PointBeV: A Sparse Approach to BeV Predictions

Loick Chambon[1,2], Éloi Zablocki[1], Mickaël Chen[1], Florent Bartoccioni[1], Patrick Pérez[*3], Matthieu Cord[1,2]

[1] Valeo.ai, Paris, France [2] Sorbonne Université, Paris, France [3] Kyutai, Paris, France

Abstract

Bird's-eye View (BeV) representations have emerged as the de-facto shared space in driving applications, offering a unified space for sensor data fusion and supporting various downstream tasks. However, conventional models use grids with fixed resolution and range and face computational inefficiencies due to the uniform allocation of resources across all cells. To address this, we propose PointBeV, a novel sparse BeV segmentation model operating on sparse BeV cells instead of dense grids. This approach offers precise control over memory usage, enabling the use of long temporal contexts and accommodating memory-constrained platforms. PointBeV employs an efficient two-pass strategy for training, enabling focused computation on regions of interest. At inference time, it can be used with various memory/performance trade-offs and flexibly adjusts to new specific use cases. PointBeV achieves state-of-the-art results on the nuScenes dataset for vehicle, pedestrian, and lane segmentation, showcasing superior performance in static and temporal settings despite being trained solely with sparse signals. We release our code with two new efficient modules used in the architecture: Sparse Feature Pulling, designed for the effective extraction of features from images to BeV, and Submanifold Attention, which enables efficient temporal modeling. The code is available at https://github.com/valeoai/PointBeV.

1. Introduction

Bird's-eye View (BeV) representations are now ubiquitously in driving applications. Indeed, a top-view egocentric grid is not only a convenient shared space for fusing inputs from multiple sensors [28, 39, 42, 49], but also a space-aware representation relevant for many downstream tasks such as detection [31, 34], segmentation [15, 39], forecasting [11], tracking [48], or planning [8, 18, 40]. BeV segmentation encompasses a broad family of tasks such as 2D instance segmentation [15], 3D instance segmentation [20, 21, 54], occupancy forecasting [22, 35] and online mapping [27]. In this paper, we focus on BeV segmentation from multiple cameras, in scenarios with or without past frames, respectively referred to as *temporal* and *static*.

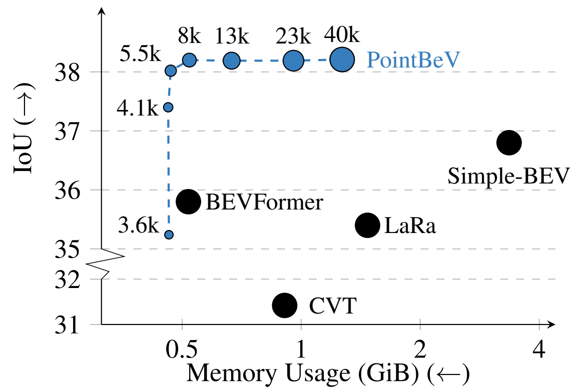

Figure 1. **BeV vehicle IoU vs. memory footprint on nuScenes [3] validation set.** Models are evaluated without visibility filtering (i.e all annotated vehicles are considered) at resolution 224×480. The memory consumption is calculated using a 40GB A100 GPU. The size of a dot represents the number of BeV points being evaluated, the smaller the better. PointBeV has the capacity to explore various trade-offs between efficiency and performance by varying the number of points being considered. The remaining points are considered as zeros in the final prediction. Using PointBeV we can achieve state-of-the-art performance with only a small portion of the points and without losing performance.

BeV representations are usually implemented using grids of fixed resolution and range [9, 13, 15, 26, 31]. This limits their efficiency in terms of compute, even more clearly when considering temporal tasks, where aggregating past frames for long horizons can be a very costly endeavor. Departing from these dense BeV grid approaches, we present in this paper PointBeV, a camera-based BeV segmentation model that operates on sparse BeV features. Our approach offers control over the model's memory usage by restricting the number of points considered, and enables adaptive focus on specific regions of interest. Notably, we develop two modules for efficient sparse operations: the *Sparse Feature Pulling* module, which retrieves features from multiple cameras using sparse coordinates, and the *Submanifold Attention* module, adapted from sub-

*Work done at Valeo.ai.

manifold attention [24], to handle our temporal aggregation strategy. At test time, PointBeV operates in a low-compute regime without retraining and possesses the flexibility to adjust computation based on varying use cases.

Despite being trained solely with sparse signals, PointBeV attains state-of-the-art results in classic BeV segmentation tasks — vehicle, pedestrian, and lane segmentation — on the nuScenes dataset [3], both in static and temporal settings. The performance and flexibility of PointBeV are demonstrated in Fig. 1.

Our main contributions are as follows:

- We introduce a sparse BeV paradigm for BeV segmentation task, allowing to operate in memory-constrained settings without performance drop.
- We introduce two sparse modules: a *Sparse Feature Pulling* module for efficient features extraction from images to BeV and, in temporal settings, *Submanifold Attention* for efficient temporal aggregation allowing to process an extended number of past frames.
- We develop specific training, inference for this sparse BeV paradigm. The sparse aspect of PointBeV broadens its utility. Without retraining, it can balance at test-time efficiency and precision, or exploit at additional information like LiDAR point clouds and HD maps.
- Our PointBeV reaches state-of-the-art results for vehicle, pedestrian and lane segmentation under different visibility settings and at different image resolutions.

2. Related Work

Vision-based BeV Segmentation. BeV offers a convenient space for aggregating multiple cameras [39] and as such has become one of the main representations for autonomous driving [4, 5, 18, 35, 46]. Nevertheless, changing perspective from camera images to build BeV maps is challenging, involving depth estimation and 3D geometry [28].

One approach learns the projection entirely from data, typically using a learnable-query-based cross-attention mechanism. Typically, geometric embeddings such as frustum 3D coordinates [32, 33, 45] or ray vectors [2, 37, 52], are injected to enrich image features with 3D knowledge.

Nonetheless, these methods lack interpretability as the mapping from perspective view to BeV is only implicit. Instead, some methods include a differentiable geometric projection in their approach. In the seminal Lift-Splat-Shoot (LSS) [39] work, a depth probability is estimated for each feature of the image, allowing to 'lift' each feature to form a 3D voxel grid. Then, the per-camera voxels are simply flattened ('splat') onto a common BeV grid. However, for this pipeline to work, precise depth estimation is essential to accurately project image features to their corresponding 3D location [41], and subsequent works focused on improving this particular aspect [28–30, 44, 47, 53]. For instance, BEVDepth [28] introduces LiDAR supervision while BEVStereo [29, 30] leverages stereo depth-estimation techniques, both aiming at helping the depth estimation. Parametric-Depth [47] instead imposes a Laplacian distribution as a prior to reduce the ambiguity of the estimation.

Unlike aforementioned methods, BEVFormer [31] and Simple-BEV [13] 'pull' features to the BeV space without relying on estimated depth maps. Instead, they define a 3D volume of coordinates over the BeV plane, project each 3D coordinate into the camera images, and pull the image features neighboring the resulting locations. The image features are thus not precisely located in the BeV but instead spread over possible locations. This greatly simplifies the process, making it both more efficient and robust to projection errors. In our work, we build upon these 'feature pulling' methods and we further improve their efficiency and flexibility using sparse representations.

Temporal Modelisation. A variety of temporal fusion methodologies has been explored, particularly warp-based techniques [1, 12, 15, 16, 19, 26, 31, 38, 51]. These methods typically factor in ego-motion to spatially align historical BeV grids to the current BeV. This facilitates the aggregation of multiple temporal frames by simply stacking the aligned BeV or with relevant neural modules such as deformable attention [31] or Conv-RNNs [1, 51]. In addition to the warping relative to the ego-motion, BEVerse [51] and PowerBEV [26] also explicitly model the other vehicles' instance flow and displace them to their predicted current location. However, naive BeV map warping techniques only retain the spatial region from past data that aligns with the current BeV, discarding potentially valuable information from other regions. This hinders long-term temporal fusion, as the overlap in BeV decreases with the ego-car's movement [9, 40]. To circumvent these issues, methods like UniFusion [40] and TBP-Former [9] propose to project the current BeV to the previous camera frames, instead of warping past BeV to the present one. Still, the aggregation now suffers from the opposite problem, as many past frames, especially far away in time, would only contain little information relevant to the current location. Yet, they would still be fully computed and propagated through the network.

Our temporal sparse approach allows us to control the compute by removing the less relevant points before the temporal aggregation, and thus to attain long past horizons. Note that some of the works cited in this section, do not tackle BeV segmentation tasks [9].

3. PointBeV

Our method PointBeV focuses on efficiently converting perspective camera views to BeV. The approach is outlined in three key components. First, in Sec. 3.1, we detail the efficient transformation of perspective camera views to BeV using sparse techniques, introducing our custom *Sparse*

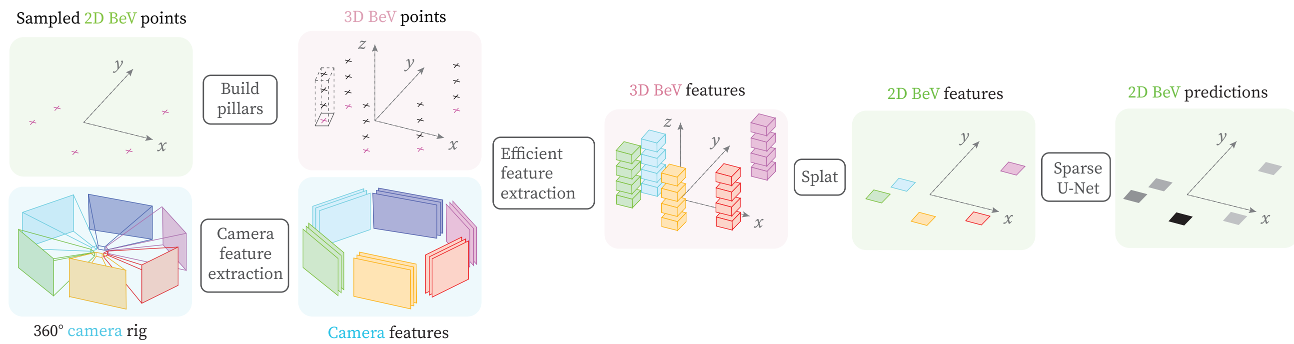

Figure 2. **PointBeV architecture.** As a sparse method, PointBeV is trained using local predictions, only for sampled 2D points provided as inputs. The selection of those points during training and at test time is illustrated in Fig. 4. The points of interest are lifted to form 3D pillars, with each 3D point pulling visual features. To achieve this, PointBeV incorporates an efficient feature extraction process through a *Sparse Feature Pulling* module, illustrated in the 'efficient feature extraction' block and further explained in Sec. 3.1 and Fig. 3. The obtained 3D BeV features are then flattened onto the 2D BeV plane and processed using a sparse U-Net with task-dependent final heads, generating local BeV predictions. For training, we only need sparse signals. At test time, points that have not been sampled are set to zero.

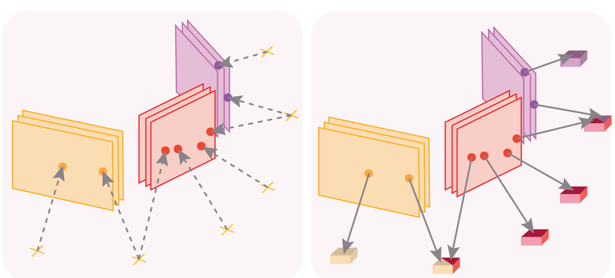

Figure 3. **Sparse Feature Pulling and Camera Fusion.** 3D BeV points are projected into the localized camera features (left). From there, camera features are bilinearly interpolated to obtain the 3D BeV features at this position (right). Where previous methods project points onto all the cameras regardless of their visibility, or pad the number of points so that there are as many per camera, we conduct feature pulling, for each camera, only on the visible 3D points. If a point is visible to a single camera, the feature pulling is done only within the corresponding feature volume.

Feature Pulling module. Second, we introduce a two-stage 'coarse/fine' learning strategy in Sec. 3.2 that explores and focuses on regions of interest, enabling the training of the model with significantly fewer points. Third, in Sec. 3.3, we leverage the sparse nature of PointBeV to develop a submanifold temporal attention aggregation strategy that focuses on regions of interest and mitigates ambiguities associated with low visibilities. Lastly, in Sec. 3.4, we present sparse sampling strategies that enable the exploration of different efficiency/accuracy trade-offs at inference time. An overview of the architecture is shown in Fig. 2.

3.1. Sparse Feature Propagation

PointBeV is a sparse approach to BeV segmentation. Specifically, given a set of 2D BeV locations, also called 'points', PointBeV predicts the occupancy state of these points from the camera images. This contrasts with traditional grid-based approach that predicts the entire BeV. The next subsections detail the selection of these 2D BeV locations during both training and inference, whereas here, we present the network architecture working with points.

Formally, for a given camera C_i out of the N_{cam} cameras, we extract visual features using any backbone network. These features are noted $\mathcal{I}_i \in \mathbb{R}^{C\times H\times W}$, where $C, H, W \in \mathbb{N}$ represent the number of channels, height and width of the feature volumes, respectively. For each 2D BeV point (x, y), we classically construct a pillar composed of 3D points evenly spaced vertically in the BeV space $p = \{p_z = (x, y, z)\}_{z\in Z}$ where Z is the evenly discretized vertical axis [25]. Then, for any given 3D pillar point p_z, we define the set of camera indices $C(p_z) := \{i \mid C_i \triangleleft p_z\}$ representing the cameras that have the pillar point p_z in their field of view ('is seeing' is noted '$\triangleleft$'). Typically, in most multi-camera datasets, a point is visible to one or two cameras depending on the overlapping areas. The 3D pillar point p_z is then projected into the camera feature volumes $\{\mathcal{I}_i\}_{i\in C(p_z)}$, and a bilinear interpolation is performed to calculate the features corresponding to the 3D point. We refer to the joint operations of projecting the 3D pillar points to the cameras, and extracting and interpolating the corresponding image features as *Sparse Feature Pulling*. When the pillar point falls in several cameras, the feature pulling is conducted separately in each camera feature volume and the features are then averaged. This part of the feature propagation pipeline is illustrated in Fig. 3.

We should note that contrary to most recent BeV approaches [13, 31], we do not project our points on cameras outside of their visibility field during feature pulling. As in most of the autonomous driving multiple-camera datasets containing six cameras, each of them seeing about ∼16% of the BeV space, we avoid ∼84% of dispensable calculations

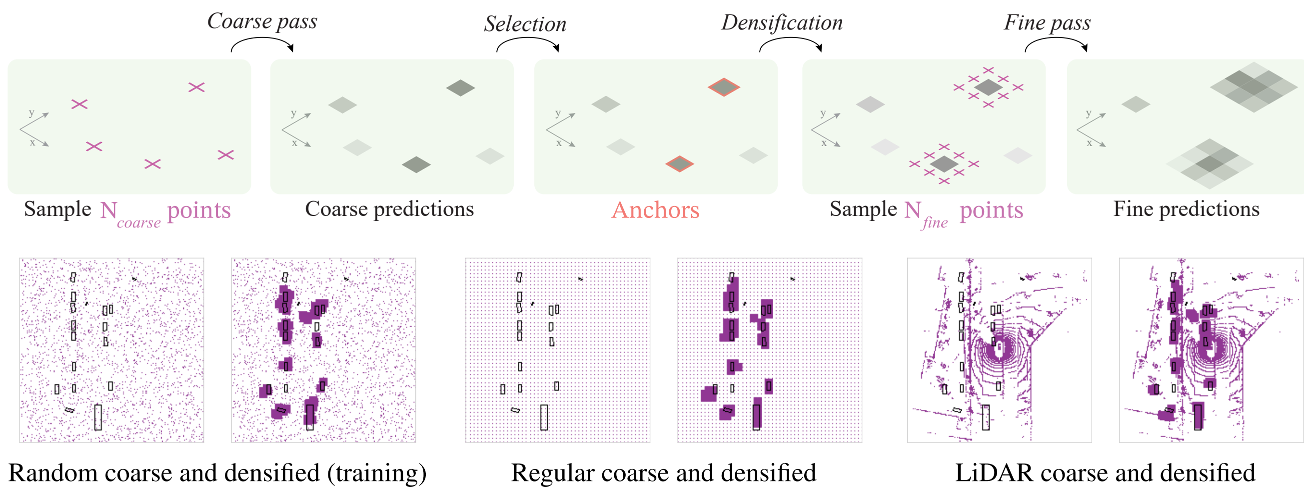

Figure 4. **Illustration of the 'coarse' and 'fine' passes.** Top row: given sampled BeV points, predictions are made at these locations in the 'coarse pass'. We select highest logits points as 'anchors'. Around these anchors, points are densely sampled using a kernel of size $k_{fine} \times k_{fine}$ (3×3 in this vizualisation). Then the 'fine pass' provides predictions for these points. The networks (Fig. 2) are shared between passes, and the camera feature extraction is only done once as the features don't change. This figure illustrates both the training and the inference stages, and we stress non-visible differences between these two. *During training*, (1) the coarse points are typically randomly sampled from a uniform distribution, and (2), the top N_{anchor} activations are selected as anchors. *During inference*, (1) the coarse points are sampled using different strategies such as the subsampled pattern (see Sec. 3.4), and (2) points having a score above the threshold τ are selected as anchors. To evaluate the entire dense BeV, we instead make a single pass with all BeV points. The bottom row displays sampling masks for three different sampling strategies, with the ground-truth vehicles' bounding boxes delineated in black for visualization.

that are often still performed in approaches. To achieve this feat, we implement a *Sparse Feature Pulling* module that can be used to replace any multi-view feature pulling scenario. We detail it and report its efficiency in App. C.

Once features are obtained for considered pillar points, they are flattened onto the 2D BeV plane and processed using a sparse U-Net, generating local BeV segmentation predictions. The sparse U-Net is a sparse adaptation of the classic U-Net model used in previous approaches [2, 13] containing commonly used ResNet blocks [14].

3.2. Coarse and fine training

Given the dense nature of segmentation tasks, previous approaches treat the BeV grid as an image. This approach necessitates a large amount of memory, typically preventing training models on small GPUs, and using large temporal windows, fine-grain resolution, or long BeV ranges. Instead, during learning, we opt to subsample points from the BeV grid in the training phase. However, naively tackling the dense segmentation task with these points results in training instabilities. To address this, we take inspiration from sampling strategies used in the NeRF literature [36] and conceive a two-stage 'coarse/fine' learning approach. The goal is threefold: enhancing performance by focusing on discriminative regions, stabilizing the training process, and controlling memory usage during training.

This 'coarse/fine' training strategy, as illustrated in Fig. 4, involves two distinct passes. The *coarse pass* aims to cover the space as efficiently as possible, while the subsequent *fine pass* focuses on areas identified as relevant by the coarse pass. In the coarse pass, we use a uniform sampling strategy to draw a fixed number N_{coarse} of points that we forward in PointBeV (Fig. 2). Then, we select anchor points as the N_{anchor} points with the highest logits. For the fine pass, we consider these anchor points and their neighbors in a square window of size $k_{fine} \in \mathbb{N}$, a strategy we term *densification*. Among all the obtained densified points, we then keep N_{fine} points that we forward through the network for predictions (Fig. 2 again). The outputs from both passes are merged before computing the cross-entropy, considering only the selected points. This strategy allows us to oversample regions of interest while reducing the total amounts $N_{point} := N_{coarse} + N_{fine}$ of points that are processed.Typically, we split equally the total number of points between the two stages: $N_{coarse} = N_{fine}$ to maintain a balance between exploration and refinement.

3.3. Sparse temporal model

Leveraging past frames to build BeV maps can help to discern static from dynamic objects, understand traffic patterns, and alleviate ambiguities caused by transient obstructions, visual artifacts, or depth uncertainty. For temporal aggregation, we take inspiration from the strategy used in BeV detection [40] to project the current locations of interest into past images, and we adapt it to a sparse setting.

Let us consider a sequence of $T \in \mathbb{N}$ frames encompassing both past and present contexts, with the goal of predicting the BeV map of the present moment. To achieve this, the coordinates of sampled points in the current map are projected into the cameras of past moments. This process yields the features of points in the current map for each time frame. However, instead of aggregating all points, we selectively retain past points based on their logit values, introducing a temporal threshold $\tau_{temp} \in [0, 1]$. Points exceeding this threshold are preserved, while those below it are discarded. This results in a sparse BeV per timestep.

To fuse the different timesteps, we introduce a *Submanifold Attention* module employed in a temporal context, illustrated in Fig. 5 and inspired by window attention from the LiDAR 3D detection method [24]. Formally, we represent a 2D BeV point (x, y) at current time t_0 already processed by our model as a query $Q_{t_0,x,y}$. Unlike more traditional approaches, the module is set with a spatiotemporal window $W := (w^t, w^x, w^y)$ where $w^t \in \mathbb{N}$ is a temporal range and $w^x, w^y \in \mathbb{N}$ are spatial windows defining a neighborhood around each query point. For any given query point, only keys and values associated to points within the neighborhood of the query are considered when computing the attention. This new attention mechanism can be written as:

$$O_{t_0,x,y} = \sum_{(t_k,x_k,y_k)\in W} A_{t_k,x_k,y_k}\left(\frac{Q_{t_0,x,y}K^{\top}_{t_k,x_k,y_k}}{\sqrt{d_k}}\right)V^{\top}_{t_k,x_k,y_k},$$

where A_{t_k,x_k,y_k} is the softmax term of the attention coefficients. Due to the sparsity of the past data, each query may have varying numbers of keys and values. This attention mechanism performs calculations only with point combinations within the specified window. Note that an infinite spatiotemporal window recovers standard attention.

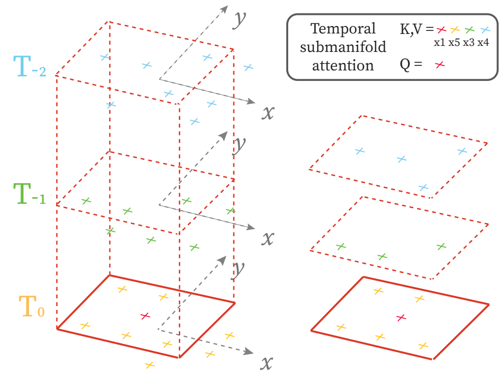

Figure 5. **Illustration of the 'Submanifold Temporal Attention' module.** Our module performs an attention between a query point (colored in red), at the center of a spatio-temporal neighborhood (red dotted lines and complete parallelepiped). The points inside this neighborhood become the keys and values for the attention mechanism. The points outside are discarded. Consequently, the number of keys and values depends on the number of points present in the vicinity of the query point. More details in Sec. 3.3.

3.4. Inference with PointBeV

To perform inference with PointBeV multiple strategies can be adopted. First, for comparative evaluations against other approaches, a setting referred to as 'dense inference' can be employed. This involves processing all grid points with PointBeV in a single pass, producing predictions akin to considering the entire image, as in a dense model.

Alternatively and interestingly, we can also use sparse sampling *during test time* to avoid allocating resources to regions of lesser interest. This is similar to the 'coarse/fine' training strategy with slight modifications. We first need to define an exploration pattern for the coarse pass. We can consider diverse options, including uniform sampling as used in training, a loose regular pattern arranged in a grid, or a distance-to-ego dependent pattern. These strategies can be tailored based on specific use cases, such as emphasizing long-range sampling for highway driving or comprehensive, closer-range sampling for parking scenarios. Besides, PointBeV can also leverage additional information present at test-time, when available. For instance, LiDAR beams directly detect physical structures like roads or vehicles, providing valuable prior information for locating semantic classes. Moreover, if available, we can use an HD map to minimize unnecessary computations in non-driveable regions such as buildings.

In the absence of such priors, we initialize the coarse pass by subsampling the BeV grid in evenly spaced locations, with each selected point spaced from its nearest points by a set parameter k on the grid. $k = 1$ recovers the dense grid. Then, anchor points are selected where the model predicts a confidence above a threshold $\tau \in [0, 1]$. These anchor points are densified with kernel $k_{fine} = k$ for the fine pass, mirroring the training strategy. Coarser sub-sampling of the grid results in a larger k_{fine}. Unless stated otherwise, this setting is used for our sparse regime evaluations.

Using such methods, however, results in sparse predictions that do not cover the dense BeV. To build a dense prediction map, for evaluation purposes typically, we simply consider the non-sampled locations as empty cells. If ground-truth elements exist at these locations and are not predicted, they are counted as false negatives. Illustrations of different strategies are shown in Fig. 4 and Fig. 6.

4. Experiments

Data, training and implementation details. Our experiments are conducted on nuScenes [3] and Lyft L5 [6] datasets. NuScenes contains 1000 scenes split into 750-150-150 scenes for the training, validation, and test sets. Lyft L5 contains 180 scenes, each 25-45 seconds in length,

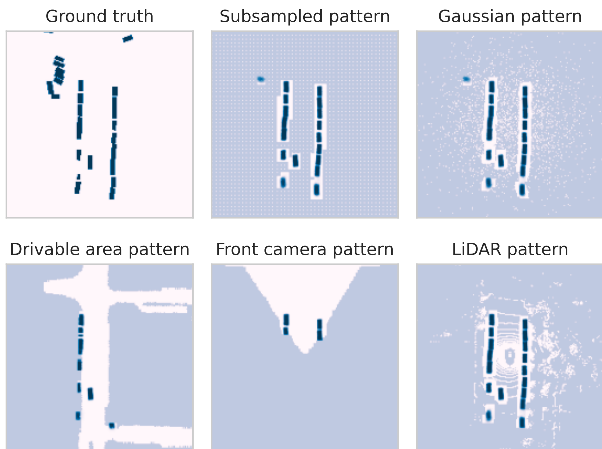

Figure 6. **Comparison of various sampling patterns with their predictions using the coarse-fine inference.** The inference coarse initialization can be declined in several strategies. On the first row we illustrated from left to right: the ground truth, the subsampled pattern and the Gaussian pattern concentrating more around points close to the ego-vehicle. On the second row we illustrate from left to right, the drivable area pattern, the front camera pattern and the LiDAR pattern. The predominantly pale colour indicates that no points have been sampled and that the associated prediction will be considered null.

annotated at 5Hz, that we split as in FIERY [15]. For all our experiments, PointBeV is trained on a maximum of 100 epochs using an Adam [23] optimizer with learning rate $\lambda = 3e^{-4}$, weight decay $w = 10^{-7}$, a one-cycle linear learning rate scheduler. We used a 100m×100m grid with a 50cm resolution resulting in a 200x200 grid. For the training, unless specified, we used a random coarse strategy with $N_{coarse} = N_{fine} = 2.5k$ corresponding to 1/16 of the points of the grid. The densification patch size k_{fine} is set to 9. The number of anchor points is defined by $N_{anchor} = 100$. We train our model using a segmentation, an offset and a centerness loss as in FIERY [15]. For sparse inference, the fine threshold is set to $\tau = 0.1$. For the temporal models, the temporal threshold is set to $\tau_{temp} = \text{sigm}(-5)$ and we use 2 seconds of context corresponding to 8 frames in the past. Details about the choice and the robustness of the parameters can be found in App. E. Unless stated otherwise, our experiments are done with a ResNet-50 [14] backbone, at image resolution 224×480, and without visibility filtering of low visilibity vehicle. An EfficientNet-b4 [43] backbone can be introduced for fair comparisons with prior work. The neck network is a simple upsampling with small convolutions between different resolutions, while the Sparse-UNet is coded using a sparse neural network library [7] following the standard UNet architecture of Simple-BEV [13]. Our *Submanifold Attention* module is coded using a graph neural network library [10] and C++, and our efficient *Sparse Feature Pulling* module implemented in CUDA .

Vehicule segm. IoU (↑)			*No visibility filtering*		*Visibility filtering*	
Method	Backb.	Temp.	224×480	448×800	224×480	448×800
FIERY static [15]	EN-b4		35.8	—	39.8	—
CVT [52]	EN-b4		31.4*	32.5*	36.0	37.7*
LaRa [2]	EN-b4		35.4	—	38.9	—
BEVFormer [31]	RN-50		35.8**	39.0**	42.0**	45.5**
Simple-BEV [13]	RN-50		36.9*	40.9*	43.0*	46.6
BAEFormer [37]	EN-b4		36.0	37.8	38.9	41.0
PointBeV	EN-b4		**38.7**	**42.1**	**44.0**	**47.6**
PointBeV	RN-50		38.1	41.7	43.7	47.0
FIERY [15]	EN-b4	✓	38.2	—	—	—
PointBeV-T	EN-b4	✓	**39.9**	**43.2**	**44.7**	**48.7**
PointBeV-T	RN-50	✓	**39.9**	**43.2**	44.1	47.7

Table 1. **BeV vehicle segmentation on nuScenes.** computed on the validation set at different resolutions and for different filtering based on vehicle's visibility. No visibility filtering means all the annotated vehicles are considered. Visibility filtering means only the vehicles having a visibility $> 40\%$ are considered. '*' indicates scores obtained using official codes, '**' indicates scores we obtained after reimplementing the model. In some cases, our reproduction yields higher scores than the ones reported in original papers (see Supplementary Materiel A for comparison details). 'EN-b4' and 'RN-50' stand for EfficientNet-b4 [43] and ResNet-50 [14] respectively. 'Temp' stands for 'temporal' models using past frames. For fair comparisons we use 8 past frames.

Vehicule segm. IoU (↑)	Long	Short
FIERY	36.7	59.4
BeVFormer (EN-b4)	44.5	69.9
BeVFormer (RN-50)	43.2	68.8
SimpleBEV (EN-b4)	44.5	70.4
SimpleBEV (RN-50)	43.6	70.7
PointBeV (EN-b4)	**45.4**	**72.6**
PointBeV (RN-50)	**44.5**	**72.3**

Table 2. **BeV vehicle segmentation on Lyft L5.** Scores are IoU (↑), models are trained at 224×480 resolution for different backbones and ranges: 30m × 30m (Short) and 100m × 100m (Long). Data splits are from FIERY [15].

4.1. State-of-the-art comparison

We compare the performance of PointBeV against a collection of BeV segmentation methods from the literature on vehicle, pedestrian, and lane segmentation tasks, in static and temporal settings, on nuScenes [3].

For vehicle segmentation, PointBeV consistently outperforms existing methods, achieving state-of-the-art performance on nuScenes [3] and Lyft L5 [6] across various settings (Tab. 1, Tab. 2) . Our method exhibits superiority at different resolutions (224×480 or 448×800), under different standard visibility settings, for different backbones, and for both static and temporal settings. In the absence of

Pedestrian segm.	Temp.	IoU (↑)
LSS [39]		15.0
FIERY [15]		17.2
ST-P3 [17]		14.5
TBP-Former static [9]		17.2
PointBeV		**18.5**
TBP-Former [9]	✓	18.6
PointBeV-T	✓	19.9

Table 3. **BeV pedestrian segmentation on nuScenes.** Scores are IoU (↑) with visibility filtering, computed on the validation set at 224×480 resolution. 'Temp.' refers to temporal models using 8 contextual frames for fair comparisons.

Lane segm.	IoU (↑)
BEVFormer [31]	25.7
PETRv2 [33]	44.8
M^2BEV [44]	38.0
MatrixVT [53]	44.8
PointBeV	**49.6**

Table 4. **BeV lane segmentation on nuScenes [3] validation set.** Scores are IoU (↑), computed using models trained at 224×480 image resolution. All models are static.

reported results in original papers, we either run the original codes or we reimplement the method. We ensure reproductions are consistent with the original scores in App. A.

We expand our evaluations to pedestrian (Tab. 3) and lane (Tab. 4) segmentation. Remarkably, without adjusting the number of training points nor the patch sizes, our approach sets new state-of-the-art for these tasks. PointBeV surpasses previous state-of-the-art by +1.3 IoU points for pedestrians (TBP-Former [9]) in both static and temporal regimes, and by +4.8 IoU points for lanes (MatrixVT [53]).

4.2. Ablations

To validate the different components of our model, we conduct an ablation study focusing on the efficiency and impact of key modules, including our custom *Sparse Feature Pulling* module (Sec. 3.1), our *Submanifold Attention* module (Sec. 3.3), and the coarse/fine training passes (Sec. 3.2).

Sparse feature pulling. To assess the efficiency of our *Sparse Feature Pulling* module, we conduct a comparative evaluation of execution time and memory usage between the default module and our proposal. The results are summarized in Tab. 5. They reveal a notable improvement in memory efficiency, ranging between 25% to 45%, when utilizing the sparse interpolation module. Simultaneously, there is a marginal increase in FPS while GPU modules are not fully optimised to handle a different number of keys and values.

	bs=1		bs=2		bs=12		bs=28	
	Mem	FPS	Mem	FPS	Mem	FPS	Mem	FPS
PointBeV	3.50	14.84	5.50	10.01	25.99	2.30	39.40	1.00
w/o Sparse Feature Pulling	4.57	13.70	7.61	9.81	38.94	2.12	—	—

Table 5. **Ablation of the Sparse Feature Pulling module.** Memory consumption and FPS for our model using or not our sparse module, computed on a 40GB A100 under dense inference setting. '—' indicates 'out of memory'. 'Mem' is for Memory footprint in GiB, and 'bs' stands for batch size.

Vehicle segm. IoU (↑)			*No visibility filtering*		*Visibility filtering*	
Method	Temp.	N_{point}	224×480	448×800	224×480	448×800
PointBeV		40k	38.09	41.66	43.70	47.20
PointBeV-T	✓	∼66k	39.93	43.19	44.06	47.67
w/o subman. att.	✓	320k	40.49	43.53	44.12	47.63

Table 6. **Ablation of the submanifold temporal attention.** Scores are IoU (↑) computed on the nuScenes [3] validation set at different resolutions and vehicle's visibility filterings.

Interestingly, the use of the module enables accommodating larger batches, notably up to a batch size of 28 on a 40GiB A100 GPU. This enhancement is key, enabling faster training with larger batches, the consideration of extended temporal contexts, and supporting increased BeV range and resolution. For detailed insights into the module's specific performance, we provide further investigations in App. C.

Submanifold Temporal Attention. In Tab. 6, we conduct an ablation study focusing on the Submanifold Temporal attention. Although the results can be very marginally improved with standard attention in some settings, achieving this enhancement comes with the cost of considering approximately $\sim 5\times$ more points and computations. Leveraging the submanifold temporal attention unlocks new possibilities for considering longer temporal contexts. Accordingly, we trained and evaluated PointBeV under higher number of frames, up to 25, and obtained an IoU of 40.73.

Coarse and fine training. Tab. 7 displays the performance evolution of BeV vehicle segmentation across various point budgets *during training*, with similar training times for all models. Remarkably, PointBeV achieves similar performances with only 20% of the total points instead of the whole BeV grid. It shows how the use of sparse training approach in PointBeV allows to optimize memory usage effectively, e.g., to train on longer temporal contexts.

Additionally, to validate the efficiency of our two-stage training strategy involving coarse and fine passes, we analyze two settings based on a given point budget N_{point}. The first setting uses only a coarse pass ($N_{coarse} = N_{point}$, $N_{fine} = 0$) and achieves an IoU of 35.4. In contrast, the second setting is our coarse/fine strategy and divides the point budget equally between both passes

N_{point} (% of full BeV)	4k (10%)	8k (20%)	20k (50%)	40k (100%)
PointBeV	38.1	38.3	38.4	38.3

Table 7. Evolution of BeV vehicle segmentation scores **IoU** (↑) **for various point budgets.** The budget is given as total number of points used in both passes and expressed as a percentage of the total number of points in a dense grid. With only 20% of points, we find similar performances as with 100%.

($N_{coarse} = N_{fine} = N_{point}/2$), yielding an IoU of 38.3 (as depicted in Tab. 7). These results highlight the superiority of splitting points between sequential coarse and fine passes. Besides, we find that the two-pass approach considerably stabilizes training.

4.3. Adaptive Inference Capabilities

One intriguing aspect of PointBeV lies in its inherent flexibility to adjust between efficiency and accuracy, and to adapt to the demands of specific use cases. This is illustrated in Fig. 1 and Fig. 7, where the choice of the number of points directly influences both performance and memory footprint. Interestingly, achieving a comparable IoU score (approximately 38.3 for Fig. 1 and 44.0 for Fig. 7) is feasible by employing only one-sixth of the total points, significantly reducing memory usage by a third at inference time. This underscores the model's efficient utilization of computational resources without compromising performance. Note that our general sparse sampling setting (Sec. 3.4) is chosen to perform well in most common scenarios. For particular cases such as small or very large objects, we anticipate that it can be further improved with specific strategies.

Lastly, as depicted in Fig. 6 and previously discussed in Sec. 3.4, PointBeV exhibits the capability to integrate external priors to guide its focus. These priors can stem from various sources such as LiDAR point clouds, which reveal physical structures, or HD maps that aid in reducing sampling in regions irrelevant to the task, such as looking for agents within buildings. In Fig. 7, we see that by sampling points with this LiDAR prior, superior IoU can be achieved compared to the dense setting (PointBeV$_{LiDAR}$ reaches 44.5 vs 44.0 for default sampling of PointBeV), with a much lower number of points and memory usage. By leveraging LiDAR data, PointBeV$_{LiDAR}$ minimizes noise in empty regions leading to improved segmentation accuracy. Otherwise, a coarse pass based on a Gaussian prior around the ego car could be used to reinforce attention to closer ranges.

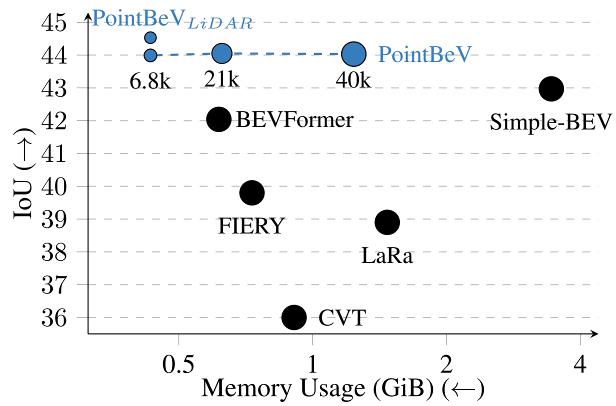

Figure 7. **BeV vehicle IoU vs. memory footprint.** The size of a dot represents the number of BeV points being evaluated (the smaller the better). PointBeV has the capacity to explore various trade-offs between efficiency (Memory usage) and the performances (IoU) by varying the number of points being evaluated. Here, we show PointBeV models with an EfficientNet-b4, and vehicle with low visibility are filtered out (unlike in Fig. 1).

5. Conclusion

We introduced PointBeV for BeV segmentation from camera inputs. By integrating sparse modules and an innovative training strategy, our method operates efficiently while setting a new state-of-the-art with or without temporal information, on nuScenes segmentation benchmarks. PointBeV also accommodates to additional test-time information or to low compute regimes without retraining.

This work is only a first step in exploring the potential of the sparse paradigm for BeV segmentation. Upcoming directions might encompass 2D occupancy forecasting and 3D voxel occupancy estimation applications. These tasks would immediately benefit from the reduced memory footprint for longer temporal horizons or higher spatial resolutions. Our sparse inference regime uses a simple dense BeV completion by setting every non-selected point as empty; introducing a light completion network may yield additional gains. Further down the road, future directions may explore dynamic sampling strategies guided by sensor inputs, by following agent flows to determine sampling locations for subsequent frames, or by the need of downstream tasks such as forecasting or planning. Finally, this new sparse paradigm is also very suited to the unification of BeV tasks at diverse resolutions and ranges within the same model.

Acknowledgments. This paper is dedicated to Laura E-R. We received support of the French Agence Nationale de la Recherche (ANR), under grant ANR-21-CE23-0032 (project MultiTrans). This work was performed using HPC resources from GENCI–IDRIS (Grant 2023-AD011014252). In addition we thank Yihong Xu and Alexandre Boulch for their contributions and exchanges of ideas throughout the process, particularly with regard to the implementation of the submanifold temporal attention.

References

[1] Adil Kaan Akan and Fatma Güney. StretchBEV: Stretching future instance prediction spatially and temporally. In *ECCV*, 2022. 2

[2] Florent Bartoccioni, Eloi Zablocki, Andrei Bursuc, Patrick Perez, Matthieu Cord, and Karteek Alahari. LaRa: Latents and rays for multi-camera bird's-eye-view semantic segmentation. In *CoRL*, 2022. 2, 4, 6

[3] 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. 1, 2, 5, 6, 7, 3, 4

[4] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. NEAT: neural attention fields for end-to-end autonomous driving. In *ICCV*, 2021. 2

[5] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. *TPAMI*, 2023. 2

[6] NikiNikatos Phil Culliton Vinay Shet Vladimir Iglovikov Christy, Maggie. Lyft 3d object detection for autonomous vehicles, 2019. 5, 6

[7] Spconv Contributors. Spconv: Spatially sparse convolution library. https://github.com/traveller59/spconv, 2022. 6

[8] Vikrant Dewangan, Basant Sharma, Tushar Choudhary, Sarthak Sharma, Aakash Aanegola, Arun Kumar Singh, and K. Madhava Krishna. UAP-BEV: uncertainty aware planning using bird's eye view generated from surround monocular images. In *CASE*, 2023. 1

[9] Shaoheng Fang, Zi Wang, Yiqi Zhong, Junhao Ge, and Siheng Chen. Tbp-former: Learning temporal bird's-eye-view pyramid for joint perception and prediction in vision-centric autonomous driving. In *CVPR*, 2023. 1, 2, 7

[10] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *CoRR*, abs/1903.02428, 2019. 6

[11] Junru Gu, Chenxu Hu, Tianyuan Zhang, Xuanyao Chen, Yilun Wang, Yue Wang, and Hang Zhao. ViP3D: End-to-end visual trajectory prediction via 3d agent queries. In *CVPR*, 2023. 1

[12] Chunrui Han, Jianjian Sun, Zheng Ge, Jinrong Yang, Runpei Dong, Hongyu Zhou, Weixin Mao, Yuang Peng, and Xiangyu Zhang. Exploring recurrent long-term temporal fusion for multi-view 3d perception. *arXiv preprint arXiv:2303.05970*, 2023. 2

[13] Adam W. Harley, Zhaoyuan Fang, Jie Li, Rares Ambrus, and Katerina Fragkiadaki. Simple-BEV: What really matters for multi-sensor bev perception? In *ICRA*, 2023. 1, 2, 3, 4, 6

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4, 6, 1, 2

[15] Anthony Hu, Zak Murez, Nikhil Mohan, Sofía Dudas, Jeffrey Hawke, Vijay Badrinarayanan, Roberto Cipolla, and Alex Kendall. FIERY: Future instance segmentation in bird's-eye view from surround monocular cameras. In *ICCV*, 2021. 1, 2, 6, 7

[16] Chunyong Hu, Hang Zheng, Kun Li, Jianyun Xu, Weibo Mao, Maochun Luo, Lingxuan Wang, Mingxia Chen, Kaixuan Liu, Yiru Zhao, Peihan Hao, Minzhe Liu, and Kaicheng Yu. FusionFormer: A multi-sensory fusion in bird's-eye-view and temporal consistent transformer for 3d objection. *arXiv preprint arXiv:2309.05257*, 2023. 2

[17] Shengchao Hu, Li Chen, Penghao Wu, Hongyang Li, Junchi Yan, and Dacheng Tao. ST-P3: end-to-end vision-based autonomous driving via spatial-temporal feature learning. In *ECCV*, 2022. 7

[18] Yihan Hu, Jiazhi Yang, Li Chen, Keyu Li, Chonghao Sima, Xizhou Zhu, Siqi Chai, Senyao Du, Tianwei Lin, Wenhai Wang, Lewei Lu, Xiaosong Jia, Qiang Liu, Jifeng Dai, Yu Qiao, and Hongyang Li. Planning-oriented autonomous driving. In *CVPR*, 2023. 1, 2

[19] Junjie Huang and Guan Huang. BEVDet4D: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint arXiv:2203.17054*, 2022. 2

[20] Yuanhui Huang, Wenzhao Zheng, Yunpeng Zhang, Jie Zhou, and Jiwen Lu. Tri-perspective view for vision-based 3d semantic occupancy prediction. In *CVPR*, 2023. 1

[21] Yupeng Jia, Jie He, Runze Chen, Fang Zhao, and Haiyong Luo. OccupancyDETR: Making semantic scene completion as straightforward as object detection. *arXiv preprint arXiv:2309.08504*, 2023. 1

[22] Jinkyu Kim, Reza Mahjourian, Scott Ettinger, Mayank Bansal, Brandyn White, Ben Sapp, and Dragomir Anguelov. StopNet: Scalable trajectory and occupancy prediction for urban autonomous driving. In *ICRA*, 2022. 1

[23] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[24] Xin Lai, Jianhui Liu, Li Jiang, Liwei Wang, Hengshuang Zhao, Shu Liu, Xiaojuan Qi, and Jiaya Jia. Stratified transformer for 3d point cloud segmentation. In *CVPR*, 2022. 2, 5

[25] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. PointPillars: Fast encoders for object detection from point clouds. In *CVPR*, 2019. 3

[26] Peizheng Li, Shuxiao Ding, Xieyuanli Chen, Niklas Hanselmann, Marius Cordts, and Juergen Gall. PowerBEV: A powerful yet lightweight framework for instance prediction in bird's-eye view. In *IJCAI*, 2023. 1, 2

[27] Siyu Li, Kailun Yang, Hao Shi, Jiaming Zhang, Jiacheng Lin, Zhifeng Teng, and Zhiyong Li. Bi-Mapper: Holistic bev semantic mapping for autonomous driving. *IEEE Robotics and Automation Letters*, 2023. 1

[28] 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. *AAAI*, 2022. 1, 2

[29] Yinhao Li, Han Bao, Zheng Ge, Jinrong Yang, Jianjian Sun, and Zeming Li. BEVStereo: Enhancing depth estimation in multi-view 3d object detection with temporal stereo. *AAAI*, 2023. 2

[30] Yinhao Li, Jinrong Yang, Jianjian Sun, Han Bao, Zheng Ge, and Li Xiao. BEVStereo++: Accurate depth estimation in multi-view 3d object detection via dynamic temporal stereo. *arXiv preprint arXiv:2304.04185*, 2023. 2

[31] 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. 1, 2, 3, 6, 7

[32] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. PETR: Position embedding transformation for multi-view 3d object detection. In *ECCV*, 2022. 2

[33] Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Aqi Gao, Tiancai Wang, and Xiangyu Zhang. PETRv2: A unified framework for 3d perception from multi-camera images. In *ICCV*, 2023. 2, 7

[34] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela L. Rus, and Song Han. BEVFusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *ICRA*, 2023. 1

[35] Reza Mahjourian, Jinkyu Kim, Yuning Chai, Mingxing Tan, Ben Sapp, and Dragomir Anguelov. Occupancy flow fields for motion forecasting in autonomous driving. *RAL*, 2022. 1, 2

[36] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 4

[37] Cong Pan, Yonghao He, Junran Peng, Qian Zhang, Wei Sui, and Zhaoxiang Zhang. BAEFormer: Bi-directional and early interaction transformers for bird's eye view semantic segmentation. In *CVPR*, 2023. 2, 6

[38] Jinhyung Park, Chenfeng Xu, Shijia Yang, Kurt Keutzer, Kris M. Kitani, Masayoshi Tomizuka, and Wei Zhan. Time Will Tell: New outlooks and A baseline for temporal multi-view 3d object detection. In *ICLR*, 2023. 2

[39] Jonah Philion and Sanja Fidler. Lift, Splat, Shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *ECCV*, 2020. 1, 2, 7

[40] Zequn Qin, Jingyu Chen, Chao Chen, Xiaozhi Chen, and Xi Li. UniFusion: Unified multi-view fusion transformer for spatial-temporal representation in bird's-eye-view. In *ICCV*, 2023. 1, 2, 4

[41] Andrea Simonelli, Samuel Rota Bulò, Lorenzo Porzi, Peter Kontschieder, and Elisa Ricci. Are we missing confidence in pseudo-lidar methods for monocular 3d object detection? In *ICCV*, 2021. 2

[42] Apoorv Singh. Vision-radar fusion for robotics BEV detections: A survey. In *IEEE Intelligent Vehicles Symposium*, 2023. 1

[43] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 6, 2, 4

[44] Enze Xie, Zhiding Yu, Daquan Zhou, Jonah Philion, Anima Anandkumar, Sanja Fidler, Ping Luo, and Jose M Alvarez. M^{2}BEV: Multi-camera joint 3d detection and segmentation with unified birds-eye view representation. *arXiv preprint arXiv:2204.05088*, 2022. 2, 7

[45] 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. 2

[46] Yihong Xu, Loïck Chambon, Éloi Zablocki, Mickaël Chen, Alexandre Alahi, Matthieu Cord, and Patrick Pérez. Towards motion forecasting with real-world perception inputs: Are end-to-end approaches competitive? In *ICRA*, 2024. 2

[47] Jiayu Yang, Enze Xie, Miaomiao Liu, and Jose M. Alvarez. Parametric depth based feature representation learning for object detection and segmentation in bird's-eye view. In *ICCV*, 2023. 2

[48] Yuxiang Yang, Yingqi Deng, Jiahao Nie, and Jing Zhang. BEVTrack: A simple baseline for 3d single object tracking in bird's-eye-view. *arXiv preprint arXiv:2309.02185*, 2023. 1

[49] Tengju Ye, Wei Jing, Chunyong Hu, Shikun Huang, Lingping Gao, Fangzhen Li, Jingke Wang, Ke Guo, Wencong Xiao, Weibo Mao, Hang Zheng, Kun Li, Junbo Chen, and Kaicheng Yu. FusionAD: Multi-modality fusion for prediction and planning tasks of autonomous driving. *arXiv preprint arXiv:2308.01006*, 2023. 1

[50] Jinqing Zhang, Yanan Zhang, Qingjie Liu, and Yunhong Wang. Sa-bev: Generating semantic-aware bird's-eye-view feature for multi-view 3d object detection, 2023. 1

[51] 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. *arXiv preprint arXiv:2205.09743*, 2022. 2

[52] Brady Zhou and Philipp Krähenbühl. Cross-view transformers for real-time map-view semantic segmentation. In *CVPR*, 2022. 2, 6, 1

[53] Hongyu Zhou, Zheng Ge, Zeming Li, and Xiangyu Zhang. Matrixvt: Efficient multi-camera to bev transformation for 3d perception. In *ICCV*, 2023. 2, 7

[54] Sicheng Zuo, Wenzhao Zheng, Yuanhui Huang, Jie Zhou, and Jiwen Lu. Pointocc: Cylindrical tri-perspective view for point-based 3d semantic occupancy prediction. *arXiv preprint arXiv:2308.16896*, 2023. 1