (meters).

Assuming that the actual depth of the image pixel is the same as the radar pixel depth turns out to be fairly inaccurate. We describe some of the problems with this model, and then propose a new pixel association model. We present a method for building this new model and show its benefit by incorporating it into radar-camera depth completion. Fig. 2.2 shows the diagram of the proposed method.

2.3.1 Radar Hit Projection Model

Single-row scanning automotive radars can be modeled as measuring points in a plane extending usually horizontally (relative to the vehicle platform) in front of the vehicle, as in [12]. While radars can measure accurate depth, often the depth they give when projected into a camera is incorrect, as can be seen in the examples in Fig. 2.3. An important source of this error is the large width of radar beams which means that the hits extend well beyond the assumed horizontal plane. In other words, the height of measured radar hits is inaccurate [91].

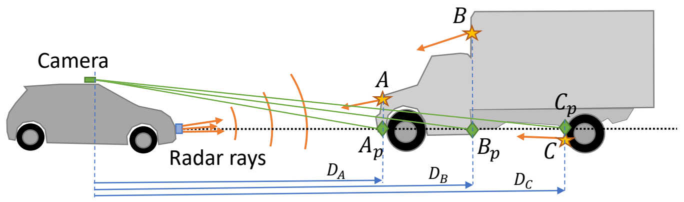
In addition to beam width, another source of projected point depth difference is occlusion caused by the significant baseline between radars on the grill, and cameras on the roof or driver mirror. Further, these depth differences only increase when radar hits are accumulated over a short interval, and thus more opportunity for occlusions.

In addition to pixel association errors, we are faced with the problem that automotive radars generate far sparser depth scans than lidar. There is typically a *single row* of returns, rather than anywhere up to 128 rows in lidar, and the azimuth spacing of radar returns can be an order of magnitude greater than lidar. This sparsity significantly increases the difficulty in depth completion. One solution is to accumulate radar pixels over a short time interval, and to account for their 3D position using both ego-motion and radial velocity. Nevertheless, this accumulation introduces additional pixel association errors (in part from not having tangential velocity) and more opportunities for occlusions.

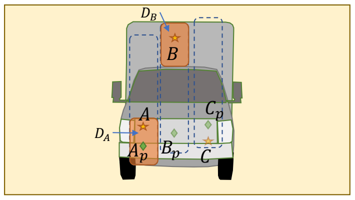
2.3.2 Radar-Camera Pixel Depth Association

In using radar to aid depth estimation we face the problem of determining which, if any, point in the image does a radar return correspond to? This radar pixel to camera pixel association is a difficult problem, and we do not have ground truth to determine this. Thus we reformulate this problem slightly to make it more tractable.

The new question we ask is: “Which pixels in the vicinity of the projected radar pixel have the same depth of that radar return?" We call this *Radar-Camera Pixel Depth Association*: RC-PDA or simply PDA. It is a one-to-many mapping, rather than one-to-one mapping, and has four key advantages. First, we do not need to distinguish between many good but ambiguous matches and rather can return many pixels with the same depth. This simplifies the problem. Second, by associating the radar return with multiple pixels, our method explicitly densifies the radar depth map, which facilitates the second stage of full-image depth estimation. Third, our question simultaneously addresses the occlusion problem; if there are no nearby pixels with that depth, then



(a)



(b)

Figure 2.4 Illustration of depth differences between camera and radar, and how our proposed association method (Pixel Depth Association: RC-PDA) can address this. (a) Radar hits are modeled on a ground-parallel plane (dashed black line). Actual returns may be outside this plane, as illustrated with orange stars A, B, C at depths D_A, D_B, D_C respectively. We project the corresponding in-plane points, A_p, B_p, C_p (green diamonds), into the camera, and call these the *radar pixels*. (b) The camera view showing the radar pixels A_p, B_p, C_p. Now the true image depth of these pixels is D_A, the front of the truck, which agrees only with A_p which is visible, and not for B_p and C_p which are occluded. This illustrates why radar pixel depths are often incorrect from the camera perspective. Finding associations from radar pixels to the projected true points A, B, C would solve this, but is difficult. Rather, we seek a neighborhood *depth association* for each radar pixel, that specifies which pixels within a neighborhood (dashed blue regions) have the same depth as the radar pixel, shown here by the orange regions. For example, the orange pixels in the neighborhood of A_p have a RC-PDA of 1 while the remaining neighborhood pixels have a RC-PDA of 0, all relative to A_p. See Sec. 2.3.2 for details.

the radar pixel is automatically inferred to be occluded. Fourth, we are able to leverage a lidar-based ground-truth depth map as the supervision, rather than a difficult-to-define “ground truth" pixel association. Fig. 2.4 illustrates image depths obtained from raw radar projections and RC-PDA around each radar pixel. It shows the height errors of measured radar points and how some hits visible to the radar are occluded from the camera.

2.3.2.1 RC-PDA Model

We model RC-PDA over a neighborhood around the projected radar pixel in the color image. At each radar pixel we define a patch around the radar location and seek to classify each pixel in this patch as having the same depth or not as the radar pixel, within a predetermined threshold. A similar connectivity model has been used for image segmentation [39]. Radar pixels and the patches around them are illustrated in Figs. 2.5(a) and (b), respectively.

The connection to each pixel in a $h \times w$ neighborhood has $N = wh$ elements, and can be encoded as an N-channel RC-PDA which we label $\mathbf{A}(i, j, k)$, where $k = 1, \cdots, N$. Here (i, j) is the radar pixel coordinate, and the k'th neighbor has offset (i_k, j_k) from (i, j). Now the label for

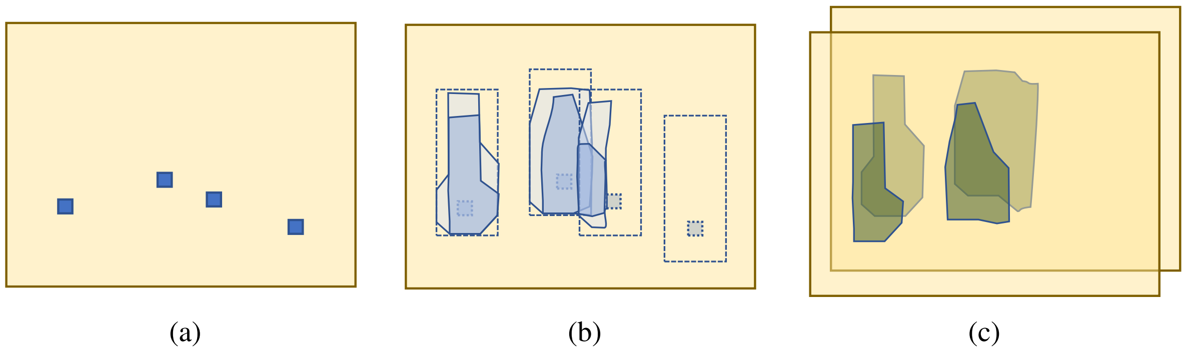


Figure 2.5 Overview of our radar depth representation. (a) Radar pixels indicate sparse depth in image space. (b) For each radar pixel, a Pixel Depth Association (RC-PDA) probability over neighboring pixels is calculated, indicated with shaded contours. (c) Radar pixel depths are propagated to neighboring pixels to create a Multi-channel Enhanced Radar (MER) image. Each channel is a densified depth at a given confidence level.

$\mathbf{A}(i, j, k)$ is 1 if the neighboring pixel has the same depth as radar pixel and 0 otherwise. More precisely, if $E_{ijk} = d(i, j) - d_T(i + i_k, j + j_k)$ is the difference between radar pixel depth, $d(i, j)$, and the neighboring lidar pixel depth, $d_T(i+i_k, j+j_k)$, and $\tilde{E}_{ijk} = E_{ijk}/d(i, j)$ is the relative depth difference, then:

$$\mathbf{A}(i, j, k) = \begin{cases} 1, & \text{if } (|E_{ijk}| < T_a) \wedge (|\tilde{E}_{ijk}| < T_r) \\ 0, & \text{otherwise.} \end{cases} \tag{2.1}$$

We note that labels $\mathbf{A}(i, j, k)$ are only defined when there is both a radar pixel at (i, j) and a lidar depth $d_T(i+i_k, j+j_k)$. We define a binary weight $w(i, j, k) \in \{0, 1\}$ to be 1 when both conditions are satisfied and 0 otherwise. During training we minimize the weighted binary cross entropy loss [39] between labels $\mathbf{A}(i, j, k)$ and predicted RC-PDA:

$$\mathcal{L}_{CE} = \sum_{i,j,k} w(i, j, k)[-\mathbf{A}(i, j, k)z(i, j, k) + \log(1 + \exp(z(i, j, k)))]. \tag{2.2}$$

The network output, $z(i, j, k)$, is passed through a Sigmoid to obtain $\hat{\mathbf{A}}(i, j, k)$, the estimated RC-PDA.

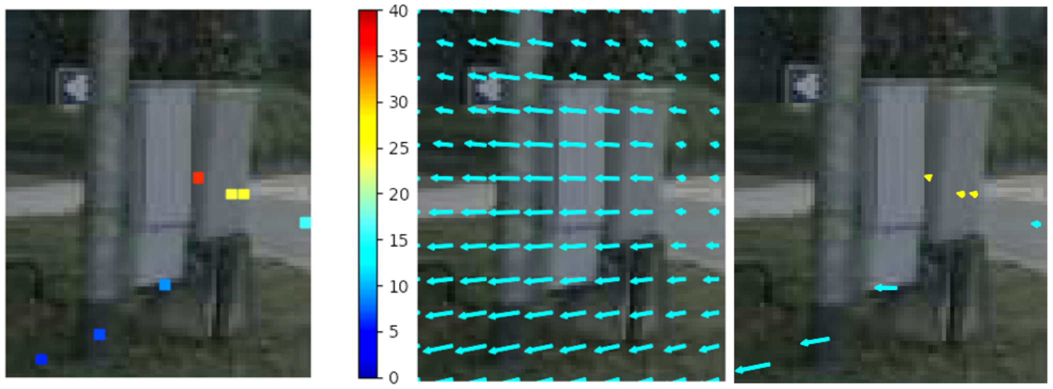
Our network thus predicts a RC-PDA confidence in a range of 0 to 1 representing the probability that each pixel in this patch has the same depth as the radar pixel. This prediction also applies to the image pixel at the same coordinates as the radar pixel, i.e., (i, j), as like other pixels, the depth at this image pixel may differ from the radar depth for a variety of reasons, including those illustrated in Fig. 2.4.

2.3.3 From RC-PDA to MER

The RC-PDA gives the probability that neighboring pixels have the same depth as the measured radar pixel. We can convert the radar depths along with predicted RC-PDA into a partially filled depth image plus a corresponding confidence as follows. Each of N neighbors to a given radar pixel is given depth $d(i, j)$ and confidence $\hat{\mathbf{A}}(i, j, k)$. If more than one radar depth is expanded to the same pixel, the radar depth with the maximum RC-PDA is kept. The expanded depth is represented as $\mathbf{D}(i, j)$ with confidence $\hat{\mathbf{A}}(i, j)$. Now many of the low-confidence pixels will have incorrect depth. Instead of eliminating low-confidence depths, we convert this expanded depth image into a multi-channel image where each channel l is given depth $\mathbf{D}(i, j)$ if its confidence $\hat{\mathbf{A}}(i, j)$ is greater than a channel threshold T_l, where $l = 1, \cdots, N_e$ and N_e is the total number of channels of the enhanced depth. The result is a *Multi-Channel Enhanced Radar* (MER) image with each channel representing radar-derived depth at a particular confidence level (see Fig. 2.5(c)).

Our MER representation for depth can correctly encode many complex cases of radar-camera projection, a few of which are illustrated in Fig. 2.4. These cases include when radar hits are occluded and no nearby pixels have similar depth. They also include cases where the radar pixel is just inside or just outside the boundary of a target. In each case, those nearby pixels with the same depth as the radar can be given the radar depth with high confidence, while the remaining neighborhood pixels are given low confidence, and their depth are specified on separate channels of the MER.

The purpose of using multiple channels for depths with different confidences in MER is to facilitate the task of Network 2 in Fig. 2.2 in performing the dense depth completion. High confidence channels give the greatest benefit, but low confidence channels may also provide useful data. In all cases they densify the depth beyond single radar pixels, easing the depth completion task.



(a) Radar depth (b) Optical flow (c) Radar flow

Figure 2.6 An example of how radar scene flow and optical flow differences are used to infer occlusions of radar pixels. The radar flows are plotted as yellow if the L_2 norm of radar/optical flow differences are larger than a threshold. Note that we do not explicitly filter radar, rather provide flow to Network 1 in Fig. 2.2 so that it can implicitly filter radar while estimating RC-PDA.

2.3.4 Estimating RC-PDA

We next select inputs to Network 1 in Fig. 2.2 from which it can learn to infer the RC-PDA. These are the image, the radar pixels with their depths as well as the image flow and the radar flow from current to a neighboring frame. Here we briefly explain the intuition for each of these.

The image provides scene context for each radar pixel, as well as object boundary information. The radar pixels provide depth for interpreting the context and a basis for predicting the depth of nearby pixels. As radar is very sparse, we accumulate radar from a short time history, 0.3 seconds, and transform it into the current frame using both ego-motion and the radial velocity similar to that done in [93].

Now a pairing of image optical flow and radar scene flow provides an occlusion and depth difference cue. For static objects, the optical flow should exactly equal radar scene flow, when the pixel depth is the same as radar pixel depth. Conversely, radar pixels that are occluded from the camera view will have different scene flow from the optical flow of a static object occluding them (Fig. 2.6). Similarly, objects moving radially will have consistent flow. By providing flow, we expect that Network 1 will learn to leverage flow similarity in predicting RC-PDA for each radar pixel.

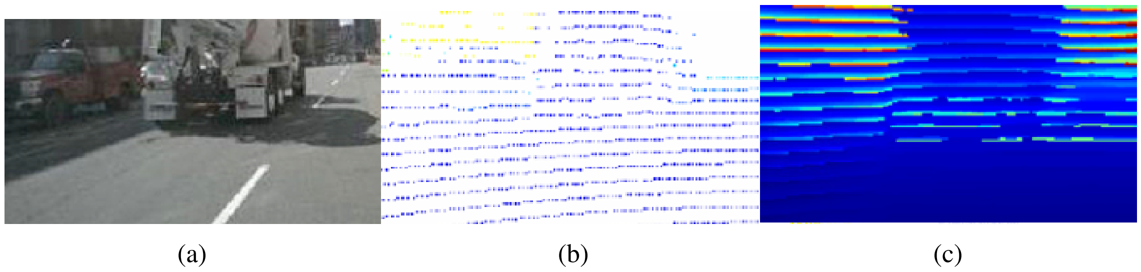


Figure 2.7 We noticed that when lidar with a regular scan pattern, as in (b) for image (a), is used to train depth completion, our network learns to predict the lidar points well, but not the remaining pixels. This leaves large artifacts, as in (c), and motivates us to create a semi-dense depth lidar training set.

2.3.5 Lidar-based Supervision

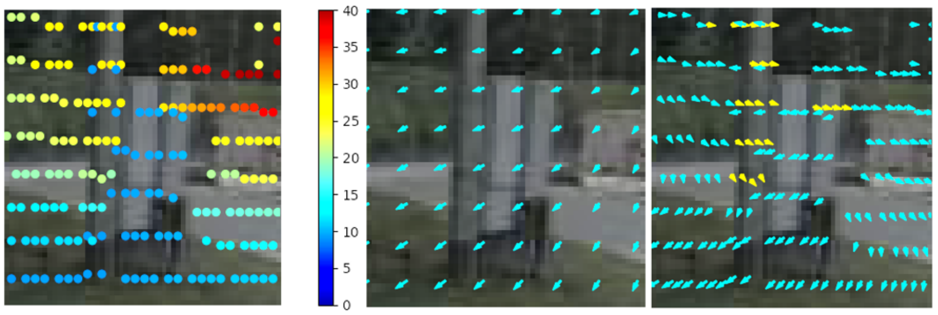
To train both the RC-PDA and the final dense depth estimate, we use a dense ground truth depth. This is because, as illustrated in Fig. 2.7, training with sparse lidar leads to significant artifacts. We now describe how we build a semi-dense depth image from lidar scans.

2.3.5.1 Lidar Accumulation

To our knowledge, there is no existing public dataset specially designed for depth completion with radar. Thus we create a semi-dense ground truth depth from nuScenes dataset, a public dataset with radar data and designed for object detection and segmentation. We use the 32-ray lidar as depth label and notice that the sparse depth label generated from a single frame will lead to a biased model predicting depth with artifacts, i.e., only predictions for pixels with ground truth are reasonable. Thus, we use semi-dense lidar depth as label, which is created by accumulating multiple lidar frames. With ego motion and calibration parameters, all static points can be transformed to destination image frame. Moving points are compensated by bounding box poses at each frame, which are estimated by interpolating bounding boxes provided by nuScenes in key frames.

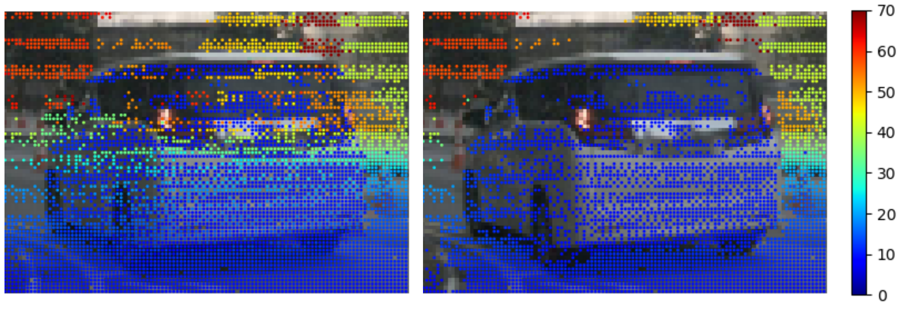
2.3.5.2 Occlusion Removal via Flow Consistency

When a foreground object occludes some of the accumulated lidar points, the resulting dense depth may include depth artifacts as the occluded pixels appear in gaps in the foreground object. KITTI [131] takes advantages of the depth from stereo images to filter out such occluded points. As no stereo images are available in nuScenes, we propose detecting and removing occluded lidar



(a) Lidar depth (b) Optical flow (c) Lidar flow

Figure 2.8 An example of how lidar scene flow and optical flow differences are used to infer occlusions of lidar pixels. Lidar flows are plotted as yellow if the L_2 norm of lidar/optical flow differences are larger than a threshold. This is used in the accumulation of lidar for building ground-truth depth maps, see Fig. 2.9.



(a) (b)

Figure 2.9 An example of using lidar flow and optical flow consistency to filter occluded pixels. (a) Accumulated lidar depth, and (b) Accumulated lidar depth with flow consistency filtering.

points based on optical-scene flow consistency.

The scene flow of lidar points, termed *lidar flow*, is computed by projecting lidar points into two neighboring images and measuring the change in their coordinates. On moving objects, the point's positions are corrected with the object motion. On static visible objects, lidar flow will equal optical flow, while on occluded surfaces lidar flow is usually different from the optical flow at the same pixel, see Fig. 2.8. We calculate optical flow with [129] pretrained on KITTI, and measure the difference between the two flows at the same pixel via the L_2 norm of their difference. Points with flow difference larger than a threshold T_f are discarded as occluded points. Fig. 2.9 shows an example of using flow consistency to filter out occluded lidar depths.

2.3.5.3 Occlusion Removal via Segmentation

Flow-based occluded pixel removal may fail in two cases. When there is little to no parallax, both optical and scene flow will be small, and their difference becomes not measurable. This occurs mostly at long range or along the motion direction. Further, lidar flow on moving objects can in some cases be identical to the occluded lidar flow behind it. In both of these cases flow consistency is insufficient to remove occluded pixels from the final depth estimate.

To solve this problem, we use a combination of 3D bounding boxes and semantic segmentation to remove occluded points appearing on top of objects. First, accurate pixel region of an instance is determined by the intersection of 3D bounding box projection and semantic segmentation. The maximum depth of bounding box corners is used to decide whether lidar points falling on the object are on it or behind it. Points within the semantic segmentation and closer than this maximum distance are kept, while points in the segmentation and behind the bounding box are filtered out as occluded lidar points. Fig. 2.10 shows an example of removing occluded points appearing on vehicle instances. We use a semantic segmentation model [15] pre-trained with CityScape [17] to segment vehicle pixels.

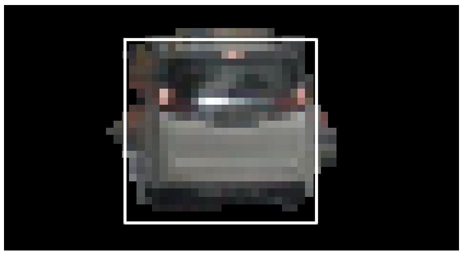
2.3.6 Algorithm Summary

We propose a two-stage depth estimation process, as in Fig. 2.2. The Stage 1 estimates RC-PDA for each radar pixel, which is transformed into our MER representation as detailed in Sec. 2.3.3 and fed into Stage 2 which performs conventional depth completion. Both stages are supervised by the accumulated dense lidar, with pixels not having a lidar depth given zero weight. Network 1 uses an encoder-decoder network with skip connections similar to U-Net [108] and [78] with details in supplementary material.

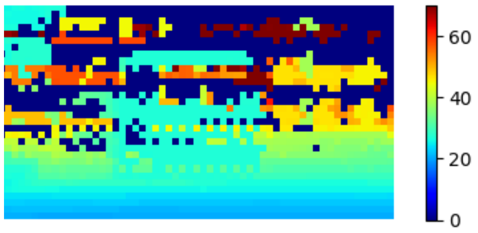
2.4 Experimental Results

Dataset We train and test on a subset of images from the nuScenes dataset [10], including $12,610$, $1,628$, and $1,623$ samples for training, validation, and testing, respectively. The data are collected with moving ego vehicle so flow calculation described in Sec. 2.3.5.2 can be applied. The depth range for training and testing is 0-50 meters. Resolutions of inputs and outputs are 400×192. As

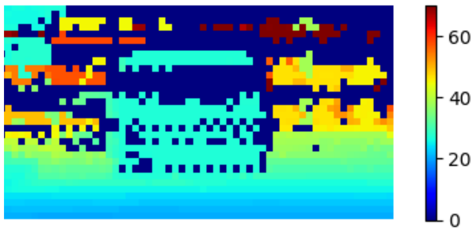
(a) Car image



(b) Semantic seg. & bound. box



(c) Depth before filtering



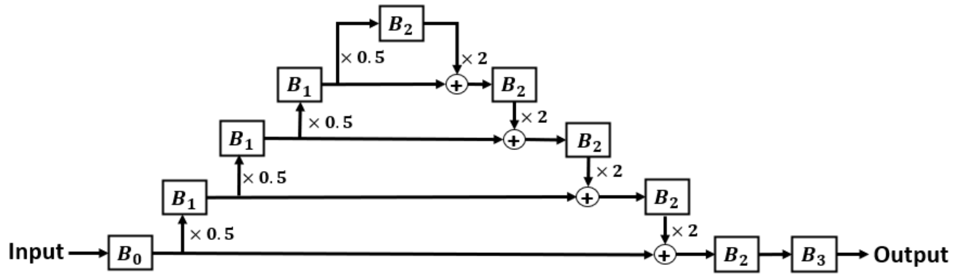
(d) Depth after filtering

Figure 2.10 For small flow instances and some movers, flow consistency is insufficient to remove accumulated but occluded lidar pixels, see (a,c). To remove these occluded pixels, we first find vehicle pixels as the intersection between semantic segmentation and 2D bounding box, see (b). From the 3D bounding box we know the maximum depth of the vehicle, and so can filter out all accumulated depths greater than this that are actually occluded, see (d).

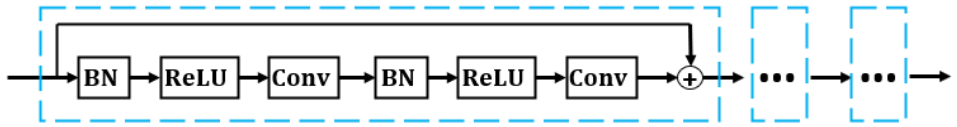
described in Sec. 2.3.5, we build semi-dense depth images by accumulating lidar pixels from 21 subsequent frames and 4 previous frames (sampled every other frame), and use these for supervision.

Implementation details For parameters, we use $T_a = 1$ m and $T_r = 0.05$ for Eq. 2.1. In Sec. 2.3.5.2 we use $T_f = 3$ to decide flow consistency. MER has 6 channels with T_1 to T_6 set as 0.5, 0.6, 0.7, 0.8, 0.9, and 0.95, respectively. At Stage 1, we use a U-Net with 5 levels of resolutions and 180 output channels, corresponding to 180 pixels in a rectangle neighborhood with size $w = 5$ and $h = 36$. As the radar points are typically on the lower part of image, to fully leveraging the neighborhood, the neighborhood center is below the rectangle center with 30 pixel above, 5 pixels below and 2 pixels on left and right to provide more space for radar points to extend upwards. At Stage 2, we employ two existing depth completion architectures, [79] and [56], originally designed for lidar-camera pairs.

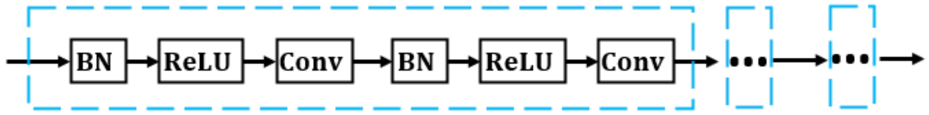
Network Details Fig. 2.11 shows a typical structure of U-Net [108] we use for both Stages 1 and architecture [79] for Stage 2. It has five levels of resolutions where downsampling and upsampling



(a)



(b)



(c)

Figure 2.11 (a) U-Net used in our experiment. (b) Diagram of Block B_1, a series of residual blocks (shown as blue boxes). (c)Diagram of block B_2, a sequence of convolutional blocks (shown as blue box). BN and $Conv$ denote batch normalization and a convolutional layer, respectively.

are achieved by max-pooling and nearest neighbor interpolation. Except the input and output, all tensors in the network have the same number of channels N_c. All convolutions use 3×3 filters. B_0 represents the input block, a convolution layer that increases the number of channels to N_c. B_1 denotes residual blocks connected in series, and B_2 is a sequence of convolutional blocks where each block consists of batch normalization, ReLU, and convolution. B_3 is the output block which includes batch normalization, ReLU, and a convolutional layer changing the number of channels from N_c to that of output channels.

In our experiment, for Stage 1 network, we set N_c to 80. There are two residual blocks in B_1 and four convolutional blocks in B_2, respectively. For Stage 2 network, N_c is set to 64. B_1 has four residual blocks, and B_2 has 8 convolutional blocks. We use RMSProp as optimizer with a learning rate of 5×10^{-5}.

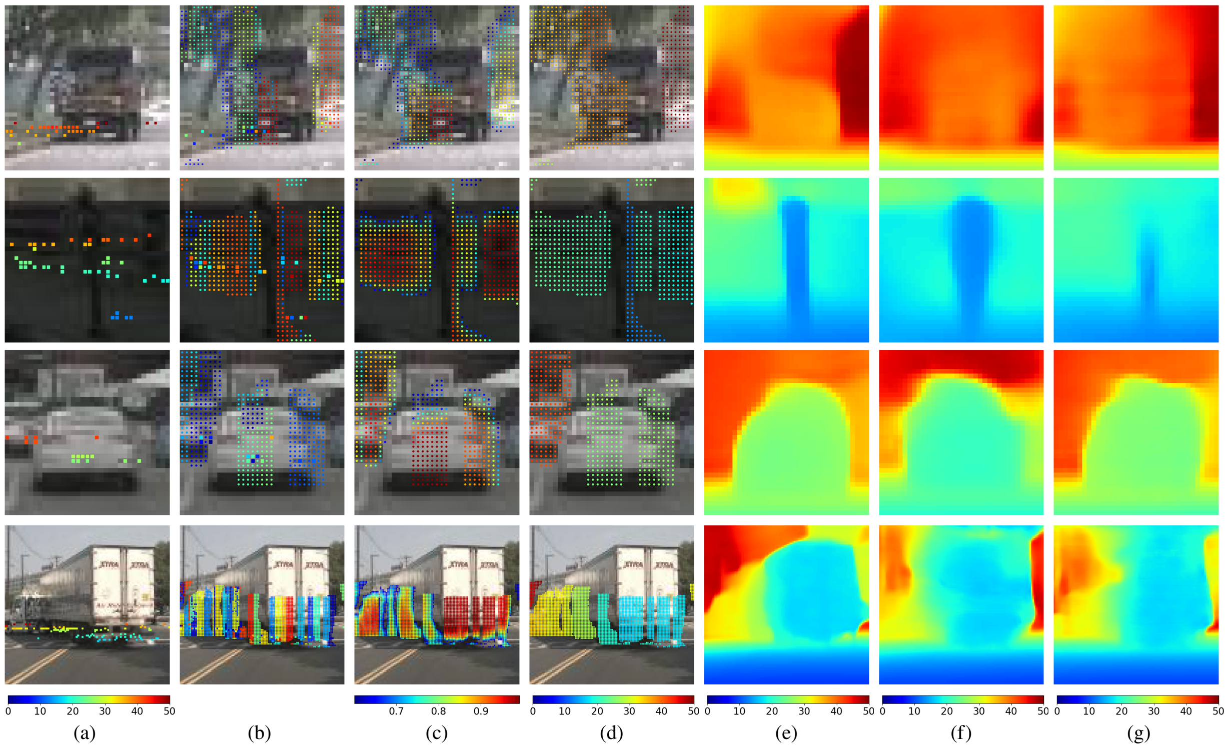


Figure 2.12 (a) Raw radar depths (b) Each color pixel with a maximum RC-PDA > 0.6 is marked with a color indicating which radar pixel it is associated with. (c) The RC-PDA score with values > 0.6 for each pixel. (d) The MER channel with RC-PDA > 0.6. (e) Our final predicted depth. (f) Depth from monocular input to Stage 2. (g) Depth from monocular and raw radar input to Stage 2.

Inference time When we use Intel Core i7-8700 CPUs and an NVIDIA GeForce RTX 2080 Ti GPU, the inference times of Network 1 and Network 2 are 4.5 ms and 8.2 ms per frame, respectively.

2.4.1 Visualization of Predicted RC-PDA

The predicted RC-PDA and estimated depths from Stage 2 [79] are visualized in Fig. 2.12. Column (a) shows the raw radar pixels plotted on images and often includes occluded radar pixels. Column (b) shows how image pixels are associated with different radar pixels according to their maximum RC-PDA. Radar pixels and their associated neighboring pixels are marked with the same color. Notice in column (c) that RC-PDA is high within objects and decreases after crossing boundaries. Occluded radar depth are mostly discarded as their predicted RC-PDA is low. In column (e), the dense depths predicted from MER are improved over predictions from (g) raw radar and/or (f) monocular. For example, in Row 2 of Fig. 2.12, our predicted pole depth in (e) has better boundaries than monocular-only in (f), and monocular plus raw radar in (g). How we achieve this

Network 2	Input	MAE	Abs Rel	RMSE	RMSE log
Ma et al. [79]	Image	2.385	0.110	3.505	0.150
	Image, radar	1.609	0.078	2.865	0.126
	Image, radar, MER	**1.229**	**0.058**	**2.651**	**0.114**
Li et al. [56]	Image, radar	1.759	0.084	3.039	0.133
	Image, radar, MER	**1.274**	**0.061**	**2.670**	**0.116**
None	MER	1.251	0.059	2.701	0.117
	Radar	7.369	0.475	10.900	0.448

Table 2.1 Depth error (m) in image regions around non-occluded radar returns, defined as regions with RC-PDA > 0.9.

can be intuitively understood by comparing raw radar in (a) with our MER in (d), the output of Stage 1. While raw radar has many incorrect depths, MER selects correct radar depths and extends these depths along the pole and background, enabling improved final depth inference.

2.4.2 Accuracy of MER

To be useful in improving radar-camera depth completion, the enhanced radar depth in the vicinity of radar points should be better than alternatives. We compare the depth error of the MER in regions where RC-PDA is > 0.9, with a few baseline methods, and results are shown in Tab. 2.1. The enhanced radar depth from Stage 1 improves over not only raw radar depth but also depth estimates from Stage 2 using monocular as well as monocular plus radar. In comparison, Stage 2 keeps the accuracy of the enhanced radar depth when using it for depth completion. The depth error for raw radar depth is very large since many of them are occluded and far behind foreground. About 35% of radar points in the test frames have a maximum RC-PDA smaller than T_1 in their neighborhood and are discarded as occluded points. Further, Fig. 2.13 shows the depth error and per-image average area of expanded depth from 6 MER channels, respectively. It shows, as confidence increases, higher RC-PDA corresponds to higher accuracy and smaller expanded areas.

2.4.3 Comparison of Depth Completion

To evaluate effectiveness of the enhanced radar depth in depth completion, we compare the depth error with and without using MER as input for Network 2, and show performance in Tabs. 2.2 and 2.3. The results show that including radar improves depth completion over monocular, while

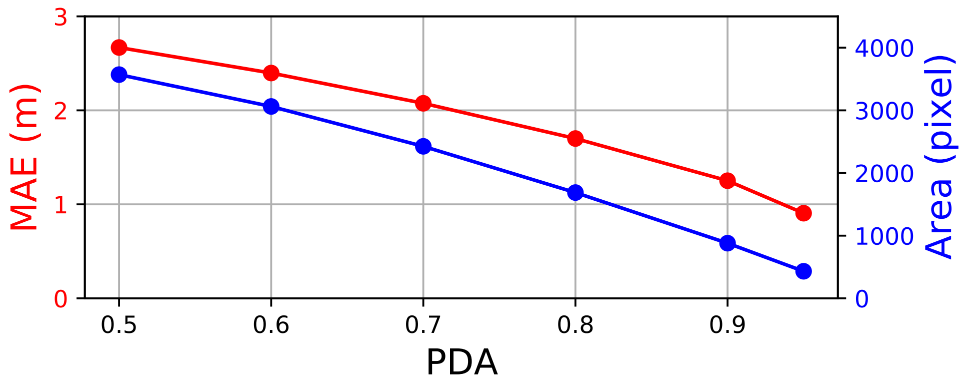


Figure 2.13 Image area and depth error of enhanced radar in MER for regions with minimal RC-PDA at 6 confidence levels.

Network 2	Input	MAE	Abs Rel	RMSE	RMSE log
Ma et al. [79]	Image	1.808	0.102	3.552	0.160
	Image, radar	1.569	0.090	3.327	0.152
	Image, radar, MER	**1.472**	**0.085**	**3.179**	**0.144**
Li et al. [56]	Image, radar	1.821	0.107	3.650	0.170
	Image, radar, MER	**1.655**	**0.094**	**3.463**	**0.159**

Table 2.2 Full-image depth estimation/completion errors (m).

Network 2	Input	MAE	Abs Rel	RMSE	RMSE log
Ma et al. [79]	Image	2.673	0.153	4.259	0.202
	Image, radar	2.263	0.134	4.028	0.194
	Image, radar, MER	**2.078**	**0.124**	**3.864**	**0.183**
Li et al. [56]	Image, radar	2.515	0.154	4.266	0.211
	Image, radar, MER	**2.189**	**0.132**	**3.943**	**0.193**

Table 2.3 Depth estimation/completion errors (m) in the low-height region (0.3-2 meters above ground).

using our proposed MER further improves the accuracy of depth completion for the same network.

Qualitative comparisons between depth completion [79] with and without using MER are shown in Fig. 2.14. This shows improvement from MER in estimating object depth boundaries including close objects (such as the traffic sign on the bottom image) and far objects.

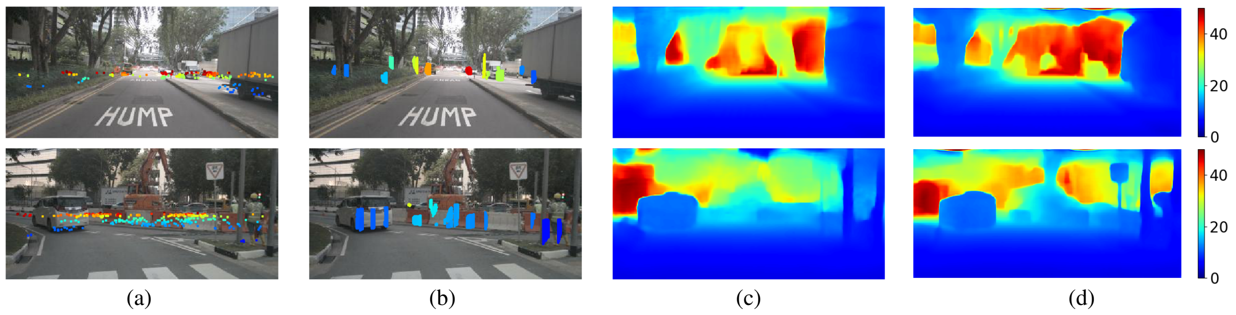


Figure 2.14 Qualitative depth completion comparison showing gains from using MER over raw radar. (a) Raw radar on top of image versus (b) A MER channel with RC-PDA > 0.8 on top of image. Depth completion (c) without and (d) with using MER.

2.4.4 Ablation Studies

2.4.4.1 Neighborhood Sizes

The neighborhood is a rectangular region around a radar pixel, which is defined as the neighborhood center. A neighborhood is designed to let the radar pixel have more space to expand upwards, and thus, typically, the neighborhood center does not correspond to the geometric center. As shown in Fig. 2.15, we use (w_1, w_2, h_1, h_2) to represent the shape of a neighborhood, where w_1 and w_2 are the number of horizontal neighboring pixels to the left and right of the neighborhood center, and h_3 and h_4 denote the number of vertical neighboring pixels above and below the center.

As shown in Table 2.4, we test different neighborhood sizes for Stage 1 network and compute their resultant depth error after using them in Stage 2 network [79]. The T_l to create MER in Stage 2 are $(0.6, 0.7, 0.8, 0.9, 0.95)$.

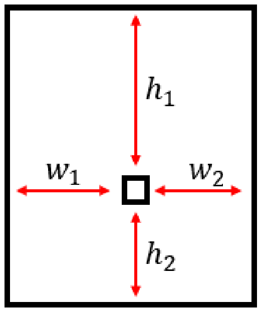


Figure 2.15 A neighborhood with shape (w_1, w_2, h_1, h_2). The small square represents a radar pixel.

Neighborhood	MAE	Abs Rel	RMSE	RMSE log
$(2, 2, 30, 5)$	**1.487**	**0.085**	**3.210**	**0.145**
$(3, 3, 20, 5)$	1.497	0.086	**3.210**	0.146
$(1, 1, 8, 1)$	1.536	0.088	3.274	0.148

Table 2.4 Full-image depth estimation/completion errors (m).

PDA Thresholds	MAE	Abs Rel	RMSE	RMSE log
$(0.6 : 0.1 : 0.9, 0.95)$	1.487	0.085	3.210	0.145
$(0.5 : 0.1 : 0.9, 0.95)$	**1.472**	0.085	**3.179**	**0.144**
$(0.6 : 0.05 : 0.9, 0.95)$	1.494	**0.084**	3.229	0.146

Table 2.5 Full-image depth estimation/completion errors with MER created by different thresholds (m). In the first column, "a:s:b" indicates a set of thresholds $a, a + s, a + 2s, \cdots, b$.

2.4.4.2 PDA Thresholds for MER

MER can be generated with different PDA threshold T_l where $l = 1, 2, ..., N_e$. With fixed neighorbood size $(2, 2, 30, 5)$, we test different PDA thresholds and compute their resultant depth error. Results are shown in Table 2.5.

2.4.5 Visualization of MER

Fig. 2.16 shows MER created by PDA thresholds of 0.6, 0.7, 0.8, 0.9, and 0.95, respectively. We can see the expanded region decreases as PDA threshold increases.

2.5 Summary

Radar-based depth completion introduces additional challenges and complexities beyond lidar-based depth completion. A significant difficulty is the large ambiguity in associating radar pixels with image pixels. We address this with RC-PDA, a learned measure that associates radar hits with nearby image pixels at the same depth. From RC-PDA we create an enhanced and densified radar image called MER. Our experiments show that depth completion using MER achieves improved accuracy over depth completion with raw radar. As part of this work we also create a semi-dense accumulated lidar depth dataset for training depth completion on nuScenes.

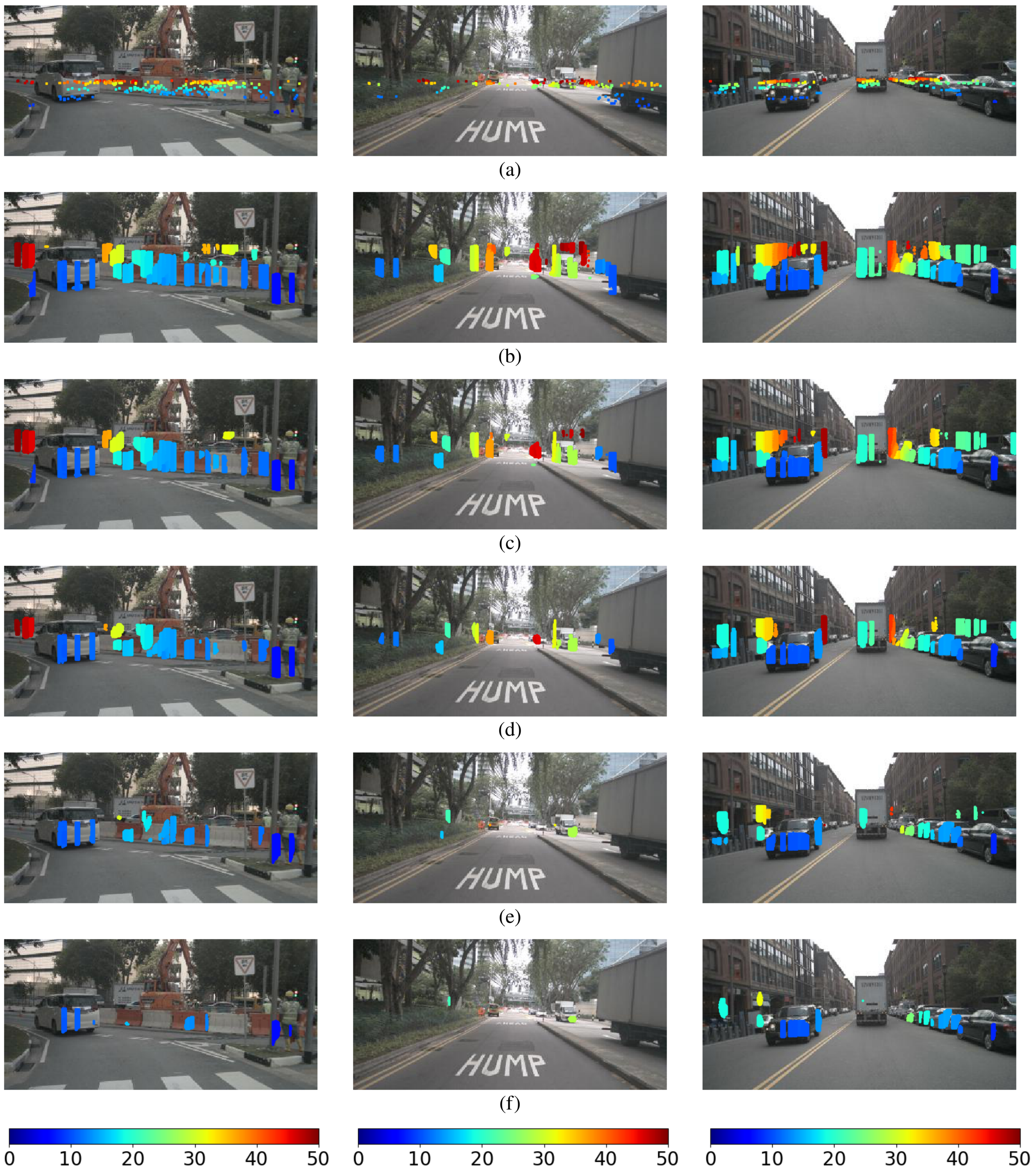


Figure 2.16 (a) Raw radar depth. MER with PDA thresholds (b) 0.6, (c) 0.7, (d) 0.8, (e) 0.9, and (f) 0.95. Occluded radar points are filtered out, and corrected depth are expanded in MER.

CHAPTER 3

FULL-VELOCITY RADAR RETURNS BY RADAR-CAMERA FUSION

A distinctive feature of Doppler radar is the measurement of velocity in the radial direction for radar points. However, the missing tangential velocity component hampers object velocity estimation as well as temporal integration of radar sweeps in dynamic scenes. Recognizing that fusing camera with radar provides complementary information to radar, in this chapter we present a closed-form solution for the point-wise, full-velocity estimate of Doppler returns using the corresponding optical flow from camera images. Additionally, we address the association problem between radar returns and camera images with a neural network that is trained to estimate radar-camera correspondences. Experimental results on the nuScenes dataset verify the validity of the method and show significant improvements over the SOTA in velocity estimation and accumulation of radar points. This chapter was previously published as [75].

3.1 Introduction

Radar is a mainstream automotive 3D sensor, and along with lidar and camera, is used in perception systems for driving assistance and autonomous driving [6, 59, 125]. Unlike lidar, radar has been widely installed on existing vehicles due to its relatively low cost and small sensor size, which makes it an easy fit into various vehicles without changing their appearance. Thus, advances in radar vision systems have potential to make immediate impact on vehicle safety. Recently, with the release of a couple of autonomous driving datasets with radar data included, e.g., Oxford Radar RobotCar [3] and nuScenes [10], there is great interest in the community to explore how to leverage radar data in various vision tasks such as object detection [91, 149].

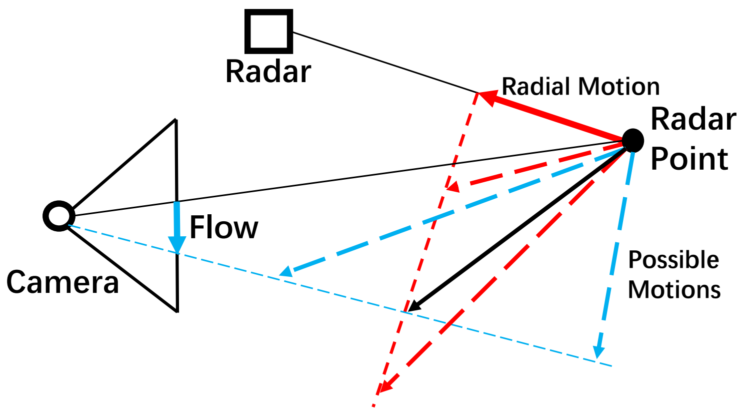
In addition to measuring 3D positions, radar has the special capability of obtaining radial velocity of returned points based on the Doppler effect. This extra capability is a significant advantage over other 3D sensors like lidar, enabling, for instance, instantaneous moving object detection. However, due to the inherently ambiguous mapping from radial velocity to full velocity, using radial velocity directly to account for the real movement of radar points is inadequate and sometimes misleading. Here, the full velocity denotes the actual velocity of radar points in 2D or

3D space. While radial velocity can well approximate full velocity when a point is moving away from or towards the radar, these two can be very different when the point is moving in the non-radial directions. An extreme case occurs for objects moving tangentially as these will have zero radial velocity regardless of target speed. Therefore, acquiring point-wise full velocity instead of radial velocity is crucial to reliably sense the motion of surrounding objects.

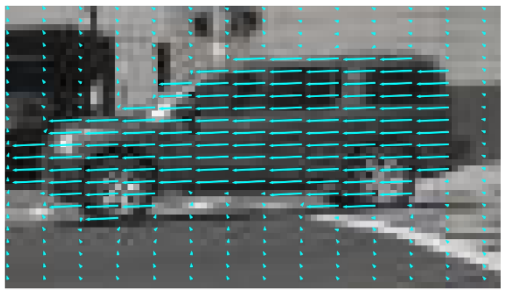
Apart from measuring the velocity of objects, another important application of point-wise velocity is the accumulation of radar points. Radar returns from a single frame are much sparser than lidar in both azimuth and elevation, e.g., typically lidar has an azimuth resolution $10\times$ higher than radar [149]. Thus, it is often essential to accumulate multiple prior radar frames to acquire sufficiently dense point clouds for downstream tasks, e.g., object detection [12, 14, 93]. To align radar frames, in addition to compensating egomotion, we shall consider the motion of moving points in consecutive frames, which can be estimated by point-wise velocity and time of movement. As the radial velocity does not reflect the true motion, it is desirable to have point-wise full velocity for point accumulation.

To solve the aforementioned dilemma of radial velocity, we propose to estimate point-wise full velocity of radar returns by fusing radar with a RGB camera. Specifically, we derive a closed-form solution to infer point-wise full velocity from radial velocity as well as associated projected image motion obtained from optical flow. As shown in Fig. 3.1, constraints imposed by optical flow resolve the ambiguities of radial-full velocity mapping and lead to a unique and closed-form solution for full velocity. Our method can be considered as a way to enhance raw radar measurement by upgrading point-wise radial velocity to full velocity, laying the groundwork for improving radar-related tasks, e.g., velocity estimation, point accumulation, and object detection.

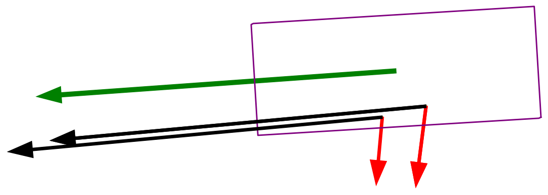
Moreover, a prerequisite for our closed-form solution is the association between moving radar points and image pixels. To enable a reliable association, we train a neural network to predict radar-camera correspondences as well as discerning occluded radar points. Experimental results demonstrate that the proposed method improves point-wise velocity estimates and their use for object velocity estimation and radar point accumulation.



(a)



(b)



(c)

Figure 3.1 (a) Full motion cannot be determined with a single sensor: all motions ending on the blue dashed line (i.e., blue dashed arrows) map to the same optical flow and all motions terminated on the red dashed line (i.e., red dashed arrows) fit the same radial motion. However, with a radar-camera pair, the full motion can be uniquely decided: only the motion drawn in black satisfies both optical flow and radial motion. (b) Optical flow in the camera-image and (c) a BEV of the observed vehicle. This shows measured radar points with radial velocity (red), our predicted point-wise, full velocity (black), and GT full velocity of the vehicle (green).

In summary, the main contributions of this work are:

- We define a novel research task for radar-camera perception systems, i.e., estimating point-wise full velocity of radar returns by fusing radar and camera.
- We propose a novel closed-form solution to infer full radar-return velocity by leveraging the radial velocity of radar points, optical flow of images, and the learned association between radar points and image pixels.
- We demonstrate SOTA performance in object velocity estimation, radar point accumulation, and 3D object localization.

3.2 Related Works

Application of Radar in Vision. Radar data differs from lidar data in various aspects [9]. In addition to the popular point representation (also named radar target [97]), an analogy to lidar points, there are other radar data representations containing more raw measurements, e.g., range-azimuth image and spectrograms, which have been applied in tasks such as activity classification [115], detection [62], and pose estimation [109]. Our method is based on radar points, with the format available in the nuScenes dataset [10].

The characteristics of radar have been explored to complement other sensors. The Doppler velocity of radar points is used to distinguish moving targets. For example, RSS-Net [40] uses radial velocity as a motion cue for image semantic segmentation. Chadwick et al. [12] use radial velocity to detect distant moving vehicles—difficult to detect with only images. Fritsche et al. [24] combine radar with lidar for measurement under poor visibility. With a longer detection range than lidar, radar is also deployed with lidar to better detect far objects [149].

The *sparsity* of radar makes it difficult to directly apply well-developed techniques for lidar on radar [62, 91]. For example, Danzer et al. [18] adopt PointNets [102] on radar points for 2D car detection, while sparsity limits it to large objects like cars. Similar to lidar-camera depth completion [33, 34], Long et al. [76] develop radar-camera depth completion by learning a probabilistic mapping from radar returns to images. To obtain denser radar points, Lombacher et al. [71] use occupancy grid [22] to accumulate radar frames. Yet, the method assumes a static scene and cannot cope with moving objects. Radar points are projected on images and represented as regions near projected points, such as vertical bars [93] and circles [12, 14], to account for uncertainty of projection due to measurement error. While accumulating radar frames is desirable, without reliably compensating object motion, these methods need to carefully decide the number of frames to trade off between the gain in accumulation and loss in accuracy due to delay [93]. Our estimated point-wise velocity can compensate object motion and realize more accurate accumulation.

Velocity Estimation in Perception Systems. Researchers have used monocular videos [8] or radial velocity of radar points to estimate *object-wise* velocity. With only radar data of a single frame, Kellner et al. [41, 42] compute full velocity of moving vehicles from radial velocities and azimuth angles of at least two radar hits. However, for a robust solution, the method requires that 1) radar captures more radar hits on each object; 2) radar points have significantly different azimuth angles; and 3) object points are clustered before velocity estimation [41, 112, 114]. Obviously due to sparsity of radar in a single frame, it is difficult to obtain at least two radar hits on distant vehicles, let alone objects of smaller sizes. Also, it is common that radar points on the same object, e.g., a distant or small object, have similar azimuth.

Recognizing the density and accuracy limitation of radar, researchers fuse radar with other sensors, e.g., lidar and camera, for object-wise velocity estimation. Specifically, existing techniques [58, 144, 157] for images or lidar are employed to obtain preliminary detections. Radar data, including radial velocity, once associated with the initial detections, are used as additional cues to predict full velocities of objects. For instance, in RadarNet [149] temporal point clouds of radar and lidar, modeled as voxels, are used to acquire initial detections and their motions. Object motion direction is used to resolve the ambiguities in radar-point association by back-projecting their radial velocities on the motion direction. Yet, a sequence of lidar frames is required to obtain the initial detection and motion estimation.

CenterFusion [91] integrates radar with camera for object-wise velocity estimation. Well-developed image-based detector is applied to extract preliminary boxes. After associating radar points with detections, the method combines radar data, radial velocity, and depth, with image features within detected regions to regress a full velocity per detection. However, without a closed-form solution, the mapping from radial to full velocity needs to be learned from a great number of labeled data. In contrast, we present a point-wise *closed-form solution* for full-velocity estimation of radar points, without performing object detection. To our knowledge, there is no prior method able to perform point-wise full-velocity estimation for radar returns.

3.3 Proposed Method

We consider the case of a camera and radar rigidly attached to a moving platform, e.g., a vehicle, observing moving objects in the environment. In this section we develop equations relating optical

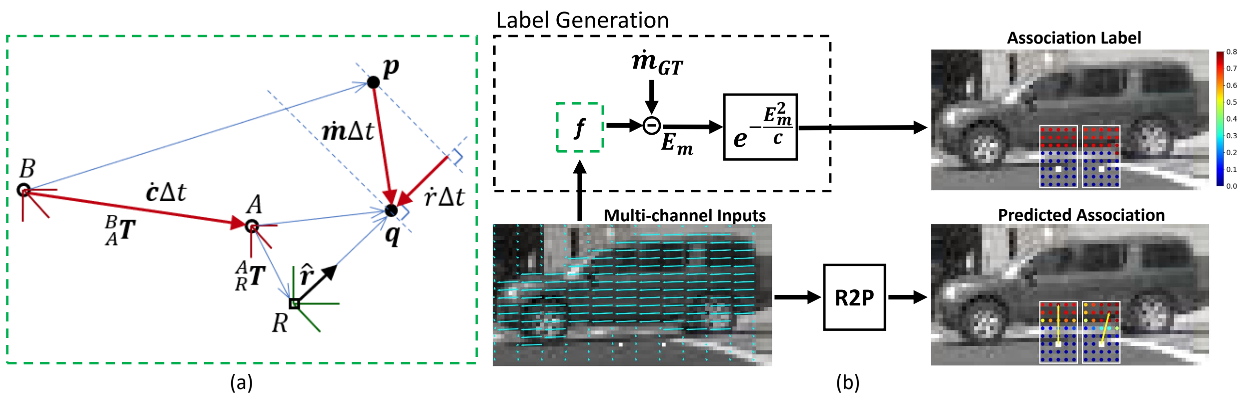


Figure 3.2 Full velocity estimation and learning to associate radar points to camera pixels. (a) A 3D point, $\boldsymbol{p}$, is observed by a camera at B. A short interval, Δt, later, the point has moved by $\dot{\boldsymbol{m}}\Delta t$ to $\boldsymbol{q}$ while the camera has moved by $\dot{\boldsymbol{c}}\Delta t$ to A. At the same time, the radar measures both the position of $\boldsymbol{q}$ and the radial speed $\dot{r}$, which is the radial component of $\dot{\boldsymbol{m}}$. Using radial speed $\dot{r}$ and the associated optical flow of $\boldsymbol{q}$ in images, we derive a closed-form equation (denoted as $\boldsymbol{f}()$) to estimate $\boldsymbol{q}$'s full velocity $\dot{\boldsymbol{m}}$. (b) As the closed-form solution requires point-wise association of two sensors, we train a Radar-2-Pixel (R2P) network to take a multi-channel input and predict the association probabilities for pixels within a neighborhood of the raw projection (white dot) obtained via known pose ${}^{A}_{R}\boldsymbol{T}$. A pixel with the highest probability (yellow arrow) is deemed as the associated pixel of a radar point. To obtain labels for training R2P, our label generation module uses $\boldsymbol{f}()$ to compute velocities of all neighboring pixels, then calculates velocity error E_m by using the GT velocity $\dot{\boldsymbol{m}}_{GT}$, and finally obtains association probabilities of these neighbors based on E_m.

flow measurements in the camera to position and velocity measurements made by the radar.

3.3.1 Physical Configuration and Notation

The physical configuration of our camera and radar measurements is illustrated in Fig. 3.2(a). Three coordinate systems are shown: A and B specifying camera poses and R specifying a radar pose. The camera at B observes a 3D point $\boldsymbol{p}$. A short interval later, Δt, the point has moved to $\boldsymbol{q}$, the camera to A and the radar to R, and both the camera and radar observe the target point $\boldsymbol{q}$. These 3D points are specified by 4-dim homogeneous vectors, and when needed, a left-superscript specifies the coordinate system in which it is specified, e.g., ${}^{A}\boldsymbol{q}$ indicates a point relative to a coordinate system A. The target velocity, $\dot{\boldsymbol{m}}$, and camera velocity $\dot{\boldsymbol{c}}$ are specified by 3-dim vectors, again optionally with a left superscript to specify a coordinate system.

Coordinate transformations, containing both a rotation and translation, are specified by 4×4 matrices, such as ${}^{B}_{A}\boldsymbol{T}$, which transforms points from the left-subscript coordinate system to the left-superscript coordinate system. In this case we transform a point from A to B with:

$$ {}^{B}\boldsymbol{q} = {}^{B}_{A}\boldsymbol{T}\ {}^{A}\boldsymbol{q}. \tag{3.1} $$

Only the rotational component of these transformations is needed to transform velocities. For example, ${}^{A}\dot{\boldsymbol{m}}$ is transformed to ${}^{B}\dot{\boldsymbol{m}}$ by the 3×3 rotation matrix ${}^{B}_{A}\boldsymbol{R}$:

$$ {}^{B}\dot{\boldsymbol{m}} = {}^{B}_{A}\boldsymbol{R}\ {}^{A}\dot{\boldsymbol{m}}. \tag{3.2} $$

A vector with a right subscript, e.g., $\boldsymbol{p}_i$, indicates the i'th element of $\boldsymbol{p}$, while a right subscript of "1:3" puts the first 3 elements in a 3-dim vector. For a matrix, the right subscript indicates the row. Thus ${}^{B}_{A}\boldsymbol{R}_i$ is a 1×3 row vector containing its i-th row. A right superscript "$^{\mathsf{T}}$" is a matrix transpose.

The projections of points $\boldsymbol{p}$ and $\boldsymbol{q}$ are specified in either undistorted raw pixel coordinates, e.g., (x_q, y_q) or their normalized image coordinates (u_q, v_q) given by:

$$ u_q = (x_q - c_x)/f_x, \qquad v_q = (y_q - c_y)/f_y. \tag{3.3} $$

Here c_x, c_y, f_x, f_y are intrinsic camera parameters, while the right subscript of the pixel refers to the point being projected. Vectors for 3D points can be expressed in terms of the normalized image coordinates:

$$ {}^{A}\boldsymbol{q} = \begin{pmatrix} u_q d_q \\ v_q d_q \\ d_q \\ 1 \end{pmatrix} \quad \text{and} \quad {}^{B}\boldsymbol{p} = \begin{pmatrix} u_p d_p \\ v_p d_p \\ d_p \\ 1 \end{pmatrix}. \tag{3.4} $$

Here d_q and d_p are depths of points ${}^{A}\boldsymbol{q}$ and ${}^{B}\boldsymbol{p}$ respectively.

We assume dense optical flow is available that maps target pixel coordinates observed in A to B as follows:

$$ \text{Flow}\left((u_q, v_q)\right) \rightarrow (u_p, v_p). \tag{3.5} $$

Further, we assume the following are known: camera motion, ${}^{B}_{A}\boldsymbol{T}$, relative radar pose, ${}^{A}_{R}\boldsymbol{T}$, and intrinsic parameters.

3.3.2 Full-Velocity Radar Returns

The Doppler velocity measured by a radar is just one component of the three-component, full-velocity vector of an object point. Here our goal is to leverage optical flow from a synchronized camera to augment radar and estimate this full-velocity vector for each radar return.

3.3.2.1 Relationship of Full Velocity to Radial Velocity

The target motion from $\boldsymbol{p}$ to $\boldsymbol{q}$ is modeled as constant velocity, $\dot{\boldsymbol{m}}$, over time Δt, such that

$$\dot{\boldsymbol{m}} = \frac{\boldsymbol{q}_{1:3} - \boldsymbol{p}_{1:3}}{\Delta t}. \tag{3.6}$$

Our goal is to estimate the full target velocity, $\dot{\boldsymbol{m}}$. Radar provides an estimate of the target position, $\boldsymbol{q}$, but not the previous target location $\boldsymbol{p}$. Radar also provides the signed radial speed, $\dot{r}$, which is one component of $\dot{\boldsymbol{m}}$. In the nuScenes dataset $\dot{r}$ is given by:

$$\dot{r} = \hat{\boldsymbol{r}}^{\mathsf{T}} \dot{\boldsymbol{m}}. \tag{3.7}$$

Here $\hat{\boldsymbol{r}}$ is the unit-norm vector along the direction to the target ${}^{R}\boldsymbol{q}$. Note that this equation is coordinate-invariant, and could be equally written in A using ${}^{A}\hat{\boldsymbol{r}}$ and ${}^{A}\dot{\boldsymbol{m}}$. Now Eq. (3.7) is actually the egomotion-corrected Doppler speed. The raw Doppler speed, $\dot{r}_{raw}$, is the radial component of the *relative* velocity between target and sensor, $\dot{\boldsymbol{m}} - \dot{\boldsymbol{c}}$, and this constraint is given by:

$$\dot{r}_{raw} = \hat{\boldsymbol{r}}^{\mathsf{T}} (\dot{\boldsymbol{m}} - \dot{\boldsymbol{c}}), \tag{3.8}$$

where $\dot{\boldsymbol{c}}$ is the known ego-velocity. Either Eq. (3.7) or (3.8) can be used in our formulation, depending on whether $\dot{r}$ or $\dot{r}_{raw}$ is available from the radar.

3.3.2.2 Relationship of Full Velocity to Optical Flow

In solving the velocity constraints, we first identify the known variables. The radar measures ${}^{R}\boldsymbol{q}$, and transforming this we obtain ${}^{A}\boldsymbol{q} = {}^{A}_{R}\boldsymbol{T}\ {}^{R}\boldsymbol{q}$ which contains d_q as the third component. Image coordinates (u_q, v_q) are obtained by projection, and using optical flow in Eq. (3.5), we can also obtain the (u_p, v_p) components of ${}^{B}\boldsymbol{p}$. The key parameter we do not know from this is the depth, d_p, in B.

Next we eliminate this unknown depth from our constraints. Eq. (3.6) can be rearranged and each component expressed in frame B:

$$
{}^{B}\boldsymbol{p}_{1:3} = {}^{B}\boldsymbol{q}_{1:3} - {}^{B}_{A}\boldsymbol{R}\ {}^{A}\dot{\boldsymbol{m}}\Delta t, \tag{3.9}
$$

where the second term on the right is the transformation of the target motion into B coordinates. The third row of this equation is an expression for d_p:

$$
d_p = {}^{B}\boldsymbol{q}_3 - {}^{B}_{A}\boldsymbol{R}_3\ {}^{A}\dot{\boldsymbol{m}}\Delta t. \tag{3.10}
$$

Substituting this for d_p, and the components of ${}^{B}\boldsymbol{p}$ from Eq. (3.4), into the first two rows of Eq. (3.9), we obtain

$$
\begin{bmatrix} u_p({}^{B}\boldsymbol{q}_3 - {}^{B}_{A}\boldsymbol{R}_3\ {}^{A}\dot{\boldsymbol{m}}\Delta t) \\ v_p({}^{B}\boldsymbol{q}_3 - {}^{B}_{A}\boldsymbol{R}_3\ {}^{A}\dot{\boldsymbol{m}}\Delta t) \end{bmatrix} = \begin{bmatrix} {}^{B}\boldsymbol{q}_1 - {}^{B}_{A}\boldsymbol{R}_1\ {}^{A}\dot{\boldsymbol{m}}\Delta t \\ {}^{B}\boldsymbol{q}_2 - {}^{B}_{A}\boldsymbol{R}_2\ {}^{A}\dot{\boldsymbol{m}}\Delta t \end{bmatrix}, \tag{3.11}
$$

and rearrange to give two constraints on the full velocity:

$$
\begin{bmatrix} {}^{B}_{A}\boldsymbol{R}_1 - u_p\,{}^{B}_{A}\boldsymbol{R}_3 \\ {}^{B}_{A}\boldsymbol{R}_2 - v_p\,{}^{B}_{A}\boldsymbol{R}_3 \end{bmatrix} {}^{A}\dot{\boldsymbol{m}} = \begin{bmatrix} \left({}^{B}\boldsymbol{q}_1 - u_p\,{}^{B}\boldsymbol{q}_3\right)/\Delta t \\ \left({}^{B}\boldsymbol{q}_2 - v_p\,{}^{B}\boldsymbol{q}_3\right)/\Delta t \end{bmatrix}. \tag{3.12}
$$

3.3.2.3 Full-Velocity Solution

We obtain three constraints on the full velocity, ${}^{A}\dot{\boldsymbol{m}}$, from Eq. (3.12) and by converting Eq. (3.7) to A coordinates. Combining these we obtain:

$$
\begin{bmatrix} {}^{B}_{A}\boldsymbol{R}_1 - u_p\,{}^{B}_{A}\boldsymbol{R}_3 \\ {}^{B}_{A}\boldsymbol{R}_2 - v_p\,{}^{B}_{A}\boldsymbol{R}_3 \\ {}^{A}\hat{\boldsymbol{r}}^{\mathsf{T}} \end{bmatrix} {}^{A}\dot{\boldsymbol{m}} = \begin{bmatrix} \left({}^{B}\boldsymbol{q}_1 - u_p\,{}^{B}\boldsymbol{q}_3\right)/\Delta t \\ \left({}^{B}\boldsymbol{q}_2 - v_p\,{}^{B}\boldsymbol{q}_3\right)/\Delta t \\ \dot{r} \end{bmatrix}. \tag{3.13}
$$

Then inverting the 3×3 coefficient of ${}^{A}\dot{\boldsymbol{m}}$ gives a closed form solution for the full velocity:

$$
{}^{A}\dot{\boldsymbol{m}} = \begin{bmatrix} {}^{B}_{A}\boldsymbol{R}_1 - u_p\,{}^{B}_{A}\boldsymbol{R}_3 \\ {}^{B}_{A}\boldsymbol{R}_2 - v_p\,{}^{B}_{A}\boldsymbol{R}_3 \\ {}^{A}\hat{\boldsymbol{r}}^{\mathsf{T}} \end{bmatrix}^{-1} \begin{bmatrix} \left({}^{B}\boldsymbol{q}_1 - u_p\,{}^{B}\boldsymbol{q}_3\right)/\Delta t \\ \left({}^{B}\boldsymbol{q}_2 - v_p\,{}^{B}\boldsymbol{q}_3\right)/\Delta t \\ \dot{r} \end{bmatrix}. \tag{3.14}
$$

Recall in Fig. 3.1(a) the red/blue dashed lines show the velocity constraints from radar/flow. The solution of Eq. (3.14) is the full velocity that is consistent with both constraints. We note that this can handle moving sensors, although Fig. 3.1(a) shows the case of a stationary camera for simplicity. Further, if we set $\Delta t < 0$, Eq. (3.14) also applies to the case that the point shifts from $\boldsymbol{q}$ to $\boldsymbol{p}$ as the camera moves from A to B. And one limitation is that Eq. (3.14) cannot estimate full velocity for radar points occluded in the camera view, although we can typically identify those occlusions.

3.3.3 Image Pixels and Radar Points Association

Our solution for point-wise velocity in Eq. (3.14) assumes that we know the pixel coordinates (u_q, v_q) of the radar-detected point, ${}^{R}\boldsymbol{q}$. It appears straightforward to obtain this pixel correspondence by projecting a radar point onto the image using the known radar-image coordinate transformation, ${}^{A}_{R}\boldsymbol{T}$. We refer to this corresponding pixel as "raw projection". However, there are a number of reasons why raw projection of radar points into an image is inaccurate. Radar beamwidth typically subtends a few degrees and is large relative to a pixel, resulting in low resolution target location in both azimuth and elevation. Also, a radar displaced from a camera can often see behind an object, as viewed by the camera, and when these returns are projected onto an image they incorrectly appear to correspond to the foreground occluding object. Using flow from an occluder or an incorrectly associated object pixel may result in incorrect full-velocity estimation. To address these issues with raw projection, we train a neural network model, termed Radar-2-Pixel (R2P) network, to estimate associated radar pixels in the neighborhood of raw projection and identify occluded radar points. Similar models have been applied to image segmentation [39] and radar depth enhancement [76].

3.3.3.1 Model Structure

Our method estimates association probabilities (ranging from 0 to 1) between a moving radar point and a set of pixels in the neighborhood of its raw projection. The R2P network is an encoder-decoder structure with inputs and outputs of image resolution. Stored in 8 channels, the input data include image, radar depth map (with depth on raw projections), and optical flow. The output has

N channels, representing predicted association probability for N pixel neighbors. The association between the radar point, ${}^{A}\boldsymbol{q}$, and the k-th neighbor of raw projection (x, y) is stored in $A(x, y, k)$, where $k = 1, 2, ..., N$.

3.3.3.2 Ground Truth Velocity of Moving Radar Points

The nuScenes [10] provides the GT velocity of object bounding boxes. We associate radar hits on an object to its labeled bounding box, and assign the velocity of the box to its associated radar points. The association is determined based on two criteria: 1) in radar coordinates, the distance between radar points and associated box is smaller than a threshold T_d; and 2) the percentage error between the radial velocity of a radar point and the radial component of the velocity of associated box is smaller than a threshold T_p.

3.3.3.3 Generating Association Labels

We can project a radar point expressed in corresponding camera coordinates, ${}^{A}\boldsymbol{q}$, to pixel coordinates (u_q, v_q), but as mentioned before, often this image pixel does not correspond to the radar return. Our proposed solution is to search in a neighboring region around (u_q, v_q) for a pixel whose motion is consistent with the radar return. This neighborhood search is shown in Fig. 3.2. If a pixel is found, then we correct the 3D radar location ${}^{A}\boldsymbol{q}$ to be consistent with this pixel, otherwise we mark this radar return as occluded.

We learn this radar-to-pixel association and correction by training the R2P network. We generate true association score between a radar point and a pixel according to the compatibility between the true velocity and the optical flow at that pixel: high compatibility indicates high association. To quantify the compatibility, assuming a pixel is associated with a radar point, we compute a hypothetical full velocity for the radar point by using the optical flow of that pixel according to Eq. (3.14). The flow is considered compatible if the hypothetical velocity is close to the GT velocity. Specifically, the hypothetical velocity can be computed as

$$ {}^{A}\dot{\boldsymbol{m}}_{est}(x, y, k) = \boldsymbol{f}\left(\breve{u}_q, \breve{v}_q, \breve{u}_p, \breve{v}_p, d_q, \dot{r}, {}^{B}_{A}\boldsymbol{T}, {}^{A}_{R}\boldsymbol{T}\right), \tag{3.15} $$

where $k = 1, \cdots, N$, $\boldsymbol{f}(\cdot)$ is the function to solve full velocity via Eq. (3.14), and (x, y) is the raw

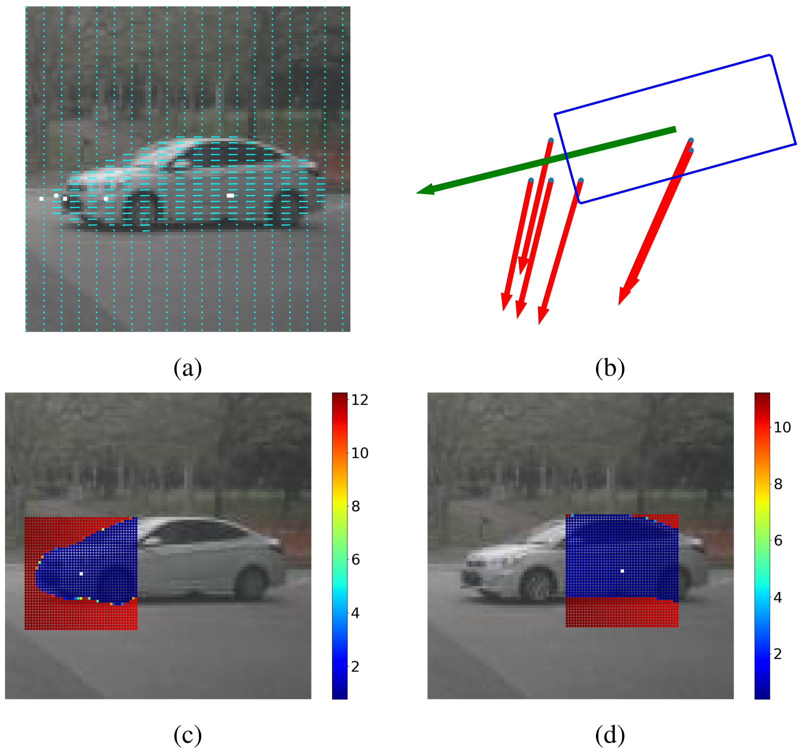


Figure 3.3 (a) Optical flow; (b) BEV of GT bounding box, radial velocity (red), and GT velocity (green); (c) and (d) show E_m, computed by using Eq. (3.16), for two radar projections (white square) over 41×41 pixel regions, respectively. For radar hits reflected from the vehicle, E_m is small for neighboring pixels on the car and large on the background.

projection of the radar point. Note that $\breve{u}_q = u_q\,[x + \Delta x(k), y + \Delta y(k)]$, $\breve{v}_q$ is defined similarly, and $[\Delta x(k), \Delta y(k)]$ is the coordinate offset from raw projection to the k-th neighbor. Using flow, Eq. (3.5), we obtain $(\breve{u}_p, \breve{v}_p)$ from $(\breve{u}_q, \breve{v}_q)$.

Second, we calculate the L_2 norm of errors between ${}^{A}\dot{\boldsymbol{m}}_{est}(x, y, k)$ and GT velocity ${}^{A}\dot{\boldsymbol{m}}_{GT}(x, y)$ by

$$E_m(x, y, k) = \|{}^{A}\dot{\boldsymbol{m}}_{est}(x, y, k) - {}^{A}\dot{\boldsymbol{m}}_{GT}(x, y)\|_2. \tag{3.16}$$

Fig. 3.3 shows examples of E_m for two radar hits on a car.

Finally, we transform E_m to an association score with

$$L(x, y, k) = e^{-\frac{E_m^2(x,y,k)}{c}}, \tag{3.17}$$

where L is used as a label for association probability between a radar and its k-th neighbor. Note that L increases with decreasing E_v, and c is a parameter adjusting the tolerance of velocity errors when converting errors to association. We use the cross entropy loss to train the model.

3.3.3.4 Estimate Association and Identify Occlusion

With a trained model, we can estimate association probability between radar points and N pixels around their raw projections (x, y), i.e., $A(x, y, k)$. Among the N neighbors, the radar return velocity may be compatible with a number of pixels, and we select the pixel with the maximum association, A_{max}, as the neighbor ID k_{max}:

$$k_{max} = \arg\max_{k}[A(x, y, k)]. \tag{3.18}$$

If A_{max} is equal or larger than a threshold T_a, we estimate the associated pixel as $[x + \Delta x(k_{max})$, $y + \Delta y(k_{max})]$. Otherwise there is no associated pixels in the neighborhood, and an occlusion is identified.

3.4 Experimental Results

3.4.1 Comparison of Point-wise Full Velocity

To the best of our knowledge, there is no existing method estimating *point-wise* full velocity for radar returns. Thus, we use point-wise radial velocity from raw radar returns as the baseline to compare with our estimation. We extract data from the nuScenes Object Detection Dataset [10], with 6432, 632, and 2041 samples in training, validation, and test set, respectively. Each sample consists of a radar scan and two images for optical flow computation, i.e., one image synchronizing with the radar and the other is a neighboring image frame. The optical flow is computed by the RAFT model [129] pre-trained on KITTI [27]. The R2P network is an U-Net [88, 108] with five levels of resolutions and 64 channels for intermediate filters. The neighborhood skips every other pixel, and its size (in pixels) is (left: 4, right: 4, top: 10, bottom: 4) and an example of the neighborhood is illustrated in Fig. 3.2(b). The threshold of association scores T_a is 0.3. Parameters associating radar points with GT bounding box are set as $T_d = 0.5$m and $T_p = 20\%$. Parameter c in Eq. (3.17) is 0.36. To obtain GT point-wise velocity, based on the criteria in Sec. 3.3.3.2, we first associate moving radar points to GT detection boxes, whose GT velocity is assigned to associated points as their GT velocity. The GT velocity of bounding boxes is estimated from GT

Mean Error (STD) (m/s)	Ours (R2P Network)	Ours (Raw Projection)	Baseline
Full Velocity	**0.433** (**0.608**)	0.577 (1.010)	1.599 (2.054)
Tangential Comp.	**0.322** (**0.610**)	0.472 (1.024)	1.536 (2.083)
Radial Comp.	0.205 (0.196)	0.205 (0.196)	0.205 (0.196)

Table 3.1 Comparison of point-wise velocity error of our methods and the baseline (raw radial velocity).

center positions in neighboring frames with timestamps.

Tab. 3.1 shows the average velocity error for moving points. The proposed method achieves substantially more accurate velocity estimation than the baseline. For instance, the error of our tangential component is only 21% of that of the baseline. We also have much smaller standard deviation, indicating more stable estimates. In addition, we list in Tab. 3.1 velocity error of our method using raw radar projection for radar-camera association. Results show that, compared with using raw projection, using R2P network achieves higher estimation accuracy. Fig. 3.4 illustrates qualitative results of our point-wise velocity estimation.

3.4.2 Comparison of Object-wise Velocity

Although there are no existing methods for point-wise velocity estimation for radar, a related work, CenterFusion [91], estimates *object-wise* full velocity via object detection with image and radar inputs. To fairly compare with CenterFusion, we convert our point-wise velocity to object-wise velocity. Specifically, we use the average velocity of radar points associated with the same detected box as our estimate of object velocity. Points are associated with detected boxes according to distance. Note the point-wise velocity to object-wise velocity conversion is straightforward for comparison purposes, and there would be more advanced approaches to integrate point-wise full velocities in a detection network, which is beyond the scope of this work. Tab. 3.2 shows that with our estimated full velocity, the velocity estimation for objects is significantly improved.

3.4.3 Velocity Estimation Error for Different Depths and α

This experiment extends the evaluation of point-wise velocity estimation discussed in Section 3.4.1. In Fig. 3.5, each heat map shows point-wise velocity error under different depth ranges,

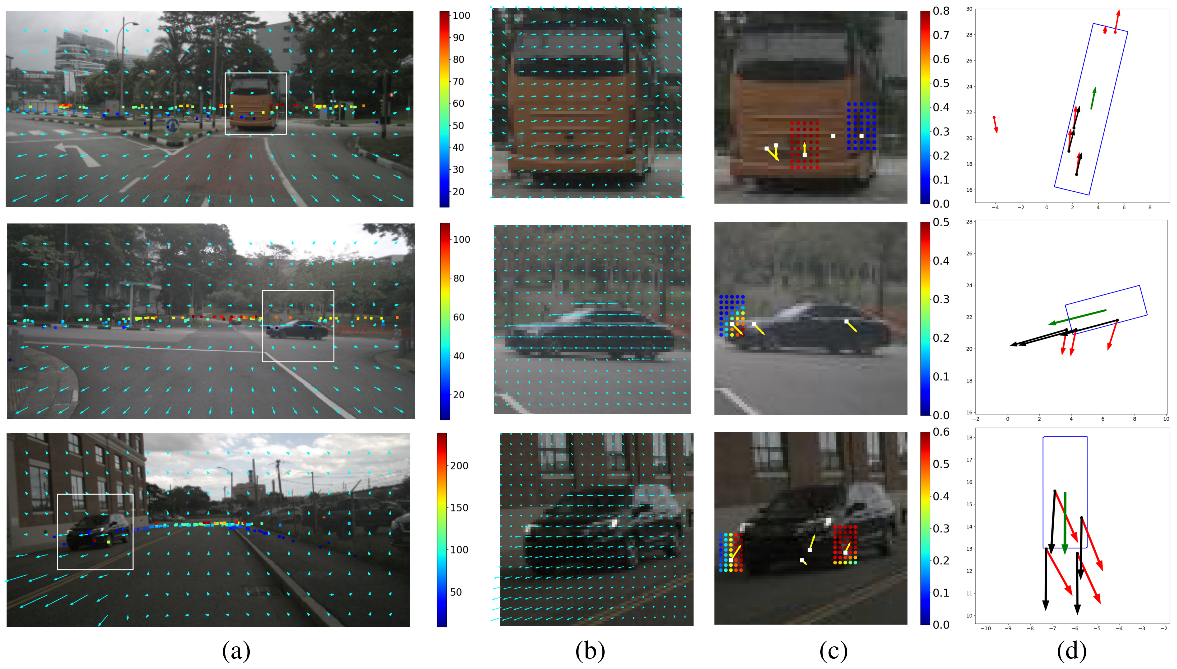
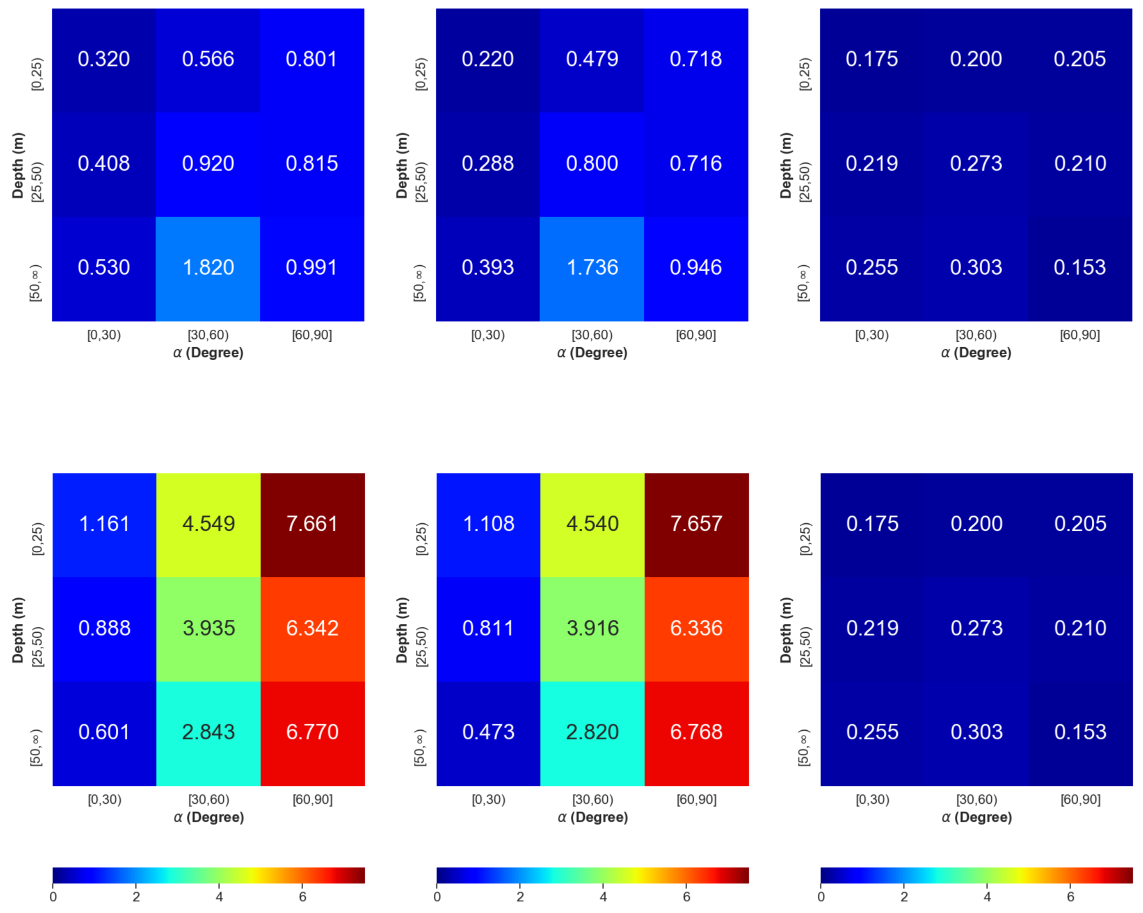


Figure 3.4 Visualization of point-wise velocity estimation: (a) depth of all measured radar returns as well as flow, (b) optical flow in the white box region, (c) association scores around the selected radar projections as well as predicted mapping from raw radar projections to image pixels (yellow arrow), and (d) radial velocity (red), estimated full velocity (black), and GT velocity (green) in BEV.

Methods	Error (m/s)
Ours	**0.451**
CenterFusion [91]	0.826

Table 3.2 Comparison of object-wise velocity errors. For a fair comparison we inherit the same set of detected objects from [91].

i.e., $[0, 25)$, $[25, 50)$, and $[50, \infty)$ meters as well as various α ranges, i.e., $[0, 30)$, $[30, 60)$, and $[60, 90]$ degrees, where α is the angle between actual moving direction and radial direction of a radar point and ranges from 0 to 90 degrees. Results of the proposed method and baseline are show in the first row and second row, respectively. The baseline (second row), with only radial measurement, suffers from large α since the the actual moving direction is very different from radial direction under large α. The proposed method outperforms the baseline in all depth and α ranges for full velocity estimation.



(a) Full Velocity Error (b) Tangential Component Error (c) Radial Component Error

Figure 3.5 Comparison of average error (in meters) of point-wise velocity estimates by the proposed method (first row) and baseline (second row). Columns 1, 2, and 3 are error of full velocity, tangential component, and radial component, respectively. Each heat map shows the error for radar points in different depth and α ranges, where α is the angle between full velocity and radial direction.

3.4.4 Radar Point Accumulation

Accumulating radar points over time can overcome the sparsity of radar hits acquired in a single sweep, achieving dense point cloud for objects and thus allowing techniques designed for processing lidar points to be applicable for radar. The point-wise velocity estimate makes it possible to compensate the motion of dynamic objects appearing in a temporal sequence of measurements for accumulation. Specifically, for a moving radar point (with estimated velocity $\dot{\boldsymbol{m}}$) in a previous frame i captured at time t_i, its motion from t_i to the time at the current frame, t_0, can be compensated by,

$$\boldsymbol{p_0} = \boldsymbol{p_i} + \dot{\boldsymbol{m}}(t_0 - t_i), \tag{3.19}$$

where $\boldsymbol{p_i}$ and $\boldsymbol{p_0}$ are the radar point coordinates at t_i and t_0 in radar coordinates of t_i. Then $\boldsymbol{p_0}$ is transformed to current radar coordinates by known egomotion from t_i to t_0.

Qualitative results. Fig. 3.6 shows accumulated points of moving vehicles in radar coordinates. For comparison, we show accumulated radar points compensated by our estimated full velocity, compensated with radial velocity (baseline) and without motion compensation. Compared with the baseline and no motion compensation, our accumulated points are more consistent with the GT bounding boxes.

Quantitative results. To quantitatively evaluate the accuracy of radar point accumulation, we use the mean distance from accumulated points (of up to 25 frames) to their corresponding GT boxes as the accumulation error. This distance for points inside the box is zero, and outside it is the distance from the radar point to the closest point on the box's boundary. In Fig. 3.7, we compare the accumulation for our method, the baseline, and accumulation without motion compensation. While error increases with the number of frames for all methods, our method has the lowest rate of error escalation.

Application of pose estimation. To demonstrate the utility of accumulated radar points for downstream applications, we apply a pose estimation method, i.e., BoxNet [92], on the accumulated 2D radar points via our full velocity and radial velocity (baseline), respectively. BoxNet takes pre-segmented 2D point clouds of an object as input and predicts a 2D bounding box with parameters as center position, length, width, and orientation. We use accumulated radar points of 5702, 559, and 2001 moving vehicles with corresponding GT bounding boxes as training, validation, and test data, respectively. Tab. 3.3 shows our accumulated radar achieves higher accuracy than the baseline.

3.4.5 Inference Time

The pipeline of our full velocity estimation includes three major components, optical flow computation, radar-camera association estimation, and closed-form solution of full velocity. The

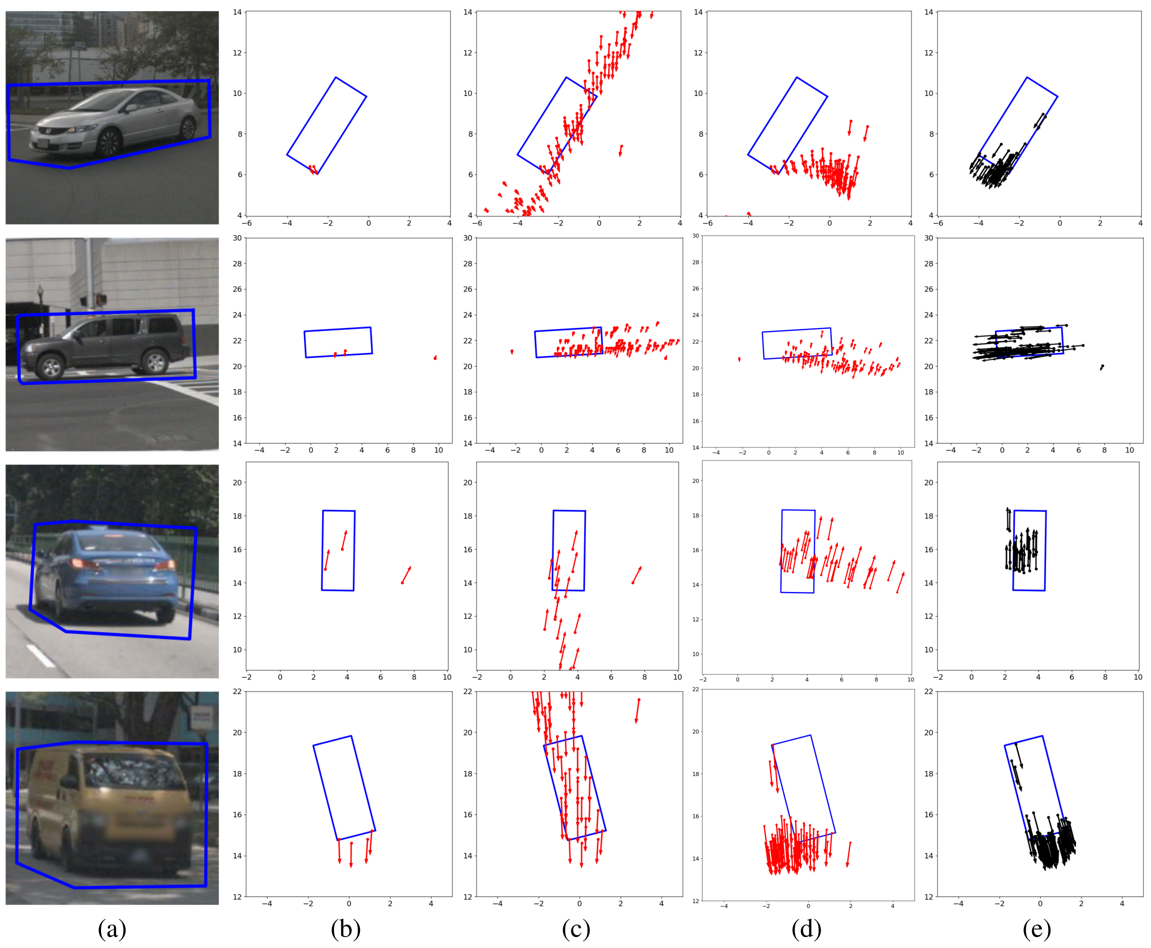


Figure 3.6 Moving radar points are plotted with point-wise radial (red) and full (black) velocity, including image with bounding box (a), single-frame radar points in BEV (b), accumulated radar points from 20 frames without motion compensation (c), with radial velocity based compensation (d), and with our full-velocity based compensation (e). Our accumulated points are tightly surrounding the bounding box, which will benefit downstream tasks such as pose estimation and object detection.

time used by each component per frame is listed in Table 3.4. Our computational platform includes Intel Core i7-8700 CPUs and a NVIDIA GeForce RTX 2080 Ti GPU. The proposed closed-form solution achieves highly efficient computation. Note the computational cost of optical flow can be improved by limiting the region of flow computation to areas with radar projections.

3.5 Summary

A drawback of Doppler radar has been that it provides only the radial component of velocity, which limits its utility in object velocity estimation, motion prediction, and radar return accumulation. This chapter addresses this drawback by presenting a closed-form solution to the full velocity

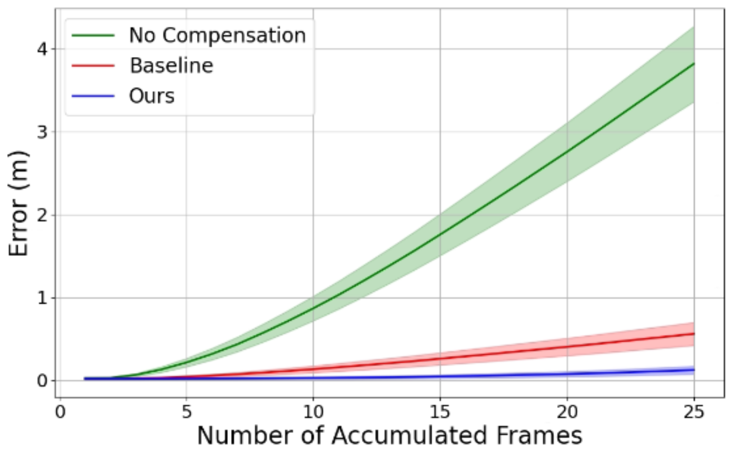


Figure 3.7 Error comparison when accumulating radar points from increasing number of frames. The lines represent mean error and shaded area ±0.1× STD. Our full velocity based accumulation outperforms the ones with radial velocity, or no compensation.

Metric	Ours	Baseline
Center Error (m) ↓	**0.834**	0.997
Orientation Error (degree) ↓	**6.873**	7.517
IoU ↑	**0.546**	0.462

Table 3.3 Comparison of pose estimation performance: average error in center and orientation as well as Intersection over Union (IoU), by using BoxNet [92] on radar points accumulated using our velocity and the radial velocity as a baseline.

Components	Time per Frame (s)
Optical Flow [129]	4.47×10^{-1}
Radar-camera Association	2.58×10^{-3}
Closed-form Velocity Computation	2.49×10^{-4}

Table 3.4 Computational time of three components in the method pipeline.

of radar returns. It leverages optical flow constraints to upgrade radial velocity into full velocity. As part of this work, we use GT bounding-box velocities to supervise a network that predicts association corrections for the raw radar projections. We experimentally verify the effectiveness of our method and demonstrate its application on motion compensation for integrating radar sweeps over time.

This method developed here may apply to additional modalities such as full-velocity estimation from Doppler lidar and cameras.

CHAPTER 4

RADIANT: RADAR-IMAGE ASSOCIATION NETWORK FOR 3D OBJECT DETECTION

As a direct depth sensor, radar holds promise as a tool to improve monocular 3D object detection, which suffers from depth errors, due in part to the depth-scale ambiguity. On the other hand, leveraging radar depths is hampered by difficulties in precisely associating radar returns with 3D estimates from monocular methods, effectively erasing its benefits. This chapter proposes a fusion network that addresses this radar-camera association challenge. We train our network to predict the 3D offsets between radar returns and object centers, enabling radar depths to enhance the accuracy of 3D monocular detection. By using parallel radar and camera backbones, our network fuses information at both the feature level and detection level, while at the same time leveraging a SOTA monocular detection technique without retraining it. Experimental results show significant improvement in mean average precision and translation error on the nuScenes dataset over monocular counterparts. This chapter was previously published as [73].

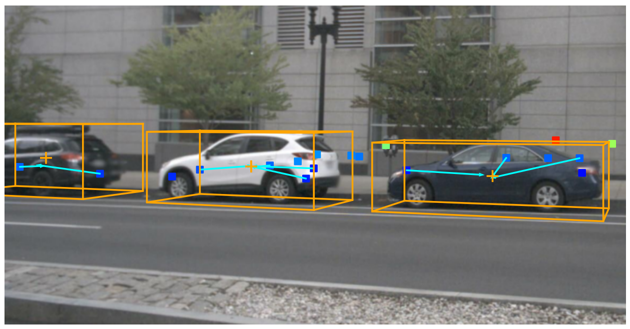
4.1 Introduction

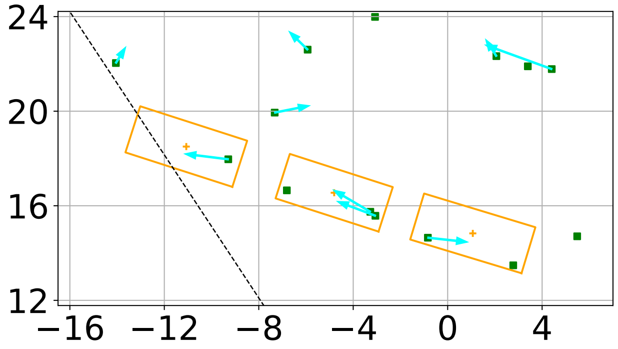
Three-dimensional object detection is a core vision problem where the task is to infer 3D position, orientation, and classification of objects in a scene. Applications that rely on this include robotics [111], gaming [106], and automotive safety [120]. In the latter application, ADAS can move beyond automated braking and use precise location and orientation of nearby vehicles and other objects to perform collision avoidance maneuvers. However, a key limiting factor is the relatively poor accuracy of 3D object detection, both in current systems and in SOTA methods that rely on widely used sensors, namely cameras [77, 99] and radars [149]. Thus, there is a significant need for improved 3D object detection, which is the goal of this chapter.

Image-based object detection achieves high 2D accuracy [7]; for instance, DD3D [99] achieves 94% 2D mean average precision (mAP) for cars on KITTI [28]. However, the performance of these same SOTA methods drops precipitously on 3D object detection with DD3D [99] only achieving 16.9% 3D mAP of cars on KITTI. The lower 3D performance of monocular detection comes largely from poor depth estimation [46, 82]. This is expected as image projection removes depth, and recovering object depth from an image suffers from the depth scale ambiguity [128], and is therefore error-prone. Indeed, when [146] uses precise depth from lidar, mAP of cars increases to over 90%. Typically lidar is a specialized sensor that is relatively expensive [47] and only available on a small fraction of vehicles. On the other hand, radar is small [61], inexpensive, and widely available on existing vehicles so we choose to use radar for the broader impact. Thus, this chapter explores the fusion of direct depth measurements available from the radar with a 3D monocular detector.

The choice to use automotive radar rather than lidar presents several challenges. At first glance, one might consider using a similar 3D object-detector on radar points as for lidar [139]. However, that does not work because the radar data has completely different characteristics from the lidar point cloud [139]. First, each radar-point sweep is much sparser than a typical lidar measurement in azimuth [149] and has a single row in elevation, leaving radar coordinates without height information. Thus, differing from lidar, radar cannot acquire accurate shapes with dense point clouds. Additionally, the radar point positions have significant azimuth errors [149] and are much less accurate [139] as measurements of object surfaces. Next, for each radar scan there is often only a single radar return at longer ranges and sometimes no radar returns for small objects. This sparsity makes 3D object detection solely based on radar points difficult. Thus, rather than detection-level fusion, our approach uses feature-level fusion to augment radar points and these augmented radar points to refine monocular object depths. Finally, since the widely-used KITTI dataset [28] does not include radar data, we use the nuScenes dataset [10] for experiments.

The association and fusion of image and radar modalities is possible at input-level, feature-level, or even at detection-level. However, any such association must address the missing height, imprecise angular location of radar returns, and the observation that occluded portions of objects also return radar points [76]. Rather than using inaccurate radar projections on image for association, this chapter uses a neural network to explicitly predict point-wise 3D object centers (see Fig. 4.1). Predicting these point-wise object centers from a trained network allows us to use radar to correct


(a) Pixel Offset



(b) BEV Offset

Figure 4.1 Predicted radar-offsets (cyan arrows) from radar points (dots) to object centers (orange plus) in (a) pixel space and (b) BEV in meters. RADIANT trains a network to predict these offsets and improves monocular 3D detection. Orange boxes represent the GT bounding boxes, and dashed lines denote borders of the camera field of view.

depth errors of monocular detection, which results in a new SOTA for radar-camera 3D object detection.

This chapter presents a radar-camera fusion method named RADar-Image Association NeTwork (RADIANT) for 3D object detection, including the following contributions:

- Our method enhances radar returns to obtain 3D object center detections from each radar return.
- We achieve camera-radar association at the detection level using the enhanced radar locations.
- Our architecture can leverage multiple different pre-trained SOTA monocular methods.
- We improve monocular object depth estimates by fusing enhanced radar depths and achieve new improved SOTA performance on nuScenes.

4.2 Related Work

Monocular Detection Differing from lidar-based detection [117], monocular 3D detection is widely applied for its low cost and simple configuration [6]. Researchers have been improving detection performance via upgrading detection frameworks [68, 141], losses [120], and non-maximum suppression [47] as well as performing joint detection and 3D reconstruction [66]. To reduce 2D to 3D ambiguities, various strategies have been developed, e.g., pseudo-lidar [80, 81, 99, 121, 138], novel convolutions [20] and backbones [46], considering camera geometry [161], using shape models [11, 70], and leveraging videos [8].

Radar-Camera Fusion Radar has been fused with lidar and camera. Radar-lidar fusion has been utilized in 3D object detection [149] and object tracking [116] as radar complements lidar in long range and motion (i.e., Doppler velocity) measurements. Nevertheless, most works combine radar with camera for advantages listed in the Introduction section. There has been a surge in research of radar-camera fusion recently since the release of new autonomous driving datasets [3, 10, 21, 95, 118, 140] with images and radar data collected. The radar data are in raw data formats [3, 21, 95, 140] (such as Range-Azimuth-Doppler format [83]) or processed formats (i.e., radar point clouds [10, 118]). Raw radar formats contain denser measurements while point clouds are sparser but have less noise. Algorithms are designed according to specific radar formats used. In this chapter, we use the radar point cloud format from the nuScenes dataset [10]. The goals for radar-camera fusion include depth completion [53, 76], full-velocity estimation [75], object tracking [89], and object detection [90, 91].

A focus of radar-camera fusion is 2D object detection on images [57, 90, 118, 148], where projected radar points are used as extra features or candidates for potential objects. For example, method [90] maps radar detections to the image coordinate system and generates anchor boxes for each mapped radar detection point. Here radar plays an important role when the image is not clear because of darkness or long distances. However, the research on 3D object detection via radar-camera fusion is still at an early stage with few publications. One top-performing work is CenterFusion [91], which directly combines monocular detections with raw radar points in the neighborhood followed by a regression head to refine the depth estimate. In light of the significant gap between radar points and object centers, we believe this apples and oranges combination limits the utility of radar depths in CenterFusion. Our method, on the other hand, estimates the 3D object center for each radar return. This geometric correction performed by the radar branch then enables our fusion module to effectively combine multiple estimates of 3D center points from radar and camera for better 3D detection.

4.3 Background and Definitions

Radar-Positives. A key component of training a detection network involves the choice of candidates and the rule to construct positives. FCOS3D [136] treats each pixel as an object candidate, and positive pixels for training detections are defined as small regions in the neighborhood of projected centers of 3D objects. To train the network to predict radar offsets, RADIANT follows the same strategy but now treats each radar return as an object candidate and considers the associated projected radar pixel as positive. Association of radar returns with an object is easy if the radar projection always falls on the object. However, radar returns returned by an object are sometimes outside the object bounding box due to measurement errors [149]. Moreover, the nuScenes dataset also does not provide GT object labels for radar points. Thus, we carry out the radar-object association for training according to the position and velocity consistency. If the distance between an outside radar point and the object GT bounding box is smaller than a threshold and the projection of GT object velocity on radial direction is close to the Doppler velocity from the radar point, we associate the radar pixel with the object and consider that radar pixel as positive.

Radar Depth Offset. Radar depth offset $\Delta\hat{z}^r$ is the depth difference between the 3D center of its associated object z and a positive radar point z^r, and is thus given by

$$\Delta\hat{z}^r = z - z^r. \tag{4.1}$$

As mentioned, this residual depth is a result of radar measurement error and relative position of radar hit on object surfaces. We infer the residual depth from both radar and image information around.

4.4 RADIANT

Monocular 3D detection without depth as input suffers from inaccurate depth estimation [82, 134], especially for far objects. Our goal is to *upgrade* 3D camera detections with more accurate depth from radar with minimal changes to the image detection pipeline. To achieve this we address two difficulties. (1) While radar returns typically provide more precise depth than camera-based detections, which part of an object they measure can be difficult to determine as it may be the front

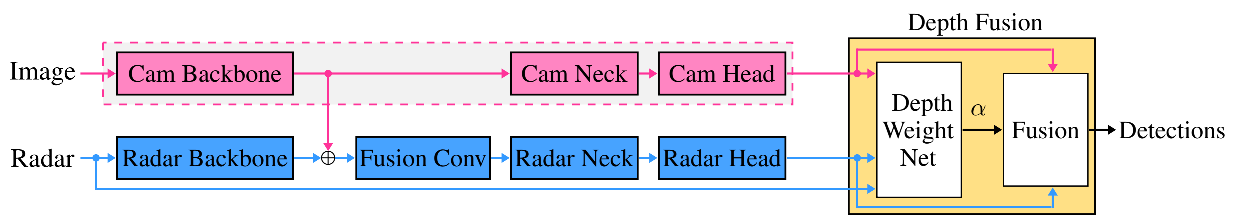


Figure 4.2 Overview of RADIANT. RADIANT architecture has two parallel input branches, an image branch and a radar branch that both operate in the image space. More details of these branches are shown in Fig. 4.3. The depth fusion module combines depth estimates from both the camera and radar heads to obtain a refined overall depth for each detection.

surface or an internal or occluded point on the object. This unknown offset can add significant error to a radar depth estimate when combined with an image-based detection. (2) The radar return point can sometimes be outside the true object bounding box, and this complicates the association between radar points and objects.

Our solution is to train a radar-focused network to predict the unknown offsets between radar points and object centers. These offsets include both an image-plane offset to the projection of the 3D center, and a depth offset to the object center. Assuming these offsets are correctly estimated, the association between radar points and objects becomes much easier. Furthermore, since they predict the object center, we use the offsets to correct the object center depth.

Our architecture consists of two branches and a fusion block as shown in Fig. 4.2. The upper branch is a monocular 3D detection network, such as FCOS3D [136], which remains unchanged, while the lower branch is the radar detection branch. The image branch predicts object centers and pixel offsets to these centers in the image plane. Now radar points are projected into the image plane, providing coarse alignment with image detections, and processed with the radar backbone network. Since radar points are sparse and lack contextual information, we bring features from the image backbone into the radar network providing image-plane spatial information to the radar pipeline. Then the radar neck and head portions perform the radar-based detection in the image space, but only at radar pixels, i.e., the projection of radar points onto the image plane, at five resolutions to maintain consistency with FCOS3D. We then fuse these two sets of detections through a *depth fusion* module that updates the predictions' depths using a confidence score.

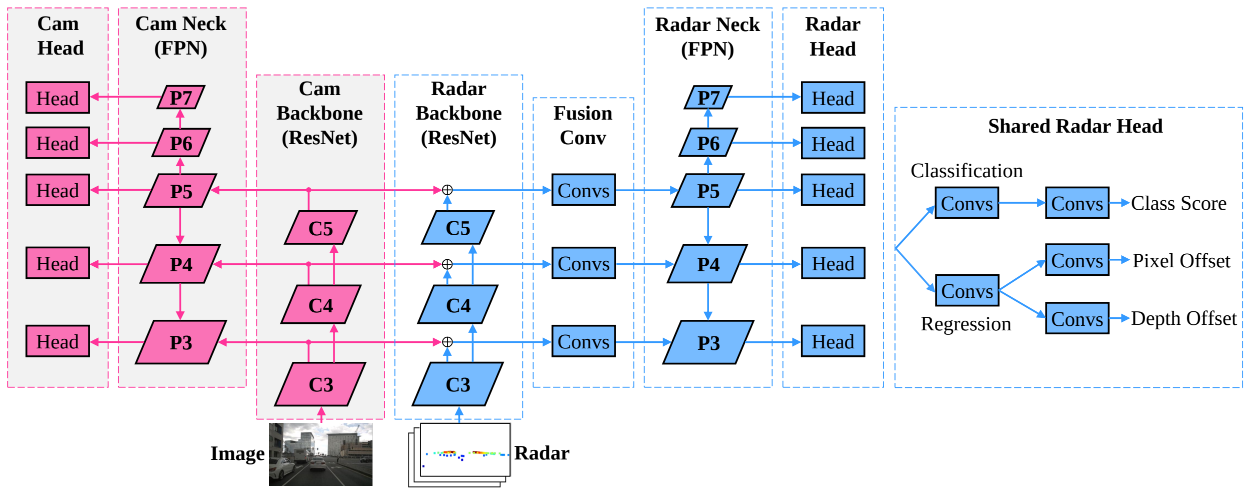


Figure 4.3 RADIANT Architecture Details. RADIANT architecture includes two parallel branches. On the left (in magenta) is the unchanged monocular detection pipeline. On the right (in blue), the radar network processes image-projected radar points and borrows features from the monocular network to predict offsets to the radar pixels and to their depths in the shared radar head. C3 to C5 denote feature maps from level 3 to 5 in the backbone, and P3 to P7 represent feature maps from level 3 to 7.

4.4.1 Radar Branch

One of the goals in designing RADIANT is to make minimal changes to the existing monocular architectures. Therefore, RADIANT builds seamlessly on top of an existing SOTA monocular network, such as FCOS3D [136], which can be separately trained. While it would be natural to simply augment color images with additional radar channels and retrain the image network, we found this ineffective, with the resulting network unable to benefit from the radar data. Instead, we use a separate backbone, ResNet-18 [31], for the radar processing and freeze the image branch while training the radar branch (see Fig. 4.3).

The inputs to the radar branch are in image coordinates with values on radar projections, consisting of radar depth, radar BEV coordinates, Doppler velocity, and a mask for radar pixels. Except for the backbone and losses, the majority of the radar branch is similar to the image branch, with a radar backbone processing inputs and generating radar features, which are concatenated with image features at three resolution levels, then go through an independent neck consisting of a Feature Pyramid Network (FPN) [64] and the radar heads. The radar branch outputs data in the same space, i.e., five levels of image resolutions, and uses the same classification and regression losses. This makes the radar branch outputs compatible with the image branch outputs. RADIANT performs image and radar fusion at two stages: feature-level fusion from the backbone and detection-level fusion after the heads.

4.4.2 Radar Heads

Radar heads take in fused radar and image features of five resolutions and predict class scores as well as relative positions to the object center, namely the depth offset and the pixel offset. The radar head ignores prediction of object sizes as radar points are too sparse to reveal shape information. We build radar heads independently from monocular camera heads because of the differences in what they predict as well as the definition and positions of positive pixels: (1) camera heads estimate the full depth of objects while radar heads estimate offset depth with respect to radar measured depth; (2) positive camera pixels are only a 3×3 region at the object center while positive radar pixels may be further away from the center. Nevertheless, we use the same strategy of FCOS3D [136] to assign positive pixels to different resolution levels so that larger objects on images are detected at lower resolutions. In summary, the radar head is complementary to the camera head with more accurate position estimation for those objects with radar hits on them.

4.4.3 Depth Fusion Module

Detection heads generate detection candidates for pixels classified as positive. As the depths from radar and camera heads are predicted independently, to take advantage of both depths, they are fused in the Depth Fusion module as follows. First, radar pixels are associated with monocular detection candidates. This is straightforward as the radar head outputs depth and pixel offsets, and these can be matched to 3D centers of detection candidates. Second, a confidence-based weight is predicted for each radar pixel enabling its depth to be combined with the monocular-based depth. Note that we focus on using radar to enhance only depth prediction since the depth accuracy is a primary merit of radar compared to camera, while other aspects of detection such as object size and image position are unlikely to be improved by radar. This fusion process is described in detail as follows.

Radar-Camera Association. RADIANT outputs two sets of detections from each head: $\{\mathbf{b}_i^c\}_i$ from camera head and $\{\mathbf{b}_j^r\}_j$ from radar head, where i and j represent indices of detections from camera and radar head, respectively. Typical outputs from the camera head are box proposals for detection $\mathbf{b}_i^c$, consisting of projected center $\left(\hat{u}_i^c, \hat{v}_i^c\right)$, depth $\hat{z}_i^c$, classification index $\hat{\mathcal{Y}}_i^c$, detection score $\hat{\sigma}_i^c$, dimensions of the 3D box, orientation, and deltas for 2D detections. The outputs from the radar head are box deltas for detection which contain the pixel offsets $(\Delta\hat{u}_j^r, \Delta\hat{v}_j^r)$ from the projected radar point (u_j^r, v_j^r), depth offset $\Delta\hat{z}_j^r$ of the object center from the radar depth z_j^r, radar classification index $\hat{\mathcal{Y}}_j^r$, and detection score $\hat{\sigma}_j^r$.

Since the radar head does not affect 2D detection, 3D dimensions and orientations, we omit these variables in the camera outputs $\mathbf{b}_i^c$ in the subsequent paragraphs for brevity. In other words, we only specify the relevant portion of $\mathbf{b}_i^c$ vector in the following text. We now write these two set of detections as

$$\begin{aligned} \mathbf{b}_i^c &= \left(\hat{u}_i^c, \hat{v}_i^c, \hat{z}_i^c, \hat{\mathcal{Y}}_i^c, \hat{\sigma}_i^c\right), \\ \mathbf{b}_j^r &= \left(\Delta\hat{u}_j^r, \Delta\hat{v}_j^r, \Delta\hat{z}_j^r, \hat{\mathcal{Y}}_j^r, \hat{\sigma}_j^r\right). \end{aligned} \tag{4.2}$$

We then filter the box proposals from both camera and radar as follows. The camera head outputs a maximum of $1,000$ boxes with the highest scores on each level and with $\hat{\sigma}_i^c > T_c$ where T_c denotes the minimum threshold for the box to be valid. We employ a similar procedure for radar detection candidates and consider radar projections with $\hat{\sigma}_j^r > T_r$. After the filtering step, we have good box proposals from the two modalities.

We now have the radar pixel $\left(u_j^r, v_j^r\right)$ and corresponding depth/pixel offsets which we use for the radar-camera association. We calculate the projected centers $\left(\hat{u}_j^r, \hat{v}_j^r\right)$ and depths $\hat{z}_j^r$ of the boxes as

$$\begin{aligned} \left(\hat{u}_j^r, \hat{v}_j^r\right) &= \left(u_j^r, v_j^r\right) + \left(\Delta\hat{u}_j^r, \Delta\hat{v}_j^r\right), \\ \hat{z}_j^r &= z_j^r + \Delta\hat{z}_j^r. \end{aligned} \tag{4.3}$$

We consider a camera proposal is associated with the radar proposal if the predicted class labels of the two modalities match, and the projected centers and the depths are close to each other. In other words, we take a camera proposal $\mathbf{b}_i^c$ and iterate through the radar proposals $\mathbf{b}_j^r$ and a match is found if the following conditions are satisfied:

$$\hat{\mathcal{Y}}_i^c = \hat{\mathcal{Y}}_j^r \tag{4.4}$$

$$|| \left(\hat{u}_i^c, \hat{v}_i^c\right) - \left(\hat{u}_j^r, \hat{v}_j^r\right) ||_2 < T_p \tag{4.5}$$

$$|\hat{z}_i^c - \hat{z}_j^r| < T_d, \tag{4.6}$$

where T_p and T_d denote the distance threshold on pixels and the depth, respectively. We use different thresholds for projected centers and depth as the errors in the two spaces are different. Thus, we obtain a set of potential corresponding radar detections $\{\mathbf{b}_j^r\}$ for each $\mathbf{b}_i^c$. The complexity of matching is $\mathcal{O}\left(MN\right)$, where M and N are the number of camera and radar proposals. The score based thresholding limits the running time of the downstream matching algorithm.

Depth Weighting Network. The high-level idea of RADIANT is to update the monocular depth of the boxes with better depth from radar. Although the radar depth is generally more accurate than the camera depth, the camera depth may be better for nearby objects because of the richer semantic information. Thus, always preferring radar depth over camera is not beneficial. In other words, there should be a better *weighting* mechanism between the two depths. Hence, to better determine association and depth weights for a potential camera-radar detection pair extracted with Eqs. (4.4) to (4.6), we train another depth weighting network (DWN) to output relative confidence in radar and camera depths.

The DWN is a 4-layer multilayer perceptron (MLP) that outputs a classification score α between 0 and 1 where 1 indicates radar is more accurate and 0 if monocular is more accurate. Its input is a vector comprised of head output features, raw depths, distance, and Doppler/predicted velocity consistency. The training labels to this network are binary. We assign the GT label for training as follows. If the GT depth z of a box is closer to the radar estimated depth, the GT label $\alpha = 1$.

Otherwise it is zero. In other words,

$$\alpha = \begin{cases} 1, & |\hat{z}_j^r - z| < |\hat{z}_i^c - z| \\ 0, & \text{otherwise.} \end{cases} \tag{4.7}$$

Fused Depth Calculation. Fusion of camera and radar depths occurs as follows. Assuming there are N radar associations $\mathbf{b}_j^r$ for $j \in set(i)$ potentially associated with a given camera detection $\mathbf{b}_i^c$. We run inference over this DWN for all these N pairs to obtain a sequence of confidence scores α_j. We then calculate the fused depth $\hat{z}_{\text{fuse}}$ from radar depths, weights α_j and the monocular depth as

$$\hat{z}_{\text{fuse}} = \begin{cases} \frac{\sum_j \alpha_j \hat{z}_j^r}{\sum_j \alpha_j}, & \text{if } \exists\, j,\ \alpha_j > T_\alpha \\ \hat{z}_i^c \quad, & \text{if } \forall\, j,\ \alpha_j \le T_\alpha \end{cases}, \tag{4.8}$$

where T_α denotes depth weighting threshold.

4.5 Experiments

We apply the proposed method on the detection task of nuScenes dataset [10], a widely used dataset with both image and radar points collected in urban driving environment. The nuScenes detection dataset consists of 28,130 training samples, 6,019 validation samples, and 6,008 test samples. We experimentally show improvements in depth estimation accuracy and overall detection performance after enhancing monocular methods with the proposed strategy. The proposed method also achieves the SOTA performance in object detection via radar-camera fusion.

On Using Pre-trained Weights We use pre-trained weights for the image-based detector as this greatly reduces computation in training, and also because it preserves the performance of the monocular detector. SOTA monocular detectors are difficult to train with various hyperparameter adjustments needed to achieve published performance. Furthermore, it is not necessarily the case that jointly training the image branch and radar branch will lead to improved performance. For example, starting from pre-trained weights and without freezing them, we jointly trained radar/cam heads for 100 iterations, the monocular performance (mAP) on 900 validation images decreased from 0.33 to 0.25. Another advantage of not re-training is the method can be easily plugged into any monocular detector without hampering the performance.

Method	≤ 10	$10-30$	≥ 30	All
Monocular Heads	0.563	1.442	6.042	3.415
Radar Heads	**0.413**	**0.649**	**1.017**	**0.791**
Raw Radar Depth	1.056	1.082	1.361	1.204

Table 4.1 Depth prediction error (in meters) on nuScenes validation subset using monocular heads and radar heads on image/radar pixels labeled as object over close, medium, and long range objects.

4.5.1 Depth Errors for Camera and Radar Heads

To show the advantage of the proposed radar head over monocular head in object depth estimation, We quantitatively compare the depth estimation accuracy of the radar head with the monocular head FCOS3D [136] on random 900 images from the nuScenes validation set in Table 4.1.

We compute the mean absolute error (MAE) for camera estimated depth, radar estimated depth and raw radar depth for monocular/radar pixels where GT depths are known. For a fair comparison, the depths are compared for objects having both positive camera and radar pixels as labels and the error for each object is averaged over all pixels associated with it and final error over all objects. In addition, we also show the error if we directly use radar depth as object depth without compensating with estimated residual depth. Table 4.1 shows that the radar heads achieve better depth estimation compared with camera heads, especially for the far objects.

We also plot the distribution of estimated and GT offset depths in Fig. 4.4. The estimated residual depth follows the GT distribution. It can be seen that, typically, the object center is a little farther than the measured depth of radar points. It demonstrates the usefulness of offset depth to compensate the error from direct radar measurement.

4.5.2 nuScenes Quantitative Results

The nuScenes [10] leaderboard evaluates detection with metrics including mAP, mean average translation error (mATE), mean average size error (mASE), mean average orientation error (mAOE), and mean average velocity error (mAVE). As this chapter focuses on using radar to improve the monocular depth estimation of objects, mAP and mATE are the most relevant metrics and are reported. Other metrics are not reported since we did not update them with radar.

To show the effectiveness of the proposed camera-radar fusion on both test set (Table 4.2)

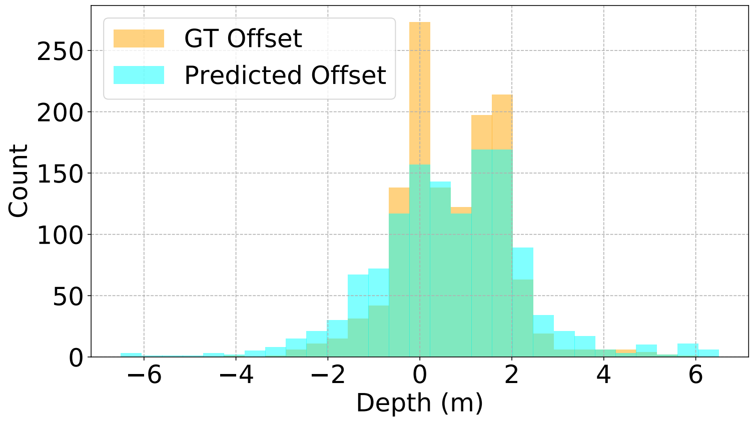


Figure 4.4 Histograms of predicted and GT depth offsets between radar returns and object centers. Both these distributions are in close agreement with each other.

R	C	Method	mATE(↓)	AP(↑)										
				Mean	Car	Truck	Bus	Trailer	CV	Ped.	Motor.	Bicycle	TC	Barrier
	✓	MonoDIS-M [120]	0.738	0.304	0.478	0.220	0.188	0.176	0.074	0.370	0.290	0.245	0.487	0.511
	✓	CenterNet [160]	0.658	0.338	0.536	0.270	0.248	0.251	0.086	0.375	0.291	0.207	0.583	0.533
	✓	FCOS3D [136]	0.690	0.358	0.524	0.270	0.277	0.255	0.117	0.397	0.345	0.298	0.557	0.538
	✓	PGD [137]	0.646	0.360	0.547	0.268	0.253	0.243	0.087	0.422	0.379	0.300	0.584	0.525
✓	✓	CenterFusion [91]	0.631	0.326	0.509	0.258	0.234	0.235	0.077	0.370	0.314	0.201	0.575	0.484
✓	✓	FCOS3D + RADIANT	0.622	0.374	0.582	0.301	0.257	**0.248**	**0.145**	0.439	0.386	0.302	0.579	0.500
✓	✓	PGD + RADIANT	**0.609**	**0.380**	**0.602**	**0.302**	**0.267**	0.242	0.107	**0.444**	**0.416**	**0.312**	**0.604**	**0.503**

Table 4.2 Performance comparison on nuScenes test set. R, C, CV, TC, Ped., and Motor. stand for radar, camera, construction vehicle, traffic cone, pedestrian, and motorcycle, respectively.

R	C	Method	mATE(↓)	AP(↑)										
				Mean	Car	Truck	Bus	Trailer	CV	Ped.	Motor.	Bicycle	TC	Barrier
	✓	FCOS3D [136]	0.739	0.326	0.494	0.236	0.316	0.115	0.057	0.416	0.306	0.303	0.549	0.465
	✓	PGD [137]	0.658	0.368	0.546	0.290	0.378	0.148	0.063	0.441	0.374	0.343	0.595	0.504
✓	✓	CenterFusion [91]	0.649	0.332	0.524	0.265	0.362	**0.154**	0.055	0.389	0.305	0.229	0.563	0.470
✓	✓	FCOS3D + RADIANT	0.653	0.363	0.587	0.291	0.371	0.120	**0.073**	0.447	0.364	0.333	0.581	0.467
✓	✓	PGD + RADIANT	**0.617**	**0.384**	**0.616**	**0.310**	**0.382**	0.141	0.068	**0.462**	**0.395**	**0.374**	**0.604**	**0.487**

Table 4.3 Performance comparison on nuScenes validation set.

and validation (Table 4.3), we compare the performance, i.e., mATE and mean/classwise average precision (AP), of monocular methods, i.e., FCOS3D [136] and PGD [137], before and after being fused with the proposed radar heads outputs, and it shows significant improvements over mAP and mATE after combined with the proposed radar heads. Note for fairness, we compare monocular and corresponding RADIANT with the same monocular weights because the performance of RADIANT is partly determined by the performance of underlying monocular detection.

Next, we compare with CenterFusion [91], the best published radar-camera fusion method on nuScenes test set. Our method outperforms CenterFusion in both mAP and mATE, which indicates that our method acquires more correct detections and smaller localization error for those positive detections. For classwise results, RADIANT shows a gain of over 18% and 20% in AP over CenterFusion [91] on Cars and Pedestrians, respectively, in Table 4.2. This improvement is significant as cars and pedestrians are common participants in traffic, with Cars accounting for about 50% of total objects in nuScenes detection dataset.

4.5.3 nuScenes Qualitative Results

Fig. 4.5 shows the detection of the monocular detector FCOS3D [136] (in magenta), detections with fusion from our proposed RADIANT (in cyan) and GT bounding boxes (in dashed orange) on image and BEV, respectively. In addition, we plot estimated position offsets for radar pixels with scores larger than 0.3. It is clear that the estimated residual depths are able to compensate the depth gap between radar measurements and actual object positions. As a result, the proposed RADIANT corrects the localization error of the FCOS3D [136] detections and achieves accurate 3D position estimates leading to better 3D detection performance. The detections from monocular and fusion have the same orientations because only depths are updated during fusion. Specific examples show the effectiveness of depth correction in both near (Fig. 4.5 (b)) and long (Fig. 4.5 (h)) ranges.

4.5.4 Ablation Studies

Our proposed RADIANT model uses DWN to predict the confidence of radar depths for depth fusion and therefore carries out the *intelligent merging* of the depths. We therefore carry the ablation of this component on the nuScenes validation set in Table 4.4 on both the monocular methods FCOS3D [136] and PGD [134]. In addition, we also consider an alternative strategy of averaging out the depth of camera box proposals and neighboring radar proposals which we call it as *Average Fusion* in the table. Table 4.4 results show that fusion methods outperform the monocular counterparts (non-fusion) methods. This is expected because the depth remains the hardest parameter to estimate for the monocular methods [82]. More importantly, the fusion with DWN outperforms the Average fusion by a significant amount on both the metrics mATE and mAP on both the monocular methods. This suggests that DWN carries out the intelligent fusion of radar and camera depths instead of blindly averaging them, proving the effectiveness of DWN.

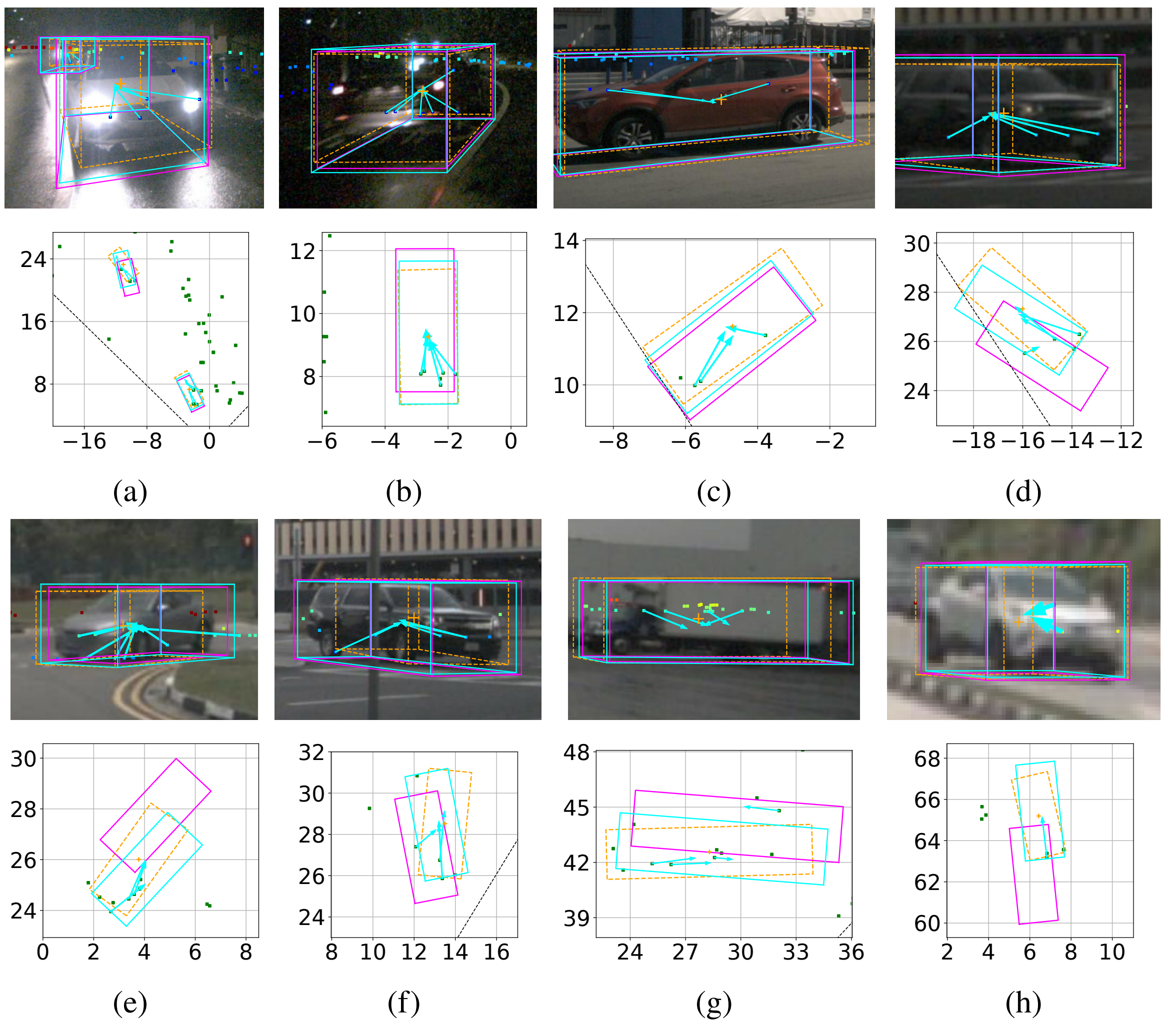


Figure 4.5 Qualitative Results. Visualization of predicted radar offsets (cyan arrows) to object centers and detections on image and BEV. The radar association of the RADIANT corrects the localization error of the FCOS3D [136], improving detection performance. Orange, magenta, and cyan boxes are GT bounding boxes, monocular detections, and RADIANT detections, respectively. The vertical axis in BEV indicates object distance in meters.

4.6 Summary

Fusing radar with camera-based detectors has proven challenging due in part to its low spatial resolution. Our network, RADIANT, provides a new way to fuse radar with 3D monocular image detectors. Using a radar branch in parallel to an image branch, RADIANT fuses both mid-level features and final detections. The mid-level features provide context to radar returns predicting the object centers offsets. RADIANT uses these offsets for association with the image detector, and to obtain accurate depth estimates for object detections, which are fused with the

Monocular Method	Fusion	mATE ($\downarrow$)	mAP ($\uparrow$)
FCOS3D [136]	None	0.739	0.326
	Average	0.711	0.342
	DWN	**0.653**	**0.363**
PGD [137]	None	0.658	0.368
	Average	0.647	0.371
	DWN	**0.617**	**0.384**

Table 4.4 Ablation on fusion strategies, i.e., average fusion and depth weighting network (DWN). The best results under the same monocular component are in bold.

camera detectors. We show that the parallel branch fusion approach in RADIANT works with two monocular detectors FCOS3D and PGD. Finally, RADIANT achieves SOTA fused radar-camera detection on the nuScenes dataset.

Limitations. Although RADIANT improves depth estimation of monocular methods, it does not enhance other detection parameters such as object sizes, and we leave velocity improvements to future work.

CHAPTER 5

RICCARDO: RADAR HIT PREDICTION AND CONVOLUTION FOR CAMERA-RADAR 3D OBJECT DETECTION

Radar hits reflect from points on both the boundary and internal to object outlines. This results in a complex distribution of radar hits that depends on factors including object category, size, and orientation. Current radar-camera fusion methods implicitly account for this with a black-box neural network. In this chapter, we explicitly utilize a radar hit distribution model to assist fusion. First, we build a model to predict radar hit distributions conditioned on object properties obtained from a monocular detector. Second, we use the predicted distribution as a kernel to match actual measured radar points in the neighborhood of the monocular detections, generating matching scores at nearby positions. Finally, a fusion stage combines context with the kernel detector to refine the matching scores. Our method achieves the SOTA radar-camera detection performance on nuScenes. This chapter has been accepted for publication as [72].

5.1 Introduction

3D object detection [10, 28] is a key component of scene understanding in autonomous vehicles. It predicts nearby objects and their attributes including 3D location, size, orientation, and category, setting the stage for navigation tasks such as path planning. The primary sensors used for 3D object detection are cameras, lidars [154], and radars, with the focus here on the two-sensor combination that is the least expensive and already ubiquitous on vehicles, namely cameras and radars. This chapter asks how to combine camera and radar data in order to achieve the best performance improvement over a single modality detection.

Detection can be performed on camera and radar individually, with each sensor having its strengths and drawbacks. Cameras are inexpensive and capture high-resolution details and texture with SOTA methods [46, 77, 99] achieving accurate object classification, as well as estimating size and orientation. One of the primary limitations is the depth-scale ambiguity, resulting in relatively inaccurate object depth estimation [46, 77]. In contrast, current automotive radar is another inexpensive sensor that directly measures range to target, as well as Doppler velocity, and is robust to adverse weather such as rain, snow, and darkness [91]. The drawbacks of radar are its very sparse scene sampling and lack of texture, making it challenging to perform tasks such as object categorization, orientation, and size estimation. For example, radar point clouds collected in nuScenes dataset [10] are 2D points on radar BEV plane without height measurements (the default height is zero). This comparison shows that the strengths of radar and camera are complementary, and indeed this chapter explores how we can combine information from these two modalities to improve 3D object detection.

While radar and lidar are both widely used depth sensors for autonomous vehicles, they differ significantly in their target sampling characteristics. Lidar points align densely with the edges of objects, and for vehicles typical form a distinct “L" or “I" distribution. These regular distributions are aligned with the object pose and enable precise shape and pose estimation from lidar scans as evidenced by top-performing lidar methods [155] on nuScenes achieving mAP of 0.702 and nuScenes detection score (NDS) of 0.736. In contrast, radars have wide beam-width with low angular resolution and often penetrate objects or reflect from their undersides. This leads to a much sparser and dispersed distribution of hits on objects. From this distribution it is much more challenging to estimate target shape, category, and pose, and the top-performing radar-only method, RadarDistill [2], achieves a far lower mAP of 0.205 and NDS of 0.437 on nuScenes.

The combination of camera with radar has potential to significantly enhance radar-only methods, with camera data providing strong results on object category, shape, and pose. Radar, with its direct range measurements, can contribute range and velocity to a combined detector. Existing radar-camera models combine camera and radar via concatenation [91], (weighted) sum [43], or attention-based operations [44, 45]. But obtaining measurable gains by fusing sparse radar and camera is difficult, with the top radar-camera method [45] having lower performance than camera-only methods [67].

To address the difficulty in combining disparate modalities, we conjecture that directly *modeling the radar distribution's dependency on target properties* will enable radar returns to be more effectively aligned and leveraged in object detection. As shown in Fig. 5.1, we introduce a model

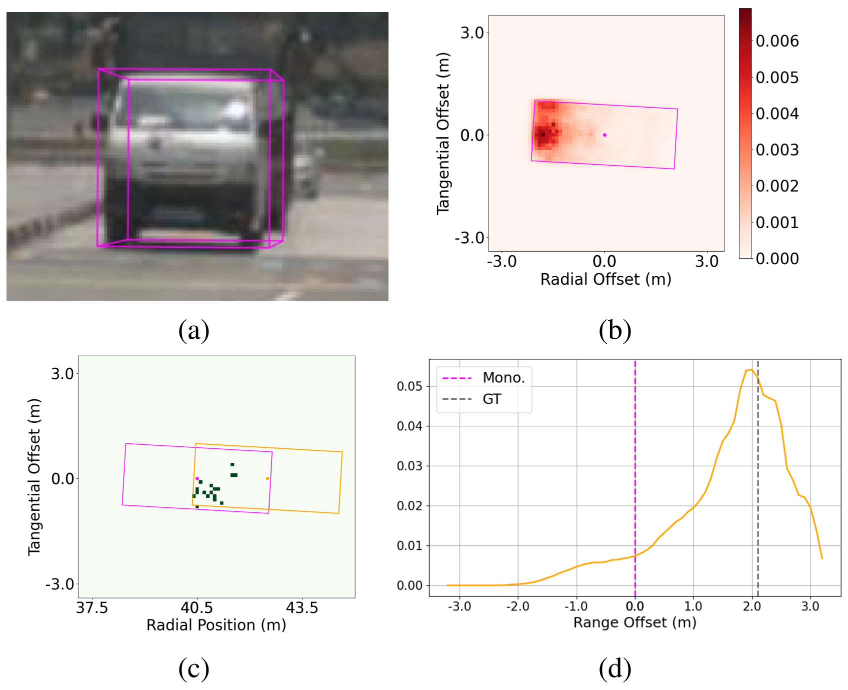


Figure 5.1 Given a (a) monocular detection, we estimate (b) radar point distribution relative to its bounding box in BEV; then we shift the distribution and convolve it with (c) actual radar measurement in the neighborhood to compute (d) similarity scores and estimate an updated position, where the matching score is maximum. In (c) the monocular bounding box (in magenta) is misaligned with radar points; the updated position (in orange) with peak matching score shifts the box to a farther range so that relative positions of radar points match the predicted distribution (radar hits concentrated at the head of vehicle instead of in the middle).

that generates a radar hit prediction (RIC) in BEV from a given monocular detection priors, i.e., bounding box size, relative pose, and object class. By moving a distribution kernel in the neighborhood of monocular estimated center and computing the similarity between the kernel and actual radar measurements, we identify potential object centers with high similarity. Those position candidates are passed through a refinement stage for final position estimation.

In nutshell, this chapter makes the following contributions:

- Builds a model to predict radar hit distributions relative to reflecting objects on BEV.
- Proposes position estimation by convolving predicted radar distribution with actual radar measurements.
- Achieves SOTA performance on the nuScenes dataset.

5.2 Related Works

Monocular 3D Detection. Monocular 3D detection is known for its low cost and simple setup, attracting extensive research optimizing every component in the detection pipeline, e.g., architectures [5, 6, 141], losses [8, 48, 120], and NMS [47]. Efforts to alleviate the intrinsic ambiguities from camera image to 3D include incorporating estimated monocular depth [105, 121, 138], designing special convolutions [46], considering camera pose [161], and taking advantage of CAD models [70]. However, depth ambiguities still remain a bottleneck to performance [48, 82] and hence, fusion with radar is a promising strategy for enhancing depth estimation while keeping low computation cost.

Camera-Radar Fusion for Detection. Camera-radar fusion has been widely applied to different vision tasks, e.g., depth estimation [63, 76, 123], semantic segmentation [151], target velocity estimation [75, 98], detection [73, 151], and tracking [127]. For detection, various camera-radar methods have been proposed which differ in representations [44, 91], fusion level [43, 91], space [43, 45, 158], and strategies [145]. There are different radar point representations. As 2D radar measures BEV XY locations without height, it is natural to represent radar points in binned BEV space [45]. To associate radar points with camera source, radar points are also modeled as pillars [91, 145] with fixed height in 3D space. In addition, radar points are represented as point feature [44] and each point as a multi-dimensional vector with elements of radar locations and other properties from measurement such as radar cross-section.

Radar and image sensors can interact at either the feature-level [43] or the detection-level [91]. While intermediate image features offer a wealth of raw information, such as textures, detection-level outputs provide directly interpretable information with clear physical meanings, making them well-suited for fusion tasks. With the rapid advancement of monocular 3D detectors, leveraging detection-level image information allows us to capitalize on their accurate estimates of object category, size, orientation, and focus on refining position estimation, particularly range estimation, where radar excels as a range sensor. Therefore, in this chapter, we utilize camera information at the detection level.

The fusion is conducted in image view [73, 91] , BEV [45] or a mix of the two [43]. Image features are naturally in image view, while radar is in BEV. To combine them in one space, fusion methods either project radar points to image space [73, 91] or lift image features to 3D space [45, 158]. Image view suffers from overlappings and occlusions while it is imprecise to transform from image view to BEV without reliable depths. In this work, we adopt BEV space for fusion as we use image source monocular detections, which are already in 3D space.

The radar-camera association is conducted by associating the radar pillars with monocular boxes in 3D space [91] or projecting the pillars on image to extract corresponding image features [145]. However, none of these methods explicitly leverage radar point distributions to address the misalignment problem in radar-camera fusion.

5.3 RICCARDO

Our goal is to enhance object position estimation using radar returns, surpassing the capabilities of monocular vision. The challenge lies in the sparse and non-obvious alignment of radar returns with object boundaries and features, unlike the dense and consistent lidar returns. To address this, we propose a method that explicitly models the statistics of radar returns on objects, taking its category, size, orientation, range, and azimuth into account. These statistics enable radar returns to improve monocular detections.

Our approach, called Radar Hit Prediction and Convolution for Camera-Radar 3D Object Detection (RICCARDO), is illustrated in Fig. 5.2 and involves three stages. The first stage predicts the radar distribution returns on an object based on monocular detector outputs. The second stage convolves the predicted distribution with accumulated and binned radar measurements to obtain a range-based score. The third and final stage refines the range-based score to obtain a final range estimate. We describe the details of each stage below.

5.3.1 Stage 1: Radar Hits Prediction (RIC) Model

The RIC model aims to predict radar hit position distributions on objects in BEV, conditioned on object category, size, orientation, range, and azimuth. This model leverages monocular detection data to predict radar returns as a distribution, enabling comparison with actual measured returns

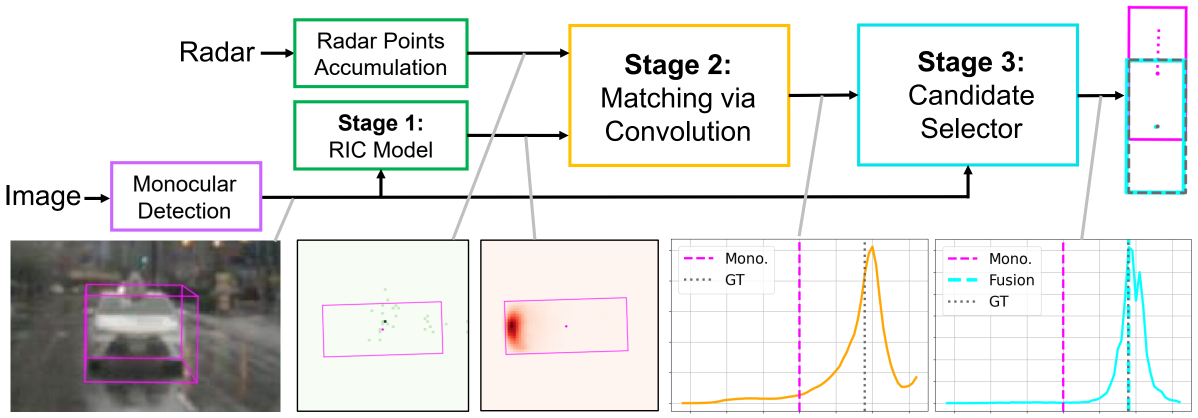

Radar
Radar Points Accumulation
Stage 1: RIC Model
Image
Monocular Detection
Stage 2: Matching via Convolution
Stage 3: Candidate Selector
Mono.
GT
Mono.
Fusion
GT

Figure 5.2 RICCARDO inference. RICCARDO leverages a monocular detector to identify objects and estimate their attributes (category, size, orientation, and approximate range) and involves three stages. Its Stage 1 then predicts the radar hit distribution (RIC) for each object. Stage 2 bins and convolves the observed accumulated radar returns with the RIC, to generate a matching score over range. A final Stage 3 fusion refines these scores to yield a precise target range estimate.

(Section 5.3.2). This section details the construction and learning of this predictive model. We model radar hit distributions as a probability of radar return over a set of grid cells.

Coordinate System. A key choice is the coordinate system to predict this distribution. Possibilities include object-aligned, sensor-aligned or ego-vehicle aligned systems. We choose an object-aligned system for modeling radar hit position distributions. This choice allows for more gradual changes in probabilities as a function of relative sensor location compared to ego-vehicle or sensor-aligned systems, which exhibit significant variations with changes in relative object pose. Consequently, the object-aligned system facilitates learning from limited data.

Architecture. Fig. 5.3 shows the overview of Stage 1 in RICCARDO. Stage 1 employs a neural network to predict the distribution of radar points in object-centric BEV coordinates, conditioned on object category, size, orientation, range, and azimuth. The output is a 2D quantized BEV probability map centered on the object, with X and Y axes aligned to the object's length and width dimensions. The RIC pixel value represents the density of radar hits at those locations. The network architecture comprises an MLP model, with parallel preprocessing branches for input parameters and a main branch that fuses these features and predicts the RIC.

Ground Truth Distribution. To construct a GT distribution for a given target, we define a grid

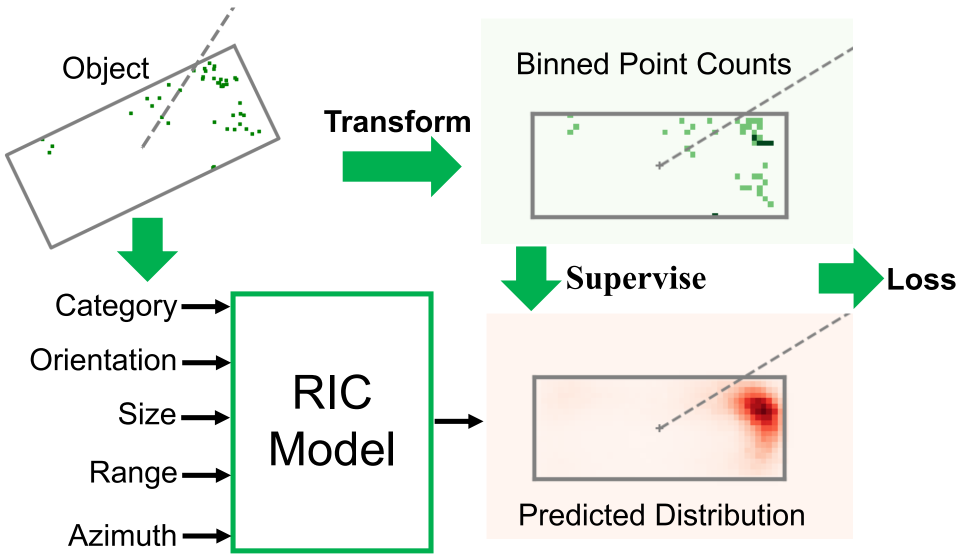


Figure 5.3 Stage-1 RIC Model Training. The radial ray from ego to target is plotted as dashed line for reference.

relative to the target's known position and accumulate radar hits over a short time interval. The normalized density of returns for a grid cell i, $\bar{P}_i$, is calculated as:

$$\bar{P}_i = \frac{c_i}{\sum_{i=1}^{N} c_i}, \tag{5.1}$$

where c_i is the number of radar returns in grid cell i, and N is the total number of cells on the RIC map. This model only includes the points within the object boundary.

Targets may move during an accumulation period in a driving scene, and directly accumulating radar hits causes smearing. Thus, we offset each radar hit position by the object motion between the radar return and the final target position. A pair of sequential annotated locations can provide GT object velocity, $\boldsymbol{v}_{\mathrm{O}}$, for calculating this. Yet, radar Doppler velocity is more accurate for the radial component; thus, we combine the Doppler velocity, $\boldsymbol{v}_{\mathrm{D}}$, with the tangential component of $\boldsymbol{v}_{\mathrm{O}}$. Given a unit vector perpendicular to radar ray, $\boldsymbol{n}_{\mathrm{T}}$, the offset $\boldsymbol{m}$ of a radar point is:

$$\boldsymbol{m} = ((\boldsymbol{v}_{\mathrm{O}} \cdot \boldsymbol{n}_{\mathrm{T}})\boldsymbol{n}_{\mathrm{T}} + \boldsymbol{v}_{\mathrm{D}})\,\Delta t, \tag{5.2}$$

where Δt is the time between the radar measurement and current sweep. We apply these offsets before calculating the probabilities in Eq. (5.1).

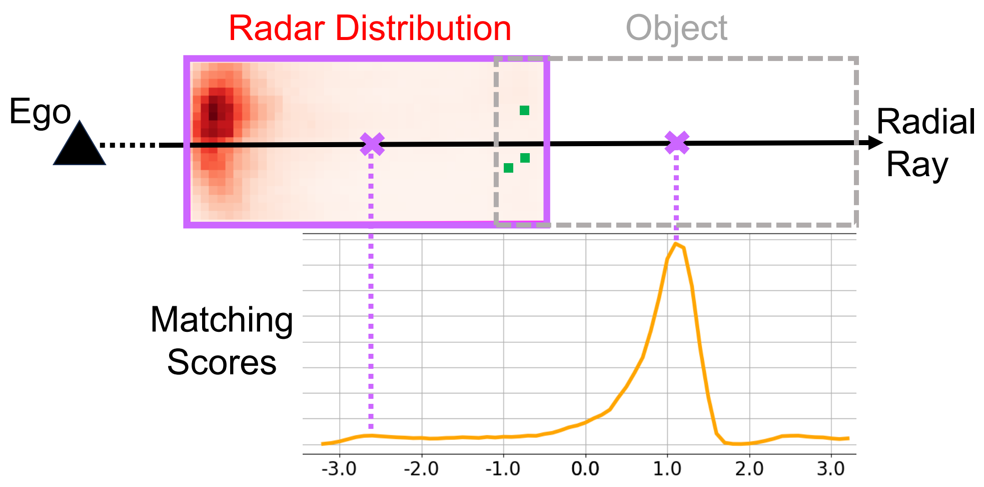


Figure 5.4 Stage 2 takes the RIC predictions and neighboring radar points. It matches predicted radar distribution with radar points in the radial direction and computes binned and range-dependent matching scores. Peak positions indicates range estimations.

Loss Function. The loss function incorporates cross-entropy loss L_{CE} between the predicted radar distribution P, and the GT distribution from the accumulated radar returns, $\bar{P}$. In addition, we include a smoothness regularization term L_{S} on P to encourage spatial smoothness:

$$L_{\mathrm{S}} = \frac{1}{N_{\mathrm{c}}(N_{\mathrm{r}}-1)} \sum_{j=1}^{N_{\mathrm{c}}} \sum_{i=1}^{N_{\mathrm{r}}-1} |P(i,j)-P(i+1,j)| + \frac{1}{N_{\mathrm{r}}(N_{\mathrm{c}}-1)} \sum_{i=1}^{N_{\mathrm{r}}} \sum_{j=1}^{N_{\mathrm{c}}-1} |P(i,j)-P(i,j+1)|, \quad (5.3)$$

where i and j are pixel indices within $N_{\mathrm{r}} \times N_{\mathrm{c}}$ grid of rows and columns. Thus, the total loss for training is $L_{\mathrm{CE}} + L_{\mathrm{S}}$.

5.3.2 Stage 2: Convolving RIC with Radar

Stage 2 uses the predicted radar density from the RIC model to score the consistency of object positions with the measured radar hits. By binning measured radar returns, we can use a convolution of the RIC to obtain similarity scores as a function of range, and so locate potential target positions.

Fig. 5.4 shows the overview of Stage 2. Both the predicted and measured radar data are resampled into a BEV space with the X-axis along radial direction (from ego to target, approximately parallel to radar rays), and the Y-axis along the tangential direction. We align the central row of the RIC map with the central row of the radar measurement map and restrict the search to be along X-axis (i.e., the radial axis) near to the monocular detected center. Our motivation is that monocular detections are more accurate tangentially (i.e., image space) and less radially. To compute the similarity between the predicted and measured densities, we slide the RIC kernel along the radial axis and calculate by sum of dot product (convolution) between the RIC and the actual point count at each position. This results in a 1D matching score profile along the radial dimension, indicating potential target locations in the vicinity of the monocular centers.

Specifically, given a binned radar distribution P with center (L_P, L_P) and resolution of $(2L_\text{P} + 1, 2L_\text{P} + 1)$ and radar binned count C with center (L_C, L_C), i.e., object center from monocular prediction, and size of $(2L_\text{C} + 1, 2L_\text{C} + 1)$, the row convolution (or cross-correlation) is

$$S_\text{STG2}(n) = \sum_{i=-L_\text{P}}^{L_\text{P}} \sum_{j=-L_\text{P}}^{L_\text{P}} P\left(L_\text{P} + i + 1, L_\text{P} + j + 1\right) C\left(L_\text{C} + i + 1, L_\text{C} + j + n + 1\right), \tag{5.4}$$

where n is offset to the object center along the row.

To create the measured densities, we first accumulate multiple radar sweeps over a short interval prior to the detection. Object motions of radar points are compensated by Doppler velocity and object velocity from monocular detections similar to Eq. (5.2). The difference is that object velocity, $\boldsymbol{v}_\text{O}$, is obtained from the monocular detector.

5.3.3 Stage 3: Camera-Radar Candidate Selector

Matching scores from Stage 2 provide position candidates, which have high scores. The purpose of Stage 3 is to select the best range predictions among Stage-2 candidates by considering the combined evidence from monocular detection and radar measurements. Stage 3 trains a neural network to rescore the candidate positions using additional evidence.

This model should consider multiple factors that indicate the confidence of each choice. For instance, high detection scores imply high confidence in monocular detection; large matching scores indicate more accurate matched positions from Stage 2; monocular detector excels at low ranges while suffers at long ranges where Stage-2 candidates may be a better choice because of the help of radar measurement.

Architecture and Loss. The Stage-3 network takes two types of inputs: (a) monocular detection parameters (class, range, and size); (b) Stage-2 matching scores at binned ranges. The network preprocesses inputs via separate linear layers, concatenates the results, feeds them to an MLP to extract features, and predicts a confidence score per range candidate. Cross-entropy loss is used for training the network. Training data are generated from true positive monocular detections with non-zero Stage-2 matching scores.

Inference. Denoting predicted Stage-3 scores as $S_{\text{STG3}}(i)$, where i is quantized offset to monocular predicted range, we estimate the range offset by finding the peak position as

$$\Delta n = \text{argmax}_i \left(S_{\text{STG3}}(i) \right), \tag{5.5}$$

and corresponding Stage-3 score $S_{\text{STG3}}(\Delta n)$. The final predicted range R_{F} can be expressed as

$$R_{\text{F}} = R_{\text{CAM}} + \Delta n \, b_{\text{p}}, \tag{5.6}$$

where R_{CAM} is monocular predicted range, and b_{p} is bin size. Typically, the estimated range is approximately at one of the peak positions from Stage-2 score or at monocular estimated range; thus Stage 3 implicitly learns to select the best position candidates from previous stages. We also update detection score by combining monocular detection score S_{CAM} and Stage-3 score as

$$S_{\text{F}} = S_{\text{CAM}} + \alpha S_{\text{STG3}}(\Delta n), \tag{5.7}$$

where α is a Stage-3 weighting parameter. Note S_{STG3} has been processed by the Softmax function, and its value ranges from 0 to 1.

5.4 Experiments

Dataset. Our experiments are based on the widely-used nuScenes dataset [10], with both images and radar points collected in urban driving scenarios. Equipped with six cameras and five radars, the ego-vehicle scans traffic environments in 360 degrees. There are 700 training scenes, 150 validation scenes, and 150 test scenes, each with 10 classes of objects specified with bounding boxes.

Data Splits. We follow the standard splits of the nuScenes detection benchmark: the test results are obtained from model trained on nuScenes training plus validation set (34K frames) and evaluated on the test set (6K frames); the validation results trained on nuScenes training set (28K frames) and evaluated on the validation set (6K frames).

Models	SparseBEV (V2-99)	SparseBEV (ResNet101))	Stage 1	Stage 3
Params (M)	94.0	63.6	19.2	0.3

Table 5.1 Number of parameters in SparseBEV with different backbones as well as in RICCARDO Stage 1 and Stage 3.

Implementation Details. Our code uses PyTorch [100] and detection package MMDetection3D [16]. Our experiments use SparseBEV [67] as the monocular detector. Note that RICCARDO is flexible and easily adaptable to other monocular methods. To preserve the premium detection performance of monocular component and focus on training the Stage-3 model, we use pretrained weights for the monocular branch, which are frozen in training and inference. We train Stage-1 and Stage-3 models with RMSProp optimizer for 120 epochs with an initial learning rate of 1×10^{-6}, which is reduced by half at the 60th epoch. We list the number of parameters in RICCARDO Stage-1 and Stage-3 models as well as in underlying monocular models with two backbones in Table 5.1. RICCARDO Stage 1 and Stage 3 are relatively lightweight compared to monocular models.

Stage 1: We implement the Stage-1 network with a lightweight MLP-like network shown in Fig. 5.5. An input sample consists of a series of vectors representing different properties of object, e.g., size, orientation, and range. They are first processed by separate linear projection

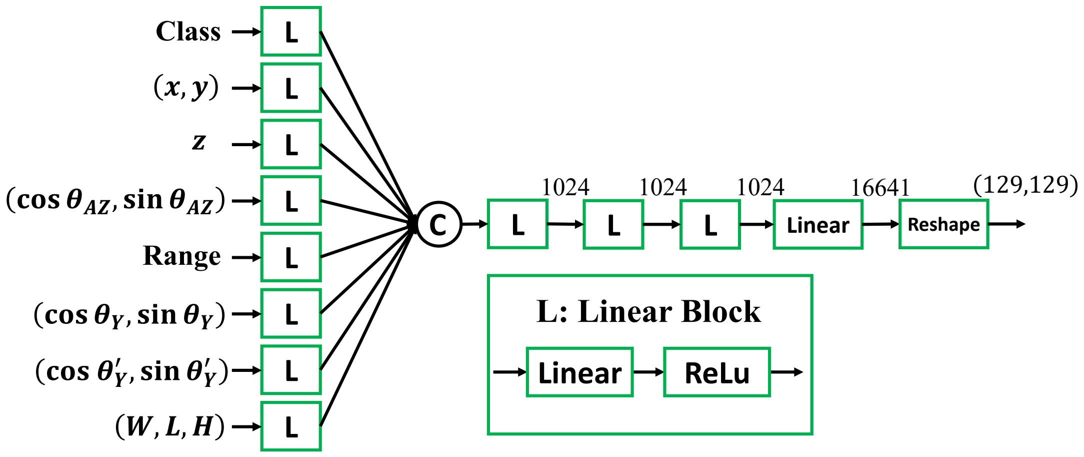

Class
(x, y)
z
$(\cos\theta_{AZ}, \sin\theta_{AZ})$
Range
$(\cos\theta_Y, \sin\theta_Y)$
$(\cos\theta'_Y, \sin\theta'_Y)$
(W, L, H)
L
C
1024
1024
1024
16641
(129,129)
Linear
Reshape
L: Linear Block
Linear
ReLu

Figure 5.5 Stage-1 Network Structure. The class input is in one-hot encoding; z represents heights of bounding box bottom faces; θ_{AZ} stands for azimuths of objects in ego coordinates; θ_{Y} and θ'_{Y} are object yaws in ego coordinates and relative yaws (i.e., $\theta_{\mathrm{Y}} - \theta_{\mathrm{AZ}}$), respectively. “C" represents concatenation, and “Linear" denotes a linear transformation layer. Feature sizes are marked besides network layers.

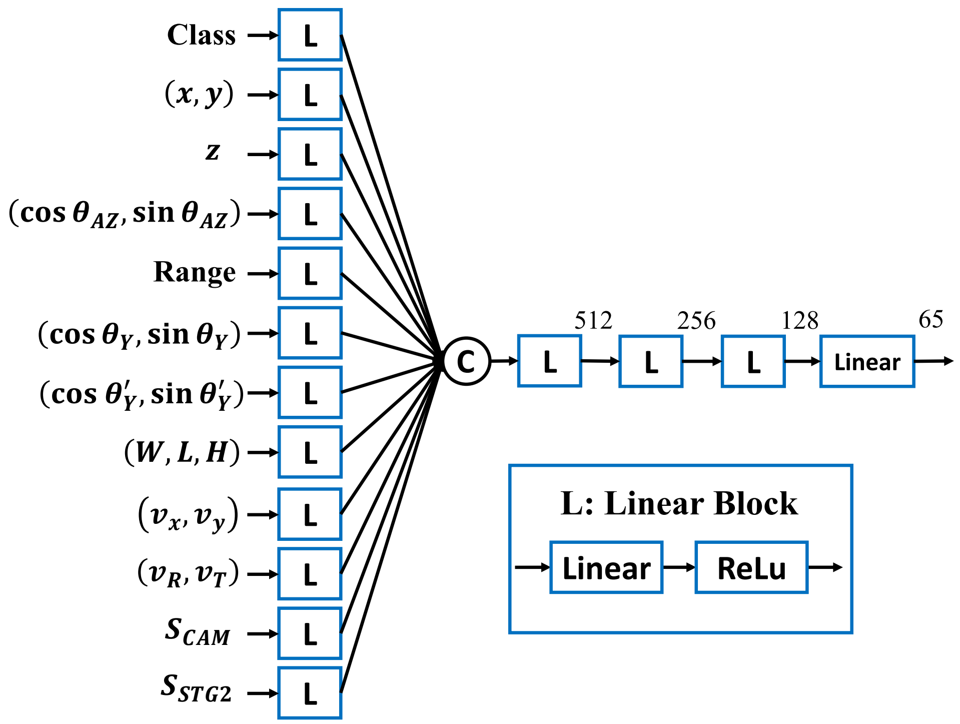


Figure 5.6 Stage-3 Network Structure. The inputs v_x and v_y are monocular estimated object velocities in ego coordinates; v_R and v_T are monocular velocities in radial and tangential directions, respectively; S_CAM and S_STG2 represent monocular detection scores and Stage-2 matching scores, respectively.

layers before being concatenated and fed into an MLP of 3 hidden layers. The network output (i.e., the binned distribution map) is defined in object local coordinates with X-axis parallel to object length, Y-axis to width, and map center at object center. It has a resolution of 129×129 with a pixel size of 0.1×0.1 meters for small and medium-sized object categories and pixel size of 0.2×0.2 meters for large-sized categories such as buses and trailers. GT distribution is generated by accumulating 13 neighboring radar sweeps (6 previous sweeps, 1 current, and 6 future ones). We assume radar points distribute within the GT bounding boxes on BEV, and thus points outside the bounding boxes are ignored and not used for training. We train Stage-1 and Stage-3 models separately since Stage-1 model is invariant to its underlying monocular model while Stage-3 network depends on the monocular model.

Stage 2: To perform Stage-2 convolution along the radial direction, the measured radar positions are binned in radial-tangential coordinates to generate a radar measurement map, with X-axis

Modality		Method	NDS (↑)	mAP (↑)	mATE (↓)	mASE (↓)	mAOE (↓)	mAVE (↓)	mAAE (↓)
Radar	Camera								
	✓	PGD [137]	0.448	0.386	0.626	0.245	0.451	1.509	0.127
	✓	SparseBEV [67]	0.675	0.603	0.425	0.239	0.311	0.172	0.116
✓	✓	MVFusion [145]	0.517	0.453	0.569	0.246	0.379	0.781	0.128
✓	✓	CRN [45]	0.624	0.575	0.416	0.264	0.456	0.365	0.130
✓	✓	RCBEVDet [65]	0.639	0.550	0.390	**0.234**	0.362	0.259	**0.113**
✓	✓	HyDRa [143]	0.642	0.574	0.398	0.251	0.423	0.249	0.122
✓	✓	HVDetFusion [55]	0.674	0.609	0.379	0.243	0.382	0.172	0.132
✓	✓	SparseBEV + **RICCARDO**	**0.695**	**0.630**	**0.363**	0.240	**0.311**	**0.167**	0.118

Table 5.2 Detection Performance on nuScenes Test Set. RICCARDO achieves SOTA performance for camera-radar fusion.

Modality		Method	NDS (↑)	mAP (↑)	mATE (↓)	mASE (↓)	mAOE (↓)	mAVE (↓)	mAAE (↓)
Radar	Camera								
	✓	PGD [137]	0.428	0.369	0.683	0.260	0.439	1.268	0.185
	✓	SparseBEV [67]	0.592	0.501	0.562	0.265	0.320	0.243	0.195
✓	✓	MVFusion [145]	0.455	0.380	0.675	**0.258**	0.372	0.833	0.196
✓	✓	CRN [45]	0.607	**0.545**	0.445	0.268	0.425	0.332	**0.180**
✓	✓	RCBEVDet [65]	0.568	0.453	0.486	0.285	0.404	0.220	0.192
✓	✓	HyDRa [143]	0.617	0.536	**0.416**	0.264	0.407	0.231	0.186
✓	✓	HVDetFusion [55]	0.557	0.451	0.527	0.270	0.473	**0.212**	0.204
✓	✓	SparseBEV + **RICCARDO**	**0.622**	0.544	0.481	0.266	**0.325**	0.237	0.189

Table 5.3 nuScenes Validation Results.

parallel to the ray from ego vehicle to target center and map center at target center detected by a monocular method. It has a resolution of 193×193 with 2 pixel sizes mentioned above. Before performing convolution, the predicted radar distribution map of Stage 1 is rotated from object local coordinates to radial-tangential coordinates. The search range of convolution is -3.2m to 3.2m relative to the range of target center estimated by the monocular method.

Stage 3: As shown in Fig. 5.6, we implement the Stage-3 network via a lightweight MLP similar to Stage 1. To create labels for GT range, we associate monocular detections with GT bounding boxes under the conditions that the GT bounding boxes fall in the search range of associated monocular detections (with radial distance $\leq$ 3.2m) and on the ray from ego to monocular detections (with tangential distance $\leq \min(0.5\text{m}, L)$, where L is the object length).

5.4.1 Quantitative Results on nuScenes

Tables 5.2 and 5.3 show the performance of RICCARDO on test and validation set, respectively. The proposed fusion of radar points with monocular detection proposals improves the performance of position estimation, which is evaluated with mAP and mATE for true positive detections. We compare performance of SOTA methods of monocular and radar-camera fusion. Note that for a fair comparison, the monocular component in RICCARDO, i.e., SparseBEV, uses exactly the same weights within Table 5.2 and likewise uses the same weights within Table 5.3. Specifically, SparseBEV uses V2-99 [54] and ResNet101 [31] as its backbones in Tables 5.2 and 5.3, respectively.

By comparing RICCARDO with its monocular counterpart (i.e., 8th row vs. 2nd row in Tables 5.2 and 5.3), we see a significant improvement in mAP and a reduction in mATE when using RICCARDO with radar data as inputs. Meanwhile it preserves good performance of its monocular component in other aspects, e.g., size and orientation estimation.

RICCARDO also achieves SOTA performance among published radar-camera fusion methods: in test set performance shown in Table 5.2 and validation set results shown in Table 5.3, RICCARDO achieves the best overall performance measured by NDS and comparable performance in other metrics. As a detection-level fusion, final performance of RICCARDO depends on the quality of its underlying monocular model. We observe a stable and significant improvement in overall performance over its monocular models in both Tables 5.2 and 5.3, although the monocular components adopt different backbones. The ease of plugging in different monocular components in our fusion architecture allows RICCARDO to capitalize on SOTA monocular models and achieve better fusion performance.

5.4.2 Evaluating Stages 2 and 3

To evaluate the Stages 2 and 3 in object range estimation, we compare the range error from monocular detection, Stage-2, and Stage-3 estimations. Stage-1 model is trained with nuScenes training set and Stage-3 model is trained with detections by the monocular method and corresponding Stage-2 matching scores from nuScenes training set.

To generate Stage-2 estimation, the trained Stage-1 model is applied to monocular detections (SparseBEV) in nuScenes validation set to generate radar distributions, which are subsequently convolved with radar measurements (in Stage 2) to obtain matching scores at binned ranges along the ray. The range with the maximum matching score is used as Stage-2 estimation for this evaluation. Finally, we estimate Stage-3 output by feeding the Stage-2 outputs and monocular detections to the Stage-3 model. From Table 5.4, we can see that simple extraction from Stage 2

Method	Class-Mean	Car	Truck	Bus	Trailer	CV	Ped.	Motor.	Bicycle	TC	Barrier
Monocular	0.83/0.57	0.65/0.38	0.86/0.56	1.04/0.80	1.23/1.08	1.13/0.88	0.85/0.55	0.81/0.50	0.65/0.40	0.51/0.24	0.59/0.31
Stage 2	0.94/0.52	0.50/0.21	0.75/0.39	0.77/**0.46**	1.53/1.05	1.11/0.75	1.13/0.56	0.82/0.35	0.86/0.44	0.93/0.41	1.01/0.55
Stage 3	**0.65/0.36**	**0.38/0.18**	**0.59/0.34**	**0.70/0.46**	**1.13/0.76**	**0.87/0.63**	**0.71/0.28**	**0.57/0.29**	**0.54/0.24**	**0.46/0.17**	**0.57/0.27**

Table 5.4 Comparison of range estimation accuracy of monocular, Stage-2, and Stage-3 estimates. We use mean/median of absolute range error (meter) as metrics. [Key: Ped.= Pedestrian, Motor.= Motorcycle, CV= Construction Vehicle, TC= Traffic Cone.]

can improve over monocular methods with lower median error but suffers from outliers with larger mean error. Stage 3 achieves the best range estimation accuracy across all 10 categories, by fusing the monocular estimation and Stage-2 outputs.

5.4.3 Qualitative Results

In Fig. 5.7, we show examples of RICCARDO being applied to monocular detections. The radar BEV map in (b) and predicted radar distribution in (c) are both centered at the monocular object center and with X-axis being radial direction and Y-axis tangential direction. We can observe the complexity of predicted radar distributions in (c), which vary according to object size and orientation relative to radar rays. The edges of objects facing the radar tend to have higher densities, and large vehicles have a wider spread (see the 4th column), with reflections by parts under the vehicle near the tires.

Fig. 5.7(d) shows Stage-2 matching scores as a function of radial offset (X-axis), with monocular estimated range (magenta) and GT range (gray). Multiple peaks in the 4th column indicate ambiguities in matching radar hits. (e) shows the predicted Stage-3 scores in blue, which typically have a sharper peak than Stage 2, illustrating Stage-3 candidate refinements. In the fourth column Stage 3 resolved ambiguity by enhancing one peak. Row (f) shows that RICCARDO improves monocular method in radial position prediction as predicted bounding boxes are closer to GT compared to monocular detections. The radial directions are plotted as dotted lines for reference.

5.4.4 Ablation Study

5.4.4.1 Ablation On Stage-1 Radar Distribution Models

To assess the benefit of using our trained RIC model to represent radar hits on objects, we compare it with using two baseline distributions, i.e., a uniform radar distribution within object

Distribution	MAE (↓)	MMS (↑)
L-Shaped	0.77	0.059
Uniform	0.67	0.078
RIC	**0.47**	**0.105**

Table 5.5 Comparison of RICCARDO using two baseline radar hit distributions for Stage 1 versus RIC. The metric is MAE of range in meters and mean matching score (MMS) between the distributions and actual measurements.

boundaries and an “L-shaped" distribution on reflecting sides of bounding boxes. The “L-shaped" distribution is a simulation of lidar point distribution. Each model is passed through Stage 2, and we estimate object range at the maximum matching score, and compute their range errors. We also use matching score (i.e., dot product of predicted distribution and pixel-wise radar point counts from accumulated measurement) as an additional metric for evaluating distribution accuracy. We compute mean matching score (MMS) over all classes. We train RIC with radar measurements and GT bounding boxes in nuScenes training set and evaluate on nuScenes validation set. Note that, in this experiment, GT bounding boxes are used to generate the radar distribution, and no monocular boxes are involved. Table 5.5 shows that, with the smallest range estimation error and the largest matching score, the RIC distribution captures more accurately the real radar distribution compared with the two baseline distributions.

5.4.4.2 Ablation on Range and Score Updating

In Stage-3 inference we update both range and detection score. Detection scores indicate confidence in prediction and have an impact on mAP computation, where predictions with higher scores have priority as true positives to be associated with GT. We update detection scores by adding Stage-3 scores weighted by α to monocular scores. We test different range and score updating options with different α and list resultant detection performance in Tab. 5.6. We can see both range and score updating improve detection performance while range updating has significantly bigger impacts on performance.

5.4.4.3 Ablation on Number of Radar Sweeps

Within 0.5s time window, there are about 7 sweeps of radar points (i.e., 1 current plus 6 past ones) from radars running at 13Hz in nuScenes Dataset [10]. We accumulate multiple radar sweeps

Update Range	Update Score	α	NDS (↑)	mAP (↑)
		-	0.590	0.501
	✓	0.5	0.593	0.503
✓		-	0.617	0.543
✓	✓	0.2	0.620	**0.545**
✓	✓	0.5	**0.621**	**0.545**
✓	✓	0.8	**0.621**	0.543
✓	✓	1.0	0.620	0.541

Table 5.6 Ablation on updating range and detection score with fusion weight α. We use SparseBEV [67] with backbone ResNet101 for the monocular components in RICCARDO. For efficiency, the data used for evaluation are subset of NuScenes Val set with 600 random samples.

Num. of Sweeps	NDS (↑)	mAP (↑)
0	0.590	0.501
1	0.597	0.512
3	0.612	0.531
5	0.618	0.541
7	**0.621**	**0.545**

Table 5.7 Ablation on Number of Radar Sweeps. More radar sweeps result in better detection performance. Key: Num.= Number

during inference, and the number of radar sweeps may impact detection performance, as more sweeps provide denser radar measurement used for Stage 2. To verify this, we run RICCARDO multiple times with radar input from 0, 1, 3, 5, and 7 sweeps, respectively and record their detection performance. Note using 0 radar sweep refers to applying only monocular detector without fusion. As shown in Tab. 5.7, more radar sweeps lead to better detection performance as expected.

5.4.5 Visualization of Radar Distributions

To visualize how predicted distribution varies with viewing angles, we simulate object parameters with different orientations and apply Stage-1 model to generate corresponding radar hit distributions. Figs. 5.8 and 5.9 shows predicted distributions for car, bus, bicycle, and barrier with different orientations and distances. We can see the distributions vary with category, orientation, and distance. For example, radar distributions are less concentrated spatially at longer range because of larger beam width. We can also notice that distributions of radar points reflected by the tail and head of cars (as shown in the 1st and last row of Fig. 5.8) are different because of their different surface shapes.

5.5 Summary

This chapter presents a novel radar-camera fusion strategy that utilizes BEV radar distributions to improve object range estimation over monocular methods. Evaluation on the nuScenes dataset shows that RICCARDO realizes stable and significant improvements in object position estimation over its underlying monocular detector, and achieves the SOTA performance in radar-camera-based 3D object detection. We believe this effective method, that is simple to implement, will broadly benefit existing and future camera-radar fusion methods.

Limitations. First, as a detection level fusion, RICCARDO only uses high-level monocular detection parameters and does not directly utilize low-level image features. Information loss is inevitable in this low-level to high-level feature transition (e.g., false negative detections), and it is difficult to make use of radar points for further improvement if an object is missed by the monocular component in the first place. Second, RICCARDO adopts BEV representation for radar points distribution. However, BEV representation has an intrinsic limitation to represent radar hits from two vertically positioned targets at the same BEV location (e.g., a person riding on a bicycle), since their reflected radar hits are mapped to the same BEV pixel.

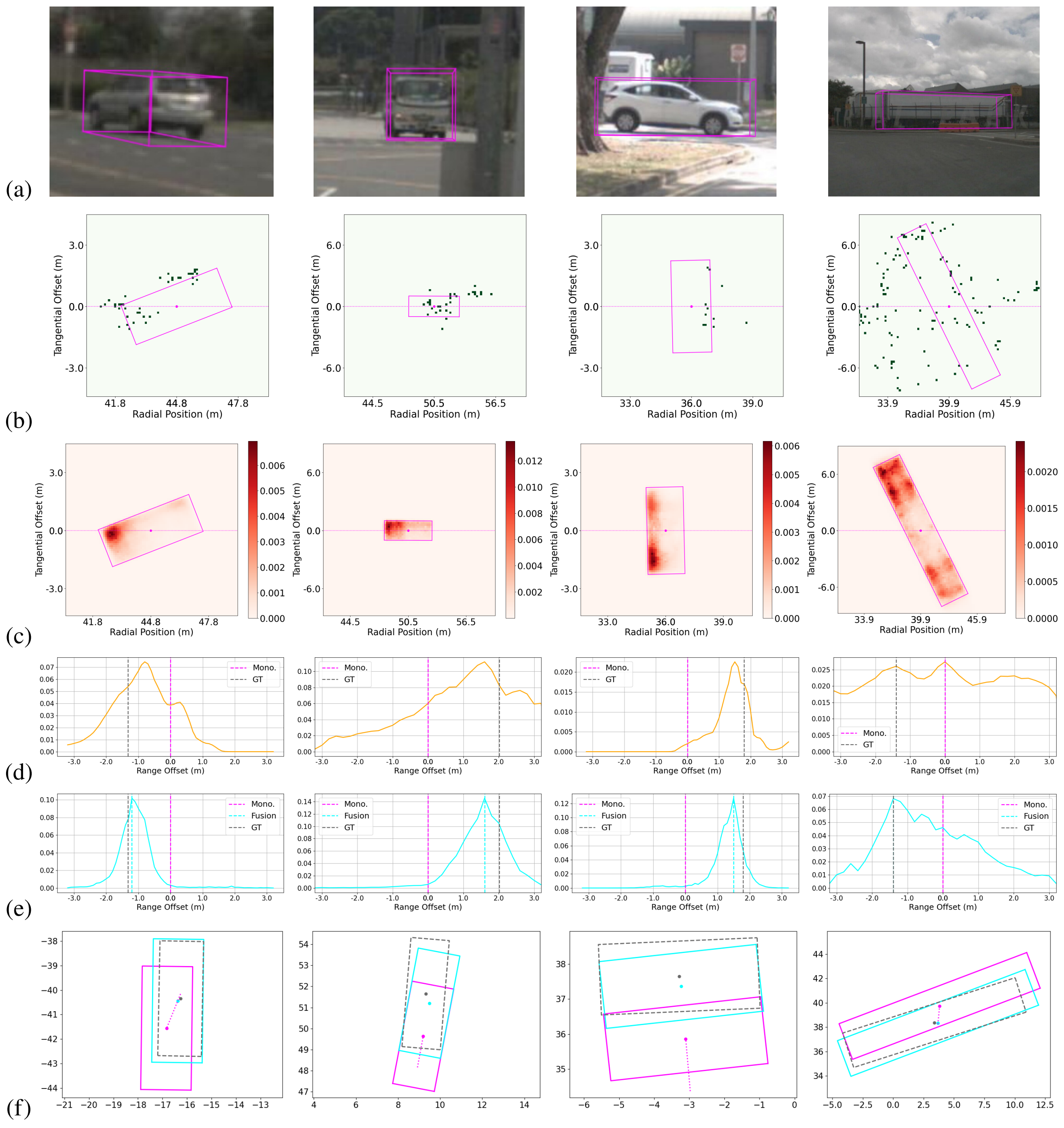


Figure 5.7 Qualitative Results. Visualizations of (a) objects in images, (b) binned radar points, (c) predicted radar hits distribution, (d) Stage-2 matching scores, (e) predicted Stage-3 scores, and (f) detections in ego BEV coordinates. Monocular, RICCARDO, and GT detections are plotted in magenta, blue, and gray, respectively.

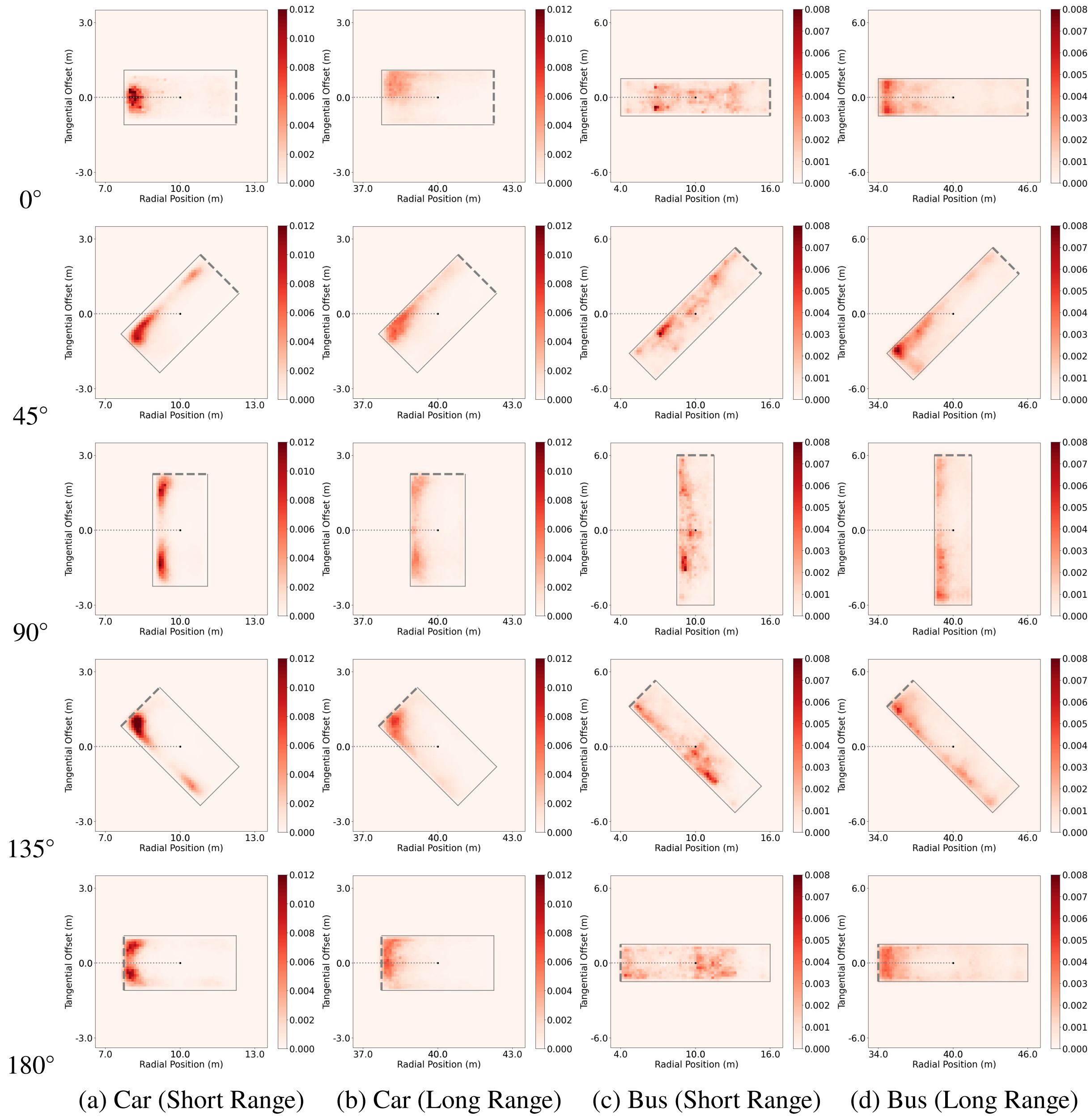
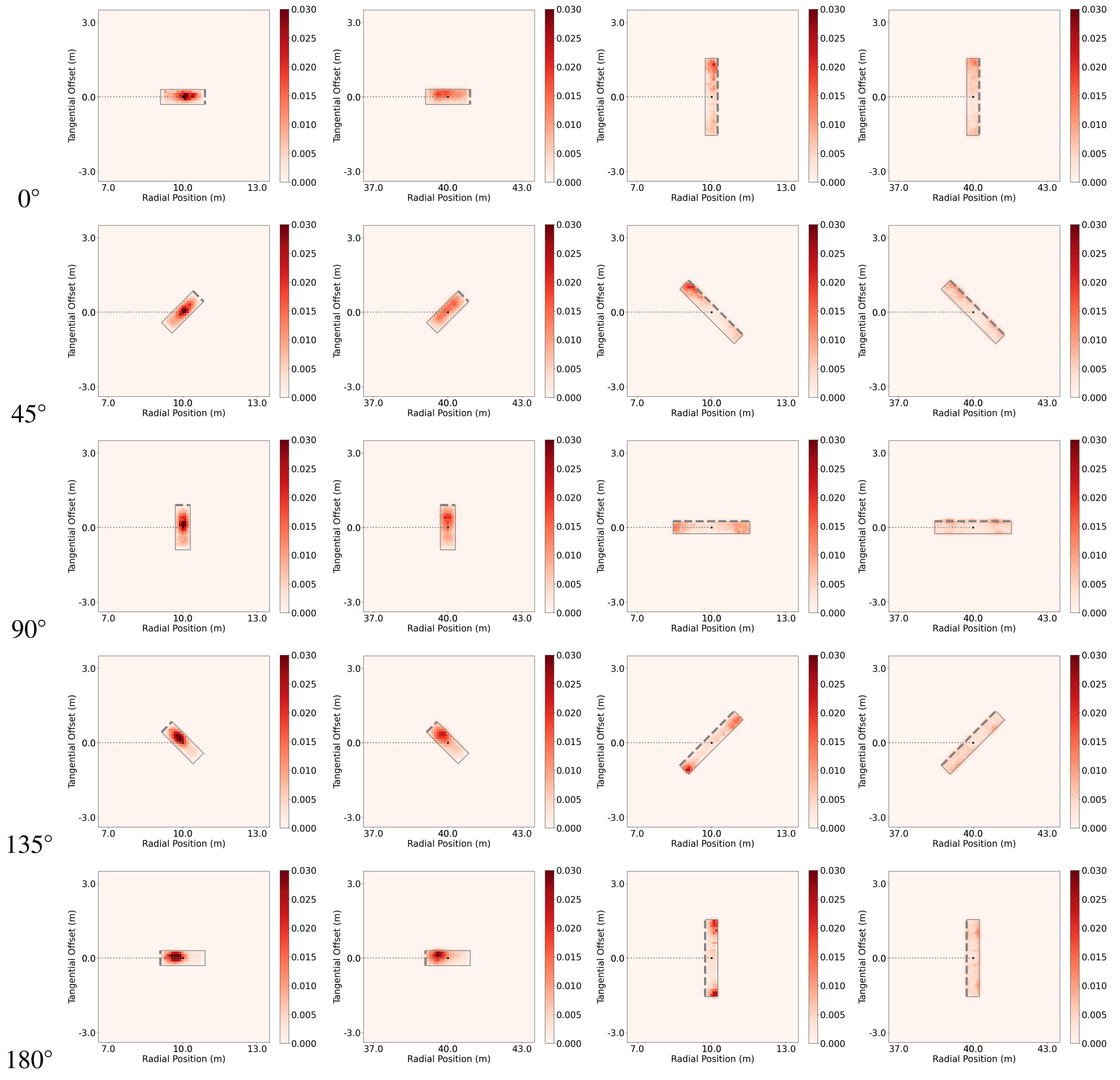


Figure 5.8 Visualization of predicted radar distributions of (a)(b) Car and (c)(d) Bus viewed from different angles and distances of 10 and 40 meters. X-axis represents radial positions, and Y-axis denotes tangential offsets to object centers. Radial rays are plotted as horizontal dotted lines. Target bounding boxes are shown on top of distributions, and dashed lines represent object head.



(a) Bicycle (Short Range)(b) Bicycle (Long Range)(c) Barrier (Short Range)(d) Barrier (Long Range)

Figure 5.9 Visualization of predicted radar distributions of (a)(b) bicycle and (c)(d) barrier viewed from five different angles and from distances of 10 and 40 meters.

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

6.1 Conclusions

This dissertation addresses challenges in radar-camera fusion for depth completion, velocity estimation, and 3D object detection. Hampered by the sparsity of radar returns and the ambiguous association between radar and image pixels, radar-based depth completion is more challenging than lidar-based depth completion. To solve this problem, in Chapter 2, we propose RC-PDA, a novel radar-camera association based on equal depth. We are able to generate enhanced radar depth called MER by densifying raw radar depth with RC-PDA. We demonstrate experimentally that MER leads to more accurate depth completion than raw radar depth. In addition, to train depth completion on nuScenes, we create GT depth by accumulating multiple frames of lidar depth over time. The inability of Doppler radar to directly measure tangential velocity limits its utility in object velocity estimation and radar point accumulation. In chapter 3, we achieve full velocity estimation of radar returns by deriving a closed-form solution combining Doppler velocity from radar with corresponding optical flows from camera. A model supervised with GT bounding box velocities is used to associate radar points to their corresponding optical flows. Experiments show the validity of our method and its application to object velocity estimation and accumulation of moving radar points. Difficulty in fusing radar with monocular detectors includes sparsity of radar points and positional inconsistency between radar hits and corresponding object centers. In Chapter 4, we present RADIANT, a novel scheme to fuse radar with monocular detectors. RADIANT adopts a radar branch, in parallel to an image branch and using mid-level features from both camera and radar as context, to predict 3D offsets from radar hits to object centers. Radar hits updated by the predicted offsets are fused with monocular detected centers to improve object depth estimation. In experiments we apply RADIANT to two monocular detectors FCOS3D and PGD, and improve their detection performance. In chapter 5, to address the problem of inferring object positions from complex radar hits and monocular detections, we propose RICCARDO, an interpretable radar-camera fusion strategy with 3 stages, from the perspective of modeling radar hit distributions in BEV. In stage 1, we train a model to predict BEV radar distributions relative to monocular predicted bounding boxes. In stage 2, we convolve the distributions with actual radar points in the neighborhood to obtain matching scores at binned ranges. In stage 3, we train a model to fuse stage-2 and monocular range predictions. Experiments show RICCARDO improves detection performance over its underlying monocular detector and SOTA radar-camera fusion methods.

6.2 Future Work

6.2.1 Full Velocity Estimation for Objects with Monocular Detector and Radar

In Chapter 3, without using a monocular detector, we estimate point-wise full velocity for radar hits from Doppler velocity and corresponding optical flows. With a monocular detector, we can extend the full velocity estimation from points to objects, by replacing optical flows with motions in detected object centers on neighboring images. In addition, using monocular detectors offers other options to compute full velocity, e.g., velocity from 3D motion of detected objects, Doppler velocity back-projected on object heading, and object velocity directly inferred by monocular models. In future work, it is worthwhile to compare those options and combine them for an optimized estimation.

Typically we assume object velocity is constant within a short period of time around a detection, which is not precise for object turning, accelerating, or decelerating. Thus, in future work, we will consider more sophisticated velocity models, e.g., angular velocity plus linear velocity with acceleration.

6.2.2 Expanding Search Space for Radar Distribution Matching in RICCARDO

RICCARDO in Chapter 5 assumes the underlying monocular detector has accurate estimation in tangential positions, sizes, and orientations and focus on improving range estimation, and also assumes the distribution remain fixed in the search space. To achieve more precise radar distribution matching, it is worthwhile in future work expanding the search space from a 1D (i.e., radial offset) to a high-dimensional space (e.g., radial and tangential offsets, size, and orientation) and adopting a variable distribution as a function of locations in the search space during matching.

6.2.3 3D Object Detection with 4D radar

In the dissertation, we use 2D radar, which suffers from sparsity and a lack of height measurements, and thus radar-camera fusion is necessary for detection. In future work, we plan to design a radar-only detection method by using a 4D radar format. Some recent autonomous driving datasets adopt 4D radar [96], which provides 4D radar tensors, i.e., range-azimuth-elevation-Doppler, as the data format. 4D radar makes it possible for radar-only detection, which is robust to adverse weather. In addition, compared with 2D radar point clouds, 4D radar tensors include one additional dimension, elevation, and are denser with more complete raw information.

BIBLIOGRAPHY

[1] Evan Ackerman. Robot trucks overtake robot cars: This year, trucks will drive themselves on public roads with no one on board. *IEEE Spectrum*, 58(1):42–43, 2020.

[2] Geonho Bang, Kwangjin Choi, Jisong Kim, Dongsuk Kum, and Jun Won Choi. RadarDistill: Boosting radar-based object detection performance via knowledge distillation from lidar features. In *CVPR*, 2024.

[3] 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 *ICRA*, 2020.

[4] Jean-François Bonnefon, Azim Shariff, and Iyad Rahwan. The social dilemma of autonomous vehicles. *Science*, 352(6293):1573–1576, 2016.

[5] Garrick Brazil, Abhinav Kumar, Julian Straub, Nikhila Ravi, Justin Johnson, and Georgia Gkioxari. Omni3D: A large benchmark and model for 3D object detection in the wild. In *CVPR*, 2023.

[6] Garrick Brazil and Xiaoming Liu. M3D-RPN: Monocular 3D region proposal network for object detection. In *ICCV*, 2019.

[7] Garrick Brazil and Xiaoming Liu. Pedestrian detection with autoregressive network phases. In *CVPR*, 2019.

[8] Garrick Brazil, Gerard Pons-Moll, Xiaoming Liu, and Bernt Schiele. Kinematic 3D object detection in monocular video. In *ECCV*, 2020.

[9] Daniel Brodeski, Igal Bilik, and Raja Giryes. Deep radar detector. In *IEEE Radar Conference*, 2019.

[10] 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 *CVPR*, 2020.

[11] Florian Chabot, Mohamed Chaouch, Jaonary Rabarisoa, Céline Teuliere, and Thierry Chateau. Deep MANTA: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image. In *CVPR*, 2017.

[12] Simon Chadwick, Will Maddern, and Paul Newman. Distant vehicle detection using radar and vision. In *ICRA*, 2019.

[13] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3D tracking and forecasting with rich maps. In *CVPR*, 2019.

[14] Shuo Chang, Yifan Zhang, Fan Zhang, Xiaotong Zhao, Sai Huang, Zhiyong Feng, and Zhiqing Wei. Spatial attention fusion for obstacle detection using mmWave radar and vision sensor. *Sensors*, 20(4):956, 2020.

[15] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-DeepLab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *CVPR*, 2020.

[16] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020.

[17] 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 *CVPR*, 2016.

[18] Andreas Danzer, Thomas Griebel, Martin Bach, and Klaus Dietmayer. 2D car detection in radar data with PointNets. In *IEEE Intelligent Transportation Systems Conference*, 2019.

[19] Raul Diaz and Amit Marathe. Soft labels for ordinal regression. In *CVPR*, 2019.

[20] Mingyu Ding, Yuqi Huo, Hongwei Yi, Zhe Wang, Jianping Shi, Zhiwu Lu, and Ping Luo. Learning depth-guided convolutions for monocular 3D object detection. In *CVPR Workshops*, 2020.

[21] Xu Dong, Pengluo Wang, Pengyue Zhang, and Langechuan Liu. Probabilistic oriented object detection in automotive radar. In *CVPR Workshops*, 2020.

[22] Alberto Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

[23] 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*, 22(3):1341–1360, 2020.

[24] Paul Fritsche, Björn Zeise, Patrick Hemme, and Bernardo Wagner. Fusion of radar, lidar and thermal information for hazard detection in low visibility environments. In *IEEE International Symposium on Safety, Security and Rescue Robotics*, 2017.

[25] Huan Fu, Mingming Gong, Chaohui Wang, Kayhan Batmanghelich, and Dacheng Tao. Deep ordinal regression network for monocular depth estimation. In *CVPR*, 2018.

[26] Fernando Garcia, Pietro Cerri, Alberto Broggi, Arturo de la Escalera, and José María Armingol. Data fusion for overtaking vehicle detection based on radar and optical flow. In *IEEE Intelligent Vehicles Symposium*, 2012.

[27] 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.

[28] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012.

[29] Vitor Guizilini, Rares Ambrus, Sudeep Pillai, Allan Raventos, and Adrien Gaidon. 3D packing for self-supervised monocular depth estimation. In *CVPR*, 2020.

[30] Jurgen Hasch. Driving towards 2020: Automotive radar technology trends. In *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility*, 2015.

[31] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[32] Yaoyu Hu, Weikun Zhen, and Sebastian Scherer. Deep-learning assisted high-resolution binocular stereo depth reconstruction. In *ICRA*, 2020.

[33] Saif Imran, Xiaoming Liu, and Daniel Morris. Depth completion with twin surface extrapolation at occlusion boundaries. In *CVPR*, 2021.

[34] Saif Imran, Yunfei Long, Xiaoming Liu, and Daniel Morris. Depth coefficients for depth completion. In *CVPR*, 2019.

[35] Joel Janai, Fatma Guney, Anurag Ranjan, Michael Black, and Andreas Geiger. Unsupervised learning of multi-frame optical flow with occlusions. In *ECCV*, 2018.

[36] Florian Janda, Sebastian Pangerl, Eva Lang, and Erich Fuchs. Road boundary detection for run-off road prevention based on the fusion of video and radar. In *IEEE Intelligent Vehicles Symposium*, 2013.

[37] Maximilian Jaritz, Raoul De Charette, Emilie Wirbel, Xavier Perrotton, and Fawzi Nashashibi. Sparse and dense data with CNNs: Depth completion and semantic segmentation. In *3DV*, 2018.

[38] Zhengping Ji and Danil Prokhorov. Radar-vision fusion for object classification. In *International Conference on Information Fusion*, 2008.

[39] Michael Kampffmeyer, Nanqing Dong, Xiaodan Liang, Yujia Zhang, and Eric P Xing. ConnNet: A long-range relation-aware pixel-connectivity network for salient segmentation. *IEEE Transactions on Image Processing*, 28(5):2518–2529, 2018.

[40] Prannay Kaul, Daniele De Martini, Matthew Gadd, and Paul Newman. RSS-Net: Weakly-supervised multi-class semantic segmentation with FMCW radar. In *IEEE Intelligent Vehi-*

cles Symposium, 2020.

[41] Dominik Kellner, Michael Barjenbruch, Klaus Dietmayer, Jens Klappstein, and Jürgen Dickmann. Instantaneous lateral velocity estimation of a vehicle using Doppler radar. In *International Conference on Information Fusion*, 2013.

[42] Dominik Kellner, Michael Barjenbruch, Jens Klappstein, Jürgen Dickmann, and Klaus Dietmayer. Instantaneous full-motion estimation of arbitrary objects using dual Doppler radar. In *IEEE Intelligent Vehicles Symposium*, 2014.

[43] Youngseok Kim, Jun Won Choi, and Dongsuk Kum. GRIF Net: Gated region of interest fusion network for robust 3D object detection from radar point cloud and monocular image. In *IROS*, 2020.

[44] Youngseok Kim, Sanmin Kim, Jun Won Choi, and Dongsuk Kum. CRAFT: Camera-radar 3D object detection with spatio-contextual fusion transformer. In *AAAI*, 2023.

[45] Youngseok Kim, Juyeb Shin, Sanmin Kim, In-Jae Lee, Jun Won Choi, and Dongsuk Kum. CRN: Camera radar net for accurate, robust, efficient 3D perception. In *ICCV*, 2023.

[46] Abhinav Kumar, Garrick Brazil, Enrique Corona, Armin Parchami, and Xiaoming Liu. DEVIANT: Depth equivariant network for monocular 3D object detection. In *ECCV*, 2022.

[47] Abhinav Kumar, Garrick Brazil, and Xiaoming Liu. GrooMeD-NMS: Grouped mathematically differentiable NMS for monocular 3D object detection. In *CVPR*, 2021.

[48] Abhinav Kumar, Yuliang Guo, Xinyu Huang, Liu Ren, and Xiaoming Liu. SeaBird: Segmentation in bird's view with dice loss improves monocular 3D detection of large objects. In *CVPR*, 2024.

[49] C-C Jay Kuo. Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, 41:406–413, 2016.

[50] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[51] Jin Han Lee, Myung-Kyu Han, Dong Wook Ko, and Il Hong Suh. From big to small: Multi-scale local planar guidance for monocular depth estimation. *arXiv preprint arXiv:1907.10326*, 2019.

[52] Seongwook Lee, Byeong-Ho Lee, Jae-Eun Lee, and Seong-Cheol Kim. Statistical characteristic-based road structure recognition in automotive FMCW radar systems. *IEEE Transactions on Intelligent Transportation Systems*, 20(7):2418–2429, 2018.

[53] Wei Lee, Ljubomir Jovanov, and Wilfried Philips. Semantic-guided radar-vision fusion for

depth estimation and object detection. In *ICCV*, 2021.

[54] Youngwan Lee and Jongyoul Park. CenterMask: Real-time anchor-free instance segmentation. In *CVPR*, 2020.

[55] Kai Lei, Zhan Chen, Shuman Jia, and Xiaoteng Zhang. HVDetFusion: A simple and robust camera-radar fusion framework. *arXiv preprint arXiv:2307.11323*, 2023.

[56] Ang Li, Zejian Yuan, Yonggen Ling, Wanchao Chi, Chong Zhang, et al. A multi-scale guided cascade hourglass network for depth completion. In *WACV*, 2020.

[57] Liang Li and Yuan Xie. A feature pyramid fusion detection algorithm based on radar and camera sensor. In *ICSP*, 2020.

[58] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A Chapman, Dongpu Cao, and Jonathan Li. Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3412–3432, 2020.

[59] You Li and Javier Ibanez-Guzman. Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems. *IEEE Signal Processing Magazine*, 37(4):50–61, 2020.

[60] Zewen Li, Fan Liu, Wenjie Yang, Shouheng Peng, and Jun Zhou. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Transactions on Neural Networks and Learning Systems*, 33(12):6999–7019, 2021.

[61] Jaime Lien, Nicholas Gillian, Emre Karagozler, Patrick Amihood, Carsten Schwesig, Erik Olson, Hakim Raja, and Ivan Poupyrev. Soli: Ubiquitous gesture sensing with millimeter wave radar. *ACM Transactions on Graphics*, 35(4):1–19, 2016.

[62] Teck-Yian Lim, Amin Ansari, Bence Major, Daniel Fontijne, Michael Hamilton, Radhika Gowaikar, and Sundar Subramanian. Radar and camera early fusion for vehicle detection in advanced driver assistance systems. In *NeurIPS Workshops*, 2019.

[63] Juan-Ting Lin, Dengxin Dai, and Luc Van Gool. Depth estimation from monocular images and sparse radar data. In *IROS*, 2020.

[64] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[65] Zhiwei Lin, Zhe Liu, Zhongyu Xia, Xinhao Wang, Yongtao Wang, Shengxiang Qi, Yang Dong, Nan Dong, Le Zhang, and Ce Zhu. RCBEVDet: Radar-camera fusion in bird's eye view for 3D object detection. In *CVPR*, 2024.

[66] Feng Liu and Xiaoming Liu. Voxel-based 3D detection and reconstruction of multiple objects

from a single image. In *NeurIPS*, 2021.

[67] Haisong Liu, Yao Teng, Tao Lu, Haiguang Wang, and Limin Wang. SparseBEV: High-performance sparse 3D object detection from multi-camera videos. In *ICCV*, 2023.

[68] Lijie Liu, Jiwen Lu, Chunjing Xu, Qi Tian, and Jie Zhou. Deep fitting degree scoring network for monocular 3D object detection. In *CVPR*, 2019.

[69] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. SSD: Single shot multibox detector. In *ECCV*, 2016.

[70] Zongdai Liu, Dingfu Zhou, Feixiang Lu, Jin Fang, and Liangjun Zhang. AutoShape: Real-time shape-aware monocular 3D object detection. In *ICCV*, 2021.

[71] Jakob Lombacher, Markus Hahn, Jürgen Dickmann, and Christian Wöhler. Potential of radar for static object classification using deep learning methods. In *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility*, 2016.

[72] Yunfei Long, Abhinav Kumar, Xiaoming Liu, and Daniel Morris. RICCARDO: Radar hit prediction and convolution for camera-radar 3D object detection. In *CVPR*, 2025.

[73] Yunfei Long, Abhinav Kumar, Daniel Morris, Xiaoming Liu, Marcos Castro, and Punarjay Chakravarty. RADIANT: Radar-image association network for 3D object detection. In *AAAI*, 2023.

[74] Yunfei Long and Daniel Morris. Lidar essential beam model for accurate width estimation of thin poles. In *IROS*, 2020.

[75] Yunfei Long, Daniel Morris, Xiaoming Liu, Marcos Castro, Punarjay Chakravarty, and Praveen Narayanan. Full-velocity radar returns by radar-camera fusion. In *ICCV*, 2021.

[76] Yunfei Long, Daniel Morris, Xiaoming Liu, Marcos Castro, Punarjay Chakravarty, and Praveen Narayanan. Radar-camera pixel depth association for depth completion. In *CVPR*, 2021.

[77] Yan Lu, Xinzhu Ma, Lei Yang, Tianzhu Zhang, Yating Liu, Qi Chu, Junjie Yan, and Wanli Ouyang. Geometry uncertainty projection network for monocular 3D object detection. In *ICCV*, 2021.

[78] Fangchang Ma, Guilherme Venturelli Cavalheiro, and Sertac Karaman. Self-supervised sparse-to-dense: Self-supervised depth completion from lidar and monocular camera. In *ICRA*, 2019.

[79] Fangchang Ma and Sertac Karaman. Sparse-to-dense: Depth prediction from sparse depth samples and a single image. In *ICRA*, 2018.

[80] Xinzhu Ma, Shinan Liu, Zhiyi Xia, Hongwen Zhang, Xingyu Zeng, and Wanli Ouyang. Rethinking pseudo-lidar representation. In *ECCV*, 2020.

[81] Xinzhu Ma, Zhihui Wang, Haojie Li, Pengbo Zhang, Wanli Ouyang, and Xin Fan. Accurate monocular 3D object detection via color-embedded 3D reconstruction for autonomous driving. In *ICCV*, 2019.

[82] Xinzhu Ma, Yinmin Zhang, Dan Xu, Dongzhan Zhou, Shuai Yi, Haojie Li, and Wanli Ouyang. Delving into localization errors for monocular 3D object detection. In *CVPR*, 2021.

[83] 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 *ICCV Workshops*, 2019.

[84] Enrique Marti, Miguel Angel de Miguel, Fernando Garcia, and Joshue Perez. A review of sensor technologies for perception in automated driving. *IEEE Intelligent Transportation Systems Magazine*, 11(4):94–108, 2019.

[85] David Metz. The myth of travel time saving. *Transport Reviews*, 28(3):321–336, 2008.

[86] Michael Meyer and Georg Kuschk. Automotive radar dataset for deep learning based 3D object detection. In *European Radar Conference*, 2019.

[87] Michael Meyer and Georg Kuschk. Deep learning based 3D object detection for automotive radar and camera. In *European Radar Conference*, 2019.

[88] Daniel Morris. A pyramid CNN for dense-leaves segmentation. In *Conference on Computer and Robot Vision*, 2018.

[89] Ramin Nabati, Landon Harris, and Hairong Qi. CFTrack: Center-based radar and camera fusion for 3D multi-object tracking. In *Intelligent Vehicles Symposium Workshops*, 2021.

[90] Ramin Nabati and Hairong Qi. RRPN: Radar region proposal network for object detection in autonomous vehicles. In *ICIP*, 2019.

[91] Ramin Nabati and Hairong Qi. CenterFusion: Center-based radar and camera fusion for 3D object detection. In *WACV*, 2021.

[92] Ehsan Nezhadarya, Yang Liu, and Bingbing Liu. BoxNet: A deep learning method for 2D bounding box estimation from bird's-eye view point cloud. In *IEEE Intelligent Vehicles Symposium*, 2019.

[93] Felix Nobis, Maximilian Geisslinger, Markus Weber, Johannes Betz, and Markus Lienkamp.

A deep learning-based radar and camera sensor fusion architecture for object detection. In *Sensor Data Fusion: Trends, Solutions, Applications*, 2019.

[94] Daniel W Otter, Julian R Medina, and Jugal K Kalita. A survey of the usages of deep learning for natural language processing. *IEEE Transactions on Neural Networks and Learning Systems*, 32(2):604–624, 2020.

[95] Arthur Ouaknine, Alasdair Newson, Julien Rebut, Florence Tupin, and Patrick Perez. CARRADA dataset: Camera and automotive radar with range-angle-doppler annotations. In *ICPR*, 2021.

[96] Dong-Hee Paek, Seung-Hyun Kong, and Kevin Tirta Wijaya. K-Radar: 4D radar object detection dataset and benchmark for autonomous driving in various weather conditions. In *NeurIPS*, 2022.

[97] 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.

[98] Aarav Pandya, Ajit Jha, and Linga Reddy Cenkeramaddi. A velocity estimation technique for a monocular camera using mmWave FMCW radars. *Electronics*, 10(19):2397, 2021.

[99] Dennis Park, Rares Ambrus, Vitor Guizilini, Jie Li, and Adrien Gaidon. Is pseudo-lidar needed for monocular 3D object detection? In *ICCV*, 2021.

[100] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019.

[101] Hendrik Purwins, Bo Li, Tuomas Virtanen, Jan Schlüter, Shuo-Yiin Chang, and Tara Sainath. Deep learning for audio signal processing. *IEEE Journal of Selected Topics in Signal Processing*, 13(2):206–219, 2019.

[102] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017.

[103] Jiaxiong Qiu, Zhaopeng Cui, Yinda Zhang, Xingdi Zhang, Shuaicheng Liu, Bing Zeng, and Marc Pollefeys. DeepLiDAR: Deep surface normal guided depth prediction for outdoor scene from sparse LiDAR data and single color image. In *CVPR*, 2019.

[104] Pierluigi Zama Ramirez, Matteo Poggi, Fabio Tosi, Stefano Mattoccia, and Luigi Di Stefano. Geometry meets semantics for semi-supervised monocular depth estimation. In *ACCV*, 2018.

[105] Cody Reading, Ali Harakeh, Julia Chae, and Steven Waslander. Categorical depth distribution network for monocular 3D object detection. In *CVPR*, 2021.

[106] Konstantinos Rematas, Ira Kemelmacher-Shlizerman, Brian Curless, and Steve Seitz. Soccer on your tabletop. In *CVPR*, 2018.

[107] Haoyu Ren, Aman Raj, Mostafa El-Khamy, and Jungwon Lee. SUW-Learn: Joint supervised, unsupervised, weakly supervised deep learning for monocular depth estimation. In *CVPR Workshops*, 2020.

[108] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *MICCAI*, 2015.

[109] Fabian Roos, Dominik Kellner, Jürgen Dickmann, and Christian Waldschmidt. Reliable orientation estimation of vehicles in high-resolution radar images. *IEEE Transactions on Microwave Theory and Techniques*, 64(9):2986–2993, 2016.

[110] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[111] Ashutosh Saxena, Justin Driemeyer, and Andrew Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008.

[112] Nicolas Scheiner, Nils Appenrodt, Jürgen Dickmann, and Bernhard Sick. A multi-stage clustering framework for automotive radar data. In *IEEE Intelligent Transportation Systems Conference*, 2019.

[113] Nicolas Scheiner, Florian Kraus, Fangyin Wei, Buu Phan, Fahim Mannan, Nils Appenrodt, Werner Ritter, Jurgen Dickmann, Klaus Dietmayer, Bernhard Sick, et al. Seeing around street corners: Non-line-of-sight detection and tracking in-the-wild using Doppler radar. In *CVPR*, 2020.

[114] Johannes Schlichenmaier, Fabian Roos, Philipp Hügler, and Christian Waldschmidt. Clustering of closely adjacent extended objects in radar images using velocity profile analysis. In *IEEE MTT-S International Conference on Microwaves for Intelligent Mobility*, 2019.

[115] Mehmet Saygın Seyfioğlu, Ahmet Murat Özbayoğlu, and Sevgi Zubeyde Gürbüz. Deep convolutional autoencoder for radar-based classification of similar aided and unaided human activities. *IEEE Transactions on Aerospace and Electronic Systems*, 54(4):1709–1723, 2018.

[116] Meet Shah, Zhiling Huang, Ankit Laddha, Matthew Langford, Blake Barber, Sidney Zhang, Carlos Vallespi-Gonzalez, and Raquel Urtasun. LiRaNet: End-to-end trajectory prediction using spatio-temporal radar fusion. In *CoRL*, 2020.

[117] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *CVPR*, 2019.

[118] Xian Shuai, Yulin Shen, Yi Tang, Shuyao Shi, Luping Ji, and Guoliang Xing. milliEye: A lightweight mmWave radar and camera fusion system for robust object detection. In *International Conference on Internet-of-Things Design and Implementation*, 2021.

[119] Heonkyo Sim, The-Duong Do, Seongwook Lee, Yong-Hwa Kim, and Seong-Cheol Kim. Road environment recognition for automotive FMCW radar systems through convolutional neural network. *IEEE Access*, 8:141648–141656, 2020.

[120] Andrea Simonelli, Samuel Bulò, Lorenzo Porzi, Manuel Antequera, and Peter Kontschieder. Disentangling monocular 3D object detection: From single to multi-class recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3):1219–1231, 2020.

[121] Andrea Simonelli, Samuel Bulò, Lorenzo Porzi, Peter Kontschieder, and Elisa Ricci. Are we missing confidence in pseudo-lidar methods for monocular 3D object detection? In *ICCV*, 2021.

[122] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[123] Akash Deep Singh, Yunhao Ba, Ankur Sarker, Howard Zhang, Achuta Kadambi, Stefano Soatto, Mani Srivastava, and Alex Wong. Depth estimation from camera image and mmWave radar point cloud. In *CVPR*, 2023.

[124] Arvind Srivastav and Soumyajit Mandal. Radars for autonomous driving: A review of deep learning methods and challenges. *IEEE Access*, 11:97147–97168, 2023.

[125] Leo Stanislas and Thierry Peynot. Characterisation of the Delphi electronically scanning radar for robotics applications. In *Australasian Conference on Robotics and Automation*, 2015.

[126] 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 *CVPR*, 2020.

[127] Xiaolin Tang, Zhiqiang Zhang, and Yechen Qin. On-road object detection and tracking based on radar and vision fusion: A review. *IEEE Intelligent Transportation Systems Magazine*, 14(5):103–128, 2021.

[128] Yunlei Tang, Sebastian Dorn, and Chiragkumar Savani. Center3D: Center-based monocular 3D object detection with joint depth understanding. In *DAGM German Conference on Pattern Recognition*, 2020.

[129] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *ECCV*, 2020.

[130] Julian Theis, Daniel Ossmann, Frank Thielecke, and Harald Pfifer. Robust autopilot design for landing a large civil aircraft in crosswind. *Control Engineering Practice*, 76:54–64, 2018.

[131] Jonas Uhrig, Nick Schneider, Lukas Schneider, Uwe Franke, Thomas Brox, and Andreas Geiger. Sparsity invariant CNNs. In *3DV*, 2017.

[132] Jorge Vargas, Suleiman Alsweiss, Onur Toker, Rahul Razdan, and Joshua Santos. An overview of autonomous vehicles sensors and their vulnerability to weather conditions. *Sensors*, 21(16):5397, 2021.

[133] Sudheendra Vijayanarasimhan, Susanna Ricco, Cordelia Schmid, Rahul Sukthankar, and Katerina Fragkiadaki. SfM-Net: Learning of structure and motion from video. *arXiv preprint arXiv:1704.07804*, 2017.

[134] Li Wang, Li Zhang, Yi Zhu, Zhi Zhang, Tong He, Mu Li, and Xiangyang Xue. Progressive coordinate transforms for monocular 3D object detection. In *NeurIPS*, 2021.

[135] Panqu Wang, Pengfei Chen, Ye Yuan, Ding Liu, Zehua Huang, Xiaodi Hou, and Garrison Cottrell. Understanding convolution for semantic segmentation. In *WACV*, 2018.

[136] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. FCOS3D: Fully convolutional one-stage monocular 3d object detection. In *ICCV Workshops*, 2021.

[137] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Probabilistic and geometric depth: Detecting objects in perspective. In *CoRL*, 2021.

[138] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3D object detection for autonomous driving. In *CVPR*, 2019.

[139] Yingjie Wang, Qiuyu Mao, Hanqi Zhu, Yu Zhang, Jianmin Ji, and Yanyong Zhang. Multi-modal 3D object detection in autonomous driving: A survey. *arXiv preprint arXiv:2106.12735*, 2021.

[140] 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.

[141] Yue Wang, Vitor Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. DETR3D: 3D object detection from multi-view images via 3D-to-2D queries. In *CoRL*,

2021.

[142] Jamie Watson, Michael Firman, Gabriel J Brostow, and Daniyar Turmukhambetov. Self-supervised monocular depth hints. In *ICCV*, 2019.

[143] Philipp Wolters, Johannes Gilg, Torben Teepe, Fabian Herzog, Anouar Laouichi, Martin Hofmann, and Gerhard Rigoll. Unleashing HyDRa: Hybrid fusion, depth consistency and radar for unified 3D perception. *arXiv preprint arXiv:2403.07746*, 2024.

[144] Yutian Wu, Yueyu Wang, Shuwei Zhang, and Harutoshi Ogai. Deep 3D object detection networks using lidar data: A review. *IEEE Sensors Journal*, 21(2):1152–1171, 2020.

[145] Zizhang Wu, Guilian Chen, Yuanzhu Gan, Lei Wang, and Jian Pu. MVFusion: Multi-view 3D object detection with semantic-aligned radar and camera fusion. In *ICRA*, 2023.

[146] Qiangeng Xu, Yin Zhou, Weiyue Wang, Charles Qi, and Dragomir Anguelov. SPG: Unsupervised domain adaptation for 3D object detection via semantic point generation. In *ICCV*, 2021.

[147] Yan Xu, Xinge Zhu, Jianping Shi, Guofeng Zhang, Hujun Bao, and Hongsheng Li. Depth completion from sparse lidar data with depth-normal constraints. In *ICCV*, 2019.

[148] Ritu Yadav, Axel Vierling, and Karsten Berns. Radar+RGB fusion for robust object detection in autonomous vehicle. In *ICIP*, 2020.

[149] Bin Yang, Runsheng Guo, Ming Liang, Sergio Casas, and Raquel Urtasun. RadarNet: Exploiting radar for robust perception of dynamic objects. In *ECCV*, 2020.

[150] Zhenheng Yang, Peng Wang, Wei Xu, Liang Zhao, and Ramakant Nevatia. Unsupervised learning of geometry with edge-aware depth-normal consistency. *arXiv preprint arXiv:1711.03665*, 2017.

[151] Shanliang Yao, Runwei Guan, Xiaoyu Huang, Zhuoxiao Li, Xiangyu Sha, Yong Yue, Eng Gee Lim, Hyungjoon Seo, Ka Lok Man, Xiaohui Zhu, et al. Radar-camera fusion for object detection and semantic segmentation in autonomous driving: A comprehensive review. *IEEE Transactions on Intelligent Vehicles*, 9(1):2094–2128, 2023.

[152] De Jong Yeong, Gustavo Velasco-Hernandez, John Barry, and Joseph Walsh. Sensor and sensor fusion technology in autonomous vehicles: A review. *Sensors*, 21(6):2140, 2021.

[153] Richard W Young. Evolution of the human hand: the role of throwing and clubbing. *Journal of Anatomy*, 202(1):165–174, 2003.

[154] Georgios Zamanakos, Lazaros Tsochatzidis, Angelos Amanatiadis, and Ioannis Pratikakis. A comprehensive survey of lidar-based 3D object detection methods with deep learning for

autonomous driving. *Computers & Graphics*, 99:153–181, 2021.

[155] Diankun Zhang, Zhijie Zheng, Haoyu Niu, Xueqing Wang, and Xiaojun Liu. Fully sparse transformer 3-D detector for lidar point cloud. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–12, 2023.

[156] Hongyang Zhang, Junru Shao, and Ruslan Salakhutdinov. Deep neural networks with multi-branch architectures are intrinsically less non-convex. In *The 22nd International Conference on Artificial Intelligence and Statistics*, 2019.

[157] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3212–3232, 2019.

[158] Taohua Zhou, Junjie Chen, Yining Shi, Kun Jiang, Mengmeng Yang, and Diange Yang. Bridging the view disparity between radar and camera features for multi-modal fusion 3D object detection. *IEEE Transactions on Intelligent Vehicles*, 8(2):1523–1535, 2023.

[159] Taohua Zhou, Mengmeng Yang, Kun Jiang, Henry Wong, and Diange Yang. MMW radar-based technologies in autonomous driving: A review. *Sensors*, 20(24):7283, 2020.

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

[161] Yunsong Zhou, Yuan He, Hongzi Zhu, Cheng Wang, Hongyang Li, and Qinhong Jiang. MonoEF: Extrinsic parameter free monocular 3D object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(12):10114–10128, 2021.

[162] Li Zhu, Fei Richard Yu, Yige Wang, Bin Ning, and Tao Tang. Big data analytics in intelligent transportation systems: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 20(1):383–398, 2018.

[163] Shengjie Zhu, Garrick Brazil, and Xiaoming Liu. The edge of depth: Explicit constraints between segmentation and depth. In *CVPR*, 2020.