

Look Before You Fuse: 2D-Guided Cross-Modal Alignment for Robust 3D Detection

Xiang Li^{1,2} Zhangchi Hu² Xu Xiao^{1,2} Bin Kong^{1†}

¹Institute of Intelligent Machines, Hefei Institutes of Physical Science, Chinese Academy of Sciences

²University of Science and Technology of China

{xiangli, HuZhangchi, xiao.xu}@mail.ustc.edu.cn, bkong@iim.ac.cn

Supplementary

A Details about misalignment

The fusion of data from multiple sensor modalities, particularly LiDAR and camera, is a cornerstone of modern 3D perception systems for autonomous vehicles. Datasets such as nuScenes provide a rich source of multi-modal data to train and validate these systems. However, a critical challenge that persists is the inherent spatial misalignment between the sensor streams. When projecting LiDAR point clouds onto corresponding camera images, a noticeable discrepancy often appears. This artifact is not uniform across the image; instead, it is most pronounced at the boundaries between foreground objects and the distant background. A canonical example is the projection of LiDAR points onto a vehicle, where points corresponding to a far-away background wall erroneously “bleed” onto the pixels of the much closer vehicle, creating a fringe of incorrectly projected points along the object’s silhouette. This document provides a detailed analysis of the primary causes of this phenomenon within the context of the nuScenes dataset.

The core of multi-modal fusion lies in the geometric projection of a 3D point from the LiDAR’s coordinate frame (L) into the 2D pixel coordinate frame of the camera (C). In an ideal system, this is a deterministic transformation. For the nuScenes dataset, this involves transforming a point $P_L \in \mathbb{R}^3$ from the LiDAR frame to the ego-vehicle frame (E), and then from the ego-vehicle frame to the camera frame. The extrinsic transformation from the LiDAR to the ego-vehicle is represented by a rigid body transformation matrix $T_{E \leftarrow L} \in SE(3)$, and similarly from the camera to the ego-vehicle, $T_{E \leftarrow C}$. Therefore, the transformation from LiDAR to camera is given by $T_{C \leftarrow L} = (T_{E \leftarrow C})^{-1} \cdot T_{E \leftarrow L}$. This matrix can be decomposed into a rotation matrix $R_{C \leftarrow L}$ and a translation vector $t_{C \leftarrow L}$. A point P_L is projected into the camera’s image plane to pixel coordinates $p \in \mathbb{R}^2$ by first converting to homogeneous

coordinates $\tilde{P}_L = [P_L; 1]$ and applying the full projection model:

$$z \begin{bmatrix} p \\ 1 \end{bmatrix} = K [R_{C \leftarrow L} | t_{C \leftarrow L}] \tilde{P}_L \quad (1)$$

where $K \in \mathbb{R}^{3 \times 3}$ is the camera’s intrinsic calibration matrix and z is the point’s depth in the camera frame. This equation represents the ideal geometric relationship, but in practice, several sources of error corrupt this process.

The first major source of error is static and stems from imperfect extrinsic calibration. The matrix $T_{C \leftarrow L}$ is estimated through a calibration procedure that, while precise, inevitably contains small, residual errors. Let the estimated, erroneous rotation matrix be $\hat{R} = R_{C \leftarrow L} \cdot \Delta R$, where ΔR represents a small angular error. The effect of this rotational error on the final projection is amplified by the depth of the 3D point. For a small angular error $\Delta\theta$ about an axis, the resulting displacement Δx in the projection plane at a depth d can be approximated by:

$$\Delta x \approx d \cdot \tan(\Delta\theta) \quad (2)$$

This relationship is fundamental to understanding the boundary-specific nature of the misalignment. A point on a background wall at a depth $d_{bg} = 60\text{m}$ will experience a projection shift four times larger than a point on a foreground vehicle at $d_{fg} = 15\text{m}$ for the exact same calibration error $\Delta\theta$. This creates a significant differential shift between the projected locations of foreground and background points, which becomes visually apparent precisely at the depth discontinuities that define object boundaries.

The second category of errors is dynamic, arising from motion during the data acquisition process. The Velodyne HDL-32E LiDAR used in nuScenes is a mechanical spinning sensor operating at 20 Hz, meaning a full 360-degree scan requires 50 ms to complete. A point cloud is therefore not an instantaneous snapshot of the world; each point is captured at a slightly different time and from a slightly different ego-vehicle pose. This is often referred to as a “rolling shutter” effect. Although nuScenes provides motion compensation to correct for this ego-motion distortion

[†] Corresponding author.

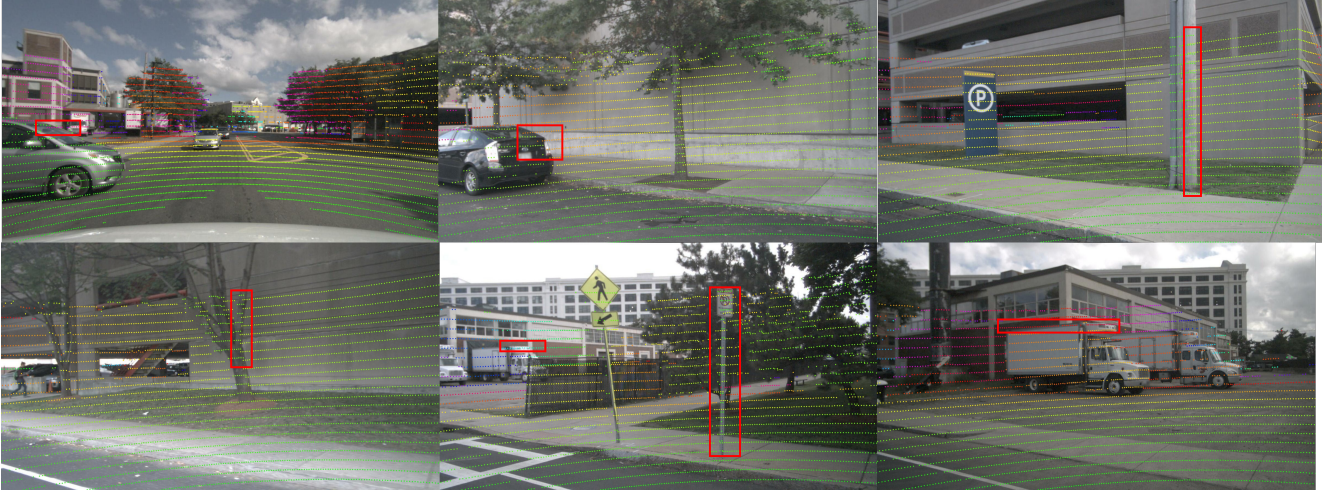


Figure 1. Spatial Misalignment (Extrinsic Calibration Error) and Temporal Misalignment (Synchronization Error) cases in nusenes dataset

by transforming all points into the coordinate system of the final timestamp, this compensation often assumes linear motion of the ego-vehicle. During complex maneuvers involving non-linear acceleration or yaw rates, residual distortions remain in the point cloud, causing static objects like straight walls to appear warped.

Furthermore, a more subtle but systematic dynamic error in nuScenes arises from the sensor synchronization and timestamping policy. The camera exposure is triggered when the LiDAR’s top laser sweeps across the center of the camera’s field of view. This moment defines the image timestamp, t_{cam} . However, the corresponding LiDAR scan is assigned a single timestamp, t_{lidar} , which corresponds to the time when the full 360-degree rotation is completed. This creates a systematic temporal offset $\Delta t = t_{\text{lidar}} - t_{\text{cam}}$, where the LiDAR data is, on average, approximately 25 ms older than the camera image it is paired with. For static scenes, this offset is negligible. For any object in motion with velocity v_{obj} , however, this leads to a physical displacement error in its perceived position:

$$\Delta s = v_{\text{obj}} \cdot \Delta t \quad (3)$$

For a vehicle moving at 50 km/h (13.9 m/s), a 25 ms offset results in a spatial error of approximately 35 cm, a significant discrepancy that further contributes to the misalignment between the LiDAR projection and the camera image.

In conclusion, the visually striking misalignment at object-background boundaries in datasets like nuScenes is not a result of a single flaw, but rather the confluence of static, dynamic, and temporal errors. The depth-dependent amplification of minute calibration errors creates a foundational differential shift between near and far objects. This is compounded by residual distortions from imperfect ego-motion compensation and systematic spatial offsets for

moving objects due to timestamp asynchrony. The reason these artifacts become so prominent at object boundaries is because these boundaries represent a sharp discontinuity in depth and motion state. At these locations, the a small error vector associated with the foreground object’s projection meets the large, dissimilar error vector of the background, making the misalignment manifest as a visible “bleeding” of points. These effects are visually summarized in Fig. 1. Recognizing that these misalignments are an inherent and predictable artifact of the data collection process is crucial for developing robust perception algorithms that can account for, and be resilient to, such real-world data imperfections.

B Hyperparameter Ablation Study

All performance results, including the primary results in the main text, are averaged over 3 separate runs. To validate our design choices, we conducted additional ablation studies on key hyperparameters within our proposed modules. The results, shown in Tab. 2, demonstrate the robustness of our framework and justify the parameter values used in our main experiments.

DAGF Block Size. We evaluated the block size used for depth and gradient densification. A smaller block size of 10×10 is more sensitive to local noise, slightly degrading performance. A larger block size of 40×40 over-smooths the geometric details, also leading to a performance drop. The chosen 20×20 block size provides the best balance between capturing geometric structure and robustness to noise.

Depth Align Module Neighbors. We tested the number of neighbors (K_s) used in the Structural Depth Smoothing algorithm. While using 5 neighbors already yields good results, increasing the count to 10 further improves perfor-

Table 1. Efficiency and performance comparison. Our primary results (71.5 mAP) are based on the bevfusion-pku backbone. When compared to methods on the same **bevfusion-mit** baseline, our approach is notably faster than GraphBEV with higher mAP and same NDS.

Method	mAP (%)	NDS (%)	Latency (ms)
bevfusion-pku (baseline)	67.9	71.0	270
Ours (bevfusion-pku based)	71.5	73.6	285
bevfusion-mit (baseline)	68.5	71.4	120
GraphBEV (bevfusion-mit based)	70.1	72.9	142
Ours (bevfusion-mit based)	71.1	72.9	137

Table 2. Ablation studies for key hyperparameters. The default configuration used in our main experiments is highlighted in bold.

Module	Hyperparameter	mAP (%)	NDS (%)
DAGF	Block Size: 10×10	71.2	73.3
	Block Size: 20×20	71.5	73.6
	Block Size: 40×40	71.0	73.2
DAM	Neighbors (K_s): 5	71.1	73.4
	Neighbors (K_s): 10	71.5	73.6
	Neighbors (K_s): 20	71.4	73.5
FEM	α_k : None (disabled)	71.0	72.9
	α_k : Shared ($\alpha = 1.1$)	71.2	73.2
	α_k : Class-Specific	71.5	73.6

Table 3. Impact of different 2D prior qualities on module configurations. We test “No Priors”, misleading “Random Priors”, and a “Full-Image Prior” (a box covering the entire image). Note that the Depth Align Module shows unique resilience to both random and full-image priors, while FEM is sensitive to any incorrect prior.

Model	2D Prior Source	mAP (%)	NDS (%)
DAM	No Priors	69.0	71.6
	Random Priors	69.1	71.8
	Full-Image Prior	69.9	72.0
Full Model	No Priors	69.0	71.6
	Random Priors	68.5	71.2
	Full-Image Prior	69.4	71.8
FEM Only	No Priors	69.0	71.6
	Random Priors	67.4	70.1
	Full-Image Prior	68.6	71.3

mance by capturing a more stable local depth distribution. Increasing to 20 neighbors did not provide significant additional gains and slightly increased computational cost, justifying our choice of $K_s = 10$.

FEM Feature Enhancement. We analyzed the impact of the feature enhancement factor α_k . Disabling enhance-

ment ($\alpha_k = \text{None}$) leads to a significant performance drop, showing the value of boosting features in critical areas. Using a single, shared factor for all classes improves results, but employing class-specific factors (e.g., $\alpha_k = 1.4$ for large objects and $\alpha_k = 1.8$ for small objects) allows the model to adapt more effectively to the unique characteristics of different object categories, yielding the best performance.

Depth Align Module Smoothing Method. To validate the effectiveness of our proposed structural smoothing method over simpler alternatives, we conducted an ablation study within our full model configuration. We replaced our method with a baseline that performs simple averaging of the depth values of the K_s nearest neighbors. As shown in Tab. 4, our structural smoothing, which intelligently selects and weights neighbors to preserve depth discontinuities, yields a clear improvement in both mAP and NDS over the simpler method. This confirms that the nuanced approach is critical for achieving the best performance by correcting misalignment without erasing important geometric features at object boundaries.

Table 4. Ablation on the depth smoothing method. The comparison is performed by swapping the smoothing component within the full model configuration. Our proposed method shows a clear advantage over a simple averaging baseline.

Smoothing Method	mAP (%)	NDS (%)
Simple Averaging	70.8	73.0
Structural Smoothing (Ours)	71.5	73.6

C Efficiency Analysis

We analyzed the computational cost and performance of our method on different backbones. Tab. 1 summarizes the mAP, NDS, and inference latency, comparing our approach with the corresponding baselines and other methods like GraphBEV.

The results show our method’s effectiveness. On the **bevfusion-pku** backbone, we significantly improve mAP

(from 67.9 to 71.5) and NDS (from 71.0 to 73.6) with a modest 15 ms overhead. On the **bevfusion-mit** backbone, our method is 5 ms (142 ms vs 137 ms) faster than Graph-BEV while operating on the same baseline.

D Detailed Analysis of 2D Detector Impact

Our investigation into the influence of 2D detector quality revealed distinct effects on our proposed modules, particularly the Depth Align Module (DAM) and the Feature Enhancement Module (FEM), especially under conditions of extremely poor 2D priors.

For the **Depth Align Module**, we observed a surprising degree of resilience to poor-quality priors. As detailed in Tab. 3, the module’s performance improves over the baseline even when guided by flawed information. Counter-intuitively, providing completely **Random Priors** still results in a performance gain (+0.1% mAP), and using a **Full-Image Prior** yields an even more significant boost (+0.9% mAP). This suggests that the Depth Align Module can extract a beneficial signal from almost any spatial guidance, leveraging the general depth distribution within the indicated region—however coarse or inaccurate—to aid the 3D perception task.

In contrast, the **Feature Enhancement Module (FEM)** exhibited greater sensitivity to the quality of 2D detections. When supplied with incorrect bounding boxes, the FEM’s performance could degrade below the baseline. This occurs because the FEM is designed to intensify features within specific, object-centric regions. An incorrect prior causes it to amplify irrelevant or misleading background features while neglecting the actual object, thereby introducing noise that can confuse the final detection head. The quantitative results of this analysis are presented in Tab. 3.

E Implementation

The code for our proposed method can be fully reproduced following the descriptions in the main paper’s Method section. Our implementation is built upon the official **bevfusion-pku** baseline. It is important to note that the specific code handling the projection is provided in our appendix file. And the code of **Structural Guidance Depth Modulator (SGDM)** is provided in appendix file.