Efficient Multi-Sensor Fusion for 3D Perception

by

Kevin Shao

S.B. in Computer Science and Engineering
Massachusetts Institute of Technology (2023)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2023

Authored by: Kevin Shao
Department of Electrical Engineering and Computer Science
August 11, 2023

Certified by: Song Han
Associate Professor
Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Efficient Multi-Sensor Fusion for 3D Perception

by

Kevin Shao

Submitted to the Department of Electrical Engineering and Computer Science
on August 11, 2023, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

As a critical component to realizing widespread autonomous driving, 3D perception systems have come to be heavily studied in the community. However, many solutions are solely focused on merely achieving the highest accuracy – overlooking other practical considerations such as speed and cost. In this thesis, I develop two multi-sensor fusion models for 3D Perception: **BEVFusion**, a camera-LiDAR fusion model, and **BEVFusion-R**, a camera-radar fusion model. BEVFusion seeks to balance accuracy and speed. By fusing features from each input modality in the shared bird's eye view space, it captures both semantic and geometric information from each input. Its simple design allows it to achieve both state-of-the-art accuracy and a 24% speedup over competing works. BEVFusion-R further incorporates cost and hardware deployment into the design consideration. By carefully designing the entire model with both performance and acceleration, BEVFusion-R achieves a 2.1% NDS improvement on nuScenes over the previous state-of-the-art with a 4.5× measured speedup. Additionally, it is capable of real-time latency on edge GPUs. The code will be publicly released at https://github.com/mit-han-lab/bevfusion

Thesis Supervisor: Song Han
Title: Associate Professor

Acknowledgments

I'd like to thank Professor Song Han, Zhijian Liu, and Haotian Tang, for patiently guiding me throughout my research, beginning as a clueless undergrad.

I'm also grateful to my family and friends, both before and during MIT.

Contents

1 Introduction **11**

1.1 Background 12

1.1.1 3D Perception 12

1.1.2 Efficient Perception 15

1.2 Problem Statement 17

1.3 Thesis Outline 21

1.3.1 Camera-LiDAR Fusion: BEVFusion 21

1.3.2 Camera-Radar Fusion: BEVFusion-R 22

2 Efficient Camera-LiDAR Fusion **24**

2.1 Introduction 24

2.2 Motivation 25

2.3 Method 27

2.3.1 Framework Overview 27

2.3.2 The View Transformation 28

2.4 Experiments 29

2.4.1 Main Results 30

2.4.2 Robustness 31

2.5 Conclusion 31

3 Efficient Camera-Radar Fusion **33**

3.1 Introduction 33

3.2 Radar 35

3.2.1 Object Coverage 37

3.2.2 Velocity Estimation 37

3.3 Method 38

3.3.1 Image Encoder 39

3.3.2 Radar Encoder 39

3.3.3 Radar-Guided View Transformer 40

3.3.4 BEV Encoder 41

3.3.5 Deployment 41

3.4 Experiments 42

3.4.1 Main Results 42

3.4.2 Ablation Studies 43

3.4.3 Latency and Real Time Deployment 45

3.5 Conclusion 46

4 Conclusion **47**

4.1 Future Work 48

List of Figures

1-1 3D object detection - localizing relevant objects in space and determining key properties - is an important first step to any autonomous driving stack. (©nuro.ai) 12

1-2 Characteristics of a practical 3D detector. Widespread commercial autonomous driving would not be possible without each of these met. 17

1-3 State-of-the-art object detection algorithms have continuously advanced performance on the Nuscenes dataset, posting ever higher NDS (Nuscenes detection score). However, their computational cost leaves much to be desired. Even benchmarked with desktop GPUs, many models fall short of the real-time threshold of 20 Hz. 19

1-4 Most performance benchmarks are carried out with powerful server GPUs, like the NVIDIA RTX 3090 or the NVIDIA A6000. In contrast, realistic drive systems will employ edge GPUs, such as the NVIDIA Jetson AGX Orin. 20

1-5 Of the sensors deployed on the nuScenes [5] data collection vehicle, the spinning Velodyne LiDAR was by far the costliest. Costs estimated from commercial price at time of writing. 21

2-1 An *image-centric* paradigm, such as [43], projects point clouds into the camera view, but loses geometric information. 25
2-2 A *point-centric* paradigm projects pixel features (in this case semantic segmentation labels) onto the points. However, many pixels are not matched, resulting in semantic loss. Image credit: [73] 26
2-3 In the BEVFusion paradigm, input features are extracted independently, then converted into the shared bird's eye view (BEV) space via view transformers. There, the features can be fused and further processed by a 2D BEV encoder. Finally, this paradigm supports different tasks with task-specific heads. 28

3-1 While outperforming competing works, our BEVFusion-R is optimization-friendly, allowing it to enjoy significantly more hardware acceleration. See Section 3.4 for more details. 34
3-2 Radar returns are far sparser than LiDAR point clouds. 36
3-3 The velocity from radar exhibits a strong linear correlation with the actual velocity. 39
3-4 **BEVFusion-R** is a highly efficient multi-modal 3D perception model that leverages information from radar and camera sensors within the shared bird's-eye-view (BEV) space. In contrast to BEVFusion [49], our approach incorporates a radar-guided view transformer, which utilizes precise depth information from 3D radar to explicitly enhance camera-to-BEV projection. 40
3-5 Ablation study on the number of radar frames included. As the number of radar points increases, both object localization (translation error) and velocity estimation (velocity error) improve. 45

3-6 Acceleration of our model via TensorRT. Despite having similar PyTorch latency as CRN[30], our model is entirely hardware friendly, whereas the red sections in CRN are not. As such, we observe a much more dramatic speedup. 46

List of Tables

2.1 Accuracy and latency results compiled from [57, 55, 49]. Despite its simplicity, BEVFusion surpasses all other methods while remaining remarkably efficient (*: Re-implemented in [49]) 30

2.2 Robustness study under varying weather conditions, from [109]. While nighttime conditions are particularly challenging for vision-dependent models, BEVFusion displays robustness. 32

3.1 Analysis of the depth distribution of various modalities. Despite being much sparser overall, radar points are relatively denser at long range, a beneficial characteristic to complement camera-based detection. . . 37

3.2 Object coverage at various distances. Despite its sparsity, radar data provides superior object coverage compared to sparse LiDAR data and is even comparable to dense LiDAR data at long range. 38

3.3 Results for single-frame camera-radar fusion detectors on the nuScenes validation set. BEVFusion-R achieves a state-of-the-art nuScenes detection score (NDS) while maintaining a 92 FPS speed, which is 4.5× faster than the best available baseline at the time of writing. . 43

3.4 Ablation study on the choice of encoder for the point cloud. VN denotes VoxelNet. PP denotes PointPillars. Despite worsening latency in all cases, using VoxelNet, a 3D based encoder, will improve performance in the 32-beam and 1-beam LiDAR settings. However, for radar inputs, it performs no better than PointPillars. 44

3.5 Ablation study on the View Transformer. Not only does explicit depth supervision enable stronger performance, but using input from other modalities further improves the accuracy. The final setting is an oracle setting: not achievable in practice, but rather serving as an upper bound for optimal depth estimation. 44

Chapter 1

Introduction

The National Highway Traffic Safety Administration (NHTSA) estimates that up to 94% of serious motor accidents involve “driver error" as the critical reason [77]. Fully autonomous vehicles come with the promise of significantly improved safety by removing the potential for human error. However, they are currently seldom seen on the roads, aside from experimental trails or extremely limited use-cases. Despite the extensive attention autonomous vehicles have received in both academic and commercial study, the reality is that they have yet to see widespread practical use. There are several bottlenecks to such widespread use, both technical and social. However, one major bottleneck is the performance of the **perception** algorithm, which feeds into the rest of the typical self-driving software stack, such as state estimation, path planning, and controls algorithms. Responsible for interpreting raw sensor inputs from the outside world, failures in this algorithm could lead to catastrophic collisions.

Development of an effective 3D perception system is an involved task, but doing so is fundamental to autonomous driving. After all, we have little hope of avoiding a

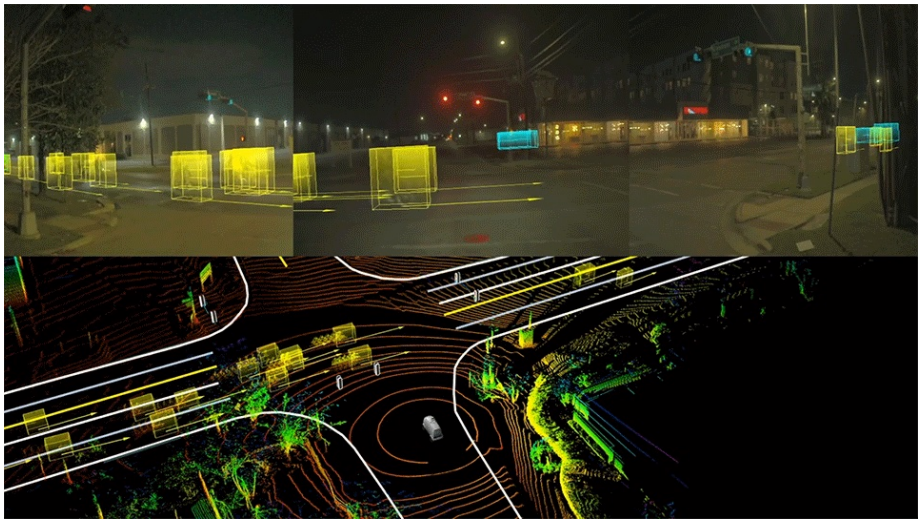

Figure 1-1: 3D object detection - localizing relevant objects in space and determining key properties - is an important first step to any autonomous driving stack. (©nuro.ai)

road hazard if we don't perceive it in time, or worse still, we never perceive it at all.

1.1 Background

1.1.1 3D Perception

Given sensory data, the goal of 3D perception is to localize objects and parse scene layouts. The research community has largely arrived at a handful of key tasks that are critical to autonomous driving, such as object detection [57], object tracking [78], map segmentation [36], and panoptic segmentation [1]. However, 3D object detection remains a particularly fundamental task, and forms the focus of this thesis.

Sensors. Most autonomous driving systems and datasets use camera, LiDAR, and radar sensors, but each sensor's strengths and weaknesses vary. Cameras exhibit poor performance in nighttime and in poor weather conditions. Moreover, localizing

objects in 3D space is difficult; there is only a 2D image to work from. However, cameras are not only cheap, but their high resolution yields rich semantic information. LiDAR sensors, while robust to nighttime, similarly suffer reduced performance in adverse weather, a significant challenge for robust safety [28]. However, its data, returned as a point cloud, comes with very precise depth, making localization quite straightforward. Finally, radar sensors mitigate the weather issue by utilizing a lower frequency. They are also an order of magnitude cheaper than LiDAR sensors and even offer accurate velocity estimation; however, the resolution is quite poor, making it challenging to localize or even detect objects altogether.

Data. As machine learning models are notoriously data hungry, curating large, multi-modal datasets has been a critical component to recent advancement. The first real-world autonomous driving data curation work proposed the KITTI[16, 14] dataset, containing 15000 LiDAR scans, 15000 images, and 200000 annotated objects. The paradigm used – equipping a vehicle with LiDAR and camera sensors and capturing data from roads – set a precedent for subsequent large datasets.

Following KITTI, modern datasets have made several improvements. First, several large-scale datasets, such as the NuScenes [5], Waymo [78], and ONCE [56] contain far more data than KITTI. For example, nuScenes has 26× more LiDAR scans and 93× more images, Waymo has 60× more object annotations, and ONCE surpasses 1 million LiDAR scans. Next, KITTI was captured in sunny, daytime conditions in Karlsruhe, Germany. As data diversity is critical for robust models, modern datasets [7, 5, 56, 91, 78] are all captured in a variety of conditions. Finally, recent datasets also include additional data beyond merely LiDAR scans, images, and object annotations. For example, [5] includes radar data, [7] provides HD maps for detailed road information, and [10] provides thermal images.

Models. Historically, LiDAR-based models and camera-based models have adopted largely different architectures. LiDAR-based object detectors principally sought to handle the sparse and irregular nature of the input point clouds, whereas image-based detectors must solve an ill-posed depth estimation problem, which renders object localization very difficult.

LiDAR-based models may choose to represent 3D points in a number of ways. Originally, PointNet [66, 67] proposed point-level feature encoding while respecting the permutation invariance. Point based detectors include PointRCNN [76] and PointFormer [62]. Such point based models require downsampling and context operations, such as ball query or furthest point sampling [57]. While the backbone network is relatively efficient, these auxiliary operations are often expensive and prohibitive for real-time computation.

Next, 3D convolution methods extend the common image processing paradigm to three dimensions. Since outdoor point clouds are very sparse, directly using 3D convolutions on a dense occupancy map, such as in [58, 50], is computationally intractable. Instead, occupied voxels are represented and operated on in a sparse manner [107]. Since the convolution operation repeatedly expands the receptive field, the submanifold convolution [18] is widely used to maintain the sparsity. Such a paradigm is used in several detectors like [96, 108, 101].

Despite leveraging the sparsity of point clouds, the computational cost of sparse convolutional methods are still relatively expensive. In order to avoid any computation in three dimensions, PointPillars [34] extended their voxels infinitely in the Z-axis, thus becoming pillars. After extracting point features via PointNet [66] within each pillar, the features are scattered back to the 2D plane, allowing them to be processed with traditional 2D convolutions. Further works leveraging this pillar representation include [74, 86].

Besides, there are also hybrid methods, such as point-voxel based methods [50, 80, 75]. Such processing techniques seek to leverage the benefits from each representation. Point-voxel based detectors benefit from the fine-grained information from the points and efficient computation from the voxels.

There are also less common approaches toward handling sparse 3D point clouds. For example, OctNet [71] uses recursive octrees to allocate more resources to dense regions of space. They propose to directly perform the 3D convolution on the octree-based data structure. Also, range-based approaches such as LaserNet [59] use a 2D representation while imbuing each pixel with depth information.

In contrast, camera-based approaches, whether monocular (single image), stereo (pair of images), or multi-view (several cameras), must contend with the challenge of ambiguous three-dimensional localization, a challenge absent from standard 2D object detection. Two stage detectors extend 2D detection methods, such as [17, 70, 69], and additionally regress 3D characteristics, such as depth, to lift the predicted bounding box into 3D space. One stage detectors either use anchor-based [33, 4] or center-based [106, 84] heads to directly predict 3D bounding boxes. Furthermore, pseudo-LiDAR methods [85, 65, 26, 25] directly transform the image pixels or features into 3D space, effectively generating a point cloud from the image.

1.1.2 Efficient Perception

The importance of 3D perception tasks is not limited to autonomous vehicles. In augmented reality, object detection and scene understanding it crucial to allowing for smooth interaction between the physical and virtual worlds. Similarly, robotic systems often require an understanding of their surroundings to carry out the desired task.

Regardless of the application, successful perception systems must excel in both accuracy and efficiency. The efficiency requirement is especially critical in real-world deployment since systems will be run on resource constrained hardware and subject to hard latency constraints. For example, self-driving cars and robotic agents must carry their hardware onboard. Even more restrictive, the hardware powering augmented reality experiences must be confined to a small headset. All the while, the systems must be able to process incoming data in real-time.

To meet these resource-constrained and latency sensitive demands, several lines of research have studied model compression, including pruning, quantization, distillation, and neural architecture search. Pruning [19, 20, 54, 13] reduces inference complexity by discarding portions of the network that have minimal impact. This can take several forms, from fine-grained weight level pruning to coarse-grained channel level pruning to activation pruning.

Quantization [82, 83, 90, 22, 60] leverages the idea that not each bit is equally important. Instead of performing computations with 32-bit floating point numbers, the arithmetic in the neural network can be performed with 16- or even 8-bit floats. At an extreme, binary neural networks [103] – those that only use 0's and 1's in their computation – have even been proposed to reduce the computational cost.

It is well known that employing several separately trained models, then averaging their predictions, will boost the overall performance in a process known as model ensembling [12]. Of course, the inference cost will scale linearly with the number of models used. As such, model distillation [23] has been proposed to compress the aggregate knowledge in an ensemble of models into a single model, thus eliminating the overhead.

Many of the above model compression techniques involve making several design choices, effectively requiring the search of a large design space. As such, Neural

Architecture Search (NAS) methods [24, 27] are crucial for alleviating both the engineering and computational overhead of searching a large space. Early methods include using evolutionary search [68, 80] or reinforcement learning [110]. These methods nonetheless still require training several models, which is an expensive task. Following, efforts to lift this requirement include differentiable architecture search [44] and one-shot [6] or even zero-shot [41] NAS.

1.2 Problem Statement

The focus of this work is to holistically develop a practical 3D perception model. Such a model ought to leverage multiple complementary sensor modalities, a technique that has been shown to significantly improve performance [3, 98, 49], in a way that maximally utilizes the information from each. Furthermore, such a model ought to be designed with three primary characteristics in mind: robust accuracy, efficiency, and cost.

Figure 1-2: Characteristics of a practical 3D detector. Widespread commercial autonomous driving would not be possible without each of these met.

Robust Accuracy. The importance of accuracy in a driving perception system needs little motivation. The primary metric used to evaluate detectors in literature is mean average precision (mAP), extended from 2D object detection. When computing recall curves, datasets use varying methods of detecting matches, such as IoU based (either 3D or BEV), center distance based, or even heading error weighted average precision [78]. Recently, [5] also noted that attribute and velocity estimation are completely ignored by mAP, and proposed their own nuScenes Detection Score (NDS) for more holistic evaluation.

However, regardless of the metric the robustness of a 3D detector is often poorly captured. It is well known that camera-based detectors perform poorly in nighttime conditions and LiDAR-based methods suffer during poor weather conditions [109]. This can create hazardous situations where the perception algorithm fails in safety critical scenarios. Indeed, using multi-sensor fusion is a promising direction for mitigating these challenges, but however it is accomplished, a practical perception model should be robust to a variety of conditions.

Efficiency. Due to the hardware and latency constraints unique to autonomous driving perception, computational efficiency is a crucial component to its success. In particular, autonomous driving perception algorithms ought to run in 50 ms, or 20 Hz, in order to keep pace with the various inputs [5]. Despite achieving impressive performance, the top perception algorithms struggle to meet that goal, especially those employing sparse 3D computations. In benchmarking tests shown in Figure 1-3, of the state-of-the-art models, only PointPillars actually achieves real-time latency at the cost of significant accuracy.

Moreover, most models are benchmarked using powerful desktop GPUs, as opposed to edge GPUs which would realistically be used in drive systems. Figure 1-4 elucidates

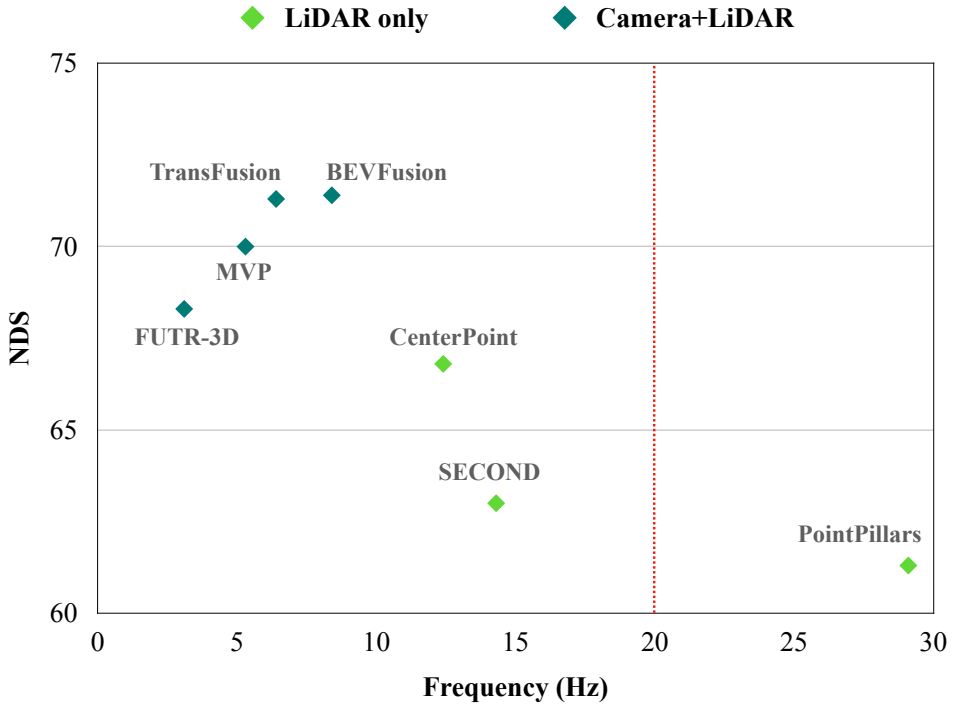

Figure 1-3: State-of-the-art object detection algorithms have continuously advanced performance on the Nuscenes dataset, posting ever higher NDS (Nuscenes detection score). However, their computational cost leaves much to be desired. Even benchmarked with desktop GPUs, many models fall short of the real-time threshold of 20 Hz.

the main differences between the two: edge GPUs trade significant performance for size, power draw, and therefore cost. As such, they perform as poorly as several times slower than their desktop counterparts. A deployment-ready 3D perception algorithm ought to run at real-time speeds on edge GPUs, a requirement not met by state-of-the-art models.

Hardware Cost. Often times, cost becomes a key bottleneck when attempting to bridge the gap between academic research and widespread commercial adoption. In

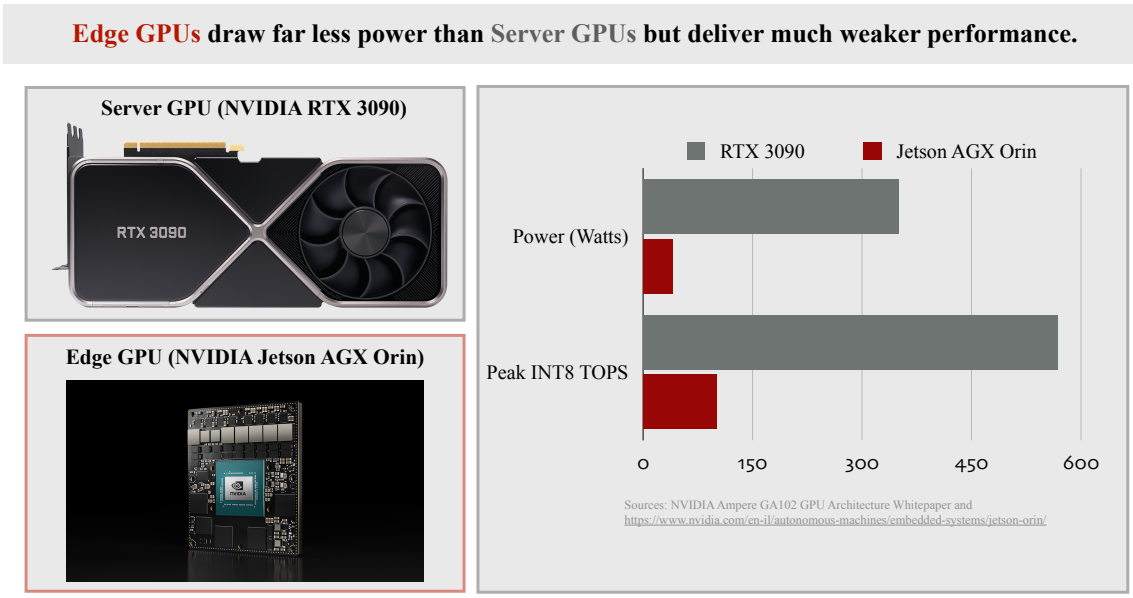

Figure 1-4: Most performance benchmarks are carried out with powerful server GPUs, like the NVIDIA RTX 3090 or the NVIDIA A6000. In contrast, realistic drive systems will employ edge GPUs, such as the NVIDIA Jetson AGX Orin.

designing a full perception system, each component's cost must be considered. The cost of the deployed sensors can constitute the majority of the expense. In particular, despite providing accurate and useful data, LiDAR sensors are prohibitively costly. Recent research [35] in both solid-state and spinning LiDARs have significantly lowered the cost of manufacturing, but LiDARs fitting for autonomous driving remain an order of magnitude higher than radars and cameras. Since economic cost is a key consideration and hurdle for widespread commercial use, practical 3D object detectors should not use LiDAR.

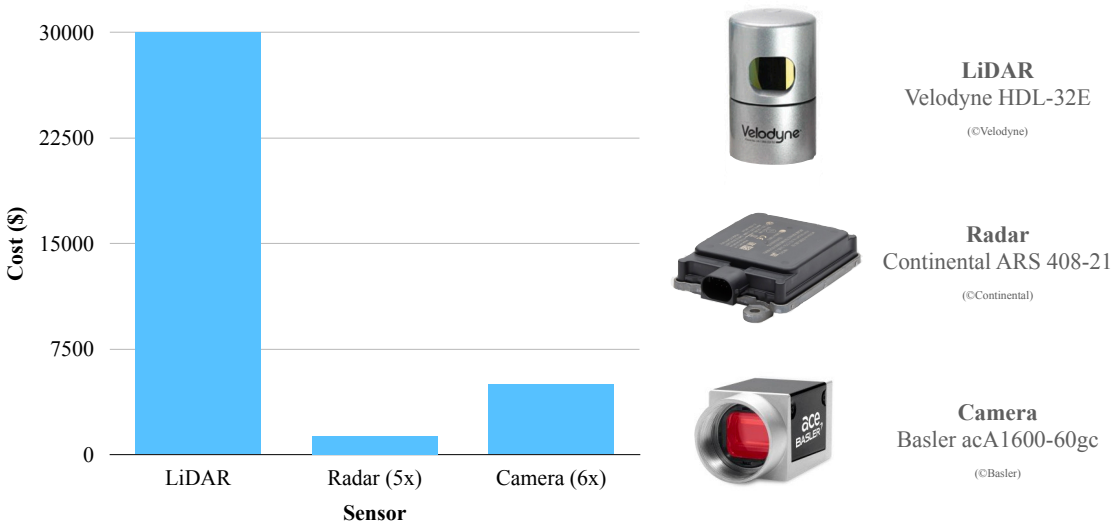

Figure 1-5: Of the sensors deployed on the nuScenes [5] data collection vehicle, the spinning Velodyne LiDAR was by far the costliest. Costs estimated from commercial price at time of writing.

1.3 Thesis Outline

1.3.1 Camera-LiDAR Fusion: BEVFusion

In Chapter 2, I develop a camera-LiDAR fusion model that solely focuses on accuracy and efficiency. **BEVFusion** performs sensor fusion in the Bird's Eye View space, allowing for straightforward adoption of sensor-specific encoders, efficient fusion, and strong robustness. I further present several experiments to verify its efficacy and flexibility.

In Section 1.1, I discussed the various input modalities in autonomous driving systems, as well as their strengths and weaknesses. As they exhibit complementary characteristics, an ideal fusion paradigm would allow the perception algorithm to fully exploit each modality's strengths and mitigate the weaknesses. However, such a fusion paradigm is nontrivial, since each modality captures information in fundamentally

different ways: images are captured in perspective view, whereas LiDAR and radar point clouds are in 3D view. Recent sensor fusion methods vary in how they handle this view discrepancy.

One natural idea is to project all 3D points to the 2D image plane via known calibration transforms. Then, this augmented 2D image could be processed by well-established 2D vision techniques like CNNs [32]. However, the valuable inherent geometric information from the point cloud is lost: not only can previously distant points can be projected near each other on the image, but previously close points can end up far away.

In the other direction, another idea is to match image features with the 3D points based on the same geometry, thus essentially augmenting the point cloud. Then, a LiDAR detector, such as VoxelNet [107] or PointPillars [34] can be employed. This approach, however, relies on the density of the point cloud. In regions of the image where there are no 3D points, the semantic information from the image is irreversibly lost.

Using this framework, which unifies the different modalities in the 2D bird's eye view (BEV) space, BEVFusion preserves the geometric information from 3D point clouds as well as the semantic information from images. Compared to the competing models at the time of writing, BEVFusion achieves state-of-the-art accuracy while achieving significant speedup.

1.3.2 Camera-Radar Fusion: BEVFusion-R

Finally, the third chapter builds **BEVFusion-R**, a fusion model that still targets accuracy and efficiency, but further considers cost efficiency and deployment readiness as well. Being designed with precisely the practical considerations from Section 1.2 in

mind. It is fast, cheap, and deployment-ready, all while sacrificing minimal accuracy and setting a new state-of-the-art for camera-radar fusion.

In order to build such a practical perception algorithm, accuracy must be balanced with cost, speed, and deployability. First, despite impressive recent advancements in LiDAR sensor manufacturing, the cost of a LiDAR sensor is still an order of magnitude higher than cameras or radars. As such, BEVFusion-R uses only camera-radar fusion.

Next, we design a novel **radar-guided view transformer** in order to further aid the multi-sensor fusion. Despite being a strong fusion paradigm, one weakness of BEVFusion is that the image-to-BEV projection depends on accurate depth estimation, which is a fundamentally difficult problem. By providing depth information from radar returns, we allow the model to more accurately perform the image-to-BEV projection, hence further reducing spatial misalignment.

Then, in order to ensure a fast and constant runtime, BEVFusion-R employs `TensorRT`[1] acceleration for hardware optimization. In order to do so, BEVFusion-R is designed from the beginning with hardware optimization in mind. Specifically, we not only avoid operators with irregular data access patterns, such as deformable convolutions, but also use `TensorRT` supported operations wherever possible.

However, despite targeting model simplicity, there are certain non-standard operators, such as BEV pooling, that are unavoidable. In order to ensure that BEVFusion-R is deployment ready, we design an engineering pipeline based on the ONNX [2] format. Upon exporting the model to an ONNX file, BEVFusion-R can be run on any `TensorRT` device with ease.

[1] https://github.com/NVIDIA/TensorRT

Chapter 2

Efficient Camera-LiDAR Fusion

2.1 Introduction

Autonomous driving systems are equipped with diverse sensors. For example, the Nuscenes dataset is captured with 1 LiDAR sensor, 5 long-range radar sensors, and 6 cameras [5]. Even more extensive, the setup for the Waymo Open Dataset has 5 LiDARs, 6 radars, and 29 cameras [78]. Such diversity, in theory provides much value to any perception task by encapsulating complementary signals: cameras capture rich semantic information, LiDARs provide accurate spatial information, while radars offer instant velocity estimation. However, in practice, fully leveraging this valuable information is far from straightforward. Therefore, multi-sensor fusion is of great importance for accurate and reliable perception [49]. In this section, I will develop BEVFusion, a camera-LiDAR fusion model. By projecting features from each modality into the shared bird's eye view, BEVFusion retains the semantic density from images while preserving geometric structure from the LiDAR points. BEVFusion is able to achieve state-of-the-art performance on the nuScenes object detection task,

while achieving a 24% speedup.

2.2 Motivation

Different features can exist in different views. For instance, camera features are in the perspective view, while LiDAR features are typically in the 3D/bird's-eye view [49]. Even for camera features, each one of them has a distinct viewing angle (i.e., front, back, left, right). This *view discrepancy* makes the feature fusion difficult since the same element in different feature tensors might correspond to completely different spatial locations. Therefore, it is crucial to find a shared representation, such that (1) all sensor features can be easily converted to it without information loss, and (2) it is suitable for different types of tasks.

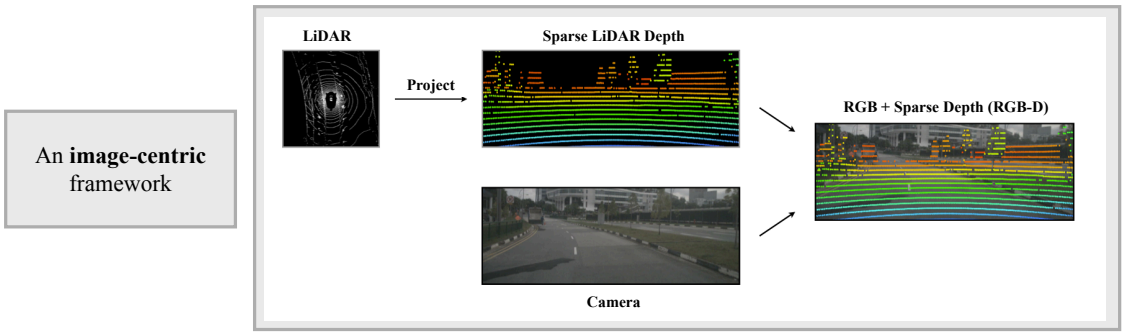

Figure 2-1: An *image-centric* paradigm, such as [43], projects point clouds into the camera view, but loses geometric information.

Camera-Centric Fusion. One choice for a shared representation is to use the perspective view that images are in. Doing so would involve projecting any LiDAR or radar point clouds to the camera plane, imbuing it with the depth, and directly considering it as an augmented 2D input [43]. However, this approach *loses geometric*

information. Points that are far away in 3D space may be projected very close to each other, despite having little geometric relationship. As such, localization specific tasks are at a significant disadvantage in this paradigm. The camera-centric paradigm is visualized in Figure 2-1

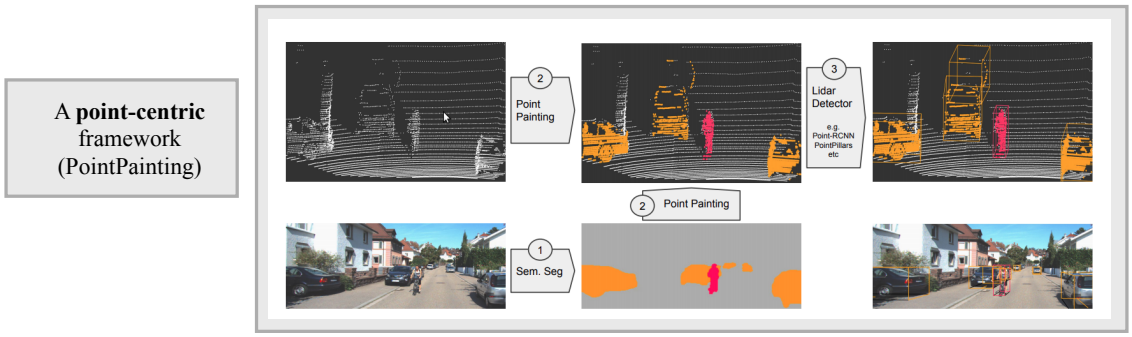

Figure 2-2: A *point-centric* paradigm projects pixel features (in this case semantic segmentation labels) onto the points. However, many pixels are not matched, resulting in semantic loss. Image credit: [73]

LiDAR-Centric Fusion. Many sensor fusion works [73, 38, 100, 81] instead project camera pixels or features onto their corresponding LiDAR or radar points. In contrast with camera plane projection, this approach *loses semantic information*. Since cameras and point clouds inherently have different densities, less than 5% of camera features will be matched to a point [49], even in the case of the densest LiDAR sensor. As such, tasks that rely heavily on semantic information suffer performance drawbacks. The point-centric paradigm is visualized in Figure 2-2.

Object-Centric Fusion. Aside from these geometric views, some recent works [8, 87, 3] propose to define object queries in 3D space. These object-centric paradigms, however, cannot generalize easily to geometric tasks like map segmentation.

Bird's Eye View. In contrast, the Bird's Eye View is a strong contender for a unified representation. Since most 3D perception tasks involve a BEV output space, there is no projection required after the sensor fusion. Moreover, it solves both aforementioned problems. Projecting point clouds to the BEV space simply involves flattening along the height dimension: no geometric distortion is introduced. Similarly, projecting images to the BEV space does not lose semantic information due to the ray projection and pillar pooling (described in the next section), which results in a dense feature map in the BEV space.

2.3 Method

BEVFusion focuses on *multi-sensor fusion* in a task-agnostic setting. As motivated by Section 2.2, this paradigm transforms each of the multi-modal features into a unified BEV representation, in contrast with point-centric fusion common at the time of writing.

2.3.1 Framework Overview

The overview of the framework is provided in Figure 2-3 [49]. Features are first extracted from various inputs via modality-specific encoders. For images, common choices include ResNet [21] and Swin-T [47], and for point clouds, typically variants of VoxelNet [107], PointPillars [34], or PointNet [66] are employed. Next, the modality-specific features are transformed to a unified BEV representation by a view transformer, which preserves both geometric and semantic information. There, features can either be simply concatenated or joined through convolution [49] or attention based fusers [30]. Finally, nearly any task-specific head can be used to support the desired

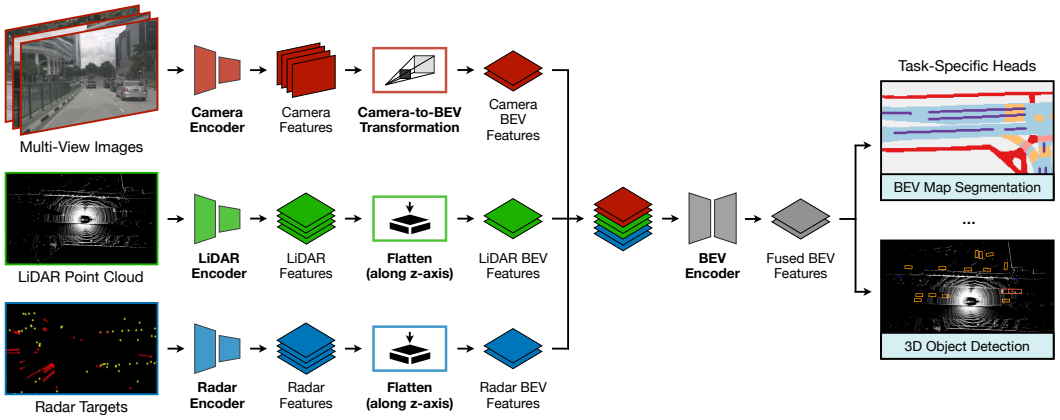

Figure 2-3: In the BEVFusion paradigm, input features are extracted independently, then converted into the shared bird's eye view (BEV) space via view transformers. There, the features can be fused and further processed by a 2D BEV encoder. Finally, this paradigm supports different tasks with task-specific heads.

task, allowing this framework to be task-agnostic.

2.3.2 The View Transformation

In contrast to point clouds, where transformation to the shared BEV space is straightforward, projecting camera features into the BEV view is difficult. Doing so requires performing depth estimation, a fundamentally ill-posed problem. Early works simply projected each camera feature to all possible depths [72]. Later, depth estimation modules were employed to estimate a depth *distribution*. In this Lift-Splat-Shoot (LSS) paradigm, features scattered are along the ray scaled according to the depth estimation confidence [65, 26].

Later, it was observed in [39] that this depth estimation is 1) not implicitly learned well and 2) crucial to 3D object detection. To show the first point, the authors replaced the depth estimation with a random vector. The accuracy only

dropped from 28.2% mAP to 24.5% mAP, implying that the depth estimation module provides limited value. However, when the depth estimation was replaced by the ground-truth depth, the model's accuracy jumped to 47.0% mAP, showcasing the potential power of accurate depth estimation. As such, BEVDepth proposed a directly supervised depth loss, along with a far stronger depth estimation module.

While BEVFusion can easily incorporate the BEVDepth paradigm, it does not provide a way for the depth estimation module to leverage the LiDAR or radar data for more accurate depth estimation. As such, one point of improvement explored in Chapter 3 is the development of a *radar-guided view transformer*, based on the motivating observation that accurate depth is essential to accurate detection.

2.4 Experiments

To empirically evaluate 3D object detectors, many existing works use the nuScenes [5] dataset, a large-scale outdoor dataset featuring 1000 scenes, 1.4 million images, 390,000 LiDAR sweeps, and 1.4M annotated object bounding boxes. One common performance metric for the object detection on the nuScenes dataset is mean average precision (mAP). It computes average precision (AP), as defined by the BEV center distance, over 4 distance thresholds and all ten classes, then reports the average. However, as mAP is only a metric of localization, the nuScenes detection score (NDS) is also widely used. The NDS weights mAP with other metrics, such as velocity, orientation, translation, and scale errors, providing a more holistic score. As such, we will select NDS as the primary reported metric throughout this thesis.

	Modality	NDS	mAP	AP (Car)	AP (Ped.)	Latency (ms)	MACs (G)
BEVDet [26]	C	39.7	47.7	-	-	529	-
PETR [45]	C	44.2	37.0	-	-	-	-
BEVDepth [39]	C	53.8	41.8	-	-	-	-
SOLOFusion [63]	C	58.2	48.3	-	-	-	-
PointPillars [34]*	L	63.2	54.7	83.4	78.7	34.4	65.5
CenterPoint [99]	L	67.3	60.3	-	-	69.8	85.0
Object DGCNN [88]	L	66.1	58.6	-	-	-	-
SECOND [96]	L	63.0	52.6	-	-	69.8	85.0
3D-CVF [102]	L+C	62.3	52.7	83.0	74.2	75	-
UVTR [37]	L+C	71.1	67.1	-	-	-	-
PointPainting [73]*	L+C	69.6	65.8	-	-	185.8	370.0
FusionPainting [94]	L+C	70.4	66.3	86.3	87.5	-	-
PointAugmenting [81]	L+C	71.0	66.8	87.5	87.9	234.4	408.5
MVP [100]	L+C	70.5	66.4	86.8	**89.1**	187.1	371.7
AutoAlign [9]	L+C	71.1	66.6	85.9	86.4	-	-
FUTR-3D [8]	L+C	68.0	64.2	86.3	82.6	321.4	1069.0
TransFusion [3]	L+C	71.3	67.9	87.1	88.4	156.6	485.8
BEVFusion [49]	L+C	**71.4**	**68.5**	**89.2**	88.0	119.2	253.2

Table 2.1: Accuracy and latency results compiled from [57, 55, 49]. Despite its simplicity, BEVFusion surpasses all other methods while remaining remarkably efficient (*: Re-implemented in [49])

2.4.1 Main Results

As shown in Table 2.1, BEVFusion outperforms all contemporary multi-sensor fusion works, and exhibits especially strong performance in localizing other cars. Moreover, BEVFusion is also highly efficient, running 23.9% faster than competing TransFusion. Indeed, its latency is near real time (119 ms latency, 8.4 FPS), albeit on a desktop GPU. By being both accurate and performant, BEVFusion is a prime candidate upon which to build a practical detector. In Chapter 3, I target a fully real-time model that must run on an edge GPU.

2.4.2 Robustness

Aside from overall accuracy, a 3D detector's robustness is also crucial to safe autonomous vehicle operation. The nuScenes dataset includes scenes in common adverse scenarios, which can lead to poor image quality or distribution shift. By splitting the scenes into Day, Night, Sunny, and Rainy scenes, the effect of each condition can be systematically analyzed.

In particular, nighttime conditions are challenging for camera-based models, as image quality becomes significantly degraded. In Table 2.2, the effect is very prominent with the camera-only detectors. However, BEVFusion is able to most effectively mitigate the accuracy drop from day to night, outperforming TransFusion by 2.0% NDS.

In rainy situations, the light beams from LiDAR sensors can be affected by atmospheric particles, leading to sensor noise and reduced performance in LiDAR-only models. For example, CenterPoint suffers a 3.7% mAP drop in rainy conditions. However, BEVFusion is able to mitigate that impact entirely, delivering on-par performance in the same conditions.

2.5 Conclusion

BEVFusion leverages a powerful and flexible paradigm that results from a simple choice of shared representation. It delivers strong results as an efficient and accurate camera-LiDAR fusion model. By projecting all inputs into the bird's eye view space, both geometric and semantic information is preserved. Moreover, this paradigm allows it to exhibit high robustness to adverse weather. However, LiDAR sensors are extremely expensive; as such, in Chapter 3, I develop BEVFusion-R, a camera-radar

	Modality	NDS				mAP			
		Day	Night	Sunny	Rainy	Day	Night	Sunny	Rainy
BEVDepth [39]	C	41.0	22.9	39.4	44.7	33.7	13.2	33.1	33.3
DETR-3D [87]	C	42.7	23.0	41.0	47.5	35.0	16.0	34.3	36.1
BEVFormer [40]	C	48.1	27.3	47.0	50.8	37.2	20.1	36.6	38.3
CenterPoint [101]	L	–	–	–	–	62.8	35.4	62.9	59.2
TransFusion [3]	C+L	71.0	44.7	70.5	71.9	67.3	39.8	67.0	67.5
BEVFusion [49]	C+L	71.5	46.7	71.3	72.2	68.5	43.9	68.4	69.5

Table 2.2: Robustness study under varying weather conditions, from [109]. While nighttime conditions are particularly challenging for vision-dependent models, BEVFusion displays robustness.

model that further incorporates cost into the practical considerations.

Chapter 3

Efficient Camera-Radar Fusion

3.1 Introduction

The task of 3D object detection has become increasingly important over the past few years, particularly in the context of autonomous driving applications. Large multi-modal datasets [15, 5, 78] have been instrumental in driving research and advancing accuracy. However, there has been relatively less emphasis on the practical deployment of these models, which requires accounting for multiple factors simultaneously. As discussed in Chapter 1, a practical detector should be both *accurate* and *fast*, while also being *cost-effective* and *deployable*.

Although LiDAR-based 3D perception [99, 53, 31] has consistently outperformed other modalities, camera-based approaches have also been extensively studied due to their cost-effectiveness [40, 45, 46, 63, 93, 52, 97]. However, camera-based detection suffers from poor distant object localization due to the ill-posed depth estimation problem. Moreover, cameras are sensitive to lighting conditions, and their performance degrades severely in nighttime conditions, which limits their usefulness in real-world

scenarios [30, 89]. Conversely, LiDAR points provide very accurate depth, but the sparsity of points at large distances can make object recognition very challenging. Therefore, research in sensor fusion techniques [95, 37, 92, 104, 98] has been motivated by the complementary features of these modalities.

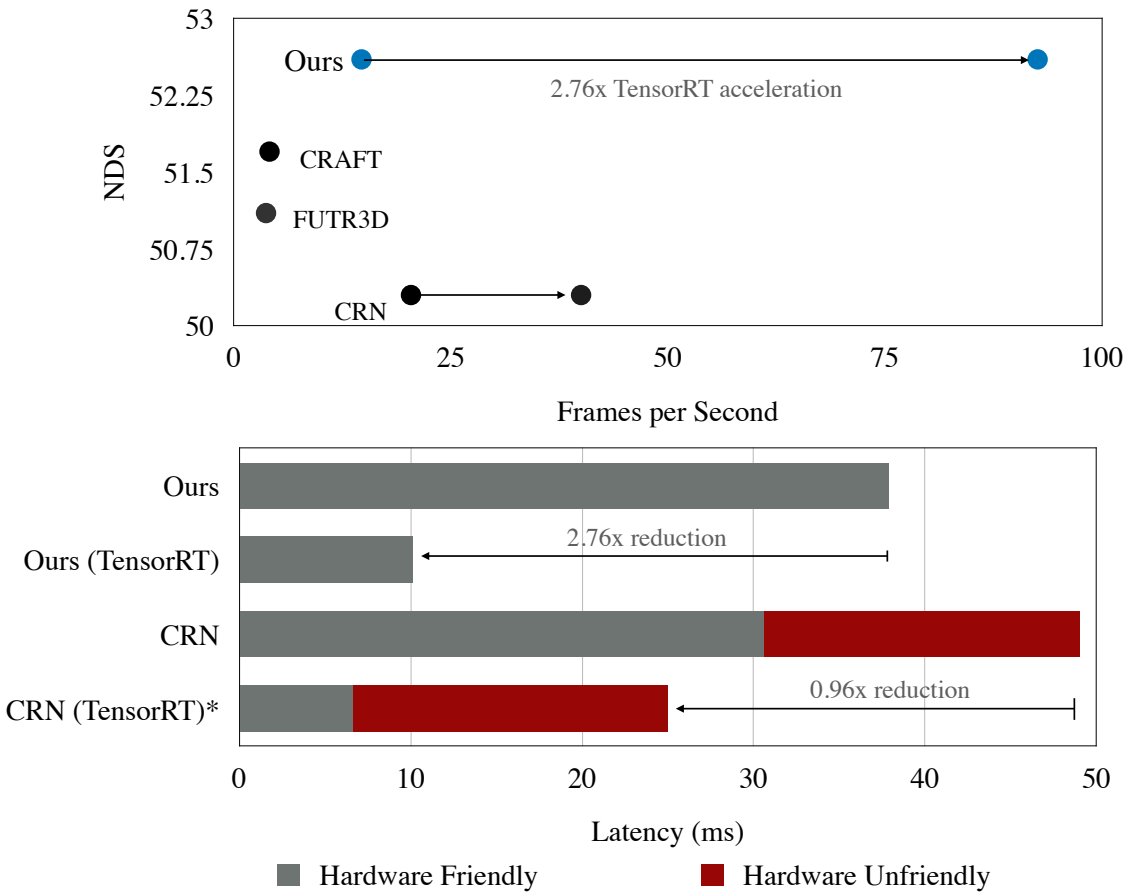

Figure 3-1: While outperforming competing works, our BEVFusion-R is optimization-friendly, allowing it to enjoy significantly more hardware acceleration. See Section 3.4 for more details.

Recently, radar has garnered notable interest as a cost-efficient alternative to LiDAR. Like LiDAR, radar data has characteristics that are largely complementary

to cameras. However, radar sensors also have advantageous characteristics as well. First, radar sensors are much cheaper than LiDAR. Despite the cost of LiDAR's decreasing nearly tenfold in recent years, a sensor still costs thousands of dollars, whereas a radar sensor costs a few hundred. Second, radar uses a lower frequency than LiDAR, allowing it to be more robust in rainy or foggy conditions. Finally, radar data offers accurate velocity estimation, whereas LiDAR data depends on temporal fusion to infer it.

In this chapter, we propose to use camera-radar fusion as a prime candidate to satisfy all these desiderata. By performing the fusion in the shared bird's-eye view (BEV) space, we retain both semantic and geometric information from the input modalities, which allows the model to exploit the complementary information from each. Furthermore, we design the radar encoder and radar-guided view transformation in an optimization-friendly manner. In this way, our model, with TensorRT acceleration, runs 4.5× faster than competing methods, while achieving 2.1% better NDS. Together, these techniques allow us to deploy the model on an edge device and realize real-time latency.

3.2 Radar

Similar to LiDAR, radar captures data in the form of a 3D point cloud. As depicted in Figure 3-2, radar data is usually sparser and noisier compared to LiDAR data. In fact, radar data can be over 100× sparser than 32-beam LiDAR data, and more than 5× sparser than 1-beam LiDAR data. Nonetheless, a surface-level comparison of the two modalities overlooks their inherent differences and the unique advantages of radar. For instance, radar has superior coverage of distant objects and enables accurate velocity estimation, which are both very crucial for 3D perception.

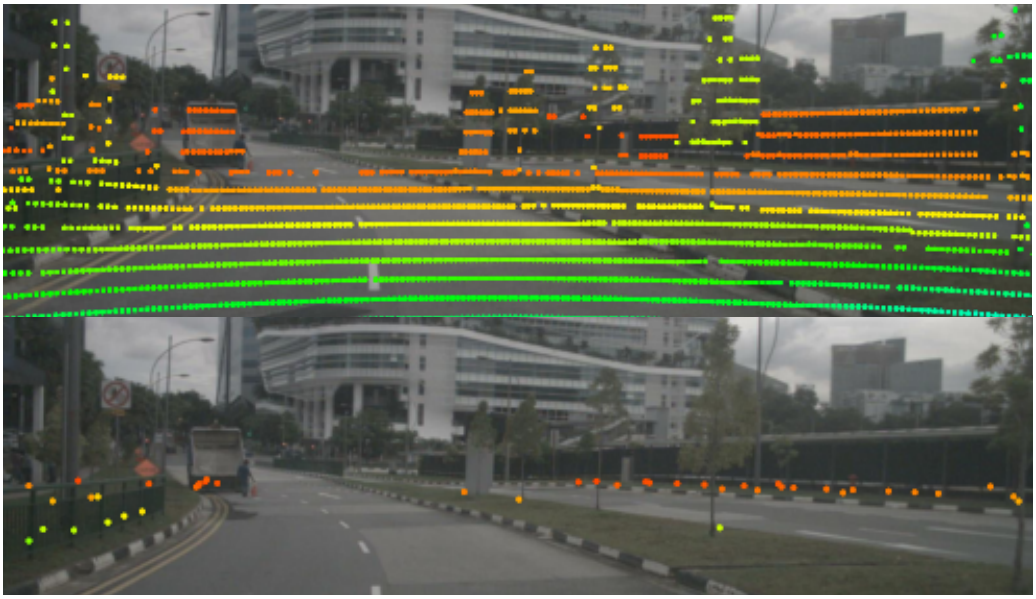

Figure 3-2: Radar returns are far sparser than LiDAR point clouds.

The incorporation of radar in 3D object detection models has been relatively tepid among the research community, primarily due to its sparsity and noise of the output point cloud, and general inferiority to LiDAR data. Such differences in quality are readily apparent when visualized, as in Figure 3-2. However, when practical characteristics like cost and robustness are considered, radar sensors become a prime candidate for complementing camera-based object detection.

In this section, we motivate our usage of camera-radar fusion by studying the characteristics of radar returns, especially compared to LiDAR and sparse LiDAR data. In particular, we use the Nuscenes [5] dataset, and compare 32-beam LiDAR, 1-beam LiDAR, and radar data. Following common practice, we aggregate 6 frames of previous radar data and 10 frames of previous LiDAR data, corresponding to the previous 0.5 seconds. The preliminary results show that radar data has favorable characteristics for complementing camera-based object detection. In particular, radar

Modality	<10m	10-25m	25-50m	>50m	All
Radar	54	391	850	264	1589
1B LiDAR	2916	3643	2267	154	8980
32B LiDAR	156818	67278	16623	1039	241758

Table 3.1: Analysis of the depth distribution of various modalities. Despite being much sparser overall, radar points are relatively denser at long range, a beneficial characteristic to complement camera-based detection.

returns maintain high coverage for distance objects and significantly improve predicted velocity error.

3.2.1 Object Coverage

As illustrated in Tables 3.1 and 3.2, while radar returns are generally sparser than LiDAR, they are relatively dense at long range, where accurate localization is especially deficient in camera-only detectors. Moreover, they offer superior object coverage than sparse LiDAR and even comparable coverage to dense LiDAR at distances greater than 50 meters. Here, an object is considered covered if at least one radar or LiDAR point falls within its corresponding 3D cuboid. Maintaining good object coverage at long ranges is crucial, particularly since camera-based localization struggles disproportionately at these distances. Thus, having excellent object coverage at long range is a desirable quality in camera-radar fusion.

3.2.2 Velocity Estimation

Accurate velocity estimation is crucial for 3D perception since it offers important information for motion prediction. Almost all LiDAR perception models employ multi-frame stacking to enable temporal reasoning. In contrast, radar data can directly measure the tangential velocity component of a moving target. In Figure 3-3,

	<10m	10-25m	25-50m	>50m	All
Objects	2.8	9.3	9.9	3.5	25.5
Radar	51%	60%	65%	28%	57%
LiDAR (1b)	46%	56%	44%	9%	44%
LiDAR (32b)	100%	100%	100%	32%	90%

Table 3.2: Object coverage at various distances. Despite its sparsity, radar data provides superior object coverage compared to sparse LiDAR data and is even comparable to dense LiDAR data at long range.

we analyze the correlation between the velocity obtained from radar returns and the ground truth velocity of each covered object. When considering only moving objects, we find a strong linear correlation of $r^2 = 0.77$, which provides a robust basis for accurate velocity estimation using camera-radar fusion.

Together, these observations motivate the focus on radar as a complementary modality to camera-based approaches for practical 3D object detection.

3.3 Method

Following many recent works [25, 49, 30], and as motivated in Chapter 2, we perform the camera-radar fusion in the bird's-eye view (BEV) space. We first extract image and radar features with encoders and then project the image features onto BEV using a radar-guided view transformer. Finally, we fuse the multi-sensory features using a BEV encoder, and generate the final prediction using a detection head. Figure 3-4 illustrates our BEVFusion-R.

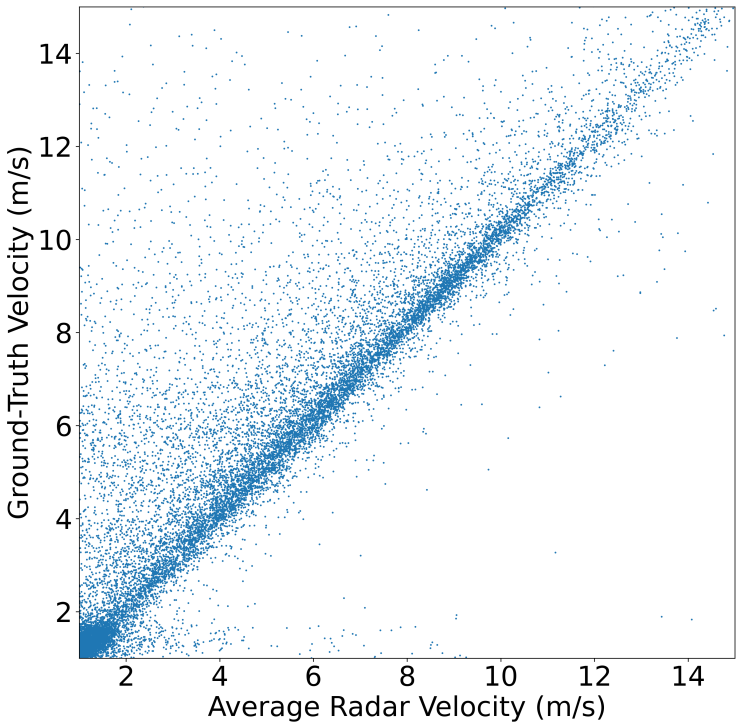

Figure 3-3: The velocity from radar exhibits a strong linear correlation with the actual velocity.

3.3.1 Image Encoder

We encode multi-view RGB images with ResNet [21] and fuse multi-scale feature maps with FPN [42]. Despite achieving higher accuracy, more advanced image encoders, like Swin Transformer [48], tend to be less efficient.

3.3.2 Radar Encoder

Most existing camera-radar fusion approaches [30] use variants of PointNet++ [67] as their radar encoders. However, they are not very efficient or hardware-friendly [50] due to their costly neighbor query and gathering operations. In this paper, we adopt PointPillars [34] as our radar encoder, which uses PointNet [66] within each pillar to

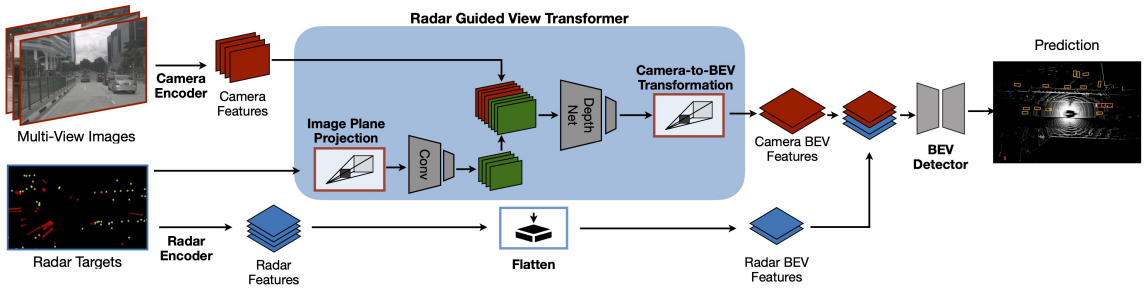

Figure 3-4: **BEVFusion-R** is a highly efficient multi-modal 3D perception model that leverages information from radar and camera sensors within the shared bird's-eye-view (BEV) space. In contrast to BEVFusion [49], our approach incorporates a radar-guided view transformer, which utilizes precise depth information from 3D radar to explicitly enhance camera-to-BEV projection.

extract features. We increase the size of the pillar to accommodate the additional noise of the radar returns. Despite its simplicity, it achieves comparable performance to strong sparse convolution-based encoders [96, 101, 79] while being 7× faster (see Table 3.4 for detailed results).

3.3.3 Radar-Guided View Transformer

Since radar features are in BEV and image features are in the perspective view, we follow LSS/BEVDet [65, 26] to transform image features into BEV. We first predict the depth distribution of each pixel and then "splat" each pixel into 3D space based on the depth prediction. As discussed in Section 2.3.2, the depth quality is a significant performance bottleneck since estimating depth from a single image is an inherently ill-posed problem. When image features are splatted to an inaccurate BEV position, the semantic information can become misaligned with the radar features. To combat this, given that radar returns provide sparse, noisy but relatively accurate spatial information, we have re-designed the view transformer to be *radar-guided*. Specifically,

we first height-expand the radar point cloud [51] and project radar points onto the image plane. Next, we feed the radar point features and one-hot encoded radar depth into a lightweight CNN. Finally, we input the resulting radar features with the image features to the depth estimation module.

3.3.4 BEV Encoder

Once both image and radar features are in BEV, we proceed to fuse them using a BEV encoder following BEVFusion [49]. Previous works have used deformable attention or deformable convolution in the BEV encoder to address spatial misalignment between modalities in the LSS paradigm. However, these operators have an irregular data access pattern, making them challenging to accelerate and deploy on hardware. Instead, we observe that additional standard convolutions with residual connections can also increase the receptive field and improve performance without requiring these hardware-unfriendly operations.

3.3.5 Deployment

As our target hardware is NVIDIA GPUs, we deploy our model using `TensorRT`[1] for the best system optimizations. Since deployment and efficiency was a consideration since the model design, almost all operators in BEVFusion-R are directly deployable using `TensorRT`, as we avoid using nearest neighbor query (in radar encoder) or deformable convolution (in BEV encoder). For unsupported operators that were unavoidable, such as the BEV pooling in the view transformer, we develop custom `ONNX` and `TensorRT` plugins for each of them. By registering these custom operations directly in PyTorch [64], then subsequently linking the corresponding `TensorRT` plugin,

[1] https://github.com/NVIDIA/TensorRT

the entire model can be exported as a single ONNX file. Moreover, this deployment pipeline is readily extendable to further custom operations and simplifies the process, in contrast with alternative approaches such as generating multiple inference engines. As a result, our entire model is deployment-ready and can be run in an end-to-end manner.

3.4 Experiments

We conduct all our experiments on nuScenes [5], which is currently the largest publicly available multimodal dataset that includes radar data. We evaluate our model and baselines using two primary metrics, NDS and mAP, on the validation set. We measure the latency numbers on a single NVIDIA RTX 3090 GPU, in FP16 with a batch size of 1. As temporal fusion is not the focus of this work, we do not employ temporal fusion, as proposed by [25].

3.4.1 Main Results

We compare BEVFusion-R with state-of-the-art single-frame camera-radar fusion detectors, as summarized in Table 3.3. Our method not only achieves the highest NDS score but also significantly reduces the inference latency of CRN [30] by **4.5**×, thanks to our deployment-friendly design. Moreover, BEVFusion-R operates at a remarkable speed of 24 frames per second (FPS) on an NVIDIA Jetson AGX Orin, enabling real-time 3D perception at the edge. We anticipate that BEVFusion-R can further leverage the benefits of multi-frame camera inputs, as observed in [30]. In order to realize the vision of a practical 3D detection algorithm, we demonstrate that our model is easily deployable on edge devices, achieving real-time latency. We choose

	Latency	NDS	mAP	mATE	mASE	mAOE	mAAE	mAVE
MVFusion [89]	–	45.5	38.0	0.675	0.258	0.372	0.198	0.394
CenterFusion [61]	219ms	45.3	33.2	0.649	0.263	0.535	0.142	0.540
FUTR-3D [8]	271ms	51.1	39.9	0.647	0.270	0.365	0.189	0.413
CRAFT [29]	244ms	51.7	41.1	0.494	0.276	0.454	0.176	0.486
RCBEV4D [105]	–	49.7	38.1	0.526	0.262	0.445	0.185	0.465
CRN [30]	49ms	50.3	42.9	0.519	–	0.577	–	0.520
BEVFusion-R	**11ms**	**52.4**	42.8	0.523	0.273	0.542	0.185	0.379

Table 3.3: Results for single-frame camera-radar fusion detectors on the nuScenes validation set. BEVFusion-R achieves a state-of-the-art nuScenes detection score (NDS) while maintaining a 92 FPS speed, which is 4.5× faster than the best available baseline at the time of writing.

the NVIDIA Orin as our target edge device.

3.4.2 Ablation Studies

In order to validate the design choices of our fusion detection model, we carry out several ablation studies on the Nuscenes *val* set.

To assess the impact of radar encoder architecture, we conducted an ablation study presented in Table 3.4. We compared VoxelNet [96] (VN) and PointPillars (PP) [34], which are commonly employed as 3D backbones in LiDAR-only and camera-LiDAR 3D object detectors. VoxelNet generally exhibits superior accuracy at the cost of increased latency, as shown in large accuracy gap for both LiDAR settings, but especially dense 32-beam LiDAR. However, this accuracy advantage diminishes rapidly when dealing with sparse 3D data, such as a single beam LiDAR or radar, since neighboring voxels become less frequent. We therefore choose PointPillars as our radar feature encoder in BEVFusion-R, which is **7**× faster than VoxelNet and delivers almost the same accuracy.

Table 3.5 shows the impact that accurate depth estimation can bring to a fusion

	NDS			Latency (ms)		
	VN	PP	Δ	VN	PP	Δ
32B LiDAR	64.8	60.2	+4.6	118.0	88.0	+30.0
1B LiDAR + Camera	53.2	50.5	+2.7	37.7	3.7	+34.0
Radar + Camera	48.7	48.6	+0.1	16.7	2.4	+14.3

Table 3.4: Ablation study on the choice of encoder for the point cloud. VN denotes VoxelNet. PP denotes PointPillars. Despite worsening latency in all cases, using VoxelNet, a 3D based encoder, will improve performance in the 32-beam and 1-beam LiDAR settings. However, for radar inputs, it performs no better than PointPillars.

Method	mAP	NDS
No depth supervision	38.9	51.0
With depth supervision	40.7	51.6
With depth supervision + Radar input	41.4	52.2
With depth supervision + LiDAR input	50.2	58.5

Table 3.5: Ablation study on the View Transformer. Not only does explicit depth supervision enable stronger performance, but using input from other modalities further improves the accuracy. The final setting is an oracle setting: not achievable in practice, but rather serving as an upper bound for optimal depth estimation.

detection model. As observed by [39], depth supervision can improve the overall performance significantly. Moreover, allowing the depth estimation module to access radar data allows it to more accurately predict the depth, again leading to better results. We hope that better estimation frameworks can further close the performance gap to the oracle setting presented in the last row.

In addition, we emphasize the significance of aggregating multiple radar frames, as depicted in Figure 3-5. Motivated by Figure 3-3, the velocity estimation performance improves by 25.7% when we increase the radar input from one frame to ten frames. Moreover, considering that radar provides accurate 3D localization for a substantial portion of objects (as demonstrated in Table 3.2), incorporating more radar frames

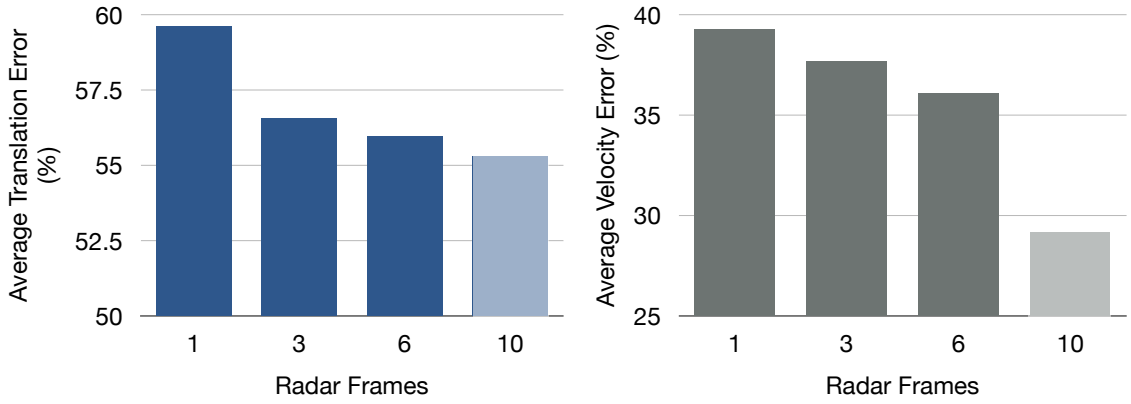

Figure 3-5: Ablation study on the number of radar frames included. As the number of radar points increases, both object localization (translation error) and velocity estimation (velocity error) improve.

also leads to a 7.2% reduction in translation error.

3.4.3 Latency and Real Time Deployment

We conduct an in-depth analysis of the latency breakdown in BEVFusion-R, shown in Figure 3-6. We compare our model with CRN [30], which is recognized as the fastest available camera-radar fusion 3D object detector. Our BEVFusion-R is executed end-to-end using TensorRT. For CRN, as the source code is unavailable at the time of writing, we estimate its deployment latency by running all TensorRT-compatible operators with the TensorRT backend while executing the remaining modules (*e.g.* PointNet++ radar encoder, multi-modal fuser based on deformable attention) in PyTorch. Consequently, our model benefits more from the faster TensorRT backend (**3.8**× *vs.* 2.0× faster) compared to CRN. In addition, BEVFusion-R achieves 24.4 FPS on Orin, whereas CRN could only run at an estimated 10.7 FPS. It is noteworthy that the nuScenes radar sensor operates at 13 FPS. Consequently, BEVFusion-R still maintains real-time performance on the edge, while CRN could not.

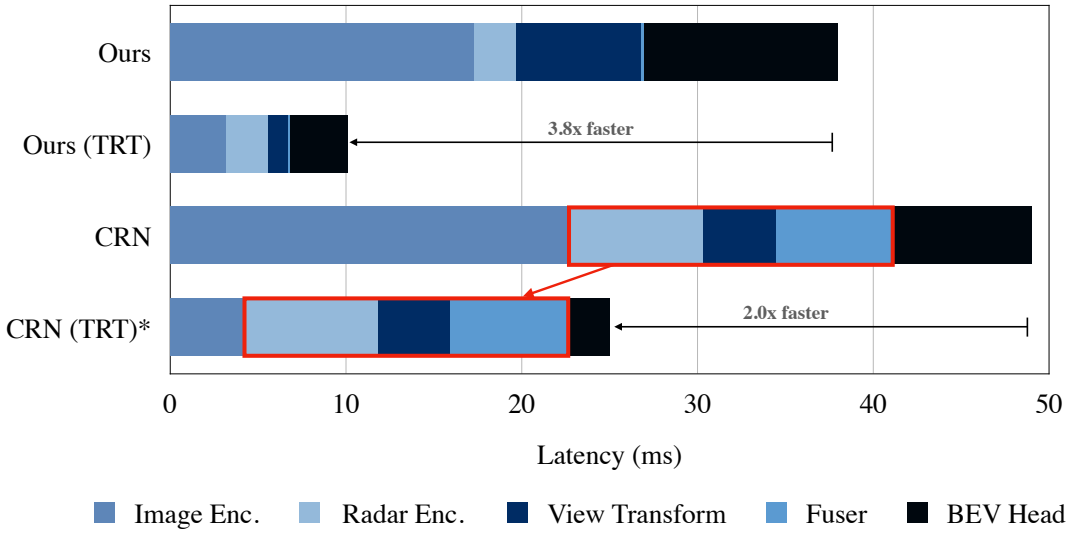

Figure 3-6: Acceleration of our model via TensorRT. Despite having similar PyTorch latency as CRN[30], our model is entirely hardware friendly, whereas the red sections in CRN are not. As such, we observe a much more dramatic speedup.

3.5 Conclusion

In this work, we developed BEVFusion-R, an efficient and performant camera-radar fusion model tailored for practical real-time autonomous driving. By performing sensor fusion in bird's-eye view space with the help of a radar-guided view transformer, we allow the model to exploit the complementary characteristics of the sensors. Moreover, by considering hardware optimization from the beginning and performing end-to-end deployment and acceleration, our model can simultaneously achieve state-of-the-art accuracy while running 4.5× faster than previous methods. Capable of real-time latency on an edge device, we hope that our method will inspire future research on lightweight camera-radar fusion for 3D perception.

Chapter 4

Conclusion

In this thesis, we have developed two models for efficient 3D detection, a key component of ubiquitous adoption of autonomous driving: BEVFusion and BEVFusion-R.

BEVFusion targets efficient camera-LiDAR fusion, which naturally achieves the highest accuracy due to the precise nature of LiDAR point clouds. By performing the multi-sensor fusion in the Bird's Eye View (BEV) space, the model is able to simultaneously retain maximal semantic and geometric information from both input modalities. Moreover, our model demonstrates strong robustness to adverse weather conditions, especially compared to competing works.

BEVFusion-R is developed as a camera-radar fusion model with cost efficiency and deployment readiness as additional considerations. Leveraging the same multi-sensor fusion paradigm, we select radar as a cost-effective alternative to LiDAR. By using a lightweight radar encoder and designing the entire model with `TensorRT` acceleration in mind, BEVFusion-R runs on the edge at real-time speeds.

4.1 Future Work

There are a few interesting directions for extension of the models developed in this thesis.

One simple extension of this work would be the incorporation of temporal fusion. While LiDAR and radar frames are aggregated for the past 0.5 seconds, it is fairly common to use BEV space temporal fusion to incorporate multiple images as well [25, 30]. While BEVFusion and BEVFusion-R are both single image frame models, incorporating this feature would allow for significant performance improvement while sacrificing minimal latency, since the BEV features used for temporal fusion can be cached and re-used.

Second, the BEVFusion model struggles with deployment readiness due to its reliance on sparse 3D convolution based operators. 3D convolutions are not well supported in `TensorRT`; however, there are separate backend engines for these workloads, such as [11, 79]. Allowing for hardware acceleration represents an interesting direction for further development of camera-LiDAR fusion models.

Finally, one shortcoming of BEVFusion is in the difficult depth perception problem in the camera-to-BEV projection. While addressed in BEVFusion-R with the radar-guided view transformer, upper-bound studies in both this thesis and [39] indicate promise in further utilizing either LiDAR or radar data in predicting accurate depth from images.

Bibliography

[1] Mehmet Aygün, Aljoša Ošep, Mark Weber, Maxim Maximov, Cyrill Stachniss, Jens Behley, and Laura Leal-Taixé. 4d panoptic lidar segmentation, 2021. 12

[2] Junjie Bai, Fang Lu, Ke Zhang, et al. Onnx: Open neural network exchange. https://github.com/onnx/onnx, 2019. 23

[3] Xuyang Bai, Zeyu Hu, Xinge Zhu, Qingqiu Huang, Yilun Chen, Hongbo Fu, and Chiew-Lan Tai. TransFusion: Robust LiDAR-Camera Fusion for 3D Object Detection with Transformers. In *CVPR*, 2022. 17, 26, 30, 32

[4] Garrick Brazil and Xiaoming Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, Seoul, South Korea, 2019. 15

[5] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A Multimodal Dataset for Autonomous Driving. In *CVPR*, 2020. 6, 13, 18, 21, 24, 29, 33, 36, 42

[6] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment, 2020. 17

[7] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, and James Hays. Argoverse: 3d tracking and forecasting with rich maps, 2019. 13

[8] Xuanyao Chen, Tianyuan Zhang, Yue Wang, Yilun Wang, and Hang Zhao. FUTR3D: A Unified Sensor Fusion Framework for 3D Detection. *arXiv*, 2022. 26, 30, 43

[9] Zehui Chen, Zhenyu Li, Shiquan Zhang, Liangji Fang, Qinghong Jiang, Feng Zhao, Bolei Zhou, and Hang Zhao. Autoalign: Pixel-instance feature aggregation for multi-modal 3d object detection, 2022. 30

[10] Yukyung Choi, Namil Kim, Soonmin Hwang, Kibaek Park, Jae Shin Yoon, Kyounghwan An, and In So Kweon. Kaist multi-spectral day/night data set for autonomous and assisted driving. *IEEE Transactions on Intelligent Transportation Systems*, 19(3):934–948, 2018. 13

[11] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks. In *CVPR*, 2019. 48

[12] Thomas G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000. 16

[13] Xuanyi Dong, Junshi Huang, Yi Yang, and Shuicheng Yan. More is less: A more complicated network with less inference complexity, 2017. 16

[14] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets Robotics: The KITTI Dataset. *IJRR*, 2013. 13

[15] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *International Journal of Robotics Research (IJRR)*, 2013. 33

[16] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361, 2012. 13

[17] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014. 15

[18] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3D Semantic Segmentation With Submanifold Sparse Convolutional Networks. In *CVPR*, 2018. 14

[19] Song Han, Huizi Mao, and William J. Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2016. 16

[20] Song Han, Jeff Pool, John Tran, and William J. Dally. Learning both weights and connections for efficient neural networks, 2015. 16

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 27, 39

[22] Yihui He, Ji Lin, Zhijian Liu, Hanrui Wang, Li-Jia Li, and Song Han. Amc: Automl for model compression and acceleration on mobile devices, 2019. 16

[23] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. 16

[24] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications, 2017. 17

[25] Junjie Huang and Guan Huang. Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint arXiv:2203.17054*, 2022. 15, 38, 42, 48

[26] Junjie Huang, Guan Huang, Zheng Zhu, Ye Yun, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021. 15, 28, 30, 40

[27] Forrest N. Iandola, Song Han, Matthew W. Moskewicz, Khalid Ashraf, William J. Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size, 2016. 17

[28] Jiyoon Kim, Bum-jin Park, and Jisoo Kim. Empirical analysis of autonomous vehicle's lidar detection performance degradation for actual road driving in rain and fog. *Sensors*, 23:2972, 03 2023. 13

[29] Youngseok Kim, Sanmin Kim, Jun Won Choi, and Dongsuk Kum. Craft: Camera-radar 3d object detection with spatio-contextual fusion transformer, 2022. 43

[30] Youngseok Kim, Sanmin Kim, Juyeb Shin, Jun Won Choi, and Dongsuk Kum. Crn: Camera radar net for accurate, robust, efficient 3d perception, 2023. 8, 27, 34, 38, 39, 42, 43, 45, 46, 48

[31] Junho Koh, Junhyung Lee, Youngwoo Lee, Jaekyum Kim, and Jun Won Choi. Mgtanet: Encoding sequential lidar points using long short-term motion-guided temporal attention for 3d object detection. *arXiv preprint arXiv:2212.00442*, 2022. 33

[32] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. 22

[33] Abhinav Kumar, Garrick Brazil, and Xiaoming Liu. Groomed-nms: Grouped mathematically differentiable nms for monocular 3d object detection, 2021. 15

[34] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, and Jiong Yang. PointPillars: Fast Encoders for Object Detection from Point Clouds. In *CVPR*, 2019. 14, 22, 27, 30, 39, 43

[35] Bingzhao Li, Qixuan Lin, and Mo Li. Frequency–angular resolving lidar using chip-scale acousto-optic beam steering. *Nature*, pages 1–7, 2023. 20

[36] Qi Li, Yue Wang, Yilun Wang, and Hang Zhao. Hdmapnet: An online hd map construction and evaluation framework, 2022. 12

[37] Yanwei Li, Yilun Chen, Xiaojuan Qi, Zeming Li, Jian Sun, and Jiaya Jia. Unifying Voxel-based Representation with Transformer for 3D Object Detection. In *NeurIPS*, 2022. 30, 34

[38] Yingwei Li, Adams Wei Yu, Tianjian Meng, Ben Caine, Jiquan Ngiam, Daiyi Peng, Junyang Shen, Bo Wu, Yifeng Lu, Denny Zhou, et al. DeepFusion: Lidar-Camera Deep Fusion for Multi-Modal 3D Object Detection. In *CVPR*, 2022. 26

[39] Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. *arXiv preprint arXiv:2206.10092*, 2022. 28, 30, 32, 44, 48

[40] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers. In *ECCV*, 2022. 32, 33

[41] Ming Lin, Pichao Wang, Zhenhong Sun, Hesen Chen, Xiuyu Sun, Qi Qian, Hao Li, and Rong Jin. Zen-nas: A zero-shot nas for high-performance deep image recognition, 2021. 17

[42] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature Pyramid Networks for Object Detection. In *CVPR*, 2017. 39

[43] Haibin Liu, Chao Wu, and Huanjie Wang. Real time object detection using lidar and camera fusion for autonomous driving. *Scientific Reports*, 13, 05 2023. 7, 25

[44] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search, 2019. 17

[45] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection, 2022. 30, 33

[46] Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Aqi Gao, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petrv2: A unified framework for 3d perception from multi-camera images, 2022. 33

[47] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin Transformer V2: Scaling Up Capacity and Resolution. In *CVPR*, 2022. 27

[48] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In *ICCV*, 2021. 39

[49] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. BEVFusion: Multi-Task Multi-Sensor Fusion with Unified Bird's-Eye View Representation. In *ICRA*, 2023. 7, 9, 17, 24, 25, 26, 27, 30, 32, 38, 40, 41

[50] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-Voxel CNN for Efficient 3D Deep Learning. In *NeurIPS*, 2019. 14, 15, 39

[51] Chen-Chou Lo and Patrick Vandewalle. Depth estimation from monocular images and sparse radar using deep ordinal regression network. In *2021 IEEE International Conference on Image Processing (ICIP)*. IEEE, sep 2021. 41

[52] Jiachen Lu, Zheyuan Zhou, Xiatian Zhu, Hang Xu, and Li Zhang. Learning ego 3d representation as ray tracing. In *European Conference on Computer Vision*, 2022. 33

[53] Tao Lu, Xiang Ding, Haisong Liu, Gangshan Wu, and Limin Wang. Link: Linear kernel for lidar-based 3d perception, 2023. 33

[54] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression, 2017. 16

[55] Yuexin Ma, Tai Wang, Xuyang Bai, Huitong Yang, Yuenan Hou, Yaming Wang, Yu Qiao, Ruigang Yang, Dinesh Manocha, and Xinge Zhu. Vision-centric bev perception: A survey, 2023. 9, 30

[56] Jiageng Mao, Minzhe Niu, Chenhan Jiang, Hanxue Liang, Jingheng Chen, Xiaodan Liang, Yamin Li, Chaoqiang Ye, Wei Zhang, Zhenguo Li, Jie Yu, Hang Xu, and Chunjing Xu. One million scenes for autonomous driving: Once dataset, 2021. 13

[57] Jiageng Mao, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 3d object detection for autonomous driving: A comprehensive survey, 2023. 9, 12, 14, 30

[58] Daniel Maturana and Sebastian Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In *IROS*, 2015. 14

[59] Gregory P. Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K. Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving, 2019. 15

[60] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training, 2018. 16

[61] Ramin Nabati and Hairong Qi. Centerfusion: Center-based radar and camera fusion for 3d object detection. *arXiv preprint arXiv:2011.04841*, 2020. 43

[62] Xuran Pan, Zhuofan Xia, Shiji Song, Li Erran Li, and Gao Huang. 3D Object Detection with Pointformer. In *CVPR*, 2021. 14

[63] Jinhyung Park, Chenfeng Xu, Shijia Yang, Kurt Keutzer, Kris Kitani, Masayoshi Tomizuka, and Wei Zhan. Time will tell: New outlooks and a baseline for temporal multi-view 3d object detection. 2023. 30, 33

[64] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison,

Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 41

[65] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *Proceedings of the European Conference on Computer Vision*, 2020. 15, 28, 40

[66] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *CVPR*, 2017. 14, 27, 39

[67] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In *NeurIPS*, 2017. 14, 39

[68] Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search, 2019. 17

[69] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018. 15

[70] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016. 15

[71] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. OctNet: Learning Deep 3D Representations at High Resolutions. In *CVPR*, 2017. 15

[72] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection, 2018. 28

[73] B. Helou S. Vora, A. H. Lang and O. Beijbom. PointPainting: Sequential Fusion for 3D Object Detection. In *CVPR*, 2020. 7, 26, 30

[74] Guangsheng Shi, Ruifeng Li, and Chao Ma. PillarNet: Real-Time and High-Performance Pillar-based 3D Object Detection. In *ECCV*, 2022. 14

[75] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection, 2021. 15

[76] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud. In *CVPR*, 2019. 14

[77] S Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. 2015. 11

[78] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *CVPR*, 2020. 12, 13, 18, 24, 33

[79] Haotian Tang, Zhijian Liu, Xiuyu Li, Yujun Lin, and Song Han. TorchSparse: Efficient Point Cloud Inference Engine. In *MLSys*, 2022. 40, 48

[80] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. In *ECCV*, 2020. 15, 17

[81] Chunwei Wang, Chao Ma, Ming Zhu, and Xiaokang Yang. Pointaugmenting: Cross-modal augmentation for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11794–11803, 2021. 26, 30

[82] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Haq: Hardware-aware automated quantization with mixed precision, 2019. 16

[83] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. Hardware-centric AutoML for mixed-precision quantization. *International Journal of Computer Vision*, 128(8-9):2035–2048, jun 2020. 16

[84] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection, 2021. 15

[85] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q. Weinberger. Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving, 2020. 15

[86] Yue Wang, Alireza Fathi, Abhijit Kundu, David Ross, Caroline Pantofaru, Thomas Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving, 2020. 14

[87] Yue Wang, Vitor Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, , and Justin M. Solomon. DETR3D: 3D Object Detection from Multi-view Images via 3D-to-2D Queries. In *CoRL*, 2021. 26, 32

[88] Yue Wang and Justin M. Solomon. Object DGCNN: 3D Object Detection using Dynamic Graphs. In *NeurIPS*, 2021. 30

[89] Zizhang Wu, Guilian Chen, Yuanzhu Gan, Lei Wang, and Jian Pu. Mvfusion: Multi-view 3d object detection with semantic-aligned radar and camera fusion, 2023. 34, 43

[90] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models, 2023. 16

[91] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, Yunlong Wang, and Diange Yang. Pandaset: Advanced sensor suite dataset for autonomous driving, 2021. 13

[92] Enze Xie, Zhiding Yu, Daquan Zhou, Jonah Philion, Anima Anandkumar, Sanja Fidler, Ping Luo, and Jose M Alvarez. M^ 2bev: Multi-camera joint 3d detection and segmentation with unified birds-eye view representation. *arXiv preprint arXiv:2204.05088*, 2022. 34

[93] Kaixin Xiong, Shi Gong, Xiaoqing Ye, Xiao Tan, Ji Wan, Errui Ding, Jingdong Wang, and Xiang Bai. Cape: Camera view position embedding for multi-view 3d object detection. 2023. 33

[94] Shaoqing Xu, Dingfu Zhou, Jin Fang, Junbo Yin, Zhou Bin, and Liangjun Zhang. Fusionpainting: Multimodal fusion with adaptive attention for 3d object detection, 2021. 30

[95] Junjie Yan, Yingfei Liu, Jianjian Sun, Fan Jia, Shuailin Li, Tiancai Wang, and Xiangyu Zhang. Cross modal transformer via coordinates encoding for 3d object dectection. *arXiv preprint arXiv:2301.01283*, 2023. 34

[96] Yan Yan, Yuxing Mao, and Bo Li. SECOND: Sparsely Embedded Convolutional Detection. *Sensors*, 2018. 14, 30, 40, 43

[97] Chenyu Yang, Yuntao Chen, Hao Tian, Chenxin Tao, Xizhou Zhu, Zhaoxiang Zhang, Gao Huang, Hongyang Li, Yu Qiao, Lewei Lu, Jie Zhou, and Jifeng Dai. Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision, 2022. 33

[98] Zeyu Yang, Jiaqi Chen, Zhenwei Miao, Wei Li, Xiatian Zhu, and Li Zhang. Deepinteraction: 3d object detection via modality interaction. In *NeurIPS*, 2022. 17, 34

[99] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Center-based 3D Object Detection and Tracking. In *CVPR*, 2021. 30, 33

[100] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. Multimodal Virtual Point 3D Detection. In *NeurIPS*, 2021. 26, 30

[101] Tianwei Yin, Xingyi Zhou, and Philipp Krähenbühl. CenterPoint++ Submission to the Waymo Real-time 3D Detection Challenge. Technical report, 2022. 14, 32, 40

[102] Jin Hyeok Yoo, Yecheol Kim, Jisong Kim, and Jun Won Choi. 3d-CVF: Generating joint camera and LiDAR features using cross-view spatial feature fusion for 3d object detection. In *Computer Vision – ECCV 2020*, pages 720–736. Springer International Publishing, 2020. 30

[103] Chunyu Yuan and Sos S. Agaian. A comprehensive review of binary neural network. *Artificial Intelligence Review*, mar 2023. 16

[104] Yunpeng Zhang, Zheng Zhu, Wenzhao Zheng, Junjie Huang, Guan Huang, Jie Zhou, and Jiwen Lu. Beverse: Unified perception and prediction in birds-eye-view for vision-centric autonomous driving, 2022. 34

[105] Taohua Zhou, Junjie Chen, Yining Shi, Kun Jiang, Mengmeng Yang, and Diange Yang. Bridging the view disparity between radar and camera features for multi-modal fusion 3d object detection. *IEEE Transactions on Intelligent Vehicles*, 8(2):1523–1535, 2023. 43

[106] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. In *arXiv preprint arXiv:1904.07850*, 2019. 15

[107] Yin Zhou and Oncel Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In *CVPR*, 2018. 14, 22, 27

[108] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-Balanced Grouping and Sampling for Point Cloud 3D Object Detection. *arXiv*, 2019. 14

[109] Zijian Zhu, Yichi Zhang, Hai Chen, Yinpeng Dong, Shu Zhao, Wenbo Ding, Jiachen Zhong, and Shibao Zheng. Understanding the robustness of 3d object detection with bird's-eye-view representations in autonomous driving, 2023. 9, 18, 32

[110] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition, 2018. 17