Asynchronous Corner Detection and Tracking for Event Cameras in Real-Time

Originally published in:
IEEE Robotics and Automation Letters 3(4), https://doi.org/10.1109/lra.2018.2849882

Funding acknowledgement:
644128 - Collaborative Aerial Robotic Workers (SBFI)
157585 - Collaborative vision-based perception for teams of (aerial) robots (SNF)

Asynchronous Corner Detection and Tracking for Event Cameras in Real-Time

Ignacio Alzugaray and Margarita Chli

***Abstract*—The recent emergence of bio-inspired event cameras has opened up exciting new possibilities in high-frequency tracking, bringing robustness to common problems in traditional vision, such as lighting changes and motion blur. In order to leverage these attractive attributes of the event cameras, research has been focusing on understanding how to process their unusual output: an asynchronous stream of events. With the majority of existing techniques discretizing the event-stream essentially forming frames of events grouped according to their timestamp, we are still to exploit the power of these cameras.**

In this spirit, this paper proposes a new, purely event-based corner detector and a novel corner tracker, demonstrating that it is possible to detect corners and track them directly on the event-stream in real-time. Evaluation on benchmarking datasets reveals a significant boost in the number of detected corners and the repeatability of such detections over the state-of-the-art even in challenging scenarios with the proposed approach, while enabling more than a 4× speed-up when compared to the most efficient algorithm in the literature. The proposed pipeline detects and tracks corners at a rate of more than 7.5 million events per second, promising great impact in high-speed applications.

***Index Terms*—Visual tracking, computer vision for other robotic applications, SLAM.**

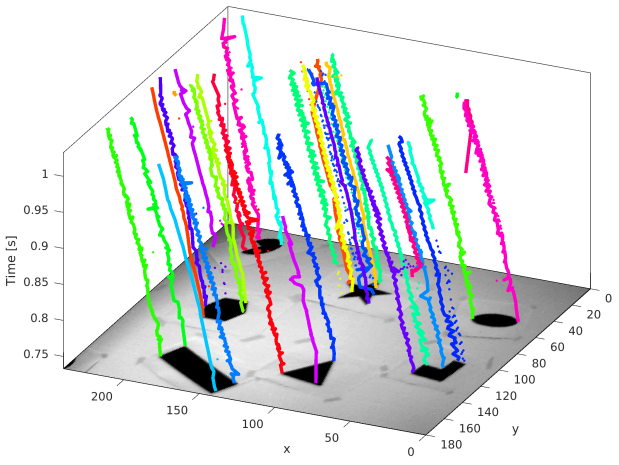

Fig. 1. The proposed algorithm operates asynchronously on the event-stream enabling reliable real-time corner detection and tracking even under high-speed motions, where conventional cameras would experience motion blur. Here, corners are identified in the event-stream on the `shapes` dataset [1] and data association is used to generate feature tracks corresponding to different colors (solid lines). The corresponding intensity image is also depicted for clarity.

I. Introduction

THE richness of information encoded in images together with the portability and affordability of visible-light cameras have justifiably set them as the *de facto* sensor of choice in a variety of applications. One of the most important milestones towards machine-vision perception has been the demonstration of the ability to estimate the egomotion and scene employing a single moving camera in real-time – commonly referred to as the Simultaneous Localisation And Mapping (SLAM) problem. Building on top of SLAM, several techniques have been developed over the years with a significant impact in Robotics, e.g. 3D scene reconstruction or place recognition

Despite their undeniable dominance in the Computer Vision, conventional frame-based cameras, however, also present several limitations derived from the most fundamentals of their design. Discretizing the visual perception of a scene at fixed frame-rate, global or rolling shutter cameras often suffer from capturing redundant information when the camera and the scene are static, for example. Conversely, in cases of a highly dynamic scene or camera motion such cameras often capture insufficient information. In the latter case, the captured images may exhibit motion blur or prohibitive image distortion when using global or rolling shutter cameras, respectively, resulting in unusable frames in both cases.

Due to these limitations, the emergence of biologically-inspired event cameras or Dynamic Vision Sensor (DVS) [2], [3] has captured the interest of the community. In event cameras, whenever the intensity of an individual pixel varies beyond a specified threshold, an 'event' triggers in such pixel location and is reported asynchronously and independently from the rest of the pixels in the image array. Going beyond the fixed frame-rate paradigm, event cameras compress the visual scene in an asynchronous event-stream with high temporal resolution (in the order of μs) while exhibiting much higher dynamic range (up to 120dB) and lower power consumption than conventional cameras.

In addition to the elimination of any conventional time-discretization (e.g., in the form of frames) in the perception pipeline, events only report incremental intensity changes at each pixel instead of the absolute intensity values (i.e., that conventional cameras capture in each frame). Consequentially, the Computer Vision community is driven to revisit even

Manuscript received: February 24, 2018; Revised May, 18, 2018; Accepted June 11, 2018. This paper was recommended for publication by Editor F. Chaumette upon evaluation of the Reviewers' comments.This research was supported by the Swiss National Science Foundation (Agreement no. PP00P2_157585) and EC's Horizon 2020 Programme under grant agreement no. 644128 (AEROWORKS).

The authors are with Vision for Robotics Lab, ETH Zurich, Zurich CH-8092, Switzerland (e-mail: ialzugaray@mavt.ethz.ch; chlim@ethz.ch). Corresponding author: Ignacio Alzugaray.

A video of the proposed algorithm is available at: https://youtu.be/bKUAZ7IQcf0

Digital Object Identifier (DOI): see top of this page.

the most established algorithms to process the event-stream. Particularly, the feature detection cannot be directly addressed by simply studying the distribution of the absolute intensity levels of a local image area. Established approaches to describe image regions, which most typically make use of the relative intensity levels in a local neighbourhood, are also compromised and thus, so is data association.

Driven by the need for effective feature processing on the event-stream, in this paper, we propose a novel feature (i.e., corner) detector and tracker, in a framework that runs asynchronously, i.e. considering only the sequential nature of the event-stream, in order to fully leverage the benefits of the event cameras regarding their high sensing rate. Moreover, the proposed approach is shown to be able to run in real-time even under high-speed camera motions. In brief, the contributions of this work are:

- a novel asynchronous **corner detector**, the 'Arc*' algorithm, with enhanced detection repeatability and significantly more efficient than the state-of-the-art,
- an efficient asynchronous **corner tracker**, able to establish correspondences across asynchronous detections in real-time and operating directly in the event-stream, and
- a simple and generic **event-based filter** that effectively reduces the event-stream to the most relevant events for corner detection and tracking, capable of speeding up any existing event-based algorithm.

II. Related work

Since the development of the DVS, several works have been dedicated to the identification of relevant features to track in the event-stream. The first approaches tackled the detection and tracking of simple features, such as lines [4], circles [5] or manually seeded regions in the scene [6].

As event cameras promise significant impact in high-rate motion and scene tracking, SLAM has become the holy grail of event-based research. Despite of the fact that the field is still in its infancy, event-based SLAM pipelines have already made their debut in the literature. Although some of them do not explicitly employ features [7], [8], avoiding the data association problem, most recent approaches implement interesting schemes to detect and track corner features [9], [10]. Nonetheless, the latter feature-based approaches, [9] and [10], make use of intermediate frame-like representations of the scene, rendered from the accumulation past events within an arbitrary window, in which they detect FAST corners [11] and track them using Expectation-Maximization (EM) and the Lukas-Kanade Tracker (KLT) [12], respectively. Both algorithms rely on the compensation of the camera motion to obtain the reliable frame-like representations of the scene, by either using an expensive EM scheme in [9] or using cues from an Inertial Measurement Unit (IMU) in [10].

In this work, we advocate for algorithms that do not require any integration of events for an intermediate frame-based representation. To exploit the true power of event cameras, the algorithms have to *asynchronously* operate directly on event-stream, this is, processing each event upon arrival while coping with the high data rate of the DVS to perform in real-time. In [13], for instance, incoming events are classified as corners upon arrival based on the optical flow orientation in the local neighbourhood [14]. The corner-events are then matched and tracked based on their estimated velocity and predicted position under the assumption of a constant velocity model. In [15] an adaptation of the original Harris corner detector [16] for event-streams is proposed. Employing a global window of events, upon the arrival of a new event, the time-stamp of the events last triggered in the neighborhood are used to compute the structure tensor from [16], achieving remarkable accuracy with moderate computational cost. An improved version of this method was later proposed in [17] (we refer to this as 'eHarris' for brevity), in which the global window of events is substituted by a local one. Additionally, [17] proposes a method to detect corners in the event-stream inspired by FAST [11] – we refer to this as 'eFAST'. Although eFAST is similar to eHarris in inspecting the timestamp of the latest events in the surrounding pixels, eFAST is significantly more efficient with a small trade-off in accuracy.

Inspired by eFAST, in this paper we present Arc*, a corner algorithm able to detect corners more than 4x faster eFAST and 50x faster than eHarris, while enhancing the repeatability of the corner detections. Additionally, we also propose an event-based corner tracker, demonstrated to operate successfully in real scenes while achieving real-time performance when compared to other state-of-the-art event-based trackers.

III. Methodology

A. Data Stream from Event-based Cameras

Let $I(x, y, t)$ be the log-intensity value measured at the pixel location (x, y), where an event triggered at time t. A new event e is generated if, after an arbitrary period of time Δt, the absolute difference of I reaches a specific threshold K. Formally,

$$\Delta I(x, y) = I(x, y, t + \Delta t) - I(x, y, t) = pK \ , \qquad (1)$$

where p denotes the event's polarity (i.e. is either 1 or -1) to indicate whether I increases or decreases. Ideally, a new event $e = \{t, x, y, p\}$ is generated as soon as this condition is met, and appended asynchronously to the event-stream. In practice, however, the triggering of events in a single pixel is also subject to the sensor/hardware internal configuration (electronic biases).

B. Filter of Redundant Events

To classify a new event as a corner-event upon arrival, we need to inspect previously triggered events in the stream. Since exploring all the previous events would not be scalable, we use a Surface of Active Events (SAE) [14], [17] to summarize the event-stream at any given instant. Briefly, the SAE $\mathcal{S}$ is defined as $\mathcal{S} : (x, y) \in \mathbb{R}^2 \mapsto t_l \in \mathbb{R}$, where t_l is the timestamp of the latest event triggered at the pixel location (x, y). Following the trend from [13], [15], [17], we separate the event-stream according to polarity and process the two sets independently in different SAEs.

The local neighbourhood in $\mathcal{S}$ of the new event's pixel location at its arrival time, i.e. the temporal ordering of the latest triggered events in the neighbouring pixels, is used

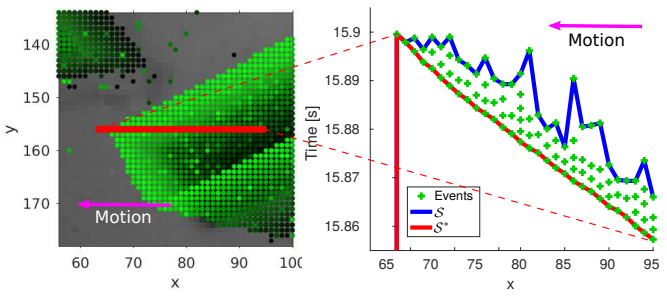

Fig. 2. On the left, real events generated due to translation of a black rectangle are depicted (brighter green indicates newer events). We illustrate the timestamps of the latest events triggered in a small region (red segment) on the right. Although a single contrast change occurs, the significant magnitude of the change induces the triggering of multiple events in each of the pixels. Our proposed filtered SAE $\mathcal{S}^*$ (red) captures the timestamps of the first event in each pixel, accurately representing the position of the rectangle at each time (note the constant velocity profile). A naive SAE $\mathcal{S}$ (blue) would capture the timestamp of consecutive latest events instead, which are subject to noise.

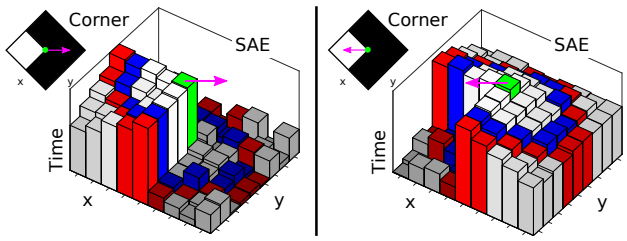

Fig. 3. We illustrate an example in which the same corner under two different motion directions (Magenta arrows) may induce completely different SAEs (Height represents the timestamps of the latest event triggered in each location). Our algorithm, Arc*, and eFAST successfully detect the corner on SAE depicted on the left, as the set of newest elements in in the inspected circles (Blue and red) are distributed continuously. However, only Arc* (and not eFAST) is able to detect the same corner from the SAE depicted on the right, where the angle of arc of newest elements is now over $180°$.

to test whether the new event is a corner. However, several factors may compromise the reliability of such ordering and thus the quality of the detection. A sudden and significant contrast change, for instance, would trigger multiple events in the same pixel almost instantly, according to Equation (1). Due to hardware limitations, however, there exists a minimum amount of time between consecutive events triggered at the same pixel. As a consequence, the latest timestamps registered in the SAE do not represent the time when visual stimulus was captured accurately, corrupting the local ordering of the events as illustrated in Fig. 2.

To overcome this undesired effect, we propose a more restrictive SAE that monitors a new variable, the reference time t_r, such that $\mathcal{S}^* : (x, y) \in \mathbb{R}^2 \mapsto (t_r, t_l) \in \mathbb{R}^2$. When querying for the values of each location in $\mathcal{S}^*$ (e.g., for corner detection), the stored reference time t_r is retrieved instead of the latest timestamp t_l as in $\mathcal{S}$. Upon the arrival of a new event at time t, the value of t_l in the location is always updated, $t_l \leftarrow t$, as for $\mathcal{S}$. However, the reference time t_r is only updated if the previous event in the same location was not triggered within the time-window κ, such that $t_r \leftarrow t$ if $t > t_l + \kappa$, or if the polarity of the latest event triggered in the same location differs from the polarity of the incoming one. Only the events that update the value of t_r in $\mathcal{S}^*$ are considered in the proposed algorithm, rejecting the rest of events as too noisy and redundant for corner detection. Experimentally, we observe that the threshold κ can be set to conservatively to high values, blocking most of events triggered in regions with smooth intensity changes, while relying on the updates of $\mathcal{S}^*$ due to change of polarity to cope with fast motions. Although κ could be dynamically modified or specifically tuned for each experiment, for the sake of fairness, we use a constant value of $\kappa = 50$ms in this work.

The benefits of using $\mathcal{S}^*$ are two-fold: (1) high contrast regions are more accurately represented spatially and temporally identifying when they were firstly detected, and (2) the event-stream to be considered for corner detection is reduced significantly by removing redundant events, drastically saving computation time. In Section IV, we evaluate both eHarris and eFAST using the proposed filter (i.e. using $\mathcal{S}^*$), referring to them as eHarris* and eFAST*, respectively, for a fair comparison with the proposed corner detector Arc*.

C. Event-based Corner Detection: The Arc* algorithm

The edges in the scene captured by the camera leave a trail of timestamps in the SAE that, ideally and in the absence of noise, decreases from the current edge's location towards the direction of relative motion (see Fig. 2) in the image plane. When detecting corner-events with the eFAST algorithm, a circular set of locations in the SAE is inspected, and a subset of these locations with newest timestamps is selected. If this subset is distributed as a contiguous circular arc and its angle is within a pre-defined range, the incoming event is classified as corner, similarly to the original FAST algorithm.

The maximum angle of the circular arc of newest elements in eFAST, however, must be less than $180°$ to prevent the incorrect classification of events generated by straight edges as corners. Nonetheless, depending on the direction of motion, the arc of newest events around the corner may exhibit an angle greater than $180°$, as illustrated in Fig. 3, which eFAST fails to detect (see Fig. 6). The proposed Arc* corner detector extends the detection initially proposed by eFAST to arcs that are greater than $180°$ by employing a significantly more efficient iterative algorithm that minimizes the number of operations to classify corner-events. As a consequence, Arc* achieves a better corner detector repeatability, effectively increasing the number of detected corners, while exhibiting better computational efficiency as shown in Section IV.

In the Arc* detector, described in Algorithm 1, the location of a new event is used as the center of a circular mask of a predefined radius as shown in Fig. 4. This circular mask defines a set of locations, from which we retrieve their value in SAE $\mathcal{S}^*$, using the same polarity as the new event, and create a circular set of elements C. We initialize the arc of newest elements A_{new} with the newest element in C and a pair of supporting elements in the adjacent clockwise (CW) and counter clockwise (CCW) positions, E_{CW} and E_{CCW}.

In each iteration of the algorithm we select the newest of the supporting elements E_{CW} or E_{CCW}. If the oldest element in A_{new} is older than the selected newest supporting

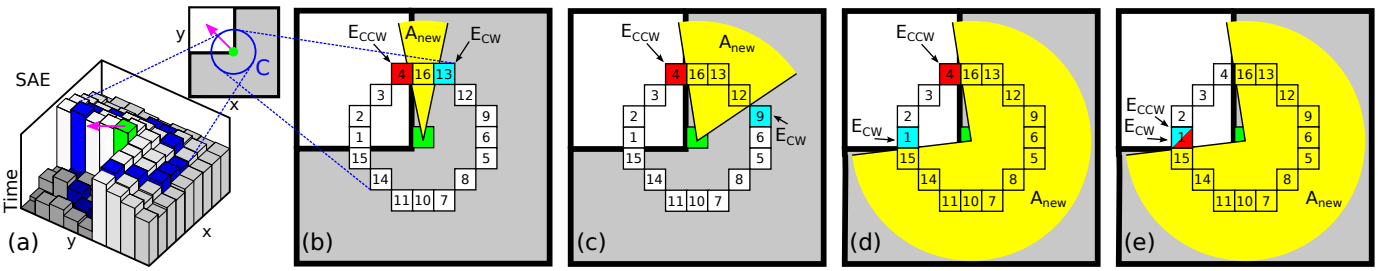

Fig. 4. Upon the arrival of a new event (green) located in a corner (grey background), we inspect a circular set of elements C (blue) in the local SAE as in (a). In (b), we initialize the arc A_{new} (yellow) from the newest element (higher values mean newer timestamps) and E_{CW} (cyan) and E_{CCW} (red) from the adjacent ones. We initially expand A_{new} up to a minimum size $L_{min} = 3$ using, in this case, the newest supporting element E_{CW} as in (c). In the following iterations, we keep updating the position of E_{CW} clockwise (since it is newer than E_{CCW} during these iterations) and only extend A_{new} as soon as E_{CW} reaches the positions with the values '14' and '15' as depicted in (d). In (e), the algorithm updates the position of the now newest supporting element E_{CCW}, until the circle C is completed. Here the incoming event is classified as a corner (conversely to eFAST) since the length of the complementary continuous arc $C \setminus A_{new}$, i.e. the elements not belonging to A_{new}, is 4 and thus, lies within the range $[L_{min}, L_{max}] = [3, 6]$ specified for this circle radius.

element, we expand A_{new} up to such element (including it) and update the position of the selected E_{CW} or E_{CCW} to the following CW or CCW position in the circle, respectively. As an initialization step, we always expand A_{new} following the previous procedure up to a minimum length of L_{min} elements, even if the aforementioned condition is not met. The iterative procedure is repeated until E_{CW} points to the same element as E_{CCW}, this is, the oldest element in C. The incoming event is classified as a corner if the final length of the continuous arc A_{new} or its complementary arc in the circle $C \setminus A_{new}$ is between L_{min} and L_{max} elements.

To robustify our algorithm against noise, we employ the same strategy as eFAST, employing two different circle radii in our detection and only classifying the incoming event as a corner if the test is passed in both. For a fair comparison with eFAST, we employ circular masks computed from Bresenham's circles of radii 3 and 4 and arc length thresholds of $[L_{min}, L_{max}] = [3, 6]$ or $[4, 8]$ elements, respectively. These thresholds account for the angle range of the detectable corners, imposing with L_{max} a maximum angle of approximately $135°$ and L_{min} preventing noisy detections using too small arcs.

Algorithm 1: Arc* – Corner-Event Detection

Input: Event $= \{t, x, y, p\}$, $\mathcal{S}^*$, Radius, L_{min}, L_{max}

$C =$ CircleMask$(x, y, p, \mathcal{S}^*,$ Radius$)$
Initialize $A_{new} =$ NewestElement(C)
Initialize $E_{CW} =$ NextElementCW(A_{new}, C)
Initialize $E_{CCW} =$ NextElementCCW(A_{new}, C)
while $E_{CW} \neq E_{CCW}$ **do**
  if $E_{CW} > E_{CCW}$ **then**
    if OldestElement$(A_{new}) \leq E_{CW}$ **OR**
    Length$(A_{new}) < L_{min}$ **then**
      ExpandUntilElement(A_{new}, E_{CW})
    $E_{CW} =$ NextElementCW(E_{CW}, C)
  else
    if OldestElement$(A_{new}) \leq E_{CCW}$ **OR**
    Length$(A_{new}) < L_{min}$ **then**
      ExpandUntilElement(A_{new}, E_{CCW})
    $E_{CCW} =$ NextElementCCW(E_{CCW}, C)
if $L_{min} \leq$ Length$(A_{new}) \leq L_{max}$ **OR**
$L_{min} \leq$ Length$(C \setminus A_{new}) \leq L_{max}$ **then**
  return $true$
else return $false$

D. Event-based Corner Tracking

Assuming that the corner-events are detected accurately, they describe continuous trajectories in the image plane due to the asynchronous nature of the detections. Exploiting this continuity of tracks, the proposed tracker establishes data association of corners in the event-stream by relying on proximity of detections. Note that the described tracker is agnostic to the corner detector used, only requiring continuous detections in the image plane.

Our event-based tracker employs a directed graph $\mathcal{G} = \{V, E\}$ to represent corner-event tracks, where the vertices $\mathbf{v} \in V$ represent the detected corner-events and the edges $\mathbf{e} \in E$ represent the associations between such detections. Each vertex encodes the same information as the corresponding corner-event, excluding its polarity, i.e. $\mathbf{v} = \{t, x, y\}$. The tracker discriminates between active and non-active vertices, indicating whether they are viable candidates for data association or not, respectively. The graph is structured in smaller tree subgraphs $T \subset \mathcal{G}$, where each tree T represents a set of different hypotheses of trajectories for the same single tracked corner defined in spatio-temporal space. The tracker, described in Algorithm 2, addresses the asynchronous growth of the trees as new corner-events are included in the graph (as illustrated in Fig. 5).

A new corner-event generates an equivalent active vertex $\mathbf{v}_{new}$, which the tracker attempts to associate with any of the newest active vertices located in its neighbourhood, namely the subset V_{neigh}. The neighbourhood expands in the image plane up to a maximum range of d_{conn} pixels centered around $\mathbf{v}_{new}$. Ideally, assuming continuous feature tracks, we could set $d_{conn} = 1$px. However, in order to cope with missed detections in real scenarios, we set $d_{conn} = 5$px in this work. The neighbourhood is also temporally constrained, so

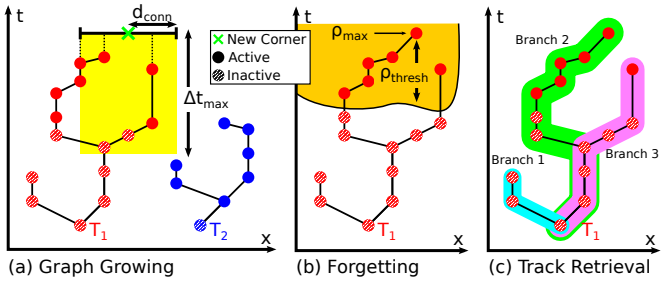

Fig. 5. In (a), we depict a simplified 1D scenario with the trees T_1 (red) and T_2 (blue), consisting of active (full circle) and inactive vertices (striped circle). Upon the arrival of a new corner-event (green), the set of newest neighbouring active vertices V_{neigh} (vertices connected by dashed lines) is retrieved from the spatio-temporal window (yellow area), determined by the parameters d_{conn} and Δt_{max}. The closest and newest vertex from T_1 is selected for data association, growing the tree as depicted in (b). However, the inclusion of the new vertex triggers the forgetting process as it increases the maximum depth of the tree ρ_{max}, inducing the deactivation of previous vertices according to ρ_{thresh}. Lastly, in (c) the best track encoded in T_1 can be retrieved from the deepest branch, i.e. 'Branch 2' in this case (green path).

that vertices older than Δt_{max} with respect to $\mathbf{v}_{new}$ are not considered. Essentially, Δt_{max} defines a temporal window to prevent data association with older and noisy parts of the graph, and is tuned conservatively to $\Delta t_{max} = 0.1$s to cope with slow feature motions.

We split the set of neighbouring vertices V_{neigh} into leaf and non-leaf vertices, namely V_{leaf} and $V_{\neg leaf}$, respectively. The closest vertex to v_{new} in the image plane from V_{leaf} is defined as $\mathbf{v}_{parent}$ or, if there are several vertices at the same distance, the newest one among them. If there are no leaf nodes ($V_{leaf} = \emptyset$), we extract $\mathbf{v}_{parent}$ from $V_{\neg leaf}$ instead. We establish then data association between both nodes, assigning $\mathbf{v}_{new}$ to the tree to which $\mathbf{v}_{parent}$ belongs, T_{parent}, and growing it by adding a new edge between vertices. In case there are no active vertices in the neighbourhood, $V_{neigh} = \emptyset$, $\mathbf{v}_{new}$ becomes the root of a new tree T_{new} instead.

Each vertex stores its relative depth to the root of its tree. If the maximum depth ρ_{max} within a tree increases after adding a new vertex, we deactivate any vertex whose relative depth compared to ρ_{max} exceeds ρ_{thresh}. This adaptive deactivation process acts as an event-based driven forgetting horizon, keeping active only the vertices in the most promising branches as the tree grows. The parameter ρ_{thresh} can be finely tuned according to the capabilities of the corner-detector to perform non-maximum suppression or the proximity of the corners in the scene. In our experiments, however, we set $\rho_{thresh} = 5$.

At any given instant, we can extract the best hypothesis for the corner trajectory from the deepest branch in the tree T, representing the branch with the highest spatio-temporal consensus. Nonetheless, note that, when the best trajectory hypothesis is computed at run time, the newest vertices that are still active and within the forgetting horizon determined by ρ_{thresh} are not reliable to determine the corner position. In our experiments, we grow the graph and establish data association in real-time and then post-process each tree to retrieve the best track, ignoring those that do not last longer than 0.5s.

Algorithm 2: Corner-Event Tracking

Input: Detection $= \{t, x, y, p\}$, d_{conn}, Δt_{max}, ρ_{thresh}
Initialize Vertex: $\mathbf{v}_{new} = \{t, x, y\}$;
$V_{neigh} =$ GetActiveNeighbourVertices$(\mathbf{v}_{new}, d_{conn})$;
$V_{neigh} =$ DiscardOldVertices$(V_{neigh}, \Delta t_{max})$;
if $V_{neigh} \neq \emptyset$ **then**
  $\{V_{leaf}, V_{\neg leaf}\} =$ SegmentLeafVertices(V_{neigh}) ;
  if $V_{leaf} \neq \emptyset$ **then**
    $\mathbf{v}_{parent} =$ GetClosestAndNewestVertex(V_{leaf});
  else
    $\mathbf{v}_{parent} =$ GetClosestAndNewestVertex$(V_{\neg leaf})$;
  $T_{parent} =$ GrowExistingTree$(\mathbf{v}_{parent}, \mathbf{v}_{new})$;
  UpdateActiveVerticesInTree$(T_{parent}, \rho_{thresh})$;
else
  $T_{new} =$ InitializeNewTreeFromVertex$(\mathbf{v})$;

IV. Experiments

We evaluated the proposed pipeline on the publicly available event-camera dataset of [1]. This dataset was recorded with a Dynamic and Active-pixel Vision Sensor (DAVIS) [3], which has a resolution of 240×180 pixels and captures both event-based cues and intensity (frame-based) images using the same chip. While the proposed approach works only on the event-stream, we use the intensity frames to extract ground-truth for some of the proposed metrics. This dataset is composed of different scenes recorded using different camera motions. For our evaluation, we select a representative subset of these scenes with increasing complexity and event-rate – namely, we test on the `shapes`, `dynamic`, `poster` and `boxes` scenes. Note that we use the same subset of experiments employed in the evaluation of eFAST against eHarris [17], in order to enable a fair comparison against all methods. Unless stated otherwise, due to space limitations, we group the performance according to the scene, combining all the recorded experiments with different motions, and report the mean for each of the proposed metrics. In our metrics, we compare the performance of our Arc* corner detector to the state-of-the-art eHarris and eFAST (using the original implementations provided by the authors[1]), and their modified versions eHarris* and eFAST* employing the proposed filter of redundant events. The proposed corner tracker algorithm is evaluated using our corner detector.

A. Ground-truth for Event Cameras

Obtaining ground-truth to evaluated purely event-based approaches is particularly challenging as simulated data does not guarantee the capture of the true sensor's behaviour. In the event-based detector literature, [13] and [15] manually labelled groups of corners in the event-stream and thus, this approach is limited in practice to simple scenes with clear corners. In [17], an automatic labelling method is proposed which evaluates the spatio-temporal distribution of events classified as corners. Such method, however, cannot account for missed detections

[1]https://github.com/uzh-rpg/rpg_corner_events

TABLE I
THE PERCENTAGE OF THE EVENTS BLOCKED BY THE PROPOSED FILTER AND THE PERCENTAGE OF CORNER-EVENTS DETECTED PER SCENE.

Experiment	Block [%]	Corner events [%]				
		eHarris	eFAST	eHarris*	eFAST*	Arc*
shapes	52.6	8.2	12.5	5.1	7.8	12.1
dynamic	48.0	4.8	3.6	3.3	2.5	7.1
poster	45.6	7.5	4.3	5.2	3.2	8.3
boxes	43.2	7.6	3.3	5.2	2.6	6.4

and does not guarantee that the detections correspond to actual corners in the scene.

In our evaluation, we make use of the available intensity frames in the dataset to detect corners using the (original) Harris detector [16] and track them using KLT [12], similarly to [18]. We exhaustively refine the tracks and interpolate their position in the image plane using cubic splines to match the temporal resolution of the events, while discarding noisy and short tracks.

In our accuracy-related metrics, we only consider the events that are triggered within a neighbourhood of up to 5 pixels (in the image space) of any intensity-based track. In the algorithms employing the proposed filter, the considered events are further restricted to those not filtered in the event-stream. Restricting our metrics to this subset of events close to intensity-based tracks, we ensure those events have actually correspondence to real corners in the image space, where ground-truth is available. While the DVS allows us to detect more corners in the scene (e.g. with low intensity contrast or motion blur), the proposed evaluation strategy is reproducible and reliable to enable ground-truth comparisons. As the selected experiments are recorded with increasing velocity, we limit the length of each experiment to the first 10 seconds, leaving out the images where significant motion blur affects the fidelity of the ground-truth tracks.

Metrics that are not related to intensity-based tracks are not subject to the aforementioned restrictions, and are thus evaluated by processing the whole event-stream for each experiment, offering evidence and insights of the applicability of the proposed pipeline even under high-dynamic motions.

B. Filtering and Reduction to Corner Events

Reducing the event-stream to the most relevant events decreases the amount of data that needs to be processed by modules further down in the pipeline. This reduction can be achieved by filtering the event-stream, as done with the proposed filter eliminating redundant events, and/or detecting corners in the stream from which a high-level representation of the scene is obtained.

Table I summarizes the reduction of the data stream as a percentage of the total number of events per scene. The proposed filter blocks between 40-50% of the events. Consequentially, the modified versions eHarris* and eFAST* employing our filter have an equivalent drop in the number of detected corners. Despite the reduced number of corners detected, the results in Section IV-C provide evidence that the quality of the detection is not significantly affected as only redundant events are filtered out whereas the computational performance experiences a substantial improvement. When limited to the filtered event-stream, the proposed corner detector detects most corners, approaching comparable figures to the number of corners detected in the unfiltered event-stream with eHarris and eFAST.

C. Corner Detector Performance

By design, eHarris has a weak response to corners with wide angles (inherited from the original Harris detector), whereas eFAST misses the detection of some of the corners depending on their relative motion with respect to the camera (as explained in Section III-C). Our method Arc* is able to extend the detection of corner-events to those cases, as depicted in Fig. 6, where we also compare the percentage of the total length of intensity-based track detected using each algorithm. A segment of the intensity-tracks is considered detected if there exists at least one corner-event detected in the past 0.1s at maximum 3.5 pixels away from the track.

To assess the quality of the detections, we report the True Positive Rate (TPR) and False Positive Rate (FPR) per detector and scene in Fig. 7, including the 'Overall' as the mean across all the scenes weighted accordingly to the number of events. To this end, a corner-event is labelled as true positive if it is closer than 3.5 pixels away from an intensity-based track or true negative otherwise, up to a maximum of 5 pixels distance. The proposed Arc* outperforms all others in terms of TPR, i.e. we detect more real corners from the scene. However, as we drastically increase the number of detections, we also incur in an equivalent increment of FPR. Note that the performance of Arc*, in terms of both TPR and FPR, approaches the performance of eHarris as the scene's complexity increases. The algorithms eHarris* and eFAST* employing our filter shows no significant difference in these metrics, although the boost in computational performance is significant as reported Section IV-E.

The reported results suggest that eHarris is the most robust algorithm for complex scenes in presence of significant noise (poster and boxes), only performing marginally worse in simpler scenes (poster and dynamic) and keeping in all the cases a stable FPR and TPR score. In contrast, eFAST is excessively restrictive in its detections, incurring in low FPR but also low TPR, and, as consequence, it can only retrieve a small number of all the corner tracks in the scene (See Fig. 6). The proposed method Arc* drastically increases the number of detections, improving the TPR but scoring worse FPR, this is, detecting more real corners of the scene but also incurring in more noisy detections. The proposed method extracts more information from the scene to the tracker, allowing the system to keep track of a higher number of features while relying on the common consensus among the detections to remove the noisy ones. Moreover, Arc* is significantly more efficient than the eFAST and eHarris (i.e. 50× and 4.5× faster, respectively), being the only method able to process all the event-stream of the tested scenes in real time.

In terms of accuracy, measured as the mean distance of a corner-event to its closest intensity-based track, no significant

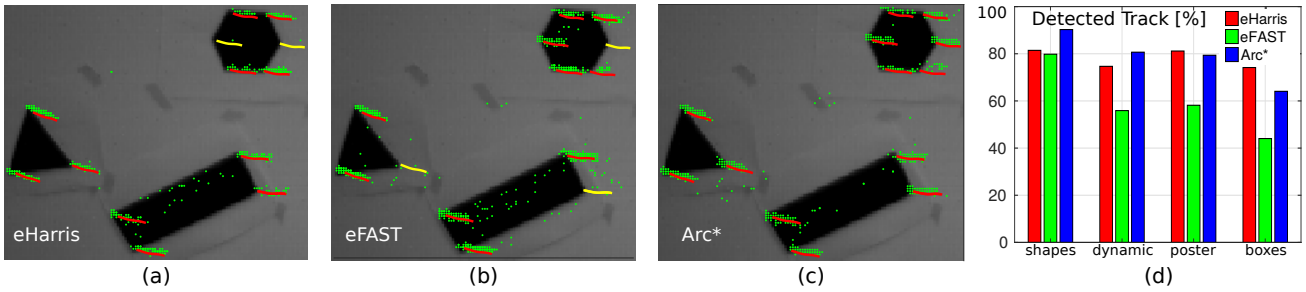

Fig. 6. Figures (a)-(c) depict the corner-events detected (green dots) in `shapes` scene using eHarris, eFAST and our proposed Arc* corner detector whereas the lines correspond to the intensity-corner tracks within a temporal window of 100ms. Coloured in red, the tracks that are strongly detected by each detector in contrast to those in yellow. Our detector Arc* is able to successfully detect all the corners, whereas eHarris have a low response in corners with wide angle and eFAST miss those whose local SAE have a specific shape as explained in Fig. 3. Note the noise reduction in the detections when employing the proposed filter. Figure (d) shows the percentage of the track length detected in the event-stream with respect to the track length computed from intensity images for each method and scene.

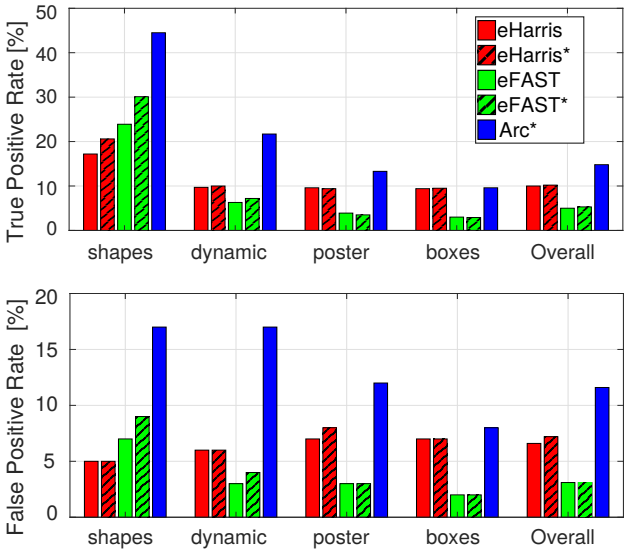

Fig. 7. The True Positive Rate and False Positive Rate of corner-event detections for each scene and detector.

TABLE II
THE PERFORMANCE OF THE PROPOSED CORNER-EVENT TRACKER IN TERMS OF THE MEAN ACCURACY, TRACK LIFETIME AND UNIQUENESS IN TRACKING THE SAME REAL CORNER PER SCENE.

Experiment	Accuracy [pixels]	Lifetime [s]	Uniqueness [%]
`shapes`	2.2	0.9	71
`dynamic`	2.6	0.5	23
`poster`	2.8	0.5	17
`boxes`	2.9	0.4	18

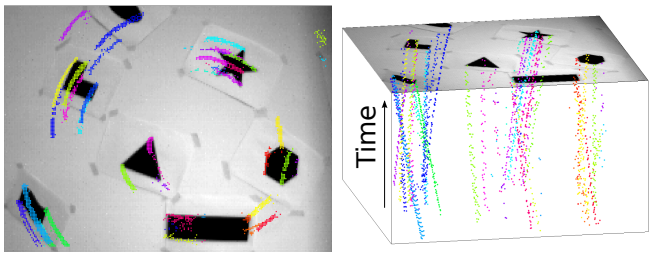

Fig. 8. Event-based tracks associating corner-events (different colors) within a time-window in the image space (left) and with respect to time (right) evaluated on `shapes` during high-speed camera rotation. The intensity image is superimposed for illustration.

is revealed across the methods, only depending on the scene complexity: namely, 2.3 ± 0.1px in `shapes`, 2.8 ± 0.2px in `dynamic`, 3.0 ± 0.1px in `poster` and 3.0 ± 0.1px in `boxes`. Such values match the expected range of 2-3px reported in previous publications such as [15].

D. Corner Tracker Performance

Illustrative examples of the proposed tracking system in action are depicted in Fig. 1 and Fig. 8 employing the corners detected with Arc*. The tracker's "Accuracy" is evaluated using the mean minimum distance between event-based and intensity-based tracks, considering only the tracks' segments closer than 5 pixels to each other, equivalently to the accuracy evaluation in corner detection. The results in Table II report an improved accuracy score compared to the corner detection and evidence that the tracker prune some of the noisy detections while composing the feature track from the most reliable ones.

Our method associates the corner-events based on proximity under the assumption that the corner detections are well localized at a real corner. When this assumption is compromised (as in highly textured environments), a single feature track associated to an specific real corner may jump to another corner during its lifetime. As our method does not explicitly report such tracking failure, for the sake of fairness, we report the mean "Lifetime" as the time a single feature track spent tracking the same intensity-based corner. In case more than a single intensity-based corner characterizes an event-based track, we report the percentage of the longest time spent tracking the same intensity-based corner as the score "Uniqueness", indicating the additional confusion incurred in complex scenes. Note that the quality of the proposed tracker is comparable to other state-of-the-art event-based corner tracker (e.g., [13]), while achieving significantly longer valid track lifetime in far more challenging scenarios.

E. Computational Performance

All experiments are run on a Xeon E3-1505M CPU with 2.80GHz and 16GB of RAM and using a single threaded C++

TABLE III
COMPUTATIONAL PERFORMANCE OF THE DIFFERENT CORNER DETECTOR METHODS AND THE PROPOSED TRACKING SYSTEM.

Algorithm	Time per event [μs/ev]	Max. event rate [Mev/s]
eHarris [15], [17]	6.96	0.14
eFAST [17]	0.60	1.67
eHarris*	4.56	0.22
eFAST*	0.35	2.85
Arc*	0.13	7.52
Corner Tracker	1.97	0.51

implementation. Table III reports the average time spent to process a single event and the equivalent maximum event rate in Millions of events per second for each algorithm. On average, the proposed Arc* corner detector runs more than a than 50$\times$ faster than eHarris and up to 4.5$\times$ faster than the eFAST using the implementation provided by the authors. While eFAST can achieve, on average, real-time performance in the low-textured scenes (`shapes` and `dynamic`), it cannot cope with the average event rate of over 2 Mev/s of the complex scenes (`poster` and `boxes`). Despite the increased robustness of eHarris compared to Arc* and eFAST, it performs always slower than real-time, e.g. 2$\times$ slower than-real time in the simplest `shapes` scene.

Note the significant computational performance boost due to the proposed filter in eHarris* and eFAST*. The cost to process a single event is improved by approximately 35%, while the performance of the algorithms is not significantly affected as reported in Section IV-C.

Finally, Table III also reports the processing time required to establish data association between corner-events with the proposed tracker. While the time per event is significantly larger than in detection, note that the tracker only processes corner-events, which, in the worst case scenario, account up to 12% (see Table I) of the the original event-stream and thus applicable in real-time even in high-speed applications.

V. CONCLUSIONS

This paper presents a novel event-based visual front-end pipeline, that pre-filters the event-stream, detects corners in it and tracks them in real-time in a completely asynchronous fashion and thus, can fully leverage the benefits of event cameras. The proposed event-based filter is shown to reduce the amount of redundant information, effectively boosting the performance of even existing state-of-the-art corner-event detectors. Evaluation on benchmarking datasets reveals a significant speed-up offered by the proposed Arc* corner-event detector, with real-time performance even under high-speed motions, and with improved repeatability and number of detections. Lastly, the proposed novel asynchronous corner tracker establishes correspondences across corner-events also in real-time and can be used to retrieve spatio-temporal event-based feature tracks. We believe that the proposed pipeline opens up exciting new research directions for high-speed camera and scene tracking towards the development of a completely asynchronous and event-based SLAM framework in the future.

REFERENCES

[1] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and slam," *International Journal of Robotics Research (IJRR)*, vol. 36, no. 2, pp. 142–149, 2017. [Online]. Available: https://doi.org/10.1177/0278364917691115

[2] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128$\times$128 120 db 15 μs latency asynchronous temporal contrast vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, Feb 2008.

[3] C. Brandli, R. Berner, M. Yang, S. C. Liu, and T. Delbruck, "A 240$\times$180 130db 3 μs latency global shutter spatiotemporal vision sensor," *IEEE Journal of Solid-State Circuits*, vol. 49, no. 10, pp. 2333–2341, Oct 2014.

[4] C. Brandli, J. Strubel, S. Keller, D. Scaramuzza, and T. Delbruck, "ELiSeD – an event-based line segment detector," in *International Conference on Event-Based Control, Communication and Signal Processing (EBCCSP)*, June 2016, pp. 1–7.

[5] A. Glover and C. Bartolozzi, "Event-driven ball detection and gaze fixation in clutter," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 2203–2208.

[6] D. R. 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*, vol. 26, no. 12, pp. 3045–3059, Dec 2015.

[7] H. Kim, S. Leutenegger, and A. J. Davison, "Real-time 3d reconstruction and 6-dof tracking with an event camera," in *Proceedings of the European Conference on Computer Vision (ECCV)*. Springer, 2016, pp. 349–364.

[8] 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*, vol. 2, no. 2, pp. 593–600, 2017.

[9] A. Zhu, N. Atanasov, and K. Daniilidis, "Event-based visual inertial odometry," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.

[10] H. Rebecq, T. Horstschaefer, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[11] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 430–443, 2006.

[12] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, ser. IJCAI'81. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1981, pp. 674–679. [Online]. Available: http://dl.acm.org/citation.cfm?id=1623264.1623280

[13] X. Clady, S.-H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Networks*, vol. 66, no. Supplement C, pp. 91 – 106, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608015000477

[14] R. Benosman, C. Clercq, X. Lagorce, S. H. Ieng, and C. Bartolozzi, "Event-based visual flow," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 2, pp. 407–417, Feb 2014.

[15] V. Vasco, A. Glover, and C. Bartolozzi, "Fast event-based harris corner detection exploiting the advantages of event-driven cameras," in *Proceedings of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, Oct 2016, pp. 4144–4149.

[16] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the Alvey Vision Conference*. Alvety Vision Club, 1988, pp. 23.1–23.6, doi:10.5244/C.2.23.

[17] E. Mueggler, C. Bartolozzi, and D. Scaramuzza, "Fast event-based corner detection," in *Proceedings of the British Machine Vision Conference (BMVC)*, 2017.

[18] A. Z. Zhu, N. Atanasov, and K. Daniilidis, "Event-based feature tracking with probabilistic data association," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, pp. 4465–4470.