Fast Event-based Corner Detection

Elias Mueggler[1]
mueggler@ifi.uzh.ch
Chiara Bartolozzi[2]
chiara.bartolozzi@iit.it
Davide Scaramuzza[1]
sdavide@ifi.uzh.ch

[1] Robotics and Perception Group
University of Zurich
Zurich, Switzerland

[2] iCub Facility
Istituto Italiano di Tecnologia
Genova, Italy

Abstract

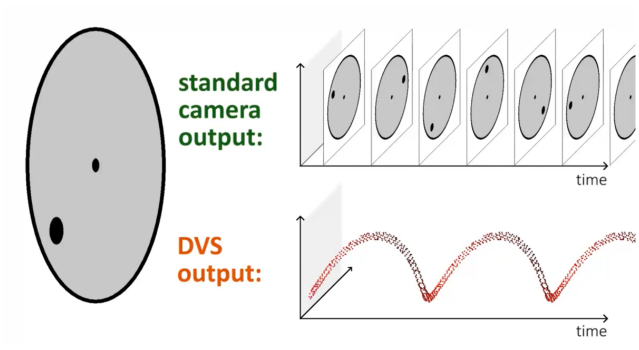
Event cameras offer many advantages over standard frame-based cameras, such as low latency, high temporal resolution, and a high dynamic range. They respond to pixel-level brightness changes and, therefore, provide a sparse output. However, in textured scenes with rapid motion, millions of events are generated per second. Therefore, state-of-the-art event-based algorithms either require massive parallel computation (e.g., a GPU) or depart from the event-based processing paradigm. Inspired by frame-based pre-processing techniques that reduce an image to a set of features, which are typically the input to higher-level algorithms, we propose a method to reduce an event stream to a *corner event* stream. Our goal is twofold: extract relevant tracking information (corners do not suffer from the aperture problem) and decrease the event rate for later processing stages. Our event-based corner detector is very efficient due to its design principle, which consists of working on the Surface of Active Events (a map with the timestamp of the latest event at each pixel) using only comparison operations. Our method asynchronously processes event by event with very low latency. Our implementation is capable of processing millions of events per second on a single core (less than a *micro*-second per event) and reduces the event rate by a factor of 10 to 20.

Multimedia Material

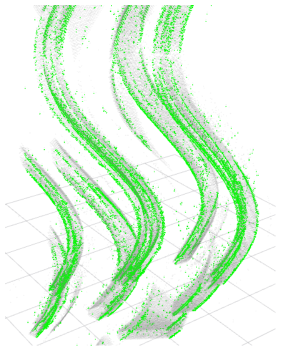
A supplemental video for this work is available on: `https://youtu.be/tgvM4ELesgI`.
The code is released open source at: `https://github.com/uzh-rpg/rpg_corner_events`.

1 Introduction

Event cameras offer great potential for virtual reality and robotics to overcome the challenges of latency, dynamic range, and high speed. Inspired by the human eye, these cameras respond to local, pixel-level brightness changes at the time they occur. These changes, called "events", are transmitted asynchronously and timestamped with *micro*-second precision. A comparison between standard frame-based and event cameras is shown in Fig. 1(a). Since each pixel is independent and can choose its own operating point, event cameras achieve a very high intra-scene dynamic range (more than 140 dB). However, due to their fundamentally different output (an event stream rather then a sequence of frames), standard computer-vision algorithms cannot be applied to such data directly and new methods to deal with this



(a) Standard vs Event Camera



(b) Raw Event vs Corner Event Stream

Figure 1: (a): Comparison of the output of a standard frame-based and an event camera when facing a rotating disk with a black dot. The standard camera outputs frames at a fixed rate, thus sending redundant information when no motion is present in the scene. Standard cameras also suffer from motion blur during rapid motion. Event cameras instead respond to pixel-level brightness changes with microsecond latency. Therefore, they do not suffer from motion blur and do not report anything when everything is at rest. An animated version of this figure can be found here: `https://youtu.be/LauQ6LWTkxM`. (b): The output of our method is a corner event stream (green), which is here overlaid on the raw event stream (gray) in space-time (time going upwards).

different way of encoding visual information (temporal contrast rather than absolute brightness) should be devised. As event cameras have become commercially available only in the last few years, *e.g*. the DVS [10] and the DAVIS [2], research on event-based vision is a relatively new topic. The most recent sensor, the DAVIS, has a resolution of 240×180 pixels and, in addition to the events, it also outputs standard grayscale images from the same physical pixels (that we only use for visualization purposes in this work).

Certain applications of event cameras, such as image reconstruction [5, 6, 14] or video decompression [3], require processing all events. However, many applications like visual odometry or object tracking are known from standard cameras to work reliably on corners alone. Corners are useful features as they are well localized, highly informative, and do not suffer from the aperture problem. Additionally, they reduce an image (composed of millions of pixels) to a few hundred measurements. Similarly, we aim at reducing the event stream to a highly-informative *corner event* stream.

The first method for event-based corner detection was presented by [4]. They estimate the optical flow by fitting planes to the Surface of Active Events (a map of the timestamp of the latest event for each pixel) and searching for intersections. However, as plane fitting is a costly operation, the number of events per second that can be processed is rather limited. A more recent work [17] computes Harris corners on artificial frames generated from the events. While this method shows convincing results, it is not computationally efficient due to the underlying data structure and the required convolutions and matrix multiplications.

Other works focus on feature *tracking* in the event stream. However, they assume to know the shape of the features a priori [9], or they extract features from the frame and only track them using the events [16]. In [18], a probabilistic feature tracking algorithm using Expectation-Maximization is presented. Similarly to [17], they detect Harris corners in artificial frames, but instead of using a binarized frame, they use the event density over a temporal window.

Recently, several methods for camera tracking and visual odometry for event cameras have been presented. In [7] a visual-odometry method was proposed that works in real time, but requires a GPU. Instead of processing single events, EVO [13] accumulates events to build artificial frames as an intermediate step. These works show that real-time performance either requires massive parallel processing power (*e.g.*, a GPU) or departing from the event-based processing paradigm (*i.e.*, each event can change the state of the system). In [8], a visual-odometry algorithm using feature tracks was presented. They showed that feature-based methods work efficiently on event cameras. However, they required frames to detect corners and extract features before they could be tracked with the events.

In this paper, we present a fast method for corner detection in an event stream. Our detector is very efficient due to its design principle, which consists of working on the Surface of Active Events using only comparison operations, as opposed to plane fitting or computing gradients by convolution, as previous works. Our method asynchronously processes event by event with very low latency, thus preserving the characteristics of the event stream. It can serve as a lightweight, low-latency preprocessing step for subsequent higher-level algorithms such as visual odometry, object tracking, or recognition. Our implementation can process more than a million events per second on a single core, and typically reduces the event rate by a factor of 10 to 20.

The remainder of this paper is structured as follows. Section 2 describes our method, which we evaluate and compare to previous work in Section 3. Results are discussed and conclusions are drawn in Sections 4 and 5, respectively.

2 Method

Inspired by the FAST [15] corner detector for frames, we propose a corner detector for event streams that only uses pixel-wise comparisons. FAST considers a pixel as a corner if n contiguous pixels along a circle around the pixel of interest have all darker (or all brighter) intensity than the center pixel plus a threshold (typically, $n = 9$ on a circle of radius 3 with 16 pixels). In event cameras, brightness is encoded in the form of temporal contrast. More precisely, an event $e = (x, y, t, pol)$ is triggered at a pixel (x, y) at time t if the (logarithmic) brightness I reaches a predefined contrast threshold C (typically 15 %),

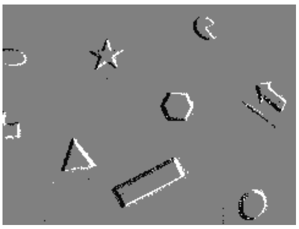
$$I(x,y,t) - I(x,y,t-\Delta t) = pol \cdot C, \tag{1}$$

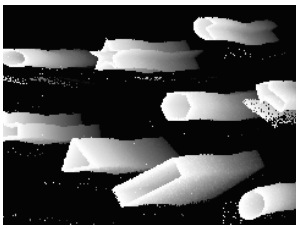
where $t - \Delta t$ is the time when the last event at that pixel was triggered, and pol, the polarity of the event, is the sign of the brightness change. Since visual information is represented by time and there is no notion of frames for event cameras, we propose to operate on the Surface of Active Events (SAE) [1], which is the function given by the timestamp of the most recent event at each pixel:

$$SAE : (x,y) \mapsto t. \tag{2}$$

Figure 2 shows a temporal window of events, the Surface of Active Events, as well as an intensity image for the same moment in time. Similarly to [17], we separate the events by polarity and process them independently.

Since this continuously and asynchronously updated representation is fundamentally different from intensity images, several changes are needed to make a FAST-like detector for


(a) Events over the last 33 ms. Positive events (white) and negative events (black)


(b) Surface of Active Events (SAE). Brightness represents time, from past (dark) to present (bright)

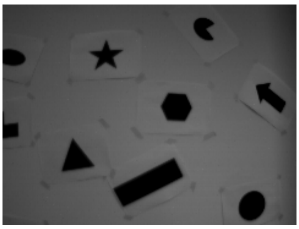
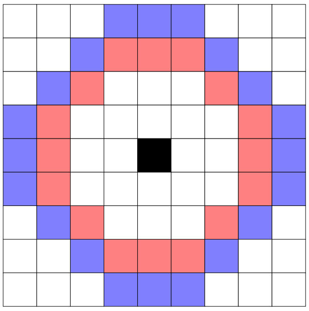
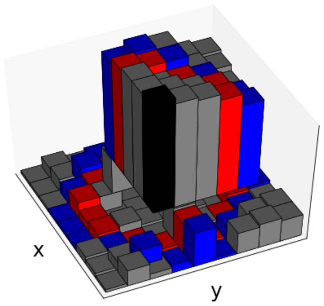

(c) Intensity Image (absolute brightness $I(x,y)$ in (1))

Figure 2: Signal used for corner detection: the Surface of Active Events (SAE).


(a) Circles of radius 3 and 4 pixels



(b) Visualization of Surface of Active Events (SAE)

Figure 3: Proposed Method. We compare the timestamps of the latest event of the pixels on two circles (red and blue) around the current event (in black). (a): The inner (red) and outer (blue) circle around the current event (black). (b): Visualization of the Surface of Active Events (SAE) around the current event (black) and of the circles used for the timestamp comparison. In this example, the event under consideration (center pixel) is classified as corner.

event cameras. First, we do not need to iterate over all pixels, but only check the current event using its local neighborhood. This check is performed asynchronously at the moment the event arrives. Second, the pixel values represent timestamps instead of intensity values and since the current event is considered the center pixel of the local neighborhood it always has the highest timestamp on the SAE. Therefore, comparisons of the pixels on the circle to the center one (as in FAST) are non-informative, and a different spatial comparison pattern (between pixels on the circle only) is required.

Our method analyzes the distribution of timestamps around the current event to decide on the presence of a corner. A moving corner will create, locally, a pixel map of timestamps such as that in Fig. 3(b), with two clearly separated regions (recent vs. old events, i.e., high vs. low values). Hence, we detect corners by searching for contiguous pixels with higher timestamps than the rest. We use circular segments for isotropic response and for efficiency (checking fewer pixels than the whole neighborhood). In contrast to existing methods [17], we completely avoid the computation of derivatives, which are expensive and amplify noise.

More specifically, we define a patch (local spatial neighborhood) of the SAE around the current event. In this patch, we focus on the pixels on two centered, concentric circles of radius 3 and 4 as shown in Fig. 3. Along each circle, the algorithm searches for segments of contiguous pixels (arcs) that are higher (*i.e.*, they have a more recent timestamp) than all other pixels on the circle. For the inner circle (red), we search for a segment of length between 3 and 6 (pixels). For the outer circle (blue), we search for a segment of length between 4 and 8. If we find such segments on both circles, we consider the current event to be a corner event. In the example in Fig. 3(b), the inner circle (red) and the outer circle have 5 and 6 contiguous pixels that are all higher than the other pixels along the circle, respectively. Therefore, the event in the center pixel is considered a corner.

Experimentally, we found that using additional inner circles (of radius 1 or 2) does not improve detection quality. We suspect that sensor noise is the main issue why corners in current event cameras cannot be located more precisely (feature-track methods [16, 18] also report localization errors in the range of 2 pixels). However, we also found that only using the inner circle of radius 3 provides significantly worse quality compared to using both circles. As can be seen from Fig. 3(a), circles of radius 3 and 4 constitute less than half of the pixels ($36/81 \approx 44\%$) in the 9×9 pixels patch. However, it is this region that we experimentally found to provide the most relevant information for corner detection and localization: larger circles do not provide good localization and smaller circles are not reliable to detect corners as they are more sensitive to sensor noise.

3 Evaluation

To evaluate the performance of our method, we first describe how we compute ground truth. Then we review the current state-of-the-art Harris detector [17] and describe our improvement. Finally, we compare the detection performance and runtime of Harris and our method on the Event-Camera Dataset [12].

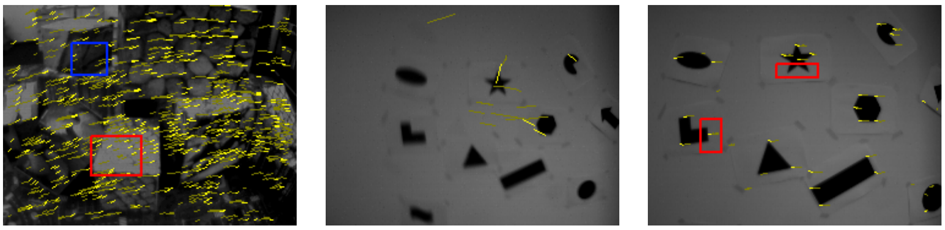
3.1 Ground Truth

3.1.1 Ground Truth using Frames

Establishing ground truth using the Lucas-Kanade tracking algorithm [11] on the frames of the DAVIS and interpolating between the frames suffers from severe limitations: (*i*) dynamic range: due to the limited dynamic range of the frames (around 55 dB), no corners are detected and tracked in over- and underexposed areas of the image (see Fig. 4(a)), (*ii*) frame rate: due to the limited frame rate of the sensor (around 25 Hz), tracking is lost in high-speed scenarios and linear interpolation is no longer a good approximation (see Fig. 4(b)), and (*iii*) corner interpretation: not all elements perceived as corners in the event stream are also recognized as corners in the images, and vice versa, even though they are repeatably detected and well tracked (see Fig. 4(c)). We experimented with different corner detectors (Harris and FAST) and different pyramid levels (up to 4 levels). Therefore, we propose a different method for establishing ground truth, which we describe next.

3.1.2 Ground Truth using Feature Tracks

Instead of using frames, we post-process the corner events to find “feature tracks”. A feature track is composed of an inner and an outer oblique cylinder in space-time (see Fig. 5(a)). We exhaustively search for feature tracks by creating hypotheses using two corner events and checking whether there are enough corner events in the inner cylinder and few corner events



(a) Dynamic Range (b) Frame Rate (c) Corner Interpretation

Figure 4: Issues with image-based ground truth. Frames from the DAVIS with superimposed detected corners (yellow traces) from frame-based Lukas-Kanade corner tracking implementation. (a): Corners in areas with overexposure (red) and underexposure (blue) are not detected in the image (frame from the `boxes` dataset). (b): Too much motion between two frames and motion blur causes Lucas-Kanade tracking to fail (frame from the `shapes` dataset). (c): Not the same corners are detected using the frames and the events (highlighted in red) (frame from the `shapes` dataset).

in the outer cylinder. We used 3 and 5 pixels for the inner and outer radius, respectively, a minimum of 30 inner events, and maximum ratio of outer-to-inner events of 25 %. We then consider all corner events belonging to such a feature track as inliers of the hypothesis and label them as correct corners.

3.2 Event-based Harris Detector

We compare our method with the event-based adaptation [17] of the Harris detector, which we describe here for completeness. Their method binarizes the SAE by the newest N events (depending on the experiment, they choose $N = 1000$ or $N = 2000$). Let Σ_b be a binary patch centered around the latest event, where 0 and 1 indicate the absence and presence of an event, respectively. Compute $I_x = \Sigma_b * G_x$ and $I_y = \Sigma_b * G_y$ as convolutions of the patch with 5×5 Sobel kernels G_x and $G_y = G_x^\top$, respectively. Compute Harris matrix

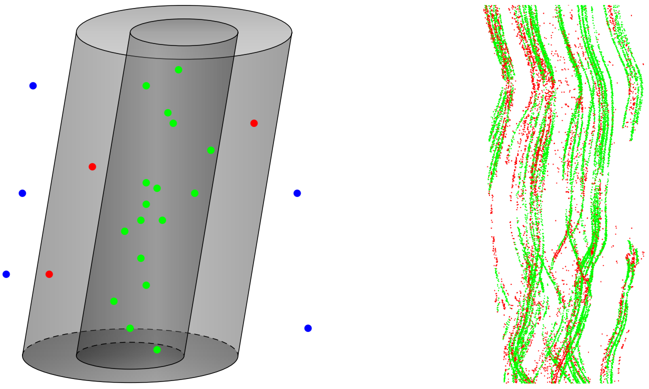
$$M = \sum_{e \in \Sigma_b} g(e)\, \nabla I(e) \nabla I^\top(e), \tag{3}$$

where g is a Gaussian weighting function with spread $\sigma = 1$ pixel, $\nabla I(e) = (I_x(e), I_y(e))^\top$ is the gradient at pixel e, and the 2×2 matrix $\nabla I(e) \nabla I^\top(e)$ is the point-wise structure tensor. Finally, the Harris score is computed as

$$H = \det(M) - k \cdot \operatorname{trace}(M)^2, \tag{4}$$

where $k = 0.04$ is a user-defined parameter. The event at the center of the patch is classified as a corner if its score H is large than a threshold S.

Spatially-Adaptive Harris Method. We propose the following improvement to the above-mentioned event-based Harris detector. Instead of choosing the newest N events for the whole image plane, which depends on the amount of texture in the scene, we choose the number of newest events locally, N_l, around the event under consideration. This choice is more sensible since only the latest events around the current one are relevant for deciding



(a) Feature Track (b) Labeled Data

Figure 5: Ground truth using feature tracks. (a) Feature tracks shown in space-time (time going upwards) represented by an inner and outer oblique cylinder. Points represent events that have been detected as corner events. Only if few corner events fall within the outer cylinder (red), the corners in inner cylinder are considered as correct detections (green). (b) Visualization of labeled data where each dot represents a corner detection. Green dots are part of a feature track (and therefore labeled as "true" corner), whereas red corners are considered false detections.

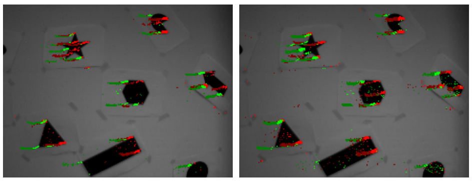
whether that event is a corner or not. Hence, our modified Harris detector adapts to the local visual information and is independent of the scene and the sensor size. We found that a patch of 9×9 pixels, the latest $N_l = 25$ events therein, and a threshold of $S = 8$ gave the best performance over a wide variety of datasets. Note that this is the same patch size as the one used in our proposed FAST-inspired corner-detection method (see Section 2).

3.3 Detector Performance

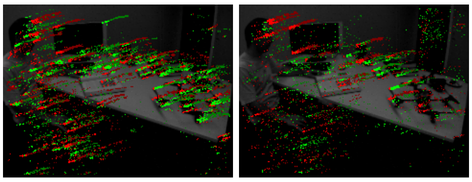
We compare the performance of our method with the spatially-adaptive Harris method described above. We evaluate the detectors on a representative subset of the datasets provided by the publicly available Event-Camera Dataset [12]. Each dataset is approximately one minute long and contains tens of millions of events. The scenes in the dataset range from simple shapes to natural and office environments. The motion speed, and therefore also the event rate, steadily increases during the datasets, reaching top values of over 3 m/s and 900 °/s, corresponding to activity peaks of 8 million events per second.[1]

The results are summarized in Table 1 and report the reduction rate in percentage (Red.) and the percentage of corner detections that could be matched to a feature track (FT, cf. Section 3.1.2). Figure 6 shows snapshots from both methods for all scenes overlaid on the frame. Figure 1(b) shows the corners in space-time together with the event stream for the `shapes` dataset during an interval of 1 s. Our method performs slightly worse than spatially-adaptive Harris on almost all datasets, but runs more than an order of magnitude faster, as shown in the next section (see Table 2). Both methods show the same trend: on scenes with low texture (such as `shapes` that contains only black-and-white patterns or `dynamic` that contains a desk, screen, books, and a moving person), both methods perform very well. On more finely-

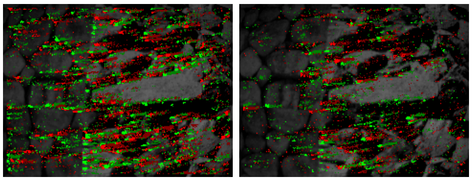
[1] Sampled at 1 ms intervals.



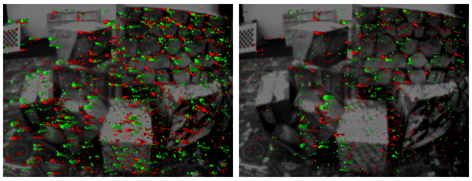
(a) `shapes`



(b) `dynamic`



(c) `poster`



(d) `boxes`

Figure 6: Visualization of corner detections of the last 33 ms and 100 ms in bright and dark color, respectively. Left and right images show the detections of Harris and our method, respectively. The datasets are from the Event-Camera Dataset [12]. The images are only shown for visualization and not used in either method. Color indicates the polarity of the corner events: green and red are events with positive and negative polarity, respectively.

textured scenes (such as `boxes` and `poster` that contain fine-grained natural pattern), fewer feature tracks can be found in the corner event stream. This does not necessarily mean that the detections are wrong, but rather that there might be corner-like structures very close in the image that cannot be separated well with the proposed ground-truth labeling technique. Further, the detector performance does not significantly depend on the motion type (rotation, translation, or 6-DOF), but rather on the level of texture in the scene.

3.4 Computational Performance

A major advantage of our algorithm is its runtime. Due to the asynchronous nature of event cameras, the event rate depends on the scene, the motion, and the sensor parameters (biases). The event rate in the Event Camera Dataset [12] is in the range of a few million of events per second (peaks of 8 million events per second). Our algorithm runs at 780 ns per event, allowing rates of up to 1.2 million events per second—more than an order of magnitude higher than previous methods. The results are summarized in Table 2. We used a single core of an Intel i7-3720QM CPU at 2.60 GHz for all experiments.

Since our method uses only a small local neighborhood of the events, parallelization is straightforward, if needed, with very little overhead. While this argument applies also to the Harris detector, the runtime per event remains critical to achieve *low-latency* performance: the DAVIS has a latency of 3 μs [2]. While the application of Harris triples this latency (11.6 μs), our method only yields around 30 % additional latency to the overall system (0.78 μs). The Harris method is slower for two main reasons: (*i*) a sort operation is required to find the latest N events on the SAE[2]; (*ii*) the computation involves convolutions (Sobel operator to compute ∇I) and matrix multiplications (Gaussian weighting). Our

[2]Just keeping the last N events in a queue is not equivalent, because pixels often fire more than one event. For Harris to work best, we need the last N events from *distinct* pixels.

Texture	**Dataset**	**Harris [17]**		**Ours**	
		Red.	**FT**	**Red.**	**FT**
low	`shapes_rotation`	92.7	74.1	88.9	74.7
	`shapes_translation`	91.7	78.3	87.8	77.5
	`shapes_6dof`	90.6	77.1	87.0	76.8
medium	`dynamic_rotation`	95.1	53.3	96.4	46.4
	`dynamic_translation`	95.3	62.1	96.7	52.1
	`dynamic_6dof`	95.4	55.9	96.4	49.4
high	`poster_rotation`	92.6	35.3	95.7	30.5
	`poster_translation`	92.3	39.5	95.8	35.9
	`poster_6dof`	92.4	35.3	95.6	32.1
high	`boxes_rotation`	92.1	32.9	96.7	25.2
	`boxes_translation`	92.4	37.0	96.7	30.5
	`boxes_6dof`	92.7	34.4	96.8	26.7

Table 1: Performance of Harris and our detector expressed as reduction rate (Red.) of the event stream and matched Feature Tracks (FT). Values are given in percentages.

Method	**Time per event** [μs]	**Max. event rate** [e/s]
Harris [17]	11.6	86,230
Ours	0.78	1,275,000

Table 2: Runtime comparison per event and corresponding maximum event rate.

method, instead, works only by direct, pixel-wise comparisons on the SAE. Thus, there are no expensive floating-point or sorting operations carried out on the pixel values. Additionally, as mentioned in Section 2 (Fig. 3(a)), our method is also fast because it processes only the most relevant part of the patch for the current event, which accounts to less than half of the pixels in the patch.

4 Discussion

The early reduction of the event stream to a *corner event* stream has several advantages. First, the corner detector acts as a filter: letting through only the most informative (i.e., less ambiguous) and well-localized events, and reducing, by more than an order of magnitude, the amount of data that must be processed at later stages in the pipeline, at little computational cost. Second, the low-latency and asynchronous nature of event camera output is maintained because each event is processed as soon as it is received. Since our algorithm runs very fast, very little additional latency is introduced. Third, the event-based paradigm of processing data on an event-by-event basis is preserved since we decide whether an event is a corner immediately and only using past events in a local neighborhood.

While the corner detection quality is slightly worse than an improved version of a state-of-the-art method, its computational performance is more than an order of magnitude faster. However, since both the Harris detector and our method operate on the same signal (the SAE), it would also be feasible to refine our corner event detections by post-processing them with the event-based Harris method.

5 Conclusion

We present a fast corner-detection algorithm that works on the asynchronous output stream of event cameras and preserves its low-latency and asynchronous characteristics. Our method reduces the event rate by 90%-95% and achieves a number of correctly-tracked features close to a state-of-the-art event-based corner detector (less than 10% difference). Since our method works directly on the Surface of Active Events using only binary comparisons, the processing time per event is very little and millions of events can be processed per second on a single core, which is more than 10 times faster than state-of-the-art methods. If needed, our method can be parallelized with almost no overhead since it uses only local information. Furthermore, as the resolution of future event cameras steadily increases, the event rate will also increase, and our algorithm will become more relevant to convert the event stream into a more manageable stream of informative and well-localized events.

Future work will include improving the detection quality further and investigation of non-maximum suppression methods, which is non-trivial due to the asynchronous nature of the events.

Acknowledgements. We thank Guillermo Gallego, Arren Glover and Valentina Vasco for valuable discussions. This research was supported by the DARPA FLA Program, the Swiss National Science Foundation through the National Center of Competence in Research Robotics (NCCR), the SNSF-ERC Starting Grant and the Qualcomm Innovation Fellowship.

References

[1] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE Trans. Neural Netw. Learn. Syst.*, 25(2):407–417, 2014. doi: 10.1109/TNNLS.2013.2273537.

[2] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240x180 130dB 3us latency global shutter spatiotemporal vision sensor. *IEEE J. Solid-State Circuits*, 49(10):2333–2341, 2014. ISSN 0018-9200. doi: 10.1109/JSSC.2014.2342715.

[3] Christian Brandli, Lorenz Muller, and Tobi Delbruck. Real-time, high-speed video decompression using a frame- and event-based davis sensor. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*, pages 686–689, June 2014. doi: 10.1109/ISCAS.2014.6865228.

[4] Xavier Clady, Sio-Hoi Ieng, and Ryad Benosman. Asynchronous event-based corner detection and matching. *Neural Netw.*, 66:91–106, 2015. ISSN 0893-6080. doi: 10.1016/j.neunet.2015.02.013.

[5] Matthew Cook, Luca Gugelmann, Florian Jug, Christoph Krautz, and Angelika Steger. Interacting maps for fast visual interpretation. In *Int. Joint Conf. Neural Netw. (IJCNN)*, pages 770–776, 2011. doi: 10.1109/IJCNN.2011.6033299.

[6] Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew J. Davison. Simultaneous mosaicing and tracking with an event camera. In *British Machine Vis. Conf. (BMVC)*, 2014. doi: 10.5244/C.28.26.

[7] Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. Real-time 3D reconstruction and 6-DoF tracking with an event camera. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 349–364, 2016. doi: 10.1007/978-3-319-46466-4_21.

[8] Beat Kueng, Elias Mueggler, Guillermo Gallego, and Davide Scaramuzza. Low-latency visual odometry using event-based feature tracks. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pages 16–23, Daejeon, Korea, October 2016. doi: 10.1109/IROS.2016.7758089.

[9] Xavier Lagorce, Cédric Meyer, Sio-Hoi Ieng, David Filliat, and Ryad Benosman. Asynchronous event-based multikernel algorithm for high-speed visual features tracking. *IEEE Trans. Neural Netw. Learn. Syst.*, 26(8):1710–1720, August 2015. ISSN 2162-237X. doi: 10.1109/TNNLS.2014.2352401.

[10] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circuits*, 43 (2):566–576, 2008. doi: 10.1109/JSSC.2007.914337.

[11] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. Artificial Intell.*, pages 121–130, 1981.

[12] 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. *Int. J. Robot. Research*, 2017. doi: 10.1177/0278364917691115.

[13] Henri Rebecq, Timo Horstschäfer, Guillermo Gallego, and Davide Scaramuzza. EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time. *IEEE Robot. Autom. Lett.*, 2:593–600, 2017. ISSN 2377-3766. doi: 10.1109/LRA.2016.2645143.

[14] Christian Reinbacher, Gottfried Graber, and Thomas Pock. Real-time intensity-image reconstruction for event cameras using manifold regularisation. In *British Machine Vis. Conf. (BMVC)*, 2016.

[15] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 430–443, 2006. doi: 10.1007/11744023_34.

[16] David Tedaldi, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. Feature detection and tracking with the dynamic and active-pixel vision sensor (DAVIS). In *Int. Conf. Event-Based Control, Comm. Signal Proc. (EBCCSP)*, pages 1–7, Krakow, Poland, June 2016. doi: 10.1109/EBCCSP.2016.7605086.

[17] Valentina Vasco, Arren Glover, and Chiara Bartolozzi. Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2016. doi: 10.1109/IROS.2016.7759610.

[18] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based feature tracking with probabilistic data association. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2017.