EMoTive: Event-guided Trajectory Modeling for 3D Motion Estimation

Supplementary Material

1. The Details of EMoTive Pipeline

Here, we provide more details about the EMoTive framework.

1.1. Spatio-temporal Feature Encoder

Spatial Feature Encoding. Building upon established architectures in optical flow estimation [**? ?**], EMoTive employs two same 2D convolutional networks to extract latent spatial representation $f_{hw} : \mathbb{R}^{D_s \times H_D \times W_D}$ and context representation f_c from the Event Voxels. Specifically, we obtain the two Event Voxels from adjacent time $V^p : \mathbb{R}^{B \times H \times W}, V^t : \mathbb{R}^{B \times H \times W}$. The bins B is set to 7. Both voxels are processed by the spatial feature encoder to extract the spatial information at adjacent time, while the latter one is processed by the context feature encoder in the meantime for initializing motion information and mask features. The spatial feature encoder uses a 2D spatial convolutional layer with residual blocks to downsample the Event Voxel V, producing spatial features $f_{hw} : \mathbb{R}^{D_s \times H_D \times W_D}$. Here, $H_D = H/8, W_D = W/8$. The context feature encoder uses the same structure but outputs initialized motion information and mask features for upsampling via *ReLU* and *Tanh* activation functions.

Temporal Feature Encoding. For temporal feature extraction, we apply a 1D convolution encoder operating on the Event Kymographs (K_x, K_y). The whole encoder framework is designed following the spatial encoder, except replacing the 2D convolution with 1D. The Event Kymographs will firstly be evenly divided into subblocks based on the N_a temporal anchors to form the sequence $K_{x|y,s} = \{K_{x|y,s}(1), \ldots, K_{x|y,s}(N_a)\}$. The encoder operates on each subblock and outputs D_t-dimensional temporal features. Meanwhile, progressive spatial downsampling is performed to align with the spatial feature. This encoder outputs two complementary temporal feature tensors: $f_{ht} : \mathbb{R}^{N_a \times D_t \times H_D}$ and $f_{wt} : \mathbb{R}^{N_a \times D_t \times W_D}$, preserving axis-specific motion pattern.

1.2. Spatio-Temporal Trajectory

The parameterized trajectory in the form of a non-uniform rational B-spline is given by:

$$\mathcal{T}(t, x, y) = \frac{\sum_i^n N_{i,p}(t) w_i \mathbf{P}_i(x, y)}{\sum_i^n N_{i,p}(t) w_i}, \tag{1}$$

where $N_{i,p}(t)$ is the p-th degree B-spline basis function function with non-uniform knot vector $\mathbf{T} = \{t_1, \ldots, t_m\}$, $w_i \in \mathbb{R}^+$ represents the event-adaptive weight and $\mathbf{P}_i \in \mathbb{R}^2$ is the i-th control point from the set $\mathcal{P} = \{\mathbf{P}_1, \ldots, \mathbf{P}_n\}$.

The spline basis function in this paper adopts the Cox-de Boor recursive formula:

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \le t < t_{i+1} \\ 0 & \text{otherwise} \end{cases}, \tag{2}$$

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t), \tag{3}$$

where t_i represents the node vector. For a clamped NURBS curve of degree p with n control points, the knot vector contains $m = n + p + 1$ elements, with only $n - p$ intermediate knots $t_c \in (t_{p+1}, t_{n+1})$ being adjustable. The trajectory $\mathcal{T}$ models pixel displacement through temporal evolution, where density-aware adaptation adjusts both knot vector distribution $\mathbf{T}$ and weight $\{w_i\}$ values, while control points are updated via spatio-temporal dual cost volumes.

1.2.1. Spatio-temporal Query

For each pixel $\mathbf{p} = (x, y)$ at at reference time $t = 0$, we sample its warped position $\mathbf{p}_t = \mathbf{p} + \mathcal{T}(t, x, y)$ along the NURBS trajectory. At each queried timestamp $\{t_{l,j}\}_{j=1}^n$ from density adaptation, we extract neighborhood correlation within radius r from the cost pyramid:

$$\mathcal{N}(\mathbf{p}_t) = \{\mathbf{p}_t + \delta\mathbf{p} \mid \delta\mathbf{p} \in \{-r, \ldots, r\}^2\}, \tag{4}$$

where bilinear interpolation handles subpixel coordinates. The radius is set to 4 in this paper.

1.2.2. Feature Fusion

We combine three information streams for control point update: (1) Temporal cost volumes at adaptively sampled $\{t_{l,j}\}$; (2) Spatial cost volume from end-time displacement $\mathcal{T}(1, x, y)$; (3) Context representation f_c from spatial feature encoding. These are concatenated into a spatio-temporal feature $\mathcal{F} \in \mathbb{R}^{H \times W \times d}$ through convolutions. The temporal cost volumes and spatial cost volume will be concated as the dual spatio-temporal cost volume, combined with parameters of control points to get the latent motion feature from motion encoder. The motion encoder is composed of dual branches with 2D convolution. Then, the latent motion feature is concated with context representation f_c to compose the final spatio-temporal feature. The number of control point iterative refinements is set to 6.

1.3. Motion in Depth Estimation

The motion in depth component $\mathcal{M}$ can be derived from the temporal gradient of the trajectory $\mathcal{T}$. Assuming a non-

rotating rigid body under perspective projection with constant velocity in world coordinates, we establish the depth motion relationship:

$$\frac{Z_1}{Z_0} = \mathcal{M} = \frac{v_0 \Delta t + \Delta x}{v_1 \Delta t + \Delta x},\ \Delta t = t_1 - t_0, \tag{5}$$

where Δx and (v_0, v_1) represent the displacement and velocities of the object along the x-axis between times t_0 and t_1, respectively (complete derivation provided in supplementary material).

To elaborate further, consider a common scenario: a non-rotating rigid body using a pinhole camera model, the velocity of an object relative to the camera along the x-axis can be expressed as:

$$v = \frac{V_x - x V_z}{Z}. \tag{6}$$

Here, we normalize the pixel coordinates x_{pix} and focal length f such that $x = \frac{x_{pix}}{f}$. Z represents the object's relative depth in the camera coordinate system, while V_x and V_z denote the 3D motion velocities of the object along the x-axis and z-axis, respectively. The term v represents the instantaneous velocity along the x-axis in the image plane, with the same expression applying to the velocity projection along the y-axis, denoted as u. Assuming the object's 3D motion velocity remains constant in the real-world coordinate, we examine the projected velocity v_0 and v_1 at times t_0 and t_1, respectively. This leads to the following equation:

$$v_0 - v_1 \frac{Z_1}{Z_0} = \frac{x_1 - x_0}{t_1 - t_0}(\frac{Z_1}{Z_0} - 1), \tag{7}$$

where x_0 and x_1 represent the positions of the object along the x-axis at times t_0 and t_1, respectively. And the velocities along the z-axis, V_z in Eq. (6) is replaced under the constant assumption: $V_z = \frac{Z_1 - Z_0}{t_1 - t_0}$. By re-organizing the above equation, we can obtain the form of motion in depth as follows:

$$\frac{Z_1}{Z_0} = \mathcal{M} = \frac{v_0(t_1 - t_0) + (x_1 - x_0)}{v_1(t_1 - t_0) + (x_1 - x_0)}. \tag{8}$$

Given the starting observation position and time x_0, t_0 set to 0, we can obtain:

$$\mathcal{M} = \frac{v_0 t_1 + x_1}{v_1 t_1 + x_1} \tag{9}$$

Once the expression for the object's motion trajectory $\mathcal{T}$ is obtained, its time gradient $\mathcal{T}'(t)$ can be estimated, corresponding to the instantaneous velocity (v, u). The gradient of the parameter t for the non-uniform rational B-spline (NURBS) curve used in this paper is given by the following:

$$(v, u) = \mathcal{T}'(t) = \frac{\sum_i N'_{i,p}(t) w_i \mathbf{P}_i(x, y) - \mathcal{T}(t) \sum_i N'_{i,p}(t) w_i}{\sum_i N_{i,p}(t) w_i}, \tag{10}$$

$$N'_{i,p}(t) = \frac{p}{t_{i+p} - t_i} N_{i,p-1} - \frac{p}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}. \tag{11}$$

Combined with Eq. (9), the estimation of motion in depth based on the trajectory is as follows:

$$\mathcal{M} = \frac{\mathcal{T}'(t_0)t_1 + \mathcal{T}(t_1)}{\mathcal{T}'(t_1)t_1 + \mathcal{T}(t_1)} = 1 + \frac{\mathcal{T}'(t_0)t_1 - \mathcal{T}'(t_1)t_1}{\mathcal{T}'(t_1)t_1 + \mathcal{T}(t_1)} \tag{12}$$

The final multi-view estimation algorithm of motion in depth is shown in Algorithm 1.

Algorithm 1 Motion in Depth Multi-View Estimation Process

INPUT: Trajectory $\mathcal{T}(t)$, timestamps: $\{t_0, t_1, \ldots, t_k\}$
OUTPUT: Motion-in-depth at t_k: $\mathcal{M}_k$

1: Retrieve trajectory values at each time point $\{\mathcal{T}(t_0), \mathcal{T}(t_1), \ldots, \mathcal{T}(t_k)\}$
2: Compute trajectory time gradients (instantaneous velocities) at each time point according to Eq. (11) $\{\mathcal{T}'(t_0), \mathcal{T}'(t_1), \ldots, \mathcal{T}'(t_k)\}$
3: Multiply the initial instantaneous velocity by each timestamp to obtain the initial path estimate $\{\mathcal{T}'(t_0)t_1, \mathcal{T}'(t_0)t_2, \ldots, \mathcal{T}'(t_0)t_k\}$
4: Multiply the instantaneous velocity at each time point by the respective timestamp to get the endpoint path estimate $\{\mathcal{T}'(t_1)t_1, \mathcal{T}'(t_2)t_2, \ldots, \mathcal{T}'(t_k)t_k\}$
5: Following Eq. (9), combine the trajectory, initial path estimate, and endpoint path estimate to compute depth motion estimates at each time point $\{\mathcal{M}_1, \mathcal{M}_2, \ldots, \mathcal{M}_k\}$
6: According to the multi-view relationship, convert depth motion observations from different times to the same time point, yielding a series of depth motion estimation at time t_k: $\{\mathcal{M}_{1,k}, \mathcal{M}_{2,k}, \ldots, \mathcal{M}_{k,k}\}$
7: Combine historical observations from different time points to stabilize the depth motion estimate at t_k: $\mathcal{M}_k = \frac{1}{k} \sum_i \mathcal{M}_{i,k}$

1.4. Parameter Upsampling

Since the spatiotemporal information for motion estimation comes from downsampled features, its output (including motion trajectory, optical flow, and depth motion) is at 1/8 of the original resolution. In this paper, the mask obtained from the context feature encoder is used to upsample the

motion estimation output to full resolution. Specifically, the motion parameters at lower resolutions are first expanded using a 3×3 grid, and then a convex combination is performed based on the mask (which has been normalized per channel using the Softmax function) to upsample and obtain the motion estimation results at full resolution.

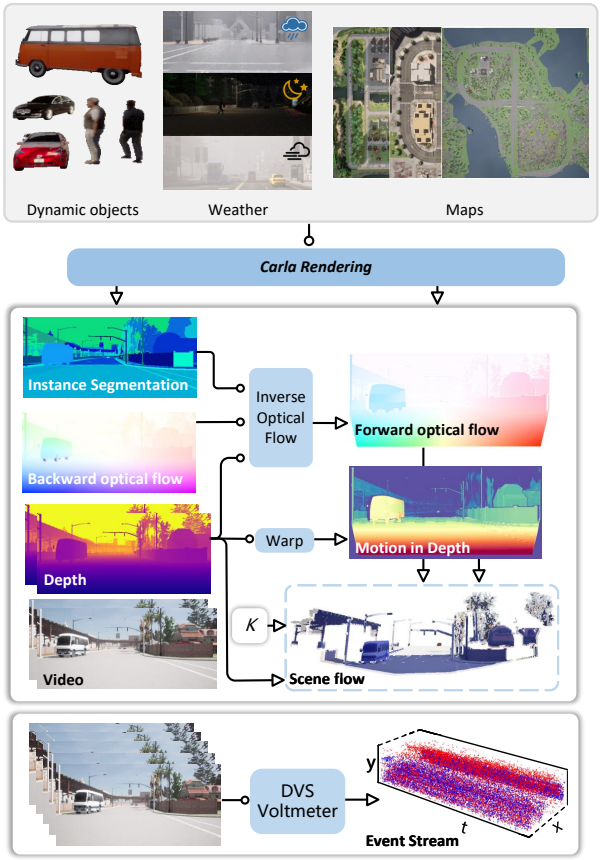


Figure 1. Data Simulation process based on Carla Simulator

2. Data Collection Process

We employ the Carla simulator [**?**] to generate the 3D motion dataset **CarlaEvent3d** in a driving environment. Carla provides realistic simulations of various weather conditions, as well as optical flow on the camera plane and relative depth labels during the driving process. For event generation, we first leverage the UE4 engine in Carla to produce high-frame-rate video and then simulate events using the DVS Voltemter algorithm [**?**] integrated into the Carla simulation workflow. The detailed data generation process is illustrated in Fig. 1. Ultimately, we obtained 75 sequences across diverse environments—including rain, fog, and night scenes—resulting in a total of 22,125 event-image-3D motion labels tuples.

2.1. Forward Optical Flow Generation

Generating optical flow via the Carla simulator presents two major challenges: first, the precision of the optical flow is limited to 10 digits in each direction; second, the simulator produces backward optical flow rather than the forward optical flow that is commonly used. This precision limitation arises because Carla employs the Emissive Color property of UE4 materials to output optical flow, yielding up to three channels of `float16` color with each channel providing up to 10 bits of valid information. To enhance accuracy, we utilize two independent materials to encode the horizontal and vertical components of the optical flow. Each material continues to use Emissive Color, but the first ten bits and the last ten bits of the optical flow are encoded into separate channels, thereby improving the final output quality. For forward optical flow generation, we adopt the efficient Inverse Optical Flow algorithm [**?**] and combine it with the depth map to transform the backward optical flow into forward optical flow. Moreover, since Carla provides instance segmentation results, we further refined the algorithm so that the inverse computation of the matching pixel position accounts for both depth approximation and semantic label consistency, ultimately yielding more accurate forward optical flow estimates.

2.2. Motion in Depth Generation

The depth motion label is computed by warping the depth value from the target moment to the initial moment using the forward optical flow:

$$\mathcal{M} = \frac{Z_1(x + v(x))}{Z_0(x)}, \tag{13}$$

where Z_1 is the target moment depth map, Z_0 is the initial moment depth map, and v represents the forward optical flow. To address instability and uncertainty in depth labeling at object boundaries, we mask out estimation results near these boundaries.

After obtaining the forward optical flow and motion in depth, we combine the initial depth value with the camera's internal parameters to derive the scene flow.

3. Comprehensive Experimental Results

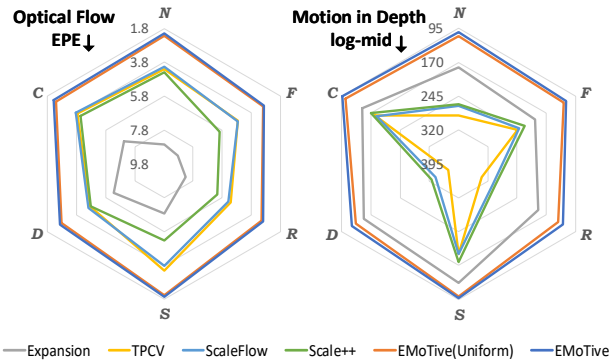
In this section, we present additional experimental results for 3D motion estimation.

3.1. Performance in Different Scenes

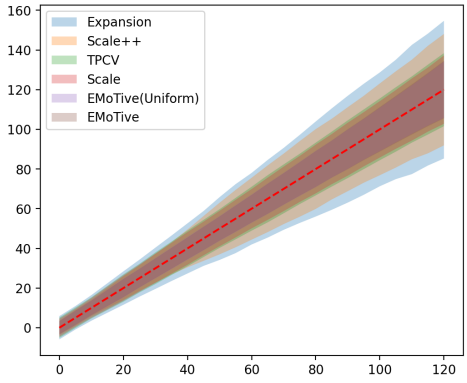
We evaluated our method across various scenes in CarlaEvent3D, including rain, fog, night, cloudy daytime, sunny, and dusk, as shown in Fig. 2a. Even under challenging conditions—such as low-light noise, insufficient contrast, and rainwater interference—EMoTive achieved excellent motion estimation, demonstrating robust algorithmic

Datasets	year	Synthetic /Real	Optical Flow	Depth	Instance Segmentation	Environment					
						Lighting			Weather		
						Daytime	Nighttime	Sunset	Cloudy	Foggy	Rainy
MVSEC [?]	2018	R	✔	✔	✘	✔	✔	✘	✔	✘	✘
DSEC [?]	2020	R	✔	✔	✘	✔	✔	✔	✔	✘	✘
Ekubric [?]	2023	S	✔	✔	✘	✔	✘	✘	✔	✘	✘
KITTI-Event [?]	2023	R+S	✔	✔	✘	✔	✘	✘	✔	✘	✘
FlyingThings-Event [?]	2023	S	✔	✔	✘	✔	✘	✘	✔	✘	✘
BlinkVision [?]	2024	S	✔	✔	✘	✔	✘	✘	✔	✘	✘
CarlaEvent3d	2024	S	✔	✔	✔	✔	✔	✔	✔	✔	✔

Table 1. Related motion estimation dataset based on event camera.



(a) Evaluation of Motion estimation performance in different environments. ***N***:Nighttime, ***D***:Daytime, ***S***:Sunset, ***C***:Cloudy, ***F***:Foggy, ***R***:Rainy.



(b) The plot for motion amplitude estimation capability

Figure 2. Verification of Motion estimation ability

stability. Notably, in foggy conditions, methods like Expansion experienced a significant decline in optical flow estimation (a decrease of 2.48 px in accuracy), whereas EMoTive exhibited only a 1.01 px change, thereby maintaining a higher degree of accuracy.

3.2. Performance of Motion Amplitude Estimation

To assess the capability of motion amplitude estimation, we evaluated the methods over a range of amplitudes from 0 to 120 px/100ms. Accuracy was quantified by the standard deviation within different motion intervals; a smaller standard deviation indicates a closer approximation to the true motion and, hence, better estimation accuracy. The validation results are presented in Fig. 2b. The findings reveal that as the motion amplitude increases, the performance of all methods deteriorates, underscoring the significant challenge posed by fast motion. However, the proposed EMoTive method maintains lower standard deviations at high speeds and exhibits a relatively consistent error range across different motion intervals. This outcome indicates that the event-guided non-uniform trajectory design provides robust tracking performance across various motion intensities, thereby enabling stable motion estimation.

4. Result Visualization on CarlaEvent3D

To illustrate the motion estimation performance, we present additional visualizations on the CarlaEvent3D dataset across various scenes. These visualizations include images at the start and end moments, the event sequence input, event voxel projection, event kymograph projection, and the motion estimation outputs from several methods, including our proposed EMoTive model (see Figs. 3 to 8). The results demonstrate that EMoTive achieves clearer spatial boundaries in both optical flow and depth motion estimation. Moreover, for fast-moving objects, its 3D motion estimation is more comprehensive and accurate. In particular, the non-uniform curve representation in EMoTive yields a motion estimation distribution that more closely aligns with the ground truth, resulting in a smoother depiction of motion.

5. Limitations

The current spline representation has limitations, such as the absence of long-term temporal integration and difficulty in distinguishing nearby homogeneous objects, which may lead to incomplete object observations in certain scenarios.

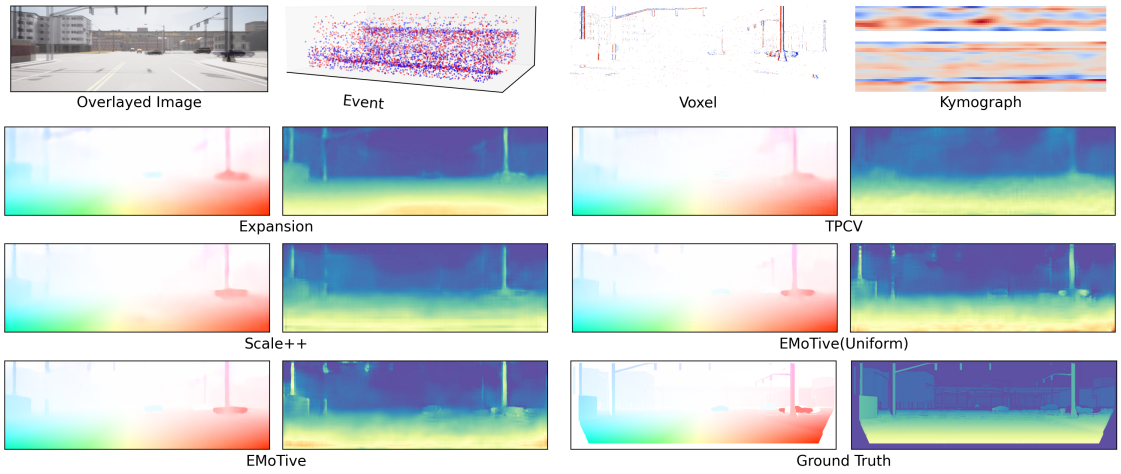


Figure 3. Qualitative comparison on the CarlaEvent3D dataset (daytime). The left is optical flow and the right is motion in depth estimation.

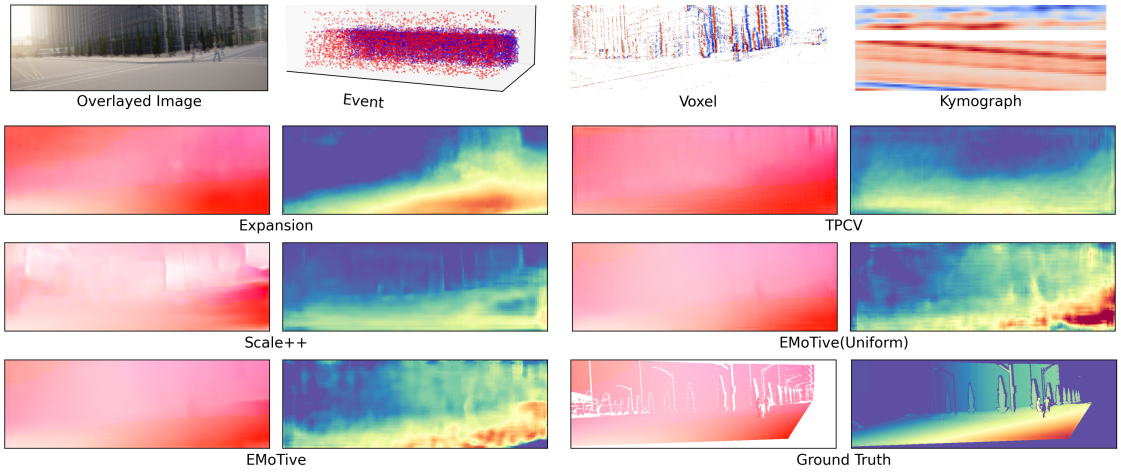


Figure 4. Qualitative comparison on the CarlaEvent3D dataset (sunset).

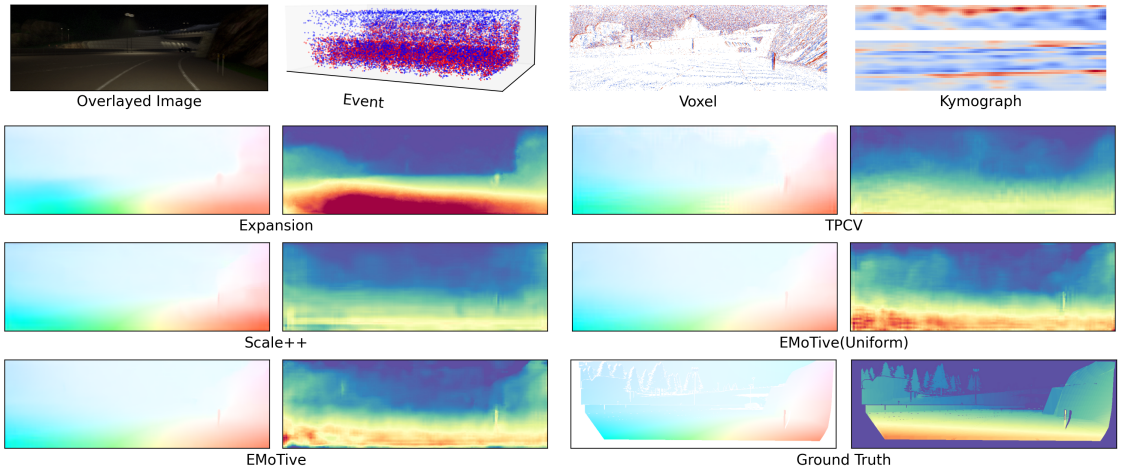


Figure 5. Qualitative comparison on the CarlaEvent3D dataset (nighttime).

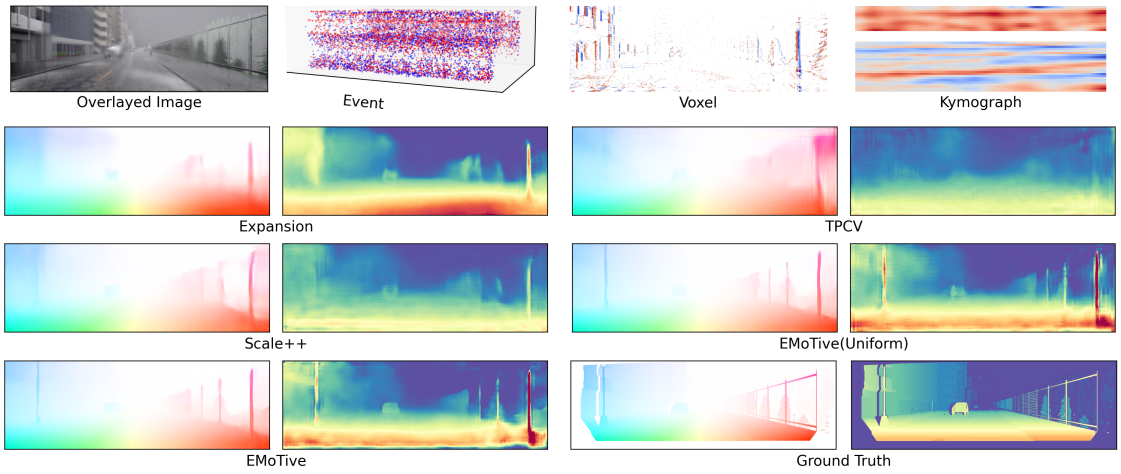


Figure 6. Qualitative comparison on the CarlaEvent3D dataset (rainy).

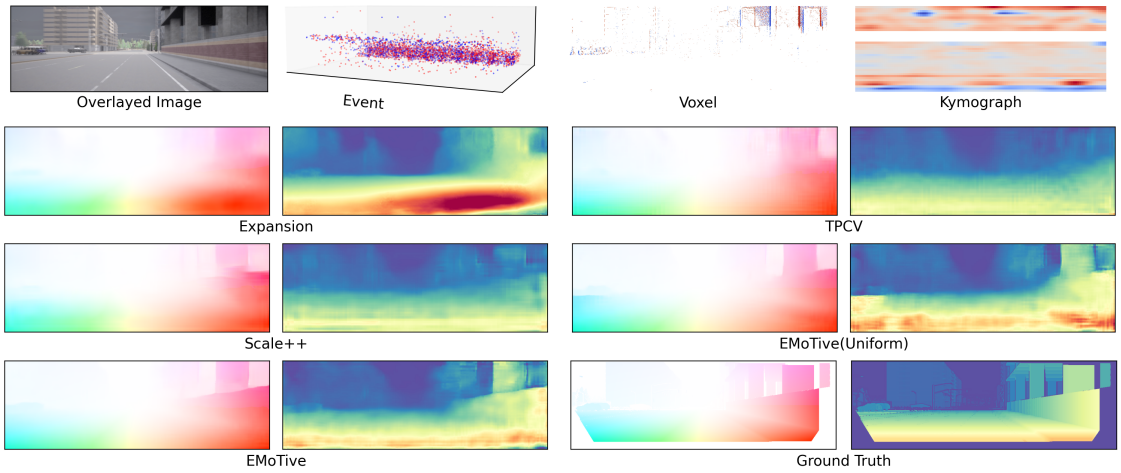


Figure 7. Qualitative comparison on the CarlaEvent3D dataset (cloudy).

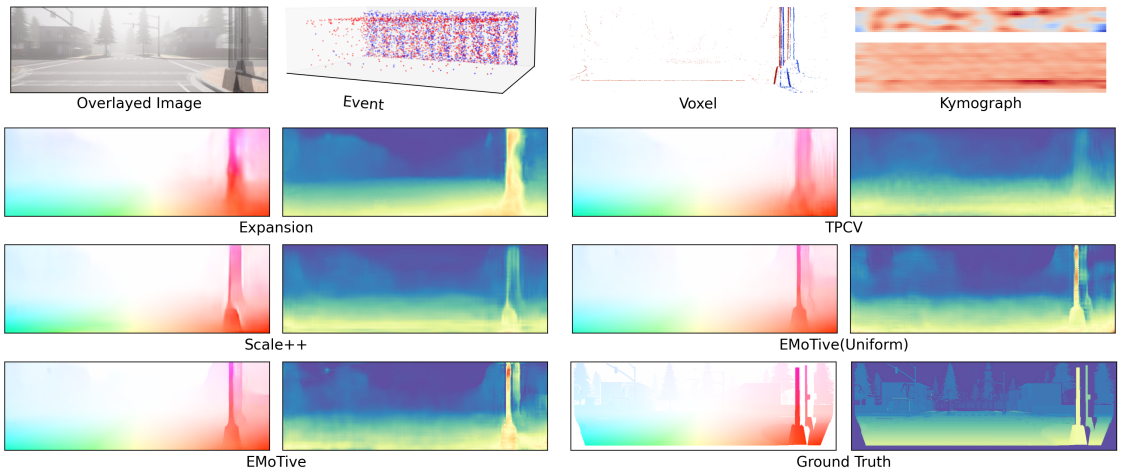


Figure 8. Qualitative comparison on the CarlaEvent3D dataset (foggy).