CAAI Transactions on Intelligence Technology

*Research Article*

# Object tracking on event cameras with offline–online learning

ISSN 2468-2322
Received on 29th December 2019
Revised on 6th April 2020
Accepted on 29th April 2020
doi: 10.1049/trit.2019.0107
www.ietdl.org

*Rui Jiang*[1] ✉, *Xiaozheng Mou*[1], *Shunshun Shi*[1], *Yueyin Zhou*[1], *Qinyi Wang*[1], *Meng Dong*[1], *Shoushun Chen*[1,2]

[1]*CelePixel Technology Co. Ltd, 71 Nanyang Drive 638075, Singapore*
[2]*School of Electrical and Electronic Engineering, Nanyang Technological University, 639798, Singapore*
✉ *E-mail: rui.jiang@celepixel.com*

**Abstract:** Compared with conventional image sensors, event cameras have been attracting attention thanks to their potential in environments under fast motion and high dynamic range (HDR). To tackle the lost-track issue due to fast illumination changes under HDR scene such as tunnels, an object tracking framework has been presented based on event count images from an event camera. The framework contains an offline-trained detector and an online-trained tracker which complement each other: The detector benefits from pre-labelled data during training, but may have false or missing detections; the tracker provides persistent results for each initialised object but may suffer from drifting issues or even failures. Besides, process and measurement equations have been modelled, and a Kalman fusion scheme has been proposed to incorporate measurements from the detector and the tracker. Self-initialisation and track maintenance in the fusion scheme ensure autonomous real-time tracking without user intervene. With self-collected event data in urban driving scenarios, experiments have been conducted to show the performance of the proposed framework and the fusion scheme.

## 1 Introduction

Multiple object tracking has become an essential module in human–computer interaction [1], automated surveillance [2], and vehicle navigation systems [3, 4]. With the rapid development of sensor technology, event cameras (or dynamic vision sensors, DVSs) have come into the market thanks to their superior performance in fast-moving scenarios and high dynamic scenes. Although many research efforts have been made, the algorithms designed for event cameras are still far from mature. In particular, event cameras generate significantly different data compared to conventional image sensors (see Section 3.1 for details). There exist two technical routes for event data processing and interpretation. The first tries to establish an event-based processing framework, while the second makes use of the off-the-shelf image processing approaches to accelerate the development. The first route may have a better chance to unleash the full potential of the sensor, but most existing methods belong to the second. Although data from event cameras can be approximately represented in frames, it is still not clear that if these data can be efficiently and effectively utilised by traditional and modern machine vision algorithms, such as feature extraction and convolutional neural networks (CNNs).

This work studies the visual tracking problem based on event cameras, Fig. 1 demonstrates the diagram of the proposed framework, which contains two stages. At stage one, offline training has been done based on YOLO detector [5] before tracking, such that object detection results with objects' types could be obtained. At stage two, online training and tracking are in a simultaneous process. The real-time generated event count images are fed into the offline-trained detector and the correlation filter (CF). After data association, those assignments are sent to a Kalman ensemble as measurements. Meanwhile, the CF also outputs its tracked results to the Kalman ensemble as supplementary measurements. Reinitialisation is triggered once inconsistencies are found between outputs from the ensemble and the CF, such that the ensemble is adaptive to various conditions with changes of appearance and scale. The main contributions are listed as follows:

- An object tracking framework has been presented for event cameras, aiming at real-time applications for ground vehicles. The tracking framework contains a YOLO-based [5] offline-trained deep detector and a CF-based [6] online-trained tracker, which complements each other.
- A fusion scheme, which incorporates the results from the detector and the tracker, has been proposed based on the Kalman filter. With automated initialisation and track maintenance strategies in the fusion scheme, real-time intervene-free operations could be attained.
- Features in event count images have been demonstrated well-learned by deep networks and feature extractors. Robustness and accuracy of the tracker have been presented using data in actual driving scenarios.

The paper is organised as follows: Section 2 presents related work in event-based object tracking. Section 3 introduces the event camera and its unique data format, and the essential modules in an online tracker. Then, the proposed tracking framework is elaborated in Section 4, followed by evaluation results in Section 5. Lastly, Section 6 concludes the paper.

## 2 Related work

Object tracking has been extensively studied in the past decades. In [7], different implementations for the five building blocks in a tracking framework, namely the motion model, the feature extractor, the observation model, the model updater, and the ensemble post-processor have been evaluated to help design a tracker. Authors in [8] review multiple object tracking approaches thoroughly. A multiple object tracker (MOT) is not a simple combination of single object trackers. In practice, manual initialisation is often infeasible; thus an initialisation scheme needs

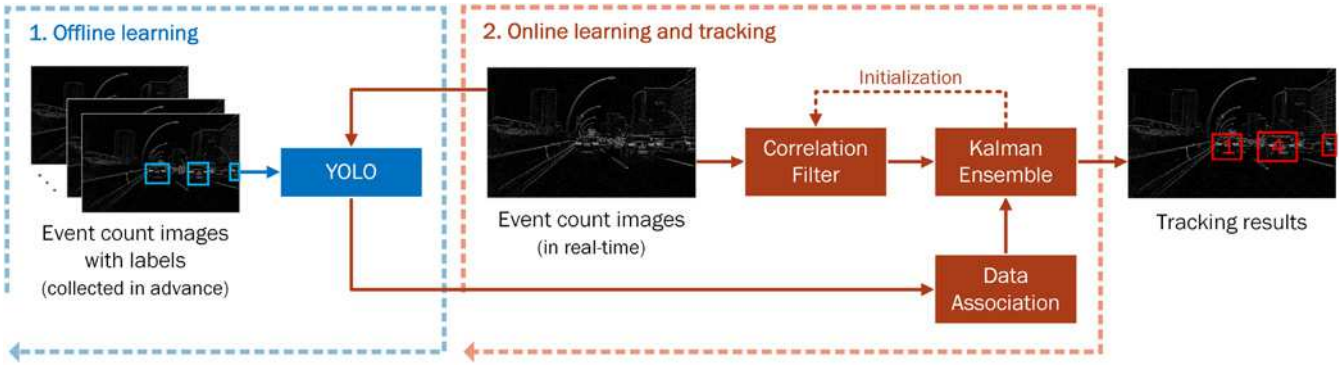


**Fig. 1** *Proposed object tracking framework. The two stages are denoted as blue and red dotted lines with arrows, respectively. The first stage needs to be completed off-line before tracking such that a YOLO detector is trained; In the second stage, the CF, which tracks single object, is initialised/reinitialised using results from the YOLO detector. The Kalman filter fuses measurements from both the detector and the tracker to achieve multi-object tracking*

to be proposed for MOTs. Moreover, a maintenance mechanism is essential in MOTs to create new tracks, delete obsolete tracks and assign consistent labels, since the number of objects may vary at any time. Besides, data association is necessary for assigning detections to trackers such that the ensemble post-processor can update objects' positions according to latest observations.

Recently, the concept of 'tracking by detection' has become increasingly popular owing to its discriminative nature compared to conventional generative model-based trackers [9–11]. Many event-based detectors have been proposed: As an early attempt, a hierarchical spiking model for object recognition has been introduced in [12], where asynchronous data are fed into the spiking neural networks that take advantage of accurate timing from event cameras. An end-to-end object detector has been proposed for DVS in [13], where the feature extraction network is designed by considering unique data structure from DVS; moreover, the adaptive temporal pooling is applied to balance triggered events between rapid and slow motions. A comparison of object detection performance between traditional image frames and event frames has been presented in [14], which shows the advantages of event-based approaches in fast and low-light conditions. Authors in [15] propose two neural network architectures for object detection based on the YOLO [5] detector.

As for event-based object tracking, an event camera has been used to assist MOT in [16], demonstrating superior performance for fast-moving objects. In [17], an expectation maximisation-based probabilistic multi-hypothesis tracker is presented for MOT with false alarm observations. In [18], a moving object detector has been proposed for event cameras. After motion correction, those areas with different event timestamps are deemed as moving objects. Then the detected objects are tracked using a Kalman filter. The similar tracking scheme is also implemented in [19]. An event-driven stereo vision system has been developed in [20], where cluster tracking and three-dimensional (3D) reconstruction are complementarily integrated such that ambiguity could be solved when occlusion occurred. By simple morphological operations, the moving objects are extracted and tracked using a Kalman filter with the constant acceleration model. Authors in [21] propose an online tracker-detector integration framework, where the local tracker is based on a support vector machine classifier, and the user-initialised global detector is responsible for failure recovery.

We have listed some state-of-the-art methods in Table 1 for comparison. Compared with [21], this work considers multiple objects in the tracking framework. Compared with [18, 19], all three papers use the Kalman filter as tracking framework, but this work focuses on improving label continuity and tracking stability. The offline-trained detector not only brings a priori information to enable self-initialisation but also improves final performance by providing drift-free, independent results for each frame. The online-trained tracker provides persistent results once initialised, and is free from drifting issues with the help of self-reinitialisation.

**Table 1** Object tracking approaches using event cameras

| | No. of objects | Configuration | Tracking scheme |
|---|---|---|---|
| [18] | multiple | monocular | probabilistic |
| [20] | multiple | stereo | event matching-based |
| [21] | single | monocular | optimisation-based |
| [22] | single | monocular | probabilistic |
| [19] | multiple | monocular | probabilistic |
| this work | multiple | monocular | probabilistic |

## 3 Preliminaries

In this preliminary section, we first introduce event cameras and their data formats for those readers who are not familiar with event cameras. Then we present how to represent the appearance of a region of interest to generate 'features', and how to utilise the features in single object tracking, and finally, how to generalise to MOT problems from a single object tracker.

### 3.1 Event cameras and event frames

Event cameras detect intensity changes at each pixel independently. When the change is larger than a user-defined threshold, an 'event' $\langle (x, y), I, t\rangle$ would be triggered, where $x$, $y$ denote pixel coordinates, $I$ is the intensity, and $t$ represents the timestamp [The explicit data format varies based on the sensor model. Events as $\langle (x, y), p, t\rangle$ where $p$ is the polarity are common in the literature [23].]. From these events, diverse processing techniques can be used to generate event-based data for easier implementation. By aggregating events in an interval as a tuple set $\mathcal{S} = \{\langle (x, y), I, t\rangle\}$, different types of event frames can be obtained. We write the fixed time interval for event frames as $\Delta T$; $t_1$ denotes the timestamp of the first event in each interval; $I_k(x, y)$ represents the intensity at pixel $(x, y)$ for the $k$ th frame. An event binary image is

$$I_k(x, y) = \begin{cases} 1, & \text{if } (x, y) \in \mathcal{S} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

An event intensity image, which utilises the intensity information, can be generated as

$$I_k(x, y) = \begin{cases} I, & \text{if } (x, y) \in \mathcal{S} \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

An event count image that accumulates events as intensity for each pixel is

$$I_k(x, y) = \sum_{(x,y)\in\mathcal{S}} 1 \tag{3}$$


*CAAI Trans. Intell. Technol.*, 2020, Vol. 5, Iss. 3, pp. 165–171

An 8-bit event full frame picture, which reflects events' sequence, is obtained from

$$I_k(x,y)=\begin{cases}\left\lfloor\dfrac{t-[t_1+(k-1)\Delta T]}{\Delta T}\times 255\right\rfloor, & \text{if } (x,y)\in\mathcal{S}\\ 0, & \text{otherwise}\end{cases} \quad (4)$$

where $\lfloor\cdot\rfloor$ denotes the floor function. In this work, we use event count images generated from (3) for both detection and tracking.

### 3.2 Essentials of the online tracker

As this paper is not a comprehensive review, only those building blocks that have been applied to our framework are explained here for readers' convenience.

*3.2.1 Appearance representation:* A natural idea to represent the region of interest on an image is to simply use the intensity values inside the region and write them as the vector form. To include more information, intensities in colour channels instead of grayscale have been used. However, those raw intensities are sometimes too specific to describe the substantial features of the interested region. The histogram of oriented gradients (HOGs) [24, 25] has been demonstrated as a useful feature in object detection. In addition to hand-crafted features, it is noted that learned features [26–28] using data-driven techniques sometimes perform better. YOLO is one of the examples that shows excellent object detection results using deep convolutional features. In the proposed framework, deep features have been used in YOLO, while either intensity features or HOG could be applied to the CF.

*3.2.2 Correlation filter:* The filter-based object trackers model the problem as regression. Given a set of features $\boldsymbol{y}_i$ with target points as training samples and their corresponding user-defined responses $g_i$, we aim to train a filter $f(\boldsymbol{y}_i)$ which gives the maximum correlation output on these points. Suppose the regression model is linear $f(\boldsymbol{y}_i)=\boldsymbol{u}^{\top}\boldsymbol{y}_i$, the problem becomes to find the model parameter $\boldsymbol{u}$ that leads to minimum squared regression error:

$$\min_{\boldsymbol{u}}\sum_i\left(f(\boldsymbol{y}_i)-g_i\right)^2+\lambda\|\boldsymbol{u}\|^2 \quad (5)$$

where $\lambda>0$ is the penalty term parameter. Correlation is computed in the Fourier domain so that fast online training could be achieved because correlation operation becomes element-wise multiplication in the Fourier domain. The analytical solution of (5) in the Fourier domain has been derived as [6, 29]

$$\boldsymbol{u}=\left(\boldsymbol{X}^{\mathrm{H}}\boldsymbol{X}+\lambda\boldsymbol{I}\right)^{-1}\boldsymbol{X}^{\mathrm{H}}\boldsymbol{g} \quad (6)$$

where $\boldsymbol{X}$ is the data matrix that contains one sample $x_i$ for each row; $(\cdot)^{\mathrm{H}}=(\cdot)^{*\mathrm{T}}$ denotes the Hermitian transpose, and $(\cdot)^*$ is the complex-conjugate notation; $\boldsymbol{I}$ is the identity matrix, and $\boldsymbol{g}$ contains regression targets $g_i$.

Authors in [6] have noted that the regression problem could be further accelerated when the training data is composed of cyclic shifts. Moreover, Kernelised CFs (KCFs) have been derived to tackle kernel ridge regression problems. The proposed approach is generalised to multi-channel images, such that more features including HOG could be applied in appearance representation. Considering both speed and accuracy, the KCF is used for online learning in this work.

*3.2.3 Data association:* One additional concern for MOT compared to single object tracking is the data association, which assigns detected objects to existing tracks. The problem is usually formulated as the minimisation of total costs, which, could be solved by the Hungarian algorithm [30, 31]. Particularly, suppose we have $n$ detections pending assignment to $m$ tracks, and a cost matrix $\boldsymbol{A}_{\mathrm{cost}}\in\mathbb{R}^{m\times n}$ representing the cost of assigning the $j$th detection to the $i$th track. The algorithm outputs pairs of tracks and corresponding detections as the first and second columns in matrix $\boldsymbol{A}_{\mathrm{assgnmt}}\in\mathbb{R}^{l\times 2}$ respectively, where $l$ denotes the number of successfully paired assignments.

## 4 Offline–online multiple object tracking

### 4.1 Offline learning

The offline learning follows the YOLOv3 framework [32] where the ground truth is manually labelled with bounding boxes on the objects in event count images. Although YOLO can detect multiple classes, only cars are considered in off-line training in this work due to a lack of manually-labelled data. The features are then extracted by undergoing the CNN of Darknet-53. Finally, the model is learned or fine-tuned by minimising the loss between the ground truth and the regression results.

### 4.2 Online learning and tracking

The online learning and tracking procedure, as the core contribution of this work, is summarised in Algorithm 1. The proposed tracking framework maintains a list of **`tracks`** that keeps necessary properties (such as Kalman filter estimates, type, CF status and age variables) of all track candidates. A priori estimate and error covariance are computed from the function **`predictKalman`**, and **`assignHungarian`** tries to find the best match between tracks and **`detections`**. Then, estimated bounding boxes on the current image $I_k$ are used as positive samples for online learning in **`correlationFilter`**. Correction is done in **`correctKalman`** to incorporate online CF and offline detector together. The function **`maintainTracks`** removes obsolete tracks, creates new tracks and updates their properties according to assignment results. The conditions of track removal is detailed in the Section 4.2.1.

*4.2.1 System modelling in Kalman ensemble:* As the Kalman filter has been widely used in object tracking [33, 34], the recursive equations are omitted for simplicity. In this work, notation $\hat{(\cdot)}$ denotes the estimated value; at discrete time $k$, subscript $(\cdot)_{k|k-1}$ and $(\cdot)_{k|k}$ represent the predicted (a priori) estimate and the updated (a posteriori) estimate, respectively.

We consider the following linear process model between $k$ and $k+1$:

$$\boldsymbol{x}_{k+1}=\boldsymbol{F}_k\boldsymbol{x}_k+\boldsymbol{w}_k=\boldsymbol{F}_k\boldsymbol{x}_k+\boldsymbol{G}_k\breve{\boldsymbol{w}} \quad (7)$$

where the state vector $\boldsymbol{x}=[x,y,v_x,v_y,w,v_w,s]^{\top}\in\mathbb{R}^7$ contains properties of the bounding box being tracked, which includes central coordinates, velocities, the width, the change rate of width, and the ratio of width to height. We assume that the process modelling error is caused by random impact in 2D image plane, thus $\breve{\boldsymbol{w}}=[a_x,a_y,a_w,a_s]^{\top}\in\mathbb{R}^4$ denotes the zero-mean independent Gaussian noise with the diagonal covariance matrix $\breve{\boldsymbol{Q}}\in\mathbb{R}^{4\times 4}$. Then we write

$$\boldsymbol{F}_k=\begin{bmatrix}1&0&\Delta t_k&0&0&0&0\\0&1&0&\Delta t_k&0&0&0\\0&0&1&0&0&0&0\\0&0&0&1&0&0&0\\0&0&0&0&1&\Delta t_k&0\\0&0&0&0&0&1&0\\0&0&0&0&0&0&1\end{bmatrix}$$

$$\boldsymbol{G}_k=\begin{bmatrix}\frac{1}{2}\Delta t_k^2&0&0&0\\0&\frac{1}{2}\Delta t_k^2&0&0\\\Delta t_k&0&0&0\\0&\Delta t_k&0&0\\0&0&\frac{1}{2}\Delta t_k^2&0\\0&0&\Delta t_k&0\\0&0&0&\frac{1}{2}\Delta t_k^2\end{bmatrix}$$

where $\Delta t_k$ is the time interval between states at $k$ and $k+1$. At time $k$, by denoting the process covariance matrix in standard form as $\boldsymbol{Q}_k$, we have

$$\begin{aligned}\boldsymbol{Q}_k &= \mathrm{Cov}[\boldsymbol{w}_k] = \mathrm{Cov}[\boldsymbol{G}_k\breve{\boldsymbol{w}}] = \mathrm{E}[(\boldsymbol{G}_k\breve{\boldsymbol{w}})(\boldsymbol{G}_k\breve{\boldsymbol{w}})^\top] \\ &= \boldsymbol{G}_k\mathrm{E}[\breve{\boldsymbol{w}}\breve{\boldsymbol{w}}^\top]\boldsymbol{G}_k^\top = \boldsymbol{G}_k\mathrm{Cov}[\breve{\boldsymbol{w}}]\boldsymbol{G}_k^\top \\ &= \boldsymbol{G}_k\breve{\boldsymbol{Q}}\boldsymbol{G}_k^\top\end{aligned} \tag{8}$$

where $\mathrm{Cov}[\cdot]$ and $\mathrm{E}[\cdot]$ denote the covariance matrix and the expectation of a random vector.

The measurement model at time $k$ is written as

$$\boldsymbol{z}_k = \boldsymbol{h}_k(\boldsymbol{x}_k) + \boldsymbol{v}_k \tag{9}$$

By omitting the time index, we have

$$\boldsymbol{h}(\boldsymbol{x}) = \begin{bmatrix} x & y & w & \frac{w}{s} & x & y \end{bmatrix}^\top \tag{10}$$

The measurement vector $\boldsymbol{z} = [\boldsymbol{z}_\mathrm{d}^\top, \boldsymbol{z}_\mathrm{c}^\top]^\top$ contains two parts, where $\boldsymbol{z}_\mathrm{d}$ and $\boldsymbol{z}_\mathrm{c}$ are from the offline-trained detector and the online-trained CF, respectively. Specifically, $\boldsymbol{z}_\mathrm{d} = [x_\mathrm{d}, y_\mathrm{d}, w_\mathrm{d}, h_\mathrm{d}]^\top$ and $\boldsymbol{z}_\mathrm{c} = [x_\mathrm{c}, y_\mathrm{c}]^\top$. Note that we have not considered width and height from the CF, since no scale change is allowed in CF for the sake of real-time performance. After linearisation, the Jacobian $\boldsymbol{H}_k = (\partial\boldsymbol{h}/\partial\boldsymbol{x})|_{\hat{\boldsymbol{x}}_{k|k-1}}$ is derived as

$$\boldsymbol{H}_k = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{1}{\hat{s}_{k|k-1}} & 0 & -\frac{\hat{w}_{k|k-1}}{\hat{s}^2_{k|k-1}} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

and $\boldsymbol{v}_k$ denotes the measurement noise that is assumed to be zero-mean Gaussian with covariance matrix $\boldsymbol{R}_k$.

It is noted that $\boldsymbol{z}_\mathrm{d}$ is not always available as (i) the detector may fail to recognise the object in each frame; or (ii) the data association may not assign the detected bounding boxes to their corresponding tracks correctly. In this case, the ensemble still works in a short time by correcting objects' position using $\boldsymbol{z}_\mathrm{c}$, which is persistent once initialised. If neither $\boldsymbol{z}_\mathrm{d}$ nor $\boldsymbol{z}_\mathrm{c}$ is available, the ensemble predicts objects' position for a certain iterations $\tau$, then removes those tracks if measurements cannot be recovered.

*4.2.2 Cost function in data association:* We use the assignment algorithm described in Section 3.2.3. The values in $\boldsymbol{A}_\mathrm{cost}$ play a central role in algorithm performance. In this work, we modified the cost function [35] at the time $k$ such that both position and type from the detector are considered in assignment:

$$\begin{aligned}A_\mathrm{cost}(i,j) = (\boldsymbol{z}_\mathrm{d} - \boldsymbol{H}_\mathrm{d}\hat{\boldsymbol{x}}_{k|k-1})^\top\Sigma_\mathrm{d}^{-1}(\boldsymbol{z}_\mathrm{d} - \boldsymbol{H}_\mathrm{d}\hat{\boldsymbol{x}}_{k|k-1}) \\ + \ln|\Sigma_\mathrm{d}| + \eta_\mathrm{type}\psi_k(i,j)\end{aligned} \tag{11}$$

where $\boldsymbol{H}_\mathrm{d} \in \mathbb{R}^{4\times 7}$ contains the first four rows of $\boldsymbol{H}_k$; $A_\mathrm{cost}(i,j)$ denotes the element at row $i$, column $j$ in matrix $\boldsymbol{A}_\mathrm{cost}$; $\Sigma_\mathrm{d} = \boldsymbol{H}_\mathrm{d}\boldsymbol{P}_{k|k-1}\boldsymbol{H}_\mathrm{d}^\top$ could be computed each time after Kalman prediction; $|\cdot|$ is the matrix determinant notation; parameter $\eta_\mathrm{type}$ is the weight of type detection error and

$$\psi_k(i,j) = \begin{cases} 0, & \text{if the track} - \text{detection pair } (i,j) \\ & \text{shares the same type at time } k \\ 1, & \text{otherwise} \end{cases}$$

1: $k = 1$
2: tracks = initializeTracks()
3: $\{\hat{\mathbf{x}}_{0|0}, \mathbf{P}_{0|0}\}$ = initializeKalman()
4: **while** true **do**
5: $\quad\{\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}\} = \mathrm{predictKalman}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{P}_{k-1|k-1})$
6: $\quad\mathbf{z}_\mathrm{d} = \mathrm{assignHungarian}(\mathrm{detections}, \mathrm{tracks})$
7: $\quad\mathbf{z}_\mathrm{c} = \mathrm{correlationFilter}(\mathrm{tracks}, I_k)$
8: $\quad\{\hat{\mathbf{x}}_{k|k}, \mathbf{P}_{k|k}\} = \mathrm{correctKalman}(\hat{\mathbf{x}}_{k|k-1}, \mathbf{P}_{k|k-1}, \mathbf{z}_k)$
9: $\quad$tracks = maintainTracks(tracks)
10: $\quad k$++
11: **end while**

**Fig. 2** *Algorithm 1 Online learning and tracking on event frames*

The discrete time index $k$ is omitted in $A_\mathrm{cost}(i,j)$, $\boldsymbol{z}_\mathrm{d}$, $\boldsymbol{H}_\mathrm{d}$ and $\Sigma_\mathrm{d}$ for simplicity. See Algorithm 1 (Fig. 2).

*4.2.3 Initialisation and reinitialisation of CF:* Since the proposed tracking framework aims to work continuously without users' intervention, it is necessary to design an initialisation scheme for the CF to deal with new objects. As the CF does not consider scale variation, reinitialisation is required to update objects' bounding boxes so that the regions out of the objects could be excluded in the CF. Moreover, due to the change of object motion, appearance and illumination conditions, results from the CF may drift unboundedly. The situation also needs to be saved by a reinitialisation scheme. In this work, a Boolean flag **initialisedCF** with the default status **false** would be assigned for each track. At time $k$, the initialisation would be done to set **initialisedCF = true** if *all* following conditions are satisfied:

- there is a reliable track from the Kalman ensemble; *and*
- the track is with the status **initialisedCF = false**; *and*
- the measurement $\boldsymbol{z}_\mathrm{d}$ is available.

Each initialised track would be constantly checked at each iteration. Reinitialisation of CF would be triggered to reset **initialisedCF = false**, when *one of* the following conditions is satisfied:

- the difference of central coordinates between the CF and Kalman ensemble exceeds the user-defined threshold; *or*
- the difference of bounding boxes' size between the CF and Kalman ensemble exceeds the user-defined threshold; *or*
- the bounding box from the CF is too close to the borders of image.

The first and second conditions deal with drifts and scale variations, respectively, while the third one is based on the fact that CF performs worse at image edges due to incomplete positive and negative learning samples.

## 5 Evaluation in driving scenarios

We have tested the approach on self-collected event data, a long sequence of 1663 event count images in urban areas, from CelePixel CeleX5 Sensor. The data show common behaviors of vehicles such as lane change and overtaking. Occlusion and scale variation are frequent due to the heavy traffic. Some typical scenarios in the dataset are shown in Fig. 3. The performance metrics used in this work include the multiple object tracking accuracy (MOTA) [36, 37] and the maximum label number (MLN). MOTA is defined as

$$\mathrm{MOTA} = 1 - \frac{\sum_k (FN_k + FP_k + \Phi_k)}{\sum_k GT_k} \tag{12}$$

where $FN_k$, $FP_k$, $\Phi_k$, and $GT_k$ represent the number of missed detections (or false negatives), of false positives, of mismatched

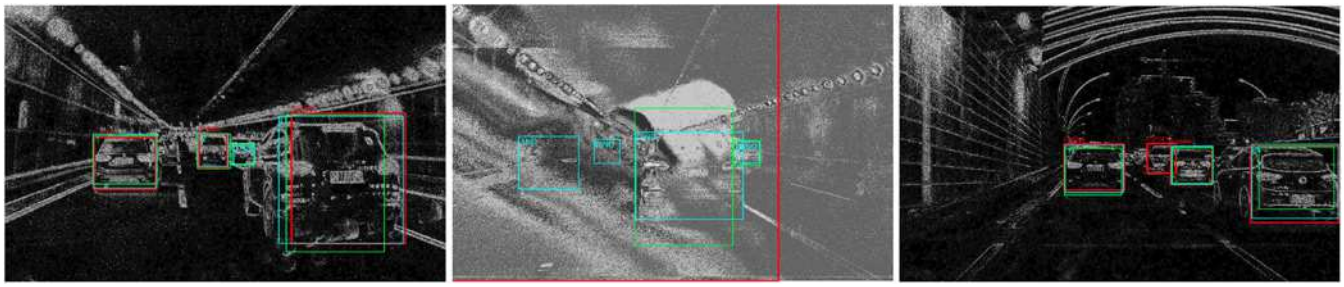

**Fig. 3** *Typical scenarios in the testing sequence. Left: lane change in a tunnel; Middle: a blurred event count image with no available detection; Right: Exiting the tunnel (high dynamic range scene). The red, green and cyan bounding boxes denote results from the detector, the CF, and the fusion scheme, respectively. Red numbers are detector confidence. Cyan numbers are object identification indices. 'ND' means no matched detection for current tracking result*

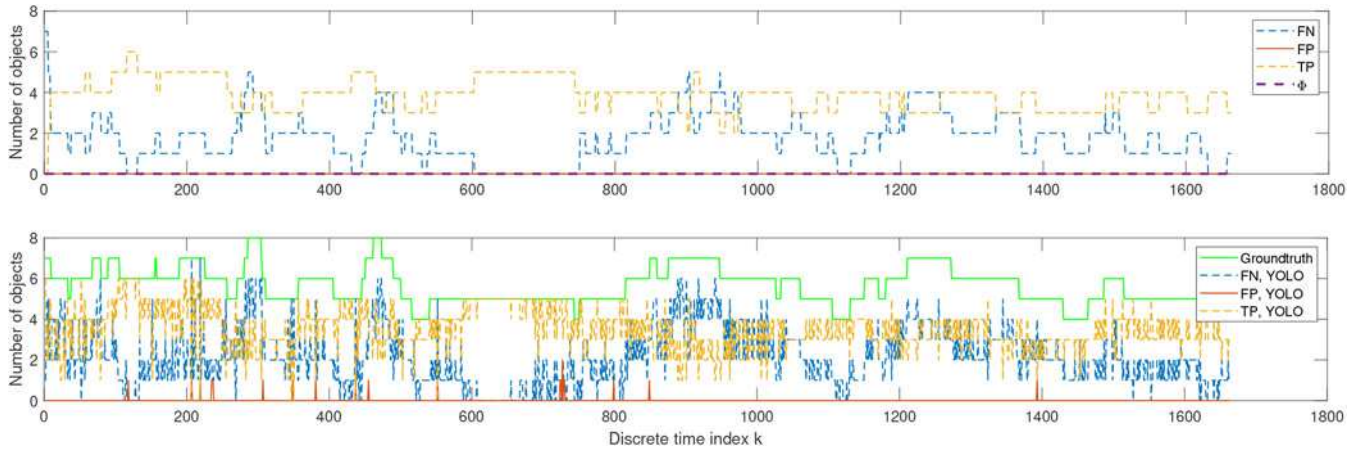


**Fig. 4** *Metrics used for MOTA calculation. The upper and lower figures show numbers of objects that are tracked in the proposed framework and the YOLO detector, respectively. 'FN': 'FP': 'TP', and ' Φ ' denote False Negative, False Positive, True Positive, and fragmentation*

objects (or fragmentations), and of groundtruth objects at time $k$, respectively. MLN is defined as the MLN from the tracker. It reflects the unnecessary initialisation of new tracks: the larger MLN is, the weaker label continuity is.

Although quantitative metrics related to bounding boxes tracking accuracy (e.g. multiple object tracking precision) are not available as the proposed approach focuses on providing smooth and continuous tracking results, the improved performance can still be demonstrated by qualitative results. It is noted that the position error of the proposed tracker is related to the performance of the YOLO detector and the CF since both are providing position measurements, while the errors on bounding box width and height are influenced by YOLO detector solely. Moreover, it is essential to tune Kalman parameters in practice, as the output confidence level in the deep learning-based model is not in the sense of probability but from the proximity to cost function. The full evaluation video has been made available online at https://youtu.be/wUAj9dMeDNI.

### 5.1 General results

We firstly run the YOLO detector alone as a baseline, which shows MOTA $= 0.62$ without considering $\Phi_k$ as there is no numbered label for YOLO detection results. With the proposed approach, it is observed that MOTA has been raised to 0.69 when using the linear kernel, HOG feature and track removal threshold $\tau = 10$. The metrics used for MOTA calculation could be found in Fig. 4. In particular, by implementing the proposed framework, the numbers of *FN* and *TP* become much smoother, as shown in Fig. 4 especially from $k = 600$ to 750. Moreover, there is no *FP* for the proposed approach after removing unreliable tracks, at the cost of a slower response to new objects. In the testing sequence, no sudden change of identification indices for the same object has been found; thus, the fragmentation $\Phi_k = 0$.

Fig. 5 shows the relation between MLN and the user-defined parameter $\tau$. It is obvious that MLN becomes smaller with a larger $\tau$, because a larger $\tau$ retains the track longer when the detection result is not available. However, a larger $\tau$ would also lead to higher drifting errors in case of no correction from the detector. In this work, by incorporating CF results, the tracking performance could be improved by setting a larger $\tau$.

Without the online-trained tracker, the whole tracking framework still works by considering the output $z_d$ as the sole measurement vector. We have compared the performance between two configurations: (i) the partial configuration without CF; and

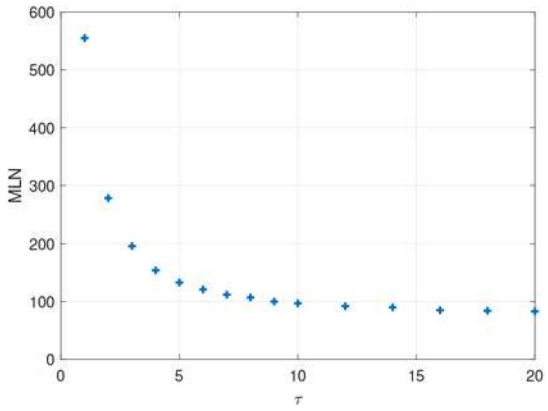


**Fig. 5** *Relation between MLN and parameter τ, which determines in how long an invisible track would be removed if not detected consecutively*

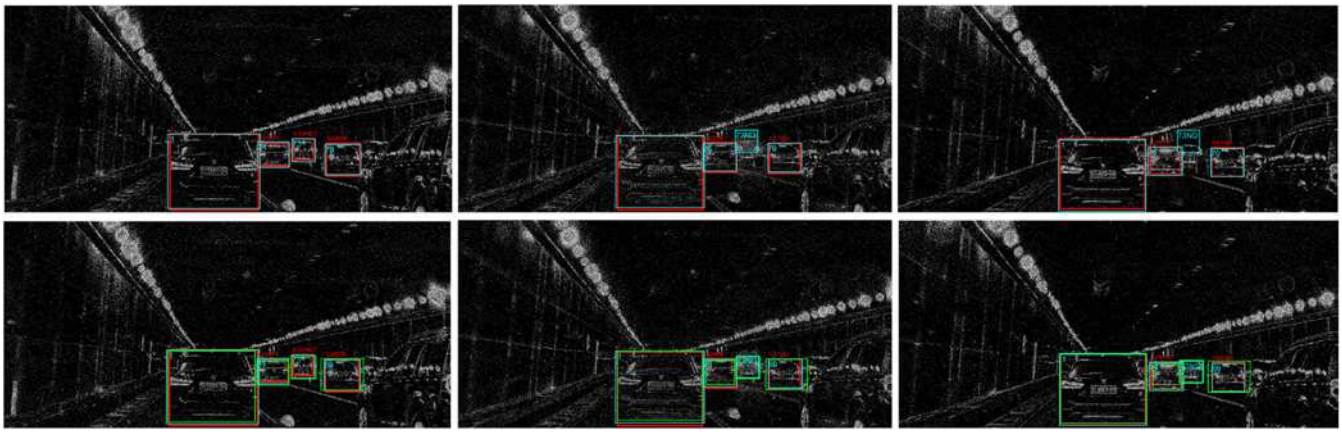

**Fig. 6** *Tracking performance comparison between full configuration (lower row) and partial configuration (upper row) for 3 frames. Note the object 73 loses detection from the second frame. The tracker deviates from the true position in partial configuration due to inaccurate motion model. The tracker in full configuration is well-performed thanks to the online tracker*

**Table 2** Real-time performance of the proposed tracking framework

| Partial configuration | Full configuration | | |
|---|---|---|---|
| | Feature | Intensity | HOG |
| 459 FPS | Linear kernal | 55.7 FPS | 50.4 FPS |
| | Gaussian kernal | 33.8 FPS | 34.6 FPS |

(ii) the full configuration with offline-online learning. It is noted that the partial configuration only contains single information source on object position. If an object is not detected, its position will be predicted by constant velocity motion model. In the circumstance of false negative detections, the full configuration is more accurate and robust, as demonstrated in Fig. 6. The real-time performance between two configurations are shown in Table 2.

The time consumption covers core operations in the framework such as CF, Kalman filtering, and data association, and does not include YOLO detection as it is executed outside the loop. The FPS is computed on a mobile workstation running MATLAB R2018b, with an Intel Core i5-8250U CPU and 8 GB memory. HOG orientation number and cell size are set to 6 and 4, respectively. Although KCF could be used to improve the performance by bringing features into high-dimensional spaces [6], it lowers computational efficiency. It is noted that the performance gap between different kernels is small when a powerful feature is selected. These results are similar to [7].

### 5.2 Influence of Kalman parameters

It is well known that parameters in Kalman ensemble, namely noise covariance matrices, affect final results. In all experiments, we set

$$\check{\boldsymbol{Q}} = \text{diag}(500, 1000, 20, 0.05) \tag{13}$$

$$\boldsymbol{R} = \text{diag}(5, 5, 20, 20, 10, 10) \tag{14}$$

by experience and parameter tuning. The difference in $x$ and $y$ components in $\check{\boldsymbol{Q}}$ is to deal with the uneven pavement or humps, where objects may jump in $y$ direction. In practice, we prefer to trust measurements from the offline-trained detector more, if the measurement is available, as the precision of CF bounding boxes' positions are less accurate since the image quality has been sacrificed for efficiency.

## 6 Conclusion

This paper presents a real-time tracking framework based on event count images from event cameras. Compared to conventional tracking-by-detection, an offline-online training scheme has been presented to improve the tracking robustness and accuracy. Deep features, HOG or raw intensity features have been demonstrated effective in the detector and the tracker. A Kalman ensemble, together with the proposed self-initialisation scheme, has been designed to incorporate measurements from the detector and the tracker such that the tracking consistency is enhanced with bounded position drifts.

Although some achievements have been made, the limitations of this work include: (i) offline training is required, which limits objects to pre-defined categories. (ii) Experiments are still at the preliminary stage, where only cars in urban areas have been considered in the detector. In the future, the authors believe that the detector could be enhanced by introducing more data in a variety of environments. As for feature selection in CF, the possibilities of applying other features (such as LBP [38], Haar [39] and other deep features) that describe event count images could be explored. The data association could be improved by adding appearance discrepancies between detected and predicted bounding boxes into the assignment cost function. It is also promising to reduce the time consumption by exploiting data compression mechanisms.

## 7 References


[1] Sridhar, S., Mueller, F., Zollhöfer, M., *et al.*: 'Real-time joint tracking of a hand manipulating an object from RGB-D input'. European Conf. on Computer Vision, Springer, Amsterdam, The Netherlands, 2016, pp. 294–310

[2] Javed, O., Shah, M.: 'Tracking and object classification for automated surveillance'. European Conf. on Computer Vision, Springer, Copenhagen, Denmark, 2002, pp. 343–357

[3] Mendes, A., Bento, L.C., Nunes, U.: 'Multi-target detection and tracking with a laser scanner'. IEEE Intelligent Vehicles Symp., 2004, Parma, Italy, 2004, pp. 796–801

[4] Zhao, D., Fu, H., Xiao, L., *et al.*: 'Multi-object tracking with correlation filter for autonomous vehicle', *Sensors*, 2018, **18**, (7), p. 2004

[5] Redmon, J., Divvala, S., Girshick, R., *et al.*: 'You only look once: unified, real-time object detection'. Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 2016, pp. 779–788

[6] Henriques, J.F., Caseiro, R., Martins, P., *et al.*: 'High-speed tracking with kernelized correlation filters', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2014, **37**, (3), pp. 583–596

[7] Wang, N., Shi, J., Yeung, D.-Y., *et al.*: 'Understanding and diagnosing visual tracking systems'. Proc. of the IEEE Int. Conf. on Computer Vision, Las Condes, Chile, 2015, pp. 3101–3109

[8] Luo, W., Xing, J., Milan, A., *et al.*: 'Multiple object tracking: a literature review'. arXiv preprint arXiv:1409.7618, 2014

[9] Okuma, K., Taleghani, A., De Freitas, N., *et al.*: 'A boosted particle filter: multitarget detection and tracking'. European Conf. on Computer Vision, Springer, Prague, Czech Republic, 2004, pp. 28–39

[10] Avidan, S.: 'Ensemble tracking', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2007, **29**, (2), pp. 261–271

[11] Breitenstein, M.D., Reichlin, F., Leibe, B., *et al.*: 'Robust tracking-by-detection using a detector confidence particle filter'. 2009 IEEE 12th Int. Conf. on Computer Vision, Kyoto, Japan, 2009, pp. 1515–1522

[12] Orchard, G., Meyer, C., Etienne-Cummings, R., *et al.*: 'HFirst: a temporal approach to object recognition', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2015, **37**, (10), pp. 2028–2040
[13] Li, J., Shi, F., Liu, W., *et al.*: 'Adaptive temporal pooling for object detection using dynamic vision sensor'. British Machine Vision Conf. (BMVC), London, UK, 2017
[14] Iacono, M., Weber, S., Glover, A., *et al.*: 'Towards event-driven object detection with off-the-shelf deep learning'. 2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 1–9
[15] Cannici, M., Ciccone, M., Romanoni, A., *et al.*: 'Event-based convolutional networks for object detection in neuromorphic cameras'. arXiv preprint arXiv:1805.07931, 2018
[16] Saner, D., Wang, O., Heinzle, S., *et al.*: 'High-speed object tracking using an asynchronous temporal contrast sensor'. VMV, Darmstadt, Germany, 2014, pp. 87–94
[17] Cheung, B., Rutten, M., Davey, S., *et al.*: 'Probabilistic multi hypothesis tracker for an event based sensor'. 2018 21st Int. Conf. on Information Fusion (FUSION), Cambridge UK, 2018, pp. 1–8
[18] Mitrokhin, A., Fermüller, C., Parameshwara, C., *et al.*: 'Event-based moving object detection and tracking'. 2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS) IEEE, Madrid, Spain, 2018, pp. 1–9
[19] Barranco, F., Fermuller, C., Ros, E.: 'Real-time clustering and multi-target tracking using event-based sensors'. 2018 IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018, pp. 5764–5769
[20] Camuñas-Mesa, L.A., Serrano-Gotarredona, T., Ieng, S.-H., *et al.*: 'Event-driven stereo visual tracking algorithm to solve object occlusion', *IEEE Trans. Neural Netw. Learn. Syst.*, 2018, **29**, (9), pp. 4223–4237
[21] Ramesh, B., Zhang, S., Lee, Z.W., *et al.*: 'Long-term object tracking with a moving event camera'. British Machine Vision Conf., Newcastle, UK, 2018, vol. 2
[22] Liu, H., Moeys, D.P., Das, G., *et al.*: 'Combined frame-and event-based detection and tracking'. 2016 IEEE Int. Symp. on Circuits and Systems (ISCAS), Montreal, Canada, 2016, pp. 2511–2514
[23] Gallego, G., Delbrück, T., Orchard, G., *et al.*: 'Event-based vision: a survey'. CoRR, vol. abs/1904.08405, 2019. Available at http://arxiv.org/abs/1904.08405=0pt
[24] Dalal, N., Triggs, B.: 'Histograms of oriented gradients for human detection'. Int. Conf. on Computer Vision & Pattern Recognition (CVPR'05), IEEE Computer Society, San Diego, CA, USA, 2005, vol. 1, pp. 886–893
[25] Vondrick, C., Khosla, A., Malisiewicz, T., *et al.*: 'Hoggles: visualizing object detection features'. Proc. of the IEEE Int. Conf. on Computer Vision, Darling Harbour, Australia, 2013, pp. 1–8
[26] Wang, N., Yeung, D.-Y.: 'Learning a deep compact image representation for visual tracking', *Adv. Neural. Inf. Process. Syst.*, 2013, **26**, pp. 809–817
[27] Danelljan, M., Hager, G., Shahbaz Khan, F., *et al.*: 'Convolutional features for correlation filter based visual tracking'. Proc. of the IEEE Int. Conf. on Computer Vision Workshops, Las Condes, Chile, 2015, pp. 58–66
[28] Yi, K.M., Trulls, E., Lepetit, V., *et al.*: 'Lift: learned invariant feature transform'. European Conf. on Computer Vision, Springer, Amsterdam, The Netherlands, 2016, pp. 467–483
[29] Rifkin, R., Yeo, G., Poggio, T., *et al.*: 'Regularized least-squares classification', *Nato Sci. Series Sub Series III Comput. Syst. Sci.*, 2003, **190**, pp. 131–154
[30] Munkres, J.: 'Algorithms for the assignment and transportation problems', *J. Soc. Ind. Appl. Math.*, 1957, **5**, (1), pp. 32–38
[31] Miller, M.L., Stone, H.S., Cox, I.J.: 'Optimizing murty's ranked assignment method', *IEEE Trans. Aerosp. Electron. Syst.*, 1997, **33**, (3), pp. 851–862
[32] Redmon, J., Farhadi, A.: 'YOLOv3: an incremental improvement'. arXiv preprint arXiv:1804.02767, 2018
[33] Kim, D.Y., Jeon, M.: 'Data fusion of radar and image measurements for multi-object tracking via kalman filtering', *Inf. Sci.*, 2014, **278**, pp. 641–652
[34] Kulikov, G.Y., Kulikova, M.V.: 'The accurate continuous-discrete extended Kalman filter for radar tracking', *IEEE Trans. Signal Process.*, 2015, **64**, (4), pp. 948–958
[35] Altendorfer, R., Wirkert, S.: 'Why the association log-likelihood distance should be used for measurement-to-track association'. 2016 IEEE Intelligent Vehicles Symp. (IV), Gotenburg, Sweden, 2016, pp. 258–265
[36] Ristani, E., Solera, F., Zou, R., *et al.*: 'Performance measures and a data set for multi-target, multi-camera tracking'. European Conf. on Computer Vision, Springer, 2016, Amsterdam, The Netherlands, pp. 17–35
[37] Bernardin, K., Stiefelhagen, R.: 'Evaluating multiple object tracking performance: the clear mot metrics', *J. Image Video Process.*, 2008, **2008**, p. 1
[38] Ojala, T., Pietikäinen, M., Mäenpää, T.: 'Multiresolution gray-scale and rotation invariant texture classification with local binary patterns', *IEEE Trans. Pattern Anal. Mach. Intell.*, 2002, **24**, (7), pp. 971–987
[39] Viola, P., Jones, M.: 'Rapid object detection using a boosted cascade of simple features'. IEEE Computer Society Conference on Computer Vision and Pattern Recognition . IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR (1), Kauai, HI, USA, 2001, vol. 1, pp. 511–518