SI: STELLA: A modular framework for SpatioTemporal Event-based Lagrangian particLe trAcking

Sebastian Sachs and Christian Cierpka
Institute of Thermodynamics and Fluid Mechanics and Institute of Micro- and Nanotechnologies, Technische Universität Ilmenau, 98683 Ilmenau, Germany

Steffen Jung, Max Kahl, and Margret Keuper
Data and Web Science Group, University of Mannheim, 68161 Mannheim, Germany and Saarland Informatics Campus, Max-Planck-Institute for Informatics, 66123 Saarbrücken, Germany

Christian Willert
Institute of Propulsion Technology, Deutsches Zentrum für Luft- und Raumfahrt (DLR), 51147 Köln, Germany

I. VELOCITY DISTRIBUTION IN THE SYNTHETIC DATASET

Within the synthetic dataset, the wavelengths $\lambda_{xy} = (333.\bar{3} \pm 56)\,\mathrm{pixel}$ and $\lambda_z = (0.\bar{3} \pm 56 \times 10^{-6})\,\mathrm{pixel}$ as well as the velocity amplitude $\hat{A} = (500 \pm 100)\,\mathrm{pixel\,s^{-1}}$ are varied. In this way, rich data is obtained to train and evaluate the employed neural networks. In Fig. S1a, the mean velocity components in x- and y-direction are depicted for all training configurations. While $\bar{U}_y$ contains a bias towards a positive velocity, $\bar{U}_x$ is centered around zero. Within the training data, the mean velocity component $\bar{U}_y$ was varied between $260\,\mathrm{pixel\,s^{-1}}$ and $800\ \mathrm{pixel\,s^{-1}}$. However, within each run, the velocity components vary at a mean standard deviation of $\sigma_{U_x} = \sigma_{U_y} \approx 344\,\mathrm{pixel\,s^{-1}}$ due to the pattern of the applied fluid flow as illustrated in Fig. S1b.

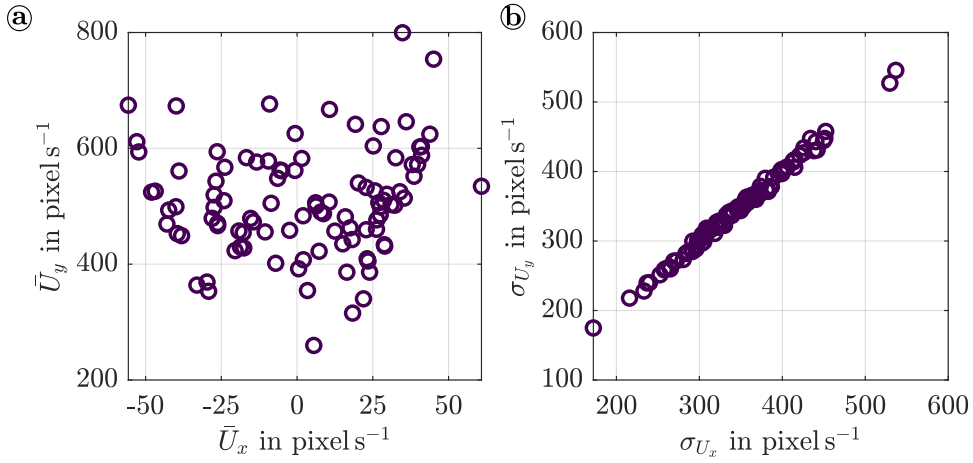

FIG. S1. Mean velocity components $\bar{U}_x$ and $\bar{U}_y$ used in the training split of the synthetic dataset (a). In (b), the corresponding standard deviation of each velocity component is depicted.

II. AI-BASED DETECTION

Different AI-based architectures are used to identify particle images in pseudo-frames or event-tensors within this work: the (i) Recurrent Vision Transformer (RVT [1]), and the (ii) Mixture-of-Experts heat conduction-based detector MvHeatDET [2].

(i) RVT is a recurrent, multi-stage vision transformer backbone designed for low-latency event-based object detection. It processes event data in short temporal windows that are discretized into T and encoded as a dense tensor. Polarity and time bins are flattened into a channel dimension, yielding an input of shape $(2T, H, W)$. Here, W and H denote the width and height of the camera sensor in pixel. Spatial feature extraction combines a convolutional layer that acts as a conditional positional embedding with a two-step attention mechanism that first captures local feature interactions and then mixes these globally using grid attention. Temporal information is separated from spatial feature extraction and aggregated by recurrent units at the end of each stage. RVT uses LSTM cells to carry latent states across successive time steps. The RVT backbone is combined with a YOLOX detection framework, optimizing standard detection losses (classification, regression, and IoU-based terms).

(ii) MvHeatDET is built around a heat-conduction-inspired backbone, designed to efficiently propagate and mix information across spatial locations by operating in the frequency domain. The method first embeds event frames with a stem network. The core blocks implement a mixture-of-experts strategy over alternative frequency transforms (e.g., DFT/IDFT, DCT/IDCT and Haar variants), where a policy network selects an expert branch. Learnable frequency embeddings modulate the diffusion (thermal diffusivity) to adapt the conduction to the content. For detection, MvHeatDET follows a DETR-style paradigm [3] and employs an IoU-based query selection mechanism to obtain higher-quality object queries for the final detection head.

III. KALMAN FILTERING

The Kalman filter works iteratively by updating its state variables with incoming events [4–6]. In the chosen model, the position $(x(t), y(t))$, velocity $(u_x(t), u_y(t))$, and acceleration $(a_x(t), a_y(t))$ are predicted independently in x- and y-direction. The resulting state vector in x-direction reads

$$\boldsymbol{X} = [x\ u_x\ a_x]^{\mathrm{T}}, \tag{S1}$$

with the corresponding covariance matrix $\boldsymbol{P} \in \mathbb{R}^{3\times3}$. However, since only the position is measured, the observation matrix becomes $\boldsymbol{C} = [1\ 0\ 0]$. The underlying model of motion is based on assuming constant acceleration within subsequent data points and is determined by the state transition matrix $\boldsymbol{F}_{\mathrm{i}}$ at time step i according to

$$\boldsymbol{F}_{\mathrm{i}} = \begin{bmatrix} 1 & \delta_{\mathrm{i}} & \frac{1}{2}\delta_{\mathrm{i}}^2 \\ 0 & 1 & \delta_{\mathrm{i}} \\ 0 & 0 & 1 \end{bmatrix}. \tag{S2}$$

Here, the time interval δ_{i} denotes the temporal difference between two successive events or, in the case of image-based or AI-based detection, between two successive detections. Since the events are recorded asynchronously, δ_{i} and $\boldsymbol{F}_{\mathrm{i}}$ are recalculated in each iteration of the Kalman filter. The Kalman filter is initialized by an uncertainty $\boldsymbol{P}_0$ and an initial estimate of the state vector

$$\boldsymbol{X}_0 = [\bar{x}_{\mathrm{e}}\ m_x\ 0]^{\mathrm{T}}, \tag{S3}$$

where $\bar{x}_{\mathrm{e}}$ is the mean value of the measured x-positions of the events and m_x corresponds to a linear estimate of the initial velocity in the first four time steps, if at least four time steps are covered in the respective event cluster. Once initialized, prediction and correction steps are performed iteratively to update the state vector $\boldsymbol{X}$. In the prediction, the state transition matrix $\boldsymbol{F}_{\mathrm{i}}$, state vector $\hat{\boldsymbol{X}}_{\mathrm{i}}$, and covariance $\hat{\boldsymbol{P}}_{\mathrm{i}}$ are predicted in the current time step i according to

$$\hat{\boldsymbol{X}}_{\mathrm{i}} = \boldsymbol{F}_{\mathrm{i}}\boldsymbol{X}_{\mathrm{i-1}}, \tag{S4}$$

$$\hat{\boldsymbol{P}}_{\mathrm{i}} = \boldsymbol{F}_{\mathrm{i}}\boldsymbol{P}_{\mathrm{i-1}}\boldsymbol{F}_{\mathrm{i}}^{\mathrm{T}} + \boldsymbol{Q}. \tag{S5}$$

The process noise covariance matrix $\boldsymbol{Q} \in \mathbb{R}^{3\times3}$ quantifies the uncertainty in the underlying motion model. Hence, $\boldsymbol{Q}$ significantly affects the flexibility of the Kalman filter in response to changes in position or smoothing of measurement noise. The process noise covariance matrix $\boldsymbol{Q}$ is typically initialized with larger values for acceleration noise than for velocity and position noise. However, careful fine-tuning of $\boldsymbol{Q}$ is necessary, especially for particles moving relatively fast. Subsequently, the predicted state vector and covariance are updated in a correction step according to

$$\boldsymbol{X}_{\mathrm{i}} = \hat{\boldsymbol{X}}_{\mathrm{i}} + \boldsymbol{K}_{\mathrm{i}}(\boldsymbol{X}_{\mathrm{e,i}} - \boldsymbol{C}\hat{\boldsymbol{X}}_{\mathrm{i}}), \tag{S6}$$

$$\boldsymbol{P}_{\mathrm{i}} = (\boldsymbol{I} - \boldsymbol{K}_{\mathrm{i}}\boldsymbol{C})\hat{\boldsymbol{P}}_{\mathrm{i}}, \tag{S7}$$

$$\boldsymbol{K}_{\mathrm{i}} = \hat{\boldsymbol{P}}_{\mathrm{i}}\boldsymbol{C}^{\mathrm{T}}(\boldsymbol{C}\hat{\boldsymbol{P}}_{\mathrm{i}}\boldsymbol{C}^{\mathrm{T}} + R)^{-1}. \tag{S8}$$

Here, $\boldsymbol{K}_{\mathrm{i}}$ denotes the Kalman gain, $\boldsymbol{X}_{\mathrm{e,i}} = x\boldsymbol{C}$ the measured position (exemplified in x-direction), and R the measurement noise covariance, which quantifies the uncertainty in the measured values. Hence, the scalar R is chosen to be large if the measurement has a high level of noise and vice versa if the measurement has a low level of noise. In this way, the position and velocity of the particle tracks are determined directly, smoothing out measurement noise and without numerical differentiation. Once the Kalman filter has been applied, subsampling is performed with a configurable number of reference points within a time step Δt.

IV. PROCESSING SPEED

The processing time for detection and tracking is evaluated based on a synthetic dataset having a length of about 1 s and an event rate of 0.65×10^6 events per second. Within the dataset, 273 particles are present simultaneously. However,

if a particle leaves the domain, it is reinitialized at the opposite side. The obtained processing time is listed in Table S1 for different detection and tracking algorithms. For detection based on pseudo-frames (pseudo-frame) and direct processing (DBSCAN, kd-tree, pixelwise extension), a desktop PC equipped with a AMD Ryzen 9 9950X Processor and 128 GB of Random Access Memory (RAM) was used. However, the inference of the neural networks (RVT, MvHeatDET) was carried out on a NVIDIA Quadro RTX 8000 Graphics Processing Unit (GPU) with 48 GB RAM.

Among the direct processing algorithms, kd-tree achieved the fastest processing for detection and matching at 46.2 s, beating the image-based processing by 30.7 s. However, the derived algorithms are not optimized for processing speed. Hence, limiting the number of simultaneous tracks and using a simpler matching scheme to cluster events within a given radius to existing clusters significantly reduces the processing time. In particular, a total processing time of 29.3 s was achieved for detection and tracking when only 100 simultaneous tracks are considered. Furthermore, GPU acceleration, efficient memory allocation and optimized algorithms could further reduce the required processing time. Particle detection based on neural networks required only 28 s when using MvHeatDET. In a similar manner, the required processing time could be reduced by an optimized data loader, hyperparameter tuning and optimized algorithms. In particular, a flatter architecture with less trainable parameters and tailored for particle detection might further reduce processing time while maintaining precision. Hence, a high potential for real-time processing can be indicated. However, as just particle positions are derived by the neural networks, additional processing time required for matching is not included.

After detection, tracking is done in parallel computing using 8 cores. When using image-based or AI-based detection, the amount of data points used for tracking is reduced to the number of time steps. In turn, the required processing time was reduced to 0.5 s when using the Kalman filter. In contrast, the processing time increases significantly when using the raw event clusters found by direct processing, i.e. performing tracking event-by-event. Furthermore, the time required for spline fitting of individual event clusters might increase in some cases due to overfitting, especially when the target RMS is set too low. However, in hybrid-$(\mathrm{X^K}, \mathrm{u^{KS}})$ and hybrid-$(\mathrm{X^{KS}}, \mathrm{u^{KS}})$, the output of the Kalman filter is used as input for the spline fitting, i.e. requiring less data points to be fitted. As a result, the processing time decreased.

TABLE S1. Processing time in seconds needed to process synthetic data with 273 active tracks, 0.65×10^6 events per second and a length of about 1 s using different detection and tracking algorithms.

	detection	Kalman	Spline	hybrid-$(\mathrm{X^K}, \mathrm{u^S})$	hybrid-$(\mathrm{X^K}, \mathrm{u^{KS}})$	hybrid-$(\mathrm{X^{KS}}, \mathrm{u^{KS}})$
pseudo-frame	76.9	5.4	10.0	10.6	11.0	11.3
DBSCAN	151.2	53.9	136.0	156.0	47.5	50.9
kd-tree	46.2	61.9	241.1	296.2	71.1	70.1
pixelwise extension	65.0	44.8	147.7	179.1	49.4	48.9
RVT	8.77	0.5	1.7	1.7	1.6	3.4
MvHeatDET	26.0	1.1	1.6	2.2	2.2	4.7

V. READOUT ERROR

To acquire an experimental dataset with known ground truth, four disks with an number of printed particle images increasing from 41 to 224 were mounted precisely on a rotating chopper wheel. The templates of the four disks (PI1-PI4) are depicted in Fig. S2a-d, respectively. As the distance between the particle images decreases simultaneously, the number of active lines in the pixel array of the event-based camera sensor that need to be read out also increases. Since the events are read out line by line, the hardware requirements for the event-based camera thus increase with the number of particle images.

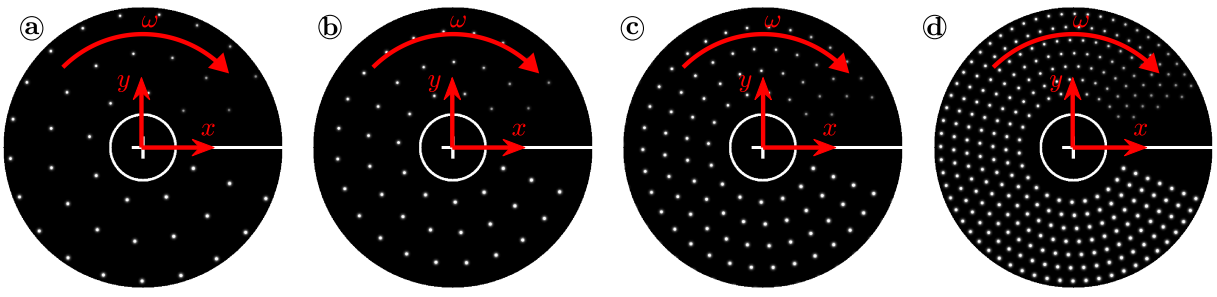

FIG. S2. Templates used to print particle images on disks, which were mounted on a rotating chopper wheel. The amount of printed particle images increases among the four disks (PI1-PI4) shown in (a) to (d), respectively.

A potential readout error initially appears as uniform timestamps along individual lines. To quantify this effect, the average ratio of unique timestamps per line N_{unique} to the total number of events per line N_{total} is depicted against the rotational speed n in Fig. S3a. For evaluation, the accumulation time was set to 200 µs. Independent of the disk used (PI1-PI4), a monotonically decreasing curve is observed, as the event rate, i.e., the number of events to be read per second, increases with increasing rotational speed. However, the event rate also depends on the number of particle images per disk, which leads to lower $N_{\text{unique}}/N_{\text{total}}$ ratios with increasing particle image count. In Fig. S3b, $N_{\text{unique}}/N_{\text{total}}$ is shown as a function of the event rate, where all four curves almost collapse onto each other. However, small differences remain, which are due to the increasing number of lines to be read out as the number of particle images increases. In addition, the $N_{\text{unique}}/N_{\text{total}}$ ratio reaches a plateau for all disks in Fig. S3a, where $N_{\text{unique}}/N_{\text{total}}$ does not decrease further despite increasing rotational speed. However, Fig. S3b shows that the event rate no longer increases significantly in this range. This is caused by a line-by-line irretrievable loss of events, which was observed for event rates greater than $100 \times 10^6\,\text{s}^{-1}$. Only datasets with a ratio $N_{\text{unique}}/N_{\text{total}} > 0.2$ were used for particle tracking and training of neural networks.

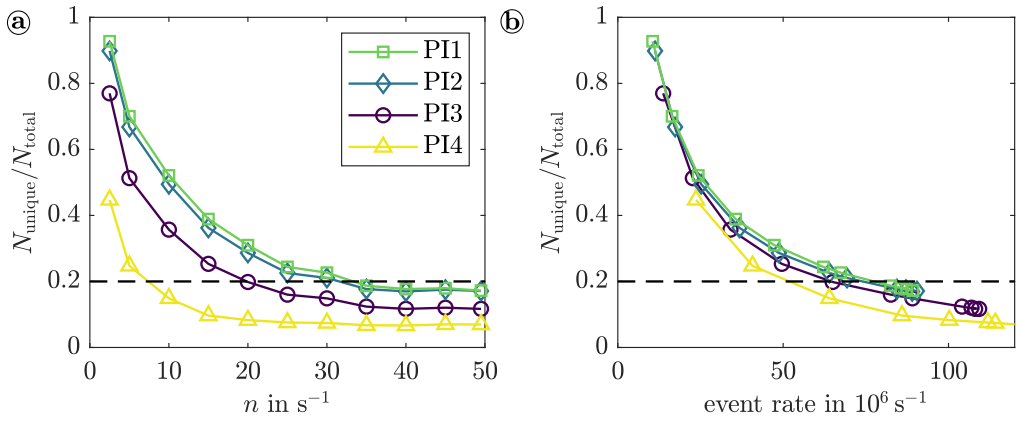

FIG. S3. Average ratio $N_{\text{unique}}/N_{\text{total}}$ as a function of the rotational speed n for the four disks of printed particle images (PI1-PI4) (a). In (b), the same ratio is depicted against the event rate obtained by the event-based camera.

VI. OUTLIER CRITERIA

To reject spurious tracks detected in the event stream recording in the wake of a cylinder, two outlier criteria were applied. (i) The first outlier criterion rejects tracks based on a threshold in the score S, which reflects the quality of the track and is calculated according to

$$S = w_\theta \Delta\tilde{\theta} + w_u \Delta\tilde{u} + w_l \tilde{l}. \tag{S9}$$

Here, $\Delta\tilde{\theta}$ denotes the normalized average angular changes of a track along its path in order to filter strong directional oscillations. Furthermore, $\Delta\tilde{u} = \sigma_u/\bar{\mathbf{u}}$ provides a normalized measure of the velocity fluctuations within a track by dividing the standard deviation σ_u of the velocity by the mean velocity of the track $\bar{\mathbf{u}}$. Finally, $\tilde{l}$ provides a normalized inverted ratio of the path length of the track under consideration relative to the median path length of all tracks. As a result, winding tracks that may jump between different event clusters are rejected. All components of the score were centered around their median value, scaled using the median absolute deviation, and then normalized based on the 95 % quantile. The applicability of the individual criteria depends on the dataset under consideration and is weighted using w_θ, w_u, and w_l.

(ii) In the second outlier criterion, spurious parts of tracks are rejected if they deviate more than defined thresholds from tracks in the immediate vicinity in terms of velocity components (u_x, u_y) or direction (θ). In particular, the median and standard deviation of the velocity components and direction of surrounding tracks are considered, Gaussian weighted with the distance to the track under consideration. As an example for the velocity component u_x, this results in

$$\frac{u_x - \text{med}(u_x)}{\sigma_{u_x}} < \text{threshold}. \tag{S10}$$

In this way, tracks that do not fit into the collective particle motion are discarded.

[1] M. Gehrig and D. Scaramuzza, Recurrent vision transformers for object detection with event cameras, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2023).
[2] X. Wang, Y. Jin, W. Wu, W. Zhang, L. Zhu, B. Jiang, and Y. Tian, Object Detection using Event Camera: A MoE Heat Conduction based Detector and A New Benchmark Dataset, arXiv , 2412.06647 (2024).
[3] Y. Zhao, W. Lv, S. Xu, J. Wei, G. Wang, Q. Dang, Y. Liu, and J. Chen, Detrs beat yolos on real-time object detection, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2024) pp. 16965–16974.
[4] R. E. Kalman, A new approach to linear filtering and prediction problems, Journal of Basic Engineering **82**, 35 (1960).
[5] S. J. Julier and J. K. Uhlmann, A new extension of the kalman filter to nonlinear systems, Signal Processing, Sensor Fusion, and Target Recognition VI **3068**, 182 (1997).
[6] O. AlSattam, M. Mongin, M. Grose, S. Gunasekaran, and K. Hirakawa, KF-PEV: a causal Kalman filter-based particle event velocimetry, Experiments in Fluids **65**, 141 (2024).