Fast-BEV: Towards Real-time On-vehicle Bird's-Eye View Perception

Bin Huang[1*] **Yangguang Li**[1*] **Enze Xie**[2*] **Feng Liang**[3*] **Luya Wang**[4]

Mingzhu Shen[1] **Fenggang Liu**[1] **Tianqi Wang**[2] **Ping Luo**[2] **Jing Shao**[1]

[1] SenseTime, [2] The University of Hong Kong
[3] The University of Texas at Austin
[4] Beijing University of Posts and Telecommunications
{huangbin1,liufenggang,shaojing}@senseauto.com
{liyangguang,shenmingzhu}@sensetime.com
{xieenze,wangtq}@connect.hku.hk, wangluya@bupt.edu.cn
jeffliang@utexas.edu, pluo@cs.hku.hk

Abstract

Recently, the pure camera-based Bird's-Eye-View (BEV) perception removes expensive Lidar sensors, making it a feasible solution for economical autonomous driving. However, most existing BEV solutions either suffer from modest performance or require considerable resources to execute on-vehicle inference. This paper proposes a simple yet effective framework, termed Fast-BEV, which is capable of performing real-time BEV perception on the on-vehicle chips. Towards this goal, we first empirically find that the BEV representation can be sufficiently powerful without expensive view transformation or depth representation. Starting from M^{2}BEV [1] baseline, we further introduce (1) a strong data augmentation strategy for both image and BEV space to avoid over-fitting (2) a multi-frame feature fusion mechanism to leverage the temporal information (3) an optimized deployment-friendly view transformation to speed up the inference. Through experiments, we show Fast-BEV model family achieves considerable accuracy and efficiency on edge. In particular, our M1 model (R18@256×704) can run over 50FPS on the Tesla T4 platform, with 47.0% NDS on the nuScenes validation set. Our largest model (R101@900x1600) establishes a new state-of-the-art 53.5% NDS on the nuScenes validation set. The code is released at: https://github.com/Sense-GVT/Fast-BEV.

1 Introduction

An accurate 3D perception system is essential for autonomous driving. Classic methods [2, 3, 4] rely on the accurate 3D information provided by Lidar point clouds. However, Lidar sensors usually cost thousands of dollars [5], hindering their applications on economical vehicles. Pure camera-based Bird's-Eye-View (BEV) methods [1, 6, 7, 8, 9] have recently shown great potential for their impressive 3D perception capability and economical cost.

To perform 3D perception from 2D image features, state-of-the-art BEV methods on nuScenes [10] either uses implicit/explicit depth based projection [11, 7, 9] or transformer based projection [12, 6].

[*] Equal contribution.

Machine Learning for Autonomous Driving Workshop at the 36th Conference on Neural Information Processing Systems (NeurIPS 2022), New Orleans, USA.

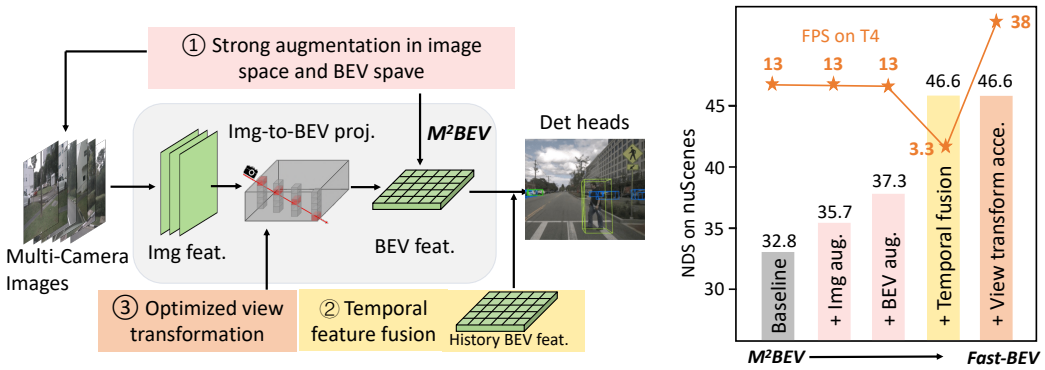

Figure 1: Built upon M^{2}BEV, the proposed Fast-BEV first incorporates ① strong augmentations to avoid over-fitting and ② a multi-frame feature fusion mechanism to leverage the temporal information, leading to state-of-the-art performance. We further propose to optimize the ③ view transformation to be more deployment friendly for on-vehicle platforms.

However, they are difficult to deploy on on-vehicle chips: (1). Method with depth distribution prediction usually requires multi-thread CUDA kernels to speed up inference, which is inconvenient to operate on chips that are resource-constrained or not supported by inference libraries. (2). Attention mechanism within transformer needs dedicated chips to support. Moreover, they are time-consuming in inference, which prevents them from actual deployment. In this paper, we aim to design a BEV perception framework with leading performance, friendly deployment, and high inference speed for *on-vehicle chips*.

Based on these observations, we choose to follow the principle of M^{2}BEV [1] which assumes a uniform depth distribution along the camera ray during image-to-BEV view transformation. We propose Fast-BEV, a stronger and faster fully convolutional BEV perception framework without expensive view transformer [13, 6] or depth representation [7, 8, 9]. We first verify the effectiveness of two training techniques by implementing them in the M^{2}BEV framework, namely, strong data augmentation [7, 9] and temporal fusion [8, 6]. More specifically, additional data augmentation is applied on both image and BEV space to avoid over-fitting, which yields better performance. We also extends the M^{2}BEV from spatial-only space to spatial-temporal space via introducing the temporal feature fusion module, enabling current key-frame leverage the information from history frames. After integrating these two training techniques, our Fast-BEV can achieve comparable performance with state-of-the-art methods [8, 9].

Fast-BEV inherits the image-to-BEV transformation from M^{2}BEV. Although it is already fast, we empirically show that it can be further optimized for on-vehicle platforms. We find that the projection from image space to voxel space dominates the latency. Specifically, M^{2}BEV first computes the 2D-to-3D projection index for each camera view. After projection, it needs to aggregate the voxel features from all views. We identify that each step can be significantly accelerated. First for the projection index, given the fact that camera positions and their intrinsic/extrinsic parameters are fixed when the perception system is built, we don't need to compute the *same* index for each iteration. Thus, we propose to pre-compute the fixed projection indexes and store them as a static look-up-table, which is super efficient during inference. Second for the voxel aggregation, we observe the voxel features are highly sparse, *e.g.*, only about 17% positions are non-zero for 6-view dataset. This is because each voxel feature only has the information of one camera. We propose to generate a dense voxel feature to avoid the expensive voxel aggregation. Specifically, we let image features from all camera views project to the *same* voxel feature (see Section. 3.5). Combining these two acceleration designs improves the speed of the original M^{2}BEV by an order of magnitude.

The proposed Fast-BEV model family shows great performance and can be easily deployed on on-vehicle platforms. On the nuScenes [10] dataset, our M1 model (R18@256×704) can run over 50FPS on the Tesla T4 platform, with considerable 46.9% NDS performance. Our largest model (R101@900x1600) establishes a new state-of-the-art 53.5% NDS on the nuScenes validation set. In conclusion, our contributions are summarized as follows:

- We verify the effectiveness of two techniques: strong data augmentation and multi-frame temporal fusion on M^{2}BEV, enabling Fast-BEV achieve the SOTA performance.

- We propose two acceleration designs: pre-computing the projection index and projecting to the same voxel feature, making Fast-BEV to be easily deployed on the on-vehicle chips with fast inference speed.

- To the best of our knowledge, the proposed Fast-BEV is the first deployment-oriented work targeted on the challenging real-time on-vehicle BEV perception. We hope our work can shed light on the industrial-level, real-time, on-vehicle BEV perception.

2 Related Work

Monocular 3D Perception. Compared to lidar, cheaper cameras provide richer semantic information and are immune to weather conditions. Camera-based 3D object detection, given only image input, aims to estimate the object category, location, dimension, and orientation in the 3D space. A practical approach in monocular 3D detection is to predict 3D bounding boxes based on 2D image features. M3D-RPN [14] proposes a 3D region proposal network and depth-aware convolutional layers to improve 3D scene understanding. Following FCOS [15], FCOS3D [16] directly predicts 3D bounding boxes for each object by converting 3D targets into the image domain. Further, PGD [17] uses the relations across the objects and a probabilistic representation to capture depth uncertainty to facilitate depth estimation for 3D object detection. DD3D [18] benefits from depth pre-training and significantly improve end-to-end 3D detection. Besides object detection, another main perception task in autonomous driving is semantic segmentation in BEV for the targets to vectorially restore the surrounding environment. Some recent works [12, 19, 20, 21] take 2D features into a 3D BEV representation based on a single view and perform BEV segmentation.

Surrounding 3D Perception. The monocular 3D perception benchmark [22] with only single-view images is insufficient for complicated tasks. Recently, some large-scale benchmarks [10, 23] have been proposed with much data and surrounding views, further promoting the field of 3D perception. The essential to 3D object detection is that all predictions correspond to the ground truth in 3D space. One typical approach is to predict the depth of the camera-based input and convert the depth s into a pseudo-LiDAR to mimic the lidar signal [24, 25, 26]. Recently there has been another surge of interest in transforming image features to stereo representation like BEV(Bird's-Eye-View) and 3D voxels [27]. [24] transforms Pseudo-LiDAR obtained by visual depth estimation into BEV features. CaDDN [28] uses a predicted categorical depth distribution for each pixel to project contextual features to BEV. LSS [11] explicitly predicts depth distribution with a proposed view transform and projects image features onto BEV. OFT [29] and ImVoxelNet [27] generate the voxel representation of the scene by projecting the pre-defined voxels onto image features. With a similar 4-stage framework [20, 21, 30], BEVDet [7] and M^2BEV [1] effectively extend CaDDN [28] and OFT [29] to multi-camera 3D object detection. Another class of methods is based on explicit pre-defined grid-shaped BEV queries. BEVFormerr [6] performs 2D-to-3D transformation based on spatial cross-attention. Following DETR [31], DETR3D [13] and Graph-DETR3D [32] generates 3D reference points from BEV queries to sample the correlative 2D features. PETR [33] further introduces the 3D coordinate generation to perceive the 3D position-aware features and avoid generating 3D reference points.

Multi-frame Fusion / Temporal Alignment. Lidar-based detectors can easily adopt multi-frame fusion [34, 3, 35] to improve the velocity estimation accuracy and achieve better 3D object detection. In contrast, the existing pure vision paradigms perform relatively poorly in predicting time-relevant targets due to the inability to access time cues. As an intermediate feature that simultaneously combines visual information from multiple multi-cameras at one time, BEV is suitable for temporal alignment. [36] proposes the Dynamics Module to use past spatial BEV features to learn a spatial-temporal BEV representation. BEVFormer [6] proposes a temporal self-attention to fuse the history BEV information recurrently, similar to the hidden state of RNN models. BEVDet4D [7] extends the BEVDet [7] by aligning the multi-frame features and exploiting spatial correlations in ego-motion. PETRv2 [37] extends the temporal version from PETR [33] to directly achieve temporal alignment in 3D space based on the perspective of 3D position embeddings.

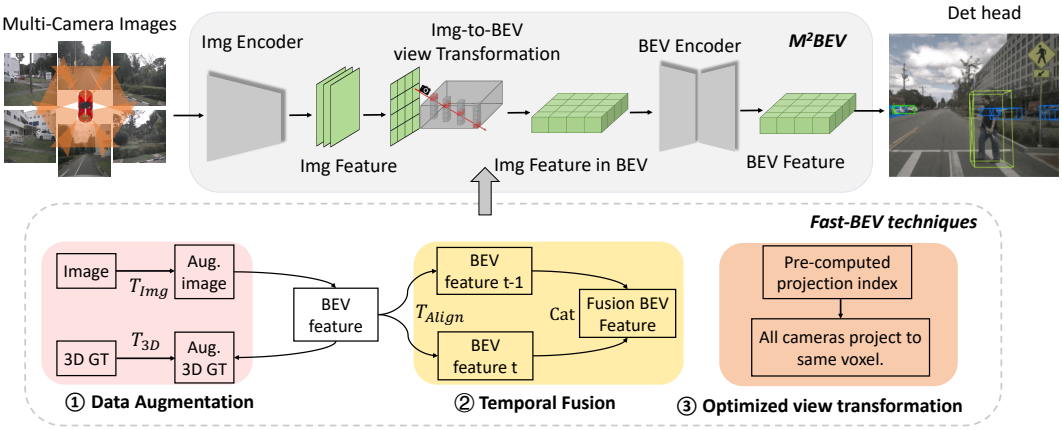

Figure 2: Overview of Fast-BEV. Same as M^{2}BEV, the multi-camera images are first fed into a image encode to extract image features. Then, image features are transformed into BEV space via view transform. A BEV encoder and task-specific heads are followed to perform perception tasks. The proposed Fast-BEV further applies ① data augmentation on image and BEV domain, ② temporal multi-frame fusion on M^{2}BEV. We further propose to pre-compute the image-to-voxel index and let all cameras project to same dense voxel to make ③ view transformation model deployment friendly.

3 Methods

3.1 Revisiting M^{2}BEV

M^{2}BEV is one of the first works to solve the **M**ulti-camera **M**ulti-task perception with unified **BEV** representation. It is also more applicable for on-vehicle platforms since it doesn't have expensive view transformer or depth representation. As denoted in the top part of Figure 2, the input of M^{2}BEV is multi-camera RGB images, and the output is the predicted 3D bounding box (including velocity) and map segmentation results. M^{2}BEV has four key modules: (1) A 2D image encoder that extracts image features from multi-camera images (2) An image-to-BEV (2D→3D) view transformation module that maps the 2D image feature into 3D BEV space (3) A 3D BEV encoder that processes the 3D features and (4) Task-specific heads that perform perception tasks, *e.g.*, 3D detection.

3.2 Overall Architecture of Fast-BEV

Although M^{2}BEV can achieve competitive results, we find its performance and efficiency can be further improved. As denoted in the bottom part of Figure 2, we integrates three techniques into M^{2}BEV, leading to our stronger and faster Fast-BEV.

① Data augmentation. We empirically observe that severe over-fitting problem happened during the later training epochs in M^{2}BEV. This is because no data augmentation is used in original M^{2}BEV. Motivated by recent work [7, 9], we add strong 3D augmentation on both image and BEV space, such as random flip, rotation *etc*. More details are in Section 3.3

② Temporal fusion. In real autonomous driving scenarios, the input is temporally continuous and has tremendous complementary information across time. For instance, one pedestrian partially occluded at current frame might be fully visible in the past several frames. Thus, we extend M^{2}BEV from spatial-only space to spatial-temporal space via introducing the temporal feature fusion module, similar with [8, 6]. More specifically, we use current frame BEV features and stored history frame features as input and train Fast-BEV in an end-to-end manner. More details are in Section 3.4

③ Optimized view transformation. We find that the projection from image space to voxel space dominates the latency. We propose to optimize the projection from two perspectives: (1) we pre-compute the fixed projection indexes and store them as a static look-up-table, which is super efficient during inference. (2) We let all the cameras project to the same voxel to avoid expensive voxel aggregation. Our proposal is not like the improved view transform schemes [9, 7, 38] based on Lift-Splat-Shoot, which requires the development of cumbersome and difficult DSP/GPU parallel

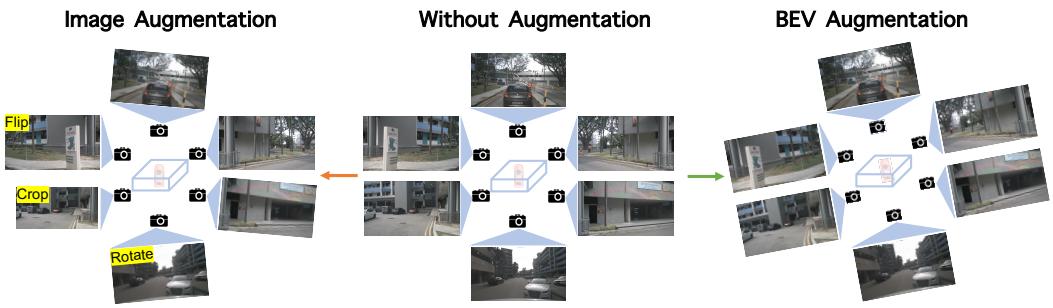

Figure 3: Examples of the data augmentation used in Fast-BEV. The middle figure shows the original M^{2}BEV, which does not use data augmentation. The left figure shows the image augmentation and some augmentation types such as random flip, crop and rotate. The right figure shows one type of BEV augmentation, random rotation.

computing, it is fast enough to use only CPU computing, which is very convenient for deployment. More details are in Section 3.5

We would like to clarify that ① and ② are inspired by the concurrent leading works [8, 6] and we do not intend to regard these two parts as novel designs. These improvements make our proposed pipeline, Fast-BEV, become a SOTA method while keeping its simplicity for on-vehicle platforms.

3.3 Data Augmentation

We add data augmentations in both image space and BEV space, mainly following BEVDet [7].

Image Augmentation. The data augmentation in 3D object detection is more challenging than that of 2D detection since the images in 3D scenarios have direct relationship with 3D camera coordinates. Thus, if we apply data augmentation on images, we need also change the camera intrinsic matrix [7]. For the augmentation operations, we basically follow the common operations, *e.g.*, flipping, cropping and rotation. In the left part of Fig. 3, we show some examples of image augmentations.

BEV Augmentation.

Similar to image augmentation, similar operations can be applied to the BEV space, such as flipping, scaling and rotation. Note that the augmentation transformation should be applied on both the BEV feature map and the 3D ground-truth box to keep consistency. The BEV augmentation transformation can be controlled by modifying the camera extrinsic matrix accordingly. In the right part of Fig. 3, we show the random rotation augmentation, a type of BEV augmentation.

3.4 Multi-Frame Feature Fusion

Inspired by BEVDet4D [8] and BEVFormer [6], we also introduce the history frame into the current frame for temporal feature fusion. Here we sample the current frame with three history keyframes; each keyframe has a 0.5s interval. We adopted the multi-frame feature alignment method from BEVDet4D. As shown in Fig. 4, after we got four aligned BEV features, we directly concatenate them and feed them to the 3D encoder. In the training phase, the history frame

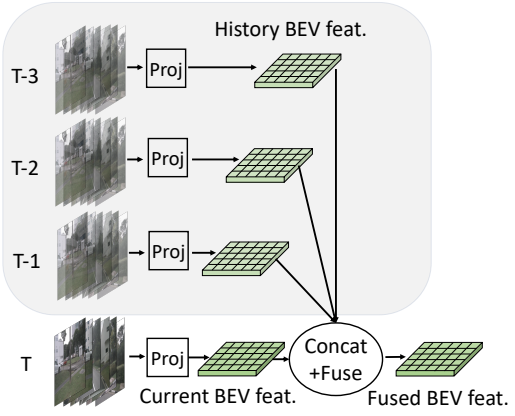

Figure 4: Illustration of the temporal multi-frame feature fusion module. The three history frames are first extracted feature and projected to the respective BEV space, then aligned to the current frame with camera extrinsic and global coordinate. Finally, we directly concatenate these multi-frame BEV features in the channel dimension.

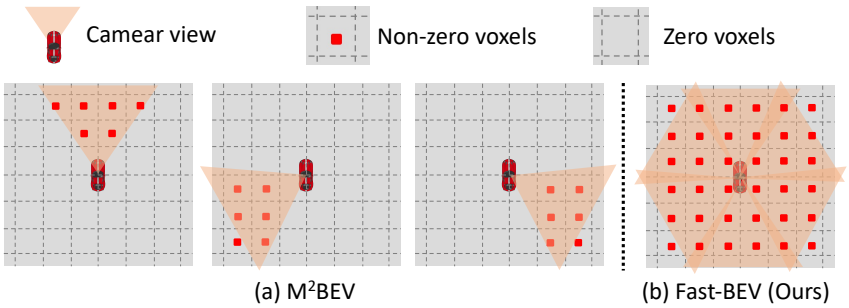

Figure 5: (a) In M^{2}BEV baseline, each camera has one sparse voxel (only $\sim$17% positions are non-zeros). An expensive aggregation operation is needed to combine the sparse voxels. (b) The proposed Fast-BEV let all cameras project to one dense voxel, avoiding the expensive voxel aggregation.

features are extracted online using the image encoder. In the testing phase, the history frame feature can be saved offline and directly taken out for acceleration.

Compare with BEVDet4D and BEVFormer. BEVDet4D only introduces one history frame, which we argue is insufficient to leverage history information. Fast-BEV uses three history frames, resulting in significant performance improvement. BEVFormer is slightly better than BEVDet4D by using two history frames. However, due to memory issues, in the training phase, the history feature is detached without grad, which is not optimal. Moreover, BEVFormer uses an RNN-style to sequentially fuse features, which is inefficient. In contrast, all the frame in Fast-BEV is trained in an end-to-end manner, which is more training friendly with common GPUs.

3.5 Optimized view transformation

View Transformation is the critical component to transform features from 2D image space to 3D BEV space, which typically takes much time in the whole pipeline. Lift-Splat-Shoot [11] is a classic method for view transformation. Although some acceleration techniques [9, 7, 38] have been proposed for advanced GPU devices (*e.g.*, NVIDIA Tesla A100, V100), but the optimization can not be easily transferred to other devices such as edge chips. Another category of view transformation is M^{2}BEV [1] which assumes the depth distribution is uniform along the ray. The advantage is that once we get the intrinsic/extrinsic parameters of cameras, we can easily know the 2D to 3D projection. Since no learnable parameters are used here, we can easily compute the corresponding matrix between points in the 2D feature maps and the BEV feature map. We follow the projection method of M^{2}BEV, and further accelerate it from two perspectives: pre-compute projection index and dense voxel feature generation.

The projection index is the mapping index from 2D image space to 3D voxel space. Because our method do not rely on the data-dependent depth prediction nor transformer, the projection index will be the *same* for every input. Thus, we can pre-compute the fixed projection index and store it. During inference, we can get the projection index via querying the look-up-table, which is a very cheap operation on edge devices. Moreover, if we extend from the single frame to multiple frames, we can also easily pre-compute the intrinsic and extrinsic parameters and pre-align them to the current frame.

M^{2}BEV will store a voxel feature for each camera view and then aggregate them to generate the final voxel feature (see Figure 5). Because each camera only has limited view angle, each voxel feature is very sparse, *e.g.*, only about 17% positions are non-zeros. We identify the aggregation of these voxel features is very expensive due to the their huge size. We propose to generate a dense voxel feature to avoid the expensive voxel aggregation. Specifically, we let image features from all camera views project to the same voxel feature, leading to one dense voxel at the end.

Table 1: Latency profiling for view transformation of different methods. The latency is tested on Tesla A100 GPU and Intel i7 CPU in milliseconds (ms).

Method	Projection	GPU Latency	CPU Latency
BEVDet [7]	Depth	$\sim$125	-
BEVDepth [9]	Depth	$\sim$1.2	-
BEVFormer [6]	Transformer	25	-
M^{2}BEV [1]	None	-	54
Fast-BEV (Ours)	None	-	0.8

In Table 1, we profile the view transformation latency of four different methods. We find that (1) BEVDepth [9] achieves the best latency on GPU but it requires dedicated parallel computing support, making it not applicable to CPU. (2) Compared with the M^{2}BEV baseline, the proposed Fast-BEV achieves orders of magnitude speedup on CPU.

4 Experiments

4.1 Setup

Dataset Description We evaluate our Fast-BEV on nuScenes dataset [10], which contains 1000 autonomous driving scenes with 20 seconds per scene. The dataset is split into 850 scenes for training/validation and the rest 150 for testing. While nuScenes dataset provides data from different sensors, we only use the camera data. The cameras have six views: `front_left`, `front`, `front_right`, `back_left`, `back`, `back_right`.

Evaluation metrics. To comprehensively evaluate the detection task, we use the standard evaluation metrics of mean Average Precision (mAP), and nuScenes detection score (NDS) for 3D object detection evaluation. In addition, in order to calculate the precision of the corresponding aspects (*e.g.*, translation, scale, orientation, velocity, and attribute), we use the mean Average Translation Error (mATE), mean Average Scale Error (mASE), mean Average Orientation Error(mAOE), mean Average Velocity Error(mAVE), and mean Average Attribute Error(mAAE) as the metric.

Implementation Details. For training, we use AdamW optimizer with learning rate $1e^{-3}$ and the weight decay is set to $1e^{-2}$. "Polylr" scheduler is adopted to gradually decrease the learning rate. We also use "warmup" strategy for the first 1000 iterations. For data augmentation hyper-parameters, we basically follow BEVDet [7]. Without specific notification, all models are trained on 32 A100 GPUs for 48 epochs. For comparison with state-of-the-art, we train 20 epochs with CBGS [39]. We mainly use ResNet-50 as the backbone for the ablation study to verify the ideas quickly. For on-vehicle inference speed test, we test with one sample per batch which contains 6 view images.

4.2 Compare with state-of-the-art methods

We comprehensively compare the proposed Fast-BEV with the baseline method M^{2}BEV [1] and other recent methods like FCOS3D [16], PETR [33], BEVDet [7], DETR3d [13], BEVDet4D [8], BEVFormer [6] and BEVDepth [9] on nuScenes val set.

As shown in Table 2, Fast-BEV shows superior performance in mAP and NDS compare with existing advanced methods. For example, with ResNet-50 as the backbone, Fast-BEV achieves 0.346 mAP and 0.477 NDS, significantly outperforms BEVDet4D-Tiny [8] (0.323 mAP and 0.453 NDS). The model also exceeds other methods with a larger input resolution such as PETR-R50 [33] (0.313 mAP and 0.381 NDS). Moreover, with the larger backbone ResNet-101 and higher image resolution, Fast-BEV establish a new state-of-the-art 0.535 NDS, exceeding BEVDet4D-Base [8] 0.515 NDS and BEVFormer-R101 [6] 0.517 NDS without depth/lidar supervision.

Efficient Model Series. In order to fit different deployment scenarios, we design a series of efficient models from M0 to M4 as shown in Table 3. We set different 2D encoder, image resolution, voxel resolution and 3D encoder to design the model sizes. Our M1 model (R18@256×704) can run over 50FPS on the Tesla T4 platform, with 0.470 NDS on the nuScenes validation set.

We also breakdown the latency into three parts in the last column of Table 3. As the 2D encoder grow larger from R18 to R50, the image resolution increase from 250×704 to 384×1056, the latency of this part increases by nearly 3 times. At the same time, when the voxel resolution increases from $200 \times 200 \times 4$ to $300 \times 300 \times 6$, the 3D encoder grows larger from 2b-192c to 6b-256c, the latency of view transformation and 3D encoder increases dramatically by nearly 4 times and 5 times, respectively.

4.3 Detailed Analysis

Unless explicitly stated, Fast-BEV analysis experiments for the detection task in this section use the ResNet-50 backbone and 2 frames to train 48 epochs. And the image resolution and voxel resolution are 250×704 and $200 \times 200 \times 6$, respectively.

Table 2: Comparison on the nuScenes *val* set. "L" denotes LiDAR, "C" denotes camera and "D" denotes Depth/LiDAR supervision. "¶" indicates our method with scale NMS and test-time augmentation.

Methods	Image Res.	Modality	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓	NDS↑
CenterPoint-Voxel [34]	-	L	0.564	-	-	-	-	-	0.648
CenterPoint-Pillar [34]	-	L	0.503	-	-	-	-	-	0.602
FCOS3D [16]	900×1600	C	0.295	0.806	0.268	0.511	1.315	0.170	0.372
BEVDet-R50 [7]	256×704	C	0.286	0.724	0.278	0.590	0.873	0.247	0.372
PETR-R50 [33]	384×1056	C	0.313	0.768	0.278	0.564	0.923	0.225	0.381
PETR-Tiny [33]	512×1408	C	0.361	0.732	0.273	0.497	0.808	0.185	0.431
BEVDet4D-Tiny [8]	256×704	C	0.323	0.674	0.272	0.503	0.429	0.208	0.453
BEVDepth-R50 [9]	256×704	C&D	0.351	0.639	0.267	0.479	0.428	0.198	0.475
Fast-BEV(R50)	256×704	C	0.334	0.665	0.285	0.393	0.388	0.210	0.473
Fast-BEV(R50)¶	256×704	C	0.346	0.667	0.285	0.401	0.393	0.208	**0.477**
FCOS3D-R101† [16]	900×1600	C	0.321	0.754	0.260	0.486	1.331	0.158	0.395
DETR3D-R101† [13]	900×1600	C	0.347	0.765	0.267	0.392	0.876	0.211	0.422
Ego3RT-V2-99† [40]	900×1600	C	0.478	0.582	0.272	0.316	0.683	0.202	0.534
M^2BEV-X101† [1]	900×1600	C	0.417	0.647	0.275	0.377	0.834	0.245	0.470
PolarFormer-T-R101† [41]	900×1600	C	0.432	0.648	0.270	0.348	0.409	0.201	0.528
PETRv2-VoVNet-99† [37]	320×800	C	0.401	0.745	0.268	0.448	0.394	0.184	0.496
DETR3D [13]	900×1600	C	0.303	0.860	0.278	0.437	0.967	0.235	0.374
BEVDet-Base [7]	512×1408	C	0.349	0.637	0.269	0.490	0.914	0.268	0.417
PETR-R101 [33]	512×1408	C	0.357	0.710	0.270	0.490	0.885	0.224	0.421
BEVDet4D-Base [8]	640×1600	C	0.396	0.619	0.260	0.361	0.399	0.189	0.515
BEVFormer-R101 [6]	900×1600	C	0.416	0.673	0.274	0.372	0.394	0.198	0.517
BEVDepth-R101 [9]	512×1408	C&D	0.412	0.565	0.266	0.358	0.331	0.190	**0.535**
Fast-BEV(R101)	900×1600	C	0.402	0.582	0.278	0.304	0.328	0.209	0.531
Fast-BEV(R101)¶	900×1600	C	0.413	0.584	0.279	0.311	0.329	0.206	**0.535**

Table 3: The modular design of the serial models denoted as M0-4 are presented. The type of 2d encoder is chosen from R18/R34/R50. Voxel resolution is denoted as x-y-z for the view transformation from 2d features to BEV features. The number of block x and channel y denoted as xb-yc are clarified for 3D encoders. The latency on T4 platform are evaluated with the sum of 3 parts including a 2D encoder, view transformation and a 3D encoder (from left to right in the latency breakdown column).

Name	2D Encoder	Image Res.	Voxel Res.	3D Encoder	mAP	NDS	Latency (ms)	Latency breakdown
M0	R18	256×704	200×200×4	2b-192c	0.284	0.427	14.5	4.6 / 2 / 7.9
M1	R34	256×704	200×200×4	4b-224c	0.326	0.470	16.9	5.8 / 2 / 9.1
M2	R34	320×880	250×250×6	4b-224c	0.332	0.472	35.2	8.6 / 4.8 / 21.8
M3	R50	320×880	250×250×6	6b-256c	0.346	0.482	39.3	10.8 / 4.8 / 23.7
M4	R50	384×1056	300×300×6	6b-256c	0.349	0.488	57.2	14.6 / 7.6 / 35.0

Augmentation. As shown in Table 4, We observe that the performance is significantly improved whether using image augmentation or BEV augmentation alone. For image augmentation, mAP and NDS increased by 3.8% and 2.8%, respectively. For BEV augmentation, mAP and NDS increased by 1.8% and 2.3%, respectively. When the two augmentation are used together, the mAP and NDS can be further improved by 4.6% and 4.4%.

Multi-Frame Feature Fusion. To investigate the effectiveness of multi-frame feature fusion, we show an ablation study in Table 5. When adding one history frame, the mAP and NDS are significantly improved with 2.8% and 7.8%. When further increase to four history frames, the mAP and NDS continue to improve by with 3.0% and 9.3%, showing that temporal information is important for 3D detection.

Table 4: Ablation study of different augmentations with single frame.

Method	mAP	NDS
Baseline	0.247	0.329
+ImgAug	0.285	0.357
+BEVAug	0.265	0.352
+ImgBEVAug	0.293	0.373

Table 5: Ablation study of sequential feature fusion from single frame to 4 frames.

Method	mAP	NDS
1F	0.293	0.373
2F	0.321	0.451
4F	0.323	0.466

Table 6: Ablation study of voxel resolution. The resolution of the image is 384×1056.

Voxel Res.	mAP	NDS
200×200×6	0.352	0.476
200×200×12	0.350	0.474
400×400×6	0.337	0.467
400×400×12	0.345	0.476

Table 7: Ablation study of image resolution.

Voxel Res.	mAP	NDS
256×448	0.280	0.419
256×704	0.321	0.451
464×800	0.342	0.466
544×960	0.345	0.472
704×1208	0.358	0.478
832×1440	0.368	0.491
928×1600	0.369	0.488

Table 8: Ablation study of baseline and Fast-BEV epochs.

Method	Epochs	mAP	NDS
Baseline	12	0.258	0.330
	24	0.257	0.338
	36	0.248	0.320
	48	0.247	0.327
Fast-BEV	12	0.273	0.381
	24	0.294	0.424
	36	0.310	0.440
	48	0.321	0.451

Table 9: Ablation study of different 2D and 3D encoders with 4 frames and 20 epochs with CBGS.

Encoder	Type	mAP	NDS
2D	R18	0.293	0.437
	R50	0.335	0.473
	R101	0.345	0.482
3D	2×Block	0.320	0.446
	4×Block	0.315	0.448
	6×Block	0.321	0.451

Resolution. To investigate the effect of different resolutions of the input image and voxel, we perform an ablation study in Table 7 and Table 6. We first fix the voxel resolution to 200×200×6, and discretely take different image resolutions from 256×448 to 928×1600 for verification. The results in Table 7 show that the increasing in resolution greatly helps to improve the performance of the model, and Fast-BEV achieves the best 49.1% NDS with 832×1440 input image size.

We then fix the image resolution to 384×1056, and try to use different voxel resolutions as shown in Table 6. We observe that $200 \times 200 \times 6$ works well for 3D detection, and increase the resolution from the spatial plane or the height dimension does not help improving the performance.

2D/3D Encoder. To evaluate the performance of different 2D encoders, we perform an ablation study in Table 9 (upper). As the encoder grows larger from ResNet-18 to ResNet-101, the mAP and NDS increases by a large margin with over 5% and 4%. In terms of 3D encoders, as shown in Table 9 (lower), when the encoder grows larger from 2 blocks to 6 blocks, the mAP and NDS for detection task increases by 0.1% and 0.5% respectively. The scale of the 2D encoder has a greater impact on performance than the 3D encoder.

Epoch. To investigate the influence of training epochs, we do an ablation study in Table 8. We observe that baseline and Fast-BEV achieve the best 33.8% NDS and 45.1% NDS at epoch 24 and 48, respectively. We find that the upper-bound of Fast-BEV is much higher than the baseline. And Fast-BEV requires more training epochs to achieve better results, because of the strong data augmentation and temporal feature fusion.

5 Conclusions

In this paper, we propose Fast-BEV, a stronger and faster fully convolutional BEV perception framework which is suitable for on-vehicle depolyment. Compared with the M^{2}BEV baseline, the new Fast-BEV introduces the strong 3D augmentation and temporal feature information, which largely boosts up the performance. We also optimize the view transformation to make it more deployment friendly. We propose to pre-compute the projection index and let all camera project to the same dense voxel. We hope our work can shed light on the industrial-level, real-time, on-vehicle BEV perception.

References

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

[2] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018.

[3] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12697–12705, 2019.

[4] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019.

[5] Lidar price for cars - neuvition: Solid-state lidar,lidar sensor suppliers, lidar technology, lidar sensor, Feb 2022.

[6] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Qiao Yu, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. *arXiv preprint arXiv:2203.17270*, 2022.

[7] Junjie Huang, Guan Huang, Zheng Zhu, and Dalong Du. Bevdet: High-performance multi-camera 3d object detection in bird-eye-view. *arXiv preprint arXiv:2112.11790*, 2021.

[8] Junjie Huang and Guan Huang. Bevdet4d: Exploit temporal cues in multi-camera 3d object detection. *arXiv preprint arXiv:2203.17054*, 2022.

[9] Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. *arXiv preprint arXiv:2206.10092*, 2022.

[10] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

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

[12] Kashyap Chitta, Aditya Prakash, and Andreas Geiger. Neat: Neural attention fields for end-to-end autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15793–15803, 2021.

[13] Yue Wang, Vitor Campagnolo Guizilini, Tianyuan Zhang, Yilun Wang, Hang Zhao, and Justin Solomon. Detr3d: 3d object detection from multi-view images via 3d-to-2d queries. In *Conference on Robot Learning*, pages 180–191. PMLR, 2022.

[14] Garrick Brazil and Xiaoming Liu. M3d-rpn: Monocular 3d region proposal network for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9287–9296, 2019.

[15] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9627–9636, 2019.

[16] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. Fcos3d: Fully convolutional one-stage monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 913–922, 2021.

[17] Tai Wang, ZHU Xinge, Jiangmiao Pang, and Dahua Lin. Probabilistic and geometric depth: Detecting objects in perspective. In *Conference on Robot Learning*, pages 1475–1485. PMLR, 2022.

[18] Dennis Park, Rares Ambrus, Vitor Guizilini, Jie Li, and Adrien Gaidon. Is pseudo-lidar needed for monocular 3d object detection? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3142–3152, 2021.

[19] Nikhil Gosala and Abhinav Valada. Bird's-eye-view panoptic segmentation using monocular frontal view images. *IEEE Robotics and Automation Letters*, 7(2):1968–1975, 2022.

[20] Bowen Pan, Jiankai Sun, Ho Yin Tiga Leung, Alex Andonian, and Bolei Zhou. Cross-view semantic segmentation for sensing surroundings. *IEEE Robotics and Automation Letters*, 5(3):4867–4873, 2020.

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

[22] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[23] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2446–2454, 2020.

[24] Yan Wang, Wei-Lun Chao, Divyansh Garg, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. 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*, pages 8445–8453, 2019.

[25] Yurong You, Yan Wang, Wei-Lun Chao, Divyansh Garg, Geoff Pleiss, Bharath Hariharan, Mark Campbell, and Kilian Q Weinberger. Pseudo-lidar++: Accurate depth for 3d object detection in autonomous driving. *arXiv preprint arXiv:1906.06310*, 2019.

[26] Rui Qian, Divyansh Garg, Yan Wang, Yurong You, Serge Belongie, Bharath Hariharan, Mark Campbell, Kilian Q Weinberger, and Wei-Lun Chao. End-to-end pseudo-lidar for image-based 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5881–5890, 2020.

[27] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. 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*, pages 2397–2406, 2022.

[28] Cody Reading, Ali Harakeh, Julia Chae, and Steven L Waslander. Categorical depth distribution network for monocular 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8555–8564, 2021.

[29] Thomas Roddick, Alex Kendall, and Roberto Cipolla. Orthographic feature transform for monocular 3d object detection. *arXiv preprint arXiv:1811.08188*, 2018.

[30] Weixiang Yang, Qi Li, Wenxi Liu, Yuanlong Yu, Yuexin Ma, Shengfeng He, and Jia Pan. Projecting your view attentively: Monocular road scene layout estimation via cross-view transformation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15536–15545, 2021.

[31] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020.

[32] Zehui Chen, Zhenyu Li, Shiquan Zhang, Liangji Fang, Qinhong Jiang, and Feng Zhao. Graph-detr3d: Rethinking overlapping regions for multi-view 3d object detection. *arXiv preprint arXiv:2204.11582*, 2022.

[33] Yingfei Liu, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. *arXiv preprint arXiv:2203.05625*, 2022.

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

[35] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018.

[36] Avishkar Saha, Oscar Mendez Maldonado, Chris Russell, and Richard Bowden. Translating images into maps. *arXiv preprint arXiv:2110.00966*, 2021.

[37] Yingfei Liu, Junjie Yan, Fan Jia, Shuailin Li, Qi Gao, Tiancai Wang, Xiangyu Zhang, and Jian Sun. Petrv2: A unified framework for 3d perception from multi-camera images. *arXiv preprint arXiv:2206.01256*, 2022.

[38] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *arXiv preprint arXiv:2205.13542*, 2022.

[39] Benjin Zhu, Zhengkai Jiang, Xiangxin Zhou, Zeming Li, and Gang Yu. Class-balanced grouping and sampling for point cloud 3d object detection. *arXiv preprint arXiv:1908.09492*, 2019.

[40] Jiachen Lu, Zheyuan Zhou, Xiatian Zhu, Hang Xu, and Li Zhang. Learning ego 3d representation as ray tracing. *arXiv preprint arXiv:2206.04042*, 2022.

[41] Yanqin Jiang, Li Zhang, Zhenwei Miao, Xiatian Zhu, Jin Gao, Weiming Hu, and Yu-Gang Jiang. Polarformer: Multi-camera 3d object detection with polar transformers. *arXiv preprint arXiv:2206.15398*, 2022.