Simple-BEV: What Really Matters for Multi-Sensor BEV Perception?

Adam W. Harley[1], Zhaoyuan Fang[2], Jie Li[3], Rares Ambrus[3], Katerina Fragkiadaki[2]
https://simple-bev.github.io/

***Abstract*— Building 3D perception systems for autonomous vehicles that do not rely on high-density LiDAR is a critical research problem because of the expense of LiDAR systems compared to cameras and other sensors. Recent research has developed a variety of camera-only methods, where features are differentiably "lifted" from the multi-camera images onto the 2D ground plane, yielding a "bird's eye view" (BEV) feature representation of the 3D space around the vehicle. This line of work has produced a variety of novel "lifting" methods, but we observe that other details in the training setups have shifted at the same time, making it unclear *what really matters* in top-performing methods. We also observe that using cameras alone is not a real-world constraint, considering that additional sensors like *radar* have been integrated into real vehicles for years already. In this paper, we first of all attempt to elucidate the high-impact factors in the design and training protocol of BEV perception models. We find that batch size and input resolution greatly affect performance, while lifting strategies have a more modest effect—even a simple parameter-free lifter works well. Second, we demonstrate that radar data can provide a substantial boost to performance, helping to close the gap between camera-only and LiDAR-enabled systems. We analyze the radar usage details that lead to good performance, and invite the community to re-consider this commonly-neglected part of the sensor platform.**

I. Introduction

There is great interest in building 3D-aware perception systems for *LiDAR-free* autonomous vehicles, whose sensor platforms typically constitute multiple RGB cameras, and multiple radar units. While LiDAR data enables highly accurate 3D object detection [1, 2], the sensors themselves are arguably too expensive for large-scale deployment [3], especially as compared to current camera and radar units. Most current works focus on producing an accurate "bird's eye view" (BEV) semantic representation of the 3D space surrounding the vehicle, using multi-view camera input alone. This representation captures the information required for driving-related tasks, such as navigation, obstacle detection, and moving-obstacle forecasting. We have seen extremely rapid progress in this domain: for example, BEV vehicle semantic segmentation IOU improved from 23.9 [4] to 44.4 [5] in just two years!

While this progress is encouraging, the focus on innovation and accuracy has come at the cost of system simplicity, and risks obscuring "what really matters" for performance [6, **?**]. There has been a particular focus on innovating new techniques for "lifting" features from the 2D image plane(s) onto the BEV plane. For example, some work has explored using homographies to warp features directly onto the ground plane [7], using depth estimates to place features at their approximate 3D locations [8, 9], using MLPs with various geometric biases [10, 11, 12], and most recently, using geometry-aware transformers [13] and deformable attention across space and time [5]. At the same time, implementation details have gradually shifted toward using more-powerful backbones and higher-resolution inputs, making it difficult to measure the actual impact of these developments in lifting. We propose a model where the "lifting" step is **parameter-free** and does not rely on depth estimation: we simply define a 3D volume of coordinates over the BEV plane, project these coordinates into all images, and average the features sampled from the projected locations. When this simple model is tuned well, it exceeds the performance of state-of-the-art models while also being faster and more parameter-efficient. We measure the independent effects of batch size, image resolution, augmentations, and 2D-to-BEV lifting strategy, and show empirically that good selection of input resolution and batch size can improve performance by more than 10 points (all other factors held equal), while the difference between the worst and best lifting methods is only 4 points – which is particularly surprising because lifting methods have been the main focus of earlier work.

We further show that results can be substantially improved by incorporating input from **radar**. While recent efforts have focused on using cameras and/or LiDAR, we note that radar sensors have been integrated into real vehicles for years already [14], and *cameras plus radar* performs far better than cameras alone. While using cameras alone may give the task a certain purity (requiring metric 3D estimates from 2D input), it does not reflect the reality of autonomous driving, where noisy metric data is freely available, not only from radar but from GPS and odometry. The few recent works that discuss radar in the context of semantic BEV mapping have concluded that the data is often too sparse to be useful [3, 10]. We identify that these prior works evaluated the use of *radar alone*, avoiding the multi-modal fusion problem, and perhaps missing the opportunity for RGB and radar to complement one another. We introduce a simple RGB+radar fusion strategy (rasterizing the radar in BEV and concatenating it to the RGB features) and exceed the performance of all published BEV segmentation models by a margin of 9 points.

This paper has two main contributions: First, we elucidate high-impact factors in the design and training protocol of BEV perception models. We show in particular that batch size and input resolution greatly affect performance, while lifting details have a more modest effect. Second, we demonstrate that radar data can provide a large boost to performance with a simple fusion method, and invite the

[1]Stanford University, aharley@cs.stanford.edu. [2]Carnegie Mellon University, {zhaoyuaf, katef}@cs.cmu.edu [3]Toyota Research Institute, {jie.li, rares.ambrus}@tri.global

community to re-consider this commonly-neglected part of the sensor platform. We also release code and reproducible models to facilitate future research in the area.

II. Related Work

A major differentiator in prior work on dense BEV parsing is the precise operator for "lifting" 2D perspective-view features to 3D, or directly to the ground plane.

Parameter-free unprojection: This strategy, pursued in a variety of object and scene representation models [15, 16, 17], uses the camera geometry to define a mapping between voxels and their projected coordinates, and copies 2D features to voxels along their 3D rays. We specifically follow the implementation of Harley et al. [18], which bilinearly samples a subpixel 2D feature for each 3D coordinate. Parameter-free lifting methods are not typically used in bird's eye view parsing tasks.

Depth-based unprojection: Several works estimate per-pixel depth with a monocular depth estimator, either pre-trained for depth estimation [8, 19, 20] or trained simply for the end-task [9, 21, 22], and used the depth to place features at their estimated 3D locations. This is an effective strategy, but note that if the depth estimation is perfect, it will only place "vehicle" features at the front visible surface of the vehicle, rather than fill the entire vehicle volume with features. We believe this detail is one reason that naive unprojection performs competitively with depth-based unprojection.

Homography-based unprojection: Some works estimate the ground plane instead of per-pixel depth, and use the homography that relates the image to the ground to create a warp [23, 24, 7], transferring the features from one plane to another. This operation tends to produce poor results when the scene itself is non-planar.

MLP-based unprojection: A popular approach is to convert a vertical-axis strip of image features to a forward-axis strip of ground-plane features, with an MLP [4, 10, 25]. An important detail here is that the initial ground-plane features are considered aligned with the camera frustum, and they are therefore warped into a rectilinear space using the camera intrinsics. Some works in this category use multiple MLPs, dedicated to different scales [26, 11], or to different categories [12]. As this MLP is parameter-heavy, Yang et al. [27] propose a cycle-consistency loss (mapping backward to the image-plane features) to help regularize it.

Geometry-aware transformer-like models: An exciting new trend is to transfer features using model components taken from transformer literature. Saha et al. [13] begin by defining a relationship between each vertical scan-line of an image, and the ground-plane line that it projects to, and use self-attention to learn a "translation" function between the two coordinate systems. Defining this transformer at the line level provides inductive bias to the model, reusing the lifting method across all lines. BEVFormer [5], which is concurrent work, proposes to use deformable attention operations to collect image features for a pre-defined grid of 3D coordinates. This is similar to the bilinear sampling operation in parameter-free unprojection, but with approximately $10\times$ more samples, learnable offsets for the sampling coordinates, and a learnable kernel on their combination.

Radar: In the automotive industry, radar has been in use for several years already [14]. Since radar measurements provide position, velocity, and angular orientation, the data is typically used to detect obstacles (e.g., for emergency braking), and to estimate the velocity of moving objects (e.g., for cruise control). Radar is longer-range and less sensitive to weather effects than LiDAR, and substantially cheaper. Unfortunately, the sparsity and noise inherent to radar make it a challenge to use [28, 29, 30, 29, 31]. Some early methods use radar for BEV semantic segmentation tasks much like in our work [28, 32, 29], but only in small datasets. Recent work within the nuScenes benchmark [33] has reported the data too sparse to be useful, recommending instead higher-density radar data from alternate sensor setups [3, 10]. Some recent works explore RGB-radar or RGB-Lidar fusion strategies [23, 30], but focus on detection and velocity estimation rather than BEV semantic labeling.

III. Simple-BEV Model

In this section, we describe the architecture and training setup of a basic neural BEV mapping model, which we will modify in the experiments to study which factors matter most for performance.

A. Setup and overview

Our model takes input from cameras, radars, and optionally even LiDAR. We assume that the data is synchronized across sensors. We assume that the intrinsics and relative poses of the sensors are known.

The model has a 3D metric span, and a 3D resolution. Following the baselines in this task, we set the left/right and forward/backward span to $100m \times 100m$, discretized at a resolution of 200×200. We set the up/down span to $10m$, and discretize at a resolution of 8. This volume is centered and oriented according to a reference camera, which is typically the front camera. We denote the left-right axis with X, the up-down axis with Y, and the forward-backward axis with Z.

Following related work, we first apply a 2D ResNet to compute features from each camera image, then lift to 3D, then reduce to a BEV plane, and finally apply a 2D ResNet in BEV to arrive at the output. These steps are illustrated in Figure 1, and will be described in more detail in the next section. Our lifting step is subtly different from prior work: while some works "splat" 2D features along their corresponding 3D rays [9, 21], we instead begin at 3D coordinates, and bilinearly sample a sub-pixel feature for each voxel. If radar or LiDAR is provided, we rasterize this data into a bird's eye view image, and concatenate this with the 3D feature volume before compressing the vertical dimension.

B. Architecture design

We featurize each input RGB image, shaped $3 \times H \times W$, with a ResNet-101 [34] backbone. We upsample the output of the last layer and concatenate it with the third layer output, and apply two convolution layers with instance normalization

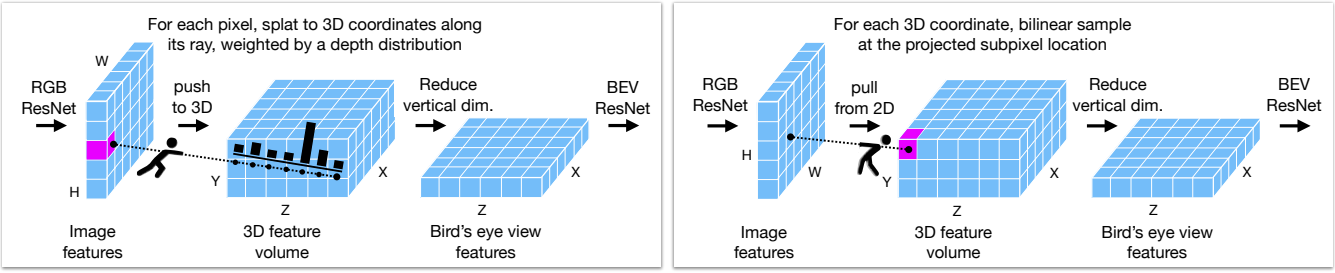

Fig. 1: **2D-to-BEV architecture, illustrated with two lifting strategies.** The left panel shows the Lift-Splat approach [9]: in this method, each 2D feature is "pushed" to 3D, filling voxels that intersect with its ray. The right panel shows our bilinear sampling approach: in this method, each 3D voxel "pulls" a feature from the 2D map, by projection and subpixel sampling.

and ReLU activations [35], arriving at feature maps with shape $C \times H/8 \times W/8$ (one eighth of the image resolution).

We project the pre-defined volume of 3D coordinates into all feature maps, and bilinearly sample there, yielding a 3D feature volume from each camera. We compute a binary "valid" volume per camera at the same time, indicating if the 3D coordinate landed within the camera frustum. We then take a valid-weighted average across the set of volumes, reducing our representation down to a single 3D volume of features, shaped $C \times Z \times Y \times X$. We then rearrange the axes so that the vertical dimension extends the channel dimension, as in $C \times Z \times Y \times X \rightarrow (C \cdot Y) \times Z \times X$, yielding a high-dimensional BEV feature map.

If radar is provided, we rasterize it to create another BEV feature map with the same spatial dimensions as the RGB-based map. We use an arbitrary number of radar channels R (including $R = 0$, meaning no radar). In nuScenes [33], each radar return consists of a total of 18 fields, with 5 of them being position and velocity, and the remainder being the result of built-in pre-processes (e.g., indicating confidence that the return is valid). We use all of this data, by using the position data to choose the nearest XZ position on the grid (if in bounds), and using the 15 non-position items as channels, yielding a BEV feature map shaped $R \times Z \times X$, with $R = 15$. If LiDAR is provided, we voxelize it to a binary occupancy grid shaped $Y \times Z \times X$, and use it in place of radar features.

We then concatenate the RGB features and radar/LiDAR features, and compress the extended channels down to a dimensionality of C, by applying a 3×3 convolution kernel. This achieves the reduction $(C \cdot Y + R) \times Z \times X \rightarrow C \times Z \times X$. At this point, we have a single plane of features, representing a bird's eye view of the scene. We process this with three blocks of a ResNet-18 [34], producing three feature maps, then use additive skip connections with bilinear upsampling to gradually bring the coarser features to the input resolution, and finally, apply two convolution layers acting as the segmentation task head. Following FIERY [21], we complement the segmentation head with auxiliary task heads for predicting centerness and offset, which serve to regularize the model. The offset head produces a vector field where, within each object mask, each vector points to the center of that object. We train the segmentation head with a cross-entropy loss, and supervise the centerness and offset fields with an L1 loss. We use an uncertainty-based learnable weighting [36] to balance the three losses.

The 3D resolution is $200 \times 8 \times 200$, and our final output resolution is 200×200. Our 3D metric span is $100m \times 10m \times 100m$. This corresponds to voxel lengths of $0.5m \times 1.25m \times 0.5m$ (in Z, Y, X order). We use a feature dimension (i.e., channel dimension C) of 128. The ResNet-101 is pre-trained for object detection [37] on COCO 2017 [38]. The BEV ResNet-18 is trained from scratch. We train end-to-end for 25,000 iterations, with the Adam-W optimizer [39] using a learning rate of 5e-4 and 1-cycle schedule [40].

C. Key factors of study

Lifting strategy: Our model is "simpler" than related work, particularly in the 2D-to-3D lifting step, which is handled by (parameter-free) bilinear sampling. This replaces, for example, depth estimation followed by splatting [9], MLPs [4, 10, 25], or attention mechanisms [13, 5, 41]. Our strategy can be understood as "Lift-Splat [9] without depth estimation", but as as illustrated in Figure 1, our implementation is different in a key detail: our method relies on *sampling* instead of *splatting*. Our method begins with 3D coordinates of the voxels, and takes a bilinear sample for each one. As a result of the camera projection, close-up rows of voxels sample very sparsely from the image (i.e., more spread out), and far-away rows of voxels sample very densely (i.e., more packed together), but each voxel receives a feature. *Splatting-based* methods [9, 13] begin with a 2D grid of coordinates, and "shoot" each pixel along its ray, filling voxels intersected by that ray, at fixed depth intervals. As a result, splatting methods yield multiple samples for up-close voxels, and very few samples (sometimes zero) for far-away voxels. As we will show in experiments, this implementation detail has an impact on performance, such that splatting is slightly superior at short distances, and sampling is slightly superior at long distances. In the experiments, we also evaluate a recently-proposed deformable attention strategy [5], which is similar to bilinear sampling but with learned sampling kernel for each voxel (i.e., learned weights and learned offsets).

Input resolution: While early BEV methods downsampled the RGB substantially before feeding it through the model (e.g., downsampling to 128×352 [9]), we note that recent works have been downsampling less and less (e.g., most recently using the full resolution [13, 5]). We believe this is an important factor for performance, and

so we train and test our RGB-only model across different input resolutions. Across variants of our model, we try resolutions from 112×208 up to 896×1600.

Batch size: Most of the related works on BEV segmentation use relatively small batch sizes in training (e.g., one [5] or four [9]). It has been reported in the image classification literature that higher batch sizes deliver superior results [42], but we have not seen batch size discussed as a performance factor in the BEV literature. This may be because of the high memory requirements of these BEV models: it is necessary to process all 6 camera images in parallel with a high-capacity module, and depending on implementation, it is sometimes necessary to store a 3D volume of features before reducing the representation to BEV. To overcome these memory issues, we accumulate gradients across multiple steps and multiple GPUs, and obtain arbitrarily-large effective batch sizes at the cost of slower wall-clock time per gradient step. For example, it takes us aprrox. 5 seconds to accumulate the forward and backward passes to create a batch size of 40, even using eight A100 GPUs in parallel. Overall, however, our training time is similar to prior work: our model converges in 1-3 days, depending on input resolution.

Augmentations: Prior work recommends the use of camera dropout and various image-based augmentations, but we have not seen these factors quantified. We experiment with multiple augmentations at training time, and measure their independent effects: (1) we apply random resizing and cropping on the RGB input, in a scale range of $[0.8, 1.2]$ (and update the intrinsics accordingly), (2) we randomly select a camera to be the "reference" camera, which randomizes the orientation of the 3D volume (as well as the orientation of the rasterized annotations), and (3) we randomly drop one of the six cameras. At test time, we use the "front" camera as the reference camera, and do not crop.

Radar usage details: Prior work has reported that the radar data in nuScenes is too sparse to be useful [3, 10], but we hypothesize it can be a valuable source of metric information, lacking from current camera-only setups. Beyond simply using radar or not, our model is flexible in terms of (1) using the radar meta-data as additional channels, vs. treating the radar as a binary occupancy image, (2) using the raw data from the sensor, vs. using outlier-filtered input, and (3) using radar accumulated from a number of sweeps, vs. only using the time-synchronized data.

IV. Experiments

Our experiments first aim to provide a unified study of the factors affecting BEV segmentation model performance, including high-interest details such as lifting strategies, and rarely-discussed details such as resolution and batch size. Second, we aim to quantify the utility of radar in this domain, and reveal the usage details which maximize performance. Finally, we compare against the state-of-the-art.

Dataset: We train and test all models in the nuScenes [33] urban scenes dataset, which is publicly available for non-commercial use. The dataset has 6 cameras, pointing front, front-left, front-right, back-left, back, and back-right, and 5 radar units, pointing front, left, right, back-left, and back-right, as well as a LiDAR unit. We use LiDAR inputs only for comparison, and focus on using RGB and RGB+radar. We use the official nuScenes training/validation split, which contains 28,130 samples in the training set, and 6,019 samples in the validation set. We follow the segmentation task setup from the official Lift-Splat [9] codebase, where any point within a "vehicle" bounding box is counted as a positive label, and all other locations are given negative labels. The "vehicle" superclass consists of bicycle, bus, car, construction vehicle, emergency vehicle, motorcycle, trailer, and truck. We evaluate using the intersection-over-union (IOU) metric.

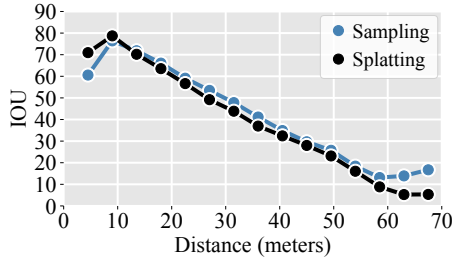

Fig. 2: Comparing IOU over distance for Lift-Splat-style splatting versus our bilinear sampling strategy, splatting is better at close range, while bilinear sampling is better at medium to long range.

A. Unified study of performance factors

Lifting strategy: Our model uses a bilinear sampling strategy to lift image features onto the BEV plane. Prior work has developed a variety of sophisticated alternate methods, but since various implementation details change across each work, it is unclear how much the lifting methods differ in performance. We present an apples-to-apples comparison in Table I, matching resolution, batch size, backbone, and augmentations across models. We see that bilinear sampling and deformable attention perform similarly, while the splatting methods fall behind. Multi-scale deformable attention (as in BEVFormer [5]) performs best, but at the cost of speed (1 day slower to train, 0.5 FPS slower at test time) and complexity (59M parameters instead of 42M, and requiring a custom CUDA kernel). We also note that uniform (unweighted) splatting is only approximately 1 point worse than depth-weighted splatting (consistent with results observed in FIERY [21]), suggesting that much of the scene structure is resolved after approximate BEV lifting. Fig. 2 shows a breakdown of IOU across distance for sampling vs. depth-weighted splatting, revealing that splatting provides

TABLE I: Effect of lifting strategy.

2D-to-BEV strategy	IOU
Unweighted splatting	43.1
Depth-based splatting [9]	44.4
Deformable attention [5]	46.5
Bilinear sampling (ours)	<u>47.4</u>
Multi-scale deform. attn. [5]	**48.9**

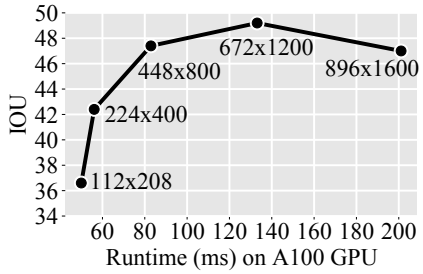

Fig. 3: Effect of input resolution.

an advantage at short distances, while sampling is better at long distances. We note that all results in our evaluation are higher than reported in the works introducing these methods, suggesting that other details in our training setup are having a substantial impact. As we show in the next subsections, the major factors here are input resolution and batch size.

Input resolution: We measure how our model's performance changes with input resolution, using the same resolution to train and test. Fig. 3 summarizes the results. Using resolutions lower than 448×800 drastically worsens performance. Our best result is **49.3** IOU at 672×1200. However, this model is substantially slower than the 47.4 IOU 448×800 model (133 ms vs 83 ms), and requires nearly twice the training time. Results drop at the highest resolution, perhaps because when the images are this large, the object scale is no longer consistent with the backbone's pre-training, leading to less-effective transfer. Performance at high resolution may improve with an alternate backbone architecture [43].

Batch size: In Fig. 4 we explore the impact of batch size on our model's performance: each increase in batch size gives an improvement in accuracy, with diminishing (but sizeable) returns. Increasing the batch size from 2 to 40 gives a nearly 14-point improvement in IOU. Considering that most prior works used batch sizes under 16, this suggests that many existing methods may benefit from simply re-training.

Backbones: Recent works have been using deeper and deeper backbones for the part of the model that creates feature maps from the input camera images (before the 2D-to-BEV lifting). Lift-Splat [9] used an EfficientNet-B0 [44], FIERY [21] used an EfficientNet-B4 [44], TIIM [13] used a ResNet-50 [34], and recently BEVFormer [5] used a ResNet-101 [34]. We explore these choices for our own model in Table II. We find, as expected, that a larger backbone gives better results, with ResNet-101 outperforming the rest. However, we note that the benefit of a specific backbone is

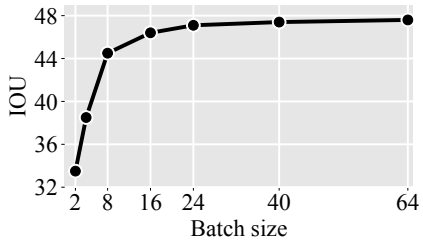

Fig. 4: Effect of batch size.

TABLE II: Effect of backbone.

Backbone	IOU
EfficientNet-B0	43.7
EfficientNet-B4	46.4
ResNet-50	46.6
ResNet-101	**47.4**

sometimes tied to the input resolution, which is fixed here at 448×800. Making the best use of the available 900×1600 data may require further exploration into architecture details.

Augmentations: When training our model, we randomly resize each camera's image to within $[0.8, 1.2]$ of the target resolution, and place it at a random offset from the center. Table IIIa shows that this augmentation gives a 1.6 point boost in IOU. When also randomize the camera selected to be the "reference" camera, which dictates the orientation of the 3D coordinate system. We show the results of this augmentation in Table IIIb. Randomizing the reference camera provides a 0.6 point boost in IOU. We believe that randomizing the reference camera helps reduce overfitting in the bird's eye view module. We have observed qualitatively that without this augmentation, the segmented cars have a slight bias for certain orientations in certain positions; with the augmentation added, this bias disappears. Prior work has reported a benefit from randomly dropping 1 of the 6 available cameras in each training sample [9]. As shown in Table IIIc, we find the opposite: using all cameras performs 1 point better. It may be that our reference-camera randomization provides enough regularization to make camera-dropout unnecessary. We have also experimented with photometric augmentations (blur, color, contrast), but found that they did not help.

B. Multi-modality fusion analysis

To analyze performance across modality combinations, we compare camera-only vs. camera plus radar vs. camera plus LiDAR, in Table IV. As hypothesized, radar indeed improves results in our setup. Relative to the camera-only model, radar improves results by 8 points, and LiDAR improves results by 13 points. High performance from LiDAR is consistent with related work in 3D object detection [1], but the gap between RGB+LiDAR and RGB+radar is smaller than might have been expected, since prior work conveyed negative results from RGB+radar fusion [10]. We show qualitative results in Fig. 5. We also visualize corresponding radar data. Qualitatively, the radar is indeed sparse and noisy as noted in related work [10, 3], but we believe it gives valuable hints about the metric scene structure, which, when

TABLE III: Study of augmentations.

Crop/resize	IOU
Off	45.8
On	**47.4**

(a) Effect of augmentations.

Ref. camera	IOU
"Front"	46.8
Random	**47.4**

(b) Effect of camera randomization.

# Cameras	IOU
5/6	46.4
6/6	**47.4**

(c) Effect of camera dropout.

TABLE IV: Effect of multi-modality fusion.

Modality	IOU
Camera	47.4
Camera + radar	55.7
Camera + LiDAR	**60.8**

fused with information acquired from RGB, enables higher-accuracy semantic segmentation in the bird's eye view. We next investigate the radar performance factors in more detail.

As shown in Table Va, our model benefits from accessing the meta-data associated with each radar point. This includes information such as velocity, which may help distinguish moving objects from the background. Removing this aspect lowers IOU by 0.7 points. As shown in Table Vb, our model benefits from having *all* radar returns as input, achieved by disabling nuScenes' built-in outlier filtering strategy. The filtering strategy attempts to discard outlier points (produced by multipath interference and other issues), but potentially discards some true returns as well. Using the filtered data instead of the raw data results in a 2 point drop in performance. As shown in Table Vc, our model benefits from aggregating multiple sweeps of radar as input. This means using radar from timesteps $(t, t-1, t-2)$ aligned to the coordinate frame of timestep t, rather than exclusively using the data from timestep t. Using a single sweep lowers performance by 2.4 points, likely because the model struggles with the extreme sparsity of the signal.

C. Comparison with state-of-the-art

In this final subsection, we compare against the published state-of-the-art. Considering that our previous experiments show the importance of various subtle training details (which vary across each prior work), the comparisons in this section should not be read as an apples-to-apples comparison, but rather as reflecting a combination of factors in the training setups. Besides the factors already considered, we note that some models train for additional categories, which may either improve *or* worsen results [5]. In Table VI, we see that our RGB model slightly outperforms all other RGB-based and temporal models, while the RGB+radar model holds a margin of 9 points above the rest. Our model has 42M parameters, which is fairly efficient compared to BEVFormer's 68.7M. Most of our parameters (37M) come from the ResNet-101, and this is also the main speed bottleneck. Our model is also 3 times faster than BEVFormer in this setup (7.3 FPS vs 2.3 FPS on a V100 GPU), largely due to using a lower RGB resolution.

TABLE V: Study of radar hyperparameters.

Input	IOU
Occ. only	55.0
Full return	**55.7**

(a) Using meta-data associated with points helps.

Filtering	IOU
On	53.7
Off	**55.7**

(b) Disabling nuScenes' outlier-filtering helps.

# Sweeps	IOU
1	53.1
3	**55.7**

(c) Aggregating multiple radar sweeps helps.

TABLE VI: Comparison against state-of-the-art methods for vehicle segmentation IOU in the nuScenes validation set.

Method	Lifting	Batch	Resolution	Inputs	IOU
FISHING [10]	MLP	-	192 × 320	RGB	30.0
				LiDAR	(44.3)
Lift, Splat [9]	Depth splat	4	128 × 352	RGB	32.1
				RGB+LiDAR	(44.5)
FIERY [21]	Depth splat	12	224 × 480	RGB	35.8
				RGB+time	38.2
TIIM [13]	Ray attn.	8	900 × 1600	RGB	38.9
				RGB+time	41.3
CVT [41]	Attention	16	224 × 448	RGB	36.0
BEVFormer [5]	Def. Attn.	1	900 × 1600	RGB	44.4
				RGB+time	46.7
Ours	Bilinear	40	448 × 800	RGB	47.4
				RGB+radar	**55.7**
				RGB+LiDAR	(**60.8**)

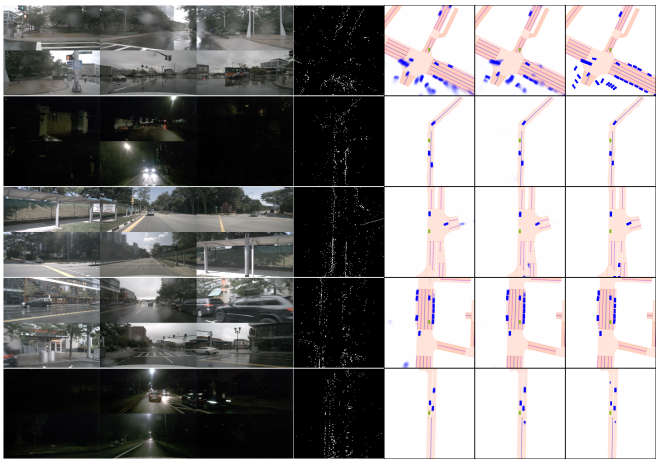

Fig. 5: Left to right: camera data, radar data, our RGB model output, our RGB+radar model output, and ground truth.

V. Discussion and Conclusion

In this work, we explore design and training choices for BEV semantic parsing, and show that batch size and image resolution play a surprisingly large role in performance, which has not been previously discussed in the literature. While recent work has developed increasingly sophisticated 2D-to-BEV lifting strategies, we show that bilinear sampling performs well also. Temporal integration is a natural fit in this setting, but we leave it for future work. Making best use of the available high-resolution images will require a more thorough exploration of backbones. Another area for future work is to explore 3D object detection, in addition to (or instead of) the dense BEV representation. Our experiments demonstrate that radar provides useful information for BEV parsing, and we hope that this insight will be applied to other approaches. Our work does not argue for the use of particular sensors over others, but rather for the use of metric information whenever available, even if sparse and noisy.

REFERENCES

[1] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3D object detection and tracking," in *CVPR*, 2021.

[2] T. Yin, X. Zhou, and P. Krähenbühl, "Multimodal virtual point 3D detection," *NeurIPS*, vol. 34, 2021.

[3] P. Li, P. Wang, K. Berntorp, and H. Liu, "Exploiting temporal relations on radar perception for autonomous driving," *arXiv:2204.01184*, 2022.

[4] B. Pan, J. Sun, H. Y. T. Leung, A. Andonian, and B. Zhou, "Cross-view semantic segmentation for sensing surroundings," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4867–4873, 2020.

[5] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Y. Qiao, and J. Dai, "BEVFormer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers," *ECCV*, 2022.

[6] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *ICIP*. IEEE, 2016, pp. 3464–3468.

[7] Y. B. Can, A. Liniger, O. Unal, D. Paudel, and L. Van Gool, "Understanding bird's-eye view of road semantics using an onboard camera," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3302–3309, 2022.

[8] S. Schulter, M. Zhai, N. Jacobs, and M. Chandraker, "Learning to look around objects for top-view representations of outdoor scenes," in *ECCV*, 2018, pp. 787–802.

[9] J. Philion and S. Fidler, "Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3D," in *ECCV*. Springer, 2020, pp. 194–210.

[10] N. Hendy, C. Sloan, F. Tian, P. Duan, N. Charchut, Y. Xie, C. Wang, and J. Philbin, "Fishing net: Future inference of semantic heatmaps in grids," *arXiv:2006.09917*, 2020.

[11] A. Saha, O. Mendez, C. Russell, and R. Bowden, "Enabling spatio-temporal aggregation in birds-eye-view vehicle estimation," in *ICRA*. IEEE, 2021, pp. 5133–5139.

[12] N. Gosala and A. Valada, "Bird's-eye-view panoptic segmentation using monocular frontal view images," *IEEE Robotics and Automation Letters*, 2022.

[13] A. Saha, O. Mendez, C. Russell, and R. Bowden, "Translating images into maps," in *ICRA*, 2022.

[14] M. Parker, "Chapter 20 – automotive radar," in *Digital Signal Processing 101*, 2nd ed., M. Parker, Ed. Newnes, 2017, pp. 253–276.

[15] R. Cheng, Z. Wang, and K. Fragkiadaki, "Geometry-aware recurrent neural networks for active visual recognition," in *NeurIPS*, 2018.

[16] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer, "DeepVoxels: Learning persistent 3D feature embeddings," in *CVPR*, 2019, pp. 2437–2446.

[17] H.-Y. F. Tung, R. Cheng, and K. Fragkiadaki, "Learning spatial common sense with geometry-aware recurrent networks," *CVPR*, 2019.

[18] A. W. Harley, S. K. Lakshmikanth, F. Li, X. Zhou, H.-Y. F. Tung, and K. Fragkiadaki, "Learning from unlabelled videos using contrastive predictive neural 3D mapping," in *ICLR*, 2020.

[19] B. Liu, B. Zhuang, S. Schulter, P. Ji, and M. Chandraker, "Understanding road layout from videos as a whole," in *CVPR*, 2020, pp. 4414–4423.

[20] C. Reading, A. Harakeh, J. Chae, and S. L. Waslander, "Categorical depth distribution network for monocular 3D object detection," in *CVPR*, 2021.

[21] A. Hu, Z. Murez, N. Mohan, S. Dudas, J. Hawke, V. Badrinarayanan, R. Cipolla, and A. Kendall, "FIERY: Future instance prediction in bird's-eye view from surround monocular cameras," in *CVPR*, 2021, pp. 15 273–15 282.

[22] H. Wang, P. Cai, Y. Sun, L. Wang, and M. Liu, "Learning interpretable end-to-end vision-based motion planning for autonomous driving with optical flow distillation," in *2021 ICRA (ICRA)*. IEEE, 2021, pp. 13 731–13 737.

[23] T.-Y. Lim, A. Ansari, B. Major, D. Fontijne, M. Hamilton, R. Gowaikar, and S. Subramanian, "Radar and camera early fusion for vehicle detection in advanced driver assistance systems," in *Machine Learning for Autonomous Driving Workshop at NeurIPS*, vol. 2, 2019, p. 7.

[24] B. Liu, B. Zhuang, and M. Chandraker, "Weakly but deeply supervised occlusion-reasoned parametric layouts," *arXiv:2104.06730*, 2021.

[25] Q. Li, Y. Wang, Y. Wang, and H. Zhao, "HDMapNet: A local semantic map learning and evaluation framework," *arXiv:2107.06307*, 2021.

[26] T. Roddick and R. Cipolla, "Predicting semantic map representations from images using pyramid occupancy networks," in *CVPR*, 2020, pp. 11 138–11 147.

[27] W. Yang, Q. Li, W. Liu, Y. Yu, Y. Ma, S. He, and J. Pan, "Projecting your view attentively: Monocular road scene layout estimation via cross-view transformation," in *CVPR*, 2021, pp. 15 536–15 545.

[28] J. Lombacher, K. Laudt, M. Hahn, J. Dickmann, and C. Wöhler, "Semantic radar grids," in *Intelligent vehicles symposium (IV)*. IEEE, 2017, pp. 1170–1175.

[29] L. Sless, B. El Shlomo, G. Cohen, and S. Oron, "Road scene understanding by occupancy grid learning from sparse radar clusters using semantic segmentation," in *CVPR Workshops*, 2019, pp. 0–0.

[30] M. Meyer and G. Kuschk, "Deep learning based 3D object detection for automotive radar and camera," in *European Radar Conference (EuRAD)*. IEEE, 2019, pp. 133–136.

[31] B. Yang, R. Guo, M. Liang, S. Casas, and R. Urtasun, "Radarnet: Exploiting radar for robust perception of dynamic objects," in *ECCV*. Springer, 2020, pp. 496–512.

[32] O. Schumann, M. Hahn, J. Dickmann, and C. Wöhler, "Semantic segmentation on radar point clouds," in *International Conference on Information Fusion (FUSION)*. IEEE, 2018, pp. 2179–2186.

[33] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A multimodal dataset for autonomous driving," *arXiv:1903.11027*, 2019.

[34] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016, pp. 770–778.

[35] V. L. Dmitry Ulyanov, Andrea Vedaldi, "Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis," in *CVPR*, 2017.

[36] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *CVPR*, 2018, pp. 7482–7491.

[37] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *ECCV*, 2020.

[38] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *ECCV*, D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, Eds., 2014.

[39] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv:1711.05101*, 2017.

[40] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019, p. 1100612.

[41] B. Zhou and P. Krähenbühl, "Cross-view transformers for real-time map-view semantic segmentation," in *CVPR*, 2022, pp. 13 760–13 769.

[42] S. L. Smith, P.-J. Kindermans, C. Ying, and Q. V. Le, "Don't decay the learning rate, increase the batch size," *arXiv:1711.00489*, 2017.

[43] D. Park, R. Ambrus, V. Guizilini, J. Li, and A. Gaidon, "Is pseudo-lidar needed for monocular 3D object detection?" in *CVPR*, 2021, pp. 3142–3152.

[44] M. Tan and Q. Le, "EfficientNet: Rethinking model scaling for convolutional neural networks," in *ICML*. PMLR, 2019, pp. 6105–6114.