CoopDiff: A Diffusion-Guided Approach for Cooperation under Corruptions

Supplementary Material

Algorithm 1 CoopDiff - Diffusion Training

Inputs: Ego feature F_i, Collaborator features $\{F_j\}_{j\neq i}$, Max timestep T
$x_0 \leftarrow D^{tea}(\{F_j\}_{j\in N})$ ▷ Clean target feature
$t \sim \mathcal{U}\{1,\dots,T\}$; $\epsilon \sim \mathcal{N}(0,I)$ ▷ Sample time & noise
$x_t \leftarrow \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$ ▷ Forward process
Build conditions:
  $c^{loc} \leftarrow \Lambda_{loc}(F_i)$ ▷ LE generates local condition
  $c^{coop} \leftarrow \Lambda_{coop}(F_i, \{F_j\}_{j\neq i}, c^{loc})$ ▷ CE generates cooperative condition
$\hat{\epsilon}_t \leftarrow U_\theta^{stu}(x_t, t, [c^{loc}, c^{coop}])$ ▷ Denoising process
Loss $\leftarrow \mathcal{L}_{diff}(\hat{\epsilon}_t, \epsilon)$ ▷ Compute loss
Update: $\theta \leftarrow \theta - \eta\nabla_\theta L_{\text{diff}}$

Algorithm 2 CoopDiff - Diffusion Inference

Inputs: F_i, $\{F_j\}_{j\neq i}$, steps T
$x_T \sim \mathcal{N}(0,I)$
for $t = T, T-1, \dots, 1$ **do**
  $c^{loc} \leftarrow \Lambda_{loc}(F_i)$
  $c^{coop} \leftarrow \Lambda_{coop}(F_i, \{F_j\}_{j\neq i}, c^{loc})$
  $\hat{\epsilon}_t \leftarrow U_\theta^{stu}(x_t, t, [c^{loc}, c^{coop}])$
  $\mu_t \leftarrow \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\hat{\epsilon}_t\right)$
  $x_{t-1} \leftarrow \mu_t$
end for
return x_0

1. Implementation Details

More Training Details. To ensure fair comparison and robust generalization to unseen scenes, all methods are trained using only the original clean point clouds without introducing any artificial corruptions or noise perturbations. Each model is trained for 30 epochs on a single RTX 4090 GPU with a batch size of 2. We do not apply any post-training fine-tuning to any method (e.g., V2X-ViT, ERMVP, etc.), which may cause slight performance differences from the original papers but guarantees fairness and consistency across all reproduced results. During training, the composite-loss coefficients $\alpha, \beta, \gamma, \delta$ are set to 1, 2, 1, and 1, respectively. $\mathcal{E}_{sem}$ is initialized as a convolutional projection that maps the one-hot classification labels into the feature space.

Diffusion Training. As detailed in **Algorithm 1**, the training process leverages the teacher-student paradigm introduced in Sec. 3.1. We first utilize the frozen Quality-Aware Teacher (D^{tea}) to process the multi-agent inputs and generate a clean target feature map x_0. We then apply the standard DDPM forward process: a timestep $t \sim \mathcal{U}\{1,\dots,T\}$ and Gaussian noise $\epsilon \sim \mathcal{N}(0,I)$ are sampled, corrupting the target into a noisy latent $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon$. The core task for the Student (U_θ^{stu}) is to predict the original noise $\hat{\epsilon}_t$. To achieve this, it is strongly conditioned on the outputs of its Dual-Branch Encoder (Sec. 3.2): the local condition $c^{\text{loc}} = \Lambda_{\text{loc}}(F_i)$ and the cooperative condition $c^{\text{coop}} = \Lambda_{\text{coop}}(F_i, \{F_j\}_{j\neq i}, c^{\text{loc}})$. The student's parameters θ are then updated by minimizing the diffusion loss $\mathcal{L}_{\text{diff}}(\hat{\epsilon}_t, \epsilon)$, while the teacher D^{tea} remains frozen.

Diffusion Inference. The inference process, detailed in **Algorithm 2**, is a generative, iterative refinement that reverses the diffusion. As noted in the main paper (Sec. 4.1), we employ a deterministic DDIM sampler for efficient and high-quality feature reconstruction. Starting with pure Gaussian noise $x_T \sim \mathcal{N}(0,I)$, the model iterates backward from $t = T$ to 1. At *each* step t, the dual-branch conditions c^{loc} and c^{coop} are recomputed from the input features to guide the denoising trajectory. The student U_θ^{stu} predicts the noise $\hat{\epsilon}_t$ given the current state x_t and the conditions. This prediction is used to compute the reverse mean μ_t (defined in Eq. 1), which estimates the "clean" feature x_0. Following the DDIM update rule, the next state is set deterministically: $x_{t-1} \leftarrow \mu_t$. After the final step, the resulting x_0 is returned as the fully denoised feature map used for downstream perception tasks.

2. Dataset Construction

We augment the original DAIR-V2X and OPV2V datasets by introducing various levels of different types of corruptions to the point cloud data using Robo3D [1], simulating the diverse degradation scenarios encountered in real-world environments. In this experiment, we fix the random seed to 42 to ensure that the added corruptions are consistent and reproducible across runs. Specifically, the steps for adding point cloud corruptions are as follows. Relevant visual examples can be referred to in Figure 1.

Fog. To reproduce atmospheric attenuation and backscattering effects of fog, we implement a two-stage simulation (see Fig. 1(2)). First, *hard-target attenuation* is applied following the Beer–Lambert law:

$$I = I_0 \exp(-2\alpha_f r_0),$$

where I_0 is the original intensity, r_0 is the range, and α_f denotes the fog attenuation coefficient. Second, *backscattering* is simulated using a precomputed lookup table that

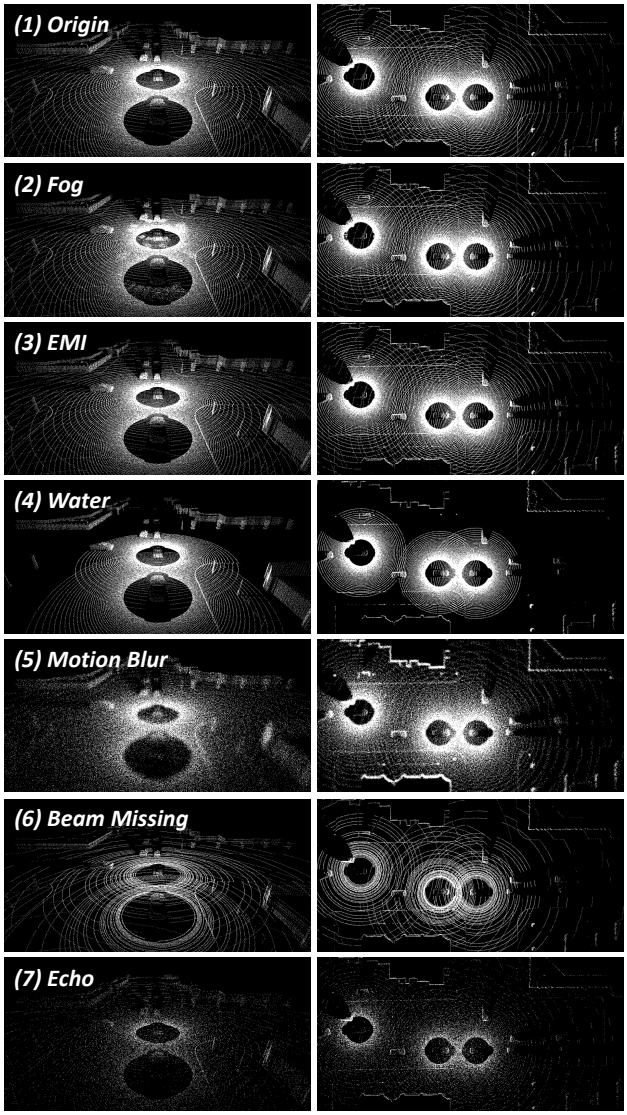

Figure 1. Visualization of the synthetic point cloud corruptions used in our benchmark. (1) Original clean data. (2) Fog, simulating atmospheric attenuation and backscattering. (3) EMI (Electromagnetic Interference). (4) Water, simulating surface reflection and absorption. (5) Motion Blur, from sensor motion. (6) Beam Missing, simulating partial sensor failure. (7) Echo, simulating incomplete returns from elevated surfaces.

provides a fog response intensity $I_{\text{fog}}(r_0)$ scaled by a scattering factor β_f. If $I_{\text{fog}}(r_0) > I$, the corresponding point is replaced by a synthetic fog point whose distance is adjusted to the maximal backscatter position. Otherwise, the original point is preserved. An additional positional noise of strength ν_f is finally applied to the fog points to introduce further uncertainty. In our experiments, these parameters are set to $\alpha_f = 0.02$, $\beta_f = 0.008$, and $\nu_f = 10$.

Electromagnetic Interference (EMI). To simulate *Electromagnetic Interference*, which introduces random sensor cross-talk and severe measurement deviations (see Fig. 1(3)), a small proportion γ_e of points is randomly selected from the cloud. For each selected point, additive Gaussian noise is applied to both spatial coordinates (x, y, z) and intensity i:

$$(x', y', z', i') = (x, y, z, i) + \eta, \quad \eta \sim \mathcal{N}(0, (3\sigma_e)^2),$$

where σ_e denotes the standard deviation of the underlying sensor noise. This corruption simulates rare but significant deviations caused by EMI without removing any points from the scene. In our experiments, we set the selection proportion $\gamma_e = 0.01$.

Water. To simulate *Water Accumulation* on the road surface (see Fig. 1(4)), a planar ground model is first fitted using RANSAC regression. If sufficient ground points are detected ($N_g > N_{th}$), where N_{th} is a predefined threshold, their incidence angles and original intensities are used to estimate a baseline reflectance model. The reflection of each ground point is then recalculated based on Fresnel equations to account for refraction and reflection through a thin water layer of height h_w. The new intensity I_w is computed as a weighted combination of the dry-ground reflectance I_0 and the transmission coefficient $T(h_w)$:

$$I_w = I_0 \cdot T(h_w).$$

Points whose I_w fall below an adaptive noise threshold τ_w are discarded to emulate absorption and scattering by the water surface. Remaining ground points are merged with unmodified non-ground points to form the final water-enhanced point cloud. In our experiments, we set the ground point threshold $N_{th} = 1000$ and the water layer height $h_w = 0.0015$.

Motion Blur. To simulate *Motion Blur* induced by rapid sensor motion or vibration, Gaussian perturbations are applied independently to each point (see Fig.1(5)). Specifically, for each point with coordinates (x, y, z), a random displacement $(\Delta x, \Delta y, \Delta z)$ is drawn from a zero-mean normal distribution with standard deviation σ_b, i.e.,

$$\Delta x, \Delta y, \Delta z \sim \mathcal{N}(0, \sigma_b^2).$$

The perturbed coordinates $(x', y', z') = (x + \Delta x, y + \Delta y, z + \Delta z)$ yield the blurred point cloud. In our experiments, we set the standard deviation $\sigma_b = 0.2$.

Beam Missing. To evaluate model robustness under partial sensor failure, we simulate *Beam Missing* noise, which emulates random malfunction of certain LiDAR beams in

an N_L-line scanner (see Fig. 1(6)). The raw point cloud is first processed to assign each point a ring identifier $r \in [0, N_L - 1]$ according to its vertical incidence angle. A fixed fraction α_b of beam IDs is randomly selected to represent inactive channels. All points whose ring ID belongs to this inactive set are removed. If the resulting point cloud becomes empty, at least one point is randomly retained to ensure non-null output. In our experiments, we simulate a 64-line scanner ($N_L = 64$) and set the missing fraction $\alpha_b = 0.5$.

Echo. The *Incomplete Echo* corruption (see Fig. 1(7)) simulates missing returns from elevated surfaces. All points with vertical coordinate $z > h_e$ (with a height threshold h_e) are identified as non-ground points. A proportion β_e of these are randomly removed, while all points with $z \leq h_e$ are retained. The resulting cloud thus preserves all ground points and only a $(1 - \beta_e)$ fraction of non-ground returns, imitating partial echo loss. In our experiments, we set the height threshold $h_e = 0.001$ and the removal proportion $\beta_e = 0.9$.

References

[1] Lingdong Kong, Youquan Liu, Xin Li, Runnan Chen, Wenwei Zhang, Jiawei Ren, Liang Pan, Kai Chen, and Ziwei Liu. Robo3d: Towards robust and reliable 3d perception against corruptions. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19994–20006, 2023. 1