SparseBEV: High-Performance Sparse 3D Object Detection from Multi-Camera Videos

Haisong Liu 1 Yao Teng 1 Tao Lu 1 Haiguang Wang 1 Limin Wang 1,2
1State Key Laboratory for Novel Software Technology, Nanjing University 2Shanghai AI Lab
{liuhs, yaoteng, taolu, haiguangwang}@smail.nju.edu.cn, lmwang@nju.edu.cn

Abstract

Camera-based 3D object detection in BEV (Bird’s Eye View) space has drawn great attention over the past few years. Dense detectors typically follow a two-stage pipeline by first constructing a dense BEV feature and then performing object detection in BEV space, which suffers from complex view transformations and high computation cost. On the other side, sparse detectors follow a query-based paradigm without explicit dense BEV feature construction, but achieve worse performance than the dense counterparts. In this paper, we find that the key to mitigate this performance gap is the adaptability of the detector in both BEV and image space. To achieve this goal, we propose SparseBEV, a fully sparse 3D object detector that outperforms the dense counterparts. SparseBEV contains three key designs, which are (1) scale-adaptive self attention to aggregate features with adaptive receptive field in BEV space, (2) adaptive spatio-temporal sampling to generate sampling locations under the guidance of queries, and (3) adaptive mixing to decode the sampled features with dynamic weights from the queries. On the test split of nuScenes, SparseBEV achieves the state-of-the-art performance of 67.5 NDS. On the val split, SparseBEV achieves 55.8 NDS while maintaining a real-time inference speed of 23.5 FPS. Code is available at https://github.com/MCG-NJU/SparseBEV.

1. Introduction

Camera-based 3D Object Detection [13, 54, 31, 11, 24, 25, 40] has witnessed great progress over the past few years. Compared with the LiDAR-based counterparts [19, 56, 4, 36], camera-based approaches have lower deployment cost and can detect long-range objects.

Previous methods can be divided into two paradigms. BEV (Bird’s Eye View)-based methods [13, 11, 25, 24, 40] follow a two-stage pipeline by first constructing an explicit dense BEV feature from multi-view features and then performing object detection in BEV space. Although achieving remarkable progress, they suffer from high computation cost and rely on complex view transformation operators. Another line of work [54, 31, 32] explores the sparse query-based paradigm by initializing a set of sparse reference points in 3D space. Specifically, DETR3D [54] links the queries to image features using 3D-to-2D projection. It has simpler structure and faster speed, but its performance still lags far behind the dense ones. PETR series [31, 32] uses dense global attention for the interaction between query and image feature, which is computationally expensive and buries the advantage of the sparse paradigm. Thus, a natural question arises whether fully sparse detectors achieve similar accuracy to the dense ones?

In this paper, we find that the key to obtain high performance in sparse 3D object detection is the adaptability of the detector in both BEV and image space. In BEV space,
the detector should be able to aggregate multi-scale features adaptively. Dense BEV-based detectors typically use a BEV encoder to encode multi-scale BEV features. It can be a stack of residual blocks with FPN [26] (e.g., BEVDet [26]), or a transformer encoder with multi-scale deformable attention [60] (e.g., BEVFormer [25]). For sparse detectors such as DETR3D, we argue that the multi-head self attention (MHSA) [49] among queries can play the role of the BEV encoder, as queries are defined in BEV space. However, the vanilla MHSA has a global receptive field, lacking an explicit multi-scale design. In image space, the detector should be adaptive to different objects with different sizes and categories. This is because although the objects have similar sizes in 3D space, they might vary greatly in images. However, the single-point sampling in DETR3D has a fixed local receptive field and the sampled feature is processed by static linear layers, hindering its performance.

To this end, we present SparseBEV, a fully sparse 3D object detector that matches or even outperforms the dense counterparts. Our SparseBEV detector contains three key designs, which are (1) scale-adaptive self attention to aggregate features with adaptive receptive field in BEV space, (2) adaptive spatio-temporal sampling to generate sampling locations under the guidance of queries, and (3) adaptive mixing to decode the sampled features with dynamic weights from the queries. We also propose to use pillars instead of reference points as the formulation of query, since pillars introduce better spatial priors.

We conduct comprehensive experiments on the nuScenes dataset. As shown in Fig. 1, our SparseBEV achieves the performance of 55.8 NDS and the speed of 23.5 FPS on the val split, surpassing all previous methods in both speed and accuracy. Besides, we can flexibly adjust of the inference speed by reducing the number of decoder layers without re-training. On test split, SparseBEV with V2-99 [20] backbone achieves 63.6 NDS without using future frames or test-time augmentation. By further utilizing future frames, SparseBEV achieves 67.5 NDS, outperforming the previous state-of-the-art BEVFormerV2 [55] by 2.7 NDS.

2. Related Work

2.1. Query-based 2D Object Detection

Recently, Transformer [49] with its attention blocks has been widely applied in the computer vision tasks [5, 8, 2]. In object detection, DETR [2] was the first model to predict objects based on learnable queries and treat the detection as a set prediction problem. A lot of works [60, 45, 30, 37, 7, 21, 57] were then proposed to accelerate the convergence of DETR by using the sampled features instead of using the global ones. For example, DeformableDETR [60] samples image features based on sparse reference points, and then applies the deformable attention on the features. Sparse R-CNN [45] uses the ROIAlign [9] to obtain local features and then performs the dynamic convolution. AdaMixer [7] combines the advantages of the sampling points and the dynamic convolution to further boost the convergence. DN-DETR [21] devises a denoising mechanism that feeds ground-truth bounding boxes with noises into decoder and trains the model to reconstruct the original boxes. DINO [57] furtherly optimizes DN-DETR [21] and DAB-DETR [30] by proposing a contrastive denoising training strategy and a mixed query selection method. There are also several methods [3, 14, 46] designed to tackle the instability of the training procedure. Our work follows the query-based detection paradigm and extends it to 3D space with temporal information.

2.2. Monocular 3D Object Detection

Monocular 3D object detection takes one single image as input and outputs predicted 3D bounding boxes of objects. A significant challenge in monocular 3D object detection is how to transfer 2D features to 3D space. Several works [53, 43, 51, 39] incorporates depth information to deal with this problem. Pseudo LiDAR [53] first estimates the depth of input images and constructs pseudo point clouds. The pseudo point clouds are then sent to a LiDAR-based detection module to predict the 3D boxes of interested objects. CaDDN [43] further proposes a fully differentiable end-to-end network which learns pixel-wise categorical depth distributions to predict appropriate depth intervals in 3D space. Inspired by FCOS [47], FCOS3D [51] projects 3D coordinates onto 2D images and decouples them as 2D attributes (centerness and classification) and 3D attributes (depths, sizes, and orientations).

2.3. 3D Object Detection in BEV

Bird’s-eye-view (BEV) object detection [38, 42, 13, 54, 31, 58, 25, 55, 40] aims at detecting objects in BEV space given either single-view or multi-view 2D images. Early works [44, 53, 43] typically transformed 2D features to BEV space based on single-view images and conducted monocular 3D object detection. LSS [42] takes six 2D images of different views as input and transforms them into 3D space based on depth estimation. Based on LSS, BEVDet [13] lifts 2D feature to BEV space and uses a BEV encoder with residual blocks and FPN to further encode the BEV features. BEVFormer [25] proposes a spatio-temporal transformer encoder that projects multi-view and multi-timestamp input to BEV representations. To ease the optimization, BEVFormerV2 [55] introduces perspective view supervision and supervises monocular 3D object detection parallel with BEV object detection. SOLOFusion [40] defines the localization potential for depth estimation and fuses short-term, high-resolution and long-term, low-resolution temporal stereo.
Figure 2: The overall architecture of SparseBEV, a fully-sparse camera-only 3D object detector. Queries are initialized to be a sparse set of pillars in BEV space. The scale-adaptive self attention further encodes the queries with adaptive receptive fields. Next, multi-view and multi-timestamp features are aggregated with adaptive spatio-temporal sampling and decoded by adaptive mixing. The decoder repeats \( L \) times to produce final predictions.

Inspired by DETR, another line of works [54, 31, 32] explore the sparse query-based paradigm. DETR3D [54] proposes a top-down framework starting from a learnable sparse set of reference points and refining them iteratively via 3D-to-2D queries. However, such 3D-to-2D projection hinders the receptive field of the query. To handle this, PETR series [31, 32] use global attention for the interaction between queries and image features, and introduce 3D positional embeddings to encode 2D features into 3D representation without explicit projection. Although achieving notable improvements, the expensive dense global attention buries the advantages of the sparse paradigm and makes it difficult to utilize long-term temporal information efficiently. In contrast, we keep the fully sparse design of DETR3D and boost the performance by enhancing the adaptability of the detector.

3. SparseBEV

As shown in Fig. 2, SparseBEV is a query-based one-stage detector with \( L \) decoder layers. The input multi-camera videos are processed frame-by-frame using image backbone and FPN. Next, a set of sparse pillar queries are initialized in BEV space and aggregated by scale-adaptive self attention. These queries then interact with the image features via adaptive spatio-temporal sampling and adaptive mixing to make 3D object predictions. We also propose a dual-branch version of SparseBEV to further enhance the temporal modeling.

3.1. Query Formulation

We first define a set of learnable queries, where each of them is represented by its translation \([x, y, z]\), dimension \([w, l, h]\), rotation \(\theta\) and velocity \([v_x, v_y]\). The queries are initialized to be pillars in BEV space where \(z\) is set to 0 and \(h\) is set to \(\sim 4m\). The initial velocity is set to \([v_x, v_y] = 0\). Other parameters \((x, y, w, l, \theta)\) are drawn from random gaussian distributions. Following Sparse R-CNN[45], we attach a \(D\)-dim query feature to each query box to encode the rich instance characteristics.

3.2. Scale-adaptive Self Attention

As mentioned above, dense BEV-based methods typically use a BEV encoder to encode multi-scale BEV features. However, in our method, since we do not explicitly build a BEV feature, how to aggregate multi-scale features in BEV space is a challenge.

In this work, we argue that the self attention can play the role of BEV encoder, since queries are defined in BEV space. The vanilla multi-head self attention has global receptive field and lacks the ability of local multi-scale aggregation. Thus, we propose scale-adaptive self attention (SASA), which learns appropriate receptive fields under the guidance of queries. First, we compute the all-pair distances \(D \in \mathbb{R}^{N \times N}\) (\(N\) is the number of queries) between the query centers in BEV space:

\[
D_{i,j} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2},
\]  

(1)
where $x_i$ and $y_i$ denotes the center of the $i$-th query. The attention considers not only the similarity between query features, but the distance between them as well. A toy example below shows how it works:

$$\text{Attn}(Q, K, V) = \text{Softmax}(\frac{QK^T}{\sqrt{d}} - \tau D)V, \quad (2)$$

where $Q, K, V \in \mathbb{R}^{N \times d}$ is the query itself and $d$ is the channel dimension. $\tau$ is a scalar to control the receptive field for each query. When $\tau = 0$, it degrades to the vanilla self attention with global receptive field. As $\tau$ grows, the attention weights for distant queries becomes smaller and the receptive field narrows.

In practice, the receptive field controller $\tau$ is adaptive to each query and specific to each head. Supposing there are $H$ heads, we use a linear transformation to generate head-specific $\tau_1, \tau_2, ..., \tau_H$ from the given query $q \in \mathbb{R}^d$:

$$[\tau_1, \tau_2, ..., \tau_H] = \text{Linear}_{d \rightarrow H}(q) \in \mathbb{R}^H, \quad (3)$$

where the weights are shared across different queries.

In our experiments, we surprisingly find that the $\tau$ for each head is learnt to uniformly distribute in a certain range regardless of the initialization. In Fig. 3, we sort the heads according to $\tau$ and visualize the attention weights for the distance part. As we can see, each head learns a different receptive field from each other, enabling the network to aggregate features in a multi-scale manner like FPN.

Scale-adaptive self attention (SASA) demonstrates the necessity of FPN, while being more flexible as it learns the scales adaptively from the query. We also find an interesting phenomenon that different categories of queries have different sizes of receptive field. For example, queries representing the bus have larger receptive field than those representing the pedestrians. More details can be found in the ablation studies.

### 3.3. Adaptive Spatio-temporal Sampling

For each frame, we use a linear layer to generate a set of sampling offsets $\{\{\Delta x_i, \Delta y_i, \Delta z_i\}\}$ adaptively from the query feature. These offsets are transformed to 3D sampling points based on the query pillar:

$$\begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} w \cdot \Delta x_i \\ l \cdot \Delta y_i \\ h \cdot \Delta z_i \end{bmatrix} + \begin{bmatrix} x \\ y \\ z \end{bmatrix}. \quad (4)$$

Compared with the deformable attention in BEVFormer, our sampling points are adaptive to both query pillar and query feature, thus better covering objects with varying sizes. Besides, these points are not restricted to the query, since we do not limit the range of the sampling offsets.

Next, we perform temporal alignment by warping the sampling points according to motions. In autonomous driving, there are two types of motion: one is ego-motion and the other is object motion. Ego-motion describes the motion of the car from its own perspective as it navigates through the environment, while object motion refers to the movement of other objects in the environment as they move around the self-driving car.

#### Dealing with Object Motion

As mentioned above, instantaneous velocity can be equal to average velocity for a short time window in self-driving. Thus, we adaptively warp the sampling points to previous timestamps using the velocity vector $[v_x, v_y]$ from the query:

$$\begin{align*}
x_{t,i} &= x_i + v_x \cdot (T_t - T_0) \\
y_{t,i} &= y_i + v_y \cdot (T_t - T_0) \\
z_{t,i} &= z_i
\end{align*} \quad (5, 6)$$

where $T_t$ denotes the timestamp of previous frame $t$ ($T_0$ denotes the current timestamp). Note that $z_{t,i}$ is identical to $z_i$ because the velocity vector is defined in BEV plane.

#### Dealing with Ego Motion

Next, we warp the sampling points based on the ego pose provided by the dataset. Points are first transformed to the global coordinate system and then to the local coordinate system of frame $t$:

$$\begin{bmatrix} x'_{t,i} \\ y'_{t,i} \\ z'_{t,i} \end{bmatrix} = E_t E_0^{-1} \begin{bmatrix} x_{t,i} \\ y_{t,i} \\ z_{t,i} \end{bmatrix}^T, \quad (7)$$

where $E_t = [R|\mathbf{t}]$ is the ego pose of frame $t$.

#### Sampling

For each timestamp, we project the warped sampling points $\{(x'_{t,i}, y'_{t,i}, z'_{t,i})\}$ to each view using the provided camera intrinsics and extrinsics. Since there are overlaps between adjacent views, the projected point may hit one or more views, which are termed as $V$. For each hit view $k$, we have multi-scale feature maps $\{F_{k,j}\} \in \{1, 2, ..., N_{\text{feat}}\}$ from the image backbone. Features are
Table 1: Performance comparison on the nuScenes val split. † benefits from perspective pretraining. ‡ indicates methods with CBGS [59] which will elongate 1 epoch into 4.5 epochs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Backbone</th>
<th>Input Size</th>
<th>Epochs</th>
<th>NDS</th>
<th>mAP</th>
<th>mATE</th>
<th>mASE</th>
<th>mAOE</th>
<th>mAVE</th>
<th>mAAE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PETRv2 [32]</td>
<td>ResNet50</td>
<td>704 × 256</td>
<td>60</td>
<td>45.6</td>
<td>34.9</td>
<td>0.700</td>
<td>0.275</td>
<td>0.580</td>
<td>0.437</td>
<td>0.187</td>
</tr>
<tr>
<td>BEVStereo [22]</td>
<td>ResNet50</td>
<td>704 × 256</td>
<td>90 ‡</td>
<td>50.0</td>
<td>37.2</td>
<td>0.598</td>
<td>0.270</td>
<td>0.438</td>
<td>0.367</td>
<td>0.190</td>
</tr>
<tr>
<td>BEVPoolv2 [12]</td>
<td>ResNet50</td>
<td>704 × 256</td>
<td>90 ‡</td>
<td>52.6</td>
<td>40.6</td>
<td>0.572</td>
<td>0.275</td>
<td>0.463</td>
<td>0.275</td>
<td>0.188</td>
</tr>
<tr>
<td>SOLOFusion [40]</td>
<td>ResNet50</td>
<td>704 × 256</td>
<td>90 ‡</td>
<td>53.4</td>
<td>42.7</td>
<td>0.567</td>
<td>0.274</td>
<td>0.511</td>
<td>0.252</td>
<td>0.181</td>
</tr>
<tr>
<td>Sparse4Dv2 [29]</td>
<td>ResNet50</td>
<td>704 × 256</td>
<td>100</td>
<td>53.9</td>
<td>43.9</td>
<td>0.598</td>
<td>0.270</td>
<td>0.475</td>
<td>0.282</td>
<td>0.179</td>
</tr>
<tr>
<td>StreamPETR † [50]</td>
<td>ResNet50</td>
<td>704 × 256</td>
<td>60</td>
<td>55.0</td>
<td>45.0</td>
<td>0.613</td>
<td>0.267</td>
<td>0.413</td>
<td>0.265</td>
<td>0.196</td>
</tr>
<tr>
<td>SparseBEV</td>
<td>ResNet50</td>
<td>704 × 256</td>
<td>36</td>
<td>54.5</td>
<td>43.2</td>
<td>0.606</td>
<td>0.274</td>
<td>0.387</td>
<td>0.251</td>
<td>0.186</td>
</tr>
<tr>
<td>SparseBEV †</td>
<td>ResNet50</td>
<td>704 × 256</td>
<td>36</td>
<td><strong>55.8</strong></td>
<td>44.8</td>
<td>0.581</td>
<td>0.271</td>
<td>0.373</td>
<td>0.247</td>
<td>0.190</td>
</tr>
<tr>
<td>DETR3D † [54]</td>
<td>ResNet101-DCN</td>
<td>1600 × 900</td>
<td>24</td>
<td>43.4</td>
<td>34.9</td>
<td>0.716</td>
<td>0.268</td>
<td>0.379</td>
<td>0.842</td>
<td>0.200</td>
</tr>
<tr>
<td>BEVFormer † [25]</td>
<td>ResNet101-DCN</td>
<td>1600 × 900</td>
<td>24</td>
<td>51.7</td>
<td>41.6</td>
<td>0.673</td>
<td>0.274</td>
<td>0.372</td>
<td>0.394</td>
<td>0.198</td>
</tr>
<tr>
<td>BEVDepth [24]</td>
<td>ResNet101</td>
<td>1408 × 512</td>
<td>90 ‡</td>
<td>53.5</td>
<td>41.2</td>
<td>0.565</td>
<td>0.266</td>
<td>0.358</td>
<td>0.331</td>
<td>0.190</td>
</tr>
<tr>
<td>Sparse4D † [28]</td>
<td>ResNet101-DCN</td>
<td>1600 × 900</td>
<td>48</td>
<td>55.0</td>
<td>44.4</td>
<td>0.603</td>
<td>0.276</td>
<td>0.360</td>
<td>0.309</td>
<td>0.178</td>
</tr>
<tr>
<td>SOLOFusion [40]</td>
<td>ResNet101</td>
<td>1408 × 512</td>
<td>90 ‡</td>
<td>58.2</td>
<td>48.3</td>
<td>0.503</td>
<td>0.264</td>
<td>0.381</td>
<td>0.246</td>
<td>0.207</td>
</tr>
<tr>
<td>SparseBEV †</td>
<td>ResNet101</td>
<td>1408 × 512</td>
<td>24</td>
<td><strong>59.2</strong></td>
<td><strong>50.1</strong></td>
<td>0.562</td>
<td>0.265</td>
<td>0.321</td>
<td>0.243</td>
<td>0.195</td>
</tr>
</tbody>
</table>

Point Mixing. Next, we then transpose the feature and apply the dynamic weights to the point dimension of it:

\[
W_p = \text{Linear}(q) \in \mathbb{R}^{P \times P} \tag{11}
\]

\[
M_p(f) = \text{ReLU}(\text{LayerNorm}(f^T W_p)), \tag{12}
\]

where \(W_p\) is the dynamic weights and is shared across different channels.

After channel mixing and point mixing, the spatio-temporal features are flattened and aggregated by a linear layer. The final regression and classification predictions are computed by two MLPs respectively.

3.5. Dual-branch SparseBEV

Inspired by SlowFast [6], we further enhance the long-term temporal modeling by dividing the input video into a slow stream and a fast stream, resulting in a dual-branch SparseBEV. The slow stream is designed to capture fine-grained appearance details and it operates at low frame rates and high resolutions. The fast stream is responsible for capturing long-term temporal stereo and it operates at high frame rates and low resolutions. Sampling points are projected to the two streams respectively and the sampled features are stacked before adaptive mixing.

In this way, we decouple the learning of static appearance and temporal motion, leading to better performance. Besides, the computation cost is significantly reduced since only a small fraction of frames needs to be processed at high resolution. However, since this dual-branch design makes the framework a little complex, we do not use it unless otherwise stated. The ablations of this part can be found in the supplementary material.
### 4. Experiments

#### 4.1. Implementation Details

We implement our model using PyTorch [41]. Following previous methods [54, 25, 32, 40], we adopt common image backbones including ResNet [10] and V2-99 [20]. The decoder consists of 6 layers and weights are shared across different layers. By default, we use $T = 8$ frames in total and the interval between adjacent frames is about 0.5s.

During training, we adopt the Hungarian algorithm [18] for label assignment between ground-truth objects and predictions. Focal loss [27] is used for classification and L1 loss is used for 3D bounding box regression. We use the AdamW [35] optimizer with a global batch size of 8. The initial learning rate is set to $2 \times 10^{-4}$ and is decayed with cosine annealing policy.

Recently, we follow the training setting of the very recent work StreamPETR [50] and refresh our results for fair comparison. For the regression loss, we change the loss weight of $x$ and $y$ to 2.0 and leave the others to 1.0. Query denoising [21] is also introduced to stabilize training and speedup convergence.

#### 4.2. Datasets and Metrics

We evaluate our model on the nuScenes dataset [1], which consists of large-scale multimodal data collected from 6 surround-view cameras, 1 lidar and 5 radars. The dataset has 1000 videos and is split into 700/150/150 videos for training/validation/testing. Each video has roughly 20s duration and the key samples are annotated every 0.5s.

For 3D object detection, there are up to 1.4M annotated 3D bounding boxes of 10 classes. The official evaluation metrics include the well-known mean Average Precision (mAP) and five true positive (TP) metrics, including ATE, ASE, AOE, AVE, and AAE for measuring translation, scale, orientation, velocity, and attribute errors respectively. The overall performance is measured by the nuScenes Detection Score (NDS), which is the composite of the above metrics.

#### 4.3. Comparison with the State-of-the-art Methods

**nuScenes val split.** In Tab. 1, we compare SparseBEV with previous state-of-the-art methods on the validation split of nuScenes. Unless otherwise stated, the image backbone is pretrained on ImageNet-1k [17] and the number of queries is set to 900. To keep the simplicity of our approach, the dual-branch design is not used here. When adopting ResNet50 as the backbone and $704 \times 256$ as the input size, SparseBEV outperforms the previous state-of-the-art method SOLOFusion by 1.2 mAP and 1.4 NDS. By further adopting nuImages [1] pretraining and reducing the number queries to 400, SparseBEV sets a new record of 55.8 NDS while maintaining a real-time inference speed of 23.5 FPS (corresponding to Fig. 1). Next, we upgrade the backbone to ResNet101 and scale the input size up to $1408 \times 512$. Under this setting, SparseBEV still surpasses SOLOFusion by 1.8 mAP and 1.0 NDS, demonstrating the scalability of our method.

**nuScenes test split.** We submit our method to the website of nuScenes and report the leaderboard in Tab. 2. Using the V2-99 [20] backbone pretrained by DD3D [39], SparseBEV achieves 62.7 NDS and 54.3 mAP without bells...
Table 3: Ablations on query formulation. Compared with reference points, pillars introduce better spatial priors and lead to better performance.

<table>
<thead>
<tr>
<th>Query Formulation</th>
<th>NDS</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Reference Points</td>
<td>55.1</td>
<td>44.0</td>
</tr>
<tr>
<td>BEV Pillars</td>
<td><strong>55.6</strong></td>
<td><strong>45.4</strong></td>
</tr>
</tbody>
</table>

Table 4: Comparison between vanilla multi-head self attention (MHSA) and scale-adaptive self attention (SASA). Our SASA achieves significant improvements over the baseline.

<table>
<thead>
<tr>
<th>Self Attention</th>
<th>Distance Function</th>
<th>NDS</th>
<th>mAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MHSA</td>
<td>-</td>
<td>53.4</td>
<td>41.4</td>
</tr>
<tr>
<td>SASA</td>
<td>$\tau D^2$</td>
<td>54.3</td>
<td>43.8</td>
</tr>
<tr>
<td></td>
<td>$\tau D$</td>
<td><strong>55.6</strong></td>
<td><strong>45.4</strong></td>
</tr>
<tr>
<td></td>
<td>$\tau \sqrt{D}$</td>
<td>55.1</td>
<td>44.3</td>
</tr>
</tbody>
</table>

Figure 4: Averaged $\tau$ over all queries and all heads for each class in SASA. Larger $\tau$ indicates smaller receptive fields. We perform statistics on the val split of nuScenes and choose the queries with a confidence score over 0.3.

4.4. Ablation Studies

In this section, we conduct ablations on the validation split of nuScenes. For all experiments, we use ResNet50 pretrained on nuImages [1] as the image backbone. The input contains 8 frames with $704 \times 256$ resolution. We use 900 queries in the decoder and the model is trained for 24 epochs without CBGS [59].

Query Formulation. In Tab. 3, we ablate different formulations of the query. The top row uses a set of reference points distributing uniformly in 3D space (e.g. DETR3D and PETR). By replacing the reference points with pillars in BEV space, we observe performance improvements (+0.5 NDS) over the baseline. This is because pillars introduce better spatial priors than reference points.

Scale-adaptive Self Attention. We study the effect of scale-adaptive self attention (SASA) in Tab. 4. Compared with the vanilla multi-head self attention, SASA achieves +4.0 mAP and +2.2 NDS improvements over the baseline. We further ablate different distance functions and find that $L_2$ distance works best. Besides, we also observe two interesting phenomena in SASA. First, different heads learn a different receptive field (see Fig. 3), enabling the model to aggregate multi-scale features in BEV space. Second, queries representing larger objects tend to have larger receptive field. In Fig. 4, we average the receptive field coefficient $\tau$ over all queries and all heads for each class. As we can see, the receptive field of large objects (such as bus and truck) is larger than small objects (such as pedestrian and traffic cone), demonstrating the effectiveness of our adaptive design.

Adaptive Spatio-temporal Sampling. In Fig. 5, we ablate the number of frames and sampling points. The performance continues to increase as the number of frames increases, proving that SparseBEV can benefit from the long-term history. Here, we use 8 frames for fair comparison with previous methods. As for the number of sampling points, we observe that dispatching 16 points for each frame leads to the best performance. We also provide the visualization of the sampling points from the last stage of the decoder in Fig. 6. Our sampling scheme has adaptive regions of interest and achieves good temporal alignments for both static and moving objects across different frames.
Figure 6: Visualization of adaptive spatio-temporal sampling over three consecutive frames. Different instances are distinguished by colors. Point size indices the depth: larger points are closer to the camera. Our sampling scheme has an adaptive receptive field and is well aligned across different timestamps.

<table>
<thead>
<tr>
<th>Ego Align</th>
<th>Obj. Align</th>
<th>NDS</th>
<th>mAP</th>
<th>mAVE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>44.4</td>
<td>34.1</td>
<td>0.510</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td>54.2</td>
<td>43.5</td>
<td>0.281</td>
</tr>
<tr>
<td>√</td>
<td>√</td>
<td>55.6</td>
<td>45.4</td>
<td>0.243</td>
</tr>
</tbody>
</table>

Table 5: Ablations on temporal alignment. Aligning both ego and object motion leads to the best performance.

Temporal Alignment. In Tab. 5, we validate the necessity of temporal alignment in spatio-temporal sampling. In our method, ego motion is aligned with the provided ego pose, while object motion is aligned with a simple constant velocity motion model. As we can see from the table, both of them contribute to the performance.

Adaptive Mixing. In Tab. 6, we validate the design of the adaptive mixing. The top row is a baseline that uses attention weights to aggregate sampled features (as done in DETR3D). Static mixing and adaptive mixing improve the baseline by 2.7 NDS and 6.5 NDS respectively, demonstrating the necessity of the adaptive design. Next, we explore different combinations of channel and point mixing. Channel mixing followed by point mixing leads to better performance, proving that the object semantics should be enhanced before point mixing.

4.5. Limitations and Future Work

One limitation of SparseBEV is the heavy reliance on ego pose. As we can see from Tab. 5, the performance drops about 10 NDS without ego-based temporal alignment. However, in the real world, the ego pose provided by IMU may be unreliable and inaccurate, seriously affecting the robustness. Another limitation is that the inference latency increases linearly with the number of frames, since we stack the sampled features along the temporal dimension.

In the future, we will explore a more elegant and concise way of decoupling spatial appearance and temporal motion. We will also try to extend SparseBEV to other 3D perception tasks such as BEV segmentation, occupancy prediction and lane detection.

5. Conclusion

In this paper, we have proposed a query-based one-stage 3D object detector, named SparseBEV, which can enjoy the benefits of the BEV space without explicitly constructing a dense BEV feature. SparseBEV improves the adaptability of the decoder by three key modules: scale-adaptive self attention, adaptive spatio-temporal sampling and adaptive mixing. We further introduce a dual-branch design to enhance the long-term temporal modeling. Experiments show that SparseBEV achieves the state-of-the-art performance on the dataset of nuScenes for both speed and accuracy. We hope this exciting result will attract more attention to the fully sparse BEV detection paradigm.

Acknowledgements. This work is supported by National Key R&D Program of China (No. 2022ZD0160900), National Natural Science Foundation of China (No. 62076119, No. 61921006), Fundamental Research Funds for the Central Universities (No. 020214380091, No. 020214380099), and Collaborative Innovation Center of Novel Software Technology and Industrialization. Besides, the authors would like to thank Ziteng Gao, Zhiqi Li and Ruopeng Gao for their help.
References


[34] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 11976–11986, 2022. 6


[53] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-


