x^2-Fusion: Cross-Modality and Cross-Dimension Flow Estimation in Event Edge Space — Supplementary Materials —

Ruishan Guo[1,*], Ciyu Ruan[1,*], Haoyang Wang[1,*], Zihang Gong[2], Jingao Xu[3], Xinlei Chen[1,† *]
[1]Shenzhen International Graduate School, Tsinghua University, [2]Harbin Institute of Technology,
[3]The University of Hong Kong

Abstract

This is the supplementary material for x^2-Fusion: Cross-Modality and Cross-Dimension Flow Estimation in Event Edge Space.

We provide the following materials in this manuscript:

- *Details of data processing and encoder architectures.*
- *The complete definition of Event Edge Strength.*
- *Realistic degradation modeling on EKubric and DSEC.*
- *More quantitative comparisons.*
- *More comparisons of scene flow estimation.*
- *Future works.*

1. Data Processing and Encoder Architectures

Our framework processes raw event streams, RGB images, and LiDAR point clouds, and builds hierarchical, modality-specific feature pyramids that are later projected into the shared Event Edge Space.

1.1. Event voxelization and encoder

Given an event stream $\mathcal{E} = \{(x_i, y_i, t_i, p_i)\}_{i=1}^{N}$ within a window $[t_0, t_0+T]$, we discretize time into B bins with centers $\{\mu_b\}_{b=1}^{B}$ and width $\Delta_t = T/B$, and accumulate events into a voxel tensor $\mathcal{V} \in \mathbb{R}^{B\times H\times W}$:

$$\mathcal{V}(b, y, x) = \textstyle\sum_i p_i \cdot k_b(t_i) \cdot \delta(x_i - x, y_i - y), \quad (1)$$

where $k_b(t_i) = \max(0, 1 - |t_i - \mu_b|/\Delta_t)$ is the linear interpolation kernel, $\delta(\cdot)$ is an indicator assigning each event to its pixel. Eq. (1) performs temporal bilinear splatting and preserves the spatiotemporal continuity while converting asynchronous events to a structured (b, y, x) volume suitable for convolution.

The voxelized tensor is then processed by a sparse 3D CNN (3D Conv–BN–ReLU blocks with hierarchical strides) to produce a multi-scale event pyramid: $\{E_l\}_{l=0}^{L}$, where $E_l \in \mathbb{R}^{B_l\times H_l\times W_l\times C_l^E}$. We then pool along the temporal-bin axis (average pooling unless noted) to obtain 2D feature maps

$$F_l^E = \mathrm{Pool}_t(E_l) \in \mathbb{R}^{H_l\times W_l\times C_l^E}, \quad (2)$$

which serve as the event embeddings used throughout the main paper (the encoder pretraining objective is described in Sec. 3.1.1). The definition and computation of the *event edge strength* used for weighting and supervision are detailed next in Sec. 2.

1.2. Image encoder (2D)

For RGB images, we form an image tensor by concatenating two frames and edge-promoting cues:

$$X^I = [\, I_t,\ I_{t+\Delta t},\ I_{t+\Delta t} - I_t,\ |\nabla I_t|,\ |\nabla I_{t+\Delta t}|\,], \quad (3)$$

where $|\nabla(\cdot)|$ denotes Sobel gradient magnitude.

After per-channel normalization, we normalize X^I and feed it to a 2D CNN built from residual Conv–BN–ReLU units with hierarchical strides. The encoder outputs a multi-scale pyramid $\{F_l^I\}_{l=0}^{L}$, where $F_l^I \in \mathbb{R}^{H_l\times W_l\times C_l^{2D}}$. The pyramid preserves photometric appearance while exposing temporal/edge cues via the difference and gradient channels. These features are used for alignment in Event Edge Space (after projection) and as inputs to the 2D task head.

1.3. LiDAR encoder (3D)

Given a raw point cloud $\mathcal{P} = \{p_i\}_{i=1}^{N}$, $p_i = (x_i, y_i, z_i)^\top$, we build a hierarchical pyramid of point sets $\{\mathcal{P}^l\}_{l=0}^{L}$ using farthest-point sampling (FPS), with decreasing density as l increases. For each level l, we extract local geometric features with PointConv [1] over k-NN neighborhoods $\mathcal{N}(i)$:

$$\mathbf{F}_i^l = \Phi_2\big(\textstyle\sum_{j\in\mathcal{N}(i)} w(\Delta p_{ij}) \ \odot\ \Phi_1(\Delta p_{ij},\ \mathbf{F}_j^{l-1})\big), \quad (4)$$

where Φ_1, Φ_2 are shared MLPs , $\Delta p_{ij} = p_j^l - p_i^l$ is the relative offset and $w(\cdot)$ is a learned spatial weighting.

[*][*]Equal Contribution. [†]Corresponding author.

To explicitly encode edge geometry, we augment each point with depth-contrast and curvature descriptors computed from its neighborhood:

$$\delta_i^z = \mathrm{mean}_{j\in\mathcal{N}(i)} |z_j^l - z_i^l|, \quad \kappa_i = \frac{\lambda_0}{\lambda_0+\lambda_1+\lambda_2}, \quad (5)$$

where $\lambda_0 \le \lambda_1 \le \lambda_2$ are eigenvalues of the local covariance $C_i = \sum_{j\in\mathcal{N}(i)} (\Delta p_{ij})(\Delta p_{ij})^\top$. We concatenate $[\delta_i^z, \kappa_i]$ to the point features at each stage before the next PointConv. This yields a multi-scale 3D feature pyramid $\{F_l^L\}_{l=0}^L$ with $F_l^L \in \mathbb{R}^{N_l \times C_l^{3D}}$ that is sensitive to geometric discontinuities. These features are later projected into Event Edge Space (via the LiDAR head) and fed to the 3D task head.

1.4. Projection heads

At each scale l, lightweight heads map modality-specific channels to the common embedding dimension C_l used in Event Edge Space:

$$Z_l^I = h_l^I(F_l^I), \qquad Z_l^L = h_l^L(F_l^L), \qquad Z_l^E \equiv F_l^E, \quad (6)$$

where h_l^I is a 1×1 Conv (with BN+ReLU) and h_l^L is a shared MLP applied point-wise. Event features Z_l^E come from the pretrained event encoder and are frozen during multimodal training.

2. Event Edge Strength

To explicitly bias F_s^E towards edge-aware representations, we define an event edge strength from raw events. For each pixel (x, y) in a window, we measure the normalized event activity $\tilde{A}^E(x, y)$ and the normalized temporal variance $\tilde{\sigma}_t(x, y)$ of its events, and combine them into

$$e^E(x, y) = \tilde{A}^E(x, y)\,\big(1 - \tilde{\sigma}_t(x, y)\big) \in [0, 1], \quad (7)$$

where larger e^E indicates stronger and more temporally coherent motion edges.

Concretely, let $\{(x_k, y_k, t_k, p_k)\}_k$ be the events within a temporal window $[t_0, t_0 + T]$.For each pixel (x, y), we first count the number of events falling onto it:

$$A^E(x, y) = \big|\{k \mid (x_k, y_k) = (x, y)\}\big|, \quad (8)$$

and then normalize this activity by the spatial mean

$$\tilde{A}^E(x, y) = \frac{A^E(x, y)}{\frac{1}{HW}\sum_{u,v} A^E(u, v) + \varepsilon}, \quad (9)$$

where H, W are the image height and width, and ε is a small constant for numerical stability.

For temporal variance, we normalize the timestamps of events at (x, y) as $\hat{t}_k = (t_k - t_0)/T \in [0, 1]$ and define

$$\bar{t}(x, y) = \frac{1}{N^E(x, y)} \sum_{k:(x_k,y_k)=(x,y)} \hat{t}_k, \quad (10)$$

$$\sigma_t(x, y) = \sqrt{\frac{1}{N^E(x, y)} \sum_{k:(x_k,y_k)=(x,y)} \big(\hat{t}_k - \bar{t}(x, y)\big)^2}, \quad (11)$$

$$\tilde{\sigma}_t(x, y) = \frac{\sigma_t(x, y)}{0.5 + \varepsilon}, \quad (12)$$

where the normalization by 0.5 maps $\sigma_t(x, y)$ to $[0, 1]$ (the maximum standard deviation of a distribution on $[0, 1]$ is 0.5). Thus $\tilde{\sigma}_t(x, y)$ is small when events at (x, y) are temporally coherent, and large when they are scattered over the window. Scale-specific edge maps e_s^E are obtained by spatial pooling of e^E to the resolution (H_s, W_s).

3. Realistic Degradation Modeling on EKubric and DSEC

We develop a physically-grounded degradation framework to systematically evaluate optical and scene flow estimation robustness under adverse conditions, applying four distinct degradation types[2, 3] to both synthetic (EKubric) and real-world (DSEC) datasets. The framework preserves ground-truth annotations while systematically introducing controlled perturbations to image and LiDAR modalities, with representative visualizations provided in Fig. 1.

For RGB images, We formulate the exposure-based degradation through a two-stage brightness transformation that explicitly controls both the target illumination level and valid intensity range. Firstly, we normalize the input image intensities to a standard range:

$$I_{norm} = \frac{I_{raw} - I_{min}}{I_{max} - I_{min}}, \quad (13)$$

where I_{min} and I_{max} represent the minimum and maximum intensity values in the original image. This normalization ensures consistent processing across different input dynamic ranges. The brightness adjustment then applies a linear scaling based on the ratio between the target mean brightness μ_t and the original mean μ_0, followed by a nonlinear clipping operation to maintain physically plausible intensity values:

$$\begin{aligned} I_{deg} &= \mathcal{T}(I_{raw}; \mu_t, \tau_{min}, \tau_{max}) \\ &= \min(\max(I_{norm} \cdot \frac{\mu_t}{\mu_0}, \tau_{min}), \tau_{max}) \end{aligned} \quad (14)$$

where μ_t and μ_0 represent the target and original mean brightness respectively, while τ_{min} and τ_{max} define the lower and upper intensity bounds. The transformation naturally progresses from initial normalization to brightness scaling, with the final clipping operation accounting for sensor saturation effects. For under-exposure simulation, we set $\mu_t = 0.03$ with a minimum threshold $\tau_{min} = 0.02$ to maintain faint details while avoiding complete darkness, while over-exposure uses $\mu_t = 0.98$ with $\tau_{max} = 1.0$ to

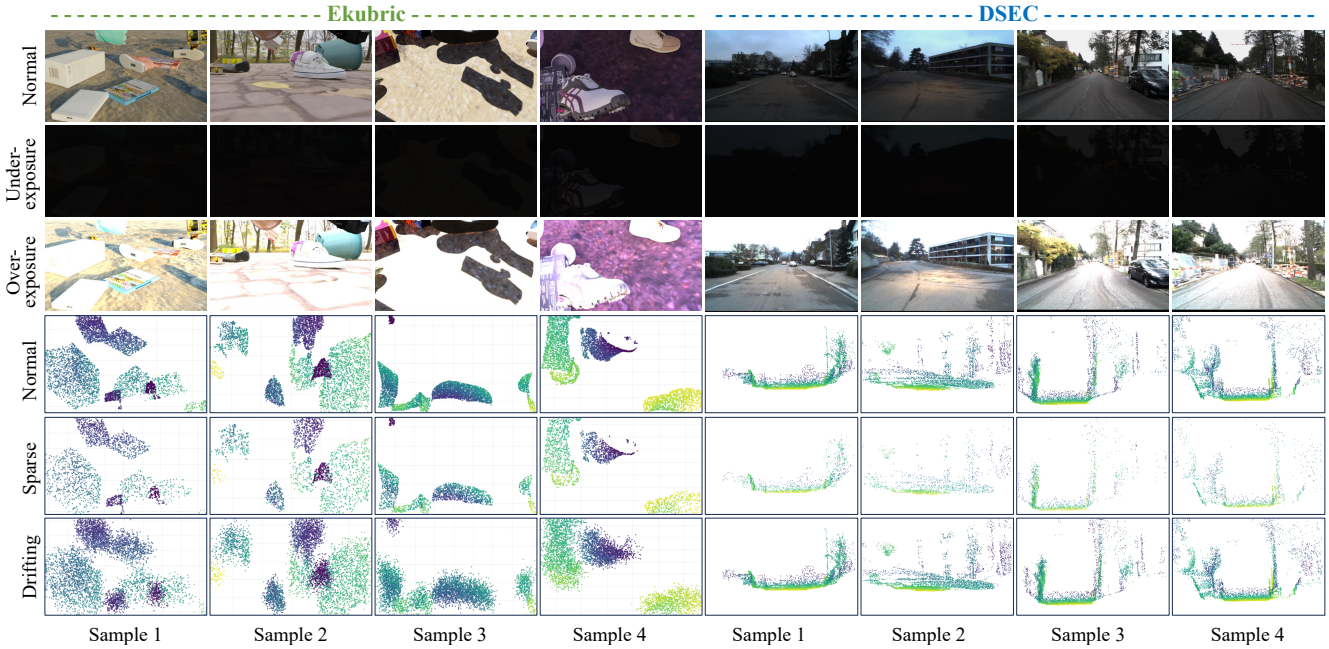

Figure 1. Representative samples of the degradation of the EKubric and DSEC dataset.

properly model highlight clipping. These parameters were carefully selected through empirical analysis of real camera responses under extreme lighting conditions[4].

For LiDAR point clouds, we implement two geometrically-motivated degradation operations. To maintain the required point count N_{max} = 8192 for network input, the degradation process is formulated as:

$$\mathcal{P}_{sparse} = p\lfloor i/r \rfloor_{i=1}^{\lfloor rN \rfloor}, \quad r = 0.25 \tag{15}$$

where $N = |\mathcal{P}|$ is the original point count and r is the downsampling ratio. For network input requiring exactly 8192 points, we perform random sampling with replacement when $\lfloor rN \rfloor < 8192$.

For the drifting degradation, we augment the sparse point cloud with positional noise to simulate measurement inaccuracies:

$$\mathcal{P}_{drift} = \{p_i + \epsilon_i | p_i \in \mathcal{P}_{sparse}\}, \quad \epsilon_i \sim \mathcal{N}(0, \sigma^2 \mathbf{I}_3) \tag{16}$$

where $\sigma = 0.15$ m controls the noise magnitude and $\mathbf{I}_3$ is the 3D identity matrix. The Gaussian noise ϵ_i is independently sampled for each point and dimension, modeling real-world LiDAR measurement errors.

4. More Quantitative Comparisons

Owing to space constraints in the main text, we provide complete quantitative evaluations of optical flow estimation performance under both normal operating conditions and extreme scenarios. This comprehensive comparison includes specialized testing for modality-specific degradation cases, where we exclusively evaluated methods that utilize the affected modality as input.

Notably, our approach attains state-of-the-art optical/scene flow accuracy across all test scenarios and remains robust under modality degradations.

In addition, our experiments reveal a critical cross-modal dependency in joint 2D optical flow and 3D scene flow estimation using RGB cameras and LiDAR. Specifically, degradation in one modality (e.g., sparse or noisy LiDAR point clouds) significantly compromises the estimation accuracy of the other modality (e.g., 2D optical flow), as observed in baseline methods like CamLiFlow. For instance, when LiDAR point clouds suffer from drift-induced degradation, CamLiFlow exhibits a notable performance drop in 2D flow prediction (reducing by 92.73% in EPE_{2D} (on DSEC dataset), see Tab. 1), highlighting the vulnerability of non-adaptive fusion frameworks to modality-specific perturbations. In contrast, our method mitigates this cascading degradation effect through adaptive feature fusion, which dynamically reweights multi-modal representations based on their reliability. This mechanism ensures robust performance across all feature dimensions (2D/3D), even under partial modality degradation. The results demonstrate that our model reduces the performance gap caused by LiDAR degradation by 67.27% compared to CamLiFlow, validating the effectiveness of our fusion strategy in preserving task coherence under real-world sensor imperfections.

Modality	Method	#Params(M)	EKubric						DSEC					
			EPE_{2D} ↓	ACC_{1px} ↑	Fl ↓	EPE_{3D} ↓	$ACC_{.05}$ ↑	$ACC_{.10}$ ↑	EPE_{2D} ↓	ACC_{1px} ↑	Fl ↓	EPE_{3D} ↓	$ACC_{.05}$ ↑	$ACC_{.10}$ ↑
Img	RAFT [5]	5.3M	0.838	93.31%	2.36%	-	-	-	0.586	88.98%	1.47%	-	-	-
Img	FlowFormer [6]	16.2M	0.702	92.58%	2.07%	-	-	-	0.567	89.82%	1.33%	-	-	-
PC	PV-RAFT [7]	239.2K	-	-	-	0.093	82.42%	92.60%	-	-	-	0.190	32.74%	55.62%
EV	E-RAFT [8]	5.3M	-	-	-	-	-	-	0.481	91.75%	1.31%	-	-	-
Img+Depth	RAFT-3D [9]	7.7M	0.714	94.39%	-	0.049	92.88%	-	0.572	89.55%	-	0.144	49.30%	-
Img+PC	CamLiFlow [10]	7.7M	0.770	95.11%	1.80%	0.035	94.90%	95.86%	0.399	94.94%	1.33%	0.129	51.08%	68.17%
Img+EV	DCEIFlow [11]	7.1M	3.109	56.37%	18.40%	-	-	-	0.970	73.01%	4.90%	-	-	-
Img+PC+EV	†VisMoFlow [12]	-	-	-	-	0.026	95.98%	96.77%	-	-	-	0.100	61.78%	75.82%
Img+PC+EV	RPEFlow [13]	9.8M	0.439	95.99%	1.48%	0.027	95.33%	96.32%	0.326	95.28%	1.15%	0.103	60.80%	74.97%
Img+PC+EV	x^2-Fusion (Ours)	8.2M	**0.430**	**96.86%**	**1.43%**	**0.024**	**96.78%**	**97.62%**	**0.322**	**95.80%**	**1.13%**	**0.094**	**64.39%**	**78.27%**
			EKubric-Img (Under-Exposure Degradation)						DSEC-Img (Under-Exposure Degradation)					
Img	RAFT [5]		8.762	26.24%	37.26%	-	-	-	3.045	46.96%	24.40%	-	-	-
Img+Depth	RAFT-3D [9]		-	-	-	-	-	-	5.960	12.59%	49.45%	0.790	0.79%	2.18%
Img+PC	CamLiFlow [10]		5.637	25.52%	38.19%	0.052	90.10%	94.65%	1.193	75.20%	7.80%	0.137	46.11%	65.50%
Img+EV	DCEIFlow [11]		8.634	24.52%	47.68%	-	-	-	1.447	52.30%	8.51%	-	-	-
Img+PC+EV	RPEFlow [13]		3.663	33.96%	27.06%	0.043	89.01%	92.27%	0.817	83.24%	5.06%	0.117	54.97%	69.94%
Img+PC+EV	x^2-Fusion (Ours)		**1.143**	**68.87%**	**9.11%**	**0.033**	**94.14%**	**95.31%**	**0.394**	**92.73%**	**1.14%**	**0.107**	**58.87%**	**71.43%**
	Absolute Improvement (vs. RPEFlow)		(↓ 2.520)	(↑ 34.91%)	(↓ 17.95%)	(↓ 0.010)	(↑ 5.13%)	(↑ 3.04%)	(↓ 0.423)	(↑ 9.49%)	(↓ 3.92%)	(↓ 0.010)	(↑ 3.90%)	(↑ 1.49%)
			EKubric-Img (Over-Exposure Degradation)						DSEC-Img (Over-Exposure Degradation)					
Img	RAFT [5]		7.360	27.37%	20.44%	-	-	-	1.038	78.30%	4.18%	-	-	-
Img+Depth	RAFT-3D [9]		-	-	-	-	-	-	2.320	43.67%	17.10%	0.78	1.11%	2.52%
Img+PC	CamLiFlow [10]		4.676	41.00%	27.62%	0.048	91.54%	94.91%	1.072	78.32%	6.44%	0.136	46.96%	65.97%
Img+EV	DCEIFlow [11]		8.450	25.11%	46.86%	-	-	-	1.475	50.75%	8.88%	-	-	-
Img+PC+EV	RPEFlow [13]		2.801	53.75%	17.63%	0.039	91.32%	95.85%	0.565	89.59%	2.73%	0.109	54.91%	70.02%
Img+PC+EV	x^2-Fusion (Ours)		**0.994**	**80.19%**	**7.05%**	**0.032**	**93.86%**	**95.93%**	**0.376**	**93.03%**	**1.13%**	**0.107**	**58.67%**	**71.09%**
	Absolute Improvement (vs. RPEFlow)		(↓ 1.807)	(↑ 26.44%)	(↓ 10.58%)	(↓ 0.007)	(↑ 2.54%)	(↑ 0.08%)	(↓ 0.189)	(↑ 3.44%)	(↓ 1.6%)	(↓ 0.002)	(↑ 3.76%)	(↑ 1.07%)
			EKubric-PC (Sparse Degradation)						DSEC-PC (Sparse Degradation)					
PC	PV-RAFT [7]		-	-	-	0.096	79.71%	91.68%	-	-	-	**0.036**	81.70%	**97.08%**
Img+PC	CamLiFlow [10]		0.783	95.03%	1.85%	0.035	94.91%	95.87%	8.311	7.05%	68.95%	0.100	48.75%	69.68%
Img+PC+EV	RPEFlow [13]		0.569	94.62%	2.02%	0.027	95.37%	96.34%	0.493	90.36%	1.19%	0.056	81.18%	88.69%
Img+PC+EV	x^2-Fusion (Ours)		**0.439**	**95.93%**	**1.49%**	**0.022**	**96.99%**	**97.43%**	**0.331**	**94.49%**	**1.18%**	**0.047**	**86.40%**	**90.31%**
	Absolute Improvement (vs. RPEFlow)		(↓ 0.130)	(↑ 1.31%)	(↓ 0.53%)	(↓ 0.005)	(↑ 1.62%)	(↑ 1.09%)	(↓ 0.162)	(↑ 4.13%)	(↓ 0.01%)	(↓ 0.009)	(↑ 5.22%)	(↑ 1.62%)
			EKubric-PC (Drifting Degradation)						DSEC-PC (Drifting Degradation)					
PC	PV-RAFT [7]		-	-	-	0.168	18.70%	54.27%	-	-	-	0.457	0.24%	1.83%
Img+PC	CamLiFlow [10]		1.949	59.37%	10.25%	0.163	52.64%	73.79%	0.769	84.10%	2.30%	0.419	0.37%	2.47%
Img+PC+EV	RPEFlow [13]		1.164	62.44%	9.92%	0.113	66.47%	73.76%	1.051	68.05%	3.82%	0.457	0.51%	2.46%
Img+PC+EV	x^2-Fusion (Ours)		**0.763**	**88.52%**	**6.06%**	**0.088**	**79.34%**	**82.80%**	**0.409**	**91.39%**	**2.13%**	**0.285**	**19.36%**	**28.87%**
	Absolute Improvement (vs. RPEFlow)		(↓ 0.401)	(↑ 26.08%)	(↓ 3.86%)	(↓ 0.025)	(↑ 12.77%)	(↑ 9.04%)	(↓ 0.642)	(↑ 23.34%)	(↓ 1.69%)	(↓ 0.172)	(↑ 18.85%)	(↑ 26.41%)

Table 1. Extended quantitative evaluation on both standard and degraded scenarios of EKubric and DSEC datasets.

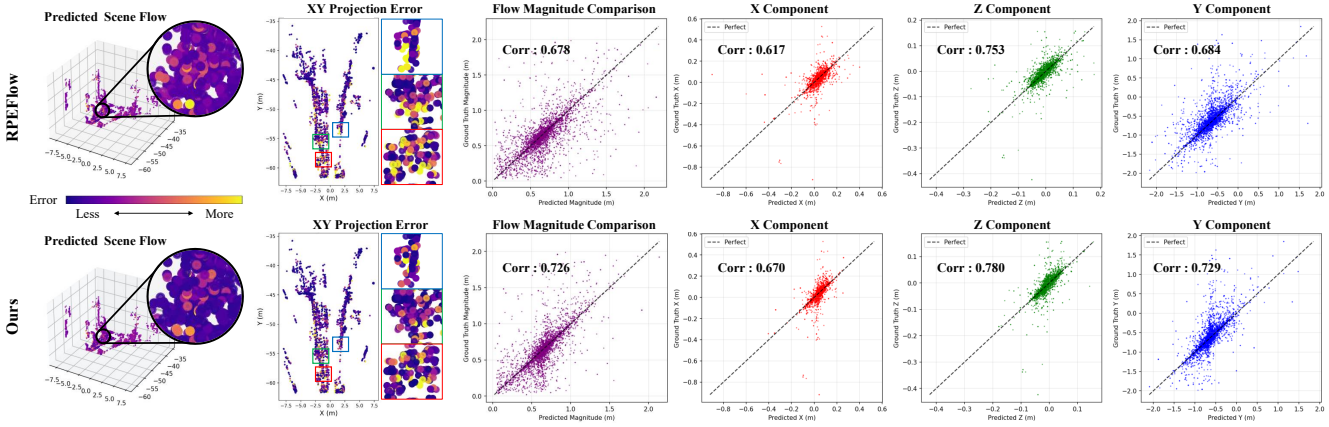

Figure 2. Extended comparisons: 3D scene flow estimation under normal conditions (DSEC). Col.1-2: End-point error for 3D point cloud and XY plane projections (brighter hues indicate larger errors). Col.3: Flow magnitude comparison. Col.4-6: X/Z/Y component comparisons between predicted and ground truth warped point clouds (diagonal indicates ideal matching).

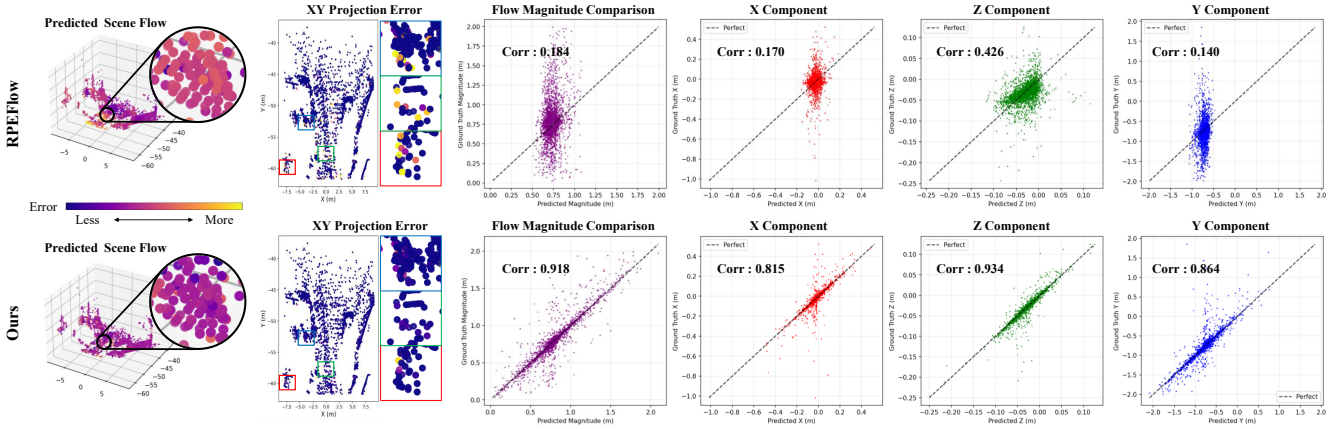

Figure 3. Extended comparisons: 3D scene flow estimation under sparse LiDAR (DSEC).

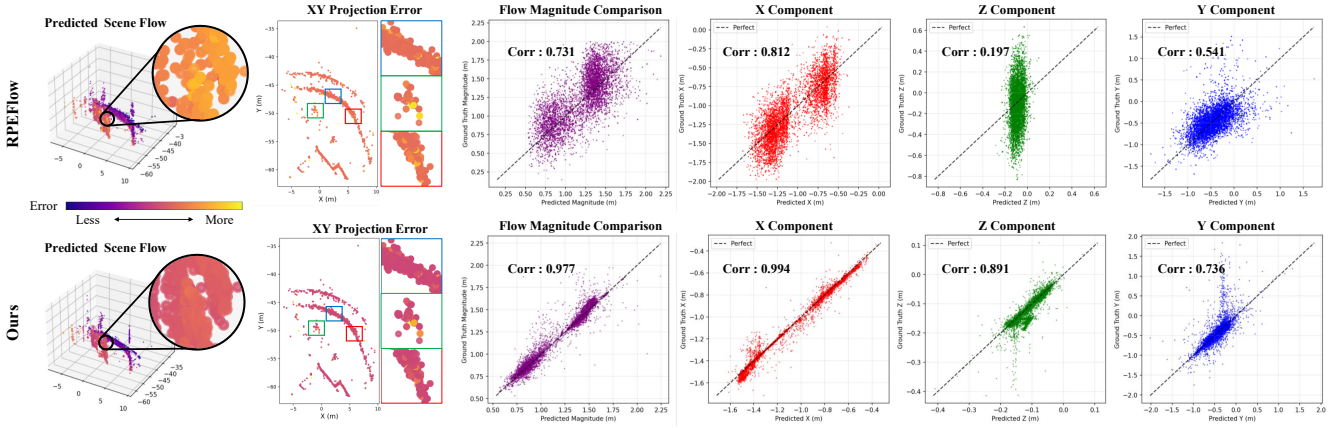

Figure 4. Extended comparisons: 3D scene flow estimation under drifting LiDAR (DSEC).

5. Extended Scene Flow Comparisons

Given that the 3D scene flow comprises four distinct dimensions (x, y, z, and magnitude), we conduct a comprehensive dimensional analysis beyond the conventional planar projection error metrics. Fig. 2-4 presents this multi-dimensional evaluation, where the horizontal and vertical axes represent the predicted scene flow and ground truth optical flow, respectively.

As shown in Fig. 2, our method attains accuracy comparable to RPEFlow (the current state of the art [13]) under normal operating conditions. The qualitative results in Figs. 3–4 further show superior convergence to ground-truth scene flow across all four dimensions (x, y, z, magnitude) under both sparsity and drift degradations, highlighting robust adaptability in challenging conditions.

6. Future Work

Although this study targets the image–LiDAR–event setting for flow estimation, EES is modality-agnostic. In future work, we intend to extend the *Event Edge Space* to additional sensors (e.g., thermal infrared, millimeter-wave radar, polarization cameras) by mapping their edge cues into the same latent space via lightweight projection heads, and evaluate generality beyond Image–Event-LiDAR through Event–X combinations and cross-dataset tests.

Additionally, we aim to explore self-supervised training approaches that leverage physics-based constraints such as IMU-guided motion consistency and neural radiance fields, thereby removing the reliance on LiDAR ground-truth annotations and enabling application in previously unseen environments.

Furthermore, to facilitate practical deployment, we will explore optimization strategies suitable for edge devices, including neural architecture search and quantization techniques, with the goal of achieving real-time performance on resource-constrained platforms like embedded GPUs while carefully balancing latency and memory requirements.

These efforts will extend x^2-Fusion's core innovations into a universal sensor-fusion paradigm, enhancing its scalability, generality, and practicality for real-world perception systems.

References

[1] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 9621–9630, 2019. 1

[2] Guoqiang Liang, Kanghao Chen, Hangyu Li, Yunfan Lu, and Lin Wang. Towards robust event-guided low-light image enhancement: a large-scale real-world event-image dataset and novel approach. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 23–33, 2024. 2

[3] Yinpeng Dong, Caixin Kang, Jinlai Zhang, Zijian Zhu, Yikai Wang, Xiao Yang, Hang Su, Xingxing Wei, and Jun Zhu. Benchmarking robustness of 3d object detection to common corruptions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1022–1032, 2023. 2

[4] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8934–8943, 2018. 3

[5] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European conference on computer vision*, pages 402–419. Springer, 2020. 4

[6] Zhaoyang Huang, Xiaoyu Shi, Chao Zhang, Qiang Wang, Ka Chun Cheung, Hongwei Qin, Jifeng Dai, and Hongsheng Li. Flowformer: A transformer architecture for optical flow. In *European conference on computer vision*, pages 668–685. Springer, 2022. 4

[7] Yi Wei, Ziyi Wang, Yongming Rao, Jiwen Lu, and Jie Zhou. Pv-raft: Point-voxel correlation fields for scene flow estimation of point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6954–6963, 2021. 4

[8] Mathias Gehrig, Mario Millhäusler, Daniel Gehrig, and Davide Scaramuzza. E-raft: Dense optical flow from event cameras. In *2021 International Conference on 3D Vision (3DV)*, pages 197–206. IEEE, 2021. 4

[9] Zachary Teed and Jia Deng. Raft-3d: Scene flow using rigid-motion embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8375–8384, 2021. 4

[10] Haisong Liu, Tao Lu, Yihui Xu, Jia Liu, Wenjie Li, and Lijun Chen. Camliflow: bidirectional camera-lidar fusion for joint optical flow and scene flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5791–5801, 2022. 4

[11] Zhexiong Wan, Yuchao Dai, and Yuxin Mao. Learning dense and continuous optical flow from an event camera. *IEEE Transactions on Image Processing*, 31:7237–7251, 2022. 4

[12] Hanyu Zhou, Yi Chang, and Zhiwei Shi. Bring event into rgb and lidar: Hierarchical visual-motion fusion for scene flow. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 26477–26486, 2024. 4

[13] Zhexiong Wan, Yuxin Mao, Jing Zhang, and Yuchao Dai. Rpeflow: Multimodal fusion of rgb-pointcloud-event for joint optical flow and scene flow estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10030–10040, 2023. 4, 5