DATA: Domain-And-Time Alignment for High-Quality Feature Fusion in Collaborative Perception

Supplementary Material

1. Detailed Information about Module Designs

1.1. Observability-constrained Discriminator Implementation

The observability-constrained discriminator (OD) employs a lightweight convolutional architecture to distinguish between ego and collaborative domains. The discriminator consists of two convolutional layers: a first layer with 256 channels and kernel size 1×1 followed by ReLU activation, and a final layer with a single output channel and kernel size 1×1 that outputs predicted domain labels.

1.2. Multi-scale Feature Processing for BEV Representation

The multi-scale feature processing module transforms features at different spatial resolutions to form a comprehensive Bird's Eye View (BEV) representation.

Assume that three scales of features, which are denoted as $F_{\text{large}} \in \mathbb{R}^{64\times H\times W}$, $F_{\text{middle}} \in \mathbb{R}^{128\times \frac{H}{2}\times \frac{W}{2}}$, and $F_{\text{small}} \in \mathbb{R}^{256\times \frac{H}{4}\times \frac{W}{4}}$, are extracted from the backbone network, and then the conversion to BEV features follows a systematic process.

Each scale undergoes a deconvolution operation to unify spatial and channel dimensions, and this operation can be expressed as

$$U_{\text{large}} = \Theta_1(F_{\text{large}}), \tag{1}$$

$$U_{\text{middle}} = \Theta_2(F_{\text{middle}}), \tag{2}$$

$$U_{\text{small}} = \Theta_3(F_{\text{small}}), \tag{3}$$

where Θ_1, Θ_2, and Θ_3 represent deconvolution networks with strides 1, 2, and 4, respectively. Each deconvolution network consists of a transposed convolution layer, batch normalization, and ReLU activation, projecting the features to a uniform spatial resolution of $H \times W$ and channel dimension of 128.

The final BEV representation is formed by concatenating these upsampled features along the channel dimension, formulated as $H_{\text{BEV}} = [U_{\text{large}}, U_{\text{middle}}, U_{\text{small}}]$. The resulting BEV feature map $H_{\text{BEV}} \in \mathbb{R}^{384\times H\times W}$ integrates information from all scales while maintaining consistent spatial dimensions. This approach, which is essential for accurate 3D object detection in the BEV space, preserves both fine-grained details from larger-scale features and semantic context from smaller-scale features.

1.3. Computational Complexity Analysis: Block-Wise vs. Global Computation

We analyze the computational complexity of block-wise computation versus global computation for calculating cosine similarity between feature maps. Consider feature maps of size $H \times W$ with C channels, then for the block-wise approach, we use the window size $l \times l$, where l is divisible by both H and W for simplicity. Herein, $\mathbf{f}_{\text{p}}$ denotes the predicted feature map and $\mathbf{f}_{\text{gt}}$ denotes the ground truth feature map.

1.3.1. Global Computation

For global computation across the entire feature map, the computational cost is listed as follows:

(i) Computing vector dot products $\langle \mathbf{f}_{\text{p}}, \mathbf{f}_{\text{gt}} \rangle$ requires $C \times H \times W$ multiplications and $(C-1)\times H \times W$ additions.

(ii) Computing ℓ_2 norms $\|\mathbf{f}_{\text{p}}\|_2$ and $\|\mathbf{f}_{\text{gt}}\|_2$ requires $2 \times C \times H \times W$ multiplications for squaring each element, $2 \times (C-1) \times H \times W$ additions for summing squared elements, and $2 \times H \times W$ square root operations.

(iii) Computing cosine similarity $s = \frac{\langle \mathbf{f}_{\text{p}}, \mathbf{f}_{\text{gt}} \rangle}{\|\mathbf{f}_{\text{p}}\|_2 \cdot \|\mathbf{f}_{\text{gt}}\|_2}$ requires $H \times W$ divisions and $H \times W$ multiplications.

(iv) Finally, computing MSE loss $(s-1)^2$ requires $H \times W$ subtractions, $H \times W$ squaring operations, and $(H \times W - 1)$ additions for the final sum.

In total, the global computation requires approximately $(3C + 2) \times H \times W$ multiplications, $(3C - 1) \times H \times W$ additions, $2 \times H \times W$ square roots, and $H \times W$ divisions, yielding a computational complexity of $\mathcal{O}(C \times H \times W)$.

1.3.2. Block-Wise Computation with Multi-Window Strategy

Our approach uses two complementary window partitioning strategies:

The first strategy (W_1 described in Eq. (11)) applies standard partitioning, yielding $\lfloor H/l \rfloor \times \lfloor W/l \rfloor$ windows. The second strategy (W_2 described in Eq. (12)) employs offset partitioning (by $l/2$ compared with W_1), yielding $\lfloor (H-l)/l \rfloor \times \lfloor (W-l)/l \rfloor$ windows due to the offset introduced along the top, bottom, left, and right boundaries of the feature map. For typical dimensions in our implementation ($H = 256, W = 128, l = 16$), we have $|W_1| = 16 \times 8 = 128$ windows and $|W_2| = 15 \times 7 = 105$ windows.

For each window in either strategy, computing dot products requires $C \times l \times l$ multiplications and $(C-1) \times l \times l$ additions. ℓ_2 norm computation requires $2 \times C \times l \times l$ multiplications, $2 \times (C-1) \times l \times l$ additions, and $2 \times l \times l$ square roots. Computing cosine similarity requires $l \times l$ divisions and $l \times l$ multiplications, and MSE loss computation requires $l \times l$ subtractions and $l \times l$ multiplications.

The total computation across both window strategies is

$$\left[\lfloor\frac{H}{l}\rfloor \times \lfloor\frac{W}{l}\rfloor + \lfloor\frac{H-l}{l}\rfloor \times \lfloor\frac{W-l}{l}\rfloor\right] \times [(3C+2) \times l \times l]. \tag{4}$$

With our typical dimensions, this is approximately $1.8 \times (3C+2) \times H \times W$, giving a computational complexity of $\mathcal{O}(C \times H \times W)$. This is asymptotically equivalent to the global approach.

1.3.3. Advantages of Block-Wise Computation

The block-wise approach with dual window strategy offers several significant advantages in temporal feature alignment as follows:

First, the windowed computation provides stronger learning signals for capturing localized motion patterns, enabling the model to better distinguish between different motion behaviors within the same scene. Traffic scenes exhibit both structured patterns and individual object movements, which benefit from this localized analysis approach.

Second, window-based computation rebalances the influence of foreground objects in similarity calculations, effectively counteracting the predominant impact of background regions that typically dominate the feature space. Consider that an object spans $m \times n$ pixels, where $m, n < l$. In global similarity computation, the computation of foreground objects would be proportional to $(m \times n)/(H \times W)$ when calculating the overall cosine similarity metric. However, in the window-based approach, where similarity is calculated independently for each window before computing the MSE loss, the foreground objects make more contribution to the similarity, reaching $(m \times n)/(l \times l)$. This would become even more significant when window size l is smaller than the feature dimensions H and W. This enhanced representation of foreground elements strengthens motion pattern modeling for salient objects, which typically occupy smaller spatial regions compared to the background.

Third, the complementary window partitioning strategies ensure comprehensive spatial coverage. Given the window size l and the offset $l/2$, any object of size up to $l \times l$ will be fully contained within at least one window from either partitioning strategy, regardless of its position in the feature map. This property prevents foreground objects from being fragmented across multiple windows, providing coherent supervision signals for learning object motion patterns.

Fourth, the block-wise approach addresses the challenge of balancing attention between foreground and background regions. By processing feature maps in window-based units, the model allocates more balanced computational attention across the feature space, rather than being dominated by background regions that typically occupy the majority of the scene. As demonstrated in our experimental results, this approach significantly improves temporal alignment quality and detection performance, particularly for foreground objects of interest in autonomous driving scenarios.

1.4. StructConv: Multi-directional Structure Enhancement

The StructConv operation employs five specialized 3×3 convolutions that collectively enhance structural details in foreground features. Each targets specific geometric patterns while maintaining computational efficiency through parameter combination. We detail these specialized convolutions as follows.

Feature Preservation Convolution. This convolution maintains the original feature representation by computing standard weighted sums across neighborhood pixels. This serves as the foundation upon which structural enhancements are built while preserving essential intensity information.

Center-Surround Contrast Convolution. This convolution enhances local contrast by modifying the center weight of the kernel to be the negative sum of all surrounding weights. This operation highlights intensity transitions at feature boundaries and enhances object contours.

Horizontal Edge Convolution. This convolution detects horizontal structures by placing positive weights on the left column, zeros in the center column, and exact negatives of the left-side weights on the right column of the kernel. This symmetrical weight pattern creates a directional gradient detector that emphasizes horizontal edges commonly found in vehicle appearance.

Vertical Edge Convolution. This convolution captures vertical features through a principle similar to the Horizontal Edge Convolution, but with positive weights at the top row, zeros in the middle row, and exact negatives of the top-row weights at the bottom row of the kernel. This arrangement enhances vertical boundaries and surface transitions in the scene.

Diagonal Structure Convolution. This convolution identifies corner features and angular transitions through a specialized weight permutation. The implementation subtracts a rotated version of the original kernel from itself, creating a pattern that responds strongly to diagonal intensity gradients present at object corners.

While these convolutions effectively enhance structural details, they may introduce spurious foreground features

that may negatively impact detection. To address this, the enhanced features undergo a coupled height-semantic verification process that selectively preserves structural enhancements while suppressing false features. This verification leverages the inherent channel-wise height and semantic correlations in pillar-encoded features to ensure that only structurally reasonable enhancements are retained.

The five convolutions are efficiently implemented by combining their weights and biases into a single operation, providing comprehensive structural enhancement without the computational overhead of separate convolutions.

1.5. Foreground Estimator Implementation

The foreground estimator $\Phi(\cdot)$, utilized in both CDAM and IFAM, is implemented as a lightweight network to identify foreground regions. The network architecture consists of a 3×3 convolution that halves the input channel dimension, followed by batch normalization and ReLU activation, and a final 1×1 convolution with sigmoid activation to produce single-channel output.

Herein, $\Phi(\cdot)$ is supervised by the foreground occupancy loss function that computes a weighted focal loss between predicted foreground maps and foreground labels across spatial locations. This loss function emphasizes the importance of correctly identifying occupied regions while accounting for the class imbalance between foreground and background pixels.

Then the foreground occupancy loss $\mathcal{L}_{foreground}$ can be formulated as

$$\mathcal{L}_{foreground} = \sum_{s=1}^{S} w_s \cdot \mathcal{L}_{focal}(M_s, M_s^{\text{gt}}), \tag{5}$$

where S is the total number of spatial locations in the feature map, M_s denotes the predicted foreground value from the foreground estimator at location s, and M_s^{gt} is the corresponding ground truth foreground label derived from bounding boxes.

The weighting factor w_s is defined as

$$w_s = \frac{M_s^{\text{gt}} \cdot w_{pos} + (1 - M_s^{\text{gt}}) \cdot w_{neg}}{\max\left(\sum_{s=1}^{S} M_s^{\text{gt}}, 1\right)}, \tag{6}$$

where $w_{pos} = 2$ is the weight for positive samples and $w_{neg} = 1$ is the weight for negative samples. The denominator normalizes the weights by the total number of positive samples, with a minimum value of 1 to prevent division by zero.

The focal loss $\mathcal{L}_{focal}$ can be calculated as

$$\begin{aligned}\mathcal{L}_{focal}(p, y) = &- (y \cdot 0.25 \cdot (1-p)^2 \cdot \log(p) \\ &+ (1-y) \cdot (1-\alpha) \cdot p^2 \cdot \log(1-p)).\end{aligned} \tag{7}$$

This supervisory signal guides the foreground estimator to accurately predict foreground occupancy while handling the inherent imbalance between foreground and background regions in the scene.

2. Detailed Dataset Information and Experiment Configuration

2.1. Detailed Dataset Information

Experiments are conducted on three collaborative perception datasets: DAIR-V2X-C [8], V2XSET [7], and V2XSIM [4]. (i) DAIR-V2X-C, which focuses on vehicle-infrastructure collaborative data, is a subset of DAIR-V2X and contains 39,000 frames. Following the official benchmark in [8], a synchronized subset VIC-Sync with 9,311 frame pairs is split into training/validation/testing sets in a ratio of 5:2:3. We adopt the complemented annotations [5], with the perception range is set to $x \in [-102.4m, 102.4m]$ and $y \in [-51.2m, 51.2m]$. (ii) V2XSIM is a cooperative perception dataset co-simulated by SUMO [3] and CARLA [1]. The dataset consists of 10,000 frames with 501K annotated boxes, split into 8,000/1,000/1,000 for training/validation/testing. The perception range is set to $x \in [-32m, 32m]$ and $y \in [-32m, 32m]$. (iii) V2XSet, co-simulated by OpenCDA [6] and CARLA, is a V2X dataset with realistic noise simulation. The dataset contains 11,447 frames, split into 6,694/1,920/2,833 for training/validation/testing. The perception range is set to $x \in [-140m, 140m]$ and $y \in [-40m, 40m]$.

2.2. Three-stage Training and Loss Function

The DATA framework employs a sequential three-stage training strategy, each with specialized loss functions tailored to specific aspects of the model.

2.2.1. Stage 1: Domain Alignment and Feature Extraction

In the first stage, we jointly optimize the encoder, CDAM, IFAM, and detection head by using synchronized data. The total loss function for this stage is formulated as

$$\mathcal{L}_{\text{stage1}} = \mathcal{L}_{\text{det}} + \lambda_{\text{fore}} \cdot \mathcal{L}_{\text{foreground}} + \lambda_{\text{domain}} \cdot \mathcal{L}_{\text{domain}} \tag{8}$$

where $\mathcal{L}_{\text{det}}$ is the detection loss comprising three components as

$$\mathcal{L}_{\text{det}} = \mathcal{L}_{\text{cls}} + \lambda_{\text{reg}} \cdot \mathcal{L}_{\text{reg}} + \lambda_{\text{dir}} \cdot \mathcal{L}_{\text{dir}}, \tag{9}$$

where $\mathcal{L}_{\text{cls}}$ denotes a Sigmoid Focal Loss for classification, $\mathcal{L}_{\text{reg}}$ represents a Weighted Smooth ℓ_1 Loss for regression, and $\mathcal{L}_{\text{dir}}$ indicates a Weighted Softmax Classification Loss for direction prediction. The weighting factors are set to $\lambda_{\text{reg}} = 2.0$, $\lambda_{\text{dir}} = 0.2$, $\lambda_{\text{fore}} = 0.4$, and $\lambda_{\text{domain}} = 1.0$.

2.2.2. Stage 2: Temporal Alignment

The second stage focuses on training PTAM with asynchronous data while freezing all parameters from the first

stage. The loss function is defined as

$$\mathcal{L}_{\text{stage2}} = \mathcal{L}_{\text{det}} + \lambda_{\text{temporal}} \cdot \mathcal{L}_{\text{temporal}}, \tag{10}$$

where the detection loss $\mathcal{L}_{\text{det}}$ maintains the same form and parameters as in Stage 1, while $\mathcal{L}_{\text{temporal}}$ is the multi-scale and multi-window temporal alignment loss described at the end of Section 3.3.3. Herein, we set $\lambda_{\text{temporal}} = 1.0$ to balance the influence of detection and temporal alignment objectives.

2.2.3. Stage 3: Compression Network

The final stage trains the transmission compression and recovery network by using asynchronous data with all parameters from previous stages frozen. The loss function combines detection objectives with reconstruction fidelity as

$$\mathcal{L}_{\text{stage3}} = \mathcal{L}_{\text{det}} + \lambda_{\text{recon}} \cdot \mathcal{L}_{\text{recon}}, \tag{11}$$

where the detection loss $\mathcal{L}_{\text{det}}$ remains consistent with the previous stages, while $\mathcal{L}_{\text{recon}}$ is implemented as a Mean Squared Error (MSE) loss that quantifies the feature reconstruction quality after compression. The compression network adopts a UNet architecture to maintain spatial feature relationships during compression and decompression. Herein, we set $\lambda_{\text{recon}} = 1.0$.

For network optimization, the Adam optimizer [2] is adopted across all three stages. In the first stage, the initial learning rate is set to 0.002 with $10\times$ decay at epochs 15 and 30, and the models are trained for 40 epochs. The second and third stages both employ a fixed learning rate of 0.001 for 10 epochs. A batch size of 2 is used throughout the entire training process. This sequential training approach allows each component to specialize in its intended functionality while maintaining overall system coherence.

3. Visualization Results

Notably, in all visualization results below, red boxes represent prediction results, and green boxes represent ground truth.

3.1. Qualitative Comparison Under Asynchronous Conditions

3.1.1. Turning Scenario

Figure 1 illustrates the delay compensation performance of DATA and FFNet in turning scenarios at 300 ms and 500 ms delays. Since turning generally involves variable-speed motion, there are higher requirements for the network to model complex movements. In Figure 1, it shows that DATA exhibits good prediction performance under these severely delayed conditions, while the prediction results of FFNet demonstrate limited capability to accurately compensate the mismatches under such severely delayed conditions. Also, the ablation results of PTAM show significant performance improvements of detection when using PTAM, demonstrating the robustness of PTAM to acquire high-quality features of temporal alignment.

3.1.2. Intersection Scenario

The figure 2 further exhibits the performance of DATA and FFNet under conditions with intersecting traffic flows, and this requires models to simultaneously capture motion trends in different regions. As observed from right bottom subgraph of Figure 2, the misalignment caused by latency can reach one vehicle length, posing significant challenges to reliable perception in autonomous driving. After implementing PTAM, DATA successfully achieves displacement compensation in both directions (horizontal and vertical directions in the figure), exhibiting both local and global motion modeling capability. In this scenario, FFNet effectively compensates for the traffic flow in the vertical direction but performs inadequately in compensation for horizontal displacement, reflecting potential deficiencies in local motion modeling that may result from its global supervision approach.

3.2. Qualitative Comparison Under Synchronous Conditions

Figure 3 illustrates a scenario from the DAIR-V2X dataset where object groups are positioned in three distinct regions: near the roadside infrastructure (collaborator), around the ego vehicle, and in the intermediate zone between them. This configuration creates three distinct perceptual regions: areas predominantly observed by the ego vehicle, areas mainly observed by the collaborator, and overlapping areas observed by both agents. This mixed observability pattern presents a significant challenge to the perception system's ability to extract domain-invariant features for robust perception. DATA, FFNet, and Where2comm all demonstrate effective detection in areas where both the point clouds of ego and collaborator exist. However, FFNet and Where2comm generate false positives in regions dominated by point clouds either from ego egent or collaborative agent, while DATA maintains accurate detection performance. This demonstrates that DATA effectively learns domain-invariant features through density-consistent and observability-consistent domain alignment, exhibiting stable detection performance in scenarios with significant domain gaps.

Figure 5 presents a demanding collaborative perception scenario with a 280 m range and noise interference from the V2XSET dataset. The scene contains dense traffic conditions with numerous occlusions and point cloud sparsity at extended distances, creating an ideal testing environment for evaluating collaborative perception capabilities. FFNet and Where2comm achieve good perception results in regions where collaborative agents provide reliable information, but exhibit detection inaccuracies and false positives at

greater distances due to point cloud sparsity and noise interference. DATA effectively mitigates noise interference and the challenges of sparse long-distance point clouds through learned domain-invariant features and foreground enhancement, achieving robust detection performance across the entire extended range. Figure 4 similarly demonstrates a noisy scenario from V2XSIM with occlusions and sparse point clouds. DATA achieves effective perception performance, while FFNet and Where2comm exhibit missed detections and false positives in regions with sparse point clouds, further validating DATA's robust and reliable perception capabilities.

References

[1] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. Carla: An open urban driving simulator. In *Conference on robot learning*, pages 1–16. PMLR, 2017. 3

[2] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 4

[3] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of sumo-simulation of urban mobility. *International journal on advances in systems and measurements*, 5(3&4), 2012. 3

[4] Yiming Li, Dekun Ma, Ziyan An, Zixun Wang, Yiqi Zhong, Siheng Chen, and Chen Feng. V2x-sim: Multi-agent collaborative perception dataset and benchmark for autonomous driving. *IEEE Robotics and Automation Letters*, 7(4):10914–10921, 2022. 3

[5] Yifan Lu, Quanhao Li, Baoan Liu, Mehrdad Dianati, Chen Feng, Siheng Chen, and Yanfeng Wang. Robust collaborative 3d object detection in presence of pose errors. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4812–4818. IEEE, 2023. 3

[6] Runsheng Xu, Yi Guo, Xu Han, Xin Xia, Hao Xiang, and Jiaqi Ma. Opencda: an open cooperative driving automation framework integrated with co-simulation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1155–1162. IEEE, 2021. 3

[7] Runsheng Xu, Hao Xiang, Zhengzhong Tu, Xin Xia, Ming-Hsuan Yang, and Jiaqi Ma. V2x-vit: Vehicle-to-everything cooperative perception with vision transformer. In *European conference on computer vision*, pages 107–124. Springer, 2022. 3

[8] Haibao Yu, Yizhen Luo, Mao Shu, Yiyi Huo, Zebang Yang, Yifeng Shi, Zhenglong Guo, Hanyu Li, Xing Hu, Jirui Yuan, et al. Dair-v2x: A large-scale dataset for vehicle-infrastructure cooperative 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21361–21370, 2022. 3

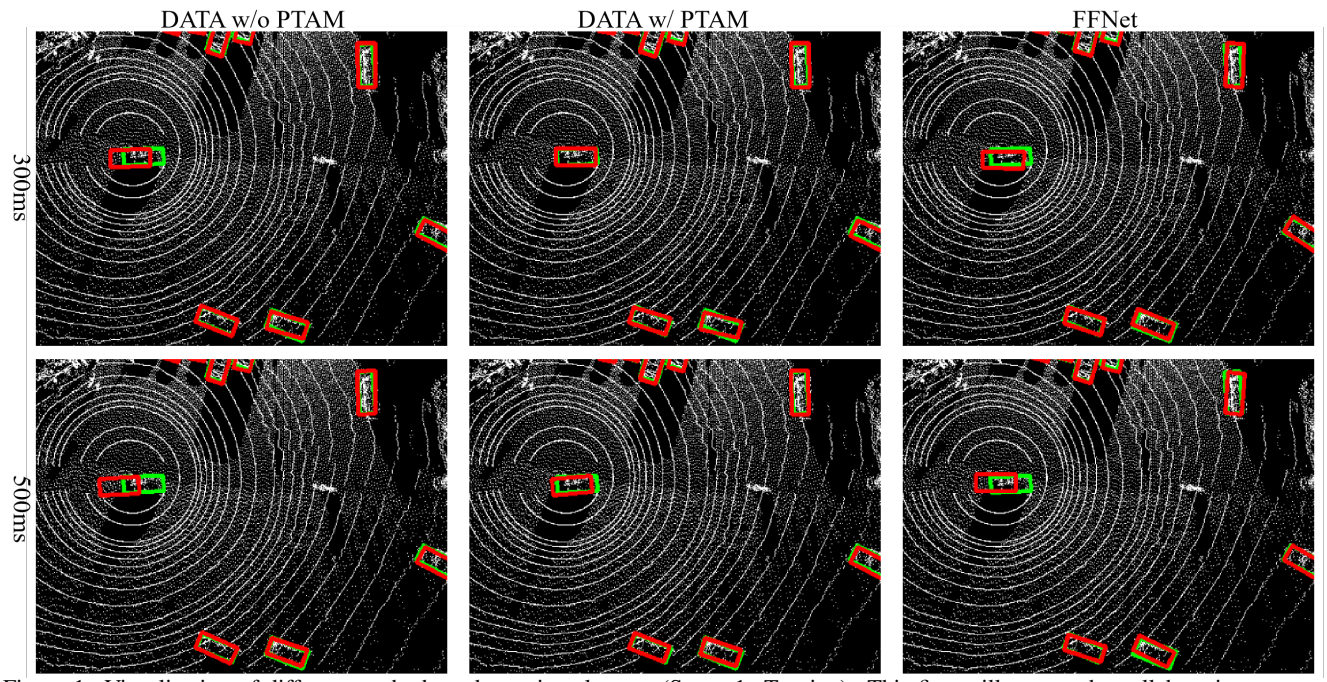

Figure 1. Visualization of different methods under various latency (Scene 1: Turning). This figure illustrates the collaborative process between a vehicle and the road infrastructure in a turning scenario. DATA showcases impressive ability to model variable-speed movements evidenced by precise compensation under severe latency. A significant improvement can be observed by adding PTAM to align the temporal desynchrony.

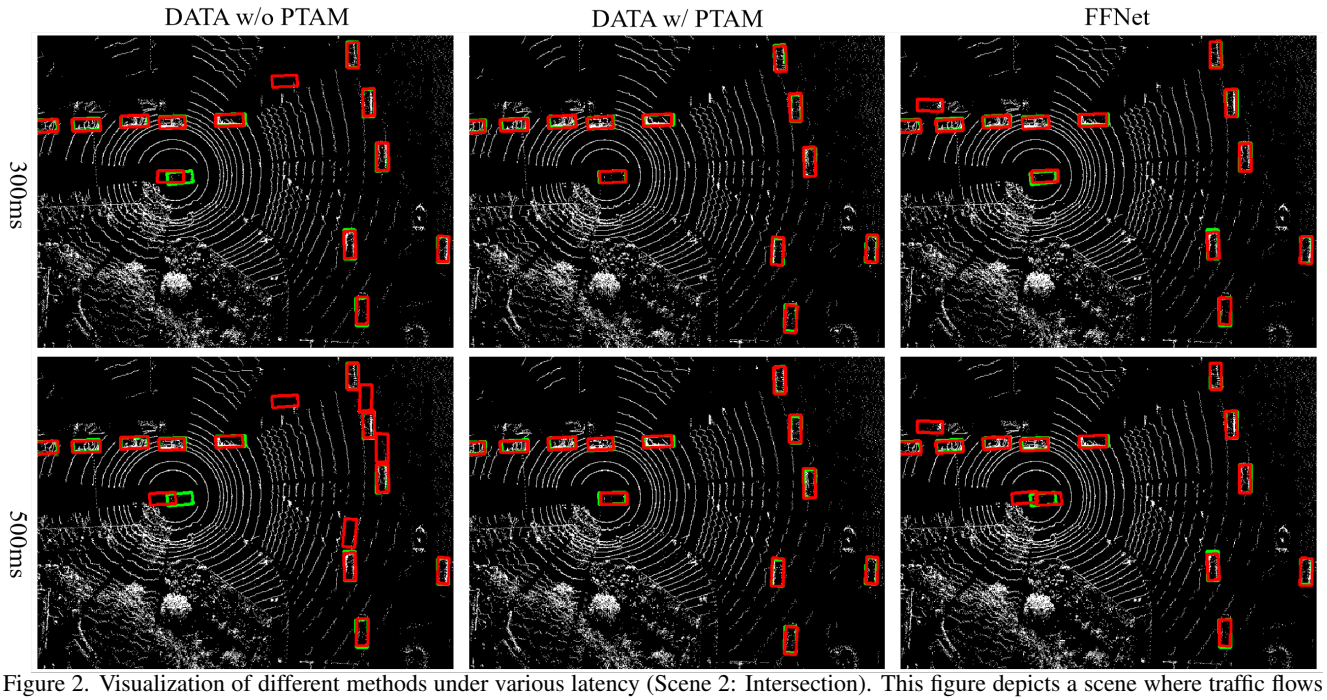

Figure 2. Visualization of different methods under various latency (Scene 2: Intersection). This figure depicts a scene where traffic flows intersect, challenging models to simultaneously capture motion trends in different regions. The result showcases that DATA aligns motions of two distinct directions properly, proving reliable capabilities in processing motion information of a scene-wide region.

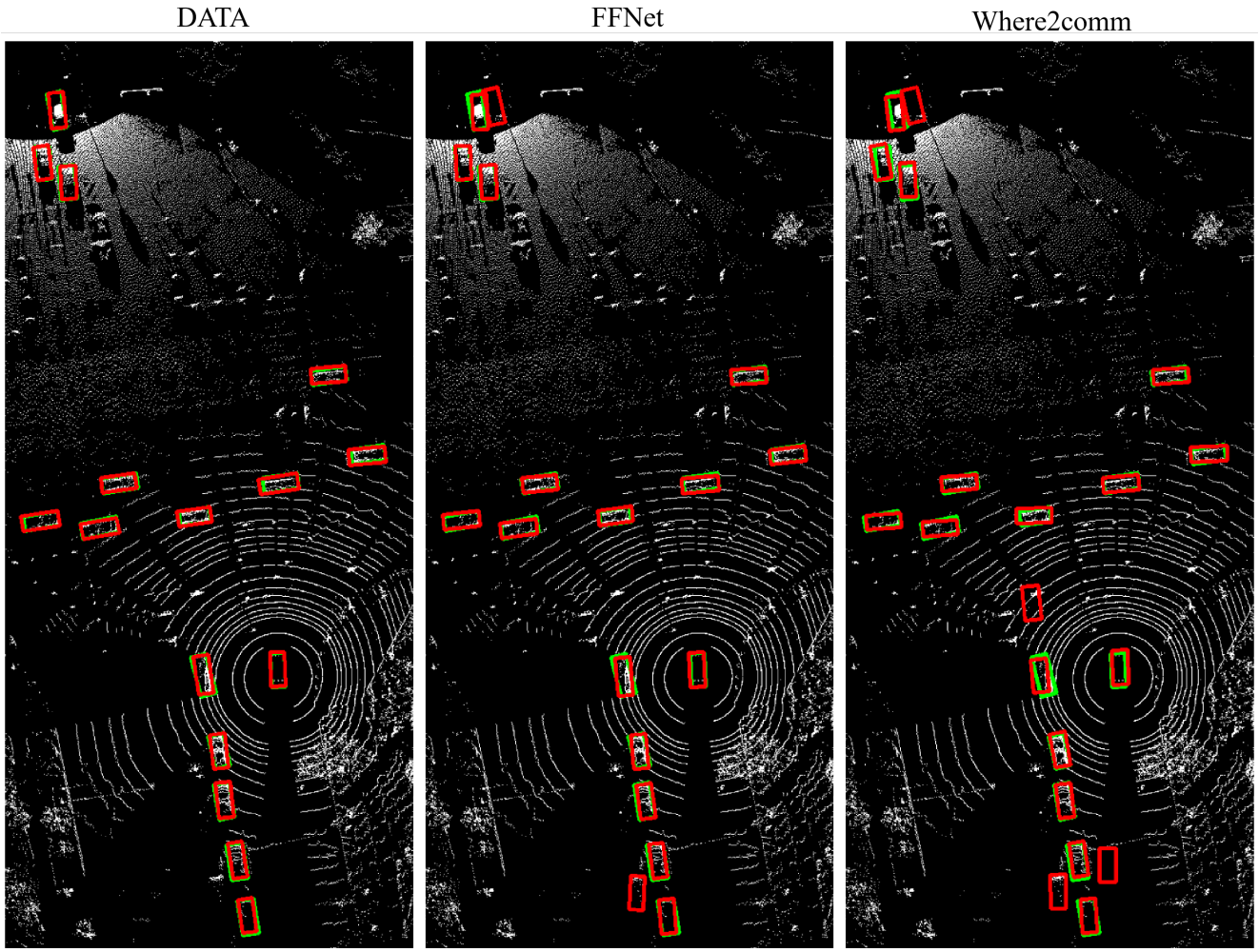

Figure 3. Visualization of different methods on DAIR-V2X. A vehicle-infrastructure collaborative perception scenario from DAIR-V2X shows that DATA outperforms FFNet and Where2comm by maintaining accurate detection across all regions despite domain gaps, demonstrating superior domain-invariant feature extraction under mixed observability challenges.

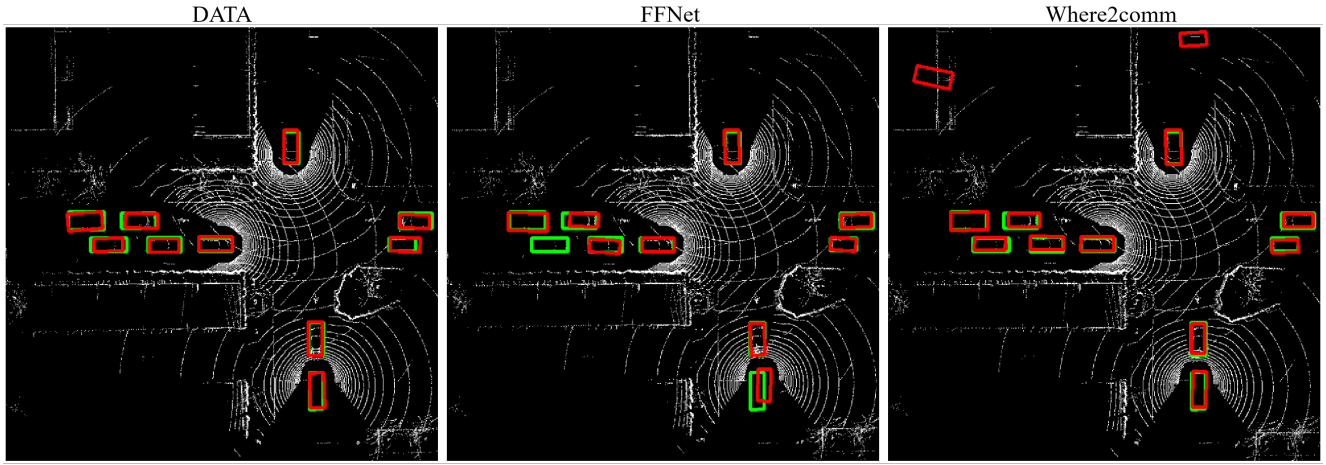

Figure 4. Visualization of different methods on V2XSIM. In a complex V2XSIM environment with two vehicles and infrastructure collaboration, DATA delivers superior perception by avoiding the missed detections and false positives that plague competitors when handling occluded objects and sparse point clouds.

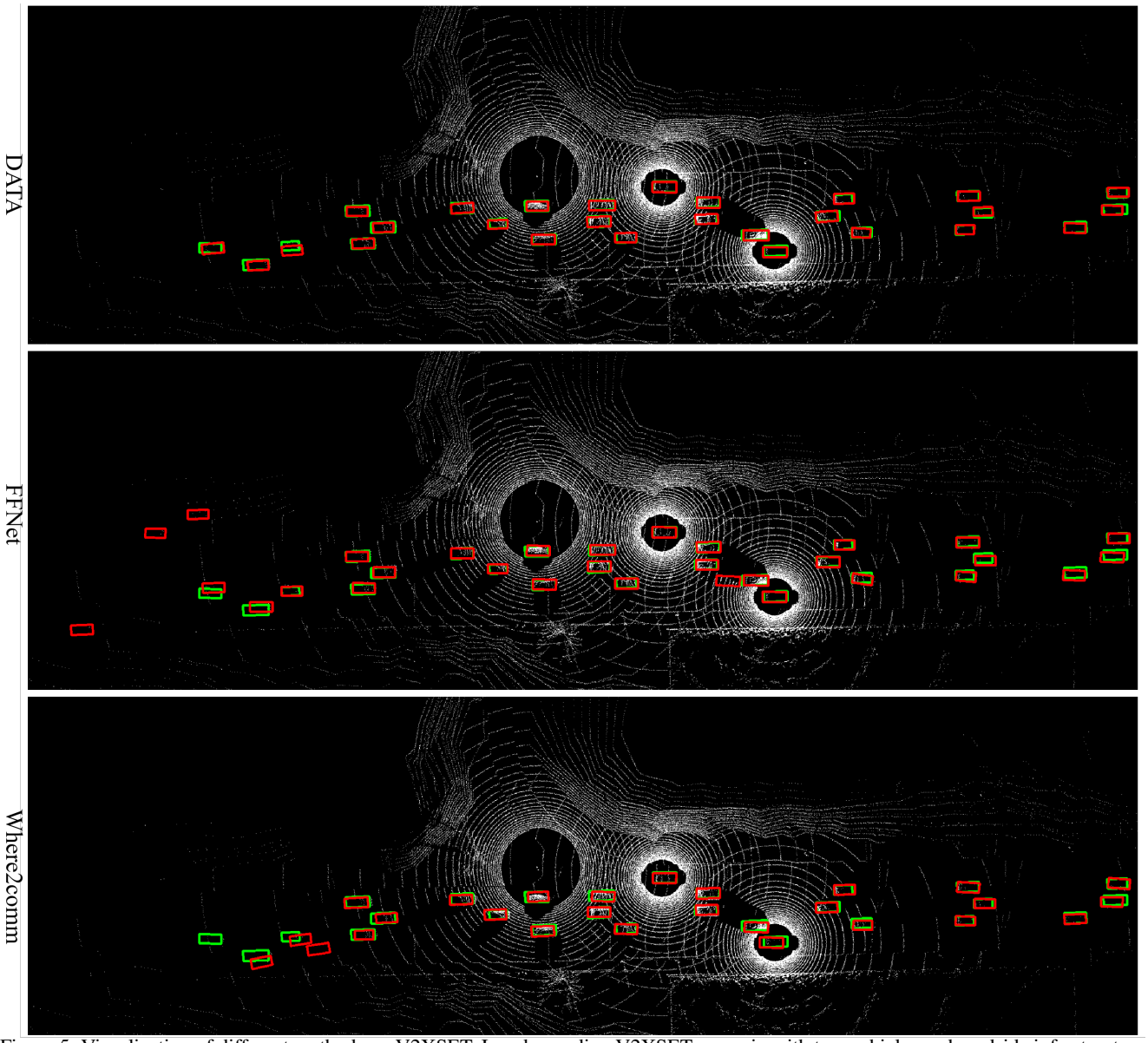

Figure 5. Visualization of different methods on V2XSET. In a demanding V2XSET scenario with two vehicles and roadside infrastructure collaborating, DATA outperforms competitors by accurately detecting objects despite dense traffic, occlusions, and sparse distant point clouds, demonstrating superior noise mitigation and domain-invariant feature learning.

Symbol	Definition	Section/Module
3.1 Problem Formulation and Overall Architecture		
N	Number of agents in collaborative perception system	Problem Formulation
i, j	Subscripts denoting ego agent and collaborative agents, where $i \neq j$	Problem Formulation
$X_i(t)$	Latest data of ego agent at current time t	Problem Formulation
$X_j(t-\tau)$	Data transmitted by collaborative agents at timestamp $t-\tau$	Problem Formulation
$X_j(t-\tau-\Delta T)$	Data of collaborative agent at timestamp $t-\tau-\Delta T$	Problem Formulation
τ	Transmission delay	Problem Formulation
ΔT	Time interval	Problem Formulation
3.2.1 Proximal-region Hierarchical Downsampling (PHD)		
$\mathcal{O}_i$	Set of all objects observable to ego agent i	CDAM-PHD
N_i^{total}	Total number of observable objects	CDAM-PHD
o_k	k-th object	CDAM-PHD
$\mathcal{O}_i^{prox}$	Objects within the proximal region	CDAM-PHD
$d_i(o_k)$	Distance from ego agent i to k-th object	CDAM-PHD
d_{th}	Distance threshold	CDAM-PHD
N_{proc}	Number of objects selected from $\mathcal{O}_i^{prox}$ for subsequent processing	CDAM-PHD
N_{max}	Predefined maximum number of objects	CDAM-PHD
$B_{i,k}^{out}$	Oriented bounding box of k-th object	CDAM-PHD
$B_{i,k}^{in}$	Inner bounding box with scaling factor α	CDAM-PHD
(x_k, y_k, z_k)	Center coordinates of k-th object	CDAM-PHD
(h_k, w_k, l_k)	Dimensions (height, width, length) of k-th object	CDAM-PHD
θ_k	Orientation of k-th object	CDAM-PHD
α	Scaling factor to adjust bounding box dimensions, $\alpha \in (0, 1)$	CDAM-PHD
$R_{i,k}^{in}, R_{i,k}^{out}$	Point sets in inner and outer regions	CDAM-PHD
β_{in}, β_{out}	High and conservative downsampling ratios for inner and outer regions	CDAM-PHD
FPS$(\cdot, \cdot)$	Farthest Point Sampling operation	CDAM-PHD
$\tilde{R}_{i,k}^{in}, \tilde{R}_{i,k}^{out}$	Downsampled inner and outer regions	CDAM-PHD
$\tilde{P}_{i,k}$	Point clouds of k-th object after hierarchical downsampling	CDAM-PHD

Table 1. Notation Table for DATA Paper - Part I: Problem Formulation and PHD

Symbol	Definition	Section/Module
3.2.2 Observability-constrained Discriminator (OD)		
H_i, H_j	BEV features of ego and collaborative agents	CDAM-OD
C, H, W	Channel dimension, height, and width of features	CDAM-OD
M_i, M_j	Observability maps, $M_i, M_j \in \mathbb{R}^{1\times H\times W}$	CDAM-OD
$\Phi(\cdot)$	Foreground estimator	CDAM-OD
$H_{j\to i}, M_{j\to i}$	Collaborative features and observability map projected onto ego coordinates	CDAM-OD
$\mathcal{V}$	Set of valid grids in transformed feature map	CDAM-OD
H_j^{comp}, M_j^{comp}	Complemented feature and observability map	CDAM-OD
$I_{\mathcal{V}}$	Indicator function (equals 1 for points in $\mathcal{V}$ and 0 for others)	CDAM-OD
W	Observability weighting map, $W \in \mathbb{R}^{1\times H\times W}$	CDAM-OD
$\mathcal{L}_{domain}$	Domain alignment objective	CDAM-OD
$\mathcal{S}$	Set of all spatial positions	CDAM-OD
sp	One spatial position	CDAM-OD
W_{flat}	Flattened observability weighting map	CDAM-OD
Ψ_θ	Feature extractor (point clouds as input and BEV features as output)	CDAM-OD
D_μ	Discriminator	CDAM-OD
$\mathcal{L}_{BCE}$	Binary cross-entropy loss	CDAM-OD
Z	Domain label (0 for ego agent and 1 for collaborative agent)	CDAM-OD
γ	Negative scaling factor in gradient reversal layer ($\gamma = -0.1$)	CDAM-OD

Table 2. Notation Table for DATA Paper - Part II: Observability-constrained Discriminator

Symbol	Definition	Section/Module
3.3 Progressive Temporal Alignment Module (PTAM)		
$F_j(t, x)$	Features at collaborative agent at time t and position x	PTAM
$v(t - \Delta t, x)$	Velocity field describing motion of features at time $t - \Delta t$	PTAM
Δt	Time interval within range of typical transmission delay	PTAM
Δp	Motion field representing velocity, $\Delta p \in \mathbb{R}^{2\times H\times W}$	PTAM
ξ	Temporal scaling factor corresponding to Δt	PTAM
F_j^{inter}	Intermediate feature predicted by collaborative agent	PTAM
$\hat{F}_j(t - \tau)$	Feature from collaborative agent j at time $t - \tau$, as received by ego agent	PTAM
$\hat{F}_j^{inter}$	Intermediate feature from collaborative agent j, as received by ego agent	PTAM
$\hat{F}_j(t)$	Temporally aligned features	PTAM
F_{latest}, F_{prev}	Latest feature and its previous feature	PTAM
ΔF	Temporal feature difference, $\Delta F = F_{latest} - F_{prev}$	PTAM
w_{samp}	Sampling weight, $w_{samp} \in \mathbb{R}^{1\times H\times W}$	PTAM
$f_{warp}(\cdot)$	Bilinear sampling operation	PTAM
$s1, s2$	Superscripts indicating first stage and second stage	PTAM
ΔM	Motion difference field	PTAM
f_M	Global context vector	PTAM
f_T	Temporal encoding with sinusoidal positional embeddings	PTAM
3.3.3 Multi-window Self-supervised Training Strategy		
s	Spatial scale index	PTAM Training
h_s, w_s	Feature plane size at spatial scale s	PTAM Training
l	Window size	PTAM Training
$\mathcal{W}_1, \mathcal{W}_2$	Two complementary window partitioning strategies	PTAM Training
$w_{m',n'}$	Window with top-left corner at position (m', n')	PTAM Training
$w_{p',q'}$	Window with top-left corner at position (p', q')	PTAM Training
N_{window}	Total number of windows	PTAM Training
F_j^{gt}	Ground truth features	PTAM Training
$\cos(\cdot, \cdot)$	Cosine similarity between predictions and ground truth features	PTAM Training
$\mathcal{L}_{inter}^s, \mathcal{L}_{final}^s$	Loss functions for intermediate and final predictions at scale s	PTAM Training
$\mathcal{L}_{temporal}$	Total temporal alignment loss computed across all three scales	PTAM Training

Table 3. Notation Table for DATA Paper - Part III: Progressive Temporal Alignment Module

Symbol	Definition	Section/Module
3.4 Instance-focused Feature Aggregation Module (IFAM)		
H_a	BEV feature of a-th agent, $H_a \in \mathbb{R}^{C\times H\times W}$	IFAM
M_a	Foreground mask, $M_a = \Phi(H_a)$	IFAM
H_a^{fore}, H_a^{back}	Foreground and background features	IFAM
H_a^{enh}	Enhanced foreground features after structural convolution	IFAM
StructConv$(\cdot)$	Structural convolution with specialized convolutions	IFAM
H_a^{cat}	Concatenated features, $H_a^{cat} \in \mathbb{R}^{2C\times H\times W}$	IFAM
W_a^s, W_a^c	Spatial attention and channel attention	IFAM
W_a^{init}	Initial attention weights, $W_a^{init} = W_a^s \oplus W_a^c$	IFAM
W_a^{verif}	Verification weights	IFAM
CS	Channel shuffle operation	IFAM
GConv$(\cdot)$	Group convolution for independent weight generation	IFAM
H_a^{verif}	Verified foreground feature	IFAM
$H_a^{refined}$	Individually refined BEV features	IFAM
ϵ	Learnable parameter balancing contribution of background features	IFAM
4. Experiments and General Symbols		
$\text{AP}_{50}, \text{AP}_{70}$	Average Precision at IoU thresholds of 0.50 and 0.70	Evaluation Metrics
IoU	Intersection-over-Union	Evaluation Metrics
μ_{pos}, σ_{pos}	Mean and standard deviation for position noise	Noise Modeling
μ_{rot}, σ_{rot}	Mean and standard deviation for orientation noise	Noise Modeling
σ_{local}	Standard deviation for localization noise	Robustness Testing
σ_{head}	Standard deviation for heading noise	Robustness Testing
$\odot$	Element-wise product	General Operations
$\oplus$	Concatenation operation	General Operations
$[\cdot]$	Concatenation along specified dimension	General Operations
$\lvert\cdot\rvert$	Cardinality of a set	General Operations
softmax$(\cdot)$	Softmax function	Activation Functions
ReLU$(\cdot)$	ReLU activation function	Activation Functions
Sigmoid$(\cdot)$	Sigmoid function	Activation Functions
MLP$(\cdot)$	Multi-layer perceptron	Network Structures
$\text{Conv}_{1\times1}(\cdot)$	1×1 convolution operation	Network Structures

Table 4. Notation Table for DATA Paper - Part IV: IFAM and General Symbols