Feature Detection and Tracking with the Dynamic and Active-pixel Vision Sensor (DAVIS)

David Tedaldi, Guillermo Gallego, Elias Mueggler and Davide Scaramuzza

***Abstract*— Because standard cameras sample the scene at constant time intervals, they do not provide any information in the blind time between subsequent frames. However, for many high-speed robotic and vision applications, it is crucial to provide high-frequency measurement updates also during this blind time. This can be achieved using a novel vision sensor, called DAVIS, which combines a standard camera and an asynchronous event-based sensor in the same pixel array. The DAVIS encodes the visual content between two subsequent frames by an asynchronous stream of events that convey pixel-level brightness changes at microsecond resolution. We present the first algorithm to detect and track visual features using both the frames and the event data provided by the DAVIS. Features are first detected in the grayscale frames and then tracked asynchronously in the blind time between frames using the stream of events. To best take into account the hybrid characteristics of the DAVIS, features are built based on large, spatial contrast variations (i.e., visual edges), which are the source of most of the events generated by the sensor. An event-based algorithm is further presented to track the features using an iterative, geometric registration approach. The performance of the proposed method is evaluated on real data acquired by the DAVIS.**

I. Introduction

Feature detection and tracking are the building blocks of many robotic and vision applications, such as tracking, structure from motion, place recognition, etc. Extensive research has been devoted to feature detection and tracking with conventional cameras, whose operation principle is to temporally sample the scene at constant time intervals. However, conventional cameras still suffer from several technological limitations that prevent their use in high speed robotic and vision applications, such as autonomous cars and drones: (i) low temporal discretization (i.e., they provide no information during the blind time between consecutive frames), (ii) high redundancy (i.e., they wastefully transfer large amounts of redundant information even when the visual content of the scene does not change), (iii) high latency (i.e., the time needed to capture and process the last frame). Since the agility of an autonomous agent is determined by the latency and temporal discretization of its sensing pipeline, all these advantages put a hard bound on the maximum achievable agility of a robotic platform.

Bio-inspired event-based sensors, such as the Dynamic Vision Sensor (DVS) [1], [2], [3] or the Asynchronous Time-based Image Sensor (ATIS) [4], [5], [6], overcome the above-mentioned limitations of conventional cameras. In an event-based sensor, each pixel operates independently of all other pixels, and transmits asynchronously pixel-level brightness changes, called "events", at microsecond resolution at the time they occur. Hence, an event camera virtually eliminates latency and temporal discretization. Also, it avoids redundancy, as no information is transmitted if the scene does not change. However, this comes at a price: the output of an event camera (a stream of events) is fundamentally different from that of conventional cameras; hence, mature computer vision algorithms cannot be simply adapted, and new, event-driven algorithms must be developed to exploit the full potential of this novel sensor.

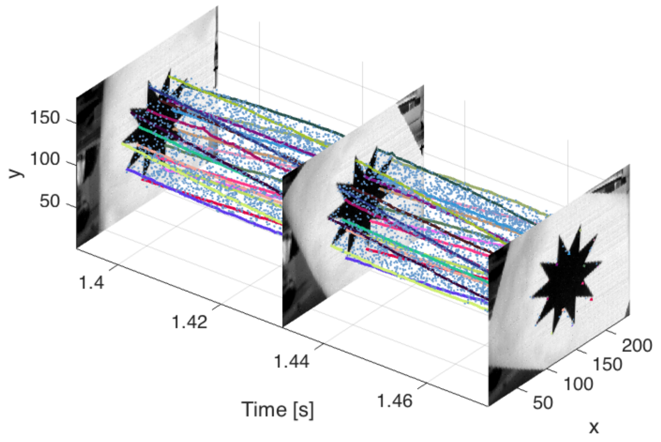


Fig. 1: Spatio-temporal view of the output of the DAVIS (frames and events) and the trajectories of the tracked features (in different colors, one for each feature). In this example, the scene consists of a rotating object. The motion in the blind time between consecutive frames is accurately tracked using the stream of events; e.g., rotation is clearly visible in the spiral-like trajectories of the event-based tracked features. To facilitate the visualization, only 10% of the events is displayed.

More recently, hybrid vision sensors that combine the benefits of conventional and event-based cameras have been developed, such as the Dynamic and Active-pixel VIsion Sensor (DAVIS) [7]. The DAVIS implements a standard grayscale camera and an event-based sensor in the same pixel array. Hence, the output consists of a stream of asynchronous high-rate (up to 1 MHz) events together with a stream of synchronous grayscale frames acquired at a low rate (on demand and up to 24 Hz).

We present the first algorithm to detect features from the DAVIS frames and perform event-driven high-temporal

The authors are with the Robotics and Perception Group, Department of Informatics, University of Zurich, Switzerland—http://rpg.ifi.uzh.ch. This research was supported by the DARPA FLA Program and the National Centre of Competence in Research Robotics (NCCR).

resolution tracking of these features in the blind time between two frames. The key challenge consists of designing an algorithm that best takes into account the hybrid characteristics of the DAVIS output to solve the detection-and-tracking problem (Fig. 1). Since events are generated by changes of brightness in the scenes, features are built based on large, spatial contrast variations (i.e., visual edges), which are the source of most of the events generated by the sensor. An event-based algorithm is further presented to track the features using an iterative, geometric registration approach.

The paper is organized as follows. Section II reviews the related work on event-based feature detection and tracking. Section III describes the DAVIS sensor. Section IV describes the proposed detection and tracking algorithm. Section V presents the experimental results. Finally, section VI draws the conclusion and gives future perspectives.

II. Related Work

A. From Frame-based to Event-based Tracking

Feature detection and tracking methods for frame-based cameras are well-known [8], [9], [10]. The pixel intensities around a corner point are used as a template that is compared frame-by-frame with the pixels around the estimated position of the corner point. The photometric error is then used to update the parameters describing the position and warping of the template in the current frame. These appearance-based methods do not apply to event cameras; however, the approach of using a parametric template model and updating its parameters according to data fitting still applies.

From a high-level point of view, two relevant questions regarding event-based tracking are *what to track* and *how to track*. The first question refers to how are the objects of interest modeled in terms of events so that object instances can be detected in the event stream. The answer to this question is application dependent; the object of interest is usually represented by a succinct parametric model in terms of shape primitives. The second question, "how to track?", then refers to how to update the parameters of the model upon the arrival of data events (caused by relative motion or by noise). For a system that answers the aforementioned questions, a third relevant question is "*what kind of object motions or distortions can be tracked?*" The above-mentioned three questions are key to understand existing tracking approaches.

B. Event-based Tracking Literature

Early event-based feature trackers were very simple and focused on demonstrating the low-latency and low-processing requirements of event-driven vision systems, hence they tracked moving objects as clustered blob-like sources of events [11], [12], [13], [14], [15] or lines [16].

Accurate tracking of general shapes can be performed by continuously estimating the warping between the model and the events. This has been addressed and demonstrated for arbitrary user-defined shapes using event-based adaptions of the Iterative Closest Point (ICP) algorithm [17], gradient descent [18], or Monte-Carlo methods [19] (i.e., by matching events against a uniformly-sampled collection of rotated and scaled versions of the template). Detection and tracking of locally-invariant features, such as corners, directly from event streams has been addressed instead in [20].

Notice, however, that all above-mentioned papers were developed for event-only vision sensors. In this paper, we build upon these previous works and present the first algorithm to automatically detect features from the DAVIS frames and perform event-driven high-temporal resolution tracking of these features in the blind time between two frames.

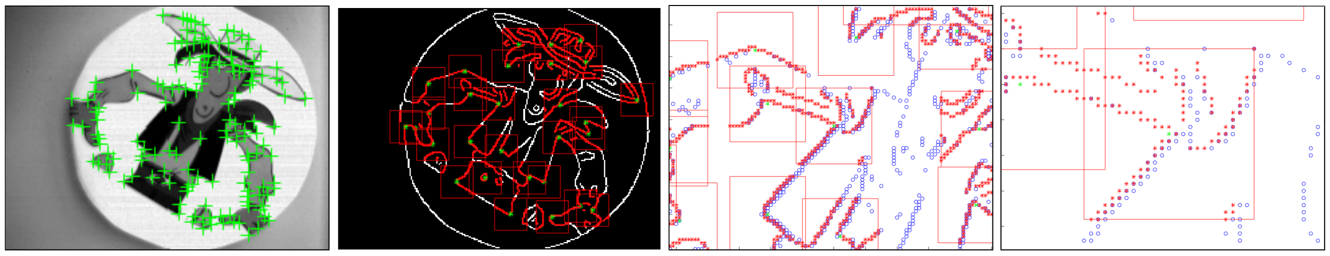
III. The Dynamic and Active-pixel VIsion Sensor

The DAVIS [7] is a novel vision sensor combining a conventional frame-based camera (active pixel sensor - APS) and a DVS in the same array of pixels. The global-shutter frames provide absolute illumination on demand and up to 24 Hz, whereas the event sensor responds asynchronously to pixel-level brightness changes, independently for each pixel. More specifically, if $I(t)$ is the illumination sensed at pixel (x, y) of the DVS, an event is triggered if relative brightness change exceeds a global threshold: $|\Delta \ln I| := |\ln I(t) - \ln I(t - \Delta t)| > C$, where Δt is the time since the last event was triggered (at the same pixel). An event is a tuple $e = (x, y, t, p)$ that conveys the spatio-temporal coordinates (x, y, t) and sign (i.e., polarity $p = \pm 1$) of the brightness change. Events are time-stamped with microsecond resolution and transmitted asynchronously when they occur, with very low latency 15 µs. The DAVIS has a very high dynamic range (130 dB) compared with the 70 dB of high-quality, traditional image sensors. The low latency, the high temporal resolution, and the very high dynamic range make the DAVIS extremely advantageous for future robotic applications in uncontrolled natural lighting, i.e., real-world scenarios.

A sample output of the DAVIS is shown in Fig. 1. The spatial resolution of the DAVIS is 240×180 pixels. This is still limited compared to the spatial resolution of state-of-the-art conventional cameras. Newer sensors, such as the color DAVIS (C-DAVIS) [21] will have higher spatial resolution (640×480 pixels), thus overcoming current limitations.

IV. Feature Detection and Tracking with the DAVIS

Since events are generated by changes of brightness, this implies that only edges are informative. Intersecting edges create corners, which are "features" that do not suffer from the aperture problem and that have been proven to be optimally trackable in frame-based approaches [10]. Therefore, event-based cameras also allow for the perception of corners, as shown in [20]. We exploit these observations to extract and describe features using the DAVIS frames, and then track them using the event stream, as illustrated in Fig. 2. Our method builds upon the customized shapes in [19] and the update scheme in [17]. The technique comprises to main steps: feature detection and tracking, as we detail in the next sections.



(a) (b) (c) (d)

Fig. 2: Feature detection and tracking. (a) Frame with centers of detected features (green crosses). (b) Edge map (black and white) and square patches defining the features (i.e, model point sets, in red) (b)-(c) Zoomed views of the data point sets (i.e., events; blue circles) and model point sets (red stars) of the features, shortly after initialization.

Algorithm 1 High temporal resolution tracking

```
Feature detection:
- Detect corner points on the frame (Harris detector).
- Run Canny edge detector (returns a binary image, 1 if edge pixel; 0 otherwise).
- Extract local edge-map patches around corner points, and convert them into model point sets.
Feature tracking:
- Initialize a data point set per patch
for each incoming event do
    - Update the corresponding data point set.
    for each corresponding data point set do
        - Estimate the registration parameters between the data and the model point sets.
        - Update registration parameters of the model points.
```

A. *Feature Detection From Frames*

The absolute brightness frames of the DAVIS are used to detect edges (e.g., Canny's method [22]) and corners (e.g., Harris detector [23]). Around the strongest corners, we use the neighboring pixels of the Canny edge-map to define patches containing the dominant source of events. We simplify the detection by converting the edge-map patches to binary masks indicating the presence (1) or absence (0) of an edge. The binary masks define the interest shapes for tracking in terms of 2D point sets, called "*model* point sets". These steps are summarized at the beginning of Algorithm 1.

We use square patches of the same size, which is an adjustable parameter. However, it is straightforward to extend the method to consider different aspect ratios and sizes.

Frames are not required to be provided at a constant rate since they are only used to initialize features; they can be acquired on demand to replace features that are lost or fall out of the field of view of the sensor.

B. *Feature Tracking From the Event Stream*

Extracted features are tracked using subsequent events from the DAVIS. The input to the event-based tracking algorithm consists of multiple, local model point sets. The second part of Algorithm 1 summarizes our tracking strategy.

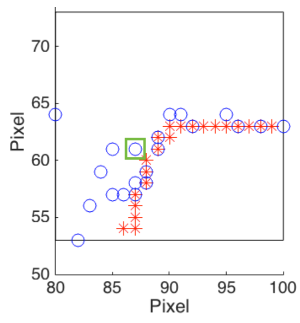
1) Sets of Events used for Feature Registration: For every feature, we define a *data* point set of the same size as the *model* point set. Therefore, the size can be different for every feature, depending on edge information. Data point sets consist of local space-time subsets of the incoming events: an event is inserted in a data point set if the event coordinates are inside the corresponding patch. Once a data point set has been filled, registration of the point sets can be done. Hence, a data point set defines the set of events that are relevant for the registration of the associated feature. Registration is carried out by minimization of the distance between the data and the model point sets, as explained next. Data point sets are continuously updated: the newest event replaces the oldest one and the registration iteration proceeds.

This procedure is event-based, i.e., the parameters of the tracked feature are updated every time an incoming event is considered relevant for that feature. The algorithm is asynchronous by design, and can process multiple features simultaneously. Several strategies to assign an incoming event to one or more overlapping patches can be used, in a way similar to [18]. We updated all models around the ambiguous event.

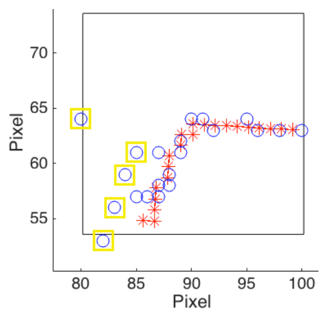
2) Registration: The data point set from the events, $\{\mathbf{p}_i\}$, is registered to the model point set (feature), $\{\mathbf{m}_i\}$, by minimization of the Euclidean distance between the sets, and including outlier rejection:

$$\arg\min_{\mathtt{A}} \sum_{(\mathbf{p}_i,\mathbf{m}_i)\in\text{Matches}} \|\mathtt{A}(\mathbf{p}_i) - \mathbf{m}_i\|^2, \qquad (1)$$

where $\mathtt{A}$ is the registration transformation between the matched point sets. For simplicity, we choose $\mathtt{A}$ within the class of Euclidean motions, but the method can be extended to more complex transformations. We choose the iterative closest point algorithm (ICP) [24] to minimize (1). Matches $\mathbf{p}_i \leftrightarrow \mathbf{m}_i$ are established according to nearest neighbor; a predefined distance of 2 pixels between the events in the data point set and the model point set is used for outlier rejection. Each algorithm iteration has three stages: first, candidate matches are established, then the geometric transformation is estimated, and, finally, the transformation is applied to the model point set. The operation proceeds until the error difference between two consecutive iterations is below a certain threshold.



(a) Before registration



(b) After registration

Fig. 3: A feature tracker, with the model point set (in red), the data point set (in blue). Same color notation as in Figs. 2c-2d. The black square represents the patch around the model point set. (a) Before registration: the current event (in green) updates the data point set and is used for registration of the point sets. (b) After registration: the events marked in yellow are classified as outliers, and the registration parameters are updated, aligning the model and data point sets.

Fig. 3a shows both the model and the data point sets. When a new event arrives, the geometric transformation that defines the tracker is updated according to the minimization of (1). The result is depicted in Fig. 3b. By discounting the points classified as outliers by the algorithm (in yellow), registration is accurate. Feature trajectories are given by the positions of the features returned by the registration step.

Due to the high temporal resolution of the DAVIS, the transformation between consecutive events (in the same feature) is close to the identity (Fig. 3b), and so, our method yields good results even after a single iteration. In practice, it is more efficient to compute the registration transformation every M events, e.g., of half the size of the model point set.

V. Experiments

We present the tests performed to validate the algorithm and to study its performance in different scenes with increasing level of complexity: a very large contrast (i.e., black and white) scene, a piecewise constant scene (a cartoon), and a natural scene (the leaves of a tree; see Fig. 11). The first scene depicts a black star on a white background; this scene has sharp transitions between the two intensity levels, showing clear edges (Fig 1) and well-localized features. The second scene consists of a cartoon image with piecewise constant regions (Fig 8a); intensity is concentrated in a few grayscale levels and there are moderately abrupt transitions between them. The third scene is a representative of a natural image, rich in texture and brightness changes of different magnitudes (Fig. 11a) coming from the leaves of a tree. The scene datasets show dominant translational and rotational motions.

We used patches of 25×25 pixels, which is approximately $1/10$ of the image width. This size was empirically found to be best for a broad class of scenes.

We measured the tracking error over time. The tracking error is computed against ground truth, which was generated using a frame-based Lucas-Kanade tracker [8] and linearly interpolating the feature motion in the time interval between frames. The ground truth has sub-pixel accuracy. Features were detected in the first frame ($t = 0$) and then tracked over the entire sequence using only events. In all scenes, the mean tracking error is less than 2 pixels. Notice that in spite of the sequences being short, they contain several million events.

A. Large-Contrast Scene (“Star”)

1) Translation: We moved a 10-point star sideways, back and forth, in front of the DAVIS. The algorithm detected one feature every two edges of the star (so there are 20 corners). The mean tracking error plot for all features is shown in Fig. 4. As it can be observed in the plot, there is a short pause after 1.5 s, marked with a constant error, before changing direction. In this interval, there are virtually no events, and so, the feature tracks do not move, waiting to observe new events in order to keep updating the features' position and orientation.

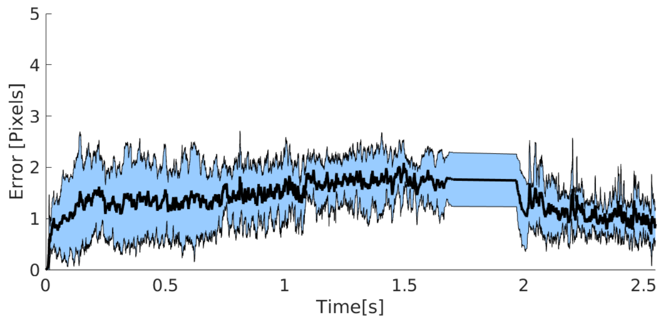


Fig. 4: *Star* (translation) dataset: feature tracking error of our event-based algorithm on translational motion facing the star shown in Fig. 1. The mean tracking error of all features is marked in black. The blue bands around the mean indicate the ± 1 standard-deviation confidence interval. The overall mean error is 1.52 pixels.

2) High-Speed Rotation: Next, we made the 10-point star pattern rotate, accelerating from rest to $1.600\,°/\mathrm{s}$ (see Fig. 5) using a electro-mechanical device. Observe that, while the overall motion of the star is a rotation approximately around the center of the image, features are much smaller than the whole star and so they only “see” parts of the peaks, consisting of at most two lines. Nevertheless, these very simple features are able to track the motion of the scene very well.

Because of the offset of the features from the rotation center of about 65 pixels, the features translate at high speeds (more than 1800 pixels/s on the image plane). For this dataset, ground truth was annotated since the frame-based solution failed: since the star is rotating by up to two points (peaks) between frames, aliasing prevented from obtaining the correct motion. This speed at which there is frame-based aliasing is not a problem for event-based tracking due to the microsecond temporal resolution of the pixels. The orientation error of the features is shown in Fig. 6. The mean orientation error remains below 20° over more than two full revolutions, leading to a relative error of 2.3 %. The feature tracks form spirals in image space-time, as shown in Fig. 7. All of the 20 features (one per vertex) of the 10-point star were accurately tracked during the entire sequence.

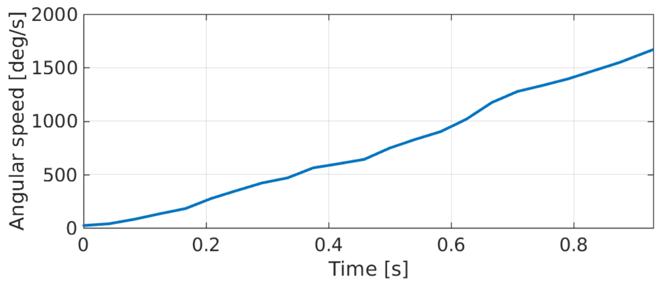


Fig. 5: *Star* (rotation) dataset: angular speed of rotating star. With an approximately constant acceleration (i.e., linear velocity profile), the angular speed reaches more than 1600 °/s.

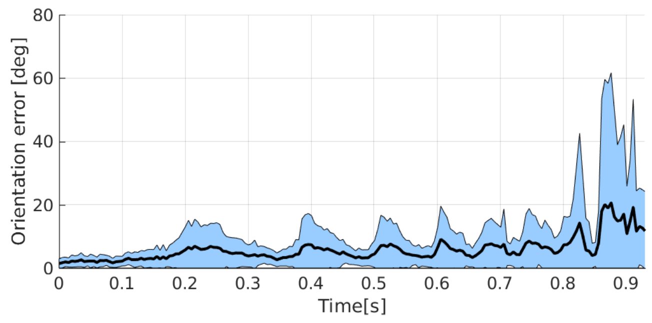


Fig. 6: *Star* (rotation) dataset: feature tracking error of our event-based algorithm on the dataset shown in Fig. 1. The mean tracking error of all features is marked in black. The blue bands around the mean indicate the ± 1 standard-deviation confidence interval. The overall mean error is 6.3°.

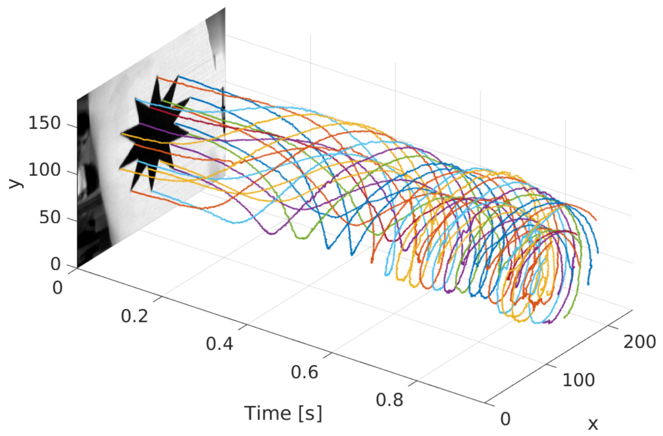


Fig. 7: *Star* (rotation) dataset: space-time locations of the features. Due to the rotation of the star, the feature tracks form spirals. The spiral step gets smaller as the angular speed increases, in this case, with constant acceleration.

B. Cartoon Scene (“Lucky Luke”)

Fig. 8 shows several snapshots of the tracked features on a sequence of the cartoon scene. The dominant motion is a horizontal translation, back and forth. We observe that 81 features well distributed in the object are correctly tracked throughout the event stream. The tracking error is reported in Fig. 9. As observed in the plot, there is a short pause after 1 s (constant error), before changing the motion direction. A slight increase of the error can be observed when the motion resumes. However, the mean error in this part of the motion is less than 2 pixel and the overall mean error is small: 1.22 pixel. Tracking in this scene is very good due to two reasons: (i) most of the events are located at the strong edges, which are captured by the features, and regions of constant intensity do not generate events. (ii) there are more than two edges per feature, and with a complex shape (edges in several directions) that make them distinctive for alignment. The tracked features in image space-time are shown in Fig. 10.

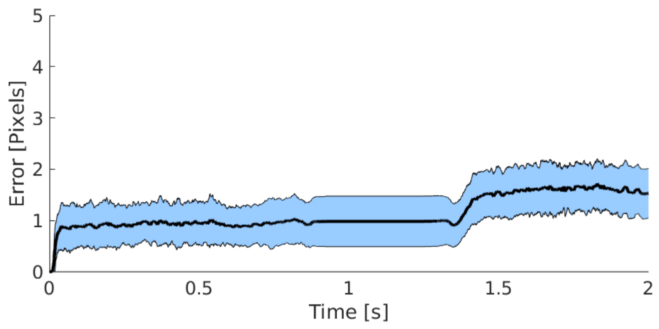


Fig. 9: *Lucky Luke* dataset: feature tracking error of our event-based algorithm. The mean tracking error of all features is marked in black. The blue bands around the mean indicate the ± 1 standard-deviation confidence interval. The overall mean error is 1.22 pixels.

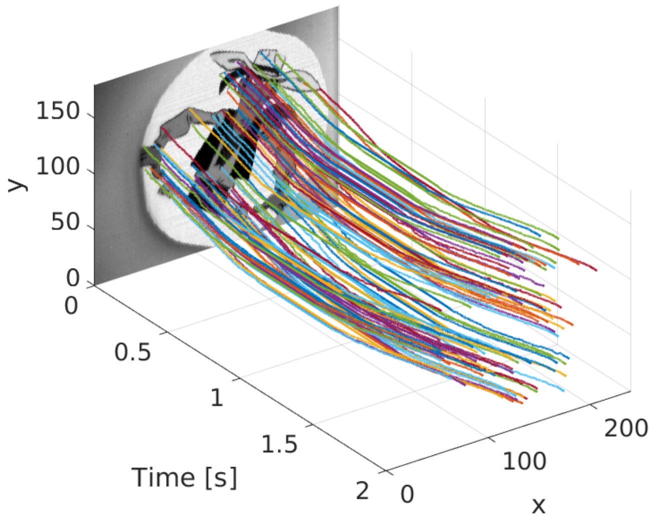
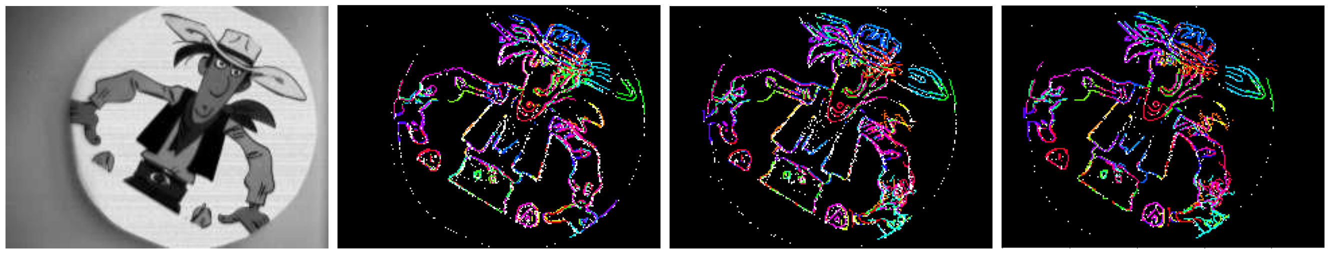


Fig. 10: *Lucky Luke* dataset: space-time view in the image plane of the tracked features' trajectories. The sideways motion is clearly visible in the feature trajectories.



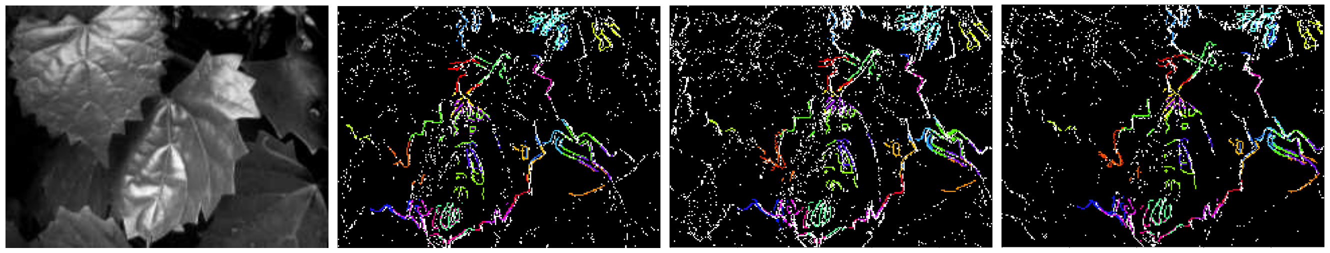
(a) DAVIS frame for initialization.

(b) Events (white over black) and features (solid colors) shortly after initialization.

(c) Features during motion.

(d) Features during motion, at a later time than (c).

Fig. 8: *Lucky Luke* (cartoon) dataset. The DAVIS is moving sideways while viewing a natural scene consisting of leaves (a). Individually tracked features (model point sets) are marked in different colors in (b) to (d).



(a) DAVIS frame for initialization.

(b) Events (white over black) and features (solid colors) shortly after initialization.

(c) Features during motion.

(d) Features during motion, at a later time than (c).

Fig. 11: *Leaves* dataset. The DAVIS is moving sideways while viewing a natural scene consisting of leaves (a). Individually tracked features (model point sets) are marked in different colors in (b) to (d).

C. Natural Scene (“Leaves”)

In natural scenes (Fig. 11a), edges can have all sort of different magnitudes, but our features still track the most dominant ones. In this experiment, we moved the DAVIS in front of a natural scene containing both edges (mostly at leave borders) and smooth intensity variations (within the leaves). The motion was oscillatory and predominantly translational (Fig. 11). Fig. 12 shows the feature position error; the mean error is 1.48 pixels.

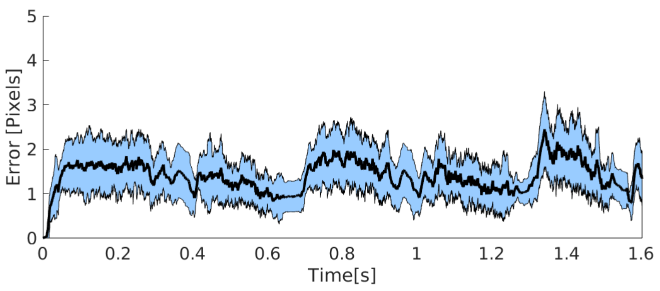


Fig. 12: *Leaves* dataset: feature tracking error of our event-based algorithm. The mean tracking error of all features is marked in black. The blue bands around the mean indicate the ± 1 standard-deviation confidence interval. The overall mean error is 1.48 pixels.

Feature tracks in image space-time are shown in Fig. 13. Fewer features are tracked compared to the simpler scenes

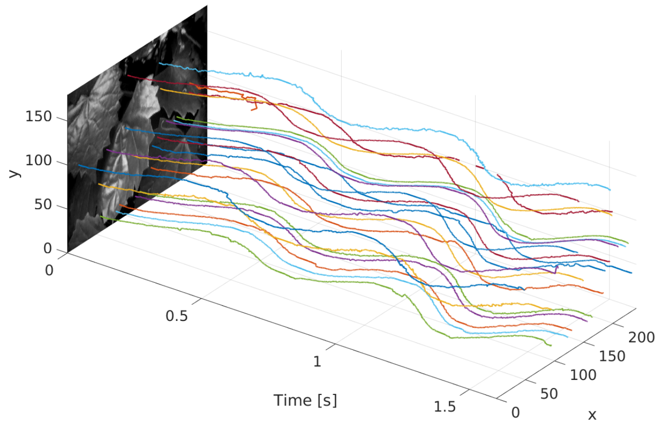


Fig. 13: *Leaves* dataset: space-time view in the image plane of the tracked features’ trajectories. The oscillating motion of the DAVIS is correctly captured by the feature trajectories.

(large contrast and cartoon) because of two reasons: (i) the detected features are based on a binary edge-map of the scene (resulted from the Canny detector), but such binary map is an exact representation of the underlying grayscale scene only if the contrast is sufficiently large. (ii) we do not model many of the non-linearities of the DAVIS, such as non-white noise and other dynamic properties, which have a larger effect on natural scenes than in simpler ones because events are triggered all over the patches. Notwithstanding, for some robotics applications there is no need to track many features; for example, in perspective-N-point problems) it is sufficient to track as few as three features [25].

Notice that, overall, all the experiments show that our proposed and automatically-detected features can be tracked for a considerable amount of time, much larger than the time between consecutive frames. Hence, lost features (e.g., falling out of the field of view) could be replaced by new ones that would be initialized using frames at a much lower rate (e.g. 1 Hz) or on demand.

Our method has been tested with real data, with different types of motion, and the results show accurate tracking (less than 2 pixels mean error). Better and more accurate results could be obtained by incorporating the edge strength and the event generation model.

VI. Conclusions

We have developed a high-temporal tracking algorithm for hybrid sensors such as the DAVIS. We used principled arguments of event data generation to justify our choice of relevant features to track, and proposed a pipeline to extract those features from the frames. Then we used an event-based tracking algorithm that exploits the asynchronous and high temporal resolution of the event stream. In our method, features are automatically and accurately initialized, and are adapted to the scene content, thus overcoming the shortcomings of existing methods. We tested the algorithm on real data from several sequences, and the results validate the approach.

Inspired by the achieved tracking accuracy, we intend to build a visual-odometry pipeline on top of this event-based feature tracking method. Finally, the frames used in our algorithm to initialize the features suffer from motion blur and limited dynamic range, as in any standard camera. To overcome these limitations, we plan to investigate methods to extract features directly from the event stream.

ACKNOWLEDGMENT

We thank Tobi Delbruck for providing the DAVIS240C and Beat Kueng for helping with data recording.

References

[1] P. Lichtsteiner, C. Posch, and T. Delbruck, "A 128x128 120dB 30mW asynchronous vision sensor that responds to relative intensity change," in *Solid-State Circuits Conference, 2006. ISSCC 2006. Digest of Technical Papers. IEEE International*, Feb 2006, pp. 2060–2069.

[2] ——, "A 128x128 120 dB 15 μs latency asynchronous temporal contrast vision sensor," *IEEE J. of Solid-State Circuits*, vol. 43, no. 2, pp. 566–576, 2008.

[3] T. Delbruck, B. Linares-Barranco, E. Culurciello, and C. Posch, "Activity-driven, event-based vision sensors," in *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, May 2010, pp. 2426–2429.

[4] C. Posch, D. Matolin, and R. Wohlgenannt, "An asynchronous time-based image sensor," in *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, May 2008, pp. 2130–2133.

[5] ——, "A QVGA 143dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel-level video compression," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2010 IEEE International*, Feb 2010, pp. 400–401.

[6] ——, "A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS," *IEEE J. of Solid-State Circuits*, vol. 46, no. 1, pp. 259–275, Jan 2011.

[7] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck, "A 240x180 130dB 3us Latency Global Shutter Spatiotemporal Vision Sensor," *IEEE J. of Solid-State Circuits*, 2014.

[8] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. 7th Int. Joint Conf. on Artificial Intelligence (IJCAI) - Volume 2*, 1981, pp. 674–679.

[9] C. Tomasi and T. Kanade, "Detection and tracking of point features," Carnegie Mellon University, Tech. Rep., 1991.

[10] J. Shi and C. Tomasi, "Good features to track," in *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, June 1994, pp. 593 –600.

[11] M. Litzenberger, C. Posch, D. Bauer, A. Belbachir, P. Schon, B. Kohn, and H. Garn, "Embedded vision system for real-time object tracking using an asynchronous transient vision sensor," in *Digital Signal Processing Workshop, 12th - Signal Processing Education Workshop, 4th*, Sept 2006, pp. 173–178.

[12] M. Litzenberger, A. Belbachir, N. Donath, G. Gritsch, H. Garn, B. Kohn, C. Posch, and S. Schraml, "Estimation of vehicle speed based on asynchronous data from a silicon retina optical sensor," in *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, Sept 2006, pp. 653–658.

[13] T. Delbruck and P. Lichtsteiner, "Fast sensory motor control based on event-based hybrid neuromorphic-procedural system," in *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, May 2007, pp. 845–848.

[14] T. Delbruck and M. Lang, "Robotic goalie with 3ms reaction time at 4% CPU load using event-based dynamic vision sensor," *Frontiers in Neuroscience*, vol. 7, no. 223, 2013.

[15] E. Piatkowska, A. Belbachir, S. Schraml, and M. Gelautz, "Spatiotemporal multiple persons tracking using Dynamic Vision Sensor," in *IEEE Comput. Soc. Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW)*, June 2012, pp. 35–40.

[16] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. Douglas, and T. Delbruck, "A Pencil Balancing Robot using a Pair of AER Dynamic Vision Sensors," in *IEEE Intl. Symp. on Circuits and Systems (ISCAS)*, 2009.

[17] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier, "Asynchronous Event-Based Visual Shape Tracking for Stable Haptic Feedback in Microrobotics," *IEEE Trans. Robotics*, vol. 28, pp. 1081–1089, 2012.

[18] Z. Ni, S.-H. Ieng, C. Posch, S. Regnier, and R. Benosman, "Visual Tracking Using Neuromorphic Asynchronous Event-Based Cameras," *Neural Computation*, vol. 27, pp. 925–953, 2015.

[19] X. Lagorce, C. Meyer, S.-H. Ieng, D. Filliat, and R. Benosman, "Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking," *IEEE Trans. Neural Networks and Learning Systems*, vol. 26, no. 8, pp. 1710–1720, Aug 2015.

[20] X. Clady, S.-H. Ieng, and R. Benosman, "Asynchronous event-based corner detection and matching," *Neural Networks*, vol. 66, pp. 91 – 106, 2015.

[21] C. Li, C. Brandli, R. Berner, H. Liu, M. Yang, S. Liu, and T. Delbruck, "An RGBW Color VGA Rolling and Global Shutter Dynamic and Active-Pixel Vision Sensor," in *International Image Sensor Workshop (IISW)*, Vaals, Netherlands, June 2015.

[22] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 8, no. 6, pp. 679–698, Nov 1986.

[23] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of The Fourth Alvey Vision Conference*, vol. 15. Manchester, UK, 1988, pp. 147–151.

[24] P. Besl and N. D. McKay, "A method for registration of 3-D shapes," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 14, no. 2, pp. 239–256, 1992.

[25] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, 2011, pp. 2969–2976.