BlinkTrack: Feature Tracking over 80 FPS via Events and Images

Supplementary Material

In this supplementary document, we first introduce more details about BlinkTrack in Sec. A, including an explanation of event representation and pre-processing in Sec. A.1, detail of the image module supervision in Sec. A.2, model definition in Kalman filter in Sec. A.3, and more implementation detail in Sec. A.4. Then we present more experiment results in Sec. B, including quantitative comparisons in Sec. B.1, qualitative comparisons in Sec. B.2, uncertainty visualization in Sec. B.3, runtime comparison between our method and other methods in Sec. B.4, and discussion on reference frame in Sec. B.5. At last, we add more details about our dataset in Sec. C, including displacement distribution analysis in Sec. C.1, data quality experiment in Sec. C.2, synthetic data quality experiment in Sec. C.3, some visualization examples in Sec. C.4, and license explanation in Sec. C.5.

A. More Details about BlinkTrack

A.1. Event Representation and Pre-processing

A single event $\mathbf{e}_k = \{x_k, y_k, t_k, p_k\}$ is triggered when pixel (x_k, y_k) have brightness change indicated by $p_k \in \{-1, +1\}$ at time t_k[23]. The event stream $E = \{\mathbf{e}_k\}_{k=1}^{n}$ is composed of a large amount of discrete single event e_k. The frame-based neural network cannot directly process such data representations; therefore, extracting meaningful information from discrete events and converting them into spatially and temporally aligned frames is essential. According to the representation experiments[14], we choose SBT-Max[20]. The representation we use has 5 temporal bins for positive and negative polarity(i.e., the sign of the brightness change), respectively, and the streaming events in between the times are merged into those bins.

We denote the event frame interval as Δt and we want to extract event frame $\mathbf{E}_j$ from events $E_j = \{\mathbf{e}_k\}_{k=1}^{n_{evt}}$ between $t_j - \Delta t$ and t_j with n_{bin} bins which means interval for each bin is $\frac{\Delta t}{n_{bin}}$, for example the time interval of first bin is $(t_j - \Delta t, t_j - \Delta t + \frac{\Delta t}{n_{bin}}]$. Each pair of positive and negative channels is formed by the maximal timestamp of positive or negative polarity, respectively, on a single pixel among events from $(t, t + \frac{\Delta t}{n_{bin}}]$. If there is no event with the expected polarity between this interval for some pixels, the number of the expected polarity in this bin for these pixels would be set to 0, see Eq. A.

$$
\begin{aligned}
\mathbf{E}_j(u, v, 2d) &= \begin{cases} max(t_k - t \mid p_k = -1, u = x_k, v = y_k) \\ 0, \text{if there is no such event} \end{cases} \\
\mathbf{E}_j(u, v, 2d+1) &= \begin{cases} max(t_k - t \mid p_k = 1, u = x_k, v = y_k) \\ 0, \text{if there is no such event} \end{cases} \\
d &= \{0, \ldots, n_{bin}\}, k = \{1, \ldots, n_{evt}\}
\end{aligned} \tag{A}
$$

A.2. Details of the Image Module Supervision

The image module is also trained on our MultiTrack dataset to align with the event module, employing similar supervision. Due to its lightweight architecture and expansive receptive field, the Kalman filter and uncertainty supervision are applied from the outset, calculating displacement loss based on both the module direct predictions $\Delta\hat{\mathbf{p}}$ and Kalman filter predictions $\Delta\tilde{\mathbf{p}}$. Given the importance of the Kalman filter's predictions in supervising both accuracy and uncertainty, the other losses are weighted by $w_2 = \frac{1}{2}$ and $w_3 = \frac{1}{2}$.

$$
\begin{aligned}
\mathcal{L}_{\hat{disp}} &= || \Delta\hat{\mathbf{p}} - \Delta\mathbf{p} ||_1 \\
\mathcal{L}_{\tilde{disp}} &= || \Delta\tilde{\mathbf{p}} - \Delta\mathbf{p} ||_1 \\
\mathcal{L}_{image} &= \mathcal{L}_{\tilde{disp}} + w_2\mathcal{L}_{\hat{disp}} + w_3\mathcal{L}_{uncert}
\end{aligned} \tag{B}
$$

Since the Kalman filter and uncertainty module are activated from the beginning, stable supervision is achieved by selecting only points visible in the initial target frame, enabling the Kalman filter to initialize velocity rather than maintain a random initial velocity. To accelerate convergence and improve performance on challenging points, supervision points are sampled based on the weight of their prediction loss, with points exhibiting higher loss being more likely to receive increased supervision.

A.3. Kalman Filter Definition

The Kalman filter[11, 22] comprises two fundamental steps: prediction and update. The prediction step estimates the state and its uncertainty at the next time step based on the current state and process model, allowing the internal state to evolve even when no new measurement is available or the measurement is highly uncertain. The prediction step is formulated as:

$$
\begin{aligned}
\hat{x}_{k|k-1} &= F\hat{x}_{k-1|k-1}, \\
P_{k|k-1} &= FP_{k-1|k-1}F^T + Q,
\end{aligned} \tag{C}
$$

where x is the state estimate, P is the state covariance, F is the state transition model, and Q is the process noise covariance. The update step incorporates new measurements

to refine the state estimate and reduce uncertainty, which is formulated as:

$$\begin{aligned} y_k &= z_k - H\hat{x}_{k|k-1}, \\ S_k &= HP_{k|k-1}H^T + R, \\ K_k &= P_{k|k-1}H^T S_k^{-1}, \\ \hat{x}_{k|k} &= \hat{x}_{k|k-1} + K_k y_k, \\ P_{k|k} &= (I - K_k H)P_{k|k-1}, \end{aligned} \tag{D}$$

where y is the measurement residual, z is the new measurement(our predicted $\Delta\hat{\mathbf{p}}$), S is the residual covariance, R is the measurement noise covariance(our predicted $\hat{\mathbf{\Sigma}}$), H is the observation model, K is the Kalman gain, and I is the identity matrix. Our final prediction $\Delta\tilde{\mathbf{p}}$ is from the first two elements in the vector $\hat{x}_{k|k}$.

The simple constant velocity model is defined in our Kalman filter. The state x, covariance matrix P, observation model H, state-transition model F, and the covariance of the process noise Q are defined as follows.

$$\begin{aligned} x &= (\mathrm{x}, \mathrm{y}, v_x, v_y)^{\mathrm{T}} \\ P &= \begin{pmatrix} \rho_{xx} & 0 & \rho_{xv_x} & 0 \\ 0 & \rho_{yy} & 0 & \rho_{yv_y} \\ \rho_{v_x x} & 0 & \rho_{v_x v_x} & 0 \\ 0 & \rho_{v_y y} & 0 & \rho_{v_y v_y} \end{pmatrix} \\ H &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \\ F &= \begin{pmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ Q &= \begin{pmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{pmatrix} \end{aligned} \tag{E}$$

To explicitly construct the motion model, the state x includes variables of position and velocity of both x and y. The elements in the covariance matrix P represent the correlation coefficients of these variables, where it is assumed that no correlation exists between x and y. We initialize state x with $\mathbf{0}$ and covariance matrix P with the Identity matrix, indicating the point is stationary initially with medium confidence. The observation model H specifies that the observations from our modules correspond to the first two elements of the state x. The state-transition model F follows a constant velocity assumption, given by $\mathrm{x}_k = \mathrm{x}_{k-1} + v_x \cdot \Delta t$, where Δt denotes the time interval between the current and the previous observations. The process noise Q reflects increasing uncertainty in the Kalman state as the duration without observations grows.

In our implementation, when a measurement, including displacement $\Delta\hat{\mathbf{p}}$ and covariance matrix $\hat{\mathbf{\Sigma}}$, arrives at timestamp t, the Kalman filter first performs a prediction step up to t, generating the predicted state x and state covariance P. The state covariance represents the uncertainty of the state, which increases with the time interval since the last prediction. The longer the interval, the higher the uncertainty. Subsequently, all measurements ($\Delta\hat{\mathbf{p}}$, $\hat{\mathbf{\Sigma}}$), along with predicted x and P, are processed in the update step, which estimates the final predicted position $\Delta\tilde{\mathbf{p}}$ and updates the state x and state covariance P within the Kalman filter.

We adopt the simplest Kalman filter to validate the effectiveness of our idea. Advanced variants (e.g., the Extended Kalman Filter) can be readily implemented by revising Eq. C, Eq. D, and Eq. E.

A.4. More Implementation Details

Event Module. As mentioned in the paper, our extracted patch has size $P_{evt} = 62$. The patches are encoded to reference and event feature maps $\mathbf{F}_{evt}$ and correlation maps $\mathbf{C}_{evt}$ with the same dimension $D_{F_{evt}} = 128$. These feature patches are extracted into a pyramid feature map with $L_{evt} = 2$ layers with size $\{62, 31\}$. In the LSTM displacement predictor, the feature map $\mathbf{F}_{evt}$ turns to a feature vector $\mathbf{f}_{evt}$ with dim $D_{f_{evt}} = 256$, which is obtained by merging a 128 dim hidden feature map patch in feature LSTM and a 256 dim hidden feature vector in displacement LSTM. In the uncertainty predictor, five convolution layers are applied to generate the uncertainty feature vector $\mathbf{f}_{uncert_j}$ with dim $D_{uncert} = 128$. We use 3 points to generate a parabola for uncertainty mapping, which maps $\{0, 0.9, 1\}$ to $\{0, 1, 10\}$. Detailed network information is listed in Tab. A.

Image Module. In our image module, the grayscale image is encoded to feature map $\mathbf{F}_{img}$ of $1/8$ image size with dim $D_{F_{img}} = 128$, and then correlated in $L_{img} = 4$ levels and extracted to patch with size $P_{img} = 7$. The point position (x, y) is embedded in 128 dim. The iterative prediction iterates 3 times. We also use 3 points to generate a parabola for uncertainty mapping, which maps $\{0, 0.5, 1\}$ to $\{0, 1, 10\}$. Detailed network information is listed in Tab. B.

B. More Experiment Results

B.1. More Quantitative Comparisons

Performance Experiment on Occlusion Data. We evaluate accuracy through δ_{avg}[5] on more methods. The result shows our multi-modality method outperforms all other baselines on both δ^{vis}_{avg}, δ^{occ}_{avg}, and δ_{avg}, and our event module with Kalman filter also outperforms all other event-based baselines, see Tab. C. The integration of a Kalman filter significantly enhances its performance in handling occluded points with minimal computational overhead, high-

	Layer	Data Size
Pyramid Feature Encoder	2× Conv2D 1×1×32	(62×62)×32
	Conv2D 7×7×64	(31×31)×64
	2× Conv2D 5×5×96	(23×23)×96
	2× Conv2D 5×5×128	(15×15)×128
	2× Conv2D 3×3×256	(5×5)×256
	2× Conv2D 3×3×384	(1×1)×384
	Up + Conv2D 1×1×384	(5×5)×384
	Conv2D 3×3×384	(5×5)×384
	Up + Conv2D 1×1×384	(15×15)×384
	Conv2D 3×3×384	(15×15)×384
	Up + Conv2D 1×1×384	(23×23)×384
	Conv2D 3×3×384	(23×23)×384
	Up + Conv2D 1×1×384	(31×31)×384
	Conv2D 3×3×384	(31×31)×384
	Up + Conv2D 1×1×384	(62×62)×384
	Conv2D 3×3×384	(62×62)×384
	2× Conv2D 3×3×384	(62×62)×384
	*Correlation	(62×62)×1
	*Conv2D 3×3×128	(62×62)×128
	Concatenate (1 + 128 + 128)	(62×62)×257
	Sample Pyramid	{(62×62), (31×31)}×257
	Squeeze and Concatenate	(31×31)×514
Uncertainty Predictor	2× Conv2D 1×1×128	(31×31)×128
	4× Conv2D 5×5×64	(15×15)×64
	2× Conv2D 3×3×64	(5×5)×64
	2× Conv2D 3×3×128	(1×1)×128
	Linear 2	2
	Softmax	1
	Parabola Function	1
LSTM Displacement Predictor	2× Conv2D 3×3×64	(15×15)×64
	2× Conv2D 3×3×128	(7×7)×128
	ConvLSTM 3×3×128	(7×7)×128
	3× Conv2D 3×3×256	(1×1)×256
	Concatenate Hidden Vector	512
	2×Linear 256	256
	Concatenate Hidden Vector	512
	Gate Layer 256	256
	Linear 2	2

Table A. **Detail architecture of event module.** * Layers are processed simultaneously.

	Layer	Data Size
Pyramid Encoder	Conv2D 7×7×64	(H/2×W/2)×64
	2× Conv2D 3×3×64	(H/2×W/2)×64
	2× Conv2D 3×3×96	(H/4×H/4)×96
	2× Conv2D 3×3×128	(H/8×H/8)×128
	2× Conv2D 3×3×128	(H/16×H/16)×128
	*Interpolate and Concatenate (64 + 96 + 128 + 128)	(H/8×H/8)×416
	Conv2D 3×3×256	(H/8×W/8)×256
	Conv2D 1×1×128	(H/8×W/8)×128
Correlation Pyramid	Input Feature Map	(H/8×W/8)×128
	Average Pool 2×2	(H/16×W/16)×128
	Average Pool 2×2	(H/32×W/32)×128
	Average Pool 2×2	(H/64×W/64)×128
	*Sample 7×7	4×(7×7)×128
	Correlate	4×(7×7)×1
	Concatenate	(7×7)×4
Prediction Head	Flatten, Embedding and Concatenate All Data (2×feature vectors + correlation map + embedded position + position) (2×128 + 7×7×4 + 128 + 2)	582
	MLP, depth 12, hidden dim 512	4

Table B. **Detail architecture of image module.** Layer with "*" have input from all previous 4 layers.

lighting the Kalman filter's robust capability in effectively addressing occlusion.

Performance Degradation Experiment on Occlusion Data. To assess the capability to handle occlusion, we conducted a series of experiments examining the impact of occlusion. We evaluated methods on synthetic and occluded versions of EC[16] and EDS[10] datasets outlined in Tab. D. The numerical decrease in performance from synthetic to occluded data provides insights into each method's robustness against occlusion. The results indicate that, compared with Deep-EV-Tracker[14], our event module demonstrates greater robustness in handling occlusions due to the larger pyramid patch and enhanced encoder. Moreover, a significant improvement is observed when incorporating the Kalman filter and the uncertainty predictor. The modules incorporating the Kalman filter exhibit less performance degradation across all data modalities. This finding further verifies the Kalman filter's strong capability in handling occlusions. Notably, EKLT[7] experiences the least performance degradation from EDS-syn to EDS-occ, which can be attributed to its already limited performance on EDS-syn.

Performance Experiment on Image Module. We also report the performance of the image-only model in Tab. E, noting that it is designed to be lightweight. The performance drop observed when adding the Kalman filter on EDS is attributed to the domain gap between the training data and the EDS dataset, particularly due to motion blur, which is not yet modeled in the training data, our MultiTrack dataset. Nevertheless, the significant performance gain from integrating the event module, image module, and Kalman filter demonstrates that the Kalman filter effectively fuses the two modalities, leveraging the strengths of both.

B.2. More Qualitative Comparison

Qualitative Comparison in Long-tern Stability. Since our method is trained on sequences of up to 23 frames, we conduct a long-term stability experiment to evaluate its ability to generalize to longer sequences. The experiment is performed on Boxes Rotation from EC[16] and EC-occ, consisting of 81 image frames and 368 event frames. Metrics are computed only on frames with ground truth. Tracking performance is reported using end-point error (EPE) and survival ratio, where a track is considered non-surviving if the L2 distance exceeds 32 pixels. From Fig. A (a) and (b), our method demonstrates robust long-term stability and consistently outperforms Deep-EV-Tracker[14]+KLT[19] by a substantial margin on both raw and occluded data.

Qualitative Comparison in Extreme Light Condition. To demonstrate the superiority of event cameras, we present a qualitative comparison with the state-of-the-art image-based method, CoTracker[12], in Fig B. The two scenes

Data	Methods	EC-occ			EDS-occ		
		$\delta_{avg}^{vis}\uparrow$	$\delta_{avg}^{occ}\uparrow$	$\delta_{avg}\uparrow$	$\delta_{avg}^{vis}\uparrow$	$\delta_{avg}^{occ}\uparrow$	$\delta_{avg}\uparrow$
Event	HASTE[3]	9.7	1.8	9.1	0.0	0.0	0.0
	EKLT[7]	21.7	9.9	20.6	16.9	6.3	16.3
	Deep-EV-Tracker[14]	26.8	20.2	26.3	31.3	21.3	30.9
	Ours(Event)	29.3	23.1	28.7	35.1	24.6	34.6
	Ours(Event+Kalman)	**30.5**	**24.1**	**29.8**	**36.7**	**25.9**	**36.2**
Event + Image	FF-KDT[21]	28.2	14.5	27.1	24.0	11.5	23.4
	Deep-EV-Tracker[14] + KLT[19]	32.1	5.8	30.3	35.0	10.7	33.7
	Ours(Event+Image)	37.2	11.4	35.5	43.1	18.6	41.6
	Ours(Event+Image+Kalman)	**44.5**	**28.5**	**43.4**	**52.0**	**26.8**	**50.6**

Table C. **Performance evaluation on synthetic data with occlusion.**

Expect FA↑							
Data	Methods	EC			EDS		
		EC-syn	EC-occ	Dcre.(%)↓	EDS-syn	EDS-occ	Dcre.(%)↓
Event	HASTE[3]	0.379	0.341	10.0†	0.043	0.085	-96.9†
	EKLT[7]	0.801	0.370	53.8	0.353	0.324	**8.3**
	Deep-EV-Tracker[14]	0.812	0.423	47.9	0.454	0.289	36.2
	Ours(Event)	**0.830**	0.522	37.2	**0.461**	0.343	25.5
	Ours(Event+Kalman)	0.828	**0.527**	**36.3**	0.459	**0.349**	23.9
Event + Image	FF-KDT[21]	0.846*	0.401*	52.6*	0.434*	0.212*	51.2*
	Deep-EV-Tracker[14] + KLT[19]	0.738	0.314	57.4	0.504	0.274	45.6
	Ours(Event+Image)	0.780	0.446	42.8	0.532	0.292	45.0
	Ours(Event+Image+Kalman)	**0.845**	**0.572**	**32.4**	**0.550**	**0.356**	**35.2**

Table D. **Performance degradation on occluded data from non-occluded data.** † HASTE[3] exhibits such poor performance that its decreased value becomes meaningless. * FF-KDT[21] can only produce estimates at image timestamps, whereas Deep-Ev-Tracker[14] and our method generate significantly more predictions; therefore, it is excluded from our comparison.

Data	Methods	EC			EDS		
		FA↑	Exp FA↑	$N_p\uparrow$	FA↑	Exp FA↑	$N_p\uparrow$
I	KLT[19]	0.734	0.729	24	0.588	0.497	75
	Ours(I)	0.778	0.772	24	**0.633**	**0.524**	75
	Ours(I w. K)	**0.784**	**0.778**	24	0.619	0.511	75

Table E. **Performance evaluation on EC and EDS.**"I" denotes image. N_p denotes the number of predictions per second of data.

represent overexposure and underexposure scenarios, where the event camera remains unaffected. In contrast, Co-Tracker fails completely, while our method continues to operate, highlighting the essential role of event cameras in extreme conditions alongside image data. Both scenes are from DSEC[8], with the first using raw data and the second manually adjusted to simulate extreme underexposure.

More Qualitative Comparison. In this section, we present more qualitative results (Fig. C). Both scenes are from EDS[10]. The first scene features a sofa with low texture, which challenges the event tracker, while the second scene suffers from insufficient lighting, posing difficulties for the image tracker. The results demonstrate that our method achieves superior performance, effectively addressing the limitations of both trackers through the integration of the Kalman filter.

B.3. Uncertainty Visualization

To evaluate the effectiveness of uncertainty training, we visualize the uncertainty during tracking in Fig. A (c). The experiment is conducted on Boxes Rotation from EC-occ, where we select a track that includes both visible and occluded states. The results show increased uncertainty during challenging scenarios (e.g., occlusion), demonstrating the effectiveness of our uncertainty modeling.

B.4. More Runtime Analysis

Runtime and Accuracy Performance Additional experiments related to runtime are presented in Fig. D. The statistics of other methods are from[14], and our experiment is conducted on the same type of GPU. The real-time factor is simply the computation time divided by input data interval[14]. Our module does not have dependencies between event patches, supporting fully parallel calculation. The perfect parallelization results were obtained by tracking one feature with one thread across multiple event frames. On the contrary, the Deep-EV-Tracker requires information exchange and motion consistency across tracks, thus not supporting perfect parallel calculation.

It can be observed from Fig. D that:

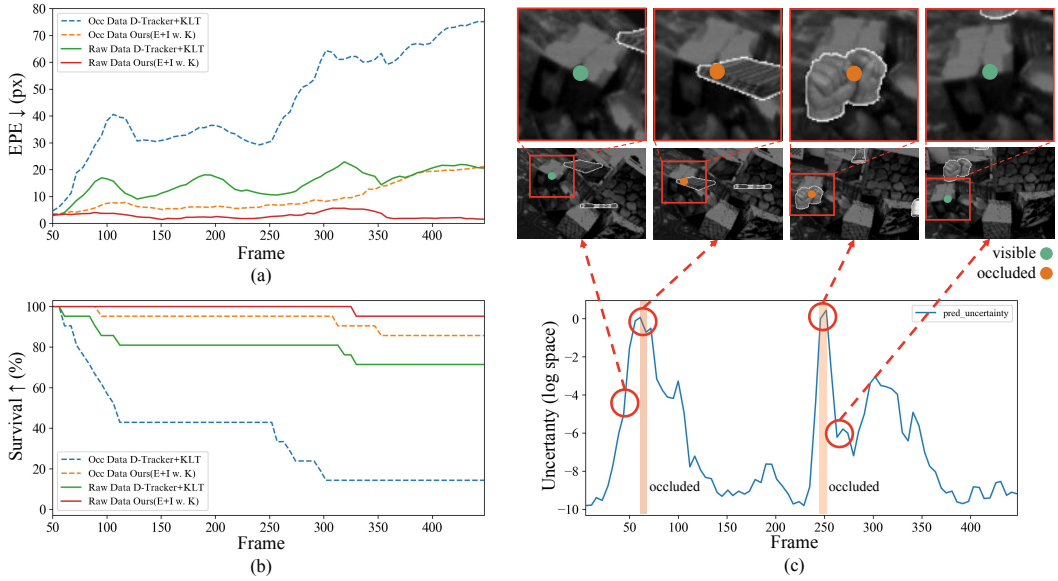

Figure A. **Long-term stability experiment and uncertainty visualization.** (a) Tracking quality (EPE). (b) Survival ratio. (c) Uncertainty.

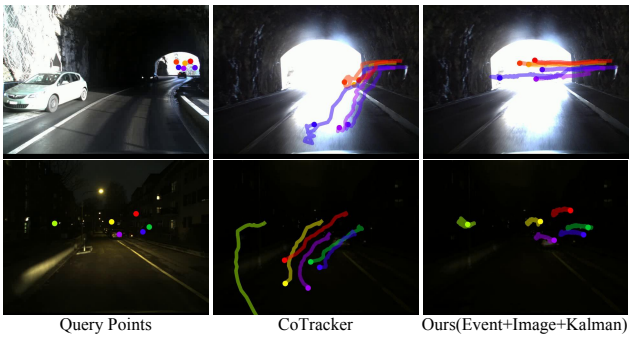

Figure B. **Qualitative comparison with pure image-based method in extreme light condition.**

Runtime(ms)↓				
Experiment	Method	EC	EDS	Paras
Feature Pyramid (Ours Event)	On	7.95	8.70	33.1M
	Off	5.88	6.91	29.3M
LSTM Module (Ours Event)	Feature+Displace	7.95	8.70	33.1M
	Feature	6.55	7.79	32.7M
	Displacement	6.54	8.16	31.9M
	Off	6.04	6.75	31.6M
Correlation Vector (Ours Event)	Feature Map Center	7.95	8.70	33.1M
	U-Net Bottleneck	8.17	9.04	35.7M

Table F. **Ablation settings runtime.**

(1) Our event module achieves the best runtime performance and accuracy on both datasets.

(2) With the assistance of the image module and the Kalman filter, our method achieves almost the same runtime performance as Deep-EV-Tracker[14] while getting much better tracking performance.

(3) In practice, our methods without perfect parallelization still exhibit a superior runtime compared to most existing methods. Further optimization of parallel computation could enhance performance.

Runtime of Ablation Settings To analyze the runtime of each component under different ablation settings, we report the runtime results in Table F. Disabling the feature pyramid reduces runtime by 2ms but leads to significant performance degradation. Removing each LSTM module saves approximately 1ms, at the cost of substantial accuracy loss. Using the feature map center as the correlation vector improves both runtime and performance.

B.5. Event Module Reference Frame Experiment

Our event module uses a greyscale image as the reference frame to ensure that both the event and image modules track the same feature points in the same reference frame, eliminating reference frame bias. Although tracking with the event module requires additional spatial alignment between the two cameras, our primary contribution lies in integrating both event and image data for tracking. Thus, spatial alignment should be considered a preliminary step rather than an additional burden. Moreover, using different modality reference frames requires synchronization at a specific reference timestamp, which is incompatible with our asynchronous tracking framework. We must note that previous methods, such as EKLT[7] and Deep-Ev-Tracker[14], also use grayscale images as reference frames, as image data provides richer information. In contrast, event data is limited in low-texture or stationary scenes.

However, our event module can also track using pure

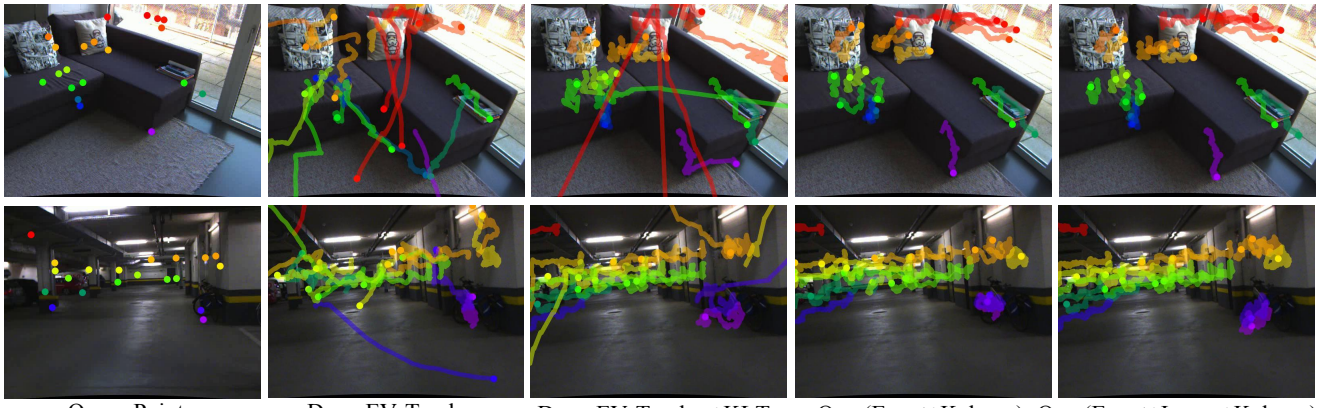

Figure C. **Qualitative comparison.**

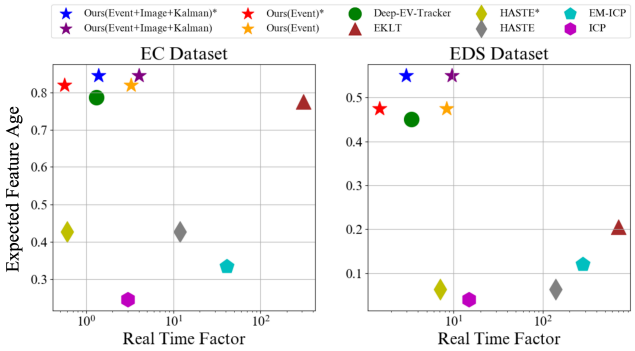

Figure D. **Runtime performance in terms of expected feature age with the real-time factor.** Methods that exhibit superior performance and computational efficiency tend to position closer to the upper left corner. The statics of EKLT[7], HASTE[3], EM-ICP[25], ICP[4] are from[14]. Deep-Ev-Tracker(E2VID)[15] has the same architecture as Deep-Ev-Tracker, so they should have the same runtime performance. Methods with "*" assume perfect parallelization for processing all feature tracks. Our method without "*" is implemented with unoptimized parallelization. The real-time factor is simply the computation time divided by input data interval[14].

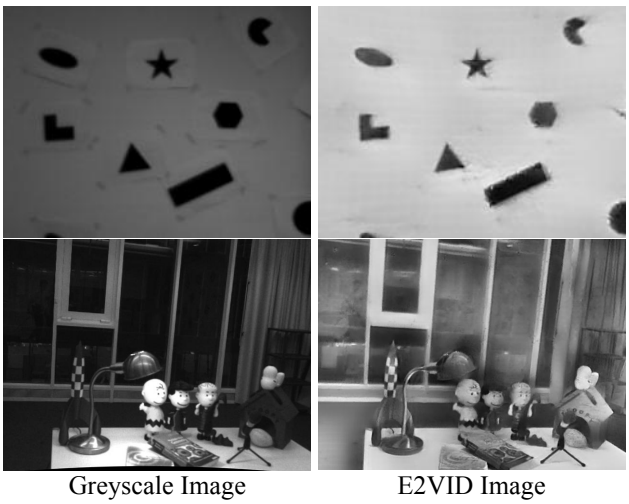

Figure E. **Comparison of greyscale image and E2VID[17, 18] image.**

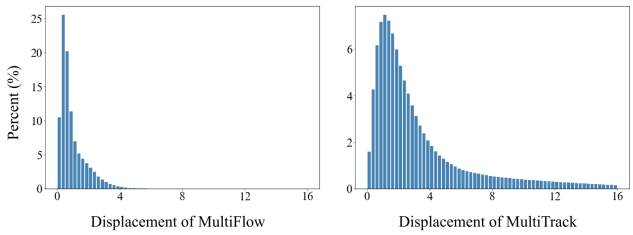

Figure F. **Ground truth displacement distribution of MultiFlow[9] and MultiTrack.**

event data. By employing event-to-frame reconstruction with E2VID[17, 18], as shown in Fig. E, we can replace the greyscale reference image with the reconstructed image[15].

C. More Details about MultiTrack, EC-occ and EDS-occ

C.1. Displacement Distribution of MultiFlow and MultiTrack

It is established that trajectory error tends to increase with greater displacement in the trajectory[9]. Based on this, we analyzed the ground truth displacement statistics between adjacent event frames to illustrate the level of difficulty, see Fig. F. The results indicate that MultiFlow[9] has most displacements below 4 pixels, whereas MultiTrack exhibits significantly larger displacements and point occlusions. These factors increase complexity, suggesting that

Expect FA↑				
Dataset	EC	EC-occ	EDS	EDS-occ
MultiTrack	**0.800**	**0.510**	**0.412**	**0.283**
MultiFlow[9]	0.777	0.453	0.405	0.269

Table G. **Training dataset comparisom.**

Expect FA↑				
Methods	EC	EC-syn	EDS	EDS-syn
Deep-EV-Tracker[14]	0.787	0.812	0.451	0.454
Ours(Event)	**0.819**	**0.830**	**0.474**	**0.461**

Table H. **Performance evaluation on real event data and synthetic event data.**

MultiTrack presents a greater challenge than MultiFlow.

Our investigation suggests that datasets with appropriate difficulty levels are essential for unlocking the potential of learning-based modules. Therefore, MultiTrack plays a crucial role in addressing the shortage of event-based feature tracking datasets by offering adjustable synthetic parameters to control difficulty levels.

C.2. Data Quality of MultiFlow and MultiTrack

In this experiment, we train our event module and Deep-EV-Tracker[14] respectively on the MultiTrack and MultiFlow[9] datasets and evaluate the average performance of the two methods on the two datasets. Although we view MultiTrack as a supplement to MultiFlow rather than a complete replacement, the results in Tab. G show that our MultiTrack dataset still outperforms the MultiFlow dataset on average metrics. This suggests that MultiTrack has better data quality and can unleash the full potential of modules.

C.3. Synthetic Data Quality of EC-occ and EDS-occ

Some hold that synthetic events exhibit significant differences from events captured by real event cameras. So we experiment by also generating events from interpolated but not occluded images, EC-syn and EDS-syn, to measure the real events and our synthetic events. Tab. H shows the slight difference in evaluation metric between real and synthetic data, which proves that the synthetic data could be credible in evaluation.

C.4. Visualization Examples of MultiTrack, EC-occ and EDS-occ

We provide some visualization examples in Fig. G. The ground truth point tracks are depicted as trajectories, with green indicating that the points are visible and purple indicating that the points are occluded at that moment.

C.5. Dataset License

In this section, we present the licensing information for the datasets we used, as well as for our generated datasets, including EC-occ, EDS-occ, and the MultiTrack dataset. The data we use are all from existing datasets, among them Google Scanned Objects[6]is under Creative Commons Attribution 4.0 License[1] which are free to share and adapt, while Flickr30k[24] and COCO2014[13] are under Flickr Terms of Use[2] which is available for researchers and educators who wish to use the dataset for non-commercial research or educational purposes. According to these, the EC-occ and EDS-occ are under Creative Commons Attribution 4.0 License, free to use by anyone, while using the open-source version of MultiTrack must abide by the Flickr Terms of Use. However, our dataset generator could replace the Flickr30k or COCO2014 with other image datasets with a Creative Commons Attribution 4.0 License, producing data friendly to commercial use.

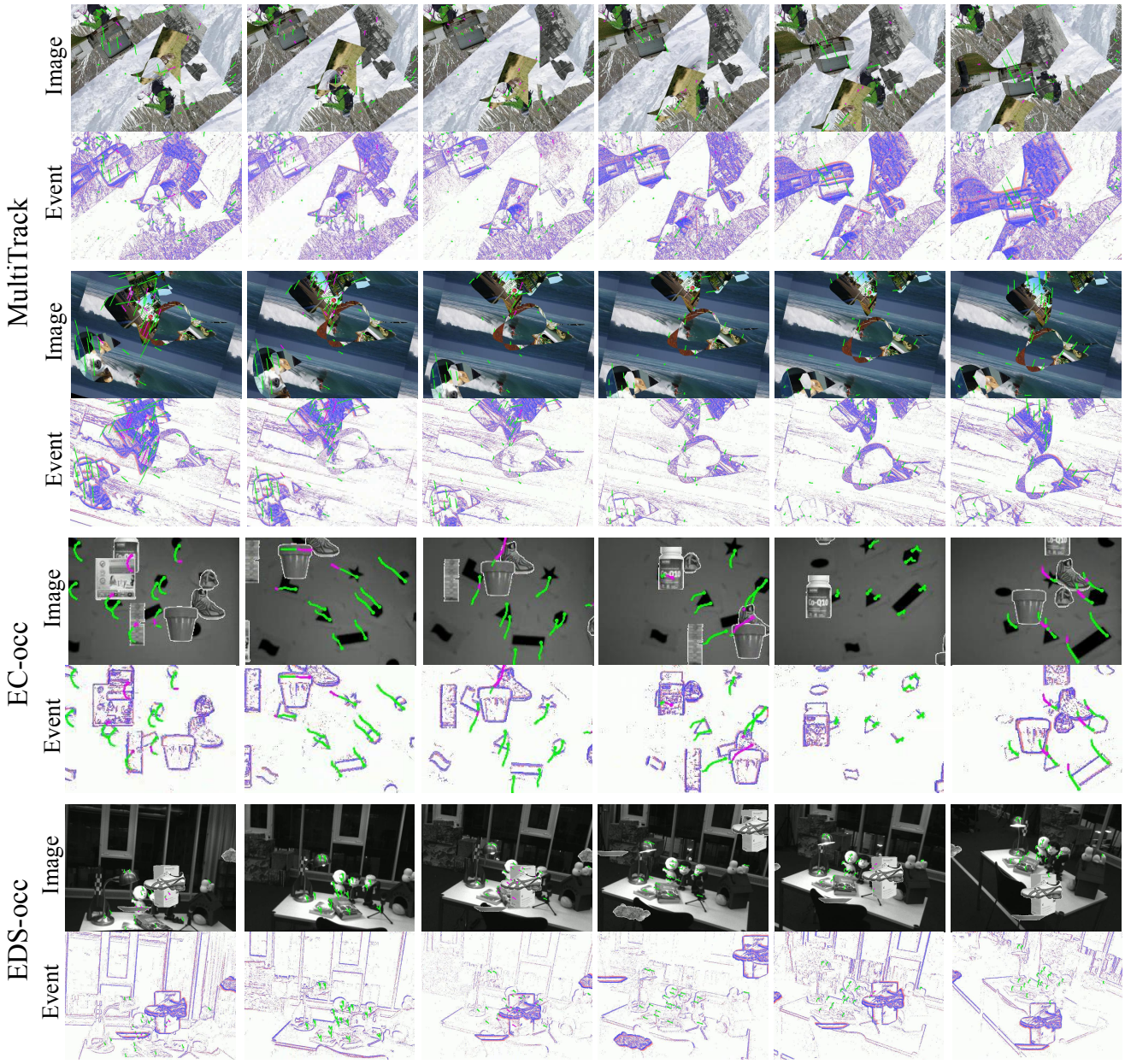

Figure G. **Visualization examples of MultiTrack, EC-occ and EDS-occ.**

References

[1] Creative Commons Attribution 4.0 License. Accessed: 2024-05-13. 7

[2] Flickr Terms of Use. Accessed: 2024-05-13. 7

[3] Ignacio Alzugaray and Margarita Chli. Haste: multi-hypothesis asynchronous speeded-up tracking of events. *British Machine Vision Conference*, 2020. 4, 6

[4] P.J. Besl and Neil D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, 1992. 6

[5] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adrià Recasens, Lucas Smaira, Yusuf Aytar, João Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *CoRR*, abs/2211.03726, 2022. 2

[6] Laura Downs, Anthony Francis, Nate Koenig, Brandon Kinman, Ryan Hickman, Krista Reymann, Thomas Barlow McHugh, and Vincent Vanhoucke. Google scanned objects: A high-quality dataset of 3d scanned household items. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2553–2560. IEEE, 2022. 7

[7] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. Eklt: Asynchronous photometric feature tracking using events and frames. *International Journal of Computer Vision*, 128:1–18, 2020. 3, 4, 5, 6

[8] 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

[9] Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. Dense continuous-time optical flow from event cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–12, 2024. 6, 7

[10] Javier Hidalgo-Carrio, Guillermo Gallego, and Davide Scaramuzza. Event-aided direct sparse odometry. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2022. 3, 4

[11] Rudolf E. Kálmán and Richard S. Bucy. New results in linear filtering and prediction theory. *Journal of Basic Engineering*, 83:95–108, 1961. 1

[12] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. Cotracker: It is better to track together. *CoRR*, abs/2307.07635, 2023. 3

[13] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. 7

[14] Nico Messikommer, Carter Fang, Mathias Gehrig, and Davide Scaramuzza. Data-driven feature tracking for event cameras. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5642–5651, 2023. 1, 3, 4, 5, 6, 7

[15] Nico Messikommer, Carter Fang, Mathias Gehrig, Giovanni Cioffi, and Davide Scaramuzza. Data-driven feature tracking for event cameras with and without frames. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–12, 2025. 6

[16] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbrück, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *CoRR*, abs/1610.08336, 2016. 3

[17] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. High speed and high dynamic range video with an event camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 6

[18] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. Events-to-video: Bringing modern computer vision to event cameras. *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 6

[19] Jianbo Shi and Tomasi. Good features to track. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994. 3, 4

[20] Lin Wang, I.S. Mohammad Mostafavi, Yo-Sung Ho, and Kuk-Jin Yoon. Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10073–10082, 2019. 1

[21] Xiangyuan Wang, Huai Yu, Lei Yu, Wen Yang, and Gui-Song Xia. Towards robust keypoint detection and tracking: A fusion approach with event-aligned image features. *IEEE Robotics and Automation Letters*, 2024. 4

[22] Greg Welch. An introduction to the kalman filter. In *International Conference on Computer Graphics and Interactive Techniques*, 1995. 1

[23] Yue Wu, Jingao Xu, Danyang Li, Yadong Xie, Hao Cao, Fan Li, and Zheng Yang. Flytracker: Motion tracking and obstacle detection for drones using event cameras. *IEEE INFOCOM*, 2023. 1

[24] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *Transactions of the Association for Computational Linguistics*, 2:67–78, 2014. 7

[25] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based feature tracking with probabilistic data association. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4465–4470, 2017. 6