Long-term object tracking with a moving event camera

Bharath Ramesh[1]
bharath.ramesh03@u.nus.edu
Shihao Zhang[2]
a0129916@u.nus.edu
Zhi Wei Lee[2]
a0104271@u.nus.edu
Zhi Gao[1]
tslgz@nus.edu.sg
Garrick Orchard[1]
garrickorchard@nus.edu.sg
Cheng Xiang[2]
elexc@nus.edu.sg

[1] Temasek Laboratories
National University of Singapore
Singapore

[2] Electrical and Computer Engineering
National University of Singapore
Singapore

Abstract

This paper presents a long-term object tracking algorithm for event cameras. A first of its kind for these revolutionary cameras, the tracking framework uses a discriminative representation for the object with online learning, and detects and re-tracks the object when it comes back into the field-of-view. One of the key novelties is the use of an event-based *local sliding window* technique that performs reliably in scenes with cluttered and textured background. In addition, Bayesian bootstrapping is used to assist real-time processing and boost the discriminative power of the object representation. Extensive experiments on a publicly available event camera dataset demonstrates the ability to track and detect arbitrary objects of various shapes and sizes. This is a significant improvement compared to earlier works that simply track objects as long as they are visible under simpler background settings. In other words, when the object re-enters the field-of-view of the camera, a *data-driven, global sliding window* based detector locates the object under different view-point conditions for subsequent tracking.

1 Introduction

Standard video cameras struggle to capture crisp images of scenes characterized by high dynamic range and motion, returning blurred or saturated images. To overcome these limitations, event cameras aim to emulate the important asynchronous property of the human retina, thus earning themselves the name "silicon retinas". Hence, an event camera has no global clock or shutter to record images in the traditional sense. Instead, each pixel individually adapts and responds to temporal changes in log intensity, and outputs an asynchronous event with the pixel address which gets a precise timestamp in the order of microseconds.

An event is characterized by a spatial location (x, y), timestamp t and a binary-valued polarity p, i.e., on-events ($p = 1$) are caused by a positive change in log-intensity and vice-versa for off-events ($p = 0$). In both cases, events triggered by brightness changes are likely to occur at the edges that delineate the structure of the scene, and thus removing redundancy with a much lower data rate compared to a standard VGA resolution video at 30 fps. Although redundancy is absent in the event stream, the higher time-resolution should in principle contain all the information of a standard video without bounds on frame-rate and dynamic range. The image reconstruction from a pure event stream lends support to this idea [8].

Despite object tracking being a major research topic in computer vision, applicability has been limited by the low camera dynamics of standard vision sensors. Increasing the frame rate only burdens the computation techniques [28], preventing a dynamic, low-latency formulation of object tracking. On the other hand, the discontinuous motion information captured using a standard 30 fps video camera is an obvious disadvantage for frame-based object tracking algorithms.

This paper introduces a simple and efficient object tracking framework, consisting of a local *tracker* and a global *detector*, by taking advantage of the sparsity and higher temporal resolution of the event camera. In other words, the position of the object in the field-of-view of the event camera changes with negligible spatio-temporal discontinuity (5-10 μs). Therefore, the key idea is to spatially limit the search region of the tracked object while the temporal limits are imposed by the rate at which events arrive within the search region. In particular, the tracker search is modeled as a discriminative classification scheme (object vs. background) using the event-based descriptor proposed in [20]. Therefore, given the initial location of the object within a short time-interval, the training phase of the tracker learns a binary classifier. Subsequently, an object detector is learned using the training samples of the tracker.

The proposed framework is similar in spirit to the Tracking-Learning-Detection (TLD) system for frame-based cameras [7], nonetheless significantly different in the methods suited to event-based vision. We term our event-based object tracking framework as *e-TLD*. In summary, the objective is to develop a real-time long-term tracking system to achieve:

1. Continuous, long-term robust tracking under background change, illumination change and scale change.
2. Re-capturing the target after temporally occluded by other objects or when it re-appears after exiting.

2 Event Cameras

We use the commercial event camera, the Dynamic and Active-pixel Vision Sensor (DAVIS) [1] shown in Fig. 1(a). It has 240 × 180 resolution, 130 dB dynamic range and 3 microsecond latency, and communicates with a host computer using USB 2.0. It concurrently outputs a stream of events and frame-based intensity read-outs using the same pixel array. As mentioned earlier in Sec. 1, an event consists of a pixel location, a binary polarity value for positive or negative change in log intensity and a timestamp in microseconds. The event camera output can be visualized as shown in Fig. 1(b) by accumulating events within a short time-period (40ms in this case). In our work, polarity of the events are not considered, so both on-events and off-events are shown in white and the black regions correspond to inactive pixels. Note that only the event data of the DAVIS is used in this work.

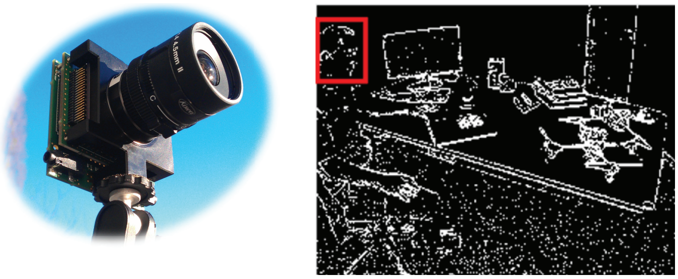

Figure 1: (a) The DAVIS event camera used in this work (source:inivation.com) (b) The accumulated events shown as a single frame, in which the tracker is initialized with the an arbitrary object position (source: The Event Camera Dataset [17]).

2.1 Related Work

The recent deep learning revolution in computer vision has also influenced neuromorphic vision with many works primarily in machine learning [5, 12, 18, 19, 25]. Besides learning, simultaneous localization and mapping (SLAM) is a trending robotics application using silicon retinas [9, 10, 16, 17, 22, 23]. On a larger front, these revolutionary cameras allow new perspectives in reformulating traditional vision problems, such as object detection and tracking, which largely remains an unexplored area of research.

A few works have used the event camera for object tracking with focus on specific application scenarios. One of the first object tracking applications demonstrated using the commercial dynamic vision sensor (DVS) was to track and control the position of a pencil balanced on a robot arm using a fast event-based Hough transform [2]. Other works focused on event-based algorithms for traffic monitoring [6, 13] from a static sensor point-of-view, and consequently, tracking can be treated without background modeling as only dynamic objects are picked out by the static event camera. Similarly, the robot goalie application [4] also takes advantage of the stationary DVS camera for tracking multiple balls.

A handful of works have attempted to tackle tracking of objects from a moving event camera. Using the DAVIS, [14] used a convolutional neural network (CNN) to detect likely target locations for tracking from a moving platform. However, a hybrid approach with frames and events naturally loses the advantages of a low-latency, purely event-driven approach. Nonetheless, general purpose object tracking works [11, 26] using event-based approaches have been proposed to track incoming blobs of events based on local shape properties. Although these methods are capable of adapting its shape and position to the distribution of incoming events, they carry motion assumptions such as a bivariate gaussian distribution. Thus, the algorithm parameters were defined experimentally according to the target to track, as acknowledged in [26]. Moreover, the previous systems are not suited to track a set of patterns/object as a whole. Finally, [11, 26] are not suited for long-term tracking, because there is no detector to re-initialize the tracker after a failed track.

In contrast to the above works, an object tracking and detection framework has been recently proposed [21] with the following limitations. As acknowledged by the authors of [21], their method works only when there is a clean background surrounding the target object. Secondly, the training phase of the detector in [21] is not data-driven (less reliable in practice) and uses computationally expensive image processing approaches for locating the most probable object candidate. Thus, [21] works only for simple shapes, as shown in their

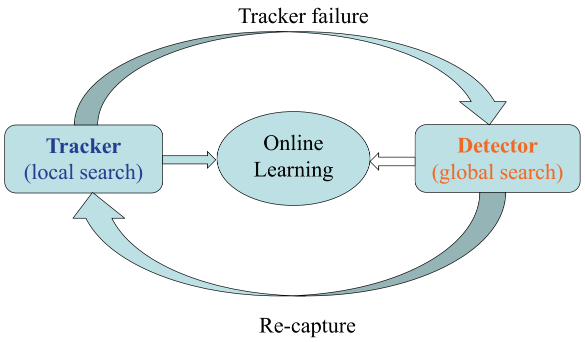

Figure 2: Tracker and detector flow of the proposed e-TLD framework.

accompanying video results.

In this paper, we propose a general purpose discriminative tracking system using a *local sliding window* approach, whose parameters are intuitive and can be easily generalized for a wide variety of objects having different shapes and sizes in cluttered settings, as shown in Fig. 1(b). Moreover, we propose a *data-driven* approach for training the object detector and a *global sliding window* method for locating the object, which allows detecting even small objects, like the drinking can near the monitor in Fig. 1(b).

3 Our method: e-TLD

The proposed e-TLD framework integrates a tracker and detector, as shown in Fig. 2. The event-based object tracker (Sec. 3.1) is a local search that requires initialization and outputs smooth trajectories. However, it cannot recover from failure on its own. The event-based object detector (Sec. 3.2), on the other hand, is a global search that does not assume anything about the previous position of the object, and is relatively slower compared to the tracker. However, we can achieve real-time processing by activating the detector only when tracker failure happens.

During the tracking process, online learning is needed to account for the changes in object appearance. In particular, the binary classifier used by the tracker is updated when the region-of-interest (ROI) is classified as the object. Updating the tracker mitigates the drifting issue and it is updated when the tracking confidence is higher than the mean tracking score. If tracking failure happens, a higher confidence value is needed to re-activate the tracker. In other words, the target will be re-tracked only when it can pass both the detector and a more "strict" tracker. The following subsections describe the e-TLD framework to jointly track and detect the object.

3.1 Event-based object tracker

Each time a ROI is classified as object, a small padding ensures the search area contains the object at the next instance of classification, as shown in Fig. 3. The position of the object is then updated with the candidate ROI with the highest classification score. This process is extremely simple, but works extremely well in challenging cluttered conditions due to the high-temporal resolution of the event camera. Note that we set the classification period in

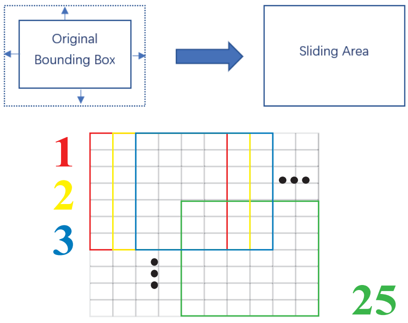

Figure 3: Local sliding window for object tracking using event cameras. A small padding ensures the sliding area contains the object in the next instance of classification. As shown in the example above, a padding of two pixels in x and y directions creates 25 candidate windows (best viewed on monitor).

terms of the number of events received within the ROI, instead of explicitly choosing a time-period. In particular, this number is chosen as a small fraction of the bounding box size and thus allows a dynamic classification rate for different object shapes and sizes.

We employ the feature descriptor proposed in [20], and thus, each event is encoded as a local descriptor. The notation $\mathbf{e}_i = (x_i, y_i, t_i, p_i, \mathbf{x}_i^T)^T$ denotes an event with pixel location x_i and y_i, timestamp t_i, polarity p_i and the feature vector $\mathbf{x}_i$.

We denote by N the number of candidate windows, and by $X^j = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{n_i})$ the collection of event descriptors contained within a candidate window W^j where $\mathbf{x}_l \in \mathbb{R}^d$, $l = 1, 2, \cdots, n_i$ is a descriptor in feature space S.

Inspired by the bag-of-words (BOW) model in computer vision [3], each feature vector $\mathbf{x}_l$ is quantized into one of K different visual words that are obtained from the training phase. The mapping to a visual word $v_k \in S$ is achieved using a quantization function $f_k(\mathbf{x}) : S \mapsto [0, 1]$. Each quantization function $f_k(\mathbf{x})$ is essentially computing the distance of the feature vector to v_k and allowing the assignment if it is minimal.

$$f_k(\mathbf{x}) = f(\mathbf{x}; v_k) = I(||\mathbf{x} - v_k|| = \rho) \tag{1}$$

where indicator function $I(z)$ outputs 1 when z is true or 0 otherwise; ρ is the Euclidean distance, $\arg\min_k ||\mathbf{x} - v_k||$. Given K visual words, or K quantization functions $\{f_k(\mathbf{x})\}_{k=1}^K$, a codeword representation is computed as,

$$h_j^k = \frac{1}{n_i} \sum_{l=1}^{n_i} f_k(\mathbf{x}_l) \tag{2}$$

The tracker representation for W^j is expressed by the vector,

$$\mathbf{h}_j = (h_j^1, h_j^2, \cdots, h_j^K) \tag{3}$$

Each incoming event in a candidate window W^j is then used to update the tracker representation $\mathbf{h}_j \in \mathbb{R}^K$. The scalar-valued discriminant function $D(\mathbf{h}_j)$ indicates the presence (class ω_1) or absence of the object (class ω_2) dynamically,

$$D(\mathbf{h}_j) \gtrless 0 \Rightarrow \mathbf{h}_j \in \left\{ \begin{array}{c} \omega_1 \\ \omega_2 \end{array} \right\} \tag{4}$$

During the training phase where the user specifies a tight ROI in space-time that contains the object, all the events including ones outside the ROI are used to obtain the parameters of D, which is the problem of constructing a classifier for two classes – object vs background.

3.1.1 Training phase

When the user specifies the spatio-temporal position of the object, the first step is create the visual words $\{v_k\}_{k=1}^{K} \in S$, which are the cluster centers generated using k-means clustering of the event descriptors inside and outside the ROI. In other words, the codebook $C = [v_1, v_2, \cdots, v_K] \in \mathbb{R}^{d\times K}$ is an unsupervised learning step. Then the events within the ROI, represented by the set of descriptors $X^{\omega_1} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{C_1})$ can be used to generate a tracker representation $\mathbf{h}_{\omega_1}$ (3). Similarly, the events outside the ROI $X^{\omega_2} = (\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{C_2})$ can be used for obtaining $\mathbf{h}_{\omega_2}$. However, training a classifier with just one sample from each class ($\mathbf{h}_{\omega_1}$ and $\mathbf{h}_{\omega_2}$) is pointless.

To solve the low sample problem, statistical bootstrapping [15] can be used to generate new subsets of descriptors $\{X_1^{\omega_1}, X_2^{\omega_1}, \cdots, X_{n_1}^{\omega_1}\}$ and $\{X_1^{\omega_2}, X_2^{\omega_2}, \cdots, X_{n_2}^{\omega_2}\}$. Specifically, bootstraping X^{ω_1} is the process of random sampling of a subset out of the C_1 descriptors, one at a time such that all descriptors have an equal probability of being selected, i.e., $1/C_1$.

However, storing a set of events or descriptors for bootstrapping (X^{ω_1} and X^{ω_2}) is impractical for online learning on an event-by-event basis. Thus, we propose bootstrapping to be interpreted in a Bayesian framework [24] that re-weights the histogram representations ($\mathbf{h}_{\omega_1}$ and $\mathbf{h}_{\omega_2}$). Let $P \sim U([0,1])$ be a uniformly distributed random variable. Mathematically,

$$h_{1\omega_1}^{k} = \lfloor P \times h_{\omega_1}^{k} \rfloor \tag{5}$$

Thus, the first bootstrapped histogram representation for the ROI is expressed by the vector,

$$\mathbf{h}_{1\omega_1} = (h_{1\omega_1}^{1}, h_{1\omega_1}^{2}, \cdots, h_{1\omega_1}^{K}) \tag{6}$$

It is to be noted that (6) is not a true bootstrap procedure since the maximum values of $h_{1\omega_1}^{k}$ need not be clipped to the corresponding maxmimum values of $h_{\omega_1}^{k}$, as seen in (5). However, the Bayesian bootstrap is operationally and inferentially similar to the true boostrap [24]. Let N_1 and N_2 denote the number of samples after bootstrapping belonging to class ω_1 and ω_2 respectively. Then, the collection of the bootstrapped representations $\{\mathbf{h}_{1\omega_1}, \mathbf{h}_{2\omega_1}, \cdots, \mathbf{h}_{N_1\omega_1}\}$ and $\{\mathbf{h}_{1\omega_2}, \mathbf{h}_{2\omega_2}, \cdots, \mathbf{h}_{N_2\omega_2}\}$ is used to train a support vector machine (SVM) classifier $D(\cdot)$ with an efficient additive χ^2 kernel [27].

3.1.2 Tracking Phase

The candidate windows $\{W^j\}_{j=1}^{N}$ each output a tracker representation $\mathbf{h}_j$. The best candidate window is chosen to be the tracker state B_t when $D(\mathbf{h}_j)$ is maximized.

$$\operatorname*{arg\,max}_{X^j \in S} D(\mathbf{h}_j(X^j)) := \{X^j, j = 1, 2, \cdots, N \mid \forall Y^j : D(\mathbf{h}_j(Y^j)) \leq D(\mathbf{h}_j(X^j))\}. \tag{7}$$

The number of events for the ROI update ("waiting time" between two instances of classification) is set as $\tau \times height \times width$ of the ROI X^j, where $\tau \in [0,1]$ is set to 0.05 in our experiments. Thus, when the sliding area contains 5% of the events relative to the number of pixels contained within the ROI, the next instance of classification is triggered (see Fig.

Algorithm 1 Event-based Object Detection

Input: Image size $(h \times w)$, detection matrix $M = 0_{h,w}$, detect threshold τ_d, count $= 0$
Output: Estimated tracker state B_t with ROI width n and height m

```
 1: For each incoming event $\mathbf{e}_i = (x_i, y_i, t_i, p_i, \mathbf{x}_i^T)^T$
 2: for $k = 1 : K$ do
 3:     Get $f_k(\mathbf{x}_i)$ using (1)
 4:     if $f_k(\mathbf{x}_i) = 1$ and $k \in O_{\omega_1} = \{k_1, k_2, \cdots, k_o\}$ then
 5:         $M(y_i, x_i) = M(y_i, x_i) + 1$
 6:         count = count + 1
 7:     end if
 8: end for
 9: if (count $> \tau_d \times h \times w$) then
10:     Perform global sliding window operation (9) on $M$ to get $O \in \mathbb{R}_+^{(h-m+1)\times(w-n+1)}$
11:     Use highest activation in $O$ to re-initialize the tracker state $B_t$
12: end if
```

3). The average SVM score after several instances is used to determine whether the next instance of tracking is successful. In case, the SVM score falls below the average score, then the object detector is instantiated to globally search for the object.

3.2 Event-based object detector

Once the tracker has lost the object, detecting the object is the problem of obtaining a candidate ROI and continuing the tracking process. Therefore, detection is a global sliding window search compared to the local sliding window search of the tracker.

3.2.1 Training phase

Similar to the training phase of the tracker (Sec. 3.1.2), the object detector uses the ROI initialization by the user. Let o denote the number of quantized clusters to which the object samples X^{ω_1} are frequently mapped, and the corresponding cluster indices be $O_{\omega_1} = \{k_1, k_2, \cdots, k_o\}$ where $o \ll K$. The objective of the proposed detector training phase is to obtain O_{ω_1} in a data-driven fashion without relying on ad-hoc threshold parameters.

The main idea is to deduce clusters that are important to X^{ω_1} while rejecting quantization results that are common to X^{ω_1} and X^{ω_2}. By making use of the Bayesian bootsrapped representations, a new vector $\mathbf{h}^{\text{diff}}_{\omega_1\omega_2} \in \mathbb{R}^K$ is used to obtain object clusters for the detection process,

$$\mathbf{h}^{\text{diff}}_{\omega_1\omega_2} = \sum_{l=1}^{N_1} \mathbf{h}_{l\omega_1} - \sum_{m=1}^{N_2} \mathbf{h}_{m\omega_2} \tag{8}$$

The positive values in $\mathbf{h}^{\text{diff}}_{\omega_1\omega_2}$ represent codewords that have been assigned to the object more times than it has been assigned to background. Therefore, these codewords are simply chosen to be O_{ω_1}. In the current e-TLD setup, the detector is not updated on-the-fly, as it requires online dictionary learning, which remains a future direction of research.

3.2.2 Detection phase

If the event camera output contains h rows and w columns, a detection matrix $M \in \mathbb{R}_+^{h \times w}$ is used to keep track of events that may belong to the object. For every incoming event, the quantization function (1) determines whether it belongs to the detector clusters $\{k_1, k_2, \cdots, k_o\}$ and the corresponding location of the event is used to increment M. The detector threshold $\tau_d \times h \times w$, where $\tau_d \in [0,1]$, determines if enough events have been accumulated within the detection matrix M. The parameter τ_d is set to 0.10, meaning at least 10% of the object events have occurred for the detection process.

A global sliding window process is then performed on M to determine the region with maximal activation due to the presence of the object (if any). If the previous successful object state B_t has m rows and n columns, then the size of the activation map after the global sliding window operation will be $h - m + 1$ rows, and $w - n + 1$ columns.

$$O(r,s) = \sum_{k=1}^{m} \sum_{l=1}^{n} M(r+k-1, s-l+1) \tag{9}$$

where r and s varies from $1, 2, \cdots, h-m+1$ and $1, 2, \cdots, w-n+1$ respectively. The region centered at $max(O(r,s))$ is the new tracker state B_t. Alg. 1 outlines the proposed event-based object detection approach.

4 Experiments

We recommend viewing our submitted videos[1] that clearly show the tracking process better than still images. We tested e-TLD using a publicly available event-camera dataset [17] and in-house data processed in real-time on a standard PC. The experimental setup for testing on the dataset are as follows.

For each object, an ROI was manually specified during the first 300ms of the recording (training data) and the rest of the recording was used for testing. A small codebook size of $K = 200$ was used to model the ROI and the background. The SVM training is performed with Bayesian bootstrapping that outputs equal number of samples as the initial number of descriptors. For example, if there are $N_1 = 840$ ROI descriptors at the user initialization state, N_1 samples having $N_1/2$ descriptors in each sample are obtained after bootstrapping.

4.1 Translational camera motion

As shown in Figure 4, e-TLD is able to track and detect objects of various sizes and shapes. In these results, the overlaid markers indicate the position of the tracked object in the field-of-view of the event camera.

4.2 6-DOF camera motion

Although the appearance of the object changed considerably during the translational camera motion, rotation was intentionally kept minimal in these recordings by the authors of the event camera dataset. Separate recordings of the same scene are available for the general 6-DOF camera motion, which induces drastic view-point change of the object. Figure 5

[1]https://youtu.be/3cigR9Al23A

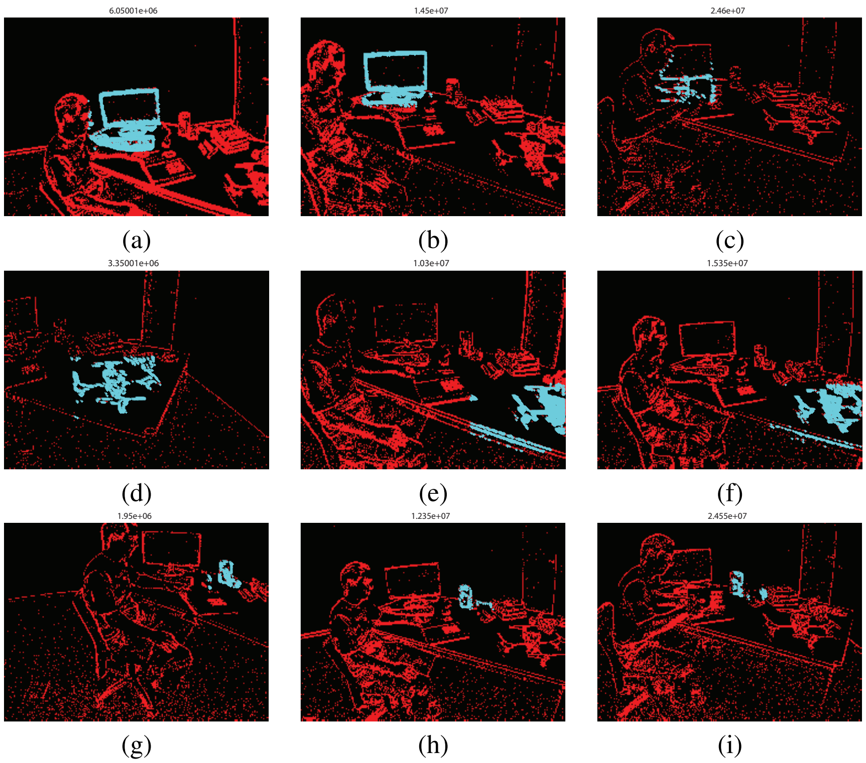

Figure 4: Tracking of objects having various shapes and sizes. Each each row shows the tracking of a single object. The marked events represent the tracked position in the field-of-view of the event camera and the title of each subfigure displays the time (μs) instance of the track.

shows the tracking of the drone object under drastic view-point variations, showing robustness of the e-TLD system to scale and rotation. Note that quantitative evaluation has not been possible due to the unavailability of ground truth in the event camera dataset. As the neuromorphic research field becomes mature, we expect ground truth comparison to play a vital role in comparison of different object tracking frameworks.

4.3 Real-time testing

Real-time testing was carried out using a DAVIS camera, interfaced and powered by an Intel ComputeStick running an Intel Core m5-6Y57 vPro processor, on-board an unmanned aerial vehicle (UAV). The e-TLD framework was implemented in C++ using Visual Studio IDE for Windows 10. A standard global shutter camera is likely to generate images with motion blur artifacts while the UAV is constantly in motion. However, the event camera and our algorithm are able to track the object, as shown in this video[2] where all elements are clear.

[2]https://youtu.be/Fv38sltqis8

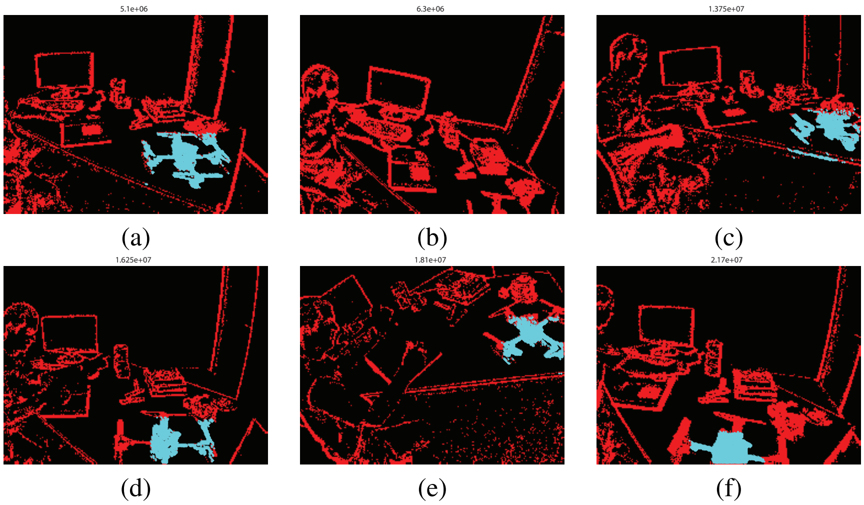

Figure 5: Tracking of the drone object under general 6-DOF camera motion.

5 Conclusion

We believe these are breakthrough results, showing how an event-based tracker and detector permits the application of an event camera to the important problem of long-term object tracking in real-time, and hopefully this opens the door to similar approaches for other related vision problems. It is worth restating that the data rate of the DAVIS event camera used in our experiments is typically in the order of 150 KB/s while a standard grayscale VGA camera outputs frames at 30Hz or about 10MB/s. The only information that is important for tracking and detection is how edges move, and the event camera naturally outputs this information while sidestepping problems of blur, low-dynamic range and limited motion information that standard cameras create.

Acknowledgements

The authors would like to thank Luca Della Vedova and Chen Yu for helping collect data and carry out experiments using the quadcopter. Additionally, the authors thank M.S. Muthukaruppan for discussion and comments on the manuscript.

References

[1] C. Brandli, R. Berner, M. Yang, S. C. Liu, and T. Delbruck. A 240 x 180 130 db 3 μs latency global shutter spatiotemporal vision sensor. *IEEE Journal of Solid-State Circuits*, 49(10):2333–2341, Oct 2014.

[2] J. Conradt, R. Berner, M. Cook, and T. Delbruck. An embedded AER dynamic vision sensor for low-latency pole balancing. In *IEEE International Conference on Computer Vision Workshop*, pages 780–785, Sept 2009.

[3] Gabriella Csurka, Christopher R. Dance, Lixin Fan, Jutta Willamowski, and Cedric Bray. Visual categorization with bags of keypoints. In *In Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.

[4] Tobi Delbruck and Manuel Lang. Robotic goalie with 3 ms reaction time at 4% CPU load using event-based dynamic vision sensor. *Frontiers in Neuroscience*, 7:223, 2013.

[5] R. Ghosh, A. Mishra, G. Orchard, and N. V. Thakor. Real-time object recognition and orientation estimation using an event-based camera and CNN. In *IEEE Conference on Biomedical Circuits and Systems (BioCAS)*, pages 544–547, 2014.

[6] Gereon Hinz, Guang Chen, Muhammad Aafaque, Florian Röhrbein, Jörg Conradt, Zhenshan Bing, Zhongnan Qu, Walter Stechele, and Alois Knoll. Online multi-object tracking-by-clustering for intelligent transportation system with neuromorphic vision sensor. In *KI 2017: Advances in Artificial Intelligence*, pages 142–154. Springer International Publishing, 2017.

[7] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1409–1422, 2012.

[8] Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew J Davison. Simultaneous mosaicing and tracking with an event camera. In *British Machine Vision Conference (BMVC)*, 2014.

[9] Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera. In *Computer Vision – ECCV 2016*, pages 349–364. Springer International Publishing, 2016.

[10] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza. Low-latency visual odometry using event-based feature tracks. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 16–23, Oct 2016.

[11] X. Lagorce, C. Meyer, S. H. Ieng, D. Filliat, and R. Benosman. Asynchronous event-based multikernel algorithm for high-speed visual features tracking. *IEEE Transactions on Neural Networks and Learning Systems*, 26(8):1710–1720, 2015.

[12] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers in Neuroscience*, 10:508, 2016.

[13] M. Litzenberger, C. Posch, D. Bauer, A. N. Belbachir, P. Schon, B. Kohn, and H. Garn. Embedded vision system for real-time object tracking using an asynchronous transient vision sensor. In *12th IEEE Digital Signal Processing Workshop*, pages 173–178, Sept 2006.

[14] H. Liu, D. P. Moeys, G. Das, D. Neil, S. C. Liu, and T. Delbruck. Combined frame- and event-based detection and tracking. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 2511–2514, May 2016.

[15] Christopher Z Mooney and Robert D Duval. *Bootstrapping: A nonparametric approach to statistical inference*. Number 94-95. SAGE, 1993.

[16] E. Mueggler, B. Huber, and D. Scaramuzza. Event-based, 6-DOF pose tracking for high-speed maneuvers. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2761–2768, 2014.

[17] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *The International Journal of Robotics Research*, 36(2): 142–149, 2017.

[18] D. Neil, M. Purghart, and S-C. Liu. Effective sensor fusion with event-based sensors and deep network architectures. In *IEEE International Symposium on Circuits and Systems*, 2016.

[19] Peter O'Connor, Daniel Neil, Shih-Chii Liu, Tobi Delbruck, and Michael Pfeiffer. Real-time classification and sensor fusion with a spiking deep belief network. *Frontiers in Neuroscience*, 7, 2013.

[20] B. Ramesh, N. A. Le Thi, G. Orchard, and C. Xiang. Spike context: A neuromorphic descriptor for pattern recognition. In *IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–4, Oct 2017.

[21] Bharath Ramesh, Hong Yang, Garrick Orchard, Ngoc Anh Le Thi, and Cheng Xiang. DART: distribution aware retinal transform for event-based cameras. *CoRR*, abs/1710.10800, 2017.

[22] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza. EVO: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. *IEEE Robotics and Automation Letters*, 2(2):593–600, 2017. ISSN 2377-3766.

[23] Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. EMVS : Event-based multi-view stereo. *Bmvc*, pages 1–11, 2016.

[24] Donald B. Rubin. The Bayesian Bootstrap. *Ann. Statist.*, 9(1):130–134, 1981.

[25] Evangelos Stromatias, Miguel Soto, Teresa Serrano-Gotarredona, and Bernabe Linares-Barranco. An event-driven classifier for spiking neural networks fed with synthetic or dynamic vision sensor data. *Frontiers in Neuroscience*, 11:350, 2017.

[26] D. Reverter Valeiras, X. Lagorce, X. Clady, C. Bartolozzi, S. H. Ieng, and R. Benosman. An asynchronous neuromorphic event-driven visual part-based shape tracking. *IEEE Transactions on Neural Networks and Learning Systems*, 26(12):3045–3059, 2015.

[27] A. Vedaldi and A. Zisserman. Efficient additive kernels via explicit feature maps. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3539–3546, 2010.

[28] Yoshihiro Watanabe, Hiromasa Oku, and Masatoshi Ishikawa. Architectures and applications of high-speed vision. *Optical Review*, 21(6):875–882, Nov 2014.