BEVFormer: Learning Bird's-Eye-View Representation from Multi-Camera Images via Spatiotemporal Transformers

Zhiqi Li[1,2*], **Wenhai Wang**[2*], **Hongyang Li**[2*], **Enze Xie**[3], **Chonghao Sima**[2], **Tong Lu**[1], **Yu Qiao**[2], **Jifeng Dai**[2✉]

[1]Nanjing University [2]Shanghai AI Laboratory [3]The University of Hong Kong

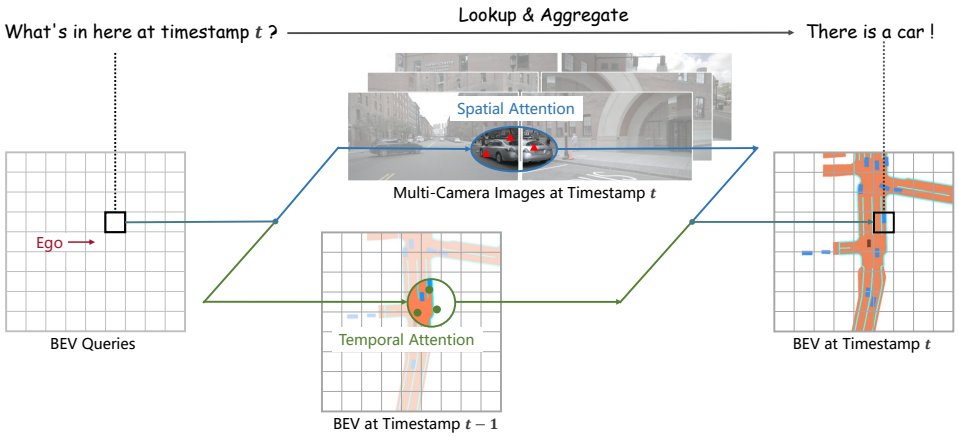

Figure 1: We propose **BEVFormer**, a paradigm for autonomous driving that applies both Transformer and Temporal structure to generate bird's-eye-view (BEV) features from multi-camera inputs. BEVFormer leverages queries to lookup spatial/temporal space and aggregate spatiotemporal information correspondingly, hence benefiting stronger representations for perception tasks.

Abstract

3D visual perception tasks, including 3D detection and map segmentation based on multi-camera images, are essential for autonomous driving systems. In this work, we present a new framework termed BEVFormer, which learns unified BEV representations with spatiotemporal transformers to support multiple autonomous driving perception tasks. In a nutshell, BEVFormer exploits both spatial and temporal information by interacting with spatial and temporal space through predefined grid-shaped BEV queries. To aggregate spatial information, we design spatial cross-attention that each BEV query extracts the spatial features from the regions of interest across camera views. For temporal information, we propose temporal self-attention to recurrently fuse the history BEV information. Our approach achieves the new state-of-the-art 56.9% in terms of NDS metric on the nuScenes `test` set, which is 9.0 points higher than previous best arts and on par with the performance of LiDAR-based baselines. We further show that BEVFormer remarkably improves the accuracy of velocity estimation and recall of objects under low visibility conditions. The code is available at https://github.com/zhiqi-li/BEVFormer.

*: Equal contribution. This work is done when Zhiqi Li is an intern at Shanghai AI Lab.
✉: Corresponding author.

1 Introduction

Perception in 3D space is critical for various applications such as autonomous driving, robotics, *etc*. Despite the remarkable progress of LiDAR-based methods [43, 20, 54, 50, 8], camera-based approaches [45, 32, 47, 30] have attracted extensive attention in recent years. Apart from the low cost for deployment, cameras own the desirable advantages to detect long-range distance objects and identify vision-based road elements (*e.g.*, traffic lights, stoplines), compared to LiDAR-based counterparts.

Visual perception of the surrounding scene in autonomous driving is expected to predict the 3D bounding boxes or the semantic maps from 2D cues given by multiple cameras. The most straightforward solution is based on the monocular frameworks [45, 44, 31, 35, 3] and cross-camera post-processing. The downside of this framework is that it processes different views separately and cannot capture information across cameras, leading to low performance and efficiency [32, 47].

As an alternative to the monocular frameworks, a more unified framework is extracting holistic representations from multi-camera images. The bird's-eye-view (BEV) is a commonly used representation of the surrounding scene since it clearly presents the location and scale of objects and is suitable for various autonomous driving tasks, such as perception and planning [29]. Although previous map segmentation methods demonstrate BEV's effectiveness [32, 18, 29], BEV-based approaches have not shown significant advantages over other paradigm in 3D object detections [47, 31, 34]. The underlying reason is that the 3D object detection task requires strong BEV features to support accurate 3D bounding box prediction, but generating BEV from the 2D planes is ill-posed. A popular BEV framework that generates BEV features is based on depth information [46, 32, 34], but this paradigm is sensitive to the accuracy of depth values or the depth distributions. The detection performance of BEV-based methods is thus subject to compounding errors [47], and inaccurate BEV features can seriously hurt the final performance. Therefore, *we are motivated to design a BEV generating method that does not rely on depth information and can learn BEV features adaptively rather than strictly rely on 3D prior.* Transformer, which uses an attention mechanism to aggregate valuable features dynamically, meets our demands conceptually.

Another motivation for using BEV features to perform perception tasks is that BEV is a desirable bridge to connect temporal and spatial space. For the human visual perception system, temporal information plays a crucial role in inferring the motion state of objects and identifying occluded objects, and many works in vision fields have demonstrated the effectiveness of using video data [2, 27, 26, 33, 19]. However, the existing state-of-the-art multi-camera 3D detection methods rarely exploit temporal information. The significant challenges are that autonomous driving is time-critical and objects in the scene change rapidly, and thus simply stacking BEV features of cross timestamps brings extra computational cost and interference information, which might not be ideal. Inspired by recurrent neural networks (RNNs) [17, 10], *we utilize the BEV features to deliver temporal information from past to present recurrently, which has the same spirit as the hidden states of RNN models.*

To this end, we present a transformer-based bird's-eye-view (BEV) encoder, termed **BEVFormer**, which can effectively aggregate spatiotemporal features from multi-view cameras and history BEV features. The BEV features generated from the BEVFormer can simultaneously support multiple 3D perception tasks such as 3D object detection and map segmentation, which is valuable for the autonomous driving system. As shown in Fig. 1, our BEVFormer contains three key designs, which are (1) grid-shaped BEV queries to fuse spatial and temporal features via attention mechanisms flexibly, (2) spatial cross-attention module to aggregate the spatial features from multi-camera images, and (3) temporal self-attention module to extract temporal information from history BEV features, which benefits the velocity estimation of moving objects and the detection of heavily occluded objects, while bringing negligible computational overhead. With the unified features generated by BEVFormer, the model can collaborate with different task-specific heads such as Deformable DETR [56] and mask decoder [22], for end-to-end 3D object detection and map segmentation.

Our main contributions are as follows:

• We propose BEVFormer, a spatiotemporal transformer encoder that projects multi-camera and/or timestamp input to BEV representations. With the unified BEV features, our model can simultaneously support multiple autonomous driving perception tasks, including 3D detection and map segmentation.

• We designed learnable BEV queries along with a spatial cross-attention layer and a temporal self-attention layer to lookup spatial features from cross cameras and temporal features from history BEV, respectively, and then aggregate them into unified BEV features.

• We evaluate the proposed BEVFormer on multiple challenging benchmarks, including nuScenes [4] and Waymo [40]. Our BEVFormer consistently achieves improved performance compared to the prior arts. For example, under a comparable parameters and computation overhead, BEVFormer achieves 56.9% NDS on nuScenes `test` set, outperforming previous best detection method DETR3D [47] by 9.0 points (56.9% *vs.* 47.9%). For the map segmentation task, we also achieve the state-of-the-art performance, more than 5.0 points higher than Lift-Splat [32] on the most challenging lane segmentation. We hope this straightforward and strong framework can serve as a new baseline for following 3D perception tasks.

2 Related Work

2.1 Transformer-based 2D perception

Recently, a new trend is to use transformer to reformulate detection and segmentation tasks [7, 56, 22].

DETR [7] uses a set of object queries to generate detection results by the cross-attention decoder directly. However, the main drawback of DETR is the long training time. Deformable DETR [56] solves this problem by proposing deformable attention. Different from vanilla global attention in DETR, the deformable attention interacts with local regions of interest, which only samples K points near each reference point and calculates attention results, resulting in high efficiency and significantly shortening the training time. The deformable attention mechanism is calculated by:

$$\text{DeformAttn}(q, p, x) = \sum_{i=1}^{N_{\text{head}}} \mathcal{W}_i \sum_{j=1}^{N_{\text{key}}} \mathcal{A}_{ij} \cdot \mathcal{W}'_i x(p + \Delta p_{ij}), \tag{1}$$

where q, p, x represent the query, reference point and input features, respectively. i indexes the attention head, and N_{head} denotes the total number of attention heads. j indexes the sampled keys, and N_{key} is the total sampled key number for each head. $W_i \in \mathbb{R}^{C \times (C/H_{\text{head}})}$ and $W'_i \in \mathbb{R}^{(C/H_{\text{head}}) \times C}$ are the learnable weights, where C is the feature dimension. $A_{ij} \in [0, 1]$ is the predicted attention weight, and is normalized by $\sum_{j=1}^{N_{\text{key}}} A_{ij} = 1$. $\Delta p_{ij} \in \mathbb{R}^2$ are the predicted offsets to the reference point p. $x(p + \Delta p_{ij})$ represents the feature at location $p + \Delta p_{ij}$, which is extracted by bilinear interpolation as in Dai *et al.* [12]. In this work, we extend the deformable attention to 3D perception tasks, to efficiently aggregate both spatial and temporal information.

2.2 Camera-based 3D Perception

Previous 3D perception methods typically perform 3D object detection or map segmentation tasks independently. For the 3D object detection task, early methods are similar to 2D detection methods [1, 28, 49, 39, 53], which usually predict the 3D bounding boxes based on 2D bounding boxes. Wang *et al.* [45] follows an advanced 2D detector FCOS [41] and directly predicts 3D bounding boxes for each object. DETR3D [47] projects learnable 3D queries in 2D images, and then samples the corresponding features for end-to-end 3D bounding box prediction without NMS post-processing. Another solution is to transform image features into BEV features and predict 3D bounding boxes from the top-down view. Methods transform image features into BEV features with the depth information from depth estimation [46] or categorical depth distribution [34]. OFT [36] and ImVoxelNet [37] project the predefined voxels onto image features to generate the voxel representation of the scene. Recently, M^{2}BEV [48] futher explored the feasibility of simultaneously performing multiple perception tasks based on BEV features.

Actually, generating BEV features from multi-camera features is more extensively studied in map segmentation tasks [32, 30]. A straightforward method is converting perspective view into the BEV through Inverse Perspective Mapping (IPM) [35, 5]. In addition, Lift-Splat [32] generates the BEV features based on the depth distribution. Methods [30, 16, 9] utilize multilayer perceptron to learn the translation from perspective view to the BEV. PYVA [51] proposes a cross-view transformer that converts the front-view monocular image into the BEV, but this paradigm is not suitable for

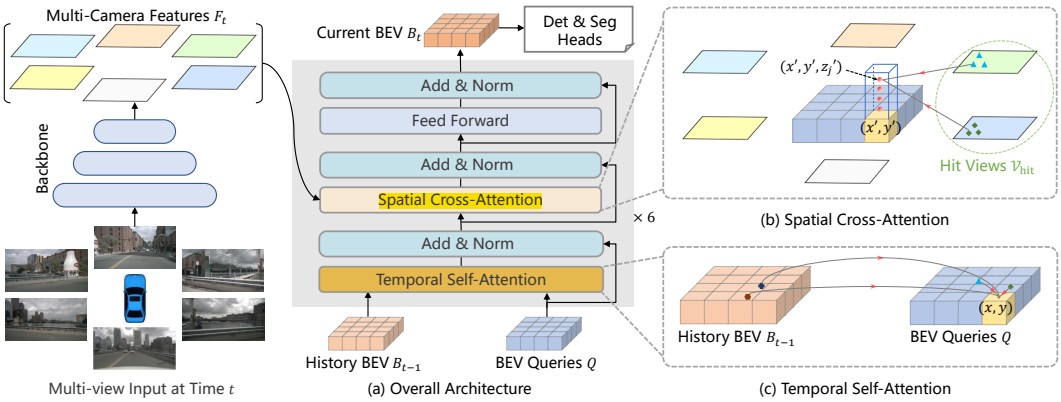

Figure 2: **Overall architecture of BEVFormer.** (a) The encoder layer of BEVFormer contains grid-shaped BEV queries, temporal self-attention, and spatial cross-attention. (b) In spatial cross-attention, each BEV query only interacts with image features in the regions of interest. (c) In temporal self-attention, each BEV query interacts with two features: the BEV queries at the current timestamp and the BEV features at the previous timestamp.

fusing multi-camera features due to the computational cost of global attention mechinism [42]. In addition to the spatial information, previous works [18, 38, 6] also consider the temporal information by stacking BEV features from several timestamps. Stacking BEV features constraints the available temporal information within fixed time duration and brings extra computational cost. In this work, the proposed spatiotemporal transformer generates BEV features of the current time by considering both spatial and temporal clues, and the temporal information is obtained from the previous BEV features by the RNN manner, which only brings little computational cost.

3 BEVFormer

Converting multi-camera image features to bird's-eye-view (BEV) features can provide a unified surrounding environment representation for various autonomous driving perception tasks. In this work, we present a new transformer-based framework for BEV generation, which can effectively aggregate spatiotemporal features from multi-view cameras and history BEV features via attention mechanisms.

3.1 Overall Architecture

As illustrated in Fig. 2, BEVFormer has 6 encoder layers, each of which follows the conventional structure of transformers [42], except for three tailored designs, namely BEV queries, spatial cross-attention, and temporal self-attention. Specifically, BEV queries are grid-shaped learnable parameters, which is designed to query features in BEV space from multi-camera views via attention mechanisms. Spatial cross-attention and temporal self-attention are attention layers working with BEV queries, which are used to lookup and aggregate spatial features from multi-camera images as well as temporal features from history BEV, according to the BEV query.

During inference, at timestamp t, we feed multi-camera images to the backbone network (*e.g.*, ResNet-101 [15]), and obtain the features $F_t = \{F_t^i\}_{i=1}^{N_{\text{view}}}$ of different camera views, where F_t^i is the feature of the i-th view, N_{view} is the total number of camera views. At the same time, we preserved the BEV features B_{t-1} at the prior timestamp $t-1$. In each encoder layer, we first use BEV queries Q to query the temporal information from the prior BEV features B_{t-1} via the temporal self-attention. We then employ BEV queries Q to inquire about the spatial information from the multi-camera features F_t via the spatial cross-attention. After the feed-forward network [42], the encoder layer output the refined BEV features, which is the input of the next encoder layer. After 6 stacking encoder layers, unified BEV features B_t at current timestamp t are generated. Taking the BEV features B_t as input, the 3D detection head and map segmentation head predict the perception results such as 3D bounding boxes and semantic map.

3.2 BEV Queries

We predefine a group of grid-shaped learnable parameters $Q \in \mathbb{R}^{H \times W \times C}$ as the queries of BEVFormer, where H, W are the spatial shape of the BEV plane. To be specific, the query $Q_p \in \mathbb{R}^{1 \times C}$ located at $p = (x, y)$ of Q is responsible for the corresponding grid cell region in the BEV plane. Each grid cell in the BEV plane corresponds to a real-world size of s meters. The center of BEV features corresponds to the position of the ego car by default. Following common practices [14], we add learnable positional embedding to BEV queries Q before inputting them to BEVFormer.

3.3 Spatial Cross-Attention

Due to the large input scale of multi-camera 3D perception (containing N_{view} camera views), the computational cost of vanilla multi-head attention [42] is extremely high. Therefore, we develop the spatial cross-attention based on deformable attention [56], which is a resource-efficient attention layer where each BEV query Q_p only interacts with its regions of interest across camera views. However, deformable attention is originally designed for 2D perception, so some adjustments are required for 3D scenes.

As shown in Fig. 2 (b), we first lift each query on the BEV plane to a pillar-like query [20], sample N_{ref} 3D reference points from the pillar, and then project these points to 2D views. For one BEV query, the projected 2D points can only fall on some views, and other views are not hit. Here, we term the hit views as $\mathcal{V}_{\text{hit}}$. After that, we regard these 2D points as the reference points of the query Q_p and sample the features from the hit views $\mathcal{V}_{\text{hit}}$ around these reference points. Finally, we perform a weighted sum of the sampled features as the output of spatial cross-attention. The process of spatial cross-attention (SCA) can be formulated as:

$$\text{SCA}(Q_p, F_t) = \frac{1}{|\mathcal{V}_{\text{hit}}|} \sum_{i \in \mathcal{V}_{\text{hit}}} \sum_{j=1}^{N_{\text{ref}}} \text{DeformAttn}(Q_p, \mathcal{P}(p, i, j), F_t^i), \tag{2}$$

where i indexes the camera view, j indexes the reference points, and N_{ref} is the total reference points for each BEV query. F_t^i is the features of the i-th camera view. For each BEV query Q_p, we use a project function $\mathcal{P}(p, i, j)$ to get the j-th reference point on the i-th view image.

Next, we introduce how to obtain the reference points on the view image from the projection function $\mathcal{P}$. We first calculate the real world location (x', y') corresponding to the query Q_p located at $p = (x, y)$ of Q as Eqn. 3.

$$x' = (x - \frac{W}{2}) \times s; \quad y' = (y - \frac{H}{2}) \times s, \tag{3}$$

where H, W are the spatial shape of BEV queries, s is the size of resolution of BEV's grids, and (x', y') are the coordinates where the position of ego car is the origin. In 3D space, the objects located at (x', y') will appear at the height of z' on the z-axis. So we predefine a set of anchor heights $\{z'_j\}_{j=1}^{N_{\text{ref}}}$ to make sure we can capture clues that appeared at different heights. In this way, for each query Q_p, we obtain a pillar of 3D reference points $(x', y', z'_j)_{j=1}^{N_{\text{ref}}}$. Finally, we project the 3D reference points to different image views through the projection matrix of cameras, which can be written as:

$$\begin{aligned} \mathcal{P}(p, i, j) &= (x_{ij}, y_{ij}) \\ \text{where } z_{ij} \cdot \begin{bmatrix} x_{ij} & y_{ij} & 1 \end{bmatrix}^T &= T_i \cdot \begin{bmatrix} x' & y' & z'_j & 1 \end{bmatrix}^T. \end{aligned} \tag{4}$$

Here, $\mathcal{P}(p, i, j)$ is the 2D point on i-th view projected from j-th 3D point (x', y', z'_j), $T_i \in \mathbb{R}^{3 \times 4}$ is the known projection matrix of the i-th camera.

3.4 Temporal Self-Attention

In addition to spatial information, temporal information is also crucial for the visual system to understand the surrounding environment [27]. For example, it is challenging to infer the velocity of moving objects or detect highly occluded objects from static images without temporal clues. To address this problem, we design temporal self-attention, which can represent the current environment by incorporating history BEV features.

Given the BEV queries Q at current timestamp t and history BEV features B_{t-1} preserved at timestamp $t-1$, we first align B_{t-1} to Q according to ego-motion to make the features at the same grid correspond to the same real-world location. Here, we denote the aligned history BEV features B_{t-1} as B'_{t-1}. However, from times $t-1$ to t, movable objects travel in the real world with various offsets. It is challenging to construct the precise association of the same objects between the BEV features of different times. Therefore, we model this temporal connection between features through the temporal self-attention (TSA) layer, which can be written as follows:

$$\text{TSA}(Q_p, \{Q, B'_{t-1}\}) = \sum_{V \in \{Q, B'_{t-1}\}} \text{DeformAttn}(Q_p, p, V), \tag{5}$$

where Q_p denotes the BEV query located at $p = (x, y)$. In addition, different from the vanilla deformable attention, the offsets Δp in temporal self-attention are predicted by the concatenation of Q and B'_{t-1}. Specially, for the first sample of each sequence, the temporal self-attention will degenerate into a self-attention without temporal information, where we replace the BEV features $\{Q, B'_{t-1}\}$ with duplicate BEV queries $\{Q, Q\}$.

Compared to simply stacking BEV in [18, 38, 6], our temporal self-attention can more effectively model long temporal dependency. BEVFormer extracts temporal information from the previous BEV features rather than multiple stacking BEV features, thus requiring less computational cost and suffering less disturbing information.

3.5 Applications of BEV Features

Since the BEV features $B_t \in \mathbb{R}^{H \times W \times C}$ is a versatile 2D feature map that can be used for various autonomous driving perception tasks, the 3D object detection and map segmentation task heads can be developed based on 2D perception methods [56, 22] with minor modifications.

For 3D object detection, we design an end-to-end 3D detection head based on the 2D detector Deformable DETR [56]. The modifications include using single-scale BEV features B_t as the input of the decoder, predicting 3D bounding boxes and velocity rather than 2D bounding boxes, and only using L_1 loss to supervise 3D bounding box regression. With the detection head, our model can end-to-end predict 3D bounding boxes and velocity without the NMS post-processing.

For map segmentation, we design a map segmentation head based on a 2D segmentation method Panoptic SegFormer [22]. Since the map segmentation based on the BEV is basically the same as the common semantic segmentation, we utilize the mask decoder of [22] and class-fixed queries to target each semantic category, including the car, vehicles, road (drivable area), and lane.

3.6 Implementation Details

Training Phase. For each sample at timestamp t, we randomly sample another 3 samples from the consecutive sequence of the past 2 seconds, and this random sampling strategy can augment the diversity of ego-motion [57]. We denote the timestamps of these four samples as $t-3$, $t-2$, $t-1$ and t. For the samples of the first three timestamps, they are responsible for recurrently generating the BEV features $\{B_{t-3}, B_{t-2}, B_{t-1}\}$ and this phase requires no gradients. For the first sample at timestamp $t-3$, there is no previous BEV features, and temporal self-attention degenerate into self-attention. At the time t, the model generates the BEV features B_t based on both multi-camera inputs and the prior BEV features B_{t-1}, so that B_t contains the temporal and spatial clues crossing the four samples. Finally, we feed the BEV features B_t into the detection and segmentation heads and compute the corresponding loss functions.

Inference Phase. During the inference phase, we evaluate each frame of the video sequence in chronological order. The BEV features of the previous timestamp are saved and used for the next, and this online inference strategy is time-efficient and consistent with practical applications. Although we utilize temporal information, our inference speed is still comparable with other methods [45, 47].

4 Experiments

4.1 Datasets

We conduct experiments on two challenging public autonomous driving datasets, namely nuScenes dataset [4] and Waymo open dataset [40].

The nuScenes dataset [4] contains 1000 scenes of roughly 20s duration each, and the key samples are annotated at 2Hz. Each sample consists of RGB images from 6 cameras and has 360° horizontal FOV. For the detection task, there are 1.4M annotated 3D bounding boxes from 10 categories. We follow the settings in [32] to perform BEV segmentation task. This dataset also provides the official evaluation metrics for the detection task. The mean average precision (mAP) of nuScenes is computed using the center distance on the ground plane rather than the 3D Intersection over Union (IoU) to match the predicted results and ground truth. The nuScenes metrics also contain 5 types of true positive metrics (TP metrics), including ATE, ASE, AOE, AVE, and AAE for measuring translation, scale, orientation, velocity, and attribute errors, respectively. The nuScenes also defines a nuScenes detection score (NDS) as $\mathrm{NDS}=\frac{1}{10}[5\mathrm{mAP}+\sum_{\mathrm{mTP}\in\mathbb{TP}}(1-\min(1,\mathrm{mTP}))]$ to capture all aspects of the nuScenes detection tasks.

Waymo Open Dataset [40] is a large-scale autonomous driving dataset with 798 training sequences and 202 validation sequences. Note that the five images at each frame provided by Waymo have only about 252° horizontal FOV, but the provided annotated labels are 360° around the ego car. We remove these bounding boxes that can not be visible on any images in training and validation sets. Due to the Waymo Open Dataset being large-scale and high-rate [34], we use a subset of the training split by sampling every 5^{th} frame from the training sequences and only detect the vehicle category. We use the thresholds of 0.5 and 0.7 for 3D IoU to compute the mAP on Waymo dataset.

4.2 Experimental Settings

Following previous methods [45, 47, 31], we adopt two types of backbone: ResNet101-DCN [15, 12] that initialized from FCOS3D [45] checkpoint, and VoVnet-99 [21] that initialized from DD3D [31] checkpoint. By default, we utilize the output multi-scale features from FPN [23] with sizes of 1/16, 1/32, 1/64 and the dimension of $C=256$. For experiments on nuScenes, the default size of BEV queries is 200×200, the perception ranges are $[-51.2\text{m}, 51.2\text{m}]$ for the X and Y axis and the size of resolution s of BEV's grid is 0.512m. We adopt learnable positional embedding for BEV queries. The BEV encoder contains 6 encoder layers and constantly refines the BEV queries in each layer. The input BEV features B_{t-1} for each encoder layer are the same and require no gradients. For each local query, during the spatial cross-attention module implemented by deformable attention mechanism, it corresponds to $N_{\text{ref}}=4$ target points with different heights in 3D space, and the predefined height anchors are sampled uniformly from -5 meters to 3 meters. For each reference point on 2D view features, we use four sampling points around this reference point for each head. By default, we train our models with 24 epochs, a learning rate of 2×10^{-4}.

For experiments on Waymo, we change a few settings. Due to the camera system of Waymo can not capture the whole scene around the ego car [40], the default spatial shape of BEV queries is 300×220, the perception ranges are $[-35.0\text{m}, 75.0\text{m}]$ for the X-axis and $[-75.0\text{m}, 75.0\text{m}]$ for the Y-axis. The size of resolution s of each gird is 0.5m. The ego car is at (70, 150) of the BEV.

Baselines. To eliminate the effect of task heads and compare other BEV generating methods fairly, we use VPN [30] and Lift-Splat [32] to replace our BEVFormer and keep task heads and other settings the same. We also adapt BEVFormer into a static model called **BEVFormer-S** via adjusting the temporal self-attention into a vanilla self-attention without using history BEV features.

4.3 3D Object Detection Results

We train our model on the detection task with the detection head only for fairly comparing with previous state-of-the-art 3D object detection methods. In Tab. 1 and Tab. 2, we report our main results on nuScenes `test` and `val` splits. Our method outperforms previous best method DETR3D [47] over 9.2 points on `val` set (51.7% NDS *vs.* 42.5% NDS), under fair training strategy and comparable model scales. On the `test` set, our model achieves 56.9% NDS without bells and whistles, 9.0 points

Table 1: **3D detection results on nuScenes `test` set.** $*$ notes that VoVNet-99 (V2-99) [21] was pre-trained on the depth estimation task with extra data [31]. "BEVFormer-S" does not leverage temporal information in the BEV encoder. "L" and "C" indicate LiDAR and Camera, respectively.

Method	Modality	Backbone	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
SSN [55]	L	-	0.569	0.463	-	-	-	-	-
CenterPoint-Voxel [52]	L	-	0.655	0.580	-	-	-	-	-
PointPainting [43]	L&C	-	0.581	0.464	0.388	0.271	0.496	0.247	0.111
FCOS3D [45]	C	R101	0.428	0.358	0.690	0.249	0.452	1.434	**0.124**
PGD [44]	C	R101	0.448	0.386	**0.626**	**0.245**	0.451	1.509	0.127
BEVFormer-S	C	R101	0.462	0.409	0.650	0.261	0.439	0.925	0.147
BEVFormer	C	R101	**0.535**	**0.445**	0.631	0.257	**0.405**	**0.435**	0.143
DD3D [31]	C	V2-99*	0.477	0.418	**0.572**	**0.249**	**0.368**	1.014	**0.124**
DETR3D [47]	C	V2-99*	0.479	0.412	0.641	0.255	0.394	0.845	0.133
BEVFormer-S	C	V2-99*	0.495	0.435	0.589	0.254	0.402	0.842	0.131
BEVFormer	C	V2-99*	**0.569**	**0.481**	0.582	0.256	0.375	**0.378**	0.126

Table 2: **3D detection results on nuScenes `val` set.** "C" indicates Camera.

Method	Modality	Backbone	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
FCOS3D [45]	C	R101	0.415	0.343	0.725	0.263	0.422	1.292	**0.153**
PGD [44]	C	R101	0.428	0.369	0.683	**0.260**	0.439	1.268	0.185
DETR3D [47]	C	R101	0.425	0.346	0.773	0.268	0.383	0.842	0.216
BEVFormer-S	C	R101	0.448	0.375	0.725	0.272	0.391	0.802	0.200
BEVFormer	C	R101	**0.517**	**0.416**	**0.673**	0.274	**0.372**	**0.394**	0.198

higher than DETR3D (47.9% NDS). Our method can even achieve comparable performance to some LiDAR-based baselines such as SSN (56.9% NDS) [55] and PointPainting (58.1% NDS) [43].

Previous camera-based methods [47, 31, 45] were almost unable to estimate the velocity, and our method demonstrates that temporal information plays a crucial role in velocity estimation for multi-camera detection. The mean Average Velocity Error (mAVE) of BEVFormer is 0.378 m/s on the `test` set, outperforming other camera-based methods by a vast margin and approaching the performance of LiDAR-based methods [43].

We also conduct experiments on Waymo, as shown in Tab. 3. Following [34], we evaluate the vehicle category with IoU criterias of 0.7 and 0.5. In addition, We also adopt the nuScenes metrics to evaluate the results since the IoU-based metrics are too challenging for camera-based methods. Due to a few camera-based works reported results on Waymo, we also use the official codes of DETR3D to perform experiments on Waymo for comparison. We can observe that BEVFormer outperforms DETR3D by Average Precision with Heading information (APH) [40] of 6.0% and 2.5% on LEVEL_1 and LEVEL_2 difficulties with IoU criteria of 0.5. On nuScenes metrics, BEVFormer outperforms DETR3D with a margin of 3.2% NDS and 5.2% AP. We also conduct experiments on the front camera to compare BEVFormer with CaDNN [34], a monocular 3D detection method that reported their results on the Waymo dataset. BEVFormer outperforms CaDNN with APH of 13.3% and 11.2% on LEVEL_1 and LEVEL_2 difficulties with IoU criteria of 0.5.

4.4 Multi-tasks Perception Results

We train our model with both detection and segmentation heads to verify the learning ability of our model for multiple tasks, and the results are shown in Tab. 4. While comparing different BEV encoders under same settings, BEVFormer achieves higher performances of all tasks except for road segmentation results is comparable with BEVFormer-S. For example, with joint training, BEVFormer outperforms Lift-Splat* [32] by 11.0 points on detation task (52.0% NDS *v.s.* 41.0% NDS) and IoU of 5.6 points on lane segmentation (23.9% *v.s.* 18.3%). Compared with training tasks individually, multi-task learning saves computational cost and reduces the inference time by sharing more modules,

Table 3: **3D detection results on Waymo `val` set under Waymo evaluation metric and nuScenes evaluation metric.** "L1" and "L2" refer "LEVEL_1" and "LEVEL_2" difficulties of Waymo [40]. *: Only use the front camera and only consider object labels in the front camera's field of view (50.4°). †: We compute the NDS score by setting ATE and AAE to be 1. "L" and "C" indicate LiDAR and Camera, respectively.

Method	Modality	Waymo Metrics				Nuscenes Metrics				
		IoU=0.5		IoU=0.7						
		L1/APH	L2/APH	L1/APH	L2/APH	NDS†↑	AP↑	ATE↓	ASE↓	AOE↓
PointPillars [20]	L	0.866	0.801	0.638	0.557	0.685	0.838	0.143	0.132	0.070
DETR3D [47]	C	0.220	0.216	0.055	0.051	0.394	0.388	0.741	**0.156**	0.108
BEVFormer	C	**0.280**	**0.241**	**0.061**	**0.052**	**0.426**	**0.440**	**0.679**	0.157	**0.101**
CaDNN* [34]	C	0.175	0.165	0.050	0.045	-	-	-	-	-
BEVFormer*	C	0.308	0.277	0.077	0.069	-	-	-	-	-

Table 4: **3D detection and map segmentation results on nuScenes `val` set.** Comparison of training segmentation and detection tasks jointly or not. *: We use VPN [30] and Lift-Splat [32] to replace our BEV encoder for comparison, and the task heads are the same. †: Results from their paper.

Method	Task Head		3D Detection		BEV Segmentation (IoU)			
	Det	Seg	NDS↑	mAP↑	Car	Vehicles	Road	Lane
Lift-Splat† [32]	✗	✓	-	-	32.1	32.1	72.9	20.0
FIERY† [18]	✗	✓	-	-	-	38.2	-	-
VPN* [30]	✓	✗	0.333	0.253	-	-	-	-
VPN*	✗	✓	-	-	31.0	31.8	76.9	19.4
VPN*	✓	✓	0.334	0.257	36.6	37.3	76.0	18.0
Lift-Splat*	✓	✗	0.397	0.348	-	-	-	-
Lift-Splat*	✗	✓	-	-	42.1	41.7	77.7	20.0
Lift-Splat*	✓	✓	0.410	0.344	43.0	42.8	73.9	18.3
BEVFormer-S	✓	✗	0.448	0.375	-	-	-	-
BEVFormer-S	✗	✓	-	-	43.1	43.2	**80.7**	21.3
BEVFormer-S	✓	✓	0.453	0.380	44.3	44.4	77.6	19.8
BEVFormer	✓	✗	0.517	**0.416**	-	-	-	-
BEVFormer	✗	✓	-	-	44.8	44.8	80.1	**25.7**
BEVFormer	✓	✓	**0.520**	0.412	**46.8**	**46.7**	77.5	23.9

including the backbone and the BEV encoder. In this paper, we show that the BEV features generated by our BEV encoder can be well adapted to different tasks, and the model training with multi-task heads performs even better on detection tasks and vehicles segmentation. However, the jointly trained model does not perform as well as individually trained models for road and lane segmentation, which is a common phenomenon called *negative transfer* [11, 13] in multi-task learning.

4.5 Ablation Study

To delve into the effect of different modules, we conduct ablation experiments on nuScenes `val` set with detection head. More ablation studies are in Appendix.

Effectiveness of Spatial Cross-Attention. To verify the effect of spatial cross-attention, we use BEVFormer-S to perform ablation experiments to exclude the interference of temporal information, and the results are shown in Tab. 5. The default spatial cross-attention is based on deformable attention. For comparison, we also construct two other baselines with different attention mechanisms: (1) Using the global attention to replace deformable attention; (2) Making each query only interact with its reference points rather than the surrounding local regions, and it is similar to previous methods [36, 37]. For a broader comparison, we also replace the BEVFormer with the BEV generation methods proposed by VPN [30] and Lift-Splat [32]. We can observe that deformable

Table 5: **The detection results of different methods with various BEV encoders on nuScenes `val` set.** "Memory" is the consumed GPU memory during training. *: We use VPN [30] and Lift-Splat [32] to replace BEV encoder of our model for comparison. †: We train BEVFormer-S using global attention in spatial cross-attention, and the model is trained with fp16 weights. In addition, we only adopt single-scale features from the backbone and set the spatial shape of BEV queries to be 100×100 to save memory. ‡: We degrade the interaction targets of deformable attention from the local region to the reference points only by removing the predicted offsets and weights.

Method	Attention	NDS↑	mAP↑	mATE↓	mAOE↓	#Param.	FLOPs	Memory
VPN* [30]	-	0.334	0.252	0.926	0.598	111.2M	924.5G	~20G
List-Splat* [32]	-	0.397	0.348	0.784	0.537	74.0M	1087.7G	~20G
BEVFormer-S†	Global	0.404	0.325	0.837	0.442	62.1M	1245.1G	~36G
BEVFormer-S‡	Points	0.423	0.351	0.753	0.442	68.1M	1264.3G	~20G
BEVFormer-S	Local	**0.448**	**0.375**	**0.725**	**0.391**	68.7M	1303.5G	~20G

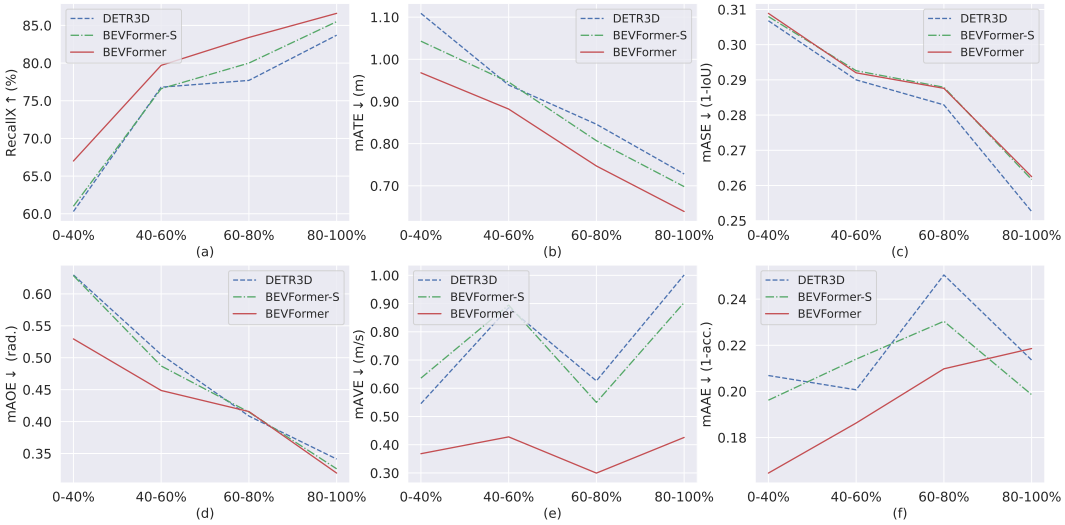

Figure 3: **The detection results of subsets with different visibilities.** We divide the nuScenes `val` set into four subsets based on the visibility that {0-40%, 40-60%, 60-80%, 80-100%} of objects can be visible. (a): Enhanced by the temporal information, BEVFormer has a higher recall on all subsets, especially on the subset with the lowest visibility (0-40%). (b), (d) and (e): Temporal information benefits translation, orientation, and velocity accuracy. (c) and (f): The scale and attribute error gaps among different methods are minimal. Temporal information does not work to benefit an object's scale prediction.

attention significantly outperforms other attention mechanisms under a comparable model scale. Global attention consumes too much GPU memory, and point interaction has a limited receptive field. Sparse attention achieves better performance because it interacts with a priori determined regions of interest, balancing receptive field and GPU consumption.

Effectiveness of Temporal Self-Attention. From Tab. 1 and Tab. 4, we can observe that BEVFormer outperforms BEVFormer-S with remarkable improvements under the same setting, especially on challenging detection tasks. The effect of temporal information is mainly in the following aspects: (1) The introduction of temporal information greatly benefits the accuracy of the velocity estimation; (2) The predicted locations and orientations of the objects are more accurate with temporal information; (3) We obtain higher recall on heavily occluded objects since the temporal information contains past objects clues, as showed in Fig. 3. To evaluate the performance of BEVFormer on objects with different occlusion levels, we divide the validation set of nuScenes into four subsets according to the official visibility label provided by nuScenes. In each subset, we also compute the average recall of all categories with a center distance threshold of 2 meters during matching. The maximum number of

Table 6: **Latency and performance of different model configurations on nuScenes `val` set.** The latency is measured on a V100 GPU, and the backbone is R101-DCN. The input image shape is 900×1600. "MS" notes multi-scale view features.

Method	Scale of BEVFormer			Latency (ms)			FPS	NDS↑	mAP↑
	MS	BEV	#Layer	Backbone	BEVFormer	Head			
BEVFormer	✓	200×200	6	391	130	19	1.7	**0.517**	**0.416**
A	✗	200×200	6	387	87	19	1.9	0.511	0.406
B	✓	100×100	6	391	53	18	2.0	0.504	0.402
C	✓	200×200	1	391	25	19	2.1	0.501	0.396
D	✗	100×100	1	387	**7**	18	**2.3**	0.478	0.374

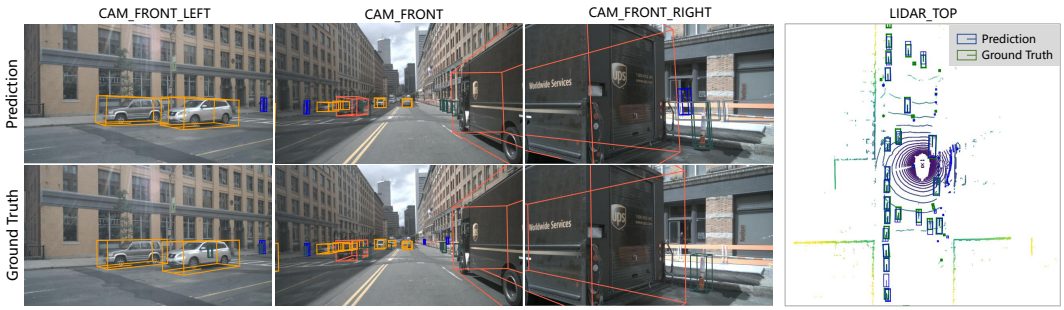

Figure 4: **Visualization results of BEVFormer on nuScenes `val` set.** We show the 3D bboxes predictions in multi-camera images and the bird's-eye-view.

predicted boxes is 300 for all methods to compare recall fairly. On the subset that only 0-40% of objects can be visible, the average recall of BEVFormer outperforms BEVFormer-S and DETR3D with a margin of more than 6.0%.

Model Scale and Latency. We compare the performance and latency of different configurations in Tab. 6. We ablate the scales of BEVFormer in three aspects, including whether to use multi-scale view features, the shape of BEV queries, and the number of layers, to verify the trade-off between performance and inference latency. We can observe that configuration C using one encoder layer in BEVFormer achieves 50.1 % NDS and reduces the latency of BEVFormer from the original 130ms to 25ms. Configuration D, with single-scale view features, smaller BEV size, and only 1 encoder layer, consumes only 7ms during inference, although it loses 3.9 points compared to the default configuration. However, due to the multi-view image inputs, the bottleneck that limits the efficiency lies in the backbone, and efficient backbones for autonomous driving deserve in-depth study. Overall, our architecture can adapt to various model scales and be flexible to trade off performance and efficiency.

4.6 Visualization Results

We show the detection results of a complex scene in Fig. 4. BEVFormer produces impressive results except for a few mistakes in small and remote objects. More qualitative results are provided in Appendix.

5 Discussion and Conclusion

In this work, we have proposed BEVFormer to generate the bird's-eye-view features from multi-camera inputs. BEVFormer can efficiently aggregate spatial and temporal information and generate powerful BEV features that simultaneously support 3D detection and map segmentation tasks.

Limitations. At present, the camera-based methods still have a particular gap with the LiDAR-based methods in effect and efficiency. Accurate inference of 3D location from 2D information remains a long-stand challenge for camera-based methods.

Broader impacts. BEVFormer demonstrates that using spatiotemporal information from the multi-camera input can significantly improve the performance of visual perception models. The advantages demonstrated by BEVFormer, such as more accurate velocity estimation and higher recall on low-visible objects, are essential for constructing a better and safer autonomous driving system and beyond. We believe BEVFormer is just a baseline of the following more powerful visual perception methods, and vision-based perception systems still have tremendous potential to be explored.

References

[1] Brazil, G., Liu, X.: M3d-rpn: Monocular 3d region proposal network for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9287–9296 (2019)

[2] Brazil, G., Pons-Moll, G., Liu, X., Schiele, B.: Kinematic 3d object detection in monocular video. In: European Conference on Computer Vision. pp. 135–152. Springer (2020)

[3] Bruls, T., Porav, H., Kunze, L., Newman, P.: The right (angled) perspective: Improving the understanding of road scenes using boosted inverse perspective mapping. In: 2019 IEEE Intelligent Vehicles Symposium (IV). pp. 302–309. IEEE (2019)

[4] Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11621–11631 (2020)

[5] Can, Y.B., Liniger, A., Paudel, D.P., Van Gool, L.: Structured bird's-eye-view traffic scene understanding from onboard images. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15661–15670 (2021)

[6] Can, Y.B., Liniger, A., Unal, O., Paudel, D., Van Gool, L.: Understanding bird's-eye view semantic hd-maps using an onboard monocular camera. arXiv preprint arXiv:2012.03040 (2020)

[7] Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)

[8] Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3d object detection network for autonomous driving. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 1907–1915 (2017)

[9] Chitta, K., Prakash, A., Geiger, A.: Neat: Neural attention fields for end-to-end autonomous driving. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15793–15803 (2021)

[10] Cho, K., Van Merriënboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. arXiv preprint arXiv:1409.1259 (2014)

[11] Crawshaw, M.: Multi-task learning with deep neural networks: A survey. arXiv preprint arXiv:2009.09796 (2020)

[12] Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision. pp. 764–773 (2017)

[13] Fifty, C., Amid, E., Zhao, Z., Yu, T., Anil, R., Finn, C.: Efficiently identifying task groupings for multi-task learning. Advances in Neural Information Processing Systems **34** (2021)

[14] Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. In: International Conference on Machine Learning. pp. 1243–1252. PMLR (2017)

[15] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)

[16] Hendy, N., Sloan, C., Tian, F., Duan, P., Charchut, N., Xie, Y., Wang, C., Philbin, J.: Fishing net: Future inference of semantic heatmaps in grids. arXiv preprint arXiv:2006.09917 (2020)

[17] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[18] Hu, A., Murez, Z., Mohan, N., Dudas, S., Hawke, J., Badrinarayanan, V., Cipolla, R., Kendall, A.: Fiery: Future instance prediction in bird's-eye view from surround monocular cameras. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15273–15282 (2021)

[19] Kang, K., Ouyang, W., Li, H., Wang, X.: Object detection from video tubelets with convolutional neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 817–825 (2016)

[20] Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O.: Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12697–12705 (2019)

[21] Lee, Y., Hwang, J.w., Lee, S., Bae, Y., Park, J.: An energy and gpu-computation efficient backbone network for real-time object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 0–0 (2019)

[22] Li, Z., Wang, W., Xie, E., Yu, Z., Anandkumar, A., Alvarez, J.M., Lu, T., Luo, P.: Panoptic segformer: Delving deeper into panoptic segmentation with transformers. arXiv preprint arXiv:2109.03814 (2021)

[23] Lin, T.Y., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 936–944 (2017)

[24] Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv: Learning (2017)

[25] Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: ICLR (2019)

[26] Luo, W., Yang, B., Urtasun, R.: Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 3569–3577 (2018)

[27] Ma, X., Ouyang, W., Simonelli, A., Ricci, E.: 3d object detection from images for autonomous driving: A survey. arXiv preprint arXiv:2202.02980 (2022)

[28] Mousavian, A., Anguelov, D., Flynn, J., Kosecka, J.: 3d bounding box estimation using deep learning and geometry. In: Proceedings of the IEEE conference on Computer Vision and Pattern Recognition. pp. 7074–7082 (2017)

[29] Ng, M.H., Radia, K., Chen, J., Wang, D., Gog, I., Gonzalez, J.E.: Bev-seg: Bird's eye view semantic segmentation using geometry and semantic point cloud. arXiv preprint arXiv:2006.11436 (2020)

[30] Pan, B., Sun, J., Leung, H.Y.T., Andonian, A., Zhou, B.: Cross-view semantic segmentation for sensing surroundings. IEEE Robotics and Automation Letters **5**(3), 4867–4873 (2020)

[31] Park, D., Ambrus, R., Guizilini, V., Li, J., Gaidon, A.: Is pseudo-lidar needed for monocular 3d object detection? In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3142–3152 (2021)

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

[33] Qi, C.R., Zhou, Y., Najibi, M., Sun, P., Vo, K., Deng, B., Anguelov, D.: Offboard 3d object detection from point cloud sequences. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 6134–6144 (2021)

[34] Reading, C., Harakeh, A., Chae, J., Waslander, S.L.: Categorical depth distribution network for monocular 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8555–8564 (2021)

[35] Reiher, L., Lampe, B., Eckstein, L.: A sim2real deep learning approach for the transformation of images from multiple vehicle-mounted cameras to a semantically segmented image in bird's eye view. In: 2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC). pp. 1–7. IEEE (2020)

[36] Roddick, T., Kendall, A., Cipolla, R.: Orthographic feature transform for monocular 3d object detection. In: BMVC (2019)

[37] Rukhovich, D., Vorontsova, A., Konushin, A.: Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. pp. 2397–2406 (2022)

[38] Saha, A., Maldonado, O.M., Russell, C., Bowden, R.: Translating images into maps. arXiv preprint arXiv:2110.00966 (2021)

[39] Simonelli, A., Bulo, S.R., Porzi, L., Lopez-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)

[40] Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., et al.: Scalability in perception for autonomous driving: Waymo open dataset. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 2446–2454 (2020)

[41] Tian, Z., Shen, C., Chen, H., He, T.: Fcos: Fully convolutional one-stage object detection. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9627–9636 (2019)

[42] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[43] Vora, S., Lang, A.H., Helou, B., Beijbom, O.: Pointpainting: Sequential fusion for 3d object detection. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 4604–4612 (2020)

[44] Wang, T., Xinge, Z., Pang, J., Lin, D.: Probabilistic and geometric depth: Detecting objects in perspective. In: Conference on Robot Learning. pp. 1475–1485. PMLR (2022)

[45] Wang, T., Zhu, X., Pang, J., Lin, D.: Fcos3d: Fully convolutional one-stage monocular 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 913–922 (2021)

[46] Wang, Y., Chao, W.L., Garg, D., Hariharan, B., Campbell, M., Weinberger, K.Q.: Pseudo-lidar from visual depth estimation: Bridging the gap in 3d object detection for autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8445–8453 (2019)

[47] Wang, Y., Guizilini, V.C., Zhang, T., Wang, Y., Zhao, H., Solomon, J.: Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In: Conference on Robot Learning. pp. 180–191. PMLR (2022)

[48] Xie, E., Yu, Z., Zhou, D., Philion, J., Anandkumar, A., Fidler, S., Luo, P., Alvarez, J.M.: Mˆ 2bev: Multi-camera joint 3d detection and segmentation with unified birds-eye view representation. arXiv preprint arXiv:2204.05088 (2022)

[49] Xu, B., Chen, Z.: Multi-level fusion based 3d object detection from monocular images. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2345–2353 (2018)

[50] Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018)

[51] Yang, W., Li, Q., Liu, W., Yu, Y., Ma, Y., He, S., Pan, J.: 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. pp. 15536–15545 (2021)

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

[53] Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019)

[54] Zhou, Y., Tuzel, O.: Voxelnet: End-to-end learning for point cloud based 3d object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 4490–4499 (2018)

[55] Zhu, X., Ma, Y., Wang, T., Xu, Y., Shi, J., Lin, D.: Ssn: Shape signature networks for multi-class object detection from point clouds. In: European Conference on Computer Vision. pp. 581–597. Springer (2020)

[56] Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. In: International Conference on Learning Representations (2020)

[57] Zhu, X., Xiong, Y., Dai, J., Yuan, L., Wei, Y.: Deep feature flow for video recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2349–2358 (2017)

Appendix

A Implementation Details

In this section, we provide more implementation details of the proposed method and experiments.

A.1 Traning Strategy

Following previous methods [47, 56], we train all models with 24 epochs, a batch size of 1 (containing 6 view images) per GPU, a learning rate of 2×10^{-4}, learning rate multiplier of the backbone is 0.1, and we decay the learning rate with a cosine annealing [24]. We employ AdamW [25] with a weight decay of 1×10^{-2} to optimize our models.

A.2 VPN and Lift-Splat

We use VPN [30] and Lift-Splat [32] as two baselines in this work. The backbone and the task heads are the same as the BEVFomer for fair comparisons.

VPN. We employ the official codes[1] in this work. Limited by the huge amount of parameters of MLP, it is difficult for VPN to generate high-resolution BEV (*e.g.*, 200$\times$ 200). To compare with VPN, in this work, we transform the single-scale view features into BEV with a low resolution of 50×50 via two view translation layers.

Lift-Splat. We enhance the camera encoder of Lift-Splat[2] with two additional convolutional layers for a fair comparison with our BEVFormer under a comparable parameter number. Other settings remain unchanged.

A.3 Task Heads

Detection Head. We predict 10 parameters for each 3D bounding box, including the 3 parameters (l, w, h) for the scale of each box, 3 parameters (x_o, y_o, z_o) for the center location, 2 parameters (cos(θ), sin(θ)) for object's yaw θ, 2 parameters (v_x, v_y) for the velocity. Only L_1 loss and L_1 cost are used during training phase. Following [47], we use 900 object queries and keep 300 predicted boxes with highest confidence scores during inference.

Sementation Head. As shown in Fig. 5, for each class of the semantic map, we follow the mask decoder in [22] to use one learnable query to represent this class, and generate the final segmentation masks based on the attention maps from the vanilla multi-head attention.

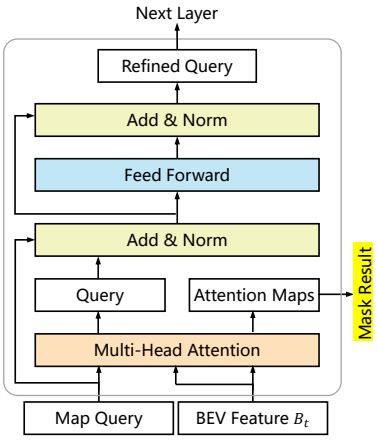

Figure 5: **Segmentation head (mask decoder) of BEVFormer.**

A.4 Spatial Cross-Attention

Global Attention. Besides deformable attention [56], our spatial cross-attention can also be implemented by global attention (*i.e.*, vanilla multi-head attention) [42]. The most straightforward way to employ global attention is making each BEV query interact with all multi-camera features, and this conceptual implementation does not require camera calibration. However, the computational cost of this straightforward way is unaffordable. Therefore, we still utilize the camera intrinsic and extrinsic to decide the hit views that one BEV query deserves to interact. This strategy makes that one BEV query usually interacts with only one or two views rather than all views, making it possible to use global attention in the spatial cross-attention. Notably, compared to other attention mechanisms that rely on precise camera intrinsic and extrinsic, global attention is more robust to camera calibration.

[1] https://github.com/pbw-Berwin/View-Parsing-Network
[2] https://github.com/nv-tlabs/lift-splat-shoot

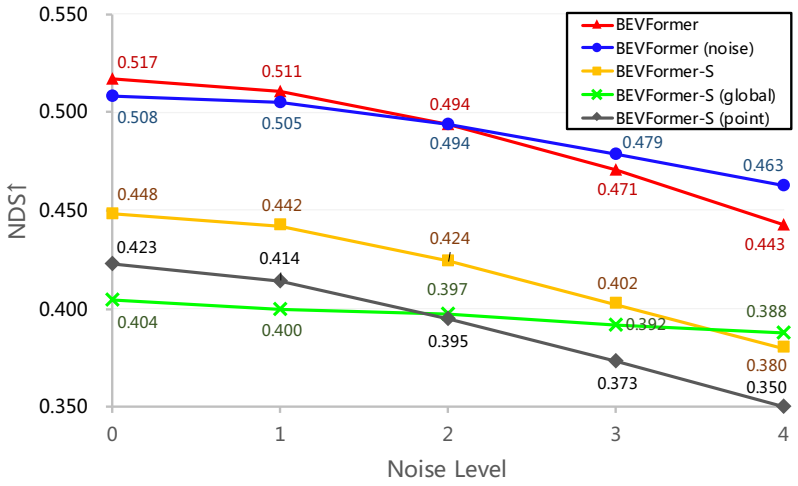

Figure 6: **NDS of methods on nuScenes val set subjected to different levels of camera extrinsics noises.** For i-th level noises, the rotation noises are sampled from a normal distribution with mean equals 0 and variance equals i (rotation noise are in degrees, and the noise of each axis is independent), and the translation noises are sampled from a normal distribution with mean equals 0 and variance equals $5i$ (translation noises are in centimeters, and the noise of each direction is independent). "BEVFormer" is our default version. "BEVFormer (noise)" is trained with noisy extrinsics (noise level=1). "BEVFormer-S" is our static version of BEVFormer with the spatial cross-attention implemented by deformable attention [56]. "BEVFormer-S (global)" is BEVFormer-S with the spatial cross-attention implemented by global attention (*i.e.*, vanilla multi-head attention) [42]. "BEVFormer-S (point)" is BEVFormer-S with point spatial cross-attention where we degrade the interaction targets of deformable attention from the local region to the reference points only by removing the predicted offsets and weights.

B Robustness on Camera Extrinsics

BEVFormer relies on camera intrinsics and extrinsics to obtain the reference points on 2D views. During the deployment phase of autonomous driving systems, extrinsics may be biased due to various reasons such as calibration errors, camera offsets, etc. As shown in Fig. 6, we show the results of models under different camera extrinsics noise levels. Compared to BEVFormer-S (point), BEVFormer-S utilizes the spatial cross-attention based on deformable attention [56] and samples features around the reference points rather than only interacting with the reference points. With deformable attention, the robustness of BEVFormer-S is stronger than BEVFormer-S (point). For example, with the noise level being 4, the NDS of BEVFormer-S drops 15.2% (calculated by $1-\frac{0.380}{0.448}$), while the NDS of BEVFormer-S (point) drops 17.3%. Compared to BEVFormer-S, BEVFormer only drops 14.3% NDS, which shows that temporal information can also improve robustness on camera extrinsics. Following [32], we show that when training BEVFormer with noisy extrinsics, BEVFormer (noise) has stronger robustness (only drops 8.9% NDS). With the spatial cross-attention based on global attention, BEVFormer (global) has a strong anti-interference ability (4.0% NDS drop) even under level 4 of the camera extrinsics noise. The reason is that we do not utilize camera extrinsics to select the RoIs for BEV queries.

Notably, under the harshest noises, we see that BEVFormer-S (global) even outperforms BEVFormer-S (38.8% NDS *vs.* 38.0% NDS).

C Ablation Studies

Effect of the frame number during training. Tab. 7 shows the effect of the frame number during training. We see that the NDS on nuScenes val set keeps rising with the growth of the frame number and begins to level off the frame number ≥ 4. Therefore, we set the frame number during training to 4 by default in experiments.

Table 7: **NDS of models on nuScenes val set using different frame numbers during training.** "#Frame" denotes the frame number during training.

#Frame	NDS↑	mAP↑	mAVE↓
1	0.448	0.375	0.802
2	0.490	0.388	0.467
3	0.510	0.410	0.423
4	**0.517**	**0.416**	0.394
5	**0.517**	0.412	**0.387**

Table 8: **Ablation Experiments on nuScenes val set.** "A." indicates aligning history BEV features with ego-motion. "R." indicates randomly sampling 4 frames from 5 continuous frames. "B." indicates using both BEV queries and history BEV features to predict offsets and weights.

#	A.	R.	B.	NDS↑	mAP↑
1	✗	✓	✓	0.510	0.410
2	✓	✗	✓	0.513	0.410
3	✓	✓	✗	0.513	0.404
4	✓	✓	✓	**0.517**	**0.416**

Effect of some designs. Tab. 8 shows the results of several ablation studies. Comparing #1 and #4, we see that aligning history BEV features with ego-motion is important to represent the same geometry scene as current BEV queries (51.0% NDS *vs.* 51.7% NDS). Comparing #2 and #4, randomly sampling 4 frames from 5 frames is a effective data augment strategy to improve performance (51.3% NDS *vs.* 51.7% NDS). Compared to only using the BEV query to predict offsets and weights during the temporal self-attention module (see #3), using both BEV queries and history BEV features (see #4) contain more clues about the past BEV features and benefits location prediction (51.3% NDS *vs.* 51.7% NDS).

D Visualization

As shown in Fig. 7, we compare BEVFormer with BEVFormer-S. With temporal information, BEVFormer successfully detected two buses occluded by boards. We also show both object detection and map segmentation results in Fig. 8, where we see that the detection results and segmentation results are highly consistent. We provide more map segmentation results in Fig. 9, where we see that with the strong BEV features generated by BEVFormer, the semantic maps can be well predicted via a simple mask decoder.

Figure 7: **Comparision of BEVFormer and BEVFormer-S on nuScenes val set.** We can observe that BEVFormer can detect highly occluded objects, and these objects are missed in the prediction results of BEVFormer-S (in red circle).

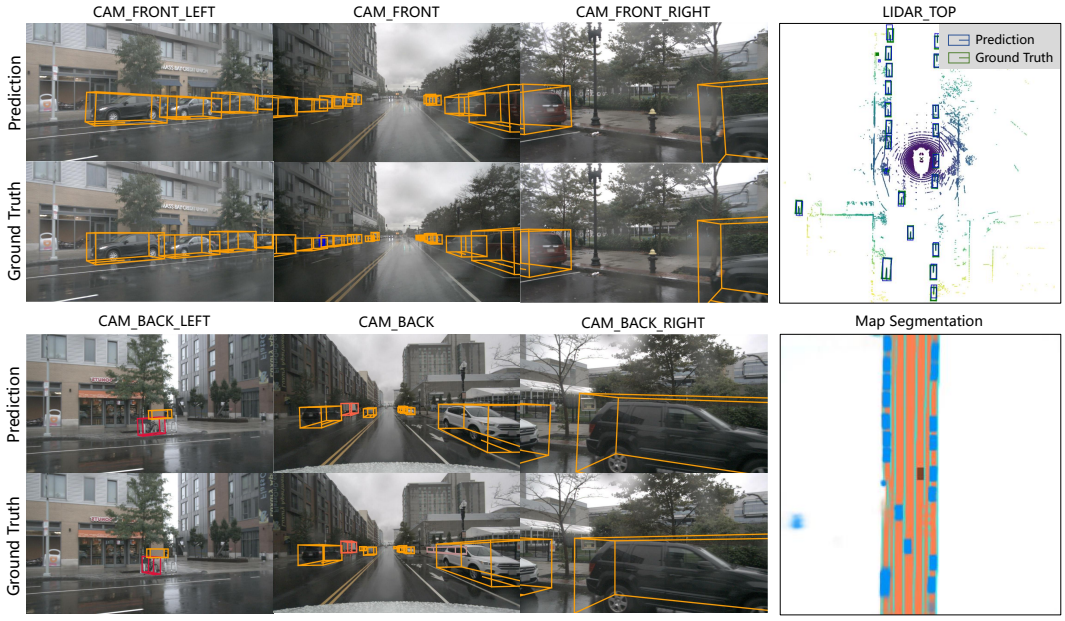

Figure 8: **Visualization results of both object detection and map segmentation tasks.** We show vehicle, road, and lane segmentation in blue, orange, and green, respectively.

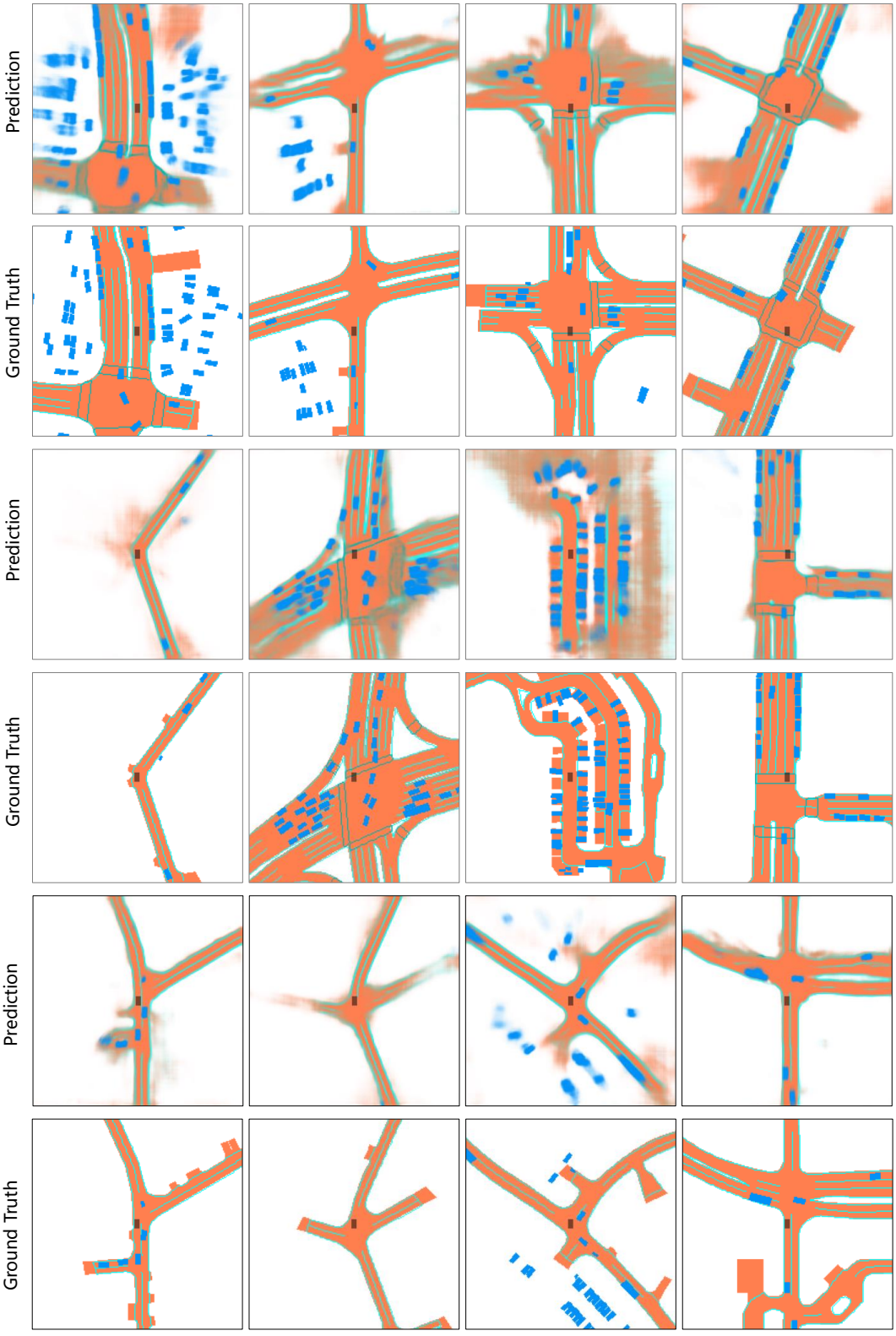

Figure 9: **Visualization results of the map segmentation task.** We show vehicle, road, ped crossing and lane segmentation in blue, orange, cyan, and green, respectively.