RPEFlow: Multimodal Fusion of RGB-PointCloud-Event for Joint Optical Flow and Scene Flow Estimation — Supplementary Materials —

Zhexiong Wan[1] Yuxin Mao[1] Jing Zhang[2] Yuchao Dai[1†]
[1]Northwestern Polytechnical University & Shaanxi Key Laboratory of Information Acquisition and Processing [2]Australian National University

Abstract

In this supplementary material, we first show the simulation details and advantages of our contributed EKubric dataset, and illustrate the data processing and training details of the three datasets we use. Then we illustrate why we are minimizing the mutual information here instead of maximizing it from the background of mutual information, and describe the details of our proposed multi-stage framework. Finally, we provide more experimental results, including the analysis of the multiple trials' consistency and the difference between simulated and real-captured events.

1. Our Simulated EKubric Dataset

We use the kubric [1] simulator to simulate 15,367 RGB-PointCloud-Event pairs with rich annotations (including RGB images, depth images, point clouds and events data collected by simulated sensors, as well as optical flow, scene flow, surface normal, semantic segmentation and object coordinates ground truths), denoted as **EKubric** dataset.

Thanks to the kubric simulator that uses pybullet [2] to simulate object physics, our simulated EKubric dataset can handle kinematic collisions and simulate realistic object motion trajectories with a gravity model. In addition, thanks to the convenience of the Blender [3] simulator for adjustable scene lighting, we can simulate more realistic ambient illumination. We set different parameters such as the camera or object moving speed, illumination level, number of objects, background difficulty, and motion blur level, and simulated six sequences of standard, blur, complex, fast, static_camera, and static_objects. We simulate a total of 15,367 pairs of samples, and uniformly sample them according to different sequences, in which 12,287 pairs are used as training sets and 3080 pairs as testing sets.

The differences between the FlyingThings3D [4] dataset and our simulated EKubric dataset are shown in Table 1. We show some visualizations of comparing FlyingThings [4] and EKubric dataset in Fig. 1. We find that there are many clipping problems in the FlyingThings dataset, and there are a few kinds of objects and some of them lack textures. However, our simulated EKubric dataset is more realistic due to the introduction of collision detection, and there are more kinds of objects with richer textures. Furthermore, in Fig. 2, we show the 2D visualizations of a representative scene, with collision detection and gravity motion model in our simulated dataset. In addition to the data used for our joint optical flow and scene flow estimation task, we also provide semantic segmentation, object coordinates and surface normal maps. These additional data are simulated by the kubric

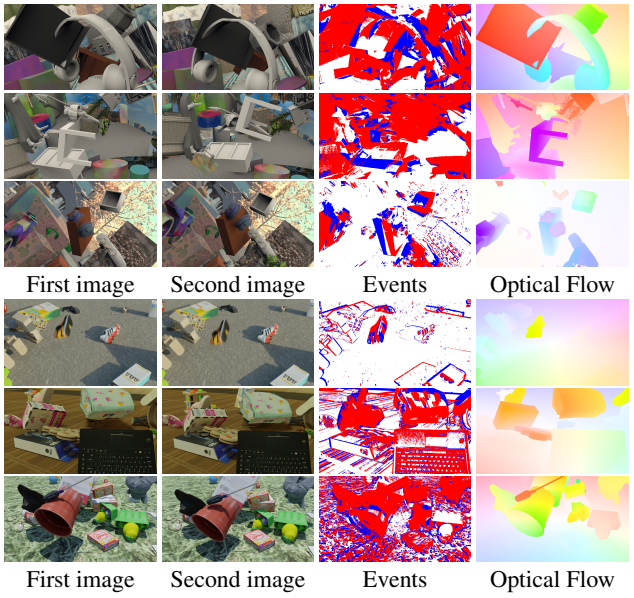

Figure 1: **Dataset visual comparisons** between the FlyingThings3D [4] dataset (top) and our simulated EKubric dataset (bottom). We find many clipping problems in FlyingThings3D, and the motion of EKubric is more realistic with collision detection.

	Sensors				Annotations					Realistic Simulation		
	RGB	Depth	Events	PCs	OF	SF	Seg	Normal	Coords	Collision	Gravity	Illumination
FlyingThings3D [4]	960 × 540		Post-Sim	✓	✓	✓	✓	×	×	×	×	limited
EKubric	1280 × 720		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 1: **Dataset comparisons** between the FlyingThings3D [4] dataset and our simulated EKubric dataset. Our proposed dataset provides more types of annotations while simulating more realistically.

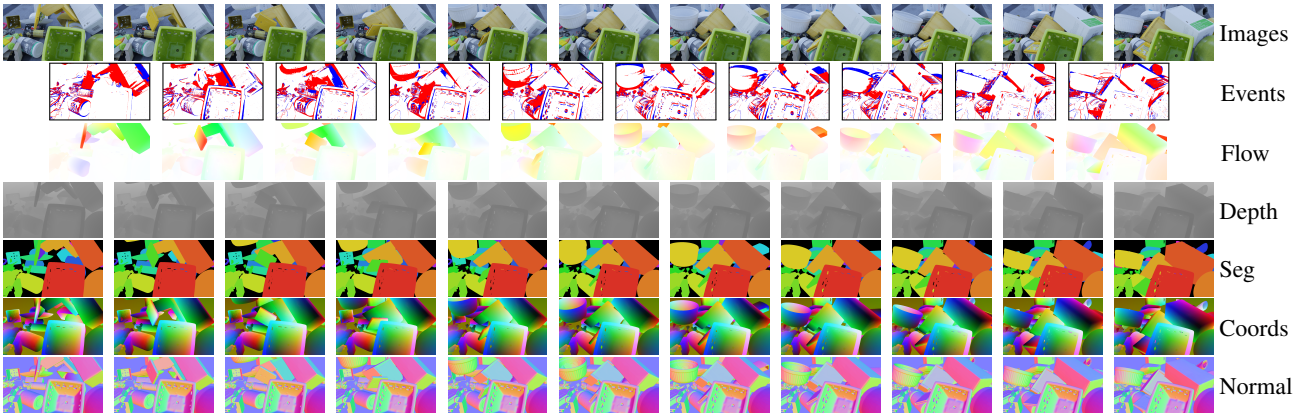

Figure 2: **Visualization of a sequence of consecutive frames in our simulated EKubric dataset.** Thanks to collision detection and gravity motion model, we can simulate realistic RGB images for multiple moving objects. At the same time, we can output events, optical flow, depth, semantic segmentation (Seg), object coordinates (Coord) and surface normal (Normal) maps.

simulator. We believe that these rich kinds of data can provide large-scale training data for event-based motion segmentation, 6-DOF pose estimation and other motion-related tasks, and our proposed dataset is of significant value to the event-based vision community.

2. Data Processing and Training Details

Because the datasets we use only provide depth or disparity maps, we follow the preprocess pipeline [5, 6] to convert them into 3D point clouds. For the FlyingThings3D dataset, we train 600 epochs with an initial learning rate of 4×10^{-4} and reduce by half at 400 and 500 epochs on the training set, then evaluate the trained model on the testing set.

For our simulated EKubric dataset, we use the semantic segmentation maps to distinguish the scene flow at foreground object locations and obtain occlusion maps by bidirectional optical flow check [7], so that we can report the scene flow accuracy for full foreground objects (Full) and unoccluded foreground objects (Non-Occ) on our simulated EKubric dataset as on the FlyingThings3D dataset. For the DSEC [8] dataset, since the official test set does not have depth or disparity annotations, we divide the official training set (8,170 frames) into a training part (6,747 frames) and a test part (1,423 frames) based on different sequences. After finetuning the divided training part, the results are evaluated on the test part. Then we take the same pipeline of converting disparity to point cloud as above, because it is inconvenient to filter out occlusion for real data, so only the full scene flow accuracy is reported. For the above two datasets, we use them to finetune the pre-trained model of FlyingThings3D. In these two finetuning stages, we train 300 epochs with an initial learning rate of 1×10^{-4} and reduce by half at 150 and 250 epochs on the training set.

3. Why Mutual Information Minimization?

Background: Let X and Y be a pair of random variables[1] with value over the space $\mathcal{X} \times \mathcal{Y} \to \mathbb{R}$, their mutual information $I(X;Y)$ is defined as:

$$I(X;Y) = \mathbb{E}_{p(x,y)} \left[\log \frac{p(x,y)}{p(x) \otimes p(y)} \right], \qquad (1)$$

which is the Kullback-Leibler divergence of the joint distribution $p_{(x,y)}$ from the the product of the marginal distributions $p(x)$ and $p(y)$. In other words, mutual information $I(X;Y)$ measures the information that X and Y share, which is typically used as a regularizer in loss function to encourage (via mutual information maximization)

[1] We use upper case to represent the variables and lower case to represent the samples.

or limit dependency (via mutual information minimization) between variables. As the log density ratio between the joint distribution $p(x, y)$ and product of marginals $p(x) \otimes p(y)$ is intractable, mutual information is usually estimated instead of computed directly, leading to both mutual information maximization with a lower bound and mutual information minimization with an upper bound.

BA Lower Bound[9]: According to the Bayesian's law, we have $p(x, y) = p(y|x)p(x)$, where $p(y|x)$ is the true conditional distribution, and can be approximated with a variational counterpart $q_\theta(y|x)$, with θ as the model parameter set. In this case, we obtain the mutual information lower bound introduced in [9], namely I_{ba} via:

$$\begin{aligned} I(X;Y) &= \mathbb{E}_{p(x,y)}[\log \frac{q_\theta(y|x)}{p(y)}] + D_{KL}(p(y|x)||q_\theta(y|x)) \\ &\geq \mathbb{E}_{p(x,y)}[\log q_\theta(y|x)] + H(Y) = I_{ba}, \end{aligned} \tag{2}$$

where $H(Y)$ is the differential entropy of Y, $q_\theta(y|x)$ is a predictor to predict y from x, which can be modeled with a deep neural network with parameters θ.

vCLUB Upper Bound [10]: Given X and Y with unknown conditional distribution $p(y|x)$ (or $p(x|y)$), [10] introduces contrastive log-ratio upper bound (CLUB) of mutual information between X and Y, which is defined as:

$$\begin{aligned} I_{vCLUB} &= \mathbb{E}_{p(x,y)}\left[\log q_\theta(y|x)\right] - \mathbb{E}_{p(x)}\mathbb{E}_{p(y)}\left[\log q_\theta(y|x)\right] \\ &\geq I(X;Y). \end{aligned} \tag{3}$$

Same as Eq. 2, $q_\theta(y|x)$ is a variational distribution with parameters θ to approximate the true distribution $p(y|x)$. As Eq. 3 involves sampling from both marginal distributions, leading to $\mathcal{O}(N^2)$ computational complexity, where N is the size of the samples. To accelerate the mutual information upper bound computation, [10] uniformly samples negative pairs $(x_i, y_{k'_i})$, treating its probability $q_\theta(y_{k'_i}|x_i)$ as unbiased estimation of the mean of the probability $\mathbb{E}_{p(y)}\left[q_\theta(y|x_i)\right]$. In this case, the sampled mutual information estimator [10] is defined as:

$$I_{vCLUBs} = \frac{1}{N}\sum_{i=1}^{N}\left[\log q_\theta(y_i|x_i) - \log q_\theta(y_{k'_i}|x_i)\right], \tag{4}$$

where N is the size of the training data or the minibatch.

Why do we use mutual information minimization? Mutual information maximization with a lower bound is effective for representation learning [11] to obtain similar features. On the contrary, mutual information minimization with an upper bound has advantages in disentangled representation learning [12]. In this paper, we claim that each modality contributes partially to the complete output, and mutual information minimization is more suitable for our task of exploring the complementary information of each modality. Specifically, we follow [10] and introduce cross-modal mutual information minimization as a regularizer term to constrain the model to focus on the exclusive feature of each modality for our joint optical flow and scene flow estimation task.

4. Network Details

Multi-modal Feature Pyramid Extraction with Feature Stage Fusion. For the given three modalities of inputs, *i.e.*, two frames of images, two frames of point clouds, and events, we first voxelize the raw events $E = \{x_i, y_i, t_i, p_i\}^K$ into one event voxel $EV \in \mathbb{R}^{H\times W\times B}$ that can be used as input to our network, where K is the number of events during the period between two frames and B is the number of intervals that need to be manually chosen to sample the original event in the time dimension. Then we use the Siamese encoders to construct feature pyramids for each modality respectively. As there are two frames of images and point clouds, we extract two feature pyramids $\{e_{r_1}, e_{r_2}\}_l$ and $\{e_{pc_1}, e_{pc_2}\}_l$ with pyramid layers $l \in [1, L]$ for images and point clouds, We only constructe one feature pyramid e_{evl} for events, because we take the events as a single independent input data.

Then we conduct the *feature stage fusion* in both 2D branch and 3D branch. As the event data spans the entire time interval, only the RGB image and point cloud data are fused in this stage. Specifically, the 3D multimodal fusion in Eq.4 is applied to fuse both features of RGB images with the corresponding point clouds, and the 2D multimodal fusion in Eq.5 is performed to achieve feature stage fusion in 2D. Note that the auxiliary feature in this case indicates unimodal representation, *i.e.* $Y^{3D}_{aux} = e^{pj}_{r_1}/e^{pj}_{r_2}$ for 3D fusion and $Y^{2D}_{aux} = e^{pj}_{pc_1}/e^{pj}_{pc_2}$ for 2D fusion. Similarly, the mutual information regularization is defined between the image feature and point cloud feature in both 2D and 3D space, where $I^{vub}(e_{r_1}; e_{pc_1}), I^{vub}(e_{r_2}; e_{pc_2})$ from Eq.6 is used. In summary, we conduct four times fusion operations at the feature stage of each pyramid layer.

Correlation Construction with Motion Stage Fusion. To model the motion correlations from two frames of images and point clouds, we first warp image and point cloud features to reference frame using the coarse optical flow and scene flow estimated from the previous pyramid layer, and then construct 2D and 3D correlations by computing 2D and 3D cost volumes. The cost volume cv_r of the 2D branch is computed by local search [13], and cv_{pc} of the 3D branch is realized by a learnable layer [14]. Similar to the feature stage fusion, *motion stage fusion* is achieved with the multi-modal attention fusion strategy and mutual information regularization (Eq.8) for the projected cost volume-based feature representation cv_r and cv_{pc} and event feature e_{ev} in 2D and 3D (Eq.5 and Eq.4). In this process, we conduct two fusion operations with the event feature in both 2D and 3D

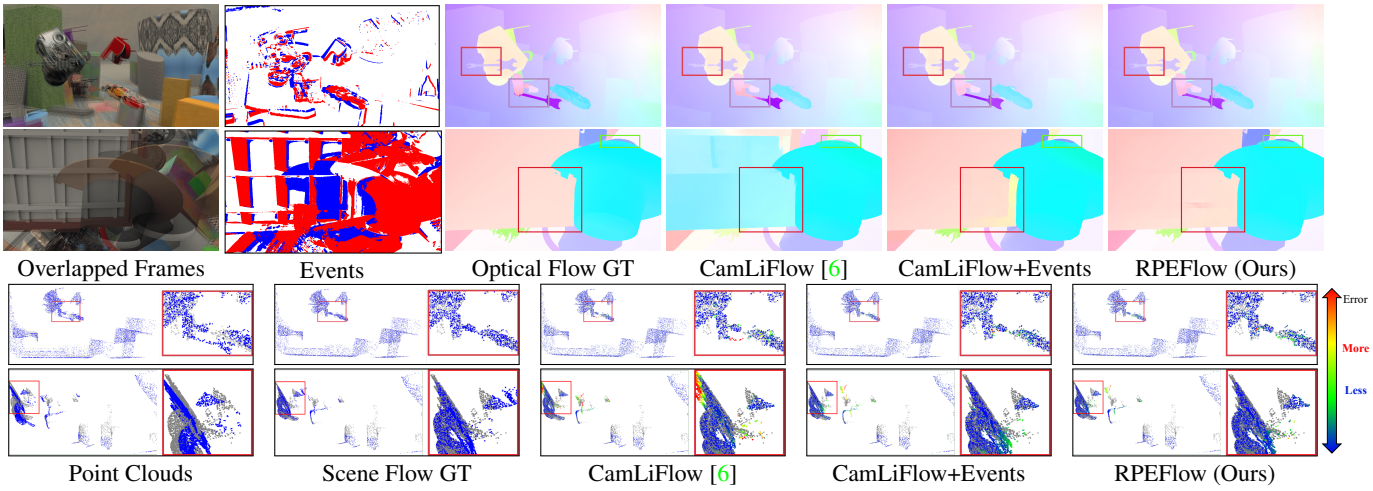

Figure 3: **Extended visual comparisons** on the "val" split of the FlyingThings3D [4] subset. For the bottom 3D comparisons, blue indicates a lower error, red indicates a higher error, and green indicates the median. Best to zoom in on the screen for detailed comparisons.

branches, because events can provide complementary motion information to enhance the motion correlation.

Flow Estimation with Estimation Stage Fusion. With the motion correlation, we then estimate the motion filed with the flow decoder and flow estimator in the corresponding 2D or 3D branch. The inputs to the flow decoder include the output of motion correlation, the reference frame feature, the event feature, and the coarse flow from the upper pyramid layer. Again, we conduct an *estimation stage fusion* to fuse the outputs of the two branches decoder with the event feature. Finally, the 2D optical flow and 3D scene flow are produced with two different flow estimators and are fed into the next pyramid layer to achieve coarse-to-fine predictions. We take the optical flow estimation in the 2D branch and the scene flow estimation in the 3D branch of the last pyramid layer as our final joint predictions.

5. Experiments

5.1. Extended Evaluations

Due to space limitations, we only show one sample of visual comparisons on the FlyingThings [4] dataset in the main submission, and here we further provide two in Fig. 3. We also conduct generalizability experiments on the KITTI [15] dataset, which is usually used in image-based optical flow and scene flow estimation methods. In Table 2, we report the qualitative results on KITTI using models pretrained on FlyingThings3D. From these additional comparisons, we can conclude that both the introduction of events and our proposed framework can significantly improve the accuracy of optical flow and scene flow estimation.

Method	Optical Flow		Scene Flow	
	EPE_{2D}	ACC_{1px}	EPE_{3D}	$ACC_{.05}$
RAFT [16]	4.296	68.14%	-	-
RAFT-3D [17]	4.725	57.67%	0.518	33.55%
CamLiFlow [6]	7.643	61.50%	0.223	43.40%
RAFT+Event	4.296	68.14%	-	-
CamLiFlow+Event	4.887	66.77%	0.190	49.47%
RPEFlow (Ours)	**3.308**	**70.02%**	**0.173**	**54.32%**

Table 2: **Generalization performance comparison** on the "training" split of KITTI [15] dataset. Both models are pretrained on the FlyingThings3D dataset and evaluated directly on the KITTI dataset without finetuning.

5.2. Real vs. virtual event camera

In Table 3, we additionally evaluate the simulated events of DSEC [18] dataset. The performance gap suggests that the simulated events cannot fully model the actual dynamics resulting in performance degradation, while real captured events can provide rich motion information. This further demonstrates that events are helpful and irreplaceable in capturing high dynamics.

References

[1] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David J Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, et al. Kubric: A scalable dataset generator. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3749–3761, 2022. 1

[2] Erwin Coumans and Yunfei Bai. Pybullet, a python mod-

Method	Event Type	EPE_{2D}	ACC_{1px}	EPE_{3D}	$ACC_{.05}$
E-RAFT [8]	Real	0.473	92.11%	-	-
E-RAFT [8]	Sim	0.625	87.11%	-	-
CamLiFlow+Event	Real	0.361	95.07%	0.116	55.26%
CamLiFlow+Event	Sim	0.437	93.51%	0.133	49.35%
RPEFlow (Ours)	Real	**0.332**	**95.27%**	**0.104**	**60.50%**
RPEFlow (Ours)	Sim	0.396	94.02%	0.119	54.46%

Table 3: Comparison between real and simulated events.

ule for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2021. 1

[3] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. 1

[4] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4040–4048, 2016. 1, 2, 4

[5] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 529–537, 2019. 2

[6] 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 *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5791–5801, 2022. 2, 4

[7] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5754–5763, 2019. 2

[8] Mathias Gehrig, Mario Millhäusler, Daniel Gehrig, and Davide Scaramuzza. E-raft: Dense optical flow from event cameras. In *International Conference on 3D Vision (3DV)*, pages 197–206, 2021. 2, 5

[9] David Barber and Felix V. Agakov. The im algorithm: A variational approach to information maximization. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 201–208, 2003. 3

[10] Pengyu Cheng, Weituo Hao, Shuyang Dai, Jiachang Liu, Zhe Gan, and Lawrence Carin. Club: A contrastive log-ratio upper bound of mutual information. In *International Conference on Machine Learning (ICML)*, pages 1779–1788, 2020. 3

[11] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. In *International Conference on Learning Representations (ICLR)*, 2019. 3

[12] Ricky T. Q. Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 3

[13] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. Pwc-net: Cnns for optical flow using pyramid, warping, and cost volume. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8934–8943, 2018. 3

[14] Wenxuan Wu, Zhi Yuan Wang, Zhuwen Li, Wei Liu, and Li Fuxin. Pointpwc-net: Cost volume on point clouds for (self-) supervised scene flow estimation. In *European Conference on Computer Vision (ECCV)*, pages 88–107. Springer, 2020. 3

[15] Moritz Menze, Christian Heipke, and Andreas Geiger. Object scene flow. *ISPRS Journal of Photogrammetry and Remote Sensing*, 140:60–76, 2018. 4

[16] Zachary Teed and Jia Deng. Raft: Recurrent all-pairs field transforms for optical flow. In *European Conference on Computer Vision (ECCV)*, pages 402–419, 2020. 4

[17] Zachary Teed and Jia Deng. RAFT-3D: Scene flow using rigid-motion embeddings. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8375–8384, 2021. 4

[18] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. Dsec: A stereo event camera dataset for driving scenarios. *IEEE Robotics and Automation Letters*, 2021. 4