Adaptive Spatial-Temporal Window: Unlocking the Potential of Event Cameras in Heterogeneous Velocity Scenarios

Supplementary Material

S1. Computation Complexity Analysis and Vectorized Implementation

The key step affecting the efficiency of Algorithm 1 is Line 6, where events are selected from the raw event stream $\mathcal{E}$ based on the dynamically determined time window length Δt_{ij} for each spatial patch. To clarify the process, we define the inputs and outputs of this operation:

- **Input:** An input event stream $\mathcal{E}$ containing N events (inherently sorted by timestamp t), the spatial patch size s, the global reference time T_{global}, and a 2D lookup table $\mathbf{T}_{\text{start}} \in \mathbb{R}^{\frac{H}{s} \times \frac{W}{s}}$ storing the calculated start time $(T_{\text{global}} - \Delta t_{ij})$ for each patch.
- **Output:** The partitioned events P_{cand} containing K selected events, which must remain sorted by timestamp for downstream processing.

The most straightforward implementation is patch-wise iteration. For each patch (a total of $P = \frac{W \times H}{s^2}$ patches), all N events are traversed to determine if they belong to the current patch and fall within its specific time window. The collected K events from all patches are then concatenated and re-sorted by timestamp. The computational complexity of this naive approach is $\mathcal{O}(P \cdot N + K \log K)$, which becomes prohibitively expensive for high-resolution sensors (large P).

A more efficient hash-based grouping approach converts the N events into a Pandas DataFrame, utilizing hash mapping to preassign events into P buckets. This eliminates the need to traverse the N events P times, reducing the complexity to $\mathcal{O}(N + K \log K)$. However, the final temporal sorting step ($\mathcal{O}(K \log K)$) remains a bottleneck.

To completely eliminate explicit patch loops and final sorting, we adopt a fully vectorized implementation. Specifically, we first vectorize the computation of each event's patch index, then use vectorized look-ups to retrieve the corresponding time window range, and determine whether to retain the event by comparing it with its timestamp. Through this fully vectorized paradigm, the total computational complexity is strictly bounded to $\mathcal{O}(N)$. The runtime is purely proportional to the number of incoming events N and is entirely independent of the number of spatial patches P, making it highly scalable for diverse spatial resolutions.

We further evaluate the memory consumption and computational overhead of different partitioning strategies, as shown in Tab. 1. When processing 4.75 M events, Event Lifetime [7] and Adaptive Global Decay [8] take 267.80 s and 79.60 s, respectively, whereas our ASTW strategy requires only 35.30 s. ASTW achieves an average latency of 7.43 μs/event, which is on the same order of magnitude as TORE [1], and speeds up processing by approximately 7.6 $\times$ and 2.3 $\times$ compared with Event Lifetime and Adaptive Global Decay, respectively. In terms of memory, ASTW increases consumption by about 23% compared with other strategies, which is acceptable. The additional memory is primarily used to store intermediate results of event density and time windows.

S2. Parameter Settings for Partitioning Strategies

Different partitioning strategies rely on specific hyperparameters, and varying their settings can substantially affect the final performance. For fair comparison, we either adopt the parameter settings recommended in the original papers, or perform hyperparameter search to obtain their optimal values. This section presents the hyperparameters involved in each strategy and the final selected settings.

For the Fixed Time Window strategy [6], its hyperparameter is the time window length. We adopt commonly used configurations for experiments, and results show that a time window of 100 ms achieves the best performance on both the Gen1 and EventVOT datasets, while either shorter or longer windows lead to performance degradation. Moreover, regardless of parameter adjustment, its performance consistently falls short of ASTW, further confirming the superiority of the ASTW strategy.

For the Fixed Number of Events strategy [10], the hyperparameter is the number of events. Similar to the Fixed Time Window strategy, experiments also show that both a smaller and a larger number lead to performance degradation. Experiments indicate that the optimal number is 30,000 on the Gen1 dataset and 70,000 on the EventVOT dataset.

Adaptive Temporal Sampling (ATS) [4] dynamically adjusts the time window length so that the number of contained events approximates an adaptive threshold. This threshold is determined jointly by the preset base event threshold θ and the time feedback coefficient η. Following the original recommendation, we set $\theta = 5000$, $\eta = 0.05$ for the Gen1 dataset. Since the EventVOT dataset has the same resolution as the 1Mpx dataset, we adopt similar parameters $\theta = 25000$, $\eta = 0.20$.

Adaptive Event Conversion (AEC) [9] uses an adaptive queue. By comparing the number of events within a short

Table 1. Comparison of latency and memory usage for different strategies.

Partitioning Strategy	Number of Events	Total Time (s)	Average Latency (μs/event)	Memory Usage (MiB)
Fixed Time Winodw [6]	4.75M	1.22	0.25	823.70
Adaptive Global Decay [8]	4.75M	79.60	16.76	837.70
TORE [1]	4.75M	9.70	2.04	837.40
Event Lifetime [7]	4.75M	267.80	56.38	796.10
ASTW (ours)	4.75M	35.30	7.43	1032.40

period t_s to a threshold L, it first determines whether the event block is sparse or dense. Sparse event blocks continue to accumulate, while dense event blocks are directly added to the queue. We adopt the parameter settings recommended in the original paper: set $ts = 6.25$ms, and choose the threshold $L = 10000$ on the Gen1 dataset and $L = 100000$ on the EventVOT dataset.

Adaptive Global Decay [8] dynamically computes event weights based on global event activity and applies a threshold w to perform event partitioning: when an event's weight decays below w, it is regarded as inactive. Experiments show that $w = 0.01$ yields the best performance on the Gen1 dataset, while $w = 0.02$ works best on the EventVOT dataset.

SpikeSlicer [2] employs a spiking neural network (SNN) as a dynamic event trigger to achieve adaptive slicing. Specifically, the raw event stream is first divided into N discrete Event Cells, each spanning a temporal interval of δt, and they are sequentially fed into the SNN. A slice is generated whenever the output neuron of the SNN fires. The SNN automatically learns optimal slicing timing through a Feedback–Update co-training strategy. For both the Gen1 and EventVOT datasets, we adopt the same configuration using the SpikeSlicer-Base model, where the number of Event Cells is set to $N = 20$, and $\delta t = 5$ms.

TORE [1] maintains a queue of length k at each pixel location. Experiments show that the optimal setting is $k = 3$ on the Gen1 dataset and $k = 4$ on the EventVOT dataset.

Event Lifetime [7] applies RANSAC to perform planar fitting on the Surface of Active Events (SAE) within an $N \times N$ neighborhood, estimating the velocity and lifetime of each event. Its parameters include the spatial size N, the inlier threshold μ, and the percentage of outliers ϵ. Following the configuration in the original paper, we set $N = 5$, $\mu = 1 \times 10^{-4}$, $\epsilon = 0.4$.

S3. HetVel Dataset

We set up a dual-modality data acquisition device to capture data in HVS. The 3D schematic and the physical photograph are shown in Fig. 1. Event modality is used to capture fast and subtle motion, while frame modality is used to capture texture and color information, and as an auxiliary modality to achieve semi-automatic data annotation. Fig. 2 shows the complete acquisition workflow of the dual-modality dataset, including the design of the optical path, circuit, and algorithm. The data acquisition equipment uses the Teledyne FLIR Blackfly-S as frame-based camera (RGB camera) and the Prophesee EVK-4 as event camera, with resolutions of 1440×1080 and 1280×720 respectively, a beam splitter to ensure that both cameras have a common field of view, and a microcontroller generates a 100 Hz square wave signal to trigger each camera for capture. When processing the dual-modality data, temporal synchronization is performed according to the trigger timestamps. For spatial registration, it is necessary to first reconstruct the event data into gray frame images through E2VID [10], and then spatially register the reconstructed images with RGB images using GeoFormer [5], and calculate the homography matrix. For semi-automatic annotation, we label RGB images using SAM [3], generate annotation information such as mask and bounding box with manually selected points as prompt, and then map them into event domain through homography matrix.

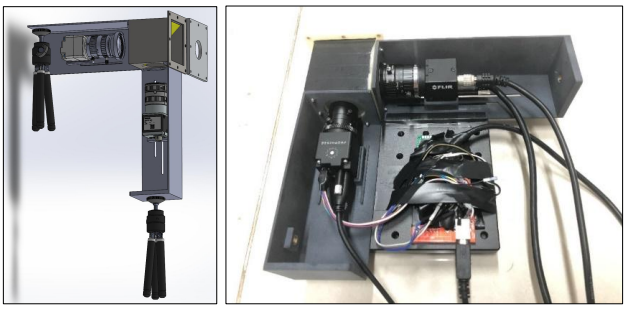

Figure 1. 3D schematic of the optical path and physical photograph of the capture device.

We propose a new dataset, termed HetVel, which consists of 33 videos of RGB-Event dual-modality data, along with high-precision segmentation masks and bounding boxes at 100 frames per second (fps). The HetVel dataset contains common HVS from everyday life, such as varying vehicle speeds in traffic and speed-changing balls in sports contexts. Fig. 3 presents visualizations of selected segments from the HetVel dataset. The dataset involves a variety of object categories, such as pedestrians, vehicles,

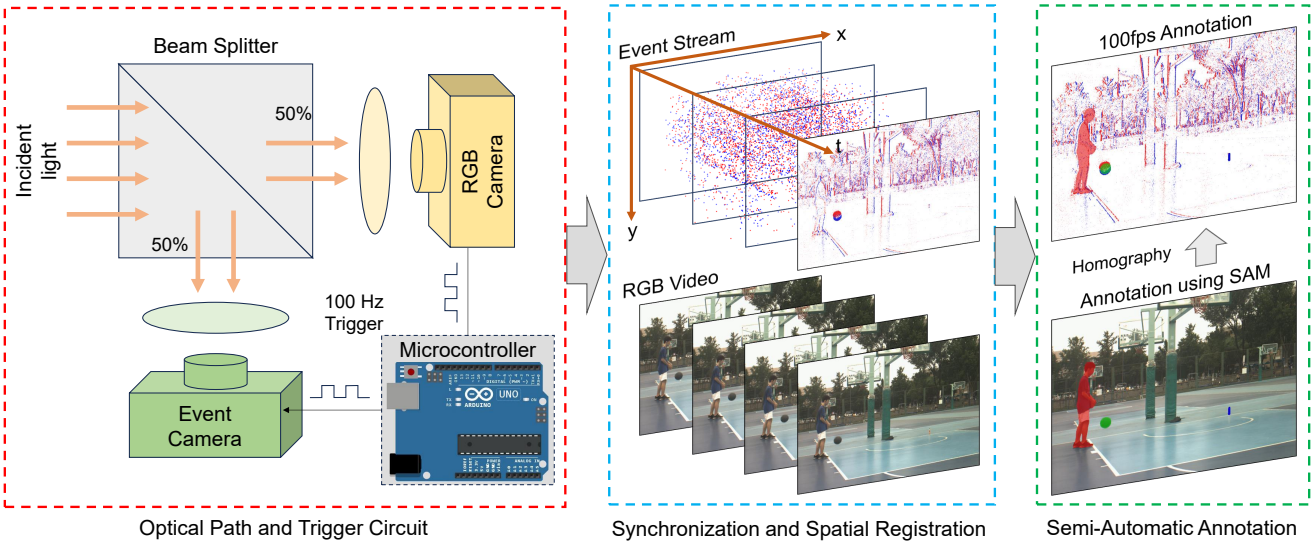

Figure 2. Comprehensive acquisition workflow for our dual-modality dataset, including optical path configuration, trigger circuitry, preprocessing, and annotation algorithm.

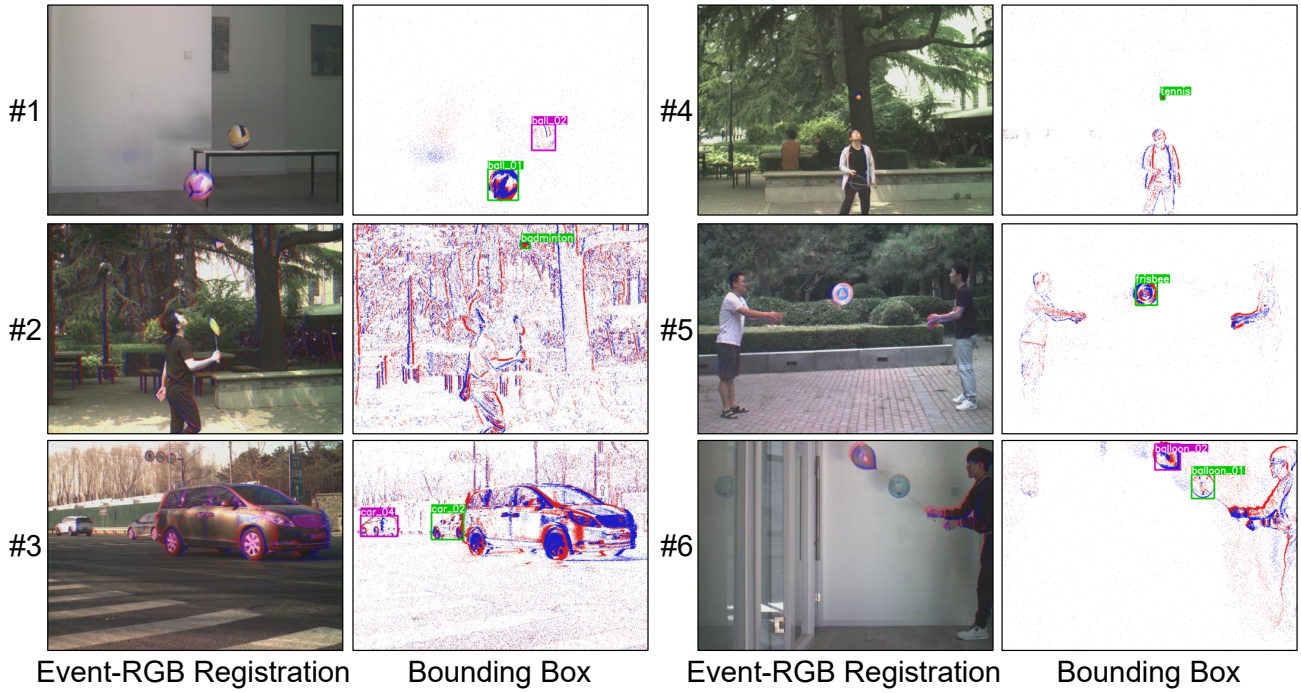

Figure 3. Visualization of the HetVel dataset. Relevant identifiable information is masked to ensure anonymity.

basketball, etc. Recording environments include roads, stadiums, indoors and other environments. Our dataset encompasses multiple attributes, including camera motion, lighting variations, partial/full occlusion, out-of-field, scale variation, background clutter, fast motion, slow motion, complete stillness, and small objects.

We analyze the dataset's velocity and occlusion distributions. Object speeds range from 0 to 4167 px/s (mean: 450 px/s; standard deviation: 459px/s), with 51.0% below 300 px/s and 34.9% between 300–900 px/s. Occlusion levels range from 0 to 99.96% (mean: 26.66%; standard deviation: 24.03%), with 35.6% of instances ex-

Table 2. Sensitivity of ASTW strategy to patch size.

Patch Size	1	2	4	8	16	32	64
mAP	49.9	50.4	**50.6**	50.3	50.0	49.3	48.9

Table 3. Sensitivity of ASTW strategy to information constant.

Information Constant	1.0	1.3	1.5	1.7	2.0
mAP	50.4	50.5	50.3	**50.6**	50.0

Table 4. Sensitivity of ASTW strategy to reference time window.

Reference Time Window	50	100	250	500	750
mAP	49.4	50.1	**50.6**	**50.6**	50.0

Table 5. Sensitivity of the ASTW strategy to $(\Delta t_{\min}, \Delta t_{\max})$.

$(\Delta t_{\min}, \Delta t_{\max})$	(2.5, 250)	(5, 250)	(10, 250)	(25, 150)
mAP	50.0	50.1	**50.6**	49.5
$(\Delta t_{\min}, \Delta t_{\max})$	(25, 200)	(50, 150)	(50, 200)	(75, 150)
mAP	49.6	47.8	48.7	48.8

ceeding 30% occlusion and 16.3% exceeding 50%. The dataset is available at https://pan.baidu.com/s/12ReqmcXwsR8eBrCw55Namw?pwd=iag2.

S4. Parameter Sensitivity Analysis

This section analyzes the sensitivity of ASTW strategy to its relevant parameters. The parameters include patch size s, information constant γ, reference time window length Δt_{ref}, and the upper and lower bounds of the time window $(\Delta t_{\min}, \Delta t_{\max})$. The parameter–sensitivity analysis is conducted on the object detection task. The following results are based on the Gen1 dataset, with ResNet-50 used as the backbone.

Regarding patch size s, Tab. 2 shows that a patch size of 4 achieves the best performance. When s is set to 2 and 1, mAP decreases by 0.2 and 0.7, respectively. We attribute this degradation to the fact that overly small patch sizes lead to statistical instability of event density D_{ij} and increase sensitivity to local noise (e.g., hot-pixel noise). When the patch size is increased to 8 and 16, mAP decreases by 0.3 and 0.6, respectively. This is because overly large patch sizes weaken spatial locality, causing the partitioning strategy to become more global.

Information constant γ reflects the desired amount of information contained within each patch. We conduct experiments within the range of 1.0-2.0, and the results are presented in Tab. 3. The mAP values remain relatively stable, indicating that the ASTW strategy is insensitive to this parameter, so we use 1.7.

Regarding the reference window length Δt_{ref}, Tab. 4 shows that the performance is stable within 250-500 ms; shorter or longer windows lead to performance degradation. We use a time window length of 250ms. We believe that shorter windows will reduce the robustness of the partitioning strategy to rapidly changing motion, while longer windows will increase the computational burden.

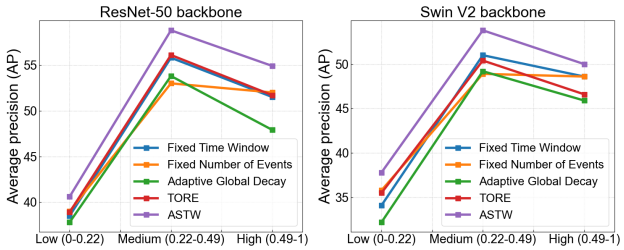

Figure 4. Detection performance under different contrasts.

Regarding the settings of $\Delta t_{\min}$ and $\Delta t_{\max}$, we test multiple combinations. The results in Tab. 5 show that the combination $\Delta t_{\min} = 10$ms and $\Delta t_{\max} = 250$ms yields the best results. When $\Delta t_{\min}$ and $\Delta t_{\max}$ become increasingly close, we observe a performance drop. This is mainly because spatial locality is weakened, causing the partitioning paradigm to degenerate toward a global scheme, which in turn degrades overall performance.

S5. Temporal Discrepancy and High-Percentile Window

This section evaluates the impact of temporal discrepancy and high-percentile windows on task performance. According to Algorithm 1 in the main text, since the partitioning of the ASTW strategy is generated adaptively, it cannot be guaranteed that the end timestamps of partitioned events are perfectly aligned with the label timestamps. There exists a random temporal discrepancy between them. Our minimum time-step strategy (Algorithm 1) minimizes misalignment by updating T_{global} at the finest temporal scale ($<$10 ms). Artificially shifting timestamps by 5 and 10 ms results in negligible mAP drops (50.6 $\rightarrow$ 50.4, 50.3), confirming that minor temporal discrepancy does not compromise result validity.

In Algorithm 1 of the main text, we adopt a minimum time-step strategy, where the minimum value among the time windows of all patches (equivalent to 0% window) is selected as the step size. Different percentile windows provide a trade-off between accuracy and computation. Our experiments show that using a 20% window (sorted from small to large) results in only a minor performance drop ($-$0.5 mAP, $<$1%), while reducing redundant updates by 28%. In contrast, a 50% window causes a substantial degradation ($-$2.3 mAP, 4.5%). These results offer a practical guideline for balancing accuracy and redundant updates.

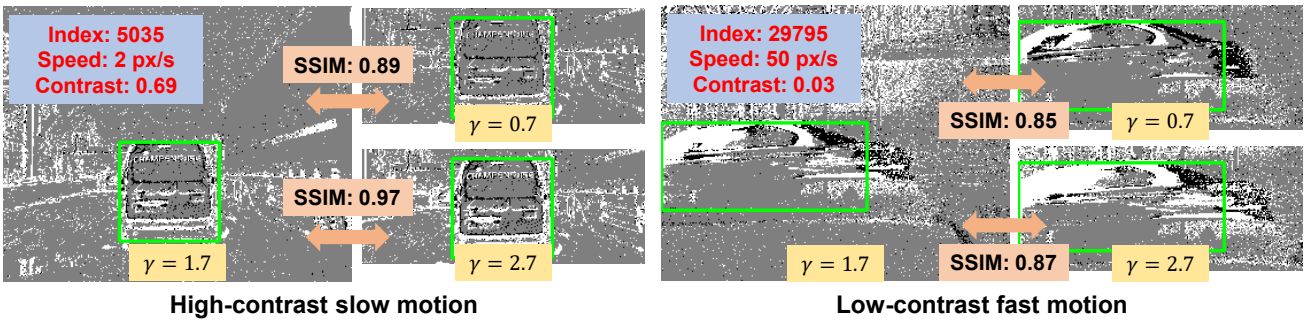

Figure 5. High-contrast slow motion and low-contrast fast motion.

S6. Constant Edge Contrast Assumption

In the main text, we adopt the constant edge contrast assumption to simplify the analysis and derive Eq. 3. In this section, we further evaluate the performance of the ASTW strategy under different edge contrast conditions. We first compute the average contrast of the object within each ground-truth bounding box and then uniformly divide the samples into three groups: low, medium, and high contrast. Fig. 4 shows that the proposed ASTW strategy achieves consistent performance improvements across different contrast levels.

Fig. 5 visualizes high-contrast slow motion and low-contrast fast motion sequences from Gen1, confirming the effectiveness of ASTW. We further evaluate the sensitivity of γ using the Structural Similarity Index (SSIM), where consistently high scores ($>$0.85) indicate strong robustness to γ.

References

[1] R Wes Baldwin, Ruixu Liu, Mohammed Almatrafi, Vijayan Asari, and Keigo Hirakawa. Time-ordered recent event (tore) volumes for event cameras. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(2):2519–2532, 2022. 1, 2

[2] Jiahang Cao et al. Spiking neural network as adaptive event stream slicer. In *NeurIPS*, 2024. 2

[3] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023. 2

[4] Jianing Li, Jia Li, Lin Zhu, Xijie Xiang, Tiejun Huang, and Yonghong Tian. Asynchronous spatio-temporal memory network for continuous event-based object detection. *IEEE Transactions on Image Processing*, 31:2975–2987, 2022. 1

[5] Jiazhen Liu and Xirong Li. Geometrized transformer for self-supervised homography estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9556–9565, 2023. 2

[6] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5419–5427, 2018. 1, 2

[7] Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego, and Davide Scaramuzza. Lifetime estimation of events from dynamic vision sensors. In *2015 IEEE international conference on Robotics and Automation (ICRA)*, pages 4874–4881. IEEE, 2015. 1, 2

[8] Urbano Miguel Nunes, Ryad Benosman, and Sio-Hoi Ieng. Adaptive global decay process for event cameras. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9771–9780, 2023. 1, 2

[9] Yansong Peng et al. Better and faster: Adaptive event conversion for event-based object detection. In *AAAI*, 2023. 1

[10] 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*, 43(6):1964–1980, 2019. 1, 2