# *CoIn3D*: Revisiting Configuration-Invariant Multi-Camera 3D Object Detection

Supplementary Material

## 1. Dataset Details

The camera configuration details of Nuscene [1], Lyft [4], and Waymo [10] are shown in Tab. 1. The three dataset have distinct discrepancies in camera configuration, including intrinsics, extrinsics, and array layouts. Given camera intrinsics $(f_u, f_v, c_u, c_v)$ and image size $(w, h)$, the horizontal filed-of-view(FoV) $FoV_{hori}$ and vertical FoV $FoV_{vert}$ are calculated as follows:

$$\begin{aligned} FoV_{hori} &= 2\arctan(\frac{c_u}{f_u}) \\ FoV_{vert} &= 2\arctan(\frac{h-c_v}{f_v}) \end{aligned} \tag{1}$$

To be noted that, for the three datasets, the ego origin are located at the center of the rear axle projected to the ground.

## 2. Implementation Details

We show more implementation details about prior maps, data augmentation, model training, and ego-centric Gaussians construction in this section.

### 2.1. Prior Maps

For the prior maps, we normalize them by a constant factor to ensure numerical stability. Specifically, focal length, ground depth, and ground gradient are divided by 500, 25, and 2 respectively.

### 2.2. Data augmentation

For the data augmentation, denote the raw focal length, camera translation, and camera rotation as $f, t_x, t_y, t_z, r_x, r_y, r_z$, the augmentation policy for each item are $f \sim \mathcal{U}(f*0.7, f*1.4)$, $t_x \sim \mathcal{U}(t_x-0.2, t_x+0.2)$, $t_y \sim \mathcal{U}(t_y-0.2, t_y+0.2)$, $t_z \sim \mathcal{U} \in (1.5, 2.2)$, $r_x \sim \mathcal{U} \in (-2, 2)$, $r_y \sim \mathcal{U} \in (-2, 2)$, $r_z \sim \mathcal{U} \in (r_z-20, r_z+20)$. We randomly select raw image or ego-centric Gaussians for training with probability 0.5.

### 2.3. Model Training

For Waymo, consider the high annotation rate ($10Hz$ vs. $2Hz$ for Nuscenes) , we select each $4^{th}$ frame in all sequences for training. We will not downsample data during testing. During training and testing, we apply timestamp alignment for each camera in Nuscenes and Lyft follow by Waymo [10]. All experiments on main text are trained for 24 epochs.

### 2.4. Ego-centric Gaussian Construction

The detailed ego-centric Gaussian construction pipeline is shown in Fig. 1.

Specifically, the training-free 3DGS can divided into four steps as shown. First, for each data sequence, we use 4D annotations to decompose raw LiDAR sequences into background LiDAR sequences and object LiDAR sequences. Specifically, for each object LiDAR slice, we apply mirror flip to compensate the occluded part. Next, we use TSDF integration [11] to reconstruct background mesh and object meshes from background LiDAR sequences and object LiDAR sequences. Moreover, we will fix the object meshes to watertight surface.

Second, for each annotated timestamp in the sequence, we first transform objects meshes from local system to global system according to the annotation. We then compose background mesh and objects meshes. Next, we use camera intrinsic and extrinsic to render the composed meshes to multi-view depth [9]. These mesh-rendered depths have advantage that its metric is naturally precise and it can ensure multi-view consistent. These merits are well-suit for our data augmentation. Finally, considerate that some hole will existed in the mesh depths, we apply depth completion [12] to get dense depths.

Third, we reconstruct some auxiliary assets including texture objects points model and under-camera points model. These assets can help complete the unseen part beyond raw camera imaging. For example, the unseen under-camera area will affect the novel-view synthesis when we adjust the camera mounting height lower, the unseen object area will affect the NVS when we adjust the camera mounting height higher. To handle this, we first down-sample points from object meshes to get object points model, and we will sample points under camera blind area which can be determined by the initial ground depth mentioned at Sec.3.2 . After constructing points model, we apply texture retrieval for each point by warping them to different images and depths of the sequences. Once the projected depth and the corresponding image depth match under a error threshold, we will allocate the corresponding color to the point. Finally, we can get the textured points model for next step.

Fourth, we construct the ego-view Gaussians for each data frame at each annotated timestamp. First, we use camera intrinsic and extrinsic to project RGB-D images to ego space to get $(r, g, b, x, y, z)$ point cloud. We use Zits++ [2] to inpaint the foreground RGB, and we use ground and background depth to inpaint the foreground depth[5]. Then, we concatenate the textured object points and under-camera points to the scene point cloud. Finally, we set covari-

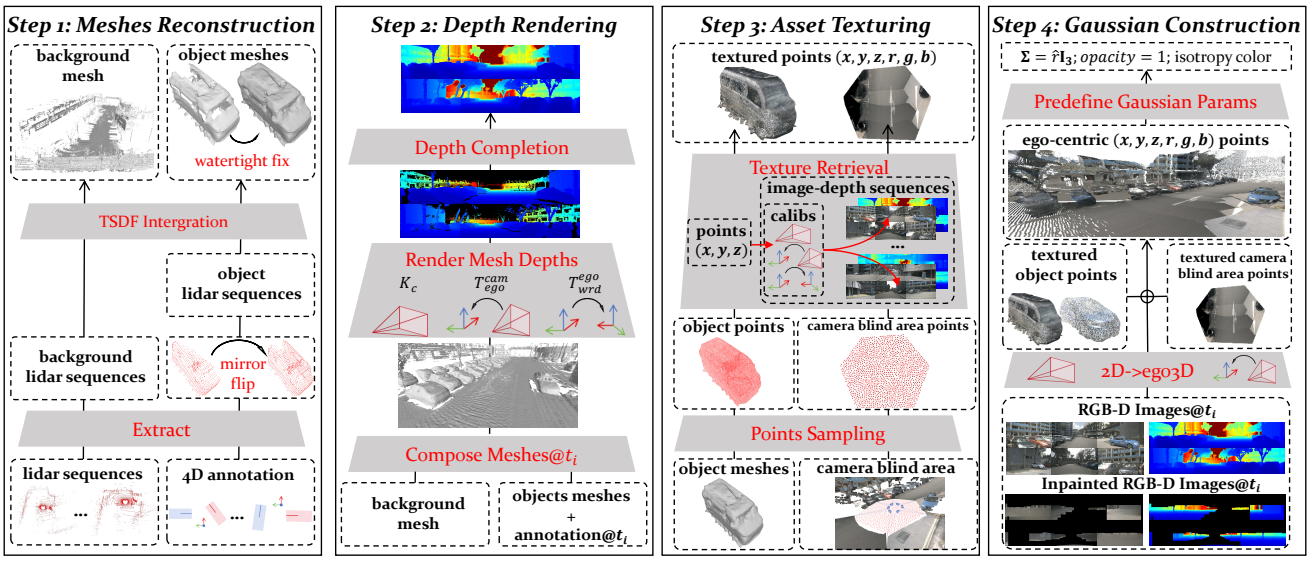


Figure 1. The detailed ego-centric Gaussian construction pipeline.

Table 1. The details of camera configurations of Nuscenes, Lyft, and Waymo. Camera flags F, FL, FR, B, BL, BR, SL, SR represent front, front left, front right, back, back left, back right, side left, and side right, respectively. Nuscenes and Lyft possess F, FL, FR, B, BL, and BR. Waymo possesses F, FL, FR, SL, and SR. Consider that camera installed pitch ($t_y$) and roll ($t_x$) angle is generally set to 0, we only show the yaw angle ($t_z$).

| Datasets | | Nuscenes | Lyft(fleet 1) | Lyft(fleet 2) | Waymo |
|---|---|---|---|---|---|
| image resolution$(w, h)$ | | (1600,900) | (1224,1024) | (1920,1080) | (1920,1280)(F,FL,FR)<br>(1920,886)(SL,SR) |
| focal length | | 1250(F,FL,FR,BL,BR)<br>800(B) | 880 | 1100 | 2050 |
| horizontal FoV (°) | | 65(F,FL,FR,BL,BR)<br>90(B) | 70 | 80 | 50 |
| vertical FoV (°) | | 40(F,FL,FR,BL,BR)<br>60(B) | 60 | 50 | 35 |
| cam numbers | | 6 | 6 | 6 | 5 |
| camera translation $(t_x, t_y, t_z)$ (m) | F | 1.7, 0, 1.5 | 1.5, 0, 1.7 | 1.5, 0, 1.65 | 1.55, 0, 2.1 |
| | FL | 1.55, 0.5, 1.5 | 1.3, 0.3, 1.7 | 1.3, 0.3, 1.65 | 1.5, 0.1, 2.1 |
| | FR | 1.55, -0.5, 1.5 | 1.3, -0.3, 1.7 | 1.3, -0.3, 1.65 | 1.5, -0.1, 2.1 |
| | B | 0, 0, 1.5 | 0.8, 0, 1.65 | 0.8, 0, 1.65 | |
| | BL/SL | 1.0, 0.5, 1.55 | 1.0, 0.3, 1.65 | 1.0, 0.3, 1.65 | 1.4, 0.1, 2.1 |
| | BR/SR | 1.0, -0.5, 1.55 | 1.0, -0.3, 1.65 | 1.0, -0.3, 1.65 | 1.4, -0.1, 2.1 |
| cam rotation $(r_z)$ (°) | F | 0 | 0 | 0 | 0 |
| | FL | 55 | 60 | 60 | 45 |
| | FR | -55 | -60 | -60 | -45 |
| | B | 180 | 180 | 180 | |
| | BL/SL | 110 | 120 | 120 | 90 |
| | BR/SR | -110 | -120 | -120 | -90 |

ance matrix of gaussian as $\mathbf{\Sigma} = \hat{r}\mathbf{I}_3$, which means that each Gaussians have no rotation and have a fixed predefined radius. The opacity of each gaussian is set to 1. We set isotropy color for each Gaussians which means that we

Table 2. Results of Nuscenes+Waymo+Lyft joint training (Mix), reported on 3 classes(car, pedestrian, two-wheel object). ScaleBEV [8] are reported on BEVDet, others are reported on BEVDepth. Oracle means training on the single source dataset.

| Setting | Method | All | | Car | | Pedestrian | | Two-wheel | |
|---|---|---|---|---|---|---|---|---|---|
| | | mAP | NDS* | mAP | NDS* | mAP | NDS* | mAP | NDS* |
| Mix → Nuscenes | Oracle | 0.255 | 0.335 | 0.385 | 0.508 | 0.205 | 0.247 | 0.175 | 0.249 |
| | Direct Mix | 0.267 | 0.350 | 0.411 | 0.537 | 0.213 | 0.258 | 0.176 | 0.255 |
| | ScaleBEV | 0.293 | 0.358 | 0.447 | 0.540 | 0.246 | 0.275 | 0.185 | 0.258 |
| | **CoIn3D** | **0.358** | **0.441** | **0.542** | **0.693** | **0.321** | **0.360** | **0.211** | **0.270** |
| Mix → Lyft | Oracle | 0.290 | 0.390 | 0.408 | 0.530 | 0.114 | 0.199 | 0.349 | 0.440 |
| | Direct Mix | 0.312 | 0.406 | 0.473 | 0.581 | 0.126 | 0.201 | 0.338 | 0.436 |
| | ScaleBEV | 0.333 | 0.422 | 0.505 | 0.603 | 0.140 | 0.212 | 0.355 | 0.453 |
| | **CoIn3D** | **0.455** | **0.532** | **0.644** | **0.718** | **0.236** | **0.323** | **0.486** | **0.556** |
| Mix → Waymo | Oracle | 0.298 | 0.376 | 0.426 | 0.536 | 0.334 | 0.328 | 0.135 | 0.265 |
| | Direct Mix | 0.306 | 0.373 | 0.432 | 0.536 | 0.342 | 0.332 | 0.144 | 0.250 |
| | ScaleBEV | 0.326 | 0.384 | 0.466 | 0.549 | 0.355 | 0.341 | 0.157 | 0.261 |
| | **CoIn3D** | **0.381** | **0.463** | **0.559** | **0.647** | **0.392** | **0.363** | **0.192** | **0.381** |

Table 3. Comparison of CoIn3D and two input-adjustment methods: Cross-dataset sensor alignment (CDSA) [14] and Unidrive [7].

| Setting | mAP | **NDS*** |
|---|---|---|
| CDSA N → L | 0.224 | 0.372 |
| UniDrive N → L | 0.264 | 0.440 |
| **CoIn3D N → L** | **0.375** | **0.534** |
| CDSA N → W | 0.116 | 0.259 |
| UniDrive N → W | 0.144 | 0.298 |
| **CoIn3D N → W** | **0.384** | **0.513** |

Table 4. Results of Nuscenes-val re-rendered in Waymo Cfg.

| Setting | mAP | **NDS*** |
|---|---|---|
| Oracle | 0.436 | 0.554 |
| DT | 0.126 | 0.342 |
| **CoIn3D** | **0.401** | **0.524** |

Table 5. Results of sequences with significant road slope in W-val (W$'$). Results are reported on BEVDepth with CoIn3D.

| Setting | mAP | **NDS*** |
|---|---|---|
| N → W$'$ | 0.516 | 0.602 |
| W → W$'$ | 0.638 | 0.702 |

use the raw $(r, g, b)$ color, the center of Gaussians are determined by the $(x, y, z)$ of point clouds. In other words, our scheme can be seen as a point-rendering scheme but we transform point cloud into Gaussians representation to take advantage of the fast rendering speed of 3DGS.

For the Gaussians radius $\hat{r}$, we set foreground objects Gaussians radius as 0.0025. As for the background, consider that small radius for ground Gaussians will cause hole in image after novel view synthesis, we linearly decrease the $\hat{r}$ from ground plane to sky according to their $z$. Specifically, $\hat{r}$ of background Gaussians with $z$ at range from $(0m, 10m)$ will be linearly mapped to range from $(0.02, 0.001)$.

# 3. More Experiment Results

## 3.1. Multi-Dataset Joint Training

To evaluate the scaling-up ability of CoIn3D, we conduct joint training using Nuscenes, Waymo, and Lyft. Tab. 2 shows that, simply mixing multiple datasets for joint training cannot scale up the performance, because there is a large camera configuration gap between different datasets. CoIn3D can bring significant improvement by bridging the configuration gap.

## 3.2. More Baseline Comparisions

Tab. 3 shows a comparison of two input-adjustment methods: Cross-dataset sensor alignment [14] and UniDrive [7]. The results show that they struggle to handle large configuration gaps, especially in N → W, because warping inputs destroys the scene geometry structure.

## 3.3. Domain-controlled Cfg Generalization

To control the dataset domain and strengthen the claim (configuration gap) of CoIn3D. We re-render Nuscenes into Nuscenes configuration (N-in-N) and Waymo configuration (N-in-W) using our training-free 3D Gaussians construction

Table 6. Source dataset performance comparison.

| Dataset | Method | mAP↑ | NDS*↑ |
|---|---|---|---|
| Nuscenes | BEVDepth [6] | 0.475 | 0.587 |
| | UDGA-BEV [3] | 0.497 | 0.603 |
| | Ours | **0.529** | **0.630** |
| Lyft | BEVDepth | 0.602 | 0.684 |
| | UDGA-BEV | **0.630** | **0.702** |
| | Ours | 0.624 | 0.699 |
| Waymo | BEVDepth | 0.552 | 0.649 |
| | UDGA-BEV | **0.547** | **0.656** |
| | Ours | 0.538 | 0.630 |

pipeline. In this way, we can alleviate the semantic gap and other configuration-irrelevant gaps. We then use N-in-N and N-in-W for training, respectively, and evaluate on N-in-W view. Tab. 4 shows that CoIn3D almost bridges the NDS* gap to unseen W-Cfg oracle.

### 3.4. Ground Corner Cases Evaluation

To evaluate the robustness of CoIn3D on corner cases under ground plane assumption, we evaluate CoIn3D on 6 sequences with the largest slope (16-22m elevation motion, selected using the 3-$\sigma$ principle) in W-val. Tab. 5 shows that CoIn3D still works well in roads with significant slope. Specifically, CoIn3D trained on either N or W perform well on these scene, even better than the overall performance. This indicates that: (1) ground prior embedding is robust because we do not constrain the ground to fixed depth like UniDrive, but embed it in a learnable way; (2) Further per-sequence analysis reveals that sequences with poor performance are characterized by low light and poor object visibility, which affect both N $\rightarrow$ W$^{'}$ and W $\rightarrow$ W$^{'}$, and are configuration-irrelevant.

### 3.5. Source Performance

Tab. 6 shows the results on source dataset after applying our framework. Our framework improve the performance on Nuscenes and Lyft. Regarding to Waymo, the performance drop is mainly attributed to our downsampled training strategy.

### 3.6. More ablations

In this section, we show more ablation experiments regrading to priors map and embedding projector. These experiments are also conduct on BEVDepth by training on Nuscenes and testing on Waymo, but trained for 6 epochs.

Tab. 7 shown more experiments about the spatial priors, all experiments are based on CDA applied. Specifically, "FSDepth" denotes using a focal embedding proposed by [13], "SFM@(InvF)" denotes using our inverse focal map to modulate feature, "SFM@(InvF,GD)" and "SFM@(InvF,PR)" denote using ground depth and plucker raymap additionally based on focal-invariant feature obtained. "Cat@(ALL)" denotes only concatenate the prior maps without any modulation. "SFM@(ALL)" denotes our final framework.

Table 7. More ablation results regrading to priors.

| Method | mAP↑ | NDS*↑ |
|---|---|---|
| FSDepth [13] | 0.223 | 0.374 |
| SFM@(InvF) | 0.260 | 0.404 |
| SFM@(InvF,GD) | 0.274 | 0.425 |
| SFM@(InvF,PR) | 0.272 | 0.429 |
| Cat@(ALL) | 0.247 | 0.391 |
| **SFM@(ALL)** | **0.320** | **0.463** |

Table 8. More ablation results regrading to embedding projectors.

| Method | mAP↑ | NDS*↑ |
|---|---|---|
| conv1 | 0.282 | 0.430 |
| conv1+conv1 | 0.307 | 0.454 |
| conv3+conv3 | 0.297 | 0.448 |
| conv3 | **0.320** | **0.463** |

The experiment yielded several conclusions. First, using inverse focal map to modulate feature is a better solution compare to FSDepth [13] in MC3D. Second, based on focal-invariant feature obtained, solely apply ground depth map or Plücker raymap both benefit the generalization, in view of the former offer explicit ground prior and the latter holistically formulate camera configurations. Third, solely concatenate prior maps without any modulation shown insufficient effectiveness, although explicit prior information is provided, the model is hard to understand these prior without designed modulation mechanism.

Tab. 8 shows more experiments about the projector structure. Specifically, "conv K" and "conv K + conv K" denote using one $K \times K$ convolution and two consecutive $K \times K$ to project raw priors into spatial embeddings. Results show that, "conv3" is the better choice. Compare to "conv1", it provide a large receptive field to fuse the neighbor priors, compare to "conv3+conv3", the shallow structure ensures that it does not overfit.

## 4. Qualitatively Results

In this section, we show the qualitatively results about the reconstructed ego-centric textured point clouds and the rendered novel-view images. Moreover, we will also show the detection results with and without using our framework.

Fig. 2, Fig. 3, and Fig. 4 show the novel view training images of Nuscenes, Lyft, and Waymo render from the ego-

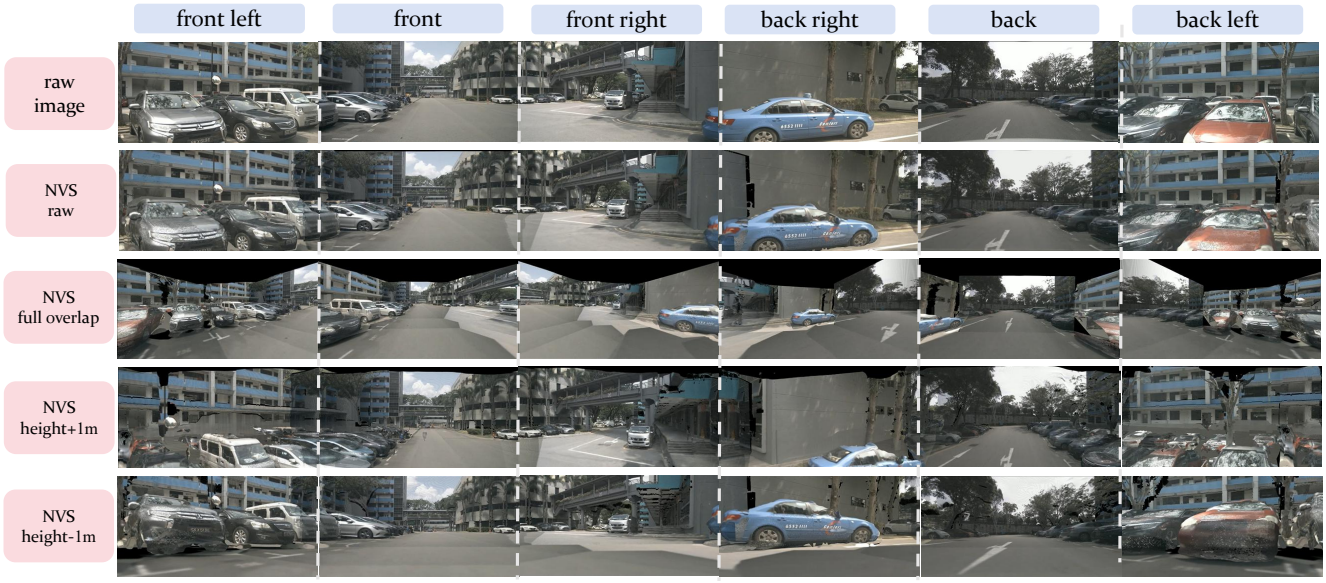


Figure 2. Qualitatively visualization of novel view images of Nuscenes.

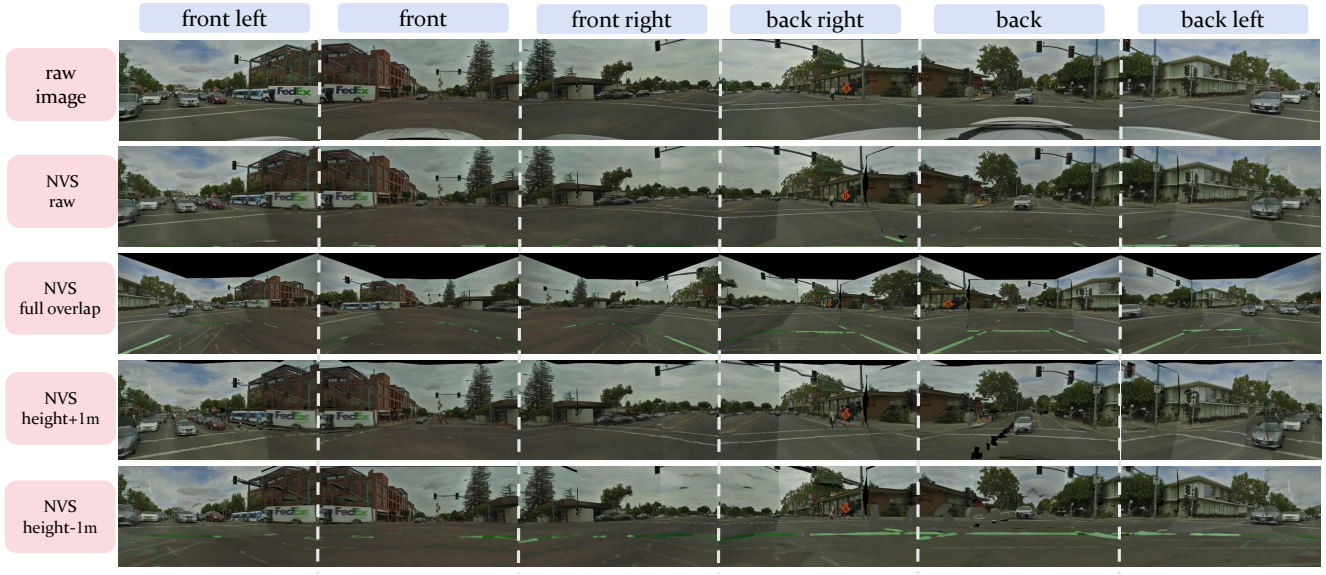


Figure 3. Qualitatively visualization of novel view images of Lyft.

centric Gaussians reconstructed by our training-free Gaussians Construction scheme. We select four setting to compare. "NVS raw" denote render image under the raw camera. Under "NVS full overlap", we set the horizontal FoV of all cameras as 120°, and we set each two neighbor camera with 60° yaw rotation to construct a full overlap array layout. Under "NVS height+1m", we rise the mounting height of each camera by 1m. Under "NVS height-1m", we lower the mounting height of each camera by 1m. These figure show that our low-cost training-free ego-centric 3D Gaussians Construction have acceptable photometric reconstruction quality. To be noted that, the reconstruction geometry is naturally precise.

Fig. 5 visualize the 3D detection results on the reconstructed textured point cloud under three cross-configuration testing settings. Fig. 5 shows that, use raw base model BEVDepth [6] to inference on data with new camera configuration results in terrible performance. After applying our CoIn3D, the cross-configuration generalization performance of model can be greatly enhanced. Then model can effectively and precisely detect 3D objects under new camera configuration.

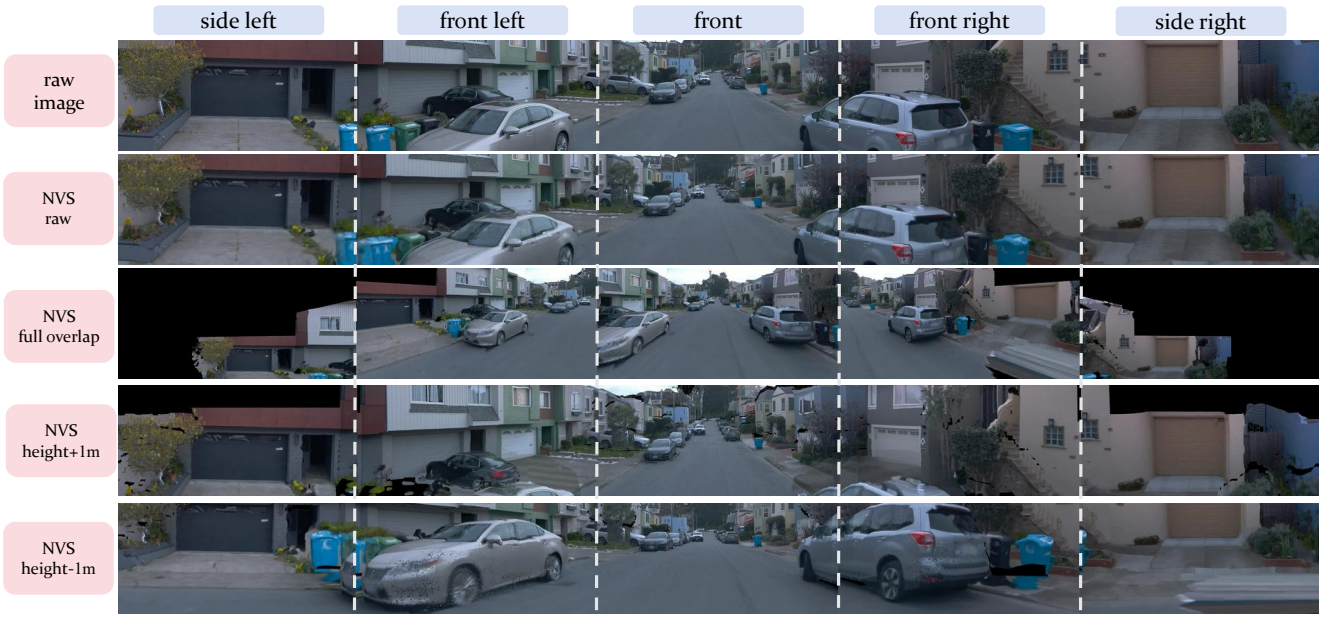


Figure 4. Qualitatively visualization of novel view images of Waymo.

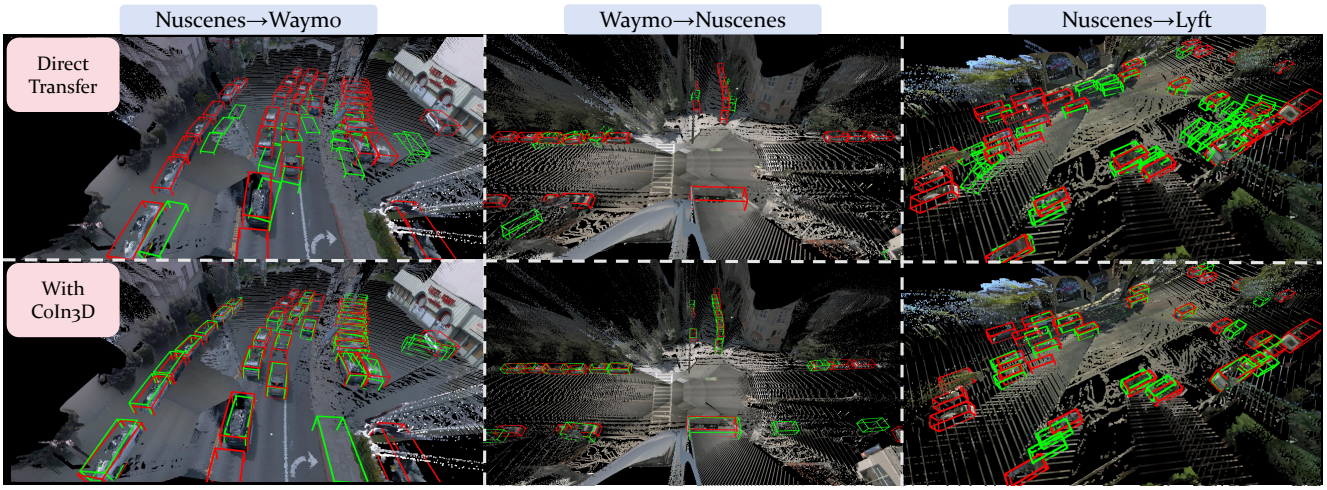


Figure 5. Qualitatively visualization of prediction results under 3d textured point cloud. Red boxes denote the ground truth boxes and green boxes denote the predicted boxes. The upper row shows results by directly using raw BEVDepth to inference on target configuration. The bottom row shows results by applying our CoIn3D.

## References


[1] 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. 1

[2] Chenjie Cao, Qiaole Dong, and Yanwei Fu. Zits++: Image inpainting by improving the incremental transformer on structural priors. *IEEE transactions on pattern analysis and machine intelligence*, 45(10):12667–12684, 2023. 1

[3] Gyusam Chang, Jiwon Lee, Donghyun Kim, Jinkyu Kim, Dongwook Lee, Daehyun Ji, Sujin Jang, and Sangpil Kim. Unified domain generalization and adaptation for multi-view 3d object detection. *Advances in Neural Information Processing Systems*, 37:58498–58524, 2024. 4

[4] Christy, Maggie, NikiNikatos, Phil Culliton, Vinay Shet, and Vladimir Iglovikov. Lyft 3d object detection for autonomous vehicles. https://kaggle.com/competitions/3d-object-detection-for-autonomous-vehicles, 2019. Kaggle. 1

[5] Zhaonian Kuang, Rui Ding, Meng Yang, Xinhu Zheng, and Gang Hua. Object-scene-camera decomposition and recom-

position for data-efficient monocular 3d object detection. *arXiv preprint arXiv:2602.20627*, 2026. 1

[6] 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. In *Proceedings of the AAAI conference on artificial intelligence*, pages 1477–1485, 2023. 4, 5

[7] Ye Li, Wenzhao Zheng, Xiaonan Huang, and Kurt Keutzer. Unidrive: Towards universal driving perception across camera configurations. *arXiv preprint arXiv:2410.13864*, 2024. 3

[8] Hao Lu, Jiaqi Tang, Xinli Xu, Xu Cao, Yunpeng Zhang, Guoqing Wang, Dalong Du, Hao Chen, and Yingcong Chen. Scaling multi-camera 3d object detection through weak-to-strong eliciting. *arXiv preprint arXiv:2404.06700*, 2024. 3

[9] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv preprint arXiv:2007.08501*, 2020. 1

[10] 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. 1

[11] Ignacio Vizzo, Tiziano Guadagnino, Jens Behley, and Cyrill Stachniss. Vdbfusion: Flexible and efficient tsdf integration of range sensor data. *Sensors*, 22(3):1296, 2022. 1

[12] Haotian Wang, Meng Yang, Xinhu Zheng, and Gang Hua. Scale propagation network for generalizable depth completion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 1

[13] Chengrui Wei, Meng Yang, Lei He, and Nanning Zheng. Fs-depth: Focal-and-scale depth estimation from a single image in unseen indoor scene. *IEEE Transactions on Circuits and Systems for Video Technology*, 34(11):10604–10617, 2024. 4

[14] Liangtao Zheng, Yicheng Liu, Yue Wang, and Hang Zhao. Cross-dataset sensor alignment: Making visual 3d object detector generalizable. In *Conference on Robot Learning*, pages 1903–1929. PMLR, 2023. 3