8-2021

Sensor Fusion for Object Detection and Tracking in Autonomous Vehicles

Mohamad Ramin Nabati
University of Tennessee, Knoxville, mnabati@vols.utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Mohamad Ramin Nabati entitled "Sensor Fusion for Object Detection and Tracking in Autonomous Vehicles." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Engineering.

Hairong Qi, Major Professor

We have read this dissertation and recommend its acceptance:

Hairong Qi, Jens Gregor, Amir Sadovnik, David Irick

Accepted for the Council:

Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Sensor Fusion for Object Detection and Tracking in Autonomous Vehicles

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Ramin Nabati

August 2021

Acknowledgments

My journey through this dissertation would not have been possible without the encouragement and support of several individuals. First and foremost, I would like to express my deepest appreciation to my advisor and dissertation chair, Dr. Hairong Qi, for taking me in as a student and offering support when I needed it most. Thank you for your patience, invaluable advice and unwavering support throughout my PhD study. The perfect balance of guidance and freedom you provided me immensely helped shape my research and future career.

To my committee member and mentor, Dr. David Irick, from you I learned hard work, discipline and dedication. I'm forever grateful for your support, encouragement and mentorship over the past four years.

I would like to thank my committee members, Dr. Jens Gregor and Dr. Amir Sadovnik for their guidance, constructive comments and suggestions. I would also like to express my gratitude to Landon Harris for spending countless hours with me brainstorming problems and for making contributions important to this study.

Lastly, I'm deeply grateful to my parents and my wife Minoo, who have always encouraged me to seek new directions in life. Thank you for your unconditional love and support throughout the years, and for helping me to be the person I am today. This journey would not have been possible if not for you.

Abstract

Autonomous driving vehicles depend on their perception system to understand the environment and identify all static and dynamic obstacles surrounding the vehicle. The perception system in an autonomous vehicle uses the sensory data obtained from different sensor modalities to understand the environment and perform a variety of tasks such as object detection and object tracking. Combining the outputs of different sensors to obtain a more reliable and robust outcome is called sensor fusion. This dissertation studies the problem of sensor fusion for object detection and object tracking in autonomous driving vehicles and explores different approaches for utilizing deep neural networks to accurately and efficiently fuse sensory data from different sensing modalities.

In particular, this dissertation focuses on fusing radar and camera data for 2D and 3D object detection and object tracking tasks. First, the effectiveness of radar and camera fusion for 2D object detection is investigated by introducing a radar region proposal algorithm for generating object proposals in a two-stage object detection network. The evaluation results show significant improvement in speed and accuracy compared to a vision-based proposal generation method. Next, radar and camera fusion is used for the task of joint object detection and depth estimation where the radar data is used in conjunction with image features to generate object proposals, but also provides accurate depth estimation for the detected objects in the scene. A fusion algorithm is also proposed for 3D object detection where where the depth and velocity data obtained from the radar is fused with the camera images to detect objects in 3D and also accurately estimate their velocities without requiring any temporal information. Finally, radar and camera sensor fusion is used for 3D multi-object tracking by introducing an end-to-end trainable and online network capable of tracking objects in real-time.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Autonomous driving vehicles are intelligent agents that need to perceive the environment, predict and decide about other agents' behavior, plan their course of action and execute their decisions in an complex and uncontrolled environment [25]. This complex system is comprised of many interconnected components and processes. The core components in autonomous driving vehicles can be categorized into four subsystems [44]: perception, localization and mapping, path planning and vehicle control, as illustrated in Fig. 1.1. Perception is the first and arguably the most important step in this process, which is responsible for understanding the environment surrounding the vehicle, including identifying all static and dynamic obstacles. The perception system itself can be broken down into different modules designed to accomplish specific tasks such as object detection, object tracking, road detection and traffic sign recognition.

All these modules need to be accurate and robust and at the same time efficient enough to run in real-time. To achieve these requirements, autonomous vehicles usually take advantage of a variety of different sensors, including cameras, radars and LIDARs. The perception system is responsible for transforming the raw sensory data obtained from these sensors into semantic information about the environment. Having multiple sensing modalities improves both accuracy and robustness, as any type of sensor suffers from different limitations that degrades its performance. Additionally, the environmental conditions such as snow, rain and bright sunlight greatly affect some sensors' ability to operate [23]. Using a combination of different sensors enables the perception system to exploit the complementary features of

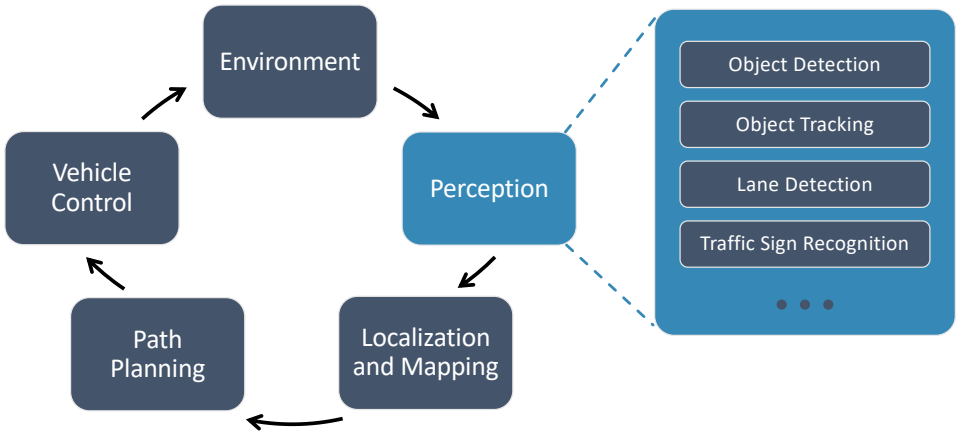


Figure 1.1: Autonomous driving subsystems and the tasks in the perception subsystem.

different sensing modalities, and compensate for the limitations of individual sensors. The process of merging the information obtained from multiple sensors to reduce uncertainty is called sensor fusion.

This dissertation aims to utilize deep learning for developing sensor fusion algorithms for autonomous driving applications. It focuses on two of the most important tasks in the perception system: object detection and object tracking.

1.1 Object Detection

Object detection is the task of determining whether or not instances of predefined object classes are present in an image, and if so, determining the spatial location and extent of each object instance [50]. Object detection has been a fundamental and challenging problem in computer vision for many years, and it's widely accepted that its progress has gone through two important and historical periods [113]. In the traditional object detection period (before 2014), the mainstream approach was to extract local features from the image and apply a machine learning method for recognition [26]. Most algorithms in this period were built based on methods for extracting hand-crafted features such as Histogram of Oriented Gradients (HOG) [16] and Scale-Invariant Feature Transform (SIFT) [54] from the image. Object detection reached a plateau after 2010, with the performance of hand-crafted features becoming saturated [113]. In 2012, a deep Convolutional Neural Network (CNN) was used for image classification [41] and in 2014 the first CNN based object detection algorithm was introduced by R. Girshick [29]. This started the second historic period and object detection has evolved at an unprecedented speed since.

2D object detection has seen a significant progress over the past few years, resulting in very accurate and efficient algorithms mostly based on CNNs [27, 15, 73, 51]. These methods are generally based on anchors and fall under one of the two main categories of two-stage or one-stage methods. Two-stage algorithms such as [27, 73] are comprised of two networks in their pipeline. The first network is designed to generates object candidates called region proposals or Region of Interest (RoI) using the features extracted from the image. The RoIs are then processed, refined and classified in the second network to obtain

the final detection results. In the one-stage detection category on the other hand, the RoI generation stage is skipped and features obtained from the image are directly processed to generate the final detection results including the location and category of the objects. These algorithms rely on a set of pre-defined anchor boxes and treat the object detection task as a regression problem, learning the class probabilities and bounding box coordinates directly from the image features [81]. YOLO [71] and SSD [51] are among the most popular one-stage detection algorithms. The two-stage methods usually achieve a higher accuracy compared to the one-stage methods, but they require more processing, storage space and inference time.

Although the anchor-based methods have been very popular, they suffer from several drawbacks. The anchors (generated or pre-defined) could introduce a class imbalance problem if most of them are only covering the background in the image. Additionally, using anchors introduces more hyper-parameters and computation to the system, which could results in longer training and inference times [96]. More recently, anchor-free detectors [52, 19, 39] have been proposed to improve the efficiency by eliminating the anchors altogether and regressing the location of the objects directly.

Object detection for autonomous driving applications presents unique challenges such as real-time requirements and memory limitations. The embedded processors used in autonomous vehicles also have processing limitations that renders many existing object detection methods not suitable for autonomous driving scenarios. Addressing these limitations requires careful design and implementation of object detection algorithms that are tailored to the specific needs of this particular application.

1.2 Object Tracking

Object tracking is the task of continuously estimating the trajectory of an object based on measurements obtained from one or multiple sensors. Object tracking algorithms can be divided into two different categories: Single Object Tracking (SOT) and Multi Object Tracking (MOT). In SOT the tracker is responsible for tracking a single object whose appearance is known a-priori. This object is provided in the first frame and needs to be tracked in all the subsequent frames. In MOT on the other hand, a detection step is necessary

to identify all objects with certain categories, and track them individually without any prior knowledge of the appearance or number of such objects [14]. This is a significantly more challenging task, as several factors such as object occlusion and objects with similar appearances might impede the tracking process.

Many algorithms have been developed in recent years to address these issues. The majority of these algorithms exploit the rich representational power of Deep Neural Network (DNN) to extract complex semantic features from the input. Tracking-by-detection is a common approach used in these algorithms, where the tracking problem is solved by breaking it into two steps: (1) detecting objects in each image, (2) associating the detected objects over time. Recently, the CNN-based object detection networks have been very successful in improving the performance in this task. As a result, many of the MOT methods adopt an existing detection method as is and focus more on improving the association step.

The first deep learning based approaches [88, 87, 86] mostly took advantage of the powerful feature representation capabilities of DNNs to better model the tracked objects, and then used traditional methods for classification and association [40]. Other methods utilized DNNs for solving the correspondence problem as well [82, 84] by directly learning the similarities between two object match candidates. Some methods go even further and propose end-to-end DNNs to optimize the entire tracking process [65, 58]. This can potentially boost the performance of the tracking algorithm, since the entire tracking pipeline which includes object representation, object detection and location prediction is jointly optimized in the network [40].

Tracking of dynamic objects surrounding the vehicle is essential for many of the tasks crucial to autonomous navigation, such as path planning and obstacle avoidance [70]. Tracking objects over time also makes it possible to estimate variables that are not directly observed by the sensors, such as velocity and acceleration of objects [40]. This information could be particularly useful in autonomous driving for identifying critical situations, as illustrated in Fig. 1.2. Tracking methods developed for autonomous vehicles use a variety of sensory data such as RGB images from vision sensors, point clouds from range sensors such as LIDARs and radars, or a combination of them as inputs.

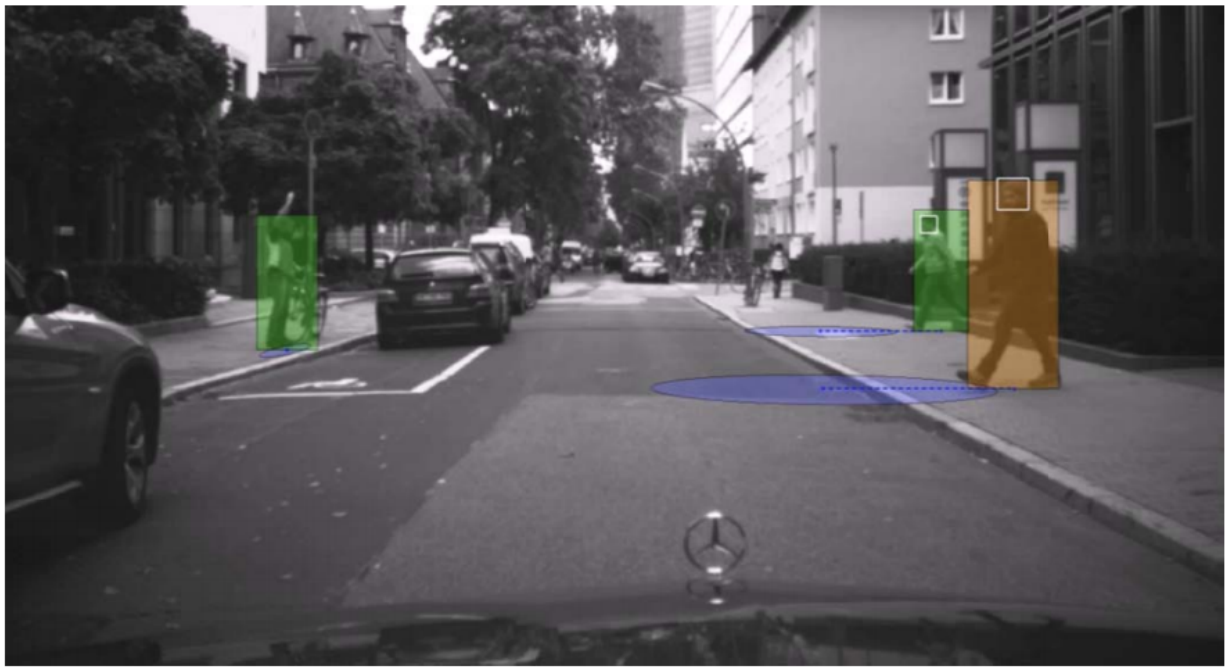


Figure 1.2: Using the objects' kinematic information to predict their location in future [40].

Incorporating the multi-modal sensory data into an object tracking framework for autonomous driving applications is not a trivial task. It requires an efficient, accurate and reliable fusion algorithm capable of utilizing the information embedded in different modalities in real time. Most multi-modal MOT methods use multiple sensing modalities in the detection stage, but only utilize features from one sensing modality in the association step. In addition, many existing MOT methods rely only on camera images [108, 111] or LIDAR point clouds [13, 80] for detection and tracking.

1.3 Sensor Fusion

Sensor fusion is the task of combining the outputs of individual sensors to produce a new outcome that is more reliable and more robust than the those generated by each individual sensor [23]. In recent years, sensor fusion has been applied in many different applications such as 2D and 3D object detection [12, 42, 47, 60], semantic segmentation [105, 56] and object tracking [2, 22]. Many of these methods take advantage of the hierarchical feature representation in neural networks to effectively combine multiple sensing modalities.

Most sensor fusion methods in the literature can be categorized into early, middle and late fusion categories. In an early fusion approach, the raw or pre-processed sensory data from different sensors are fused together, enabling the network to learn a joint representation from different sensing modalities. These methods generally have a lower computation requirements, but could be very sensitive to spatial or temporal misalignment of the input data [25]. On the other hand, a late fusion approach combines the data from different modalities at the decision level and provides more flexibility for introducing new sensing modalities to the network. However, a late fusion approach does not exploit the full potential of the available sensing modalities, as it does not acquire the intermediate features obtained by learning a joint representation. A compromise between the early and late fusion approaches is referred to as middle fusion. In this approach, features from different modalities are first extracted individually and then combined at an intermediate stage, enabling the network to learn joint representations and creating a balance between sensitivity and flexibility. Different sensor fusion categories are illustrated in Fig. 1.3.

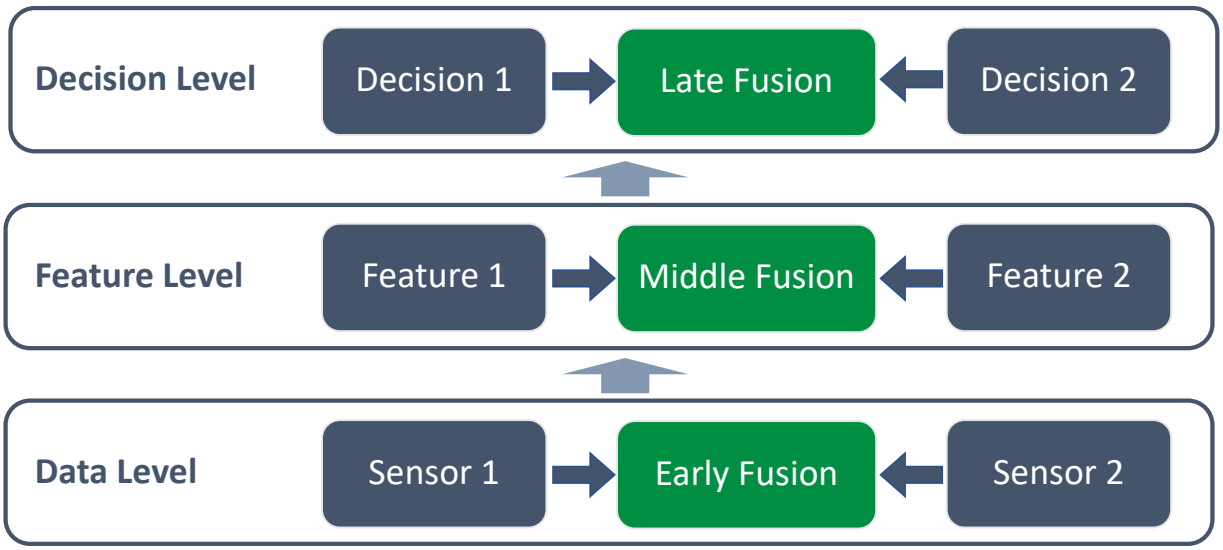


Figure 1.3: Different sensor fusion levels [23].

Although fusion helps with building a more accurate and robust perception system, it also adds to the complexity and processing requirements of the system, making the design of real-time sensor fusion algorithms very challenging. This is particularly important in applications such as autonomous driving, where operation in real-time is a requirement.

1.4 Motivation and Contribution

This dissertation focuses on sensor fusion for object detection and object tracking in autonomous driving applications, with an emphasis on utilizing radars and cameras in the process. The selection of the proper group of sensors is one of the major considerations in designing the perception system in an autonomous vehicle. Although many studies have been conducted on fusing different sensing modalities for autonomous driving applications, most recent works focus on exploiting LIDARs and cameras for object detection and object tracking applications.

LIDARs use the time of flight of laser light pulses to calculate distance to surrounding objects. They provide accurate 3D measurements at close range, but the resulting point cloud becomes sparse at long range, reducing the system's ability to accurately detect far away objects. Additionally, due to their unstructured nature, processing point clouds obtained from LIDARs is both challenging and computationally expensive. LIDARs are also very expensive compared to other sensors such as radars or cameras. Vision cameras are one of the essential sensors used in autonomous driving applications, generating a high resolution view of the environment surrounding the vehicle. Cameras are relatively inexpensive and are very effective for tasks such as object detection and classification, but they do not provide any depth information about the environment. To address this problem, two or more cameras could be used in a stereo camera setup to estimate depth [23]. This solution, however, is usually less accurate than dedicated depth sensors such as LIDARs, and is susceptible to calibration errors.

The complementary features of LIDARs and cameras have made LIDAR-camera sensor fusion a topic of interest in recent years. This combination has been proven to achieve high accuracy in 3D object detection for many applications including autonomous driving,

but it has some limitations. Cameras and LIDARs are both sensitive to adverse weather conditions such as snow, fog and rain, which can significantly reduce their field of view and sensing capabilities. Additionally, LIDARs and cameras are not capable of detecting objects' velocity without using temporal information. Estimating objects' velocity is an important requirement for collision avoidance in many scenarios, and relying on the temporal information might not be a feasible solution in time-critical situations.

For many years, radars have been used in vehicles for Advanced Driving Assistance System (ADAS) applications such as collision avoidance and Adaptive Cruise Control (ACC). Radars operate by measuring the reflection of radio waves from objects, and use the Doppler effect to estimate objects' velocity without requiring any temporal information. While objects' velocity information is extremely useful in tasks such as object tracking, as it can be used for predicting objects' path and displacement. Radars are capable of detecting objects at much longer range compared to LIDARs and cameras (up to 200 meters for automotive radars), while being very robust to adverse weather conditions. Compared to LIDARs and cameras, the raw data obtained from radars require less processing before they can be used as object detection results. This is because of the internal processing performed on the data in automotive radars, which filters out most of the noise and duplicate detections and also provided useful features such as relative object speed, detection validity probability and stationary or moving classification for each detected object. In contrast, LIDAR point clouds require significant processing to extract semantic information about objects in the scene. Radars, however, are not particularly good at classifying objects. This makes the fusion of radar and other sensors such as cameras a very interesting topic in autonomous driving applications.

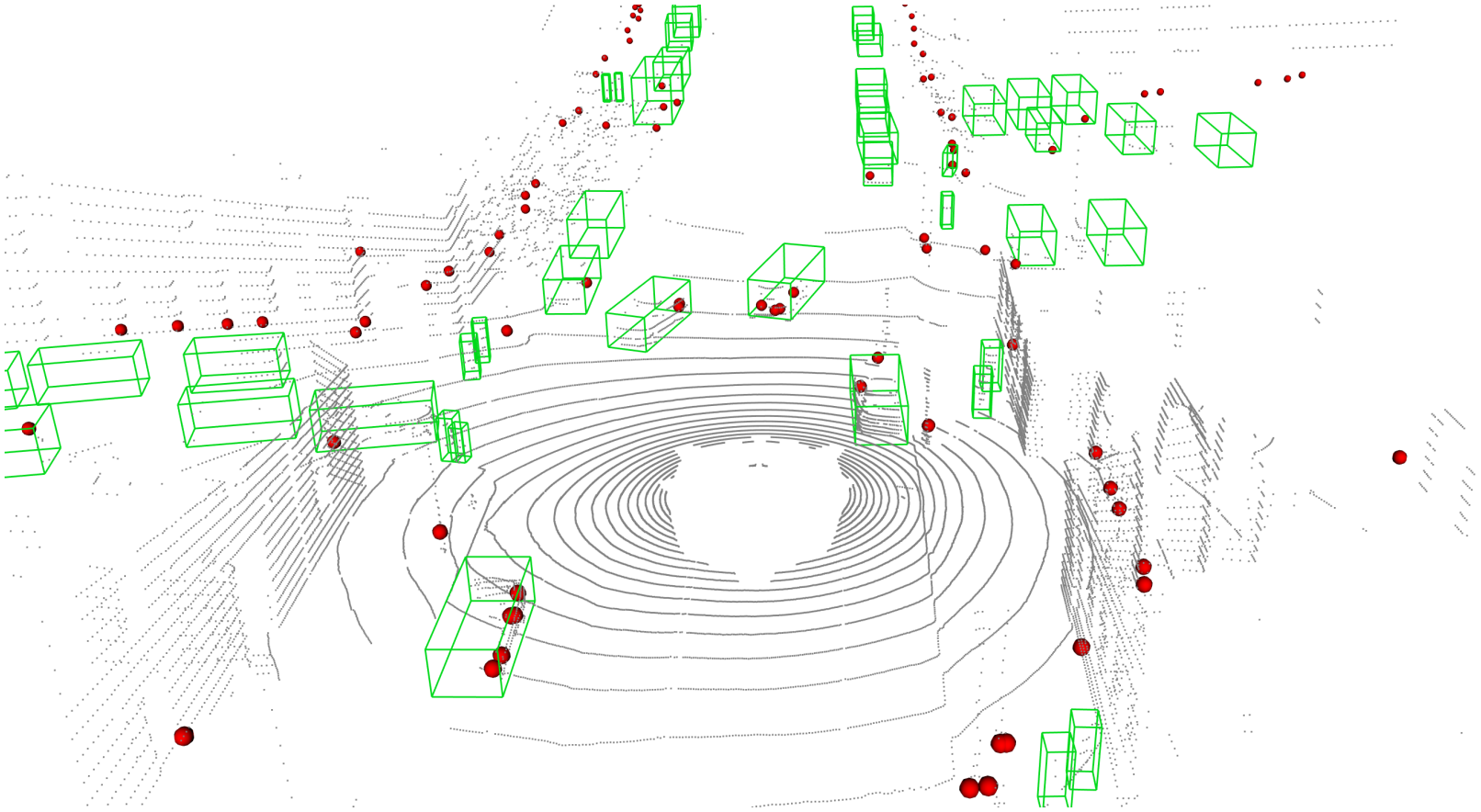
Despite radars' popularity in the automotive industry, few studies have focused on fusing radar data with other sensing modalities. One reason for this is the fact that there are not many publicly available datasets containing radar data for autonomous driving applications, which makes conducting research in this area difficult. On the other hand there are many large-scale datasets containing LIDAR and camera data, which has resulted in significantly more studies focusing on LIDAR point clouds processing and fusion with camera data. Due to inherent differences between LIDARand radar point clouds, applying or adapting

existing LIDAR-based feature extraction and classification algorithms to radar point clouds is impractical, if not infeasible. Many of the existing point cloud processing methods first project the points to different views or use voxels to represent it in a compact form. 2D or 3D convolutional networks are then used to extract features. Other methods extract features from the raw point clouds directly using networks such as PointNet [69]. All of these methods are usually designed for dense LIDARpoint clouds and do not perform well on sparse point clouds. Radar point clouds are significantly more sparse compared to LIDARpoint clouds. For a single object, an ideal radar only reports one point, compared to tens or hundreds of points obtained from a LIDAR for the same object. This sparsity makes it impractical to extract geometry information about the objects in the scenes. Aggregating multiple radar readings obtained in different time-stamps can help provide more points in the point cloud, but these points are usually not a good representation of the objects' shape and size. This aggregation also introduces latency in the system. Additionally, most automotive radars do not provide any height information for the detected objects, essentially making the radar point clouds a 2-dimensional signal, as opposed to the 3-dimensional point clouds obtained from LIDARs. Fig. 1.4 visualizes some of the differences between radar and LIDAR point clouds.

This dissertation investigates fusing radar and camera data for object detection and object tracking in autonomous driving applications. In particular, several fusion algorithms are proposed to perform 2D and 3D object detection, depth estimation and object tracking based on DNNs. These algorithms provide insight into the effectiveness of radars in improving both accuracy and run time of object detection and tracking tasks when used in conjunction with cameras.

1.5 Dataset

The NuScenes [8] dataset is extensively used in this dissertation to train and evaluate the proposed algorithms. NuScenes is a large-scale and publicly available dataset for autonomous driving, featuring a full sensor suite including radar, camera, LIDAR, GPS and IMU. With 3D ground truth annotations for 25 object classes and 1.4M radar sweeps, NuScenes is the



(a)

Figure 1.4: Sample data from the NuScenes dataset showing radar point cloud (red), 3D ground truth boxes (green) and LIDAR point cloud (grey).

first large-scale dataset to publicly provide synchronized and annotated camera and radar data collected in highly challenging driving situations. The dataset includes images from 6 different cameras around the vehicle and radar detections from four corner radars and one front-facing radar. The NuScenes dataset is organized into 1000 different scenes of 20 seconds length annotated at 2Hz, resulting in a total of 40k annotated key frames. The dataset also provides a benchmark to evaluate and compare object detection and object tracking algorithms, with well defined evaluation metrics for both applications.

1.6 Dissertation Organization

The rest of this dissertation is organized as follows. Chapter 2 provides a thorough literature survey on both traditional and modern sensor fusion algorithms. Although this dissertation only focuses on utilizing modern DNNs for sensor fusion, a background on traditional fusion algorithms provides more insight into the topics discussed in this dissertation. Chapter 3 introduces a radar-based region proposal algorithm, demonstrating the effectiveness of radar detections in generating 2D object proposals for a two-stage object detection algorithm. In chapter 4, radar-based proposals are used to perform joint 2D object detection and distance estimation, taking advantage of the accurate depth information provided by the radar detections. A radar-camera fusion network for 3D object detection is proposed in chapter 5. In addition to the detection bounding boxes, this network also estimates objects' velocity without requiring any temporal information. Chapter 6 introduces a 3D object tracking algorithm, utilizing radar and camera data to track 3D object bounding boxes through using an end-to-end trained DNN. Finally, in Chapter 7 the contributions of this dissertation are summarized and several directions for future research in this area are discussed.

Chapter 2

Literature Survey

Autonomous vehicles are usually equipped with a variety of sensors with different modalities to understand the environment. The data obtained from these sensors is used to perform different tasks crucial to autonomous navigation, such as object detection and object tracking. An enormous amount of research has been conducted over the past decade to adopt and improve sensor fusion methods for this application [23]. This chapter first provides a brief review of the existing object detection and object tracking algorithms, then focuses on the classical and modern methods of sensor fusion. The classical fusion algorithms were most popular before the deep learning era, but some of them are still very useful in robotics and autonomous driving applications. The modern methods of sensor fusion are mostly based on DNNs and have been very successful in establishing the state of the art results in recent years.

2.1 Object Detection

Most vision-based object detection networks follow one of the two approaches: two-stage or single-stage detection pipelines [25]. In two-stage detection networks, a set of class-agnostic object proposals are generated in the first stage, and are refined, classified and scored in the second stage. R-CNN [29] is the pioneering work in this category, using proposal generation algorithms such as Selective Search [83] in the first stage and a CNN-based detector in the second stage. Fast R-CNN [28] also uses an external proposal generator,

but eliminates redundant feature extraction by utilizing the global features extracted from the entire image to classify each proposal in the second stage. Faster R-CNN [73] unifies the proposal generation and classification by introducing the Region Proposal Network (RPN), which uses the global features extracted from the image to generate object proposals.

One-stage object detection networks on the other hand directly map the extracted features to bounding boxes by treating the object detection task as a regression problem. YOLO [71] and SSD [51] detection networks are in this category, regressing bounding boxes directly from the extracted feature maps. RetinaNet [48] addressed the foreground-background class imbalance problem in single-stage object detection and achieved better results than the state-of-the-art two-stage detection networks.

Monocular 3D object detection methods use a single camera to estimate 3D bounding boxes for objects. 3D RCNN [43] uses Fast R-CNN [28] with an additional head and 3D projection. It also uses a collection of CAD models to learn class-specific shape priors for objects. Deep3DBox [59] regresses a set of 3D object properties using a convolutional neural network first, then uses the geometric constraints of 2D bounding boxes to produce a 3D bounding box for the object. CenterNet [109] takes a different approach and uses a keypoint detection network to find objects' center point on the image. Other object properties such as 3D dimension and location are obtained by regression using only the image features at the object's center point.

LiDARs have been widely used for 3D object detection and tracking in autonomous driving applications in recent years. The majority of LiDAR-based methods either use 3D voxels [45, 110] or 2D projections [46, 12, 99, 101] for point cloud representation. Voxel-based methods are usually slow as a result of the voxel grid's high dimensionality, and projection-based methods might suffer from large variances in object shapes and sizes depending on the projection plane. PointRCNN [78] directly operates on raw point clouds and generates 3D object proposals in a bottom-up manner using point cloud segmentation. These proposals are refined in the second stage to generate the final detection boxes.

2.2 Object Tracking

Object tracking methods have many applications in different computer vision tasks such as autonomous driving, surveillance and activity recognition. Most existing methods on MOT use the tracking-by-detection approach [111, 98, 21], relying on the performance of an underlying object detection algorithm [72, 73, 102] and focusing on improving the association between detections. One major drawback in this approach is that the association task does not utilize the valuable features extracted in the detection step. More recently, the *joint detection and tracking* approach is trending for MOT where an existing object detection network is converted into an object tracker to accomplish both tasks in the same framework [24, 5, 36].

From another perspective, MOT algorithms can be split into batch and online methods. Batch methods use the entire sequence of frames to find the global optimal association between the detections. Most methods in this category are based on optical flow algorithms and create a flow graph from the entire sequence [76, 104]. Online methods, on the other hand, only use the information up to the current frame for tracking objects. Many of these algorithms generate a bipartite graph matching problem which is solved using the Hungarian algorithm [6]. More modern methods in this category use deep neural networks to solve the association problem [4, 93].

MOT methods can also be divided into 2D and 3D categories. Most 3D MOT methods are developed as an extension of existing 2D tracking models, with the distinction that input detections are in the 3D space rather than the 2D image plane. Some of the 3D MOT methods use LIDAR point clouds [92] or a combination of point clouds and images [106] as their inputs.

2.2.1 2D Multi-Object Tracking

DeepSORT [94] uses an overlap-based association method with a bipartite matching algorithm, in addition to appearance features extracted by a deep network. In [24] authors use the current and previous frames as inputs to a siamese network that predicts the offset between the bounding boxes in different frames. Tracktor [5] exploits the bounding

box regression of the object detector network to directly propagate the region proposals' identities, which eliminates the need for a separate association step. Since it is assumed that the bounding boxes have a large overlap between consecutive frames, low frame-rate sequences would require a motion model in this approach. Zhu et al. [112] propose flow-guided feature aggregation, where an optical flow network estimates the motion between the current and previous frames. The feature maps from previous frames are then warped to the current frame using the flow motion and an adaptive weighting network is used to aggregate and feed them into the detection network. Integrated detection [107] proposes an early integration of the detection and tracking tasks, where the outputs of the object detector are conditioned on the tracklets computed over the prior frames. A bipartite-matching association method is then used to associate the bounding boxes.

2.2.2 3D Multi-Object Tracking

Hu et al. [32] combine 2D image-based feature association and 3D Long Short Term Memory (LSTM)-based motion estimation for 3D object tracking. Their method leverages 3D box depth-ordering matching and 3D trajectory prediction to improve instance association and re-identification of occluded objects. Weng et al. [92] propose a real-time MOT system called AB3DMOT that uses LIDAR point clouds for object detection and a combination of Kalman filter and the Hungarian algorithm for state estimation and data association. CenterTrack [108] takes a pair of images and detections from prior frames as input to a point-based framework, where each object is represented by the center point of its bounding box. The network estimates an offset vector from the center point of objects in the current frame to their corresponding center points in the previous frame, and uses a greedy algorithm for object association. Besides images and point clouds, some methods use map information to improve the tracking performance for autonomous driving applications. Argoverse [11] uses detailed map information such as lane direction, ground height and drivable area to improve the accuracy of 3D object tracking.

2.3 Sensor Fusion

2.3.1 Classical Methods

The classical fusion methods can be divided into three categories [9]: data association, state estimation and decision fusion.

Methods in the data association category are concerned with establishing the set of observations or measurements that correspond to a set of targets [9]. Nearest Neighbor (NN) is among the simplest data association methods, grouping the most similar values based on a predefined distance metric. NN does not perform very well in noisy or cluttered environments, as it can introduce error propagation by providing many pairs with similar probabilities [7]. K-Means [53] is an modified and iterative version of the NN algorithm that finds the best localization of the cluster centers. This algorithm needs to know the number of clusters a priori and is not always able to find the optimal cluster centers. A common practice with K-Means is to start with a small number of clusters and increase it until adequate result is obtained. The Probabilistic Data Association (PDA) [3] is another method in this category that assigns a probability of association to each hypothesis from every target observation, and computes the state of the target as a wighted sum of the estimated states under all hypotheses. The target measurements are validated first by checking if they are within a certain range of the predicted measurement. When used for tracking objects, PDA performs well when objects do not change their movement pattern abruptly, but will most likely lose the target otherwise [9].

State estimation methods aim at finding the value of a state vector (position, size, velocity, etc.) by finding the best fit to the observed data, where these observations might be noisy or corrupted. The Maximum Likelihood (ML) method is one of the most famous approaches in this category and is based on probabilistic theory. The likelihood function $\lambda(x)$ is defined as a Probability Density Function (PDF) of the sequence of observations given the true value of the state, and the ML estimator finds the state according to the following equation [9]:

$$\lambda(x) = p(z|x) \tag{2.1}$$

$$\hat{x}(k) = arg\max_{x} \lambda(x) \tag{2.2}$$

where x is the state being estimated, and z is the sequence of k previous observations of x. In order to calculate the likelihood function, the ML method requires the knowledge of the empirical or analytical model of the sensor to provide the prior distribution. The Maximum A Posteriori (MAP) estimation method is another method in this category and is based on the Bayesian theory. The MAP estimator finds the value of state x that maximizes the posterior probability distribution:

$$\hat{x}(k) = arg\max_{x} p(x|z) \tag{2.3}$$

The difference between ML and MAP is in the assumptions made about the state x. While MAP considers x to be the output of a random variable with a known a priori PDF, ML assumes x is a fixed but unknown point in the parameter space [9].

The Kalman filter [35] is the most popular method in the state estimation category. It assumes the state x at time k is evolved from the state at time $k-1$, and uses the following space-time model to estimated it:

$$x(k) = F(k)x(k-1) + B(k)u(k) + w(k) \tag{2.4}$$

$$z(k) = H(k)x(k) + v(k) \tag{2.5}$$

where $z(k)$ is the observation at time k, $H(k)$ is the measurement matrix, v and w are the observation and process noise modeled as random Gaussian variables, $F(k)$ is the state transition matrix, $B(k)$ is the input transition matrix and $u(k)$ is the input vector. Being an recursive estimator, the Kalman filter only needs the current measurements and the estimated states from the previous time step to estimate the state for the current time step. This process is usually broken down into the "predict" phase where the state in the current time stamp is predicted using the state estimate from the previous time stamp, and the "update" phase where the current prediction and observations are combined to generate a refined state estimate, called the *a posteriori* state estimate. In the predict step the state is calculates as below:

$$\hat{x}(k|k-1) = F(k)\hat{x}(k-1|k-1) + B(k)u(k) \tag{2.6}$$

$$P(k|k-1) = F(k)P(k-1|k-1)F(k)^T + Q(k) \tag{2.7}$$

where $P(k|k-1)$ is the predicted estimate covariance matrix, and $Q(k)$ is the covariance matrix of the process noise w. In the update phase the state is refined using the formula below:

$$\hat{x}(k|k) = (I - K(k))(H(k)\hat{x}(k|k-1)) + (K(k))(H(k)x(k) + v(k)) \tag{2.8}$$

$$K(k) = P(k|k-1)H(k)^T S(k)^{-1} \tag{2.9}$$

$$S(k) = H(k)P(k|k-1)H(k)^T + R(k) \tag{2.10}$$

where $K(k)$ is called the Kalman gain, $S(k)$ is called the innovation covariance and $R(k)$ is the covariance matrix of the observation noise $v(k)$. The Kalman filter obtains optimal estimations if the system can be explained with a linear model and the noise is Gaussian [55]. For non-linear models, the Extended Kalman Filter (EKF) can be used [90], which is computationally more expensive and requires the calculations of Jacobians. The Unscented Kalman Filter (UKF) [34] is another variation of the Kalman filter designed for nonlinear systems. It uses a deterministic sampling approach to find the minimum set of points around the mean, capturing the true mean and covariance. These points are then propagated through nonlinear functions to obtain the covariance of the estimation [9].

The decision fusion methods use the detected targets to make a high level inference about the events produced by those targets. Bayesian methods are among the most famous in this category, combining the evidence according to probability theory rules [9]. A probability distribution represents uncertainty and the inference is obtained based on the Bayes rule:

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)} \tag{2.11}$$

where $P(Y|X)$ represents the belief in hypothesis Y given the information X. The Bayesian inference requires the knowledge of $P(X)$ and $P(X|Y)$, which is the main disadvantage of this method. The Dempster-Shafer inference [17, 77] generalizes the Bayesian inference, providing a way to represent incomplete knowledge and updating beliefs, allowing an explicit representation of the uncertainty [67]. Unlike Bayesian inference, Dempster-Shafer does not require the knowledge of a priori probabilities.

2.3.2 Modern Methods

Modern sensor fusion methods developed for autonomous driving applications usually use a combination of radars, cameras and LIDARs as their sensing modalities as these are the most commonly used sensors in autonomous driving vehicles. Authors in [66] proposed a LIDAR and vision-based pedestrian detection system using both a centralized and decentralized fusion architecture. In the former, authors proposed a feature level fusion system where features from LIDAR and vision spaces are combined in a single vector which is classified using a single classifier. In the latter, two classifiers are employed, one per sensor-feature space. MV3D [12] extracts features from the front view and Bird's Eye View (BEV) representations of the LIDAR data, in addition to the RGB image. The features obtained from the LIDAR's BEV are then used to generate 3D object proposals, and a deep fusion network is used to combine features from each view and predict the object class and box orientations. PointFusion [97] processes the image and LIDAR data using a CNN and a PointNet model respectively, and then generate 3D object proposals using the extracted features. Frustum PointNet [68] directly operates on the raw point clouds obtained from an RGB-D camera and uses the RGB image and a 2D object detector to localize objects in the point cloud.

In [1] authors use the up-sampled representation of the sparse LIDAR's range data, the high-resolution map from LIDAR's reflectance data and the RGB image from a monocular color camera as three input modalities to their network. The network uses a late-fusion strategy to performs bounding box detections in each one of these modalities. [64] also uses a late fusion approach, fusing the classification outputs from independent pretrained CNNs. The inputs to the classifiers are 3D point clouds and image data. In [75] authors up-sample the LIDAR point cloud to generate a dense depth map and then extract three features representing different aspects of the 3D scene. The extracted features are used as extra image channels. Authors in [89] use the BEV representation of the LIDAR point clouds, and construct non-homogeneous pooling layer to transform features between the BEV map and the front view map. The mapping between these two maps is constructed using the sparse LIDAR point cloud. The constructed pooling layer allows efficient fusion of the features

from these two views at any stage of the network, and the methods shows to be particularly good at detecting pedestrians in BEV.

Few studies have focused on fusing radars with other sensors for autonomous driving applications. In [37] authors use the radar distance measurements and a motion stereo technique to detect object boundaries in a sequence of images. [33] projects radar detections to the image and generate object proposals for a small CNN classification network. In [10], Chadwick *et al.* project radar detections to the image plane and use them to boost the object detection accuracy for distant objects. CRF-Net [63] also projects radar detections to the image plane, but represents them as vertical lines on the image, where the pixel values correspond to the depth of each detection point. The image data is then augmented with the radar information and used in a convolutional network to perform 2D object detection. Authors in [57] use a BEV projection of radar detections with six height maps and a density map in addition to the RGB image as inputs to a network similar to [42], generating 3D detections. Authors in [18] propose a radar and LIDAR fusion method based on Kalman filter for obstacle detection. Their method uses the operating range of the sensors and real-time sensor data to compute the observation and measurement noise data for the Kalman filter. RadarNet [100] fuses radar and LIDAR data for 3D object detection using a deep learning approach. It uses an early fusion mechanism to learn joint representations from the two sensors, and a late-fusion mechanism to exploit radar's radial velocity evidence and improve the estimated object velocity.

Chapter 3

Radar Region Proposal Network

Radars are one of the most popular sensors in autonomous vehicles and have been studied for decades in different automotive applications. Authors in [30] were among the first researchers discussing such applications for radars, providing a detailed approach for utilizing them on vehicles. While radars can provide accurate range and range-rate information for the detected objects, they are not suitable for tasks such as object classification. Cameras on the other hand, are very effective sensors for object classification, making radar and camera sensor fusion an interesting topic in autonomous driving applications.

This chapter focuses on the utilization of radar data for generating object proposals in a two-stage object detection network. Because of the extra processing required for generating object proposals, Two-stage object detection methods are usually slower than the one-stage methods. A radar-based Region Proposal Network (RPN) called Radar Region Proposal Network (RRPN) is proposed in this chapter, which skips the computationally expensive vision-based region proposal step, and uses only radar point clouds to generate 2D object proposals. These proposals are used in the second stage of the detection network to localize and classify object. The resulting detection network can be categorized as a middle-fusion algorithm, as it processes the radar and image data individually before merging them in the second stage of the object detection network. RRPN also inherently provides an attention mechanism that focuses the available computational resources on the more important parts of the image. While in other object detection applications the entire image may be of equal importance, in an autonomous vehicles more attention needs to be given to the objects

on the road. Because of the dependency on radar detections to generate object proposals, the proposed object detection network focuses only on the physical objects surrounding the vehicle. The proposed detection network is evaluated on the NuScenes dataset [8], which provides synchronized and annotated data from radars and cameras integrated on a vehicle. In the evaluations, RRPN, is used as the RPN in the Fast R-CNN object detection network, replacing the original RPN, Selective Search. Evaluation results show that RRPN achieves higher mean Average Precision (AP) and mean Average Recall (AR) compared to the Selective Search algorithm, while generating proposals up to 100 times faster for each image.

RRPN consists of three steps: perspective transformation, anchor generation and distance compensation, each discussed individually in the following sections.

3.1 Perspective Transformation

The first step in the proposed method is mapping the radar detections to the image plane. Radar detections are usually reported in a BEV perspective as shown in Fig. 3.1 (a), with the object's range d and azimuth α reported in the radar's coordinate system. These detections need to be mapped to the vehicle coordinate system first using the radar calibration parameters provided in the dataset. The detections are then mapped to the camera coordinate system and the image plane using the extrinsic and intrinsic calibration parameters respectively. This enables the association of the objects detected by the Radar to their corresponding object in the image. The projective relation between a 3D radar detection point $P = [X; Y; Z; 1]$ and its image $p = [x; y; 1]$ in the camera-view plane can be expressed as below:

$$p = K[R|t] * P \tag{3.1}$$

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.2}$$

Where K is the camera intrinsics matrix and $[R|t]$ is the extrinsic matrix using rotation and translation parameters to transform the point from the radar frame to the camera frame. The above transformation is applied to all radar detections. All detections mapped outside the image after applying the transformation are ignored.

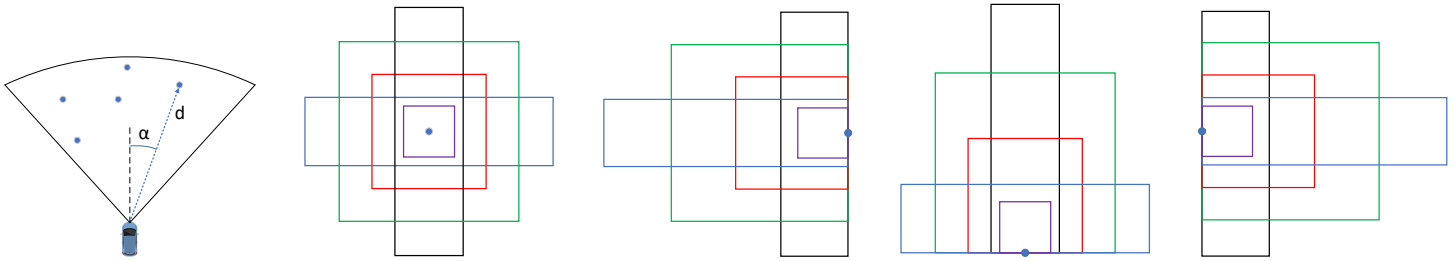
3.1.1 Anchor Generation

Mapping radar detections to the image provides an approximate location for the corresponding objects on the image. This information is obtained without any direct processing on the image itself. Having the mapped radar detections, hereafter referred to as Point of Interest (PoI), a simple approach for generating object proposals would be introducing a bounding box centered at every PoI. One problem with this approach is that radar detections are not always mapped to the center of the detected objects in every image. Another problem is the fact that Radars do not provide any information about the size of the detected objects and proposing a fixed-size bounding box for objects of different sizes would not be an effective approach. To address these problems, the concept of anchor bounding boxes from Faster R-CNN [73] is used to to generate the proposals. Specifically, several bounding boxes with different sizes and aspect ratios are generated and centered at each PoI. For every PoI, four different anchor sizes and three different aspect ratios are used to generate these proposals, as illustrated in Fig. 3.1 (b).

To account for the fact that the PoI is not always mapped to the center of the object in the image, shifted versions of these anchors are also generated. These translated anchors provide more accurate bounding boxes when the PoI is mapped towards the right, left or the bottom of the object, as shown in Fig. 3.1 c-e.

3.1.2 Distance Compensation

The distance of each object from the vehicle plays an important role in determining its size in the image. Generally, objects' sizes in an image have an inverse relationship with their distance from the camera. Radar detections have the range information for every detected



(a) Bird's eye view **(b)** Centered anchors **(c)** Right aligned **(d)** Bottom aligned **(e)** Left aligned

Figure 3.1: Generating anchors of different shapes and sizes for each radar detection, shown here as the blue circle.

object, which is used in this step to scale all generated anchors. The following formula is used to determine the scaling factor for every anchor:

$$S_i = \alpha \frac{1}{d_i} + \beta \tag{3.3}$$

where d_i is the distance to the ith object, and α and β are two parameters used to adjust the scale factor. These parameters are learned by maximizing the Intersection over Union (IoU) (IoU) between the generated bounding boxes and the ground truth bounding boxes in each image, as shown in Eq. 3.4 below.

$$\underset{\alpha,\beta}{\text{argmax}} \sum_{i=1}^{N} \sum_{j=1}^{M_i} \max_{1<k<A_i} IoU_{jk}^{i}(\alpha, \beta) \tag{3.4}$$

In this equation, N is the number of training images, M_i is the number of ground truth bounding boxes in image i, A_i is the number of anchors generated in image i, and IoU_{jk}^{i} is the IoU between the jth ground truth bounding box in image i and the kth proposed anchor in that image. This equation finds the parameters α and β that maximize the IoU between the ground truth and proposed bounding boxes. A simple grid search approach is used to find α and β.

3.2 Experiments and Results

3.2.1 Dataset and Implementation Details

To use the NuScenes dataset for evaluating RRPN, all 3D bounding boxes are first converted to their equivalent 2D boxes. The classes used for training and evaluation are Car, Truck, Person, Motorcycle, Bicycle and Bus. Two subsets of the samples available in the dataset are used in the evaluation. The first subset only contains data from the front camera and front radar, with 23k samples. This subset is referred to as NS-F . The second subset contains data from the rear camera and two rear Radars, in addition to all the samples from NS-F . This subset has 46k images and is called NS-FB . Since front radars usually have a longer range compared to the corner radars, NS-F gives us more accurate detections for objects far

away from the vehicle. Each dataset is further split with a 0.85/0.15 ratio for training and testing, respectively.

The RoIs generated by RRPN are used in the Fast R-CNN object detection network. Two different backbone networks have been used with Fast R-CNN: the original ResNet-101 network [31] hereafter called R101 and the ResNeXt-101 [95], an improved version of ResNet, hereafter called X101. In the training stage, a model pretrained on the COCO dataset is used as the initial model and fine-tuned on NS-F and NS-FB . The detection results using RRPN proposals are compared with those of the Selective Search algorithm [83], which uses a variety of complementary image partitionings to find objects in images. In both RRPN and Selective Search, the number of object proposals are limited to 2000 per image.

The evaluation metrics used in these experiments are the same metrics used in the COCO dataset [49], namely mean AP and mean AR. The AP calculated with 0.5 and 0.75 IOU is also reported, in addition to AR for small, medium and large objects areas.

3.2.2 Results

The Fast R-CNN object detection results for the two RPN networks on NS-F and NS-FB datasets are shown in Table 3.1. According to these results, RRPN is outperforming Selective Search in almost all metrics. Table 3.2 shows the per-class AP results for the NS-F and NS-FB datasets, respectively. For the NS-F dataset, RRPN outperforms Selective Search in the Person, Motorcycle and Bicycle classes with a wide margin, while following Selective Search closely in other classes. For the NS-FB dataset, RRPN outperforms Selective Search in all classes except for the Bus class.

Figures 3.2, 3.3, 3.4 and 3.5 show selected examples of the object detection results, with the first row showing the ground truth and mapped radar detections. The next two rows are the detected bounding boxes using the region proposals from Selective Search and RRPN respectively. According to these figures, RRPN has been very successful in proposing accurate bounding boxes even under hard circumstances such as object occlusion and overlap. In these experiments, RRPN was able to generate proposals for about 70 to 90 images per second, depending on the number of radar detections, while Selective Search took between 2-7 seconds per image.

Table 3.1: Detection results for the NS-F and NS-FB datasets

method	AP	AP50	AP75	AR	ARs	ARm	ARl
SS + X101 - F	0.368	0.543	0.406	0.407	0.000	0.277	0.574
SS + R101 - F	0.418	0.628	0.450	0.464	0.001	0.372	0.316
RRPN + X101 - F	0.419	**0.652**	0.463	0.478	**0.041**	0.406	0.573
RRPN + R101 - F	**0.430**	0.649	**0.485**	**0.486**	0.040	**0.412**	**0.582**
SS + X101 - FB	0.332	0.545	0.352	0.382	0.001	0.291	0.585
SS + R101 - FB	0.336	0.548	0.357	0.385	0.001	0.291	**0.591**
RRPN + X101 - FB	0.354	**0.592**	0.369	0.420	0.202	**0.391**	0.510
RRPN + R101 - FB	**0.355**	0.590	**0.370**	**0.421**	**0.211**	**0.391**	0.514

Table 3.2: Per-class AP for the NS-F and NS-FB datasets

method	Car	Truck	Person	Motorcycle	Bicycle	Bus
SS + X101 - F	0.424	0.509	0.117	0.288	0.190	0.680
SS + R101 - F	**0.472**	**0.545**	0.155	0.354	0.241	**0.722**
RRPN + X101 - F	0.428	0.501	0.212	0.407	0.304	0.660
RRPN + R101 - F	0.442	0.516	**0.220**	**0.434**	**0.306**	0.664
SS + X101 - FB	0.390	0.415	0.122	0.292	0.179	0.592
SS + R101 - FB	0.392	0.420	0.121	0.291	0.191	**0.600**
RRPN + X101 - FB	0.414	**0.449**	**0.174**	0.294	**0.215**	0.579
RRPN + R101 - FB	**0.418**	0.447	0.171	**0.305**	0.214	0.572

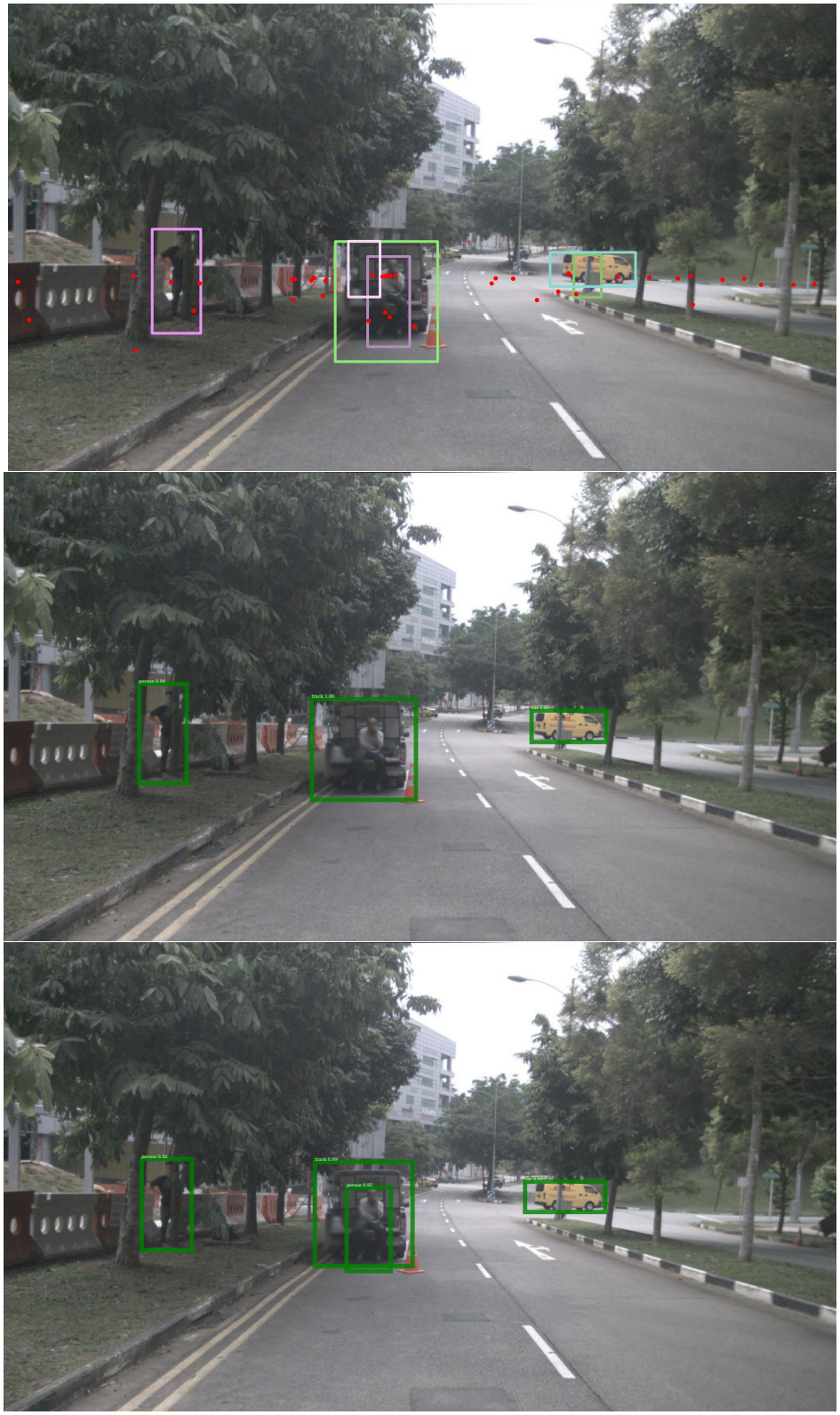


Figure 3.2: Detection results. Top row: ground truth, middle row: Selective Search, bottom row: RRPN

Figure 3.3: Detection results, cont. Top row: ground truth, middle row: Selective Search, bottom row: RRPN

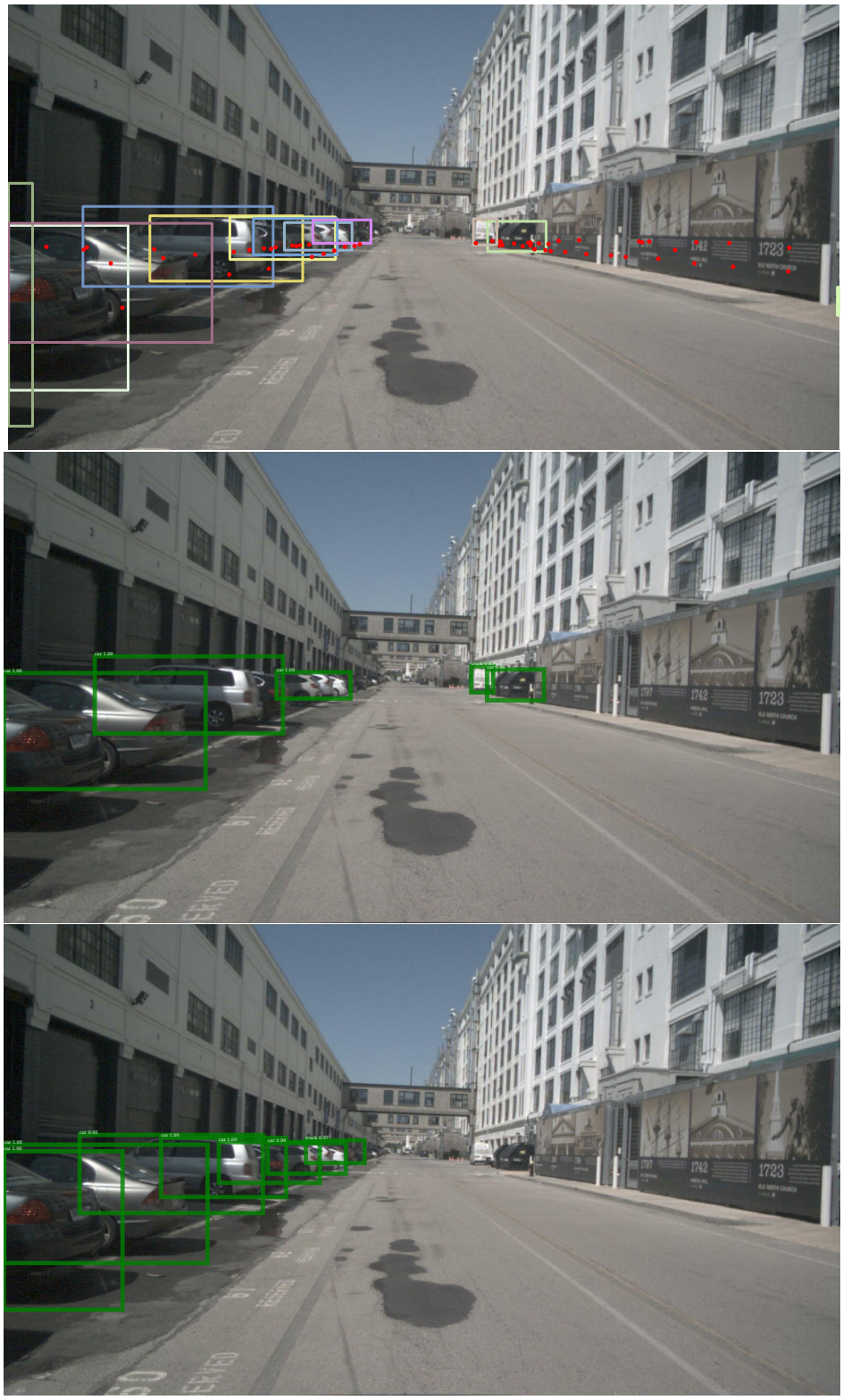


Figure 3.4: Detection results, cont. Top row: ground truth, middle row: Selective Search, bottom row: RRPN

Figure 3.5: Detection results, cont. Top row: ground truth, middle row: Selective Search, bottom row: RRPN

3.3 Conclusion

A real-time region proposal network for object detection in autonomous driving applications was proposed in this chapter. By only relying on radar detections to propose RoIs, this method is extremely fast while at the same time achieving a higher precision and recall compared to the Selective Search algorithm. A two-stage object detection network based on RRPN acts as a middle-fusion algorithm by individually processing radar and camera data first and then combining them to obtain faster and more accurate detections. The experiments conducted on RRPN show that it operates more than 100x faster than the Selective Search algorithm, while resulting in better detection average precision and recall.

In the next chapter, a different approach to using radar detections for generating object proposals is presented, where the depth information provided by the radar detections are also utilized to estimate objects' distance from the ego vehicle.

Chapter 4

Joint Object Detection and Range Estimation

In this chapter a radar-camera fusion algorithm for joint object detection and distance estimation is presented. The proposed method is designed as a two-stage object detection network that fuses radar point clouds and image features to generate accurate object proposals. For every object proposal, a depth value is also calculated to estimate the object's distance from the vehicle. These proposals are then fed into the second stage of the detection network for object classification.

A radar-camera sensor fusion system can provide valuable depth information for all detected objects in an autonomous driving scenario, while at the same time eliminates the need for computationally expensive 3D object detection using LIDAR point clouds. The proposed sensor fusion network is shown in Fig. 4.1. This network takes radar point clouds and RGB images as input and fuses them to generates object proposals for an object classifier in a two-stage object detection approach. A middle-fusion approach for fusing the radar and image data is used here, where outputs of each sensor are processed independently first, and are merged at a later stage for more processing. More specifically, the radar detections are first used to generate 3D object proposals, which are further improved by using the image features extracted by a backbone network. These proposals are then merged with image-based proposals and are fed to the second stage for classification. All generated proposals

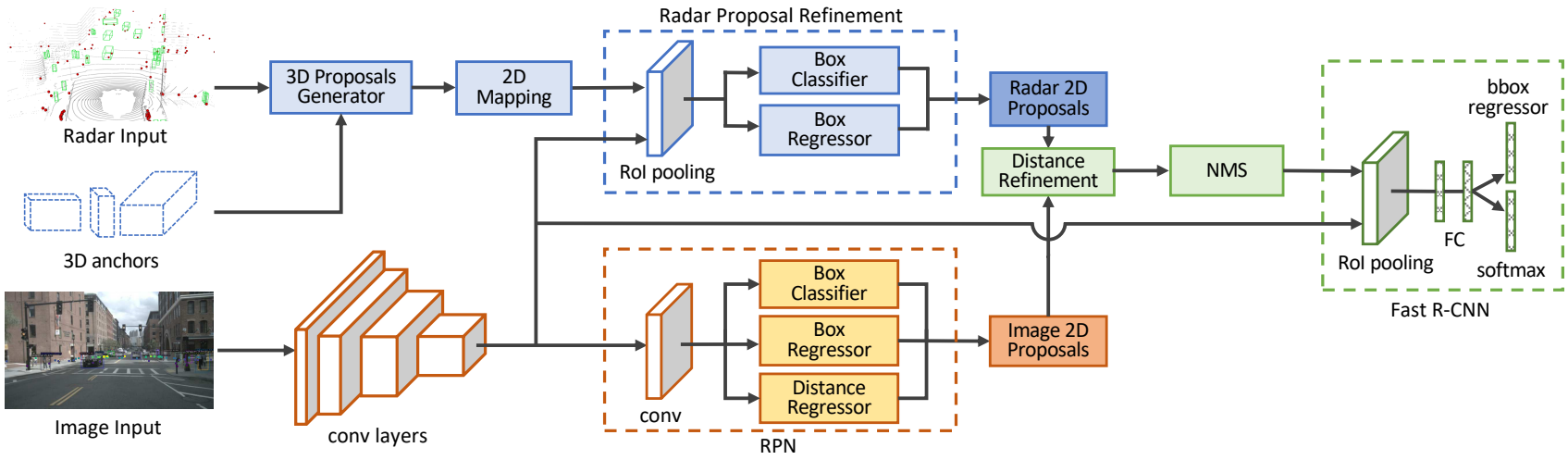


Figure 4.1: The proposed network architecture. Inputs to the network are radar point cloud, camera image and 3D anchor boxes. radar-based object proposals are generated from the point cloud and fused with image features to improve box localization.

are associated with an estimated depth, calculated either directly from the radar detections or via a distance regression layer using the extracted image features.

4.1 Proposal Generation

4.1.1 Radar Proposal Network

The proposed method directly uses the radar detection to generates 3D object proposals without any feature extraction. The proposals are generated using predefined 3D anchor boxes tailored to each object class in the dataset. Each 3D anchor is parameterized as (x, y, z, w, l, h, r), where (x, y, z) is the center, (w, l, h) is the size, and (r) is the orientation of the box in vehicle's coordinate system. The anchor size, (w, l, h), is fixed for each object category, and is set to the average size of the objects in that category in the training dataset. For every anchor box, two different orientations at 0° and 90° are used, referenced from the vehicle's centerline. The center location for each anchor is obtained from the 3D position of the radar detection in vehicle's coordinate system. This results in $2n$ boxes for every radar detection where n is the number of object classes in the dataset, each having two different orientations.

In the next step, all 3D anchors are mapped to the image plane and converted to equivalent 2D bounding boxes by finding the smallest 2D enclosing box for each mapped 3D anchor. Since every 3D proposal is generated from a radar detection, it has an accurate distance associated with it. This distance is used as the proposed distance for the corresponding 2D bounding box. Because the size of each 3D anchor was chosen based on the size of the object in the corresponding class, the resulting proposals capture the true size of the objects as they appear in the image. This eliminates the need for adjusting the size of radar proposals based on their distance from the vehicle, which was done in the method discussed in the previous chapter.

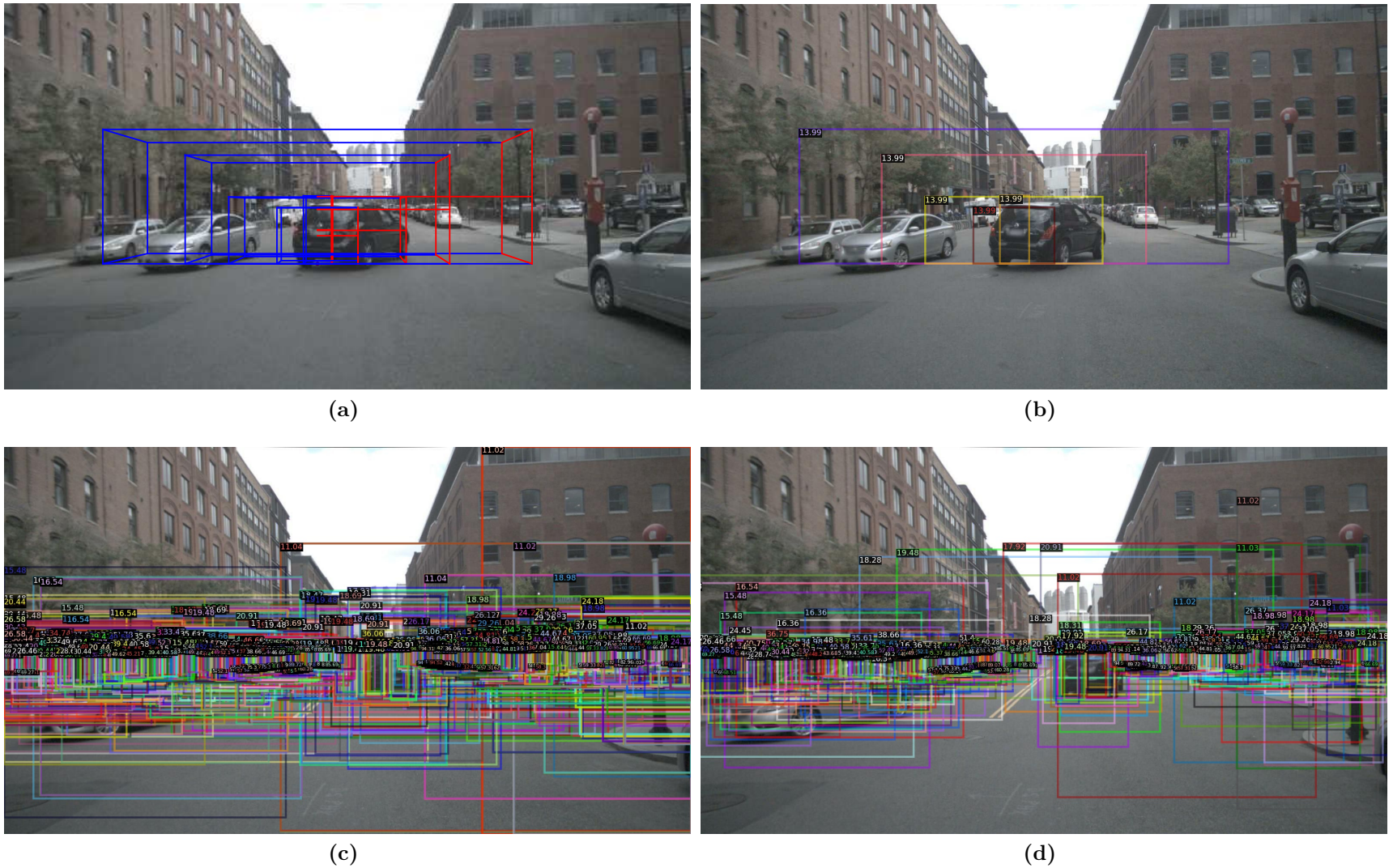
Fig. 4.2(b) illustrates 3D anchors and equivalent 2D proposals generated for a sample image. As shown in this figure, radar-based proposals are always focused on objects that

are on the road plane. This prevents unnecessary processing of areas of the image where no physical object exists, such as the sky or buildings in this image.

In the next step, all generated 2D proposals are fed into the Radar Proposal Refinement (RPR) subnetwork. This is where the information obtained from the radars (radar proposals) is fused with the information obtained from the camera (image features). RPR uses the features extracted from the image by the backbone network to adjust the size and location of the radar proposals on the image. As radar detections are not always centered on the corresponding objects in the image, the generated 3D anchors and corresponding 2D proposals might be offset as well. The box regressor layer in the RPR uses the image features inside each radar proposal to regress offset values for each proposal's corner points. The RPR also contains a box classification layer, which estimates an objectness score for every radar proposal. The objectness score is used to eliminate proposals generated by radar detections coming from background objects, such as buildings and light poles. The inputs to the box regressor and classifier layers are image features inside negative and positive radar proposals. following [73], positive proposals are defined as ones with an IoU overlap higher than 0.7 with any ground truth bounding box, and negative proposals as ones with an IoU below 0.3 for all ground truth boxes. Radar proposals with an IoU between 0.3 and 0.7 are not used for training. Since radar proposals have different sizes depending on their distance, object category and orientation, a RoI Pooling layer is used before the box regression and classification layers to obtain feature vectors of the same size for all proposals. Fig. 4.2(d) shows the radar proposals after the refinement step.

4.1.2 Image Proposal Network

The proposed architecture also uses a RPN network to generate object proposals from the image. The radar proposal network is not always successful in generating proposals for certain object categories that are harder for radars to detect but are easily detected in the image, such as pedestrian or bicycles. On the other hand, the image-based proposal network might fail to detect far away objects that are easily detected by the radar. Having an image-based object proposal network in addition to the radar-based network improves the object



(a) **(b)** **(c)** **(d)**

Figure 4.2: Radar-based proposals. (a): 3D anchors for one radar detection ($r = 90°$). (b): 2D proposals obtained from 3D anchors. (c): 2D proposals for all radar detections inside the image. (d): Refined radar proposals after applying box regression. Radar-based distances in meters are shown on the bounding boxes.

detection accuracy, as they complement each other by using two different modalities for proposal generation and distance estimation.

Image-based object proposals are generated by a network similar to the RPN introduced in Faster R-CNN [73]. The input to this network is the image feature maps extracted by the backbone CNN. To estimate distance for every object proposal, a fully connected distance regression layer is added on top of the convolutional layer in the RPN, as shown in Fig. 4.1. This layer is implemented with a 1×1 convolutional layer similar to the box-regression and box-classification layers in the RPR network. The distance regression layer generates k outputs, where k is the number of 2D anchor boxes used in the RPN network at each location on the feature map. A cross entropy loss is used for object classification and a Smooth L1 loss is used for the box distance regressor layers.

4.1.3 Distance Refinement

The outputs of the radar and image proposal networks need to be merged for the second stage of the object detection network. Before using the proposals in the next stage, redundant proposals are removed by applying Non-Maximum Suppression (NMS). The NMS layer would normally remove overlapping proposals without discriminating based on the bounding box's origin, but radar-based proposals have a more reliable distance estimation than the image-based proposals. This is because image-based distances are estimated only from 2D image feature maps with no depth information. To make sure the radar-based distances are not unnecessarily discarded in the NMS process, the IoU between radar and image proposals are first calculated, then an IoU threshold is used to find the matching proposals and overwrite the image-based distances by their radar-based counterparts. The calculated IoU values are reused in the next step where NMS is applied to all proposals, regardless of their origin. The remaining proposals are then fed into the second stage of the detection network to calculate the object class and score.

4.2 Detection Network

The inputs to the second stage detection network are the feature map from the image and object proposals. The structure of this network is similar to Fast R-CNN [28]. The feature map is cropped for every object proposals and is fed into the RoI pooling layer to obtain feature vectors of the same size for all proposals. These feature vectors are further processed by a set of fully connected layers and are passed to the softmax and bounding box regression layers. The output is the category classification and bounding box regression for each proposal, in addition to the distance associated to every detected object. Similar to the RPN network, a cross entropy loss is used for object classification and a Smooth L1 loss is used for the box regression layer. Similar to Faster R-CNN [73], the following multi-task loss is used as the objective function:

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*).$$

where i is the anchor index, p_i is the i'th anchor's objectness score, p_i^* is the ground truth score (1 if anchor is positive and 0 if negative), t_i is the vector of 4 parameters representing the predicted bounding box and t_i^* is the ground truth bounding box. The log loss over two classes is used for the classification loss L_{cls}, and the the smooth L_1 loss for the regression loss, L_{reg}. N_{cls} and N_{reg} are normalization factors and λ is a balancing parameter.

4.3 Experiments and Results

4.3.1 Dataset and Implementation Details

The proposed network uses FPN [48] with ResNet-50 [31] pretrained on ImageNet as the backbone for image feature extraction. The same RPN architecture as Faster R-CNN [73] is used, and only the distance regression layer has been added on top of its convolution layer for distance estimation. For the second stage of the network, the classification stage, the same architecture as Fast R-CNN is used.

The nuScenes dataset [8] is used to evaluate the network. Out of 23 different object classes in this dataset, 6 classes are used as shown in Table 4.2. Only samples from the

front- and rear-view cameras are used, together with detection from all the radars for both training and evaluation. The ground truth annotations in the nuScenes dataset are provided in the form of 3D boxes in the global coordinate system. As a preprocessing step, the annotations and radar point clouds are first transformed to the vehicle coordinate, then all 3D annotations are converted to their equivalent 2D bounding boxes. This is achieved by mapping the 3D boxes to the image and finding the smallest 2D enclosing bounding box. For every 3D annotation, the distance from vehicle to the box is calculated and used as the ground truth distance for its 2D counterpart. The official nuScenes splits are used for training and evaluation, and images are used at their original resolution (900×1600) for both steps. No data augmentation is used as the number of labeled instances for each category is relatively large. PyTorch is used to implement the network and all experiments were conducted on a computer with two Nvidia Quadro P6000 GPUs.

4.3.2 Results

The performance of the proposed method is shown in Table 4.1. This table shows the overall AP and AR for the detection task, and Mean Absolute Error for the distance estimation task. The Faster R-CNN network is used as the image-based detection baseline, and results from the proposed method is compared with RRPN and CRF-Net[63], which also use radar and camera fusion for object detection. CRF-Net only uses images from the front-view camera and also uses a weighted AP score based on the number of object appearances in the dataset. For fair comparison, the weighted AP scores are used to compare the proposed method with this network. The CRF-Net also reports some results after filtering the ground truth to consider only objects that are detected by at least one radar, and filtering radar detections that are outside 3D ground truth bounding boxes. These filtering operations are not applied and the comparison is made only with their results on the unfiltered data. Since CRF-Net does not report AR, per-class AP, or AP for different IoU levels, only the proposed method's overall AP is compared with theirs.

According to Table 4.1, the proposed method outperforms RRPN and CRF-Net for the detection task, improving the AP score by 0.15 and 0.54 points respectively. The proposed method also accurately estimates the distance for all detected objects, as visualized in Figures

Table 4.1: Performance on the nuScenes validation set.

	Weighted AP	AP	AP50	AP75	AR	MAE
Faster R-CNN	No	34.95	58.23	36.89	40.21	-
RRPN	No	35.45	59.00	37.00	**42.10**	-
Ours	No	**35.60**	**60.53**	**37.38**	**42.10**	2.65
Faster R-CNN	Yes	43.78	-	-	-	-
CRF-Net	Yes	43.95	-	-	-	-
Ours	Yes	**44.49**	-	-	-	-

Table 4.2: Per-class performance

	Car	Truck	Person	Bus	Bicycle	Motorcycle
Faster R-CNN	51.46	33.26	27.06	47.73	24.27	25.93
RRPN	41.80	**44.70**	17.10	**57.20**	21.40	**30.50**
Ours	**52.31**	34.45	**27.59**	48.30	**25.00**	25.97

Table 4.3: Per-class Mean Absolute Error (MAE) for distance estimation

Category	Car	Truck	Person	Bus	Bicycle	Motorcycle
MAE	2.66	3.26	2.99	3.187	1.97	2.81

4.3, 4.4 and 4.5. The Mean Absolute Error (MAE) is used as the evaluation metric for distance estimation. The proposed method achieves an MAE of 2.65 on all images. The per-class MAE values are provided in Table 4.3. According to this table, larger objects such as trucks and buses have a higher distance error compared to other classes. This behavior is expected and could be explained by the fact that radars usually report multiple detections for larger objects, which results in several object proposals with different distances for the same object. Additionally, most radar detections happen to be at the edge of objects, while the ground truth distances are measured from the center of objects. This results in higher distance mismatch error for larger objects, where the distance between the edge and center of the object is significant.

4.4 Conclusion

In this chapter a radar-camera fusion algorithm for joint object detection and distance estimation in autonomous driving applications was proposed. The proposed architecture uses a middle-fusion approach to employ radar point clouds and image feature maps to generate accurate object proposals. The network also uses both radar detections and image features to estimate the distance for every generated proposal. These proposals are fed into the second stage of the detection network for object classification. Experiments on the nuScenes dataset show that the proposed method outperforms other radar-camera fusion-based object detection methods, while at the same time accurately estimates the distance to every detection.

In the next chapter, a different approach to radar and camera fusion is discussed where the focus is shifted on 3D object detection. It also discusses how the velocity values reported by the radar can be utilized to estimate the velocity of the objects in each frame without requiring any temporal information.

Figure 4.3: Object detection and distance estimation results. Top: detection outputs, Bottom: ground truth.

Figure 4.4: Object detection and distance estimation results, cont. Top: detection outputs, Bottom: ground truth.

Figure 4.5: Object detection and distance estimation results, cont. Top: detection outputs, Bottom: ground truth.

Chapter 5

Radar-Camera Fusion for 3D Object Detection

In this chapter a new radar-camera sensor fusion algorithm for 3D object detection and velocity estimation is proposed. The proposed method, hereafter referred to as CenterFusion, focuses on associating radar detections to preliminary vision-based detections and generates radar feature maps to fuse with image features and estimate 3D bounding boxes for objects. Particularly, preliminary 3D detections are generated using a center-based detection network, and a novel frustum-based radar association method is used to accurately associate radar detections to their corresponding objects in the 3D space. These radar detections are then mapped to the image plane and used to create feature maps to complement the image-based features. Finally, the fused features are used to accurately estimate objects' 3D properties such as depth, rotation and velocity.

The center-based object detection network proposed in [109] is used here to detect objects' center points on the image, and regress to other object properties such as 3D location, orientation and dimensions. A middle-fusion mechanism is proposed that associates radar detections to their corresponding object's center point and exploits both radar and image features to improve the preliminary detections by re-estimating their depth, velocity, rotation and attributes.

The key in the proposed fusion mechanism is accurate association of radar detections to objects. The center point object detection network generates a heat map for every object

category in the image. The peaks in the heat map represent possible center points for objects, and the image features at those locations are used to estimate other object properties. To exploit the radar information in this setting, radar-based features need to be mapped to the center of their corresponding object on the image, which requires an accurate association between the radar detections and objects in the scene.

CenterFusion is evaluated on the nuScenes [8] dataset, where it outperforms all camera-based object detection methods in the NuScenes 3D object detection benchmark. It's also shown in the results section that exploiting radar information significantly improves velocity estimation for objects, withoud requiring any temporal information.

5.1 Preliminary

5.1.1 Radar Point Cloud

Radars are active sensors that transmit radio waves to sense the environment and measure the reflected waves to determine the location and velocity of objects. Although radar point clouds are usually represented as points in the 3D coordinate system, automotive radars usually report the detected objects as 2D points in BEV, providing the azimuth angle and radial distance to the object. As a result, the height information in a point cloud representation of radar detections is usually zero or not accurate. For every detection, the radar also reports the instantaneous velocity of the object in the radial direction. This radial velocity does not necessarily match the object's actual velocity vector in it's direction of movement. Fig. 5.1 illustrates the difference between the radial as reported by the radar, and actual velocity of the object in the vehicle's coordinate system.

Each radar detection is represented as a 3D point in the egocentric coordinate system and parameterized as $P = (x, y, z, v_x, v_y)$ where (x, y, z) is the position and (v_x, v_y) is the reported radial velocity of the object in the x and y directions. The radial velocity is compensated by the ego vehicle's motion. For every scene, 3 sweeps of the radar point cloud are aggregated (detections within the past 0.25 seconds). The nuScenes dataset provides the calibration

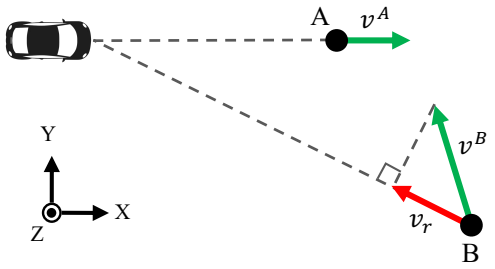


Figure 5.1: Difference between actual and radial velocity. For target A, velocity in the vehicle coordinate system and the radial velocity are the same (v^A). For target B on the other hand, radial velocity (v_r) as reported by the radar is different from the actual velocity of the object (v^B) in the vehicle coordinate system.

parameters needed for mapping the radar point clouds from the radar coordinates system to the egocentric and camera coordinate systems.

5.1.2 CenterNet

CenterNet [109] represents the state-of-the-art in 3D object detection using a single camera. It takes an image $I \in \mathbb{R}^{W \times H \times 3}$ as input and generates a keypoint heatmap $\hat{Y} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ as output where W and H are the image width and height, R is the downsampling ratio and C is the number of object categories. A prediction of $\hat{Y}_{x,y,c} = 1$ as the output indicates a detected object of class c centered at position (x, y) on the image. The ground-truth heatmap $Y \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ is generated from the ground-truth 2D bounding boxes using a Gaussian kernel. For each bounding box center point $p_i \in \mathcal{R}^2$ of class c in the image, a Gaussian heatmap is generated on $Y_{:,:,c}$. The final value of Y for class c at position $q \in \mathcal{R}^2$ is defined as [109]:

$$Y_{qc} = \max_i \exp\left(-\frac{(p_i - q)^2}{2\sigma_i^2}\right) \tag{5.1}$$

where σ_i is a size-adaptive standard deviation, controlling the size of the heatmap for every object based on its size. A fully convolutional encode-decoder network is used to predict $\hat{Y}$.

To generate 3D bounding boxes, separate network heads are used to regress object's depth, dimensions and orientation directly from the detected center points. Depth is calculated as an additional output channel $\hat{D} \in [0,1]^{\frac{W}{R} \times \frac{H}{R}}$ after applying the inverse sigmoidal transformation used in Eigen *et al.* [20] to the original depth domain. The object dimensions are directly regressed to their absolute values in meter as three output channels $\hat{\Gamma} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times 3}$. Orientation is encoded as two bins with 4 scalars in each bin, following the orientation representation in Mousavian *et al.* [59]. For each center point, a local offset is also predicted to compensate for the discretization error caused by the output strides in the backbone network [109].

Given the annotated objects $p_0, p_1, ...$ in an image, the training objective is defined as below based on the focal loss [48]:

$$L_k = \frac{1}{N} \sum_{xyc} \begin{cases} (1 - \hat{Y}_{xyc})^{\alpha} \log(\hat{Y}_{xyc}) & Y_{xyc} = 1 \\ (1 - Y_{xyc})^{\beta} (\hat{Y}_{xyc})^{\alpha} \log(1 - \hat{Y}_{xyc}) & \text{otherwise} \end{cases},$$

where N is the number of objects, $Y \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ is the annotated objects' ground-truth heatmap and α and β are focal loss hyperparameters.

5.2 Center Point Detection

The CenterFusion network architecture is shown in Fig. 5.2. It adopts the CenterNet [109] detection network for generating preliminary detections on the image. The image features are first extracted using a fully convolutional encoder-decoder backbone network. Similar to CenterNet [109], a modified version of the Deep Layer Aggregation (DLA) network [103] is used as the backbone in this architecture. The extracted image features are then used to predict object center points on the image, as well as the object 2D size (width and height), center offset, 3D dimensions, depth and rotation. These values are predicted by the primary regression heads as shown in Fig. 5.2. Each primary regression head consists of a 3×3 convolution layer with 256 channels and a 1×1 convolutional layer to generate the desired output. This provides an accurate 2D bounding box as well as a preliminary 3D bounding box for every detected object in the scene.

5.3 Radar Association

The center point detection network only uses the image features at the center of each object to regress to all other object properties. To fully exploit radar data in this process, the radar detections first need to be associated to their corresponding object on the image plane. To accomplish this, a naïve approach would be mapping each radar detection point to the image plane and associating it to an object if the point is mapped inside the 2D bounding box of that object. This is not a very robust solution, as there is not a one-to-one mapping between radar detections and objects in the image; Many objects in the scene generate multiple radar detections, and there are also radar detections that do not correspond to any object.

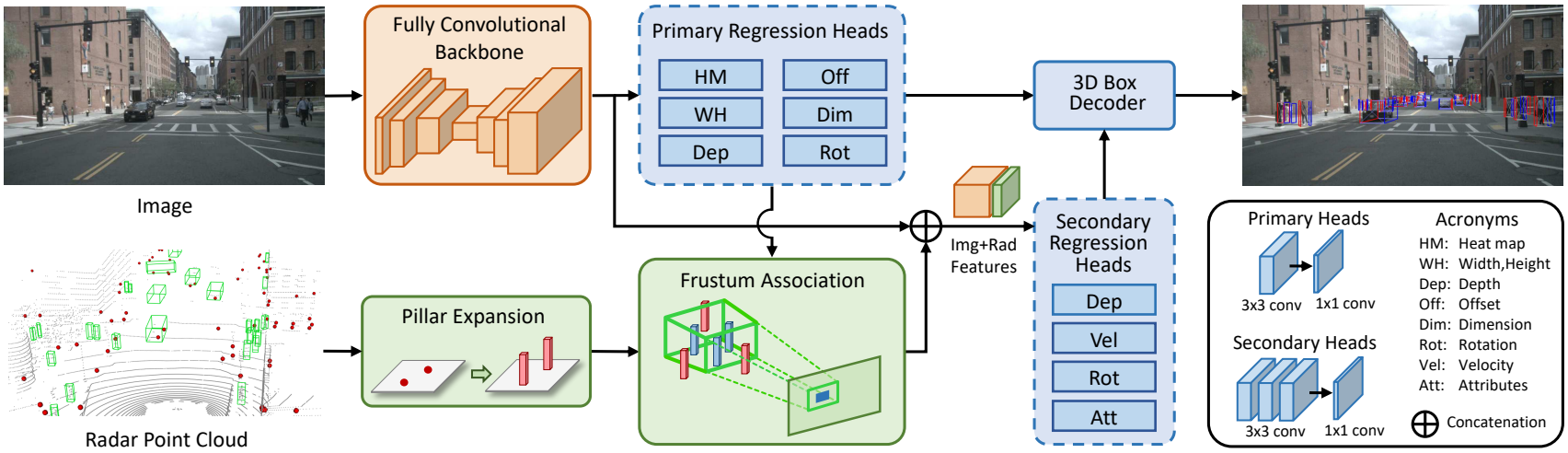


Figure 5.2: CenterFusion network architecture. Preliminary 3D boxes are first obtained using the image features extracted by the backbone. The frustum association module uses the preliminary boxes to associate radar detections to objects and generate radar feature maps. The image and radar features maps are then concatenated and used to refine the preliminary detections by recalculating depth and rotation as well as estimating objects' velocity and attributes.

Additionally, because the z dimension of the radar detection is not accurate (or does not exist at all), the mapped radar detection might end up outside the 2D bounding box of its corresponding object. Finally, radar detections obtained from occluded objects would map to the same general area in the image, which makes differentiating them in the 2D image plane difficult, if possible at all.

5.3.1 Frustum Association Mechanism:

A frustum-based association method is developed to use the object's 2D bounding boxes as well as their estimated depth and size to create a 3D Region of Interest (RoI) frustum for each object. Having an accurate 2D bounding box for an object, a frustum is created for that object as shown in Fig. 5.3. This significantly narrows down the radar detections that need to be checked for association, as any point outside this frustum can be ignored. The estimated object depth, dimension and rotation are then used to create a RoI around the object to further filter out radar detections that are not associated with this object. If there are multiple radar detections inside this RoI, the closest point is considered as the radar detection corresponding to this object.

In the training phase, the object's 3D ground truth bounding box is used to create a tight RoI frustum and associate radar detections to the object. In the test phase, the RoI frustum is calculated using the object's estimated 3D bounding box as explained before. In this case, a parameter δ is used to control the size of the RoI frustum as shown in Fig. 5.3.

This is to account for inaccuracy in the estimated depth values, as the depth of the object at this stage is solely determined using the image-based features. Enlarging the frustum using this parameter increases the chance of including the corresponding radar detections inside the frustum even if the estimated depth is slightly off. The value of δ should be carefully selected, as a large RoI frustum can include radar detections of nearby objects.

The RoI frustum approach makes associating overlapping objects effortless, as objects are separated in the 3D space and would have separate RoI frustums. It also eliminates the multi-detection association problem, as only the closest radar detection inside the RoI frustum is associated to the object. It does not, however, help with the inaccurate z dimension problem,

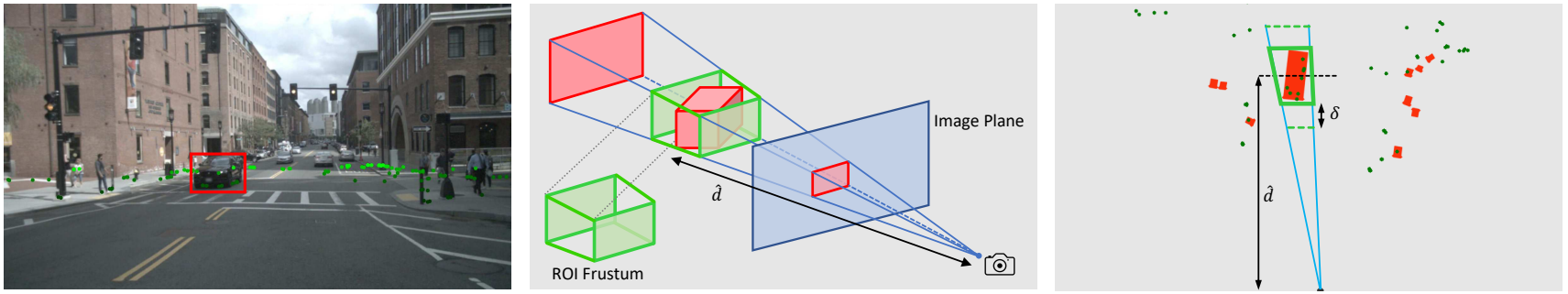


Figure 5.3: Frustum association. An object detected using the image features (left), generating the ROI frustum based on object's 3D bounding box (middle), and the BEV of the ROI frustum showing radar detections inside the frustum (right). δ is used to increase the frustum size in the testing phase. $\hat{d}$ is the ground truth depth in the training phase and the estimated object depth in the testing phase.

as radar detections might be outside the ROI frustum of their corresponding object due to their inaccurate height information.

5.3.2 Pillar Expansion:

To address the inaccurate height information problem, a radar point cloud preprocessing step called pillar expansion is introduced, where each radar point is expanded to a fixed-size pillar as illustrated in Fig. 5.4. Pillars create a better representation for the physical objects detected by the radar, as these detections are now associated with a dimension in the 3D space. Having this new representation, a radar detection is simply considered to be inside a frustum if all or part of its corresponding pillar is inside the frustum, as shown in Fig. 5.2.

5.4 Radar Feature Extraction

After associating radar detections to their corresponding objects, the depth and velocity of the radar detections are used to create complementary features for the image. Particularly, for every radar detection associated to an object, three heat map channels centered at and inside the object's 2D bounding box are generated as shown in Fig. 5.4. The width and height of the heatmaps are proportional to the object's 2D bounding box and are controlled by a parameter α. The heatmap values are the normalized object depth (d) and also the x and y components of the radial velocity (v_x and v_y) in the egocentric coordinate system:

$$F^j_{x,y,i} = \frac{1}{M_i} \begin{cases} f_i & |x - c^j_x| \leq \alpha w^j \quad \text{and} \\ & |y - c^i_y| \leq \alpha h^j \\ 0 & \text{otherwise} \end{cases},$$

where $i \in 1, 2, 3$ is the feature map channel, M_i is a normalizing factor, f_i is the feature value (d, v_x or v_y), c^j_x and c^j_y are the x and y coordinates of the jth object's center point on the image and w^j and h^j are the width and height of the jth object's 2D bounding box. If two objects have overlapping heatmap areas, the one with a smaller depth value dominates, as only the closest object is fully visible in the image.

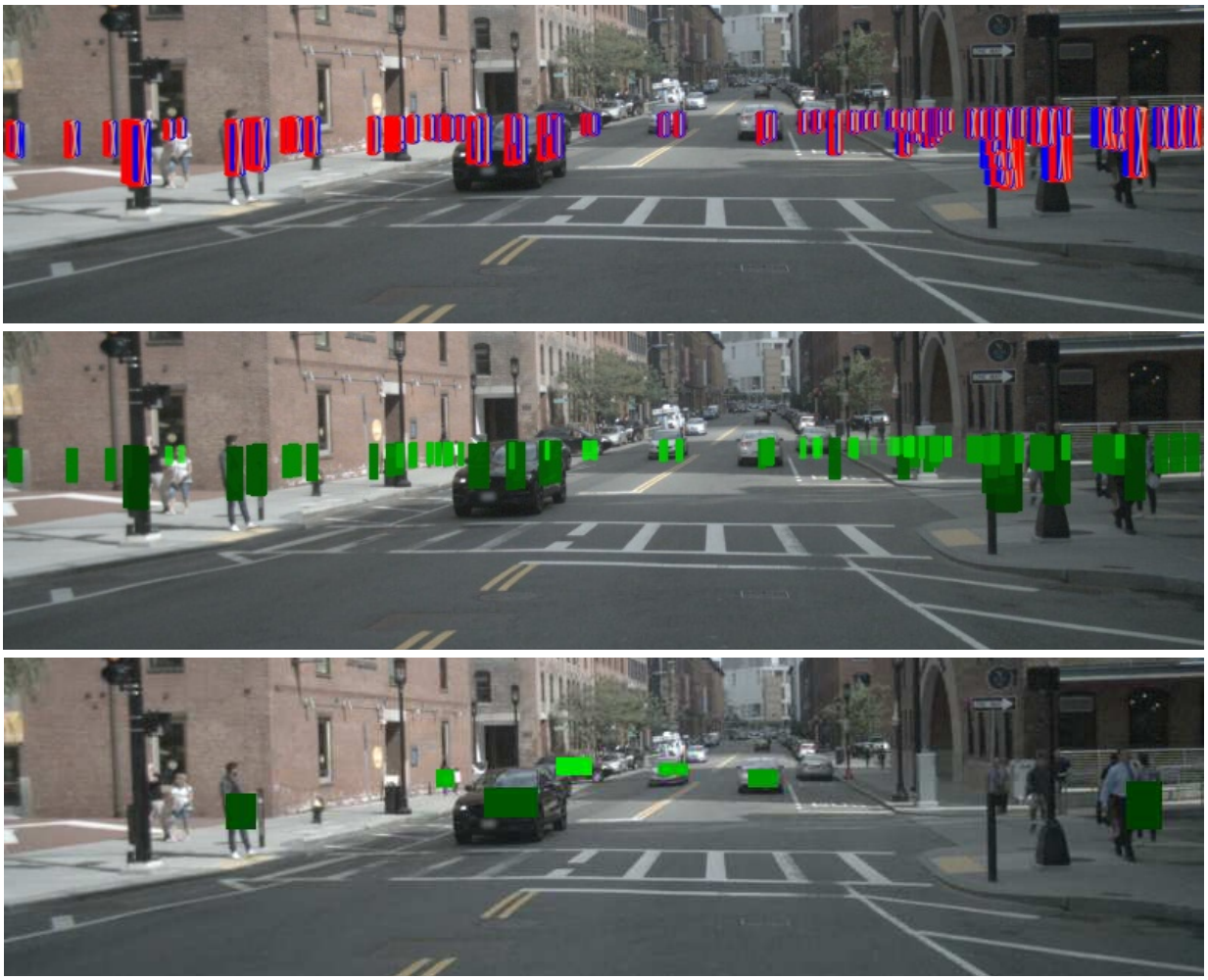


Figure 5.4: Expanding radar points to 3D pillars (top image). Directly mapping the pillars to the image and replacing with radar depth information results in poor association with objects' center and many overlapping depth values (middle image). Frustum association accurately maps the radar detections to the center of objects and minimizes overlapping (bottom image). Radar detections are only associated to objects with a valid ground truth or detection box, and only if all or part of the radar detection pillar is inside the box. Frustum association also prevents associating radar detections caused by background objects such as buildings to foreground objects, as seen in the case of pedestrians on the right hand side of the image.

The generated heat maps are then concatenated to the image features as extra channels. These features are used as inputs to the secondary regression heads to recalculate the object's depth and rotation, as well as velocity and attributes. The velocity regression head estimates the x and y components of the object's actual velocity in the vehicle coordinate system. The attribute regression head estimates different attributes for different object classes, such as moving or parked for the Car class and standing or sitting for the Pedestrian class. The secondary regression heads consist of three convolutional layers with 3×3 kernels followed by a 1×1 convolutional layer to generate the desired output. The extra convolutional layers compared to the primary regression heads help with learning higher level features from the radar feature maps. The last step is decoding the regression head results into 3D bounding boxes. The box decoder block uses the estimated depth, velocity, rotation, and attributes from the secondary regression heads, and takes the other object properties from the primary heads.

5.5 Implementation Details

The pre-trained CenterNet [109] network with the DLA [103] backbone is used as the object detection network. DLA uses iterative deep aggregation layers to increase the resolution of feature maps. CenterNet compares its performance using different backbone architectures, with the Hourglass network [62] performing better than others. The DLA network was chosen for this architecture because it takes significantly less time to train while providing a reasonable performance.

The released CenterNet model trained for 140 epochs on the nuScenes dataset was used to initialize the network. This model by default does not provide velocity and attribute predictions.The velocity and attribute heads were trained for 30 epochs, and the resulting model was then used as the baseline image-based method to compare to. The secondary regression heads in CenterFusion are added on top of the CenterNet backbone network, and are trained using the image and radar features for an additional 60 epochs with a batch size of 26 on two Nvidia P5000 GPUs.

During both training and testing, the image resolution was reduced from the original 1600×900 pixels to 800×450 pixels. Data augmentation is used during training, with random right-left flipping (with a probability of 0.5) and random shifting (from 0 to 20 percent of image size). The same augmentations are also applied to the radar point cloud in reference to the camera coordinate system. No scaling augmentation is applied here as it changes the 3D measurements in the scene. At testing time, only the flip test augmentation was used where an image and its flipped version are fed into the network and the average of the network outputs is used for decoding the 3D bounding boxes. Multi-scale test augmentation as used by CenterNet is not used in the proposed architecture. The pillar size is set to $[0.2, 0.2, 1.5]$ meters in the $[x, y, z]$ directions and δ is set to increase the length of the RoI frustum by 20% in the radial direction at test time.

The L1 loss is used for most of the regression heads, with the exception of the center point heat map head which uses the focal loss and the attributes regression head that uses the Binary Cross Entropy (BCE) loss.

5.6 Results

The proposed radar and camera fusion network is compared with the published state-of-the-art camera-based models on the nuScenes benchmark, and also a LIDARbased method. Table 5.1 shows the results on both test and validation splits of the nuScenes dataset.

CenterFusion is compared with OFT [74], MonoDIS [79] and CenterNet [109] which are camera-based 3D object detection networks, as well as InfoFocus [85] which is a LIDAR-based method. As seen in Table 5.1, CenterFusion outperforms all other methods in the nuScenes NDS score, which is a weighted sum of the mAP and the error metrics. On the test dataset, CenterFusion shows a 12.25% and 16.9% relative increase in the NDS score compared to CenterNet and MonoDIS respectively. The LIDAR-based method InfoFocus shows a better performance in the mAP score compared to other methods, but is significantly outperformed by CenterFusion in the orientation, velocity and attribute error metrics. While CenterNet with the Hourglass [62] backbone network results in a better mAP score compared to CenterFusion (1.2% difference) on the test split, the results on the validation split show

Table 5.1: Performance comparison for 3D object detection on nuScenes dataset. mATE, mASE, mAOE, mAVE and mAAE stand for average translation, scale, orientation, velocity and attribute errors respectively. ↑ indicates that higher is better and ↓ indicates that lower is better. "C", "R" and "L" specify camera, radar and LIDARmodalities respectively.

		Modality					Error ↓				
Method	Dataset	C	R	L	NDS ↑	mAP ↑	mATE	mASE	mAOE	mAVE	mAAE
InfoFocus [85]	test			✓	0.395	**0.395**	**0.363**	0.265	1.132	1.000	0.395
OFT [74]	test	✓			0.212	0.126	0.820	0.360	0.850	1.730	0.480
MonoDIS [79]	test	✓			0.384	0.304	0.738	0.263	0.546	1.533	0.134
CenterNet (HGLS) [109]	test	✓			0.400	0.338	0.658	**0.255**	0.629	1.629	0.142
CenterFusion (DLA)	test	✓	✓		**0.449**	0.326	0.631	0.261	**0.516**	**0.614**	**0.115**
CenterNet (DLA) [109]	val	✓			0.328	0.306	0.716	0.264	0.609	1.426	0.658
CenterFusion (DLA)	val	✓	✓		**0.453**	**0.332**	**0.649**	**0.263**	**0.535**	**0.540**	**0.142**

that CenterFusion outperforms CenterNet by 2.6% when both networks use the same DLA [103] backbone. The validation set results also show CenterFusion improving CenterNet in all the other metrics. CenterFusion shows an absolute gain of 38.1% and 62.1% relative increase in the NDS and velocity error metrics compared to CenterNet, which demonstrates the effectiveness of using radar features.

Table 5.2 compares the per-class mAP results for both test and validation splits. While CenterNet with an Hourglass backbone has a higher mAP than CenterFusion for most classes in the test set, it is outperformed by CenterFusion on the validation set where the DLA backbone is used for both methods. The most improved classes on the validation set are the motorcycle and car with 5.6% and 4.0% absolute mAP increase respectively.

Figures 5.5, 5.6, 5.7 and 5.8 demonstrates the 3D object detection results in both camera and BEV. It shows the detection results from CenterFusion (row 1 & 2) and CenterNet (row 3 & 4) for 4 different scenes. The radar point clouds are also shown in the CenterFusion BEV results. Compared to CenterNet, the results from CenterFusion show a better fit for 3D boxes in most cases, especially objects at a larger distance, such as the far vehicle in the second scene. Additionally, the velocity vectors estimated by CenterFusion show a significant improvement compared to the CenterNet results, as seen in the second and third scenes.

5.6.1 Ablation Study

The effectiveness of the proposed fusion algorithm is validated by conducting an ablation study on the nuScenes validation set. The CenterNet model is used as the baseline, and the effectiveness of the pillar expansion, frustum association and flip testing steps are studied on the detection results. Table 5.3 shows the overall detection results of the ablation study.

In the first experiment, only pillar expansion is applied to the radar point clouds, and map the 3D pillars to the image plane and obtain their equivalent 2D bounding boxes. These boxes are then filled with the depth and velocity values of their corresponding radar detections and used as the radar feature maps, as shown in Fig. 5.4. According to Table 5.3, this simple association method results in a 15.4% relative improvement on the NDS score and 1.0% absolute improvement on the mAP compared to the baseline.

Table 5.2: Per-class performance comparison for 3D object detection on nuScenes dataset.

Method	Dataset	Modality			mAP ↑									
		C	R	L	Car	Truck	Bus	Trailer	Const.	Pedest.	Motor.	Bicycle	Traff.	Barrier
InfoFocus [85]	test			✓	**0.779**	**0.314**	**0.448**	**0.373**	**0.107**	**0.634**	0.290	0.061	0.465	0.478
MonoDIS [79]	test	✓			0.478	0.220	0.188	0.176	0.074	0.370	0.290	**0.245**	0.487	0.511
CenterNet (HGLS) [109]	test	✓			0.536	0.270	0.248	0.251	0.086	0.375	0.291	0.207	**0.583**	**0.533**
CenterFusion (DLA)	test	✓	✓		0.509	0.258	0.234	0.235	0.077	0.370	**0.314**	0.201	0.575	0.484
CenterNet (DLA) [109]	val	✓			0.484	0.231	0.340	0.131	0.035	0.377	0.249	**0.234**	0.550	0.456
CenterFusion (DLA)	val	✓	✓		**0.524**	**0.265**	**0.362**	**0.154**	**0.055**	**0.389**	**0.305**	0.229	**0.563**	**0.470**

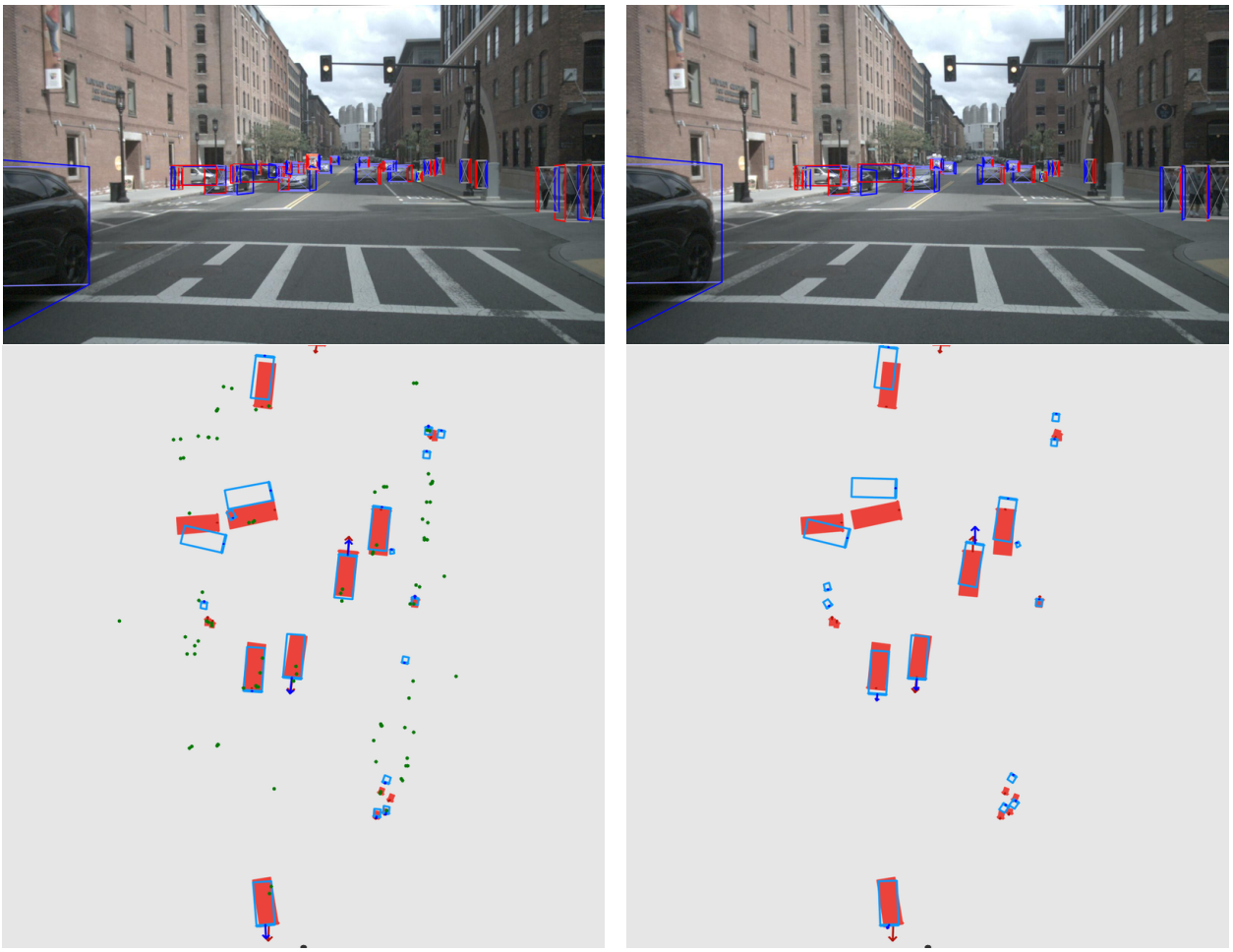


Figure 5.5: Qualitative results from CenterFusion (left) and CenterNet (right) in camera view and BEV. In the BEV plots, detection boxes are shown in cyan and ground truth boxes in red. The radar point cloud is shown in green. Red and blue arrows on objects show the ground truth and predicted velocity vectors respectively.

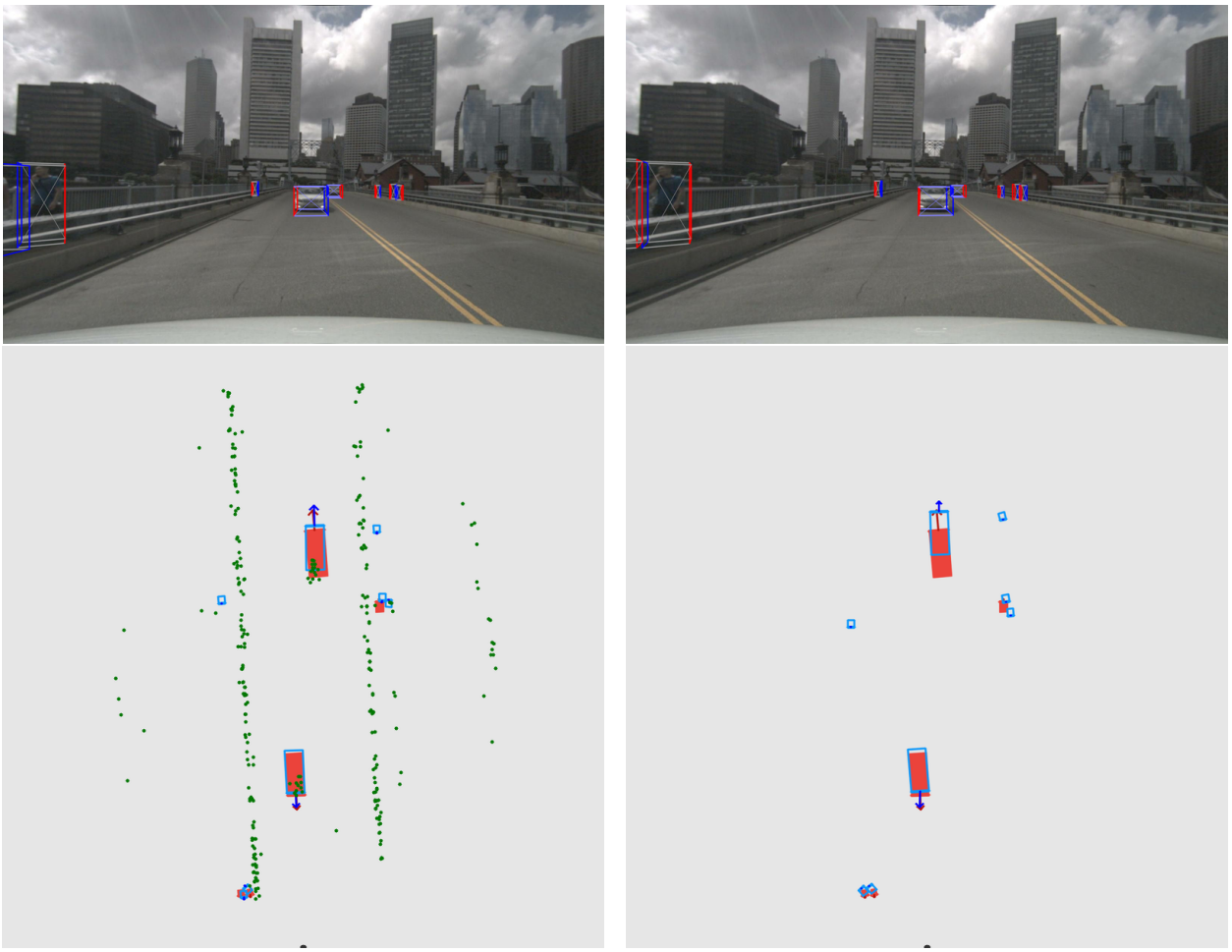


Figure 5.6: Qualitative results from CenterFusion (left) and CenterNet (right) in camera view and BEV, cont.

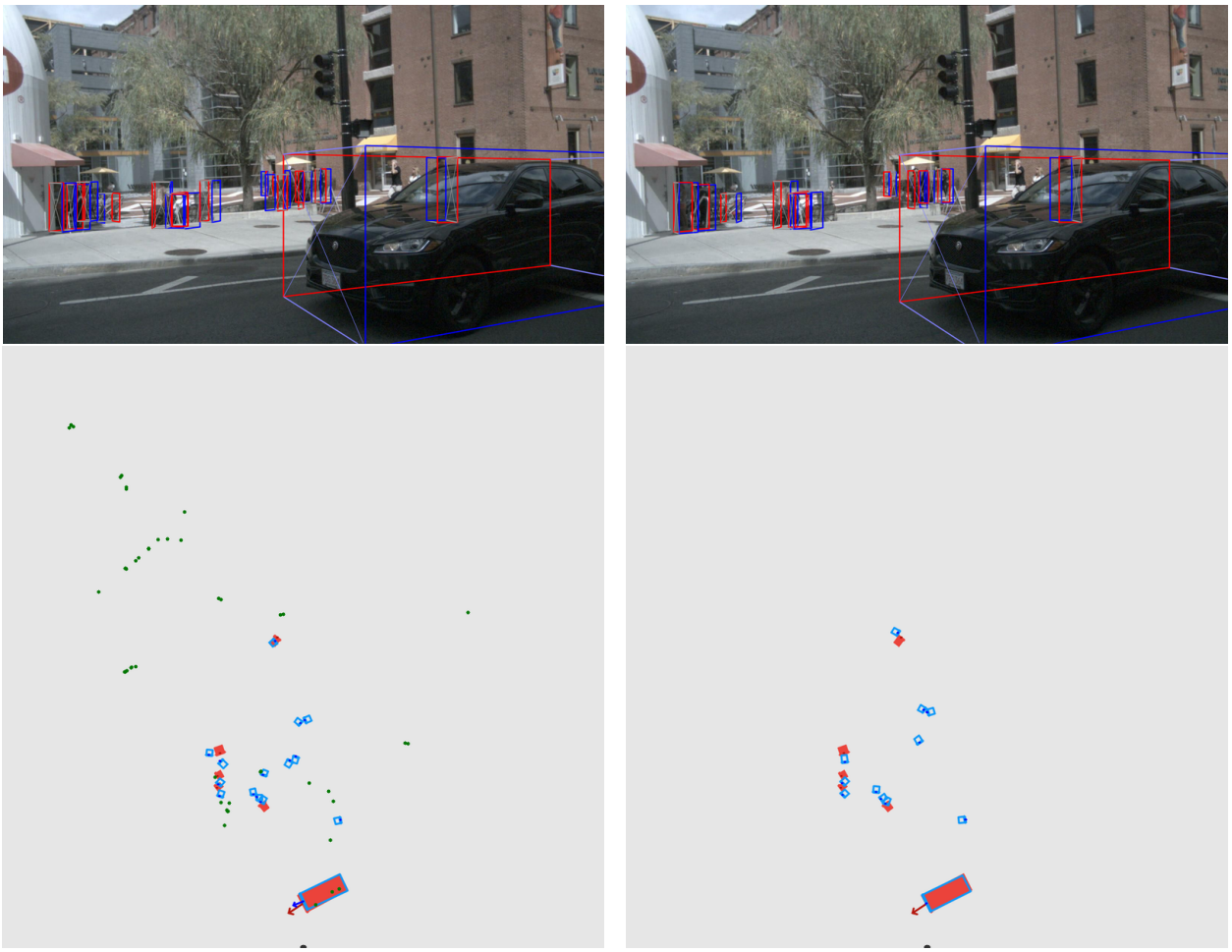


Figure 5.7: Qualitative results from CenterFusion (left) and CenterNet (right) in camera view and BEV, cont.

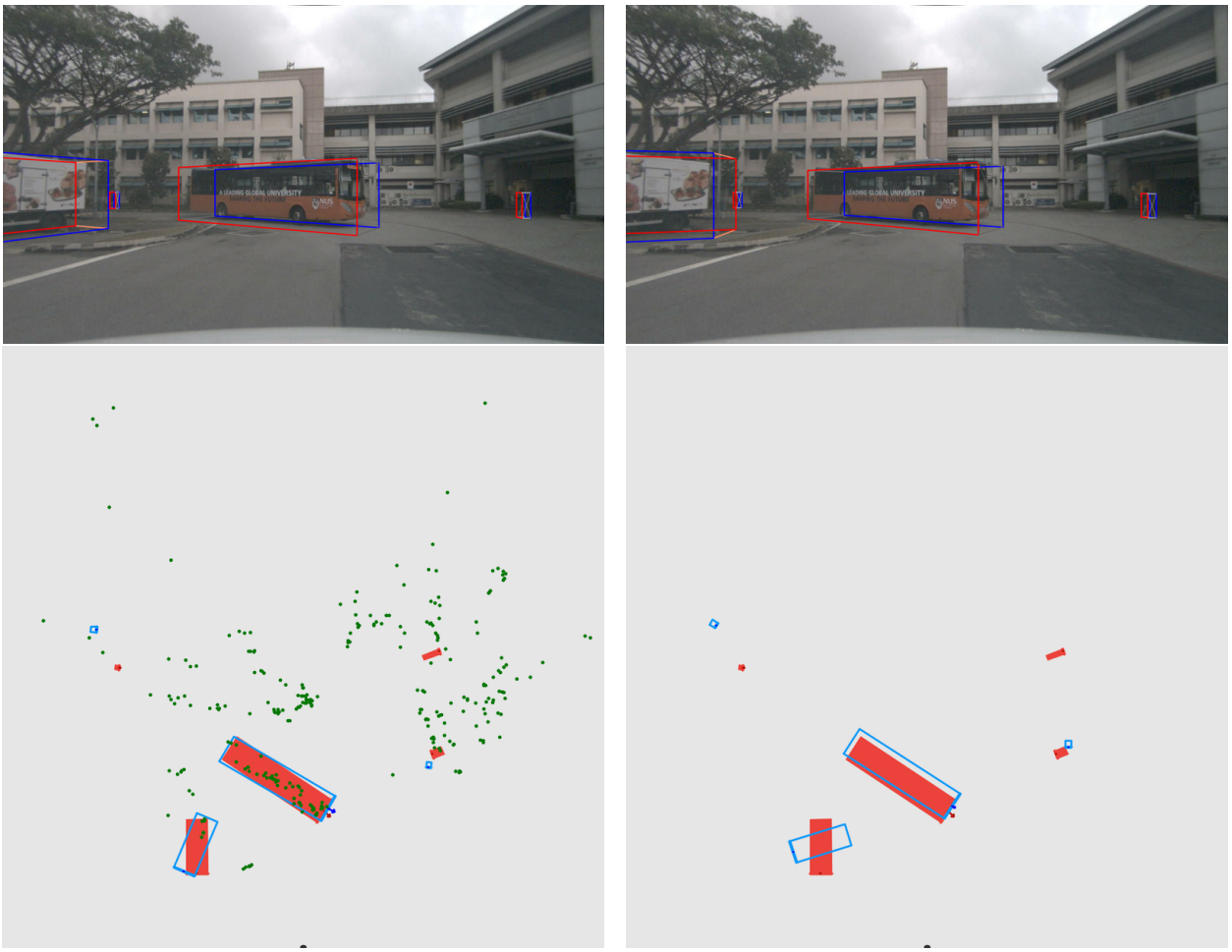


Figure 5.8: Qualitative results from CenterFusion (left) and CenterNet (right) in camera view and BEV, cont.

Table 5.3: Overall ablation study on nuScenes validation set. Improvement percentages in each row are relative to the baseline method. (PE: Pillar Expansion, FA: Frustum Association, FT: Flip Test)

Method	Cam	Rad	PE	FA	FT	NDS ↑	mAP ↑	mATE ↓	mASE ↓	mAOE ↓	mAVE ↓	mAAE ↓
Baseline	✓	-	-	-	-	0.328	0.306	0.716	0.264	0.609	1.426	0.658
CenterFusion	✓	✓	✓	-	-	+15.4%	+1.0%	-2.0%	+1.1%	-4.4%	-13.1%	-68.6%
CenterFusion	✓	✓	-	✓	-	+25.9%	+2.0%	-2.8%	+1.0%	-7.4%	-48.1%	-75.9%
CenterFusion	✓	✓	✓	✓	-	+34.5%	+4.3%	-5.3%	+1.1%	-10.0%	-61.9%	-78.0%
CenterFusion	✓	✓	✓	✓	✓	**+37.8%**	**+8.4%**	**-9.4%**	**-0.5%**	**-11.6%**	**-62.0%**	**-78.3%**

Table 5.4: Class-based ablation study results on nuScenes validation set.

Method	Cam	Rad	PE	FA	FT	Car	Truck	Bus	Trailer	Const.	Pedest.	Motor.	Bicycle	Traff.	Barrier
Baseline	✓	-	-	-	-	48.4	23.1	34.0	13.1	3.5	37.7	24.9	23.4	55.0	45.6
CenterFusion	✓	✓	✓	-	-	+0.6	+0.7	-2.1	+0.9	+0.6	+0.9	+1.9	-2.5	+0.1	+0.8
CenterFusion	✓	✓	-	✓	-	+1.0	+1.0	-2.1	+0.9	+0.9	0.0	+2.1	-1.9	+0.2	+0.8
CenterFusion	✓	✓	✓	✓	-	+2.8	+2.1	-1.2	+1.4	+1.1	+0.1	+3.8	-1.1	+0.4	+0.8
CenterFusion	✓	✓	✓	✓	✓	**+4.1**	**+3.4**	**+2.7**	**+1.8**	**+1.8**	**+1.2**	**+5.5**	-0.7	**+1.3**	**+1.5**

In the next experiment, only the frustum association method is used by directly applying it on the radar point clouds without converting them to pillars first. This improves the NDS score by 25.9% relatively and mAP by 2.0%. Applying both pillar expansion and frustum association results in a relative 35.5% and absolute 4.3% improvement on the NDS and mAP scores respectively. Flip testing adds another 3.3% improvement on the NDS score and 3.9% on the mAP, resulting in a total of 37.8% and 8.4% improvement on NDS and mAP compared to the baseline method.

Table 5.4 shows the per-class contribution of each step on the mAP. According to the results, both pillar expansion and frustum association steps have contributed to the improvement of mAP in most object classes. The only class that has not improved from the baseline is the bicycle class, in which the CenterNet mAP score is better than CenterFusion by 0.5%.

5.7 Conclusion

In summary, a new radar and camera fusion algorithm called CenterFusion was proposed to exploit radar information for robust 3D object detection. CenterFusion accurately associates radar detections to objects on the image using a frustum-based association method, and creates radar-based feature maps to complement the image features in a middle-fusion approach. The proposed frustum association method uses preliminary detection results to generate a RoI frustum around objects in 3D space, and maps the radar detection to the center of objects on the image. A pillar expansion method was also used to compensate for the inaccuracy in radar detections' height information, by converting radar points to fixed-size pillars in the 3D space. The proposed method was evaluated on the challenging nuScenes 3D detection benchmark, where it outperformed the state-of-the-art camera-based object detection methods.

Chapter 6

Radar-Camera Fusion for 3D Object Tracking

3D multi-object tracking is a crucial component in the perception system of autonomous driving vehicles. Tracking all dynamic objects around the vehicle is essential for tasks such as obstacle avoidance and path planning. While sensor fusion has been widely used in object detection networks in recent years, most existing multi-object tracking algorithms either rely on a single input modality, or do not fully exploit the information provided by multiple sensing modalities. In this chapter, an end-to-end network for joint object detection and tracking based on radar and camera sensor fusion is proposed. The proposed method uses the center-based radar-camera fusion algorithm introduced in the previous chapter for object detection, and utilizes a greedy algorithm for object association in consecutive frames. The proposed greedy algorithm uses the depth, velocity and 2D displacement of the detected objects to associate them through time. This makes the proposed tracking algorithm very robust to occluded and overlapping objects, as the depth and velocity information are very effective cues for distinguishing these objects. This algorithms, hereafter referred to as CFTrack, is evaluated on the challenging nuScenes dataset, where it achieves 20.0 Average Multi Object Tracking Accuracy (AMOTA) and outperforms all vision-based 3D tracking methods in the benchmark, as well as the nuScenes' baseline LIDAR-based method. The proposed network takes as input the current image frame and radar detections in addition to the previous frame and detected objects. The outputs are 3D object detection results

and tracking IDs for all detected objects. Every detected object is also associated with an estimated absolute velocity in the global coordinate system. CFTrack is online and real-time with a runtime of 35ms per image, making it very suitable for autonomous driving applications.

The object association step in CFTrack is based on a simple greedy algorithm similar to CenterTrack [108]. While CenterTrack only uses the objects' 2D displacement in consecutive images to associate them, CFTrack utilizes a greedy algorithm based on a weighted cost function calculated from the object's estimated depth and velocity in addition to their 2D displacement. This significantly improves the ability of the network to correctly associate occluded and overlapping objects, as the depth and velocity information provide valuable clues to distinguish these objects. Additionally, CFTrack uses the fused radar and image features to predict the objects' displacement in consecutive frames, which makes these predictions more accurate compared to just using the visual information.

Experiments conducted on the challenging nuScenes dataset [8] show that CFTrack outperforms all other image-based tracking methods on the nuScenes benchmark, as well as the baseline LIDAR-based method AB3DMOT [91]. It achieves 20.0% AMOTA, outperforming CenterTrack [111] by a factor of 4, while running at 28 frames per second.

6.1 Preliminaries

The proposed 3D tracking algorithm is based on CenterFusion [61], a 3D object detection algorithm introduced in the previous chapter. CenterFusion takes an image $I \in \mathbb{R}^{W \times H \times 3}$ and a set of radar detections $P_i = (x^i, y^i, z^i, v_x^i, v_y^i)$ where (x^i, y^i, z^i) are the coordinates of the point i in the radar point cloud, and (v_x^i, v_y^i) are the radial velocities in the x and y directions, respectively. These coordinates are according to the vehicle coordinate system, where x is forward, y is to the left and z is upward from the drivers point of view.

CenterFusion first uses a center point detection method called CenterNet [109] to detect the centerpoint of objects by estimating a heatmap $\hat{Y} \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ where R is the down-sampling factor and C is the number of object categories in the dataset. The local maxima in the estimated heatmap $\hat{Y}$ correspond to the centers of detected objects in the image. The

ground truth heatmap Y is generated by rendering a Gaussian-shaped peak at the center points of each object, calculated as the center point of their corresponding bounding box.

The network uses regression layers to generate preliminary 3D bounding boxes for all objects, then associates the radar detections to these preliminary 3D detections using a frustum-based association method. To do this, the radar detections are first expanded into pillars with predefined dimensions. A frustum is then formed around each detected object, and radar pillars inside the frustum are associated with that object. If there are multiple radar pillars inside the frustum, the closest one is kept and others are discarded.

Based on the association results, the depth and velocity of the radar detection are mapped to their corresponding objects on the image. These values are represented as separate heatmap channels and are concatenated to the image-based features. These fused features are then used to improve the preliminary detection results by re-calculating the depth, size, orientation and other object attributes. Additionally, a velocity vector is estimated for every detected object.

CenterFusion uses an objective function based on the focal loss, defined as:

$$L_i = \frac{1}{N} \sum_{xyc} \begin{cases} (1 - \hat{Y}_{xyc})^{\alpha} \log(\hat{Y}_{xyc}) & Y_{xyc} = 1 \\ (1 - Y_{xyc})^{\beta} (\hat{Y}_{xyc})^{\alpha} \log(1 - \hat{Y}_{xyc}) & \text{otherwise} \end{cases}, \tag{6.1}$$

where N is the number of objects, $Y \in [0,1]^{\frac{W}{R} \times \frac{H}{R} \times C}$ is the annotated objects' ground-truth heatmap and α and β are the hyper-parameters of the focal loss. After detecting objects' center point, different regression heads are used to regress to size, orientation, depth and velocity of the detected objects.

6.2 CFTrack

Following CenterTrack [108], the tracking problem is approached from a local perspective where an object's identity is preserved across consecutive frames without re-establishing associations if the object leaves the frame. Both camera and radar data from the previous frame are used to improve the ability to track occluded objects in the current frame. CFTrack uses the fused radar and image features to estimate objects' displacement in consecutive

frames, which is used for object association through time. In the association step, a greedy algorithm is proposed that leverages objects' velocity and depth information in addition to their 2D displacement for accurate association through time.

6.2.1 Problem Formulation

The inputs to CFTrack are the current and previous image frames $I^{(t-1)}, I^{(t)} \in \mathbb{R}^{W \times H \times 3}$, the current and previous radar detections $P^{(t-1)}, P^{(t)} \in \mathbb{R}^{N \times 5}$ where N is the number of radar detections, and the tracked objects from the previous frame $T^{(t-1)} = \{b_0^{(t-1)}, b_1^{(t-1)}, ...\}$. The tracked objects are represented by $b = (p, d, v, w, id)$ where $p \in \mathbb{R}^2$ is the object's center location, $d \in \mathbb{R}$ is the object's depth, $v \in \mathbb{R}^2$ is object's velocity, $w \in [0, 1]$ is the detection confidence and id is an integer representing the unique identity of the tracked object. For every frame, the goal is to detect and track objects $T^{(t)} = \{b_0^{(t)}, b_1^{(t)}, ...\}$ and assign a consistent id to the objects in consecutive frames. The detection and association of objects are done in a single deep network trained end-to-end.

6.2.2 Detection Network

The overall network architecture is shown in Fig. 6.1. The CenterFusion network is modified to take as input the current image frame $I^{(t)}$ and radar detections $P^{(t)}$, in addition to the previous image frame $I^{(t-1)}$, radar detections $P^{(t-1)}$ and detected objects. The outputs are 3D bounding boxes for all detected objects and an absolute velocity for each object, reported in the x and y directions in the vehicle's coordinate system. The previous detections are represented as a single channel heatmap using a 2D Gaussian kernel. Including the previous image, radar detections and detected objects helps the network to better estimate the location of objects in the current frame. The radar information from previous frame further improves the ability of network to detect objects even if the visual evidence is not present due to occlusion.

Besides the object detection results for the current frame, the modified network also estimates the 2D displacement of the detected objects between the current and previous frames, using the concatenated radar and image features. Having the radar depth and

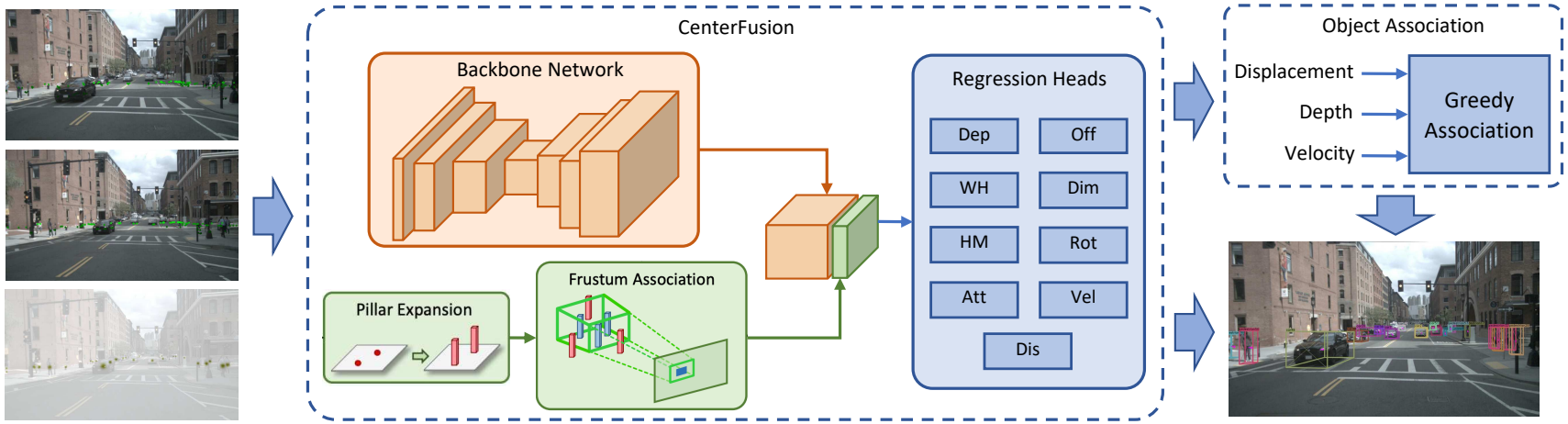


Figure 6.1: CFTrack network architecture. The inputs to the network are shown on the left which includes the current image and radar point clouds (top), the previous image frame and radar point clouds (middle) and the previous detection results in the form of class-agnostic heatmaps (bottom). The radar point clouds are shown on the input images. An additional regression head ("Dis") is added to the model, which uses the fused radar and image features to predict objects displacement in consecutive frames. The greedy algorithm in the association step uses the displacement, depth and velocity of each object to associate it to previous detections. The output is 3D bounding boxes and track IDs for all detected objects.

velocity information in addition to the image features from the current and previous frames helps the network to generate more accurate object displacement predictions.

6.2.3 Object Association

A greedy algorithm is used to associate the detected objects over time. These objects are represented by $a = (p, d, v, c)$ where $p \in \mathbb{Z}^2$ is the object's center in pixels, $d \in \mathbb{R}$ is the object's depth, $v \in \mathbb{R}^2$ is the object's velocity, and $c \in C$ is the object's category. Similar to [108], the displacement is calculated by a regression layer in the form of two output channels $\hat{D}^{(t)} \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$ representing the displacement of the center of the objects on the image, as shown in Fig. 6.2. Similar to the other regression heads, the L1 loss is used as the objective function to train this layer.

To associate objects across time, a cost function is defined based on the objects' depth, velocity and displacement on the image:

$$Cost_{t,t-1} = \begin{cases} \alpha \cdot \mathcal{L}_{pixel} + \beta \cdot \mathcal{L}_{depth} + \delta \cdot \mathcal{L}_{velocity} & c_t = c_{t-1} \\ \infty & c_t \neq c_{t-1} \end{cases} \tag{6.2}$$

$$\mathcal{L}_{pixel} = (x_t - x_{t-1})^2 + (y_t - y_{t-1})^2 \tag{6.3}$$

$$\mathcal{L}_{depth} = (d_t - d_{t-1})^2 \tag{6.4}$$

$$\mathcal{L}_{velocity} = (vx_t - vx_{t-1})^2 + (vy_t - vy_{t-1})^2 \tag{6.5}$$

where x, y is the object's center, d is the object's depth, and vx, vy are the velocity of the object in x and y directions respectively. $\alpha, \beta, \delta \in \mathbb{R}^+$, are tunable parameters.

For every detected object at position p, The greedy algorithm looks for prior detections within a radius r from $p - D_p$. If there are unmatched prior detections at that position, the above cost function is calculated to determine the distance between these detections, and match the object with the previous detections with the lowest cost. For every unmatched detection, a new track is created.

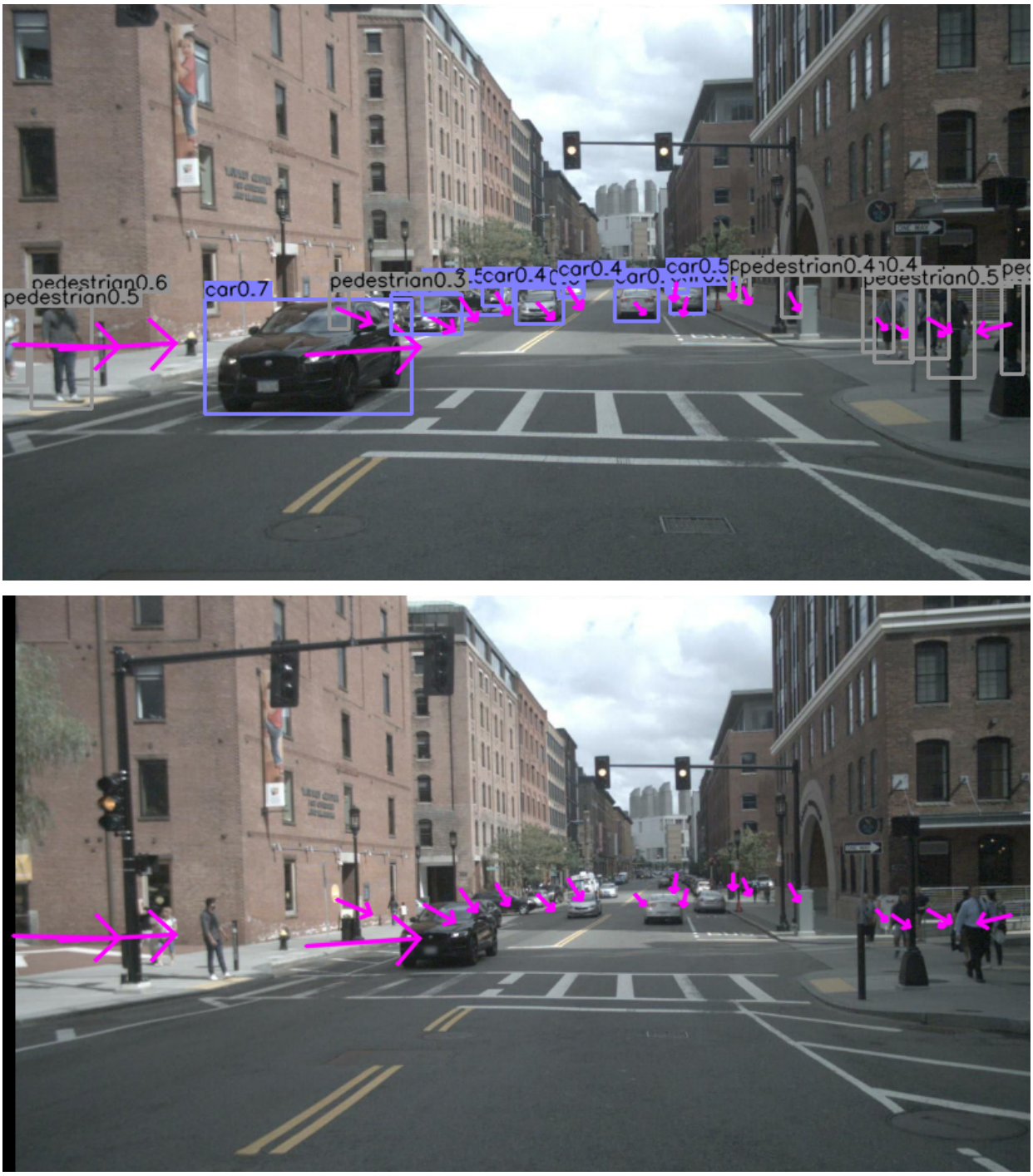


Figure 6.2: Top: Object displacement from the previous frame, represented by arrows pointing from the center of each object to the estimated center of the same object in the previous frame. Bottom: Previous frame. Displacement arrows re-drawn for comparison.

6.3 Experiments

6.3.1 Dataset and Evaluation Metrics

CFTrack is evaluated on the nuScenes dataset [8], a large-scale dataset for autonomous driving with annotations for 3D object detection and tracking containing camera, radar and LIDAR data. It provides 1000 different sequences from which 700 sequences are used for training, 150 sequences for validation and 150 sequences for testing. Each sequence is comprised of 40 annotated frames, each containing camera, radar and LIDAR samples. Samples are obtained from 6 different cameras and 5 different radars.

The main evaluation metric used in the nuScenes benchmark, AMOTA, is a weighted average of the recall-normalized Multi Object Tracking Accuracy (MOTA) metric at different recall thresholds:

$$MOTAR = \max(0, 1 - \frac{IDS_r + FP_r + FN_r - (1-r) * P}{r * P})$$

$$\text{AMOTA} = \frac{1}{n-1} \sum_{r \in \{\frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1\}} MOTAR$$

where r is the recall threshold, IDS_r is the number of identity switches, FP_r is the number of false positives, FN_r is the number of false negatives and P is the total number of annotated objects in all frames.

6.3.2 Implementation Details

Following CenterFusion [61], an input resolution of 800×448 is used and horizontal flipping and random shifts are applied for regularization. The Deep Layer Aggregation (DLA) network is used as the backbone for extracting image features, optimized with the Adam [38]. The CFTrack network is trained for 60 epochs with a batch size of 24 and a learning rate of $1.2e$-4, starting from a pre-trained CenterFusion network trained for 170 epochs. The network is trained on a machine with an Intel Xeon E5-1650 CPU and two Quadro P5000 GPUs. The runtime is tested on a machine with an Intel Xeon E5-1607 CPU and a TITAN X GPU.

6.4 Results

Table 6.1 compares CFTrack with some other published methods in the nuScenes object tracking benchmark. Specifically, it is compared with the CenterTrack [108] algorithm using both image-based and LIDAR-based detection results, as well as the AB3DMOT [91] with three different LIDAR-based object detection algorithms. According to the table, CFTrack outperforms both methods by achieving an AMOTA score of 20.0%, improving the vision-based CenterTrack algorithm by about 15% (by a factor of 4) and the LIDAR-based CenterTrack algorithm by 9.2%. CFTrack also outperforms CenterTrack in the MOTAR, MOTA, MOTP and Recall metrics.

Given the similarity of the tracking algorithms in CFTrack and CenterTrack, these results demonstrate the effect of utilizing radar data in both detection and tracking stages. Both methods use a greedy algorithm for associating objects, but CFTrack also takes advantage of the depth and velocity of the detected objects to better associate them through time. Additionally, the velocity data provided by the radar enables the network to predict the objects' displacement in the image more accurately, further improving object association.

Table 6.2 shows AMOTA for each class. According to the results, CFTrack significantly outperforms all the other methods in the Car, Pedestrian, Motorcycle and Bicycle categories, while AB3DMOT with the Megvii detector performs better in the Truck, Bus and Trailer categories. Note that all categories where CFTrack is outperformed are large objects, namely Truck, Bus and Trailer. One explanation could be the fact that it is more difficult for the underlying fusion algorithm to correctly associate many radar detections obtained from these large objects to their corresponding 3D bounding boxes, resulting in lower accuracy in estimated depth and velocity for these objects.

On average, CFTrack achieves a runtime of 35ms per image (28 fps), which makes it suitable for the real-time autonomous driving applications.

Table 6.1: Evaluation of CFTrack on the nuScenes test set. Metrics are defined in [8].

Method	Modality			Time(ms)	AMOTA	AMOTP	MOTAR	MOTA	MOTP	Recall
	Cam	Rad	LIDAR							
AB3DMOT [91] + Mapillary			✓	-	0.018	**1.790**	0.091	0.020	**0.903**	0.353
AB3DMOT [91] + PointPillars			✓	-	0.029	1.703	0.243	0.045	0.824	0.297
AB3DMOT [91] + Megvii			✓	-	0.151	1.501	**0.552**	**0.154**	0.402	0.276
CenterTrack [108]	✓			45	0.046	1.543	0.231	0.043	0.753	0.233
CenterTrack [108] + Megvii	✓		✓	45	0.108	0.989	0.267	0.085	0.349	0.412
CFTrack	✓	✓		35	**0.200**	1.292	0.353	0.151	0.766	**0.420**

Table 6.2: Per-class evaluation results.

Method	Modality			AMOTA						
	Cam	Rad	LIDAR	Car	Truck	Bus	Trailer	Pedest.	Motor.	Bicycle
AB3DMOT [91] + Mapillary			✓	0.125	0.000	0.000	0.000	0.000	0.000	0.000
AB3DMOT [91] + PointPillars			✓	0.094	0.000	0.066	0.000	0.039	0.000	0.000
AB3DMOT [91] + Megvii			✓	0.278	**0.013**	**0.408**	**0.136**	0.141	0.081	0.000
CenterTrack [108]	✓			0.202	0.004	0.072	0.000	0.030	0.011	0.000
CenterTrack [108] + Megvii	✓		✓	0.341	0.012	0.256	0.000	0.142	0.005	0.000
CFTrack	✓	✓		**0.546**	0.000	0.107	0.075	**0.346**	**0.206**	**0.114**

Chapter 7

Conclusion and Future Work

This dissertation focused on radar and camera sensor fusion for autonomous driving applications and proposed different algorithms fro 2D and 3D object detection, 3D multi-object tracking and velocity estimation. It was shown that radar detections could be successfully utilized to generate 2D object proposals for two-stage object detection networks. This substantially reduced the amount of processing and time required for generating object proposals by eliminating the need to process the image using a region proposal network.

Radar data was also used in a two-stage joint object detection and depth estimation network where several 3D object proposals are generated from each radar detection. These 3D proposals are then used in conjunction with image features obtained from a deep neural network to generate 2D object proposals as well as accurate depth estimations for each proposal. The experiments conducted in this dissertation show that the proposed method outperforms other published radar-camera fusion-based object detection methods in the nuScenes benchmark while at the same time accurately estimates the distance to every detection. It is important to mention that an in-depth comparison between the radar-based proposal generation algorithms used in this work and RRPN was not performed and is left as a possible future work. This comparison could be very helpful in determining how much improvement is made by modifying the radar-based proposal generation method, and compare that to the improvements made by adding image-based proposal generation as well.

This dissertation also studied the effectiveness of radar and camera fusion for 3D object detection and velocity estimation. The proposed algorithm, CenterFusion, extracts and fuses

radar and image features to perform 3D object detection as well as object velocity estimation without requiring any temporal information. This is done by first obtaining preliminary detections from the image, and then using a frustum-based association method to accurately associate the radar detections to their corresponding objects in the image. CenterFusion outperforms all image-based 3D object detection methods on the nuScenes benchmark, demonstrating the effectiveness of radar and camera fusion for 3D object detection.

Moreover, this dissertation also focused on object tracking by proposing a online and real-time 3D MOT and velocity estimation network based on radar and camera fusion. While visual object tracking methods usually have difficulty tracking occluded or overlapping objects, the distance values provided by the radar could be very helpful cues for identifying and distinguishing these objects. Additionally, the velocity values obtained from the radar could be utilized to predict objects' direction of movement and consequently improve tracking results. This dissertation proposed a method called CFTrack, which modifies CenterFusion and provides an end-to-end trainable network capable of detecting and tracking objects in 3D and also accurately estimating objects' velocity. CFTrack is currently the only tracking algorithm based on radar and camera data on the nuScenes tracking benchmark, outperforming all published vision-only methods as well as the baseline LIDAR-based method in this benchmark.

Radar data could also be fused with data from other depth sensors such as LIDARs and RGBD cameras for both object detection and object tracking applications. Radar point clouds do not provide much information about the objects' dimensions and geometry, rendering them not suitable for object classification. Point clouds obtained from LIDARs or RGBD cameras on the other hand are more dense and could be used for objects classification. The depth and velocity information obtained from radars could complement this information and help with detecting objects and predicting their direction of movement. This could be investigated further as a future research direction, with the goal of design and implementation of a point-cloud-based fusion algorithm for 3D object detection and tracking.

Bibliography

[1] Asvadi, A., Garrote, L., Premebida, C., Peixoto, P., and Nunes, U. (2017). Multimodal vehicle detection: Fusing 3D-LIDAR and color camera data. *Pattern Recognition Letters*, 115. 21

[2] Asvadi, A., Girão, P., Peixoto, P., and Nunes, U. (2016). 3d object tracking using rgb and lidar data. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pages 1255–1260. 7

[3] Bar-Shalom, Y. and Tse, E. (1975). Tracking in a cluttered environment with probabilistic data association. *Automatica*, 11(5):451–460. 18

[4] Baser, E., Balasubramanian, V., Bhattacharyya, P., and Czarnecki, K. (2019). Fantrack: 3d multi-object tracking with feature association network. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1426–1433. IEEE. 16

[5] Bergmann, P., Meinhardt, T., and Leal-Taixe, L. (2019). Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 941–951. 16

[6] Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE. 16

[7] Blackman, S. S. (1990). Association and fusion of multiple sensor data. *Multitarget-multisensor tracking: advanced applications*. 18

[8] Caesar, H., Bankiti, V., Lang, A. H., Vora, S., Liong, V. E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., and Beijbom, O. (2019). nuscenes: A multimodal dataset for autonomous driving. *arXiv preprint arXiv:1903.11027*. viii, 11, 24, 41, 49, 70, 76, 78

[9] Castanedo, F. (2013). A Review of Data Fusion Techniques. *The Scientific World Journal*, 2013:1–19. 18, 19, 20

[10] Chadwick, S., Maddern, W., and Newman, P. (2019). Distant Vehicle Detection Using Radar and Vision. *arXiv:1901.10951 [cs]*. 22

[11] Chang, M.-F., Lambert, J., Sangkloy, P., Singh, J., Bak, S., Hartnett, A., Wang, D., Carr, P., Lucey, S., Ramanan, D., et al. (2019). Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757. 17

[12] Chen, X., Ma, H., Wan, J., Li, B., and Xia, T. (2017). Multi-view 3d object detection network for autonomous driving. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 7, 15, 21

[13] Choi, J., Ulbrich, S., Lichte, B., and Maurer, M. (2013). Multi-target tracking using a 3d-lidar sensor for autonomous vehicles. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pages 881–886. IEEE. 7

[14] Ciaparrone, G., Sánchez, F. L., Tabik, S., Troiano, L., Tagliaferri, R., and Herrera, F. (2020). Deep learning in video multi-object tracking: A survey. *Neurocomputing*, 381:61–88. 5

[15] Dai, J., Li, Y., He, K., and Sun, J. (2016). R-fcn: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems*, pages 379–387. 3

[16] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1. 3

[17] Dempster, A. P. (1968). A generalization of Bayesian inference. *Journal of the Royal Statistical Society: Series B (Methodological)*, 30(2):205–232. 20

[18] Devagiri, R., Iyer, N. C., and Maralappanavar, S. (2020). Real-time RADAR and LIDAR sensor fusion for automated driving. In Agarwal, S., Verma, S., and Agrawal, D. P., editors, *Machine Intelligence and Signal Processing*, pages 137–147, Singapore. Springer Singapore. 22

[19] Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., and Tian, Q. (2019). Centernet: Keypoint triplets for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6569–6578. 4

[20] Eigen, D., Puhrsch, C., and Fergus, R. (2014). Depth map prediction from a single image using a multi-scale deep network. In *Advances in neural information processing systems*, pages 2366–2374. 51

[21] Fang, K., Xiang, Y., Li, X., and Savarese, S. (2018). Recurrent autoregressive networks for online multi-object tracking. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 466–475. IEEE. 16

[22] Fang, Y., Zhao, H., Zha, H., Zhao, X., and Yao, W. (2019). Camera and lidar fusion for on-road vehicle tracking with reinforcement learning. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1723–1730. IEEE. 7

[23] Fayyad, J., Jaradat, M. A., Gruyer, D., and Najjaran, H. (2020). Deep Learning Sensor Fusion for Autonomous Vehicle Perception and Localization: A Review. *Sensors (Basel, Switzerland)*, 20(15). ix, 1, 7, 8, 9, 14

[24] Feichtenhofer, C., Pinz, A., and Zisserman, A. (2017). Detect to track and track to detect. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3038–3046. 16

[25] Feng, D., Haase-Schuetz, C., Rosenbaum, L., Hertlein, H., Glaeser, C., Timm, F., Wiesbeck, W., and Dietmayer, K. (2020). Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *arXiv:1902.07830 [cs]*. 1, 7, 14

[26] Fujiyoshi, H., Hirakawa, T., and Yamashita, T. (2019). Deep learning-based image recognition for autonomous driving. *IATSS Research*, 43(4):244–252. 3

[27] Girshick, R. (2015a). Fast R-CNN. *2015 IEEE International Conference on Computer Vision (ICCV)*. 3

[28] Girshick, R. (2015b). Fast r-cnn. *2015 IEEE International Conference on Computer Vision (ICCV)*. 14, 15, 41

[29] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 3, 14

[30] Grimes, D. M. and Jones, T. O. (1974). Automotive Radar: A Brief Review. *Proceedings of the IEEE*, 62(6):804–822. 23

[31] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 28, 41

[32] Hu, H.-N., Cai, Q.-Z., Wang, D., Lin, J., Sun, M., Krahenbuhl, P., Darrell, T., and Yu, F. (2019). Joint monocular 3d vehicle detection and tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5390–5399. 17

[33] Ji, Z. and Prokhorov, D. (2008). Radar-vision fusion for object classification. *Proceedings of the 11th International Conference on Information Fusion, FUSION 2008*, 2:265–271. 22

[34] Julier, S. J. and Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. In *Signal Processing, Sensor Fusion, and Target Recognition VI*, volume 3068, pages 182–193. International Society for Optics and Photonics. 20

[35] Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. 19

[36] Kang, K., Li, H., Xiao, T., Ouyang, W., Yan, J., Liu, X., and Wang, X. (2017). Object detection in videos with tubelet proposal networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 727–735. 16

[37] Kato, T., Ninomiya, Y., and Masaki, I. (2002). An obstacle detection method by fusion of radar and motion stereo. *IEEE transactions on intelligent transportation systems*, 3(3):182–188. 22

[38] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 76

[39] Kong, T., Sun, F., Liu, H., Jiang, Y., Li, L., and Shi, J. (2020). Foveabox: Beyound anchor-based object detection. *IEEE Transactions on Image Processing*, 29:7389–7398. 4

[40] Krebs, S., Duraisamy, B., and Flohr, F. (2017). A survey on leveraging deep neural networks for object tracking. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pages 411–418. ix, 5, 6

[41] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90. 3

[42] Ku, J., Mozifian, M., Lee, J., Harakeh, A., and Waslander, S. L. (2018). Joint 3d proposal generation and object detection from view aggregation. *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 7, 22

[43] Kundu, A., Li, Y., and Rehg, J. M. (2018). 3D-RCNN: Instance-Level 3D Object Reconstruction via Render-and-Compare. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3559–3568, Salt Lake City, UT, USA. IEEE. 15

[44] Levinson, J., Askeland, J., Becker, J., Dolson, J., Held, D., Kammel, S., Kolter, J. Z., Langer, D., Pink, O., Pratt, V., Sokolsky, M., Stanek, G., Stavens, D., Teichman, A., Werling, M., and Thrun, S. (2011). Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168. 1

[45] Li, B. (2017). 3D Fully Convolutional Network for Vehicle Detection in Point Cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1513–1518. IEEE. 15

[46] Li, B., Zhang, T., and Xia, T. (2016). Vehicle Detection from 3D Lidar Using Fully Convolutional Network. *arXiv:1608.07916 [cs]*. 15

[47] Liang, M., Yang, B., Chen, Y., Hu, R., and Urtasun, R. (2019). Multi-task multi-sensor fusion for 3d object detection. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 7

[48] Lin, T.-Y., Goyal, P., Girshick, R., He, K., and Dollár, P. (2018). Focal Loss for Dense Object Detection. *arXiv:1708.02002 [cs]*. 15, 41, 51

[49] Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. *Lecture Notes in Computer Science*, page 740–755. 28

[50] Liu, L., Ouyang, W., Wang, X., Fieguth, P., Chen, J., Liu, X., and Pietikäinen, M. (2019). Deep Learning for Generic Object Detection: A Survey. *arXiv:1809.02165 [cs]*. 3

[51] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., and Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In *European Conference on Computer Vision*. 3, 4, 15

[52] Liu, Z., Zheng, T., Xu, G., Yang, Z., Liu, H., and Cai, D. (2020). Training-time-friendly network for real-time object detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11685–11692. 4

[53] Lloyd, S. (1982). Least squares quantization in PCM. *IEEE transactions on information theory*, 28(2):129–137. 18

[54] Lowe, D. (1999). Object recognition from local scale-invariant features. In *Proceedings of the Seventh IEEE International Conference on Computer Vision*, pages 1150–1157 vol.2, Kerkyra, Greece. IEEE. 3

[55] Luo, R. C. and Kay, M. G. (1992). Data fusion and sensor integration: State-of-the-art 1990s. *Data fusion in robotics and machine intelligence*, pages 7–135. 20

[56] Meyer, G. P., Charland, J., Hegde, D., Laddha, A., and Vallespi-Gonzalez, C. (2019). Sensor fusion for joint 3d object detection and semantic segmentation. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 7

[57] Meyer, M. and Kuschk, G. (2019). Deep learning based 3d object detection for automotive radar and camera. In *2019 16th European Radar Conference (EuRAD)*, pages 133–136. IEEE. 22

[58] Milan, A., Rezatofighi, S. H., Dick, A., Reid, I., and Schindler, K. (2016). Online multi-target tracking using recurrent neural networks. *arXiv preprint arXiv:1604.03635*. 5

[59] Mousavian, A., Anguelov, D., Flynn, J., and Kosecka, J. (2017). 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7074–7082. 15, 51

[60] Nabati, R. and Qi, H. (2019). RRPN: Radar region proposal network for object detection in autonomous vehicles. *2019 IEEE International Conference on Image Processing (ICIP)*. 7

[61] Nabati, R. and Qi, H. (2020). CenterFusion: Center-based Radar and Camera Fusion for 3D Object Detection. 70, 76

[62] Newell, A., Yang, K., and Deng, J. (2016). Stacked Hourglass Networks for Human Pose Estimation. *arXiv:1603.06937 [cs]*. 58, 59

[63] Nobis, F., Geisslinger, M., Weber, M., Betz, J., and Lienkamp, M. (2019). A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection. In *2019 Sensor Data Fusion: Trends, Solutions, Applications (SDF)*, pages 1–7. 22, 42

[64] Oh, S.-I. and Kang, H.-B. (2017). Object Detection and Classification by Decision-Level Fusion for Intelligent Vehicle Systems. *Sensors (Basel, Switzerland)*, 17(1). 21

[65] Ondruska, P. and Posner, I. (2016). Deep tracking: Seeing beyond seeing using recurrent neural networks. *arXiv preprint arXiv:1602.00991*. 5

[66] Premebida, C., Ludwig, O., and Nunes, U. (2009). Lidar and vision-based pedestrian detection system. *Journal of Field Robotics*, 26(9):696–711. 21

[67] Provan, G. M. (1992). The validity of Dempster-Shafer belief functions. *International Journal of Approximate Reasoning*, 6(3):389–399. 20

[68] Qi, C. R., Liu, W., Wu, C., Su, H., and Guibas, L. J. (2018). Frustum PointNets for 3D Object Detection from RGB-D Data. *arXiv:1711.08488 [cs]*. 21

[69] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108. 11

[70] Rangesh, A. and Trivedi, M. M. (2019). No blind spots: Full-surround multi-object tracking for autonomous vehicles using cameras and lidars. *IEEE Transactions on Intelligent Vehicles*, 4(4):588–599. 5

[71] Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788. IEEE. 4, 15

[72] Ren, J., Chen, X., Liu, J., Sun, W., Pang, J., Yan, Q., Tai, Y.-W., and Xu, L. (2017). Accurate single stage detector using recurrent rolling convolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5420–5428. 16

[73] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Neural Information Processing Systems (NIPS)*. 3, 15, 16, 25, 38, 40, 41

[74] Roddick, T., Kendall, A., and Cipolla, R. (2018). Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*. 59, 60

[75] Schlosser, J., Chow, C. K., and Kira, Z. (2016). Fusing LIDAR and images for pedestrian detection using convolutional neural networks. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2198–2205. 21

[76] Schulter, S., Vernaza, P., Choi, W., and Chandraker, M. (2017). Deep network flow for multi-object tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6951–6960. 16

[77] Shafer, G. (1976). *A Mathematical Theory of Evidence*, volume 42. Princeton university press. 20

[78] Shi, S., Wang, X., and Li, H. (2019). PointRCNN: 3D Object Proposal Generation and Detection from Point Cloud. *arXiv:1812.04244 [cs]*. 15

[79] Simonelli, A., Bulò, S. R. R., Porzi, L., López-Antequera, M., and Kontschieder, P. (2019). Disentangling Monocular 3D Object Detection. *arXiv:1905.12365 [cs]*. 59, 60, 62

[80] Song, S., Xiang, Z., and Liu, J. (2015). Object tracking with 3d lidar via multi-task sparse learning. In *2015 IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 2603–2608. IEEE. 7

[81] Soviany, P. and Ionescu, R. T. (2018). Optimizing the trade-off between single-stage and two-stage object detectors using image difficulty prediction. *arXiv preprint arXiv:1803.08707*. 4

[82] Tao, R., Gavves, E., and Smeulders, A. W. (2016). Siamese instance search for tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1420–1429. 5

[83] Uijlings, J. R. R., van de Sande, K. E. A., Gevers, T., and Smeulders, A. W. M. (2013). Selective Search for Object Recognition. *International Journal of Computer Vision*, 104(2):154–171. 14, 28

[84] Varior, R. R., Shuai, B., Lu, J., Xu, D., and Wang, G. (2016). A siamese long short-term memory architecture for human re-identification. In *European Conference on Computer Vision*, pages 135–153. Springer. 5

[85] Wang, J., Lan, S., Gao, M., and Davis, L. S. (2020). Infofocus: 3d object detection for autonomous driving with dynamic information modeling. *arXiv preprint arXiv:2007.08556*. 59, 60, 62

[86] Wang, L., Ouyang, W., Wang, X., and Lu, H. (2015a). Visual tracking with fully convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3119–3127. 5

[87] Wang, N., Li, S., Gupta, A., and Yeung, D.-Y. (2015b). Transferring rich feature hierarchies for robust visual tracking. *arXiv preprint arXiv:1501.04587*. 5

[88] Wang, N. and Yeung, D.-Y. (2013). Learning a deep compact image representation for visual tracking. In *Advances in Neural Information Processing Systems*, pages 809–817. 5

[89] Wang, Z., Zhan, W., and Tomizuka, M. (2018). Fusing Bird View LIDAR Point Cloud and Front View Camera Image for Deep Object Detection. *arXiv:1711.06703 [cs]*. 21

[90] Welch, G. and Bishop, G. (1995). *An Introduction to the Kalman Filter*. Citeseer. 20

[91] Weng, X. and Kitani, K. (2019). A baseline for 3d multi-object tracking. *arXiv preprint arXiv:1907.03961*. 70, 77, 78

[92] Weng, X., Wang, J., Held, D., and Kitani, K. (2020a). 3d multi-object tracking: A baseline and new evaluation metrics. *arXiv preprint arXiv:1907.03961*. 16, 17

[93] Weng, X., Wang, Y., Man, Y., and Kitani, K. M. (2020b). Gnn3dmot: Graph neural network for 3d multi-object tracking with 2d-3d multi-feature learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6499–6508. 16

[94] Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE. 16

[95] Xie, S., Girshick, R., Dollar, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 28

[96] Xin, Y., Wang, G., Mao, M., Feng, Y., Dang, Q., Ma, Y., Ding, E., and Han, S. (2021). Pafnet: An efficient anchor-free object detector guidance. *arXiv preprint arXiv:2104.13534*. 4

[97] Xu, D., Anguelov, D., and Jain, A. (2018). PointFusion: Deep Sensor Fusion for 3D Bounding Box Estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 244–253. 21

[98] Xu, J., Cao, Y., Zhang, Z., and Hu, H. (2019). Spatial-temporal relation networks for multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3988–3998. 16

[99] Yan, Y., Mao, Y., and Li, B. (2018). SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 18(10):3337. 15

[100] Yang, B., Guo, R., Liang, M., Casas, S., and Urtasun, R. (2020). RadarNet: Exploiting Radar for Robust Perception of Dynamic Objects. *arXiv:2007.14366 [cs]*. 22

[101] Yang, B., Luo, W., and Urtasun, R. (2019). PIXOR: Real-time 3D Object Detection from Point Clouds. *arXiv:1902.06326 [cs]*. 15

[102] Yang, F., Choi, W., and Lin, Y. (2016). Exploit all the layers: Fast and accurate cnn object detector with scale dependent pooling and cascaded rejection classifiers. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2129–2137. 16

[103] Yu, F., Wang, D., Shelhamer, E., and Darrell, T. (2018). Deep Layer Aggregation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2403–2412, Salt Lake City, UT. IEEE. 52, 58, 61

[104] Zhang, L., Li, Y., and Nevatia, R. (2008). Global data association for multi-object tracking using network flows. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE. 16

[105] Zhang, R., Candra, S. A., Vetter, K., and Zakhor, A. (2015). Sensor fusion for semantic segmentation of urban scenes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1850–1857. 7

[106] Zhang, W., Zhou, H., Sun, S., Wang, Z., Shi, J., and Loy, C. C. (2019). Robust multi-modality multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2365–2374. 16

[107] Zhang, Z., Cheng, D., Zhu, X., Lin, S., and Dai, J. (2018). Integrated object detection and tracking with tracklet-conditioned detection. *arXiv preprint arXiv:1811.11167*. 17

[108] Zhou, X., Koltun, V., and Krähenbühl, P. (2020). Tracking objects as points. In *European Conference on Computer Vision*, pages 474–490. Springer. 7, 17, 70, 71, 74, 77, 78

[109] Zhou, X., Wang, D., and Krähenbühl, P. (2019). Objects as points. *arXiv preprint arXiv:1904.07850*. 15, 48, 51, 52, 58, 59, 60, 62, 70

[110] Zhou, Y. and Tuzel, O. (2017). VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *arXiv:1711.06396 [cs]*. 15

[111] Zhu, J., Yang, H., Liu, N., Kim, M., Zhang, W., and Yang, M.-H. (2018). Online multi-object tracking with dual matching attention networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 366–382. 7, 16, 70

[112] Zhu, X., Wang, Y., Dai, J., Yuan, L., and Wei, Y. (2017). Flow-guided feature aggregation for video object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 408–417. 17

[113] Zou, Z., Shi, Z., Guo, Y., and Ye, J. (2019). Object Detection in 20 Years: A Survey. *arXiv:1905.05055 [cs]*. 3

Vita

Ramin Nabati received his bachelor's degree in Electrical Engineering - Telecommunication from the Urmia University, Urmia, Iran in 2011 and his master's degree in Electrical Engineering - Communication Systems from the Isfahan University of Technology (IUT), Isfahan, Iran in 2015. Shortly after graduating from IUT, he joined the Department of Electrical Engineering and Computer Science (EECS) at the University of Tennessee, Knoxville to pursue a PhD degree in Computer Engineering. He received the EECS Department Excellence Fellowship in 2015 and 2016, the College of Engineering Extraordinary Professional Promise award in 2020, and the Tennessee's Top 100 fellowship in 2020.

Ramin's research interests include Computer Vision, Machine Learning and Deep Learning, with a focus on Sensor Fusion for Object Detection and Object Tracking in Autonomous Vehicles. In 2017 he joined the University of Tennessee EcoCAR team as the Advanced Driver Assistance Systems (ADAS) team lead in the EcoCAR3 competition, an Advanced Vehicle Technology Competition (AVTC) sponsored by the U.S. Department of Energy, General Motors and MathWorks. He is currently the Connected and Automated Vehicles (CAV) team lead for team Tennessee in the EcoCAR Mobility Challenge competition.