GraphBEV: Towards Robust BEV Feature Alignment for Multi-Modal 3D Object Detection

Ziying Song[1,2], Lei Yang[3], Shaoqing Xu[4], Lin Liu[1,2], Dongyang Xu[3], Caiyan Jia[1,2] ⋆, Feiyang Jia[1,2], and Li Wang[5]

[1] School of Computer and Information Technology, Beijing Jiaotong University
[2] Beijing Key Lab of Traffic Data Analysis and Mining, China
[3] School of Vehicle and Mobility, Tsinghua University
[4] Department of Electrome chanical Engineering, University of Macau
[5] School of Mechanical Engineering, Beijing Institute of Technology
{songziying,cyjia}@bjtu.edu.cn
https://github.com/adept-thu/GraphBEV

Abstract. Integrating LiDAR and camera information into Bird's-Eye-View (BEV) representation has emerged as a crucial aspect of 3D object detection in autonomous driving. However, existing methods are susceptible to the inaccurate calibration relationship between LiDAR and the camera sensor. Such inaccuracies result in errors in depth estimation for the camera branch, ultimately causing misalignment between LiDAR and camera BEV features. In this work, we propose a robust fusion framework called GraphBEV. Addressing errors caused by inaccurate point cloud projection, we introduce a LocalAlign module that employs neighbor-aware depth features via Graph matching. Additionally, we propose a GlobalAlign module to rectify the misalignment between LiDAR and camera BEV features. Our GraphBEV framework achieves state-of-the-art performance, with an mAP of 70.1%, surpassing BEVFusion by 1.6% on the nuScenes validation set. Importantly, our GraphBEV outperforms BEVFusion by 8.3% under conditions with misalignment noise.

Keywords: 3D Object Detection · Multi-Modal Fusion · Feature Alignment · Bird's-Eye-View (BEV) Representation

1 Introduction

3D object detection is a critical component of autonomous driving systems, designed to accurately identify and localize objects such as cars, pedestrians, and other elements within a 3D environment [51, 61]. Current practices predominantly adhere to a multi-modal fusion paradigm such as BEVFusion [29, 35] for robust and high-quality detection. Different modalities often provide complementary information. For instance, images contain rich semantic representations but lack depth information. In contrast, point clouds offer geometric and depth

---

⋆ Corresponding author

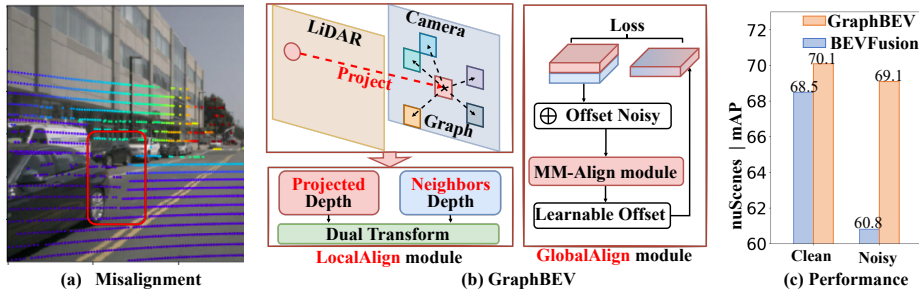**(a) Misalignment**   **(b) GraphBEV**   **(c) Performance**

Fig. 1: (a) Feature misalignment primarily arises from overlooking the projection matrix errors between LiDAR and camera, leading to LiDAR-to-camera providing inaccurate depth for surrounding neighbors. **(b)** We propose GraphBEV, enhancing LiDAR-to-camera projected depth with neighboring depths through graph-based neighbor information construction for enriched contextual depth feature learning. Subsequently, we achieve the alignment of global multi-modal features by simulating LiDAR and camera BEV features' offsets and employing learnable offsets. **(c)** Empirical results reveal that our GraphBEV surpasses the BEVFusion [35] on the nuScenes by a margin of 1.6% mAP on nuScenes validation dataset [4] and by over 8.3% on noisy misalignment settings [12].

information but are sparse and devoid of semantic information. Therefore, effectively leveraging the advantages of multi-modal data while mitigating their limitations is crucial for enhancing the robustness of perception systems [61].

Feature misalignment is a significant challenge in practical applications of multi-modal 3D object detection, primarily caused by calibration matrix errors between LiDAR and camera sensors [12,51,75], as shown in Figure 1(a). Multi-modal 3D object detection has evolved from early point-level [17,34,48,58,59,73] and feature-level [2,9,10,49,52,54] methods to the currently prevalent methods of Bird's Eye View (BEV) fusion like BEVFusion [29,35]. Although effective on clean datasets like nuScenes [4], the performance of BEVFusion [35] deteriorates on misaligned data, as illustrated in Figure 1(c). This performance decline is primarily due to calibration errors between LiDAR and the camera, exacerbated by factors like road vibrations [75]. These inherent errors cannot be corrected through online calibration, presenting a significant challenge [12,75].

Most feature-level multi-modal methods [1,10,24,71,78] employ Cross Attention to query features of a specific modality, circumventing the need for projection matrices. A few feature-level multi-modal methods [9,19,22,23,49,52,74,76] have sought to mitigate these errors through the use of projection offsets or neighboring projections. A few BEV-based methods, such as ObjectFusion [5], eliminate the camera-to-BEV transformation during fusion to align object-centric features across different modalities. MetaBEV [14] utilizes Cross Deformable Attention for feature misalignment but overlooks depth estimation errors in view transformation, aligning features only during LiDAR and camera BEV fusion.

BEVFusion [29,35] unites camera and LiDAR data in BEV space, enhancing detection but overlooks **feature misalignment** in real-world applications. It

is primarily evident in two aspects: 1) BEVFusion [35] transforms multi-image features into a unified BEV representation using BEVDepth's [26] explicit depth supervision from LiDAR-to-camera. While this LiDAR-to-camera strategy offers more reliable depth than LSS [41], it overlooks the misalignment between LiDAR and camera in real-world scenarios, leading to **local misalignment**. 2) In the LiDAR-camera BEV fusion, misalignment of BEV features due to depth inaccuracies is overlooked by directly concatenating representations and applying basic convolution, as described in BEVFusion [35], leading to **global misalignment**.

In this work, we propose a robust fusion framework, named GraphBEV to address the above feature misalignment problems. To tackle the local misalignment issue in camera-to-BEV transformation, we propose a LocalAlign module that first obtains neighbor depth information through a Graph in the View Transformation step of BEVFusion's [35] camera Branch, based on the explicit depth provided by the LiDAR-to-camera projection. Subsequently, we propose a GlobalAlign module that encodes the projected depth from LiDAR-to-camera and the neighboring depth through dual depth encoding to generate a new Reliable depth representation incorporating neighbor information. Furthermore, to address the global misalignment problem in the fusion of LiDAR and camera BEV features, we effectively resolve the global misalignment between the two BEV feature representations by dynamically generating offsets. To validate the effectiveness of GraphBEV, we evaluated it on the nuScenes dataset, a well-known benchmark for 3D object detection. Our GraphBEV not only achieves state-of-the-art (SOTA) performance under clean settings but also outperforms BEVFusion by over 8.3% on non-aligned noisy settings nuScenes dataset provided by Dong et al. [12]. Notably, our GraphBEV experiences only a slight performance drop under noisy settings compared to clean settings. Notably, our GraphBEV bolsters the real-world applicability of BEV-based methods [29, 35] by addressing sensor calibration errors. Our contributions are summarized below:

1. We propose a robust fusion framework, named GraphBEV, to address feature misalignment arising from projection errors between LiDAR and camera inputs.
2. By deeply analyzing the fundamental causes of feature misalignment, we propose LocalAlign and GlobalAlign modules within our GraphBEV framework to address local misalignments from imprecise depth and global misalignments between LiDAR and camera BEV features.
3. Extensive experiments validate the effectiveness of our GraphBEV, demonstrating competitive performance on the nuScenes dataset. Notably, GraphBEV maintains comparable performance across both clean settings and misaligned noisy conditions.

2   Related Work

2.1   LiDAR-based 3D Object Detection

LiDAR-based 3D object detection methods can be categorized into three primary types based on point cloud representation: Point-based, Voxel-based, and

PV-based (Point-Voxel). Point-based methods [27, 42–44, 46] extend PointNet's [43, 44] principle, directly processing raw point clouds with stacked Multi-Layer Perceptrons (MLPs) to extract point features. Voxel-based methods [8, 11, 30, 60, 66, 79] typically convert point clouds into voxels and apply 3D sparse convolutions for voxel feature extraction. In addition, PointPillars [21] converts irregular raw point clouds into pillars and encodes them on a 2D backbone, achieving a very high FPS. Some Voxel-based methods [13, 15, 37, 62] further exploit Transformers [57] post-voxelization to capture long-range voxel relationships. PV-based methods [36, 38, 45, 53, 70] combine voxel and point-based strategies, extracting features from point clouds' diverse representations using both approaches, achieving higher accuracy albeit with increased computational demand.

2.2   Camera-based 3D Object Detection

Camera-based 3D object detection methods have gained increasing attention in academia and industry, mainly due to the significantly lower cost of camera sensors compared to LiDAR [51]. Early methods [3, 47, 64] focused on augmenting 2D object detectors with additional 3D bounding box regression heads. Current camera-based methods have rapidly evolved since LSS [41] introduced the concept of unifying multi-view information onto a bird's-eye view (BEV) through 'Lift, splat.' LSS-based methods [25, 26, 39, 41, 68, 69] like BEVDepth [26] extract 2D features from multi-view images and provide effective depth supervision via LiDAR-to-camera projections before unifying multi-view features onto the BEV. Subsequent works [25, 39] have introduced multi-view stereo techniques to improve depth estimation accuracy and achieve SOTA performance. Additionally, inspired by the success of transformer-based architectures such as DETR [6] and Deformable DETR [81] in 2D detection, transformer-based detectors have emerged for 3D object detection. Following DETR3D [63], these methods design a set of object queries [18, 31, 32] or BEV grid queries [28, 67], then perform view transformation through cross-attention between queries and image features.

2.3   Multi-modal 3D Object Detection

Multi-modal 3D object detection refers to using data features from different sensors and integrating these features to achieve complementarity, thus enabling the detection of 3D objects. Previous multi-model methods can be coarsely classified into three types by fusion, i.e., point-level, feature-level, and BEV-based methods. Point-level [17, 34, 48, 58, 59, 73] and feature-level [2, 9, 10, 49, 52, 54, 55, 65, 77] typically leverage image features to augment LiDAR points or 3D object proposals. BEV-based methods [5, 14, 29, 35, 50] efficiently unify the representations of LiDAR and camera into BEV space. Although BEVFusion [29, 35] achieves high performance, they are typically tested on clean datasets like nuScenes [4], overlooking real-world complexities, especially **feature misalignment**, which hampers their application.
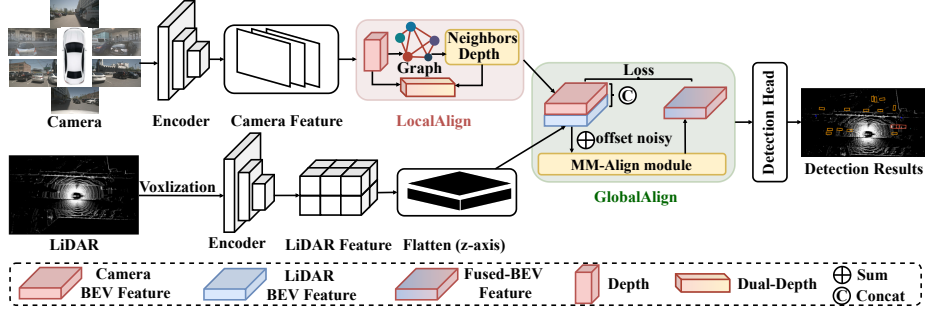
Fig. 2: The overview of **GraphBEV** framework. The LiDAR branch largely follows the baselines [1, 35] to generate LiDAR BEV features. In the camera branch, we first extract camera BEV features using the proposed LocalAlign module, which aims to address local misalignment due to sensor calibration errors. Subsequently, we simulate the offset noise of LiDAR and camera BEV features, followed by aligning global multi-modal features through learnable offsets. Notably, we only add offset noise to the GlobalAlign module during **training** to simulate global misalignment issues. Finally, we employ a dense detection head [1] to accomplish the 3D detection task.

3   Method

To address the **feature misalignment** issue in previous BEV-based methods [29, 35], we propose a robust fusion framework named **GraphBEV**. We provide an overview of our framework in Figure 2. Taking input from different sensors, including LiDAR and camera, we first apply modality-specific encoders, Swin-Transformer [33] as camera encoder and Second [66] as LiDAR encoder, to extract their respective features. Then, the camera features are transformed into the camera BEV feature through our proposed **LocalAlign** module, aiming to mitigate the local misalignment caused by the projection error between LiDAR and the camera in the camera-to-BEV process of previous BEV-based methods [29, 35]. The LiDAR features are then compressed along the Z-axis to represent 3D features as 2D LiDAR BEV features. Next, we propose a **GlobalAlign** module that further alleviates global misalignment between different modalities, including LiDAR and camera BEV features. Finally, we append detection head [1] to accomplish the 3D detection task. Our baseline is the BEVFusion [35], within which we detail the introduction of the LocalAlign and GlobalAlign modules below.

3.1   LocalAlign

To facilitate the transformation of camera features into BEV features, BEVFusion [29, 35] adopts the LSS-based [41] methods like BEVDepth [26] leverages LiDAR-to-camera to provide projected depth, thereby enabling the fusion of depth and image features. In the process of camera-to-BEV, BEVFusion [35] and BEVDepth [26] operate under the assumption that the depth information
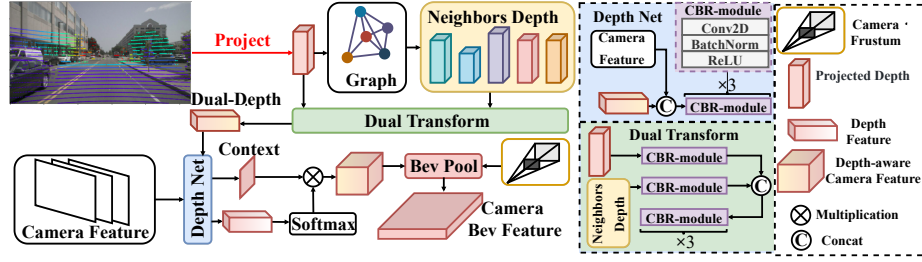
Fig. 3: The overview of **LocalAlign** pipeline. The LocalAlign module addresses Local Misalignment from LiDAR-to-camera by enhancing the camera-to-BEV Transform with neighboring depth features using a KD-Tree algorithm for nearest-neighbor relations.

provided by LiDAR-to-camera projection is accurate and reliable. However, they overlook the complexities inherent in real-world scenarios, where most of the projection matrices between LiDAR and cameras are calibrated manually. Such calibration inevitably introduces projection errors, leading to **depth misalignment**—where the depths of surrounding neighbors are projected as the pixel's depth. This depth misalignment results in inaccuracies within the depth features, causing misalignment during multi-view transformation into BEV representations. Given that LSS-based methods [26,41] rely on depth estimation from pixel-level features with inaccuracies in detailing, this leads to **local misalignment** within camera BEV features. It underscores the challenges of ensuring precise depth estimation within the BEVFusion [35] and highlights the importance of robust methods to address projection errors.

We propose a LocalAlign module to address Local Misalignment, with its pipeline depicted in Figure 3. Specifically, LiDAR-to-camera provides projected depth, defined as $D_S \in \mathbb{R}^{B_S \times N_C \times 1 \times H \times W}$, where B_S represents the batch size, N_C denotes the number of multi-views (six in the case of nuScenes), and H and W are the height and width of the images, respectively. The projection from LiDAR-to-camera maps 3D point clouds onto an image plane, from which we can obtain the indices of the projected pixels, defined as $M_{\text{Coords}} \in \mathbb{R}^{N_P \times 2}$, where N_P refers to the number of points projected onto pixels, and 2 represents the pixel coordinates (u, v) as illustrated below.

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = h\mathcal{K} \begin{bmatrix} R\ T \end{bmatrix} \begin{bmatrix} P_x \\ P_y \\ P_z \\ 1 \end{bmatrix} \tag{1}$$

where, P_x, P_y, P_z denote the LiDAR point's 3D locations, u, v denote the corresponding 2D locations, and z_c represents the depth of its projection on the image plane, $\mathcal{K}$ denotes the camera intrinsic parameter, R and T denote the rotation and the translation of the LiDAR with respect to the camera reference system, and h denotes the scale factor due to down-sampling.

Algorithm 1: Graph for Finding Neighbors

Input:

The indices of the projected pixels $M_{\text{Coords}} \in \mathbb{R}^{N_P \times 2}$.

Hyper-parameter: Number of neighbors $K_{\text{graph}} = 8$.

Output: Neighbors $M_{K_{\text{Coords}}} \in \mathbb{R}^{N_P \times K_{\text{graph}} \times 2}$.

1 while *LocalAlign* **do**

2   **Function** *KD-Tree (M_{Coords}, $M_{Coordsi}$, K_{graph})*

3   Compute the Euclidean distance between $M_{\text{Coords}i}$ and M_{Coords}

4   Indices = argsort(distances)

5   **return** $M_{\text{Coords}}[1 : K_{\text{graph}}]$

6   **for** $i = 1 \ldots N_P$ **do**

7   Neighbors = KD-Tree(M_{Coords}, $M_{\text{Coords}i}$, K_{graph})

8 $M_{\text{Coords}i}$ = Neighbors

9   **end**

10 end

We employ the KD-Tree algorithm to obtain the indices of the projected pixels' neighbors, defined as $M_{K_{\text{Coords}}} \in \mathbb{R}^{N_P \times K_{\text{graph}} \times 2}$, where K_{graph} denotes the number of neighbors for each projected pixel. The process is outlined in Algorithm (1). It is worth noting that we simplified the process of the KD-Tree algorithm, and the code can be referred to in scipy[6]. Then, we obtain the surrounding neighbor depth $D_K \in \mathbb{R}^{B_S \times N_C \times K_{\text{graph}} \times H \times W}$ by indexing D_S with M_{Coords}. Then, D_S and D_K simultaneously enter the Dual Transform module for deep feature encoding. The shapes of D_S and D_K are respectively modified to $[B_S \times N_C, 1, H, W]$ and $[B_S \times N_C, K_{\text{graph}}, H, W]$ before being fed into the Dual Transform module. This module comprises straightforward components, including convolutional layers, Batch Normalization, and ReLU activations, as illustrated in Figure 3. The outcome of this process is the Dual Depth feature, denoted as D_{SK}, and its shape is $[B_S \times N_C, C_{\text{SK}}, \frac{H}{8}, \frac{W}{8}]$. The camera Encoder outputs multi-scale image features from the FPN, including $F_{\text{Cam.}} \in \mathbb{R}^{(B_S \times N_C) \times C_{\text{Cam}} \times \frac{H}{8} \times \frac{W}{8}}$ for richer semantic information and another at a reduced resolution of $\frac{H}{16}, \frac{W}{16}$. We opt to utilize the feature with the resolution $\frac{H}{8}, \frac{W}{8}$ due to its more comprehensive semantic content.

The design of DepthNet is straightforward, as illustrated in Figure 3. We input both $F_{\text{Cam.}}$ and D_{SK} into DepthNet to fuse depth features with multi-view camera features. Initially, $F_{\text{Cam.}}$ and D_{SK} are concatenated, followed by processing through three sets of CBR-module. This results in the generation of a new depth-aware camera feature, denoted as $F_{\text{DC}} \in \mathbb{R}^{(B_S \times N_C) \times C_{\text{DC}} \times \frac{H}{8} \times \frac{W}{8}}$. Subsequently, F_{DC} is split along the C_{DC} dimension into two new features: a novel depth feature, define as $\hat{F_D} \in \mathbb{R}^{(B_S \times N_C) \times \hat{C_D} \times \frac{H}{8} \times \frac{W}{8}}$ and a novel image context feature, define as $\hat{F_C} \in \mathbb{R}^{(B_S \times N_C) \times \hat{C_C} \times \frac{H}{8} \times \frac{W}{8}}$. It's important to note that $C_{\text{DC}} = \hat{C_C} + \hat{C_D}$, indicating the division of the combined feature space into distinct depth and image feature components. Subsequently, $\hat{F_D}$ is subjected to a soft-

---

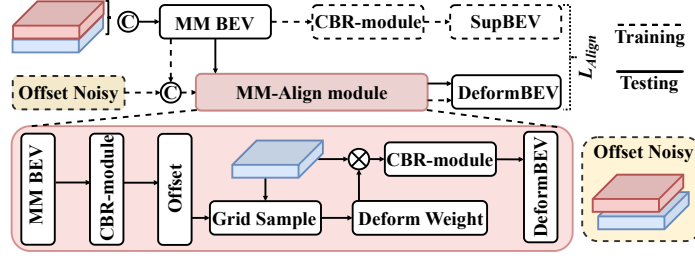[6] https://github.com/minrk/scipy-1/blob/master/scipy/spatial/ckdtree.c

Fig. 4: The overview of **GlobalAlign** pipeline. The GlobalAlign module addresses the issue of misalignment in LiDAR-camera BEV feature fusion. During training, we add offset noise to simulate the global misalignment problem in the camera and LiDAR BEV features. It is supervised through a simple CBR-module to learn the offsets of camera BEV features. We do not introduce noise during testing and employ learnable offsets for forward inference.

max operation and then multiplied with $\hat{F_C}$, resulting in a novel image feature with depth information, represented as $\hat{F}_{\mathrm{DC}} \in \mathbb{R}^{(B_S \times N_C) \times \hat{C_C} \times \hat{C_D} \times \frac{H}{8} \times \frac{W}{8}}$. Finally, adopting operations consistent with LSS [41] and BEVDepth [26], we utilize pre-generated 3D spatial coordinates and $\hat{F_{DC}}$ with BEV Pooling to output the camera BEV feature, thereby completing the camera-to-BEV transformation, and finally outputs the camera BEV feature, define as $F_B^C \in \mathbb{R}^{B_S \times \hat{C_C} \times H_B \times W_B}$.

3.2   GlobalAlign

In the real world, feature misalignment due to calibration matrix discrepancies between LiDAR and camera sensors is inevitable. While the LocalAlign module mitigates local misalignment issues in the camera-to-BEV process, deviations may still exist in camera-BEV features. During LiDAR-camera BEV fusion, despite being in the same spatial domain, inaccuracies in depth from the view transformer and overlooking of global offsets between LiDAR BEV and camera BEV features lead to global misalignment.

   To tackle the global misalignment issue described above, we introduce the GlobalAlign module, employing learnable offsets to achieve alignment of global multi-modal BEV features. As shown in Figures 2 and 4, we use clean datasets such as nuScenes [4] for training, which exhibit minimal deviations that can be considered negligible. The supervised information is derived from the features obtained after the fusion and convolution of LiDAR and camera BEV features. During training, we introduce global offset noise and employ learnable offsets. In the LiDAR branch, the LiDAR feature is flattened along the Z-axis to form the LiDAR BEV feature, defined as $F_B^L \in \mathbb{R}^{B_S \times \hat{C_L} \times H_B \times W_B}$. Initially, we concatenate F_B^L and F_B^C to obtain a fused BEV feature, denoted as $F_B^{MM} \in \mathbb{R}^{B \times (\hat{C_C} + \hat{C_L}) \times H_B \times W_B}$. Subsequently, F_B^{MM} undergoes a convolution operation, resulting in a new fused feature, denoted as $\hat{F_B} \in \mathbb{R}^{B_S \times \hat{C_L} \times H_B \times W_B}$. Notably, $\hat{F_B}$ will be utilized as a supervision signal during the training process.

As shown in Figure 4, we introduce random offset noise to the camera dimension of F_B^{MM} to obtain a new noisy feature $F_N^{MM} \in \mathbb{R}^{B_S \times (\hat{C}_C + \hat{C}_L) \times H_B \times W_B}$, simulating global misalignment issues originating from camera BEV features. Notably, the LiDAR BEV feature is directly flattened, thus more accurate. Then, F_N^{MM} is input into the MM-Align module for global offset learning. F_N^{MM} is processed through the CBR-module with basic convolution operations to learn offsets, defined as $F^O \in \mathbb{R}^{B_S \times 2 \times H_B \times W_B}$, where two refers to the offset coordinates (u, v). Subsequently, LiDAR BEV features F_B^L and F^O undergo grid sampling to generate new deform weights, defined as $F_W^D \in \mathbb{R}^{B_S \times \hat{C}_L \times H_B \times W_B}$. The purpose of grid sampling is to utilize offsets for spatial transformation of LiDAR BEV feature F_B^L, with learnable shifts dynamically adjusting to capture spatial dependencies more flexibly than standard convolution operations. Afterward, F_W^D is multiplied by LiDAR BEV features F_B^L to dynamically adjust features, followed by standard convolution operations through the CBR-module, culminating in the output Deform BEV defined as $F_B^D \in \mathbb{R}^{B_S \times \hat{C}_L \times H_B \times W_B}$. Finally, during training, we supervise F_B^D using $\hat{F}_B$ previously mentioned, and employ the L_{Align} for supervision as follows:

$$L_{\text{Align}} = \frac{1}{N_B} \sum_{i=1}^{N_B} (\hat{F}_{Bi} - F_{B\,i}^D)^2 \tag{2}$$

wherein, $N_B = B_S \times H_B \times W_B$ represents the total number of elements, and $\hat{F}_{Bi}$ and $F_{B\,i}^D$ denote the value of the ith element in $\hat{F}_B$ and F_B^D, respectively. This formula calculates the mean of the squared differences between corresponding positions in the two feature maps, serving as the loss.

4   Experiments

In this section, we present the experimental setup of GraphBEV and evaluate the performance of 3D object detection on the nuScenes [4] dataset. Additionally, we followed the solution provided by Ref. [12] to simulate feature misalignment scenarios.

4.1   Experimental Setup

Dataset and Metric. We evaluate GraphBEV on the challenging large-scale nuScenes dataset [4], which was collected using a 32-beam LiDAR and six cameras. The dataset is typically split into 700/150/150 scenes for training, validation, and testing. The six images cover a 360-degree view of the surroundings, and the dataset provides calibration matrices that enable precise projection from 3D points to 2D pixels. We use mAP and NDS across all categories as the primary metrics for evaluation, following [1, 29, 35]. Note that the NDS metric is a weighted average of mAP and other breakdown metrics (e.g., translation, scale, orientation, velocity, and attribute errors). In addition, to validate the robustness of **feature alignment**, we followed Ref. [12] to simulate misalignment that

LiDAR and camera projection metric errors. It is worth noting that Ref. [12] only added noise to the validation dataset, without the train and test datasets.

Implementations. We implement GraphBEV within the PyTorch [40], built upon the open-source BEVFusion [35] and OpenPCDet [56]. For the LiDAR branch, feature encoding is performed using SECOND [66] to obtain LiDAR BEV features, with voxel dimensions set to [0.075m, 0.075m, 0.2m] and point cloud ranges specified as [-54m, -54m, -5m, 54m, 54m, 3m] across the X, Y, and Z axes, respectively. The camera branch employs a Swin Transformer [33] as the backbone, integrating Heads of numbers 3, 6, 12, 24, and utilizing FPN [16] to fuse multi-scale feature maps. The resolution of input images is adjusted and cropped to 256×704. In the LSS [41] configuration, frustum ranges are set with X coordinates [-54m, 54m, 0.3m], Y coordinates: [-54m, 54m, 0.3m], Z coordinates [-10m, 10m, 20m], and depth range: [1m, 60m, 0.5m].

During training, we employ data augmentation for ten epochs, including random flips, rotations (within the range $[-\frac{\pi}{4}, \frac{\pi}{4}]$), translations (with std=0.5), and scaling in [0.9, 1.1] for LiDAR data enhancement. We use CBGS [80] to resample the training data. Additionally, we use random rotation in $[-5.4°,$ $5.4°]$ and random resizing in [0.38, 0.55] to augment the images. The Adam optimizer [20] is used with a one-cycle learning rate policy, setting the maximum learning rate to 0.001 and weight decay to 0.01. The batch size is 24, and training is conducted on 8 NVIDIA GeForce RTX 3090 24G GPUs. During inference, we remove Test Time Augmentation (TTA) data augmentation, and the batch size is set to 1 on an A100 GPU. All latency measurements are taken on the same workstation with an A100 GPU.

4.2   Comparisons with State-of-the-Art Methods

3D Object Detection. We first compare the performances of our GraphBEV and other SOTA methods on the nuScenes validation and test for 3D object detection task, as shown in Table 1. Our GraphBEV achieves the best performances on the validation set (70.1% in mAP and 72.9% in NDS), which consistently outperforms all single-modality and multi-modal fusion methods. Among the multi-model methods, PointPainting [58], PointAugmenting [59], and MVP [73], serving as point-level methods, are unable to avoid feature misalignment issues. On the other hand, GraphAlign [52], AutoAlignV2 [9], TransFusion [1], and DeepInteraction [71], as feature-level methods, address the misalignment between LiDAR and camera features. However, these methods do not include the camera-to-BEV process, thus they cannot resolve the BEVFusion's [29, 35] feature misalignment issue caused by depth estimation.

As shown in Table 1, our GraphBEV outperforms the baseline BEVFusion [35] by +1.6% in mAP and +1.5% in NDS. The open-source nuScenes dataset generally has minimal feature alignment issues but cannot entirely avoid them, as illustrated in Figure 1(a). Specifically, our GraphBEV utilizes the KD-Tree algorithm to search the neighbor depth around the projected depth provided

Table 1: Comparison with the SOTA methods on the nuScenes validation and test set. 'C.V.', 'Motor.', 'Ped.' and 'T.C.' are short for construction vehicles, motorcycles, pedestrians, and traffic cones. The Modality column: 'L' = only LiDAR data, 'L.C.' = using both LiDAR and camera data. † means using TTA (test-time augmentation). The best performances are marked with **bold** font.

Method	Modality	mAP	NDS	Car	Truck	C.V.	Bus	Trailer	Barrier	Motor.	Bike	Ped.	T.C.
Performances on validation set													
TransFusion-L [1]	L	65.1	70.1	86.5	59.6	25.4	74.4	42.2	74.1	72.1	56.0	86.6	74.1
FUTR3D [7]	LC	64.2	68.0	86.3	61.5	26.0	71.9	42.1	64.4	73.6	63.3	82.6	70.1
TransFusion [1]	LC	67.3	71.2	87.6	62.0	27.4	75.7	42.8	73.9	75.4	63.1	87.8	77.0
BEVFusion-PKU [29]	LC	67.9	71.0	88.6	65.0	28.1	75.4	41.4	72.2	76.7	65.8	88.7	76.9
ObjectFusion [5]	LC	69.8	72.3	89.7	**65.6**	**32.0**	**77.7**	42.8	75.2	79.4	65.0	89.3	81.1
BEVFusion-MIT [35]	LC	68.5	71.4	89.2	64.6	30.4	75.4	42.5	72.0	78.5	65.3	88.2	79.5
GraphBEV(Ours)	LC	**70.1**	**72.9**	**89.9**	64.7	31.1	76.0	**43.8**	**76.0**	**80.1**	**67.5**	**89.2**	**82.2**
		+1.6	*+1.5*						*+4.0*		*+2.2*	*+2.7*	
Performances on test set													
PointPillar [21]	L	40.1	55.0	76.0	31.0	11.3	32.1	36.6	56.4	34.2	14.0	64.0	45.6
CenterPoint [72]†	L	60.3	67.3	85.2	53.5	20.0	63.6	56.0	71.1	59.5	30.7	84.6	78.4
PointPainting [58]	LC	46.4	58.1	77.9	35.8	15.8	36.2	37.3	60.2	41.5	24.1	73.3	62.4
PointAugmenting [59]†	LC	66.8	71.0	87.5	57.3	28.0	65.2	60.7	72.6	74.3	50.9	87.9	83.6
MVP [73]	LC	66.4	70.5	86.8	58.5	26.1	67.4	57.3	74.8	70.0	49.3	89.1	85.0
GraphAlign [52]	LC	66.5	70.6	87.6	57.7	26.1	66.2	57.8	74.1	72.5	49.0	87.2	86.3
AutoAlignV2 [9]	LC	68.4	72.4	87.0	59.0	33.1	69.3	59.3	-	72.9	52.1	87.6	-
TransFusion [1]	LC	68.9	71.7	87.1	60.0	33.1	68.3	60.8	78.1	73.6	52.9	88.4	86.7
DeepInteraction [71]	LC	70.8	73.4	87.9	60.2	37.5	70.8	63.8	80.4	75.4	**54.5**	90.3	87.0
BEVFusion-PKU [29]	LC	69.2	71.8	88.1	**60.9**	34.4	69.3	62.1	78.2	72.2	52.2	89.2	85.2
ObjectFusion [5]	LC	71.0	73.3	**89.4**	59.0	40.5	71.8	63.1	76.6	**78.1**	53.2	90.7	87.7
BEVFusion-MIT [35]	LC	70.2	72.9	88.6	60.1	39.3	69.8	63.8	80.0	74.1	51.0	89.2	86.5
GraphBEV(Ours)	LC	**71.7**	**73.6**	89.2	60.0	**40.8**	**72.1**	**64.5**	**80.1**	76.8	53.3	**90.9**	**88.9**
		+1.5	*+0.7*				*+2.3*				*+2.7*	*+2.3*	*+2.4*

by LiDAR-to-camera. Compared to BEVFusion [35], our GraphBEV demonstrates significant improvements in small objects, such as Barrier by +4.0%, Bike by +2.2%, and Pedestrian by +2.7%. It is primarily caused by small objects being more sensitive to feature misalignment, where even slight misalignment between LiDAR and the camera can result in greater misalignment in small objects. Furthermore, our GraphBEV and ObjectFusion tackle the issue of feature misalignment in BEV-based multi-modal methods [29,35]. While ObjectFusion [5] introduces a post-fusion paradigm utilizing RoI Pooling fusion, our GraphBEV does not alter the paradigm of BEVFusion [29,35], slightly outperforms ObjectFusion by 0.7% in mAP and 0.3% in NDS. Additionally, our GraphBEV achieved SOTA performance when we submitted the detection results for the nuScenes test set to the official evaluation server. It surpassed the baseline BEVFusion [35] by 1.5% in mAP and 0.7% in NDS. Overall, we could address feature misalignment issues in BEVFusion [29,35] without altering its excellent paradigm.

BEV map segmentation. To evaluate 3D object detection, we also assessed the generalization capability in the BEV Map Segmentation (semantic segmentation) task on the nuScenes validation set, as shown in Table 2. Following the

Table 2: Comparison with the SOTA methods on BEV map segmentation on nuScenes validation set. The Modality column: 'L' = only LiDAR data, 'LC' = using both LiDAR and camera data.

Method	Modality	Drivable	Ped. Cross.	Walkway	Stop Line	Carpark	Divider	Mean
PointPillars [21]	L	72.0	43.1	53.1	29.7	27.7	37.5	43.8
CenterPoint [72]	L	75.6	48.4	57.5	36.5	31.7	41.9	48.6
PointPainting [58]	LC	75.9	48.5	57.1	36.9	34.5	41.9	49.1
MVP [73]	LC	76.1	48.7	57.0	36.9	33.0	42.2	49.0
BEVFusion [35]	LC	85.5	60.5	67.6	52.0	57.0	**53.7**	62.7
GraphBEV(Ours)	LC	**86.3**	**60.9**	**69.1**	**53.1**	**57.5**	53.1	**63.3**

same training strategy as the baseline BEVFusion, we conducted evaluations within the [-50m, 50m]×[-50m, 50m] region around the ego car for each frame. We reported Intersection-over-Union (IoU) scores for drivable area, pedestrian crossing, walkway, stop line, car park, and divider. Significant improvements were observed for drivable area, pedestrian crossing, walkway, stop line, and car park, with only a minor decrease observed for the divider. Overall, our Graph-BEV demonstrates not only significant performance in 3D object detection but also strong generalization capability in BEV Map Segmentation.

4.3 Ablation Study

Roles of Different Modules in GraphBEV for Feature Alignment. To analyze the impact of misalignment, we conducted comparative experiments between our GraphBEV and BEVFusion [35]. It is noteworthy that in Table 3, we introduced misalignment into the nuScenes validation set, rather than in the training and testing sets, as follow Ref. [12]. We trained on the clean nuScenes [4] training dataset and evaluated performance under both clean and noisy misalignment conditions. In the clean setting, GraphBEV outperforms BEVFusion [35] significantly, while in the noisy setting, the performance improvement is substantial. Furthermore, it is evident that BEVFusion [35] exhibits a significant decrease in metrics such as mAP and NDS when transitioning from clean to noisy conditions, whereas GraphBEV demonstrates a smaller decrease in performance metrics.

Notably, adding the LocalAlign or GlobalAlign module to BEVFusion [35] has minimal impact on latency compared to BEVFusion [35] alone, and the latency is lower than that of TransFusion [1]. When only the LocalAlign module is added to BEVFusion [35] and the KD-Tree algorithm is used to build proximity relationships, while fusing projected depth with neighbor depth to prevent feature misalignment, significant enhancements are observed in both clean and noisy misalignment settings. Adding only the GlobalAlign module to BEVFusion [35] also leads to noticeable improvements. Particularly, the simultaneous addition of the LocalAlign and GlobalAlign modules exhibits strong performance in both clean and noisy settings.

Table 3: Roles of Different Modules in GraphBEV for Feature Alignment on nuScenes validation set under clean setting and noisy misalignment setting. 'C.V.', 'Motor.', 'Ped.' and 'T.C.' are short for construction vehicles, motorcycles, pedestrians, and traffic cones. '+L (only)' indicates the addition of only the LocalAlign module, and '+G (only)' indicates only the GlobalAlign module. GraphBEV denotes the addition of both LocalAlign and GlobalAlign modules. 'L.T. (ms)' represents latency. All latency measurements are conducted on the same workstation with an A100 GPU.

	Method	mAP	NDS	LT(ms)	Car	Truck	C.V.	Bus	Trailer	Barrier	Motor.	Bike	Ped.	T.C.
Clean	TransFusion [1]	67.3	71.2	164.6	87.6	62.0	27.4	75.7	42.8	73.9	75.4	63.1	87.8	77.0
	Baseline [35]	68.5	71.4	133.2	89.2	64.6	30.4	75.4	42.5	72.0	78.5	65.3	88.2	79.5
	+L (only)	69.7	72.4	136.3	89.5	64.4	30.6	75.9	43.5	75.6	79.6	67.1	88.8	82.3
		+1.2	*+1.0*	*+3.1*	*+0.3*	*-0.2*	*+0.2*	*+0.5*	*+1.0*	*+3.6*	*+1.1*	*+1.8*	*+0.6*	*+2.8*
	+G (only)	68.9	71.7	138.1	89.6	64.7	30.5	75.7	43.4	72.2	79.2	65.8	88.7	79.9
		+0.4	*+0.3*	*+4.9*	*+0.4*	*+0.1*	*+0.1*	*+0.3*	*+0.9*	*+0.2*	*+0.7*	*+0.5*	*+0.5*	*+0.4*
	GraphBEV	70.1	72.9	140.9	89.9	64.7	31.1	76.0	43.8	76.0	80.1	67.5	89.2	82.2
		+1.6	*+1.5*	*+7.7*	*+0.7*	*+0.1*	*+0.7*	*+0.6*	*+1.3*	*+4.0*	*+1.6*	*+2.2*	*+1.0*	*+2.7*
Noisy	TransFusion [1]	66.4	70.6	164.6	86.3	61.8	26.9	75.1	42.0	73.1	74.9	62.5	85.2	75.9
	Baseline [35]	60.8	65.7	132.9	83.1	50.3	26.5	66.4	38.0	65.0	64.9	52.8	86.1	75.1
	+L (only)	67.0	70.1	136.2	86.4	60.3	29.1	73.3	40.3	74.0	78.0	62.1	86.8	79.9
		+6.2	*+4.4*	*+3.3*	*+3.3*	*+10.0*	*+2.6*	*+6.9*	*+2.3*	*+9.0*	*+13.1*	*+9.3*	*+0.7*	*+4.8*
	+G (only)	63.1	67.2	137.9	84.2	51.7	27.8	68.6	39.5	68.8	68.7	57.2	86.2	77.8
		+2.3	*+1.5*	*+5.0*	*+1.1*	*+1.4*	*+1.3*	*+2.2*	*+1.5*	*+3.8*	*+3.8*	*+4.4*	*+0.1*	*+2.7*
	GraphBEV	69.1	72.0	141.0	88.1	63.5	30.0	75.1	42.7	75.3	79.8	64.9	88.9	82.2
		+8.3	*+6.3*	*+8.1*	*+5.0*	*+13.2*	*+3.5*	*+8.7*	*+4.7*	*+10.3*	*+14.9*	*+12.1*	*+2.8*	*+7.1*

Table 4: Effect of the Hyperparameters K_{graph} for Feature Misalignment. We analyze the effect of hyperparameter K_{graph} in LocalAlign module for feature alignment under noisy misalignment settings on the nuScenes validatio set. 'LT(ms)' represents latency. All latency measurements are conducted on the same workstation with an A100 GPU.

Baseline [35]			$K_{\mathrm{graph}}=5$			$K_{\mathrm{graph}}=8$			$K_{\mathrm{graph}}=12$			$K_{\mathrm{graph}}=16$			$K_{\mathrm{graph}}=25$		
mAP	NDS	LT	mAP	NDS	LT	mAP	NDS	LT	mAP	NDS	LT	mAP	NDS	LT	mAP	NDS	LT
60.8	65.7	132.9	67.1	70.9	138.2	69.1	72.0	141.0	69.8	72.2	143.4	68.8	70.5	145.3	67.1	69.9	150.0

Table 5: Robustness to weather conditions on nuScenes [4] clean validation set. Notably, the evaluation metric is mAP.

Method	Different Weather Conditions			
	Sunny	Rainy	Day	Night
Baseline [35]	68.2	69.9	68.5	42.8
GraphBEV	70.1	70.2	69.7	45.1

Table 6: Robustness to different ego distances, different sizes on nuScenes [4] clean validation set. Notably, the evaluation metric is mAP.

Method	Different Ego Distances			Different Object Sizes		
	Near	Middle	Far	Small	Moderate	Large
TransFusion-L [1]	77.5	60.9	34.8	44.7	54.5	60.4
Baseline [35]	79.4	64.9	40.0	50.3	58.7	64.0
GraphBEV	78.6	65.3	42.1	55.4	58.3	63.1

Effect of the Hyperparameters K_{graph} for Feature Misalignment. As shown in Table 4, to analyze the impact of the hyperparameter K_{graph} in the LocalAlign module on feature misalignment, we studied its effects under noisy misalignment settings on the nuScenes validation set. K_{graph}, which is the num-

ber of nearest depths for LiDAR-to-camera projected depth in the LocalAlign module, influences the expressive capability of neighboring depth features. It is observed that our GraphBEV achieves optimal overall performance when K_{graph} is set to 8. Therefore, selecting an appropriate value for K_{graph} is crucial and may vary across different datasets. Furthermore, despite significant fluctuations in mAP resulting from changes in K_{graph}, the overall performance still surpasses that of BEVFusion.

Robustness to Weather Conditions. As shown in Table 5, we present a robustness analysis of our GraphBEV with respect to different weather conditions. Various weather conditions influence the 3D object detection task. Following the approach of BEVFusion [35] and ObjectFusion [5], we partition the scenes in the validation set into sunny, rainy, day, and night conditions. We outperform BEV-Fusion [35] under different weather conditions, especially in night scenes. Overall, our GraphBEV improves performance in sunny weather through accurate feature alignment and enhances performance in adverse weather conditions.

Robustness to Ego Distances and Object Sizes. As shown in Table 6, we analyzed the impact of different ego distances and object sizes on the performance of GraphBEV. We categorized annotation and prediction ego distances into three groups: Near (0-20m), Middle (20-30m), and Far (>30m), and summarized the size distributions for each category, defining three equal-proportion size levels: Small, Moderate, and Large. It is evident that GraphBEV demonstrates significant performance improvements for distant and small objects. Compared to BEVFusion [35], our GraphBEV consistently enhances performance across all ego distances and object sizes, further narrowing the performance gaps. Overall, our GraphBEV exhibits greater robustness to changes in ego distances and object sizes.

5   Conclusion

In this work, we propose a robust fusion framework, GraphBEV, to address the feature misalignment issue for BEV-based methods. To mitigate feature misalignment caused by inaccurately projected depth provided by LiDAR, we propose the LocalAlign module, which utilizes neighbor-aware depth features through graph matching. Furthermore, to prevent global misalignment during the fusion of LiDAR and camera BEV features, we designed the GlobalAlign module to simulate offset noise, followed by aligning global multi-modal features through learnable offsets. Our GraphBEV significantly outperforms BEV-Fusion on the nuScenes validation set, especially in noisy misalignment settings. It demonstrates that our GraphBEV greatly advances the application of BEV-Fusion in real-world scenarios, particularly under noisy misalignment conditions.

Acknowledgements

References

1. Bai, X., Hu, Z., Zhu, X., Huang, Q., Chen, Y., Fu, H., Tai, C.L.: Transfusion: Robust lidar-camera fusion for 3d object detection with transformers 2, 5, 9, 10, 11, 12, 13
2. Bi, J., Wei, H., Zhang, G., Yang, K., Song, Z.: Dyfusion: Cross-attention 3d object detection with dynamic fusion. IEEE Latin America Transactions **22**(2), 106–112 (2024) 2, 4
3. 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) 4
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) 2, 4, 8, 9, 12, 13
5. Cai, Q., Pan, Y., Yao, T., Ngo, C.W., Mei, T.: Objectfusion: Multi-modal 3d object detection with object-centric fusion. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 18067–18076 (October 2023) 2, 4, 11, 14
6. 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) 4
7. Chen, X., Zhang, T., Wang, Y., Wang, Y., Zhao, H.: Futr3d: A unified sensor fusion framework for 3d detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 172–181 (2023) 11
8. Chen, Y., Liu, J., Zhang, X., Qi, X., Jia, J.: Largekernel3d: Scaling up kernels in 3d sparse cnns. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13488–13498 (2023) 4
9. Chen, Z., Li, Z., Zhang, S., Fang, L., Jiang, Q., Zhao, F.: Autoalignv2: Deformable feature aggregation for dynamic multi-modal 3d object detection (Jul 2022) 2, 4, 10, 11
10. Chen, Z., Li, Z., Zhang, S., Fang, L., Jiang, Q., Zhao, F., Zhou, B., Zhao, H.: Autoalign: Pixel-instance feature aggregation for multi-modal 3d object detection. In: Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence (Jul 2022). https://doi.org/10.24963/ijcai.2022/116 2, 4
11. Deng, J., Shi, S., Li, P., Zhou, W., Zhang, Y., Li, H.: Voxel r-cnn: Towards high performance voxel-based 3d object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 1201–1209 (2021) 4
12. Dong, Y., Kang, C., Zhang, J., Zhu, Z., Wang, Y., Yang, X., Su, H., Wei, X., Zhu, J.: Benchmarking robustness of 3d object detection to common corruptions in autonomous driving (Mar 2023) 2, 3, 9, 10, 12

13. Fan, L., Pang, Z., Zhang, T., Wang, Y.X., Zhao, H., Wang, F., Wang, N., Zhang, Z.: Embracing single stride 3d object detector with sparse transformer. In: 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (Jun 2022). https://doi.org/10.1109/cvpr52688.2022.00827 4
14. Ge, C., Chen, J., Xie, E., Wang, Z., Hong, L., Lu, H., Li, Z., Luo, P.: Metabev: Solving sensor failures for bev detection and map segmentation. arXiv preprint arXiv:2304.09801 (2023) 2, 4
15. He, C., Li, R., Li, S., Zhang, L.: Voxel set transformer: A set-to-set approach to 3d object detection from point clouds 4
16. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the IEEE international conference on computer vision. pp. 2961–2969 (2017) 10
17. Huang, T., Liu, Z., Chen, X., Bai, X.: Epnet: Enhancing point features with image semantics for 3d object detection. In: European Conference on Computer Vision. pp. 35–52. Springer (2020) 2, 4
18. Jiang, Y., Zhang, L., Miao, Z., Zhu, X., Gao, J., Hu, W., Jiang, Y.G.: Polarformer: Multi-camera 3d object detection with polar transformer. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 1042–1050 (2023) 4
19. Kim, Y., Park, K., Kim, M., Kum, D., Choi, J.W.: 3d dual-fusion: Dual-domain dual-query camera-lidar fusion for 3d object detection. arXiv preprint arXiv:2211.13529 (2022) 2
20. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 10
21. 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) 4, 11, 12
22. Li, X., Ma, T., Hou, Y., Shi, B., Yang, Y., Liu, Y., Wu, X., Chen, Q., Li, Y., Qiao, Y., He, L.: Logonet: Towards accurate 3d object detection with local-to-global cross-modal fusion (Mar 2023) 2
23. Li, X., Shi, B., Hou, Y., Wu, X., Ma, T., Li, Y., He, L.: Homogeneous multi-modal feature fusion and interaction for 3d object detection. In: European Conference on Computer Vision. pp. 691–707. Springer (2022) 2
24. Li, Y., Yu, A., Meng, T., Caine, B., Ngiam, J., Peng, D., Shen, J., Wu, B., Lu, Y., Zhou, D., Le, Q., Yuille, A., Tan, M.: Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection 2
25. Li, Y., Bao, H., Ge, Z., Yang, J., Sun, J., Li, Z.: Bevstereo: Enhancing depth estimation in multi-view 3d object detection with temporal stereo. Proceedings of the AAAI Conference on Artificial Intelligence **37**(2), 1486–1494 (Jun 2023). https://doi.org/10.1609/aaai.v37i2.25234, https://ojs.aaai.org/index.php/AAAI/article/view/25234 4
26. Li, Y., Ge, Z., Yu, G., Yang, J., Wang, Z., Shi, Y., Sun, J., Li, Z.: Bevdepth: Acquisition of reliable depth for multi-view 3d object detection. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 1477–1485 (2023) 3, 4, 5, 6, 8
27. Li, Z., Wang, F., Wang, N.: Lidar r-cnn: An efficient and universal 3d object detector. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7546–7555 (2021) 4
28. Li, Z., Wang, W., Li, H., Xie, E., Sima, C., Lu, T., Qiao, Y., Dai, J.: Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In: European conference on computer vision. pp. 1–18. Springer (2022) 4

29. Liang, T., Xie, H., Yu, K., Xia, Z., Lin, Z., Wang, Y., Tang, T., Wang, B., Tang, Z.: Bevfusion: A simple and robust lidar-camera fusion framework. Advances in Neural Information Processing Systems **35**, 10421–10434 (2022) 1, 2, 3, 4, 5, 9, 10, 11

30. Liu, L., Song, Z., Xia, Q., Jia, F., Jia, C., Yang, L., Pan, H.: Sparsedet: A simple and effective framework for fully sparse lidar-based 3d object detection. arXiv preprint arXiv:2406.10907 (2024) 4

31. Liu, Y., Wang, T., Zhang, X., Sun, J.: Petr: Position embedding transformation for multi-view 3d object detection. In: European Conference on Computer Vision. pp. 531–548. Springer (2022) 4

32. Liu, Y., Yan, J., Jia, F., Li, S., Gao, A., Wang, T., Zhang, X.: Petrv2: A unified framework for 3d perception from multi-camera images. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3262–3272 (2023) 4

33. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 10012–10022 (2021) 5, 10

34. Liu, Z., Huang, T., Li, B., Chen, X., Wang, X., Bai, X.: Epnet++: Cascade bi-directional fusion for multi-modal 3d object detection. IEEE Transactions on Pattern Analysis and Machine Intelligence **45**(7), 8324–8341 (2023). https://doi.org/10.1109/TPAMI.2022.3228806 2, 4

35. Liu, Z., Tang, H., Amini, A., Yang, X., Mao, H., Rus, D.L., Han, S.: Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation pp. 2774–2781 (2023) 1, 2, 3, 4, 5, 6, 9, 10, 11, 12, 13, 14

36. Liu, Z., Tang, H., Lin, Y., Han, S.: Point-voxel cnn for efficient 3d deep learning. Advances in Neural Information Processing Systems **32** (2019) 4

37. Mao, J., Xue, Y., Niu, M., Bai, H., Feng, J., Liang, X., Xu, H., Xu, C.: Voxel transformer for 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3164–3173 (2021) 4

38. Miao, Z., Chen, J., Pan, H., Zhang, R., Liu, K., Hao, P., Zhu, J., Wang, Y., Zhan, X.: Pvgnet: A bottom-up one-stage 3d object detector with integrated multi-level features. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (Jun 2021). https://doi.org/10.1109/cvpr46437.2021.00329 4

39. Park, J., Xu, C., Yang, S., Keutzer, K., Kitani, K., Tomizuka, M., Zhan, W.: Time will tell: New outlooks and a baseline for temporal multi-view 3d object detection. arXiv preprint arXiv:2210.02443 (2022) 4

40. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019) 10

41. Philion, J., Fidler, S.: Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIV 16. pp. 194–210. Springer (2020) 3, 4, 5, 6, 8, 10

42. Qi, C.R., Liu, W., Wu, C., Su, H., Guibas, L.J.: Frustum pointnets for 3d object detection from rgb-d data. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 918–927 (2018) 4

43. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 652–660 (2017) 4

44. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017) 4

45. Shi, S., Jiang, L., Deng, J., Wang, Z., Guo, C., Shi, J., Wang, X., Li, H.: Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. International Journal of Computer Vision **131**(2), 531–551 (2023) 4

46. Shi, S., Wang, X., Li, H.: Pointrcnn: 3d object proposal generation and detection from point cloud. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 770–779 (2019) 4

47. Simonelli, A., Bulo, S.R., Porzi, L., López-Antequera, M., Kontschieder, P.: Disentangling monocular 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1991–1999 (2019) 4

48. Sindagi, V.A., Zhou, Y., Tuzel, O.: Mvx-net: Multimodal voxelnet for 3d object detection. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 7276–7282. IEEE (2019) 2, 4

49. Song, Z., Jia, C., Yang, L., Wei, H., Liu, L.: Graphalign++: An accurate feature alignment by graph matching for multi-modal 3d object detection. IEEE Transactions on Circuits and Systems for Video Technology (2023) 2, 4

50. Song, Z., Jia, F., Pan, H., Luo, Y., Jia, C., Zhang, G., Liu, L., Ji, Y., Yang, L., Wang, L.: Contrastalign: Toward robust bev feature alignment via contrastive learning for multi-modal 3d object detection. arXiv preprint arXiv:2405.16873 (2024) 4

51. Song, Z., Liu, L., Jia, F., Luo, Y., Zhang, G., Yang, L., Wang, L., Jia, C.: Robustness-aware 3d object detection in autonomous driving: A review and outlook. arXiv preprint arXiv:2401.06542 (2024) 1, 2, 4

52. Song, Z., Wei, H., Bai, L., Yang, L., Jia, C.: Graphalign: Enhancing accurate feature alignment by graph matching for multi-modal 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 3358–3369 (2023) 2, 4, 10, 11

53. Song, Z., Wei, H., Jia, C., Xia, Y., Li, X., Zhang, C.: Vp-net: Voxels as points for 3d object detection. IEEE Transactions on Geoscience and Remote Sensing (2023) 4

54. Song, Z., Zhang, G., Xie, J., Liu, L., Jia, C., Xu, S., Wang, Z.: Voxelnextfusion: A simple, unified, and effective voxel fusion framework for multimodal 3-d object detection. IEEE Transactions on Geoscience and Remote Sensing **61**, 1–12 (2023). https://doi.org/10.1109/TGRS.2023.3331893 2, 4

55. Song, Z., Zhang, G., Liu, L., Yang, L., Xu, S., Jia, C., Jia, F., Wang, L.: Robofusion: Towards robust multi-modal 3d object detection via sam. arXiv preprint arXiv:2401.03907 (2024) 4

56. Team, O.D.: Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet (2020) 10

57. 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) 4

58. 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) 2, 4, 10, 11, 12

59. Wang, C., Ma, C., Zhu, M., Yang, X.: Pointaugmenting: Cross-modal augmentation for 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11794–11803 (2021) 2, 4, 10, 11

60. Wang, L., Song, Z., Zhang, X., Wang, C., Zhang, G., Zhu, L., Li, J., Liu, H.: Sat-gcn: Self-attention graph convolutional network-based 3d object detection for autonomous driving. Knowledge-Based Systems **259**, 110080 (2023) 4

61. Wang, L., Zhang, X., Song, Z., Bi, J., Zhang, G., Wei, H., Tang, L., Yang, L., Li, J., Jia, C., et al.: Multi-modal 3d object detection in autonomous driving: A survey and taxonomy. IEEE Transactions on Intelligent Vehicles (2023) 1, 2

62. Wang, L., Zhang, X., Zhao, F., Wu, C., Wang, Y., Song, Z., Yang, L., Xu, B., Li, J., Ge, S.S.: Fuzzy-nms: Improving 3d object detection with fuzzy classification in nms. IEEE Transactions on Intelligent Vehicles (2024) 4

63. 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) 4

64. 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) 4

65. Xu, S., Li, F., Song, Z., Fang, J., Wang, S., Yang, Z.X.: Multi-sem fusion: multi-modal semantic fusion for 3d object detection. IEEE Transactions on Geoscience and Remote Sensing (2024) 4

66. Yan, Y., Mao, Y., Li, B.: Second: Sparsely embedded convolutional detection. Sensors **18**(10), 3337 (2018) 4, 5, 10

67. Yang, C., Chen, Y., Tian, H., Tao, C., Zhu, X., Zhang, Z., Huang, G., Li, H., Qiao, Y., Lu, L., et al.: Bevformer v2: Adapting modern image backbones to bird's-eye-view recognition via perspective supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 17830–17839 (2023) 4

68. Yang, L., Tang, T., Li, J., Chen, P., Yuan, K., Wang, L., Huang, Y., Zhang, X., Yu, K.: Bevheight++: Toward robust visual centric 3d object detection. arXiv preprint arXiv:2309.16179 (2023) 4

69. Yang, L., Yu, K., Tang, T., Li, J., Yuan, K., Wang, L., Zhang, X., Chen, P.: Bevheight: A robust framework for vision-based roadside 3d object detection. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21611–21620 (2023) 4

70. Yang, Z., Sun, Y., Liu, S., Shen, X., Jia, J.: Std: Sparse-to-dense 3d object detector for point cloud. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1951–1960 (2019) 4

71. Yang, Z., Chen, J., Miao, Z., Li, W., Zhu, X., Zhang, L.: Deepinteraction: 3d object detection via modality interaction (Aug 2022) 2, 10, 11

72. Yin, T., Zhou, X., Krahenbuhl, P.: Center-based 3d object detection and tracking. In: 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (Jun 2021). https://doi.org/10.1109/cvpr46437.2021.01161 11, 12

73. Yin, T., Zhou, X., Krähenbühl, P.: Multimodal virtual point 3d detection. Advances in Neural Information Processing Systems **34**, 16494–16507 (2021) 2, 4, 10, 11, 12

74. Yoo, J.H., Kim, Y., Kim, J., Choi, J.W.: 3d-cvf: Generating joint camera and lidar features using cross-view spatial feature fusion for 3d object detection. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVII 16. pp. 720–736. Springer (2020) 2

75. Yu, K., Tao, T., Xie, H., Lin, Z., Wu, Z., Xia, Z., Liang, T., Sun, H., Deng, J., Hao, D., Wang, Y., Liang, X., Wang, B.: Benchmarking the robustness of lidar-camera fusion for 3d object detection 2

76. Zhang, C., Wang, H., Cai, Y., Chen, L., Li, Y., Sotelo, M.A., Li, Z.: Robust-fusionnet: Deep multimodal sensor fusion for 3-d object detection under severe weather conditions. IEEE Transactions on Instrumentation and Measurement **71**, 1–13 (2022) 2

77. Zhang, G., Xie, J., Liu, L., Wang, Z., Yang, K., Song, Z.: Urformer: Unified representation lidar-camera 3d object detection with transformer. In: Chinese Conference on Pattern Recognition and Computer Vision (PRCV). pp. 401–413. Springer (2023) 4

78. Zhang, Y., Chen, J., Huang, D.: Cat-det: Contrastively augmented transformer for multi-modal 3d object detection 2

79. 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) 4

80. Zhu, B., Jiang, Z., Zhou, X., Li, Z., Yu, G.: Class-balanced grouping and sampling for point cloud 3d object detection. arXiv preprint arXiv:1908.09492 (2019) 10

81. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020) 4