A Simple Baseline for BEV Perception Without LiDAR

Adam W. Harley
Carnegie Mellon University
aharley@cs.cmu.edu

Zhaoyuan Fang
Carnegie Mellon University
zhaoyuaf@cs.cmu.edu

Jie Li
Toyota Research Institute
jie.li@tri.global

Rares Ambrus
Toyota Research Institute
rares.ambrus@tri.global

Katerina Fragkiadaki
Carnegie Mellon University
katef@cs.cmu.edu

Abstract: Building 3D perception systems for autonomous vehicles that do not rely on LiDAR is a critical research problem because of the high expense of LiDAR systems compared to cameras and other sensors. Current methods use multi-view RGB data collected from cameras around the vehicle and neurally "lift" features from the perspective images to the 2D ground plane, yielding a "bird's eye view" (BEV) feature representation of the 3D space around the vehicle. Recent research focuses on the way the features are lifted from images to the BEV plane. We instead propose a simple baseline model, where the "lifting" step simply averages features from all projected image locations, and find that it outperforms the current state-of-the-art in BEV vehicle segmentation. Our ablations show that batch size, data augmentation, and input resolution play a large part in performance. Additionally, we reconsider the utility of radar input, which has previously been either ignored or found non-helpful by recent works. With a simple RGB-radar fusion module, we obtain a sizable boost in performance, approaching the accuracy of a LiDAR-enabled system.[1]

1 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 43.2 [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, 7]. 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, prior work has explored using homographies to warp features directly onto the ground plane [8], using depth estimates to place features at their approximate 3D locations [9, 10], using MLPs with various geometric biases [11, 12, 13], and most recently, using geometry-aware transformers [14] and deformable attention across space and time [5]. We instead propose a simple baseline 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. Surprisingly, our simple baseline exceeds the performance of state-of-the-art models, while also being faster and having fewer parameters. Our ablations show that batch size, data augmentation, and input resolution play a large part in performance.

As we establish a new simple baseline, we also take the opportunity to question the currently-popular paradigm of relying on cameras *alone*, instead of fusing freely-available metric information from

[1] Project page: http://cs.cmu.edu/~aharley/bev

radar. Radar sensors are not only cheap compared to LiDAR, but have been integrated in real vehicles for several years already [15]. While using RGB cameras alone may give the task a certain purity (requiring metric 3D estimates from 2D input alone), it does not reflect the reality of autonomous driving. 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, 11]. 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, obtaining a score only 5 points behind a LiDAR-enabled system.

This paper does not a contribute new innovative architecture, but instead (1) provides a simple baseline for BEV semantic segmentation, with state-of-the-art results and extensive ablative analysis, and (2) invites the community to reconsider the utility of radar in autonomous driving perception systems. We also release code and reproducible models to facilitate future research in the area.

2 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 [16, 17, 18], uses the camera geometry to define a mapping between voxels and their projected coordinates, and collects features by bilinearly sampling at the projected coordinates. This places each image feature into multiple 3D coordinates, essentially tiling the feature along the ray's extent in the volume. This method of lifting is not typically used in bird's eye view semantic tasks.

Depth-based unprojection Several works estimate per-pixel depth with a monocular depth estimator, either pre-trained for depth estimation [9, 19, 20] or trained simply for the end-task [10, 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 "truck" features at the front visible surface of the truck, rather than fill the entire truck 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, 8], transferring the features from one plane to another. This operation tends to produce poor results when the scene itself is non-planar (e.g., tall objects get spread out over a wide area).

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, 11, 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, 12], or to different categories [13]. 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. [14] begin by defining a relationship between each vertical scan-line of an image, and the ground-plane line that it projects to, and show that transformer self-attentions can learn an effective "translation" function between the two coordinate systems. Defining this transformer at the line level helps provide inductive bias to the model, since the lifting task should be similar across 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 the parameter-free unprojection, but with approximately $10\times$ more samples, learnable offsets for the sampling coordinates, and a learnable kernel on their combination.

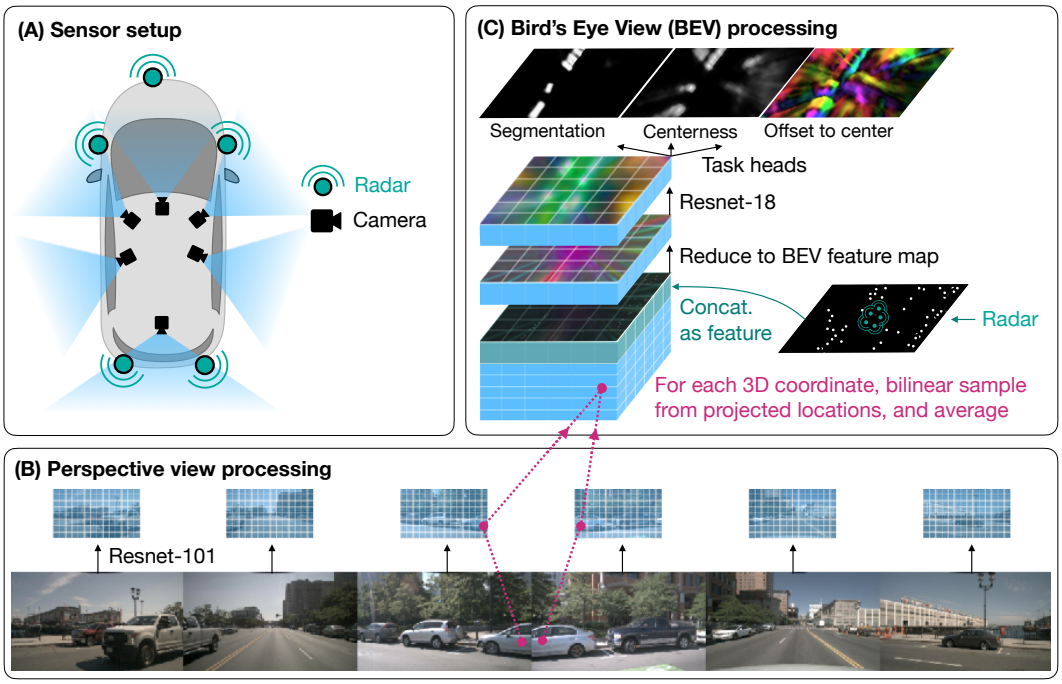

Figure 1: **A baseline for neural BEV mapping**. (A) Our sensor setup consists of multiple cameras and radar units. (B) We begin by featurizing each camera image with a ResNet-101. (C) We define a set of 3D coordinates around the ego-vehicle, project these coordinates into all images, and bilinearly sample features at the projected locations, yielding a 3D volume of features. We then concatenate a rasterized radar image, and reduce the vertical dimension of the volume to yield a BEV feature map. We process this BEV map with a Resnet-18. Finally, task heads produce semantic segmentation, a centerness heatmap that indicates which BEV locations correspond to object centers, and an offset field that indicates the closest object centroid from each BEV location.

Radar In the automotive industry, radar has been in use for several years already [15]. 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, 11]. Some recent works explore RGB-radar or RGB-Lidar fusion strategies [23, 30], similar to our work, but targeting detection and velocity estimation rather than BEV semantic labelling.

3 Neural BEV Mapping Baseline

3.1 Setup and overview

The setup and architecture of our model are shown in Figure 1. Our model uses as input a flexible number of cameras, any number of radar units, and optionally even LiDAR. We assume that the data is synchronized across sensors. We assume that the intrinsics and extrinsics of each sensor 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.

Our model first computes features from each camera image, as shown in Figure 1-B. Then, we populate our 3D volume with features, with bilinear sampling: using the 3D coordinate at the center of each discrete cell, we use the extrinsics and intrinsics to compute its 2D coordinate in each 2D feature map, and bilinearly sample to obtain a feature. If radar is provided, we rasterize its returns into a bird's eye view image, and concatenate this with the 3D feature volume. We then reduce the vertical dimension, and process the resulting feature map with a 2D convolutional net. Finally, task heads produce quantities of interest, such as segmentation, centerness scores, and an offset map, all in a 2D bird's eye view. The segmentation map contains a categorical distribution over semantic categories. The centerness map indicates probabilities of a grid cell being the center of an object. The offset map contains a vector field, where each vector points to the nearest object center. The 3D/BEV processing is illustrated in Figure 1-C.

3.2 Architecture

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 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 our pre-defined volume of 3D coordinates into all feature maps, and bilinearly sample features 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 emphasize that our lifting step is parameter-free. We rearrange the 3D feature volume dimensions, 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.

We next rasterize the radar information, to create another BEV feature map. We may 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 (only for comparison).

We then concatenate the RGB features and radar 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 task-specific heads. Each head is two convolution layers with instance normalization, and ReLU after the norm in the first layer.

The model is trained on three tasks and is equipped with three corresponding task heads: segmentation, centerness, and offset. The segmentation head produces per-pixel vehicle/background segmentation. The centerness head produces a heatmap where high values indicate high probability that the grid cell is an object center. 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 compute the ground truth from 3D box annotations. We use an uncertainty-based learnable weighting [36] to balance the three losses.

Compared to related models, the architecture choices for the perspective-view and bird's-eye-view encoders are most similar to those in Lift-Splat [10] and FIERY [21]. Our multi-task setup is the same as FIERY [21].

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 [10], MLPs [4, 11, 25], or attention mechanisms [14, 5]. Besides the implementation simplicity, our model is also faster and has fewer parameters than the next-best existing model, as we will detail in the experiments.

3.3 Implementation details

Our model uses an RGB input resolution of 448×960. The ResNet-101 has a total stride of 8. We use a feature dimension (i.e., channel dimension) of 128. Our 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). The Resnet-101 is pre-trained for object detection [37] on COCO 2017 [38]. The BEV Resnet-18 is trained from scratch.

At training time, 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). We apply random cropping on the RGB input, by resizing the original images to 558×992, and taking a random 448×960 crop inside (and updating the intrinsics accordingly). At test time, we use the "front" camera as the reference camera, and take a center crop.

We train end-to-end for 40,000 iterations, with a batch size of 40, with the Adam-W optimizer [39] at a constant learning rate of 3e-4. We accumulate gradients across eight V100 GPUs and five iterations, to obtain an effective batch size of 40 for each gradient step. The model takes 2-3 days to train.

4 Experiments

We train and test our model 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. The sensor setup is illustrated in Figure 1-A. 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 use annotations from the "vehicle" superclass, which includes bicycle, bus, car, construction vehicle, emergency vehicle, motorcycle, trailer, and truck. We evaluate on vehicle/background segmentation, using the intersection-over-union (IOU) metric in a bird's eye view. (We follow related work [21] in treating centerness and offset purely as auxiliary tasks.)

4.1 Main results

In this section we present our BEV vehicle segmentation results on the nuScenes validation set, and compare with the state-of-the-art.

We first compare against single-frame RGB-only models, in Table 1. Our RGB-only method obtains 47.0 IOU, outperforming all other single-frame RGB-only models. Second-best is BEVFormer [5] (concurrent work) at 43.2. The main difference between these two methods is that BEVFormer uses a deformable attention-based strategy to lift features from 2D to BEV, in place of our bilinear sampling.

Method	IOU
FISHING [11]	30.0
PON [26]	31.4
Lift, Splat [10]	32.1
FIERY [21]	35.8
Translating Images Into Maps [14]	38.9
BEVFormer [5]	43.2
Ours	**47.0**

Table 1: State-of-the-art comparisons against *single-frame RGB-only* methods for vehicle segmentation IOU in the nuScenes validation set.

We next open the comparison to all methods of all modalities, in Table 2. The best model we are aware of is BEVFormer's temporal variant, which obtains 46.7 IOU—just under the accuracy of our single-frame RGB-only model. Our RGB+radar model improves over this by 9 points, reaching 55.7 IOU. For reference, we also compute our model's performance using RGB+LiDAR, obtaining a new high

of 60.8. 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 [11]. Note that integrating over time is orthogonal to the strengths of our model, so it should be possible to push results even higher using time.

Method	Inputs	IOU
FISHING [11]	RGB	30.0
	LiDAR	(44.3)
FIERY [21]	RGB	35.8
	RGB+time	38.2
Translating Images Into Maps [14]	RGB	38.9
	RGB+time	41.3
BEVFormer [5]	RGB	43.2
	RGB+time	46.7
Ours	RGB	47.0
	RGB+radar	**55.7**
	RGB+LiDAR	(**60.8**)

Table 2: State-of-the-art comparisons against best known methods for vehicle segmentation IOU in the nuScenes validation set. Since our focus is on LiDAR-free methods, we put LiDAR-enhanced results in parentheses, including them only for reference.

Speed and complexity Our model runs at 7.3 FPS on a V100 GPU. This is more than $3\times$ faster than BEVFormer [5], which runs at 2.3 FPS. Our model also has fewer parameters: 47.2M, compared to BEVFormer's 68.7M. Most of our parameters (44.5M) come from the Resnet-101, and this is also the main speed bottleneck, due to the high RGB resolution.

Qualitative results We show qualitative results in Figure 2. We also visualize corresponding LiDAR and radar data, to show the scene structure as captured by those sensors. Qualitatively, the radar is indeed sparse and noisy as noted in related work [11, 3], but we believe it gives valuable hints about the metric scene structure, which, when fused with information acquired from RGB, enables higher-accuracy semantic segmentation in the bird's eye view.

4.2 Ablation studies

Considering the simplicity of our method compared to prior work, we next aim to answer the question: what really matters for performance? While prior work has focused on the details of the 2D-to-BEV lifting strategy, our 2D-to-BEV step is parameter-free, so we study other factors: input resolution, batch size, and augmentations. We first perform ablations using our RGB-only model, and then turn to evaluating the details of our radar processing. Each ablation involves re-training the model with a single specific difference with respect to the proposed model. In all ablations we report vehicle segmentation IOU in the nuScenes validation set (where the proposed model achieves 47.0).

Input resolution The nuScenes dataset provides high-resolution RGB images, which are 900×1600. While earlier work downsampled the RGB substantially before feeding it through the model (e.g., downsampling to 128×352 [10]), we note that recent works have been downsampling less and less (e.g., most recently using the full resolution [5]). We believe this is an important factor for performance, and so we train and test our RGB-only model across different input resolutions. Table 3 summarizes the results. We find that our model obtains its best result at 448×960, which

RGB resolution	IOU
112×240	36.3
224×480	42.7
448×960	**47.0**
672×1440	44.0

Table 3: Effect of input resolution.

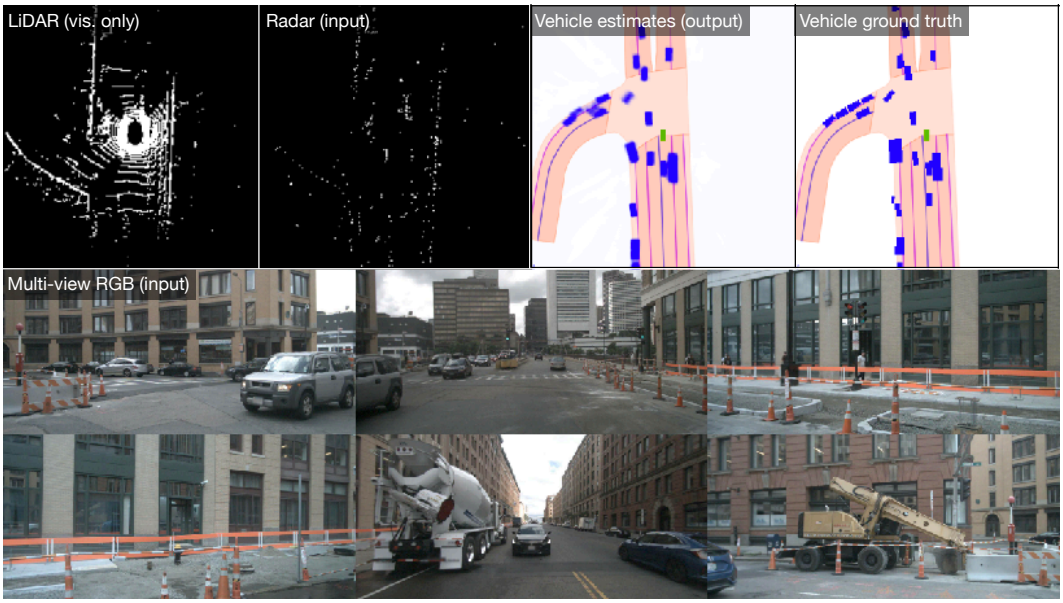

Figure 2: Visualization of our best model's inputs (RGB and radar) and outputs (BEV vehicle segmentation) overlaid on a road visualization. We display LiDAR and ground truth for comparison. Please see the supplementary file for a video visualization.

is approximately half of the total available resolution. It may be that when the images are too large or too small, the typical object scale is no longer consistent with the backbone's pre-training (on COCO [38]), leading to less-effective transfer.

Batch size It has been reported in the image classification literature that higher batch sizes deliver superior results [40], but we have not seen batch size discussed in BEV literature. In Table 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 10-point improvement in IOU. It appears that increasing the batch size further might improve performance still, but this is beyond our current compute capacity.

Batch size	IOU
2	36.8
4	41.8
20	43.7
40	**47.0**

Table 4: Effect of batch size.

Augmentations When training our model, we randomize the camera selected to be the "reference" camera, which dictates the orientation of the 3D coordinate system. To the best of our knowledge ours is the first work to do this. We show the results of this augmentation in Table 5. Randomizing the reference camera provides approximately a 1 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 [10]. Interestingly, we find the opposite: using all cameras performs better. It may be that our reference-camera randomization provides enough regularization to make camera-dropout unnecessary.

We have also experimented with color, contrast, and blur augmentations, and found that these worsened results. Such augmentations are known to be beneficial in image classification, but it may be that the model benefits from sensitivity to these factors in the current data.

Reference camera	IOU
"Front"	46.0
Random	**47.0**

Table 5: Effect of randomizing the reference camera.

Number of cameras	IOU
5/6	45.9
6/6	**47.0**

Table 6: Effect of camera dropout.

Radar usage details Since ours is the first model to report strong results from RGB+radar fusion in this domain, we aim to reveal the important hyperparameter choices in the radar setup.

As shown in Table 7a, 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 2.5 points.

As shown in Table 7b, 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 2.3 point drop in performance.

As shown in Table 7c, 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.

Input	IOU
Full return	**55.7**
Occupancy only	53.2

(a) Using the meta-data associated with each point improves performance.

Filtering	IOU
Off	**55.7**
On	53.4

(b) Turning off nuScenes' outlier-filtering strategy improves performance.

# Sweeps	IOU
3	**55.7**
1	53.3

(c) Aggregating radar data across multiple sweeps improves performance.

Table 7: Ablation study on radar hyperparameters.

Discussion and limitations In this work, we propose a simple baseline architecture for BEV semantic parsing, and show its surprising effectiveness. Our work highlights that radar is useful sparse metric information for BEV parsing, and this insight can be applied to other approaches. Similarly, our training techniques may lead to improvements for other models. We did not discuss any temporal integration strategies. Temporal integration is very natural to include in this setting, and previous works have shown it gives a performance boost of 3-4 points. We leave this for future work. Finally we note that training these models is costly in terms of both GPU time and carbon footprint [41]. We aim to release our models and encourage their re-use.

5 Conclusion

LiDAR-free BEV perception is a critical step toward low-cost autonomous vehicles. This paper proposes a simple baseline model with a parameter-free 2D-to-BEV lifting step that outperforms the state-of-the-art. We then reconsider the assumption that radar data is too sparse to be useful, and propose a simple strategy to fuse its noisy returns with the 3D representation acquired from RGB. The resulting radar-enhanced model exceeds the performance of all published models, including ones that integrate information across time, and is only 5 points behind the LiDAR-enhanced model in BEV vehicle segmentation. We will make our code publicly available. We hope that this simple model will serve as a useful baseline in the future.

References

[1] T. Yin, X. Zhou, and P. Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021.

[2] T. Yin, X. Zhou, and P. Krähenbühl. Multimodal virtual point 3d detection. *Advances in Neural Information Processing Systems*, 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*, 5(3):4867–4873, 2020.

[5] Z. Li, W. Wang, H. Li, E. Xie, C. Sima, T. Lu, Q. Yu, and J. Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv preprint arXiv:2203.17270*, 2022.

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

[7] X. Weng, J. Wang, D. Held, and K. Kitani. 3d multi-object tracking: A baseline and new evaluation metrics. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 10359–10366. IEEE, 2020.

[8] 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*, 7(2): 3302–3309, 2022.

[9] S. Schulter, M. Zhai, N. Jacobs, and M. Chandraker. Learning to look around objects for top-view representations of outdoor scenes. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 787–802, 2018.

[10] J. Philion and S. Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *European Conference on Computer Vision*, pages 194–210. Springer, 2020.

[11] 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 preprint arXiv:2006.09917*, 2020.

[12] A. Saha, O. Mendez, C. Russell, and R. Bowden. Enabling spatio-temporal aggregation in birds-eye-view vehicle estimation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5133–5139. IEEE, 2021.

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

[14] A. Saha, O. M. Maldonado, C. Russell, and R. Bowden. Translating images into maps. *arXiv preprint arXiv:2110.00966*, 2021.

[15] M. Parker. Chapter 20 – automotive radar. In M. Parker, editor, *Digital Signal Processing 101 (Second Edition)*, pages 253–276. Newnes, second edition edition, 2017. ISBN 978-0-12-811453-7. doi:https://doi.org/10.1016/B978-0-12-811453-7.00020-2. URL https://www.sciencedirect.com/science/article/pii/B9780128114537000202.

[16] R. Cheng, Z. Wang, and K. Fragkiadaki. Geometry-aware recurrent neural networks for active visual recognition. In *NIPS*, 2018.

[17] V. Sitzmann, J. Thies, F. Heide, M. Nießner, G. Wetzstein, and M. Zollhofer. Deepvoxels: Learning persistent 3d feature embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2437–2446, 2019.

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

[19] B. Liu, B. Zhuang, S. Schulter, P. Ji, and M. Chandraker. Understanding road layout from videos as a whole. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4414–4423, 2020.

[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 *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15273–15282, 2021.

[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 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13731–13737. IEEE, 2021.

[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 the 33rd Conference on Neural Information Processing Systems*, volume 2, page 7, 2019.

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

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

[26] T. Roddick and R. Cipolla. Predicting semantic map representations from images using pyramid occupancy networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11138–11147, 2020.

[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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15536–15545, 2021.

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

[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 *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

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

[31] B. Yang, R. Guo, M. Liang, S. Casas, and R. Urtasun. Radarnet: Exploiting radar for robust perception of dynamic objects. In *European Conference on Computer Vision*, pages 496–512. Springer, 2020.

[32] O. Schumann, M. Hahn, J. Dickmann, and C. Wöhler. Semantic segmentation on radar point clouds. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2179–2186. IEEE, 2018.

[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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[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 *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7482–7491, 2018.

[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 D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *ECCV*, 2014.

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

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

[41] E. Strubell, A. Ganesh, and A. McCallum. Energy and policy considerations for deep learning in nlp. *arXiv preprint arXiv:1906.02243*, 2019.