EKLT: Asynchronous Photometric Feature Tracking Using Events and Frames

Gehrig, Daniel ; Rebecq, Henri ; Gallego, Guillermo ; Scaramuzza, Davide

Abstract: We present EKLT, a feature tracking method that leverages the complementarity of event cameras and standard cameras to track visual features with high temporal resolution. Event cameras are novel sensors that output pixel-level brightness changes, called "events". They offer significant advantages over standard cameras, namely a very high dynamic range, no motion blur, and a latency in the order of microseconds. However, because the same scene pattern can produce different events depending on the motion direction, establishing event correspondences across time is challenging. By contrast, standard cameras provide intensity measurements (frames) that do not depend on motion direction. Our method extracts features on frames and subsequently tracks them asynchronously using events, thereby exploiting the best of both types of data: the frames provide a photometric representation that does not depend on motion direction and the events provide updates with high temporal resolution. In contrast to previous works, which are based on heuristics, this is the first principled method that uses intensity measurements directly, based on a generative event model within a maximum-likelihood framework. As a result, our method produces feature tracks that are more accurate than the state of the art, across a wide variety of scenes.

DOI: https://doi.org/10.1007/s11263-019-01209-w

Accepted for publication at the International Journal of Computer Vision (IJCV), Aug. 2019

EKLT: Asynchronous Photometric Feature Tracking using Events and Frames

Daniel Gehrig · Henri Rebecq · Guillermo Gallego · Davide Scaramuzza

Received: Jan. 2019 / Accepted: Aug. 2019

Abstract We present EKLT, a feature tracking method that leverages the complementarity of event cameras and standard cameras to track visual features with high temporal resolution. Event cameras are novel sensors that output pixel-level brightness changes, called "events". They offer significant advantages over standard cameras, namely a very high dynamic range, no motion blur, and a latency in the order of microseconds. However, because the same scene pattern can produce different events depending on the motion direction, establishing event correspondences across time is challenging. By contrast, standard cameras provide intensity measurements (frames) that do not depend on motion direction. Our method extracts features on frames and subsequently tracks them asynchronously using events, thereby exploiting the best of both types of data: the frames provide a photometric representation that does not depend on motion direction and the events provide updates with high temporal resolution. In contrast to previous works, which are based on heuristics, this is the first principled method that uses intensity measurements directly, based on a generative event model within a maximum-likelihood framework. As a result, our method produces feature tracks that are more accurate than the state of the art, across a wide variety of scenes.

The authors are with the Robotics and Perception Group, Dept. of Informatics, University of Zurich, and Dept. of Neuroinformatics, University of Zurich and ETH Zurich, Switzerland — http://rpg.ifi.uzh.ch.

Multimedia Material

A supplemental video for this work is available at https://youtu.be/ZyD1YPW1h4U and the tracking code can be found here: https://github.com/uzh-rpg/rpg_eklt. In addition, the evaluation code can be found here: https://github.com/uzh-rpg/rpg_feature_tracking_analysis.

1 Introduction

Event cameras, such as the Dynamic Vision Sensor (DVS) [1], work very differently from traditional cameras (Fig. 1). They have independent pixels that send information (called "events") only in presence of brightness changes in the scene at the time they occur. Thus, their output is not an intensity image but a *stream of asynchronous events*. Event cameras excel at sensing motion, and they do so with very low latency (1 microsecond). However, they do not provide absolute intensity measurements, rather they measure only *changes of intensity*. Conversely, standard cameras provide *direct* intensity measurements for every pixel, but with comparatively much higher latency (10–20 ms). Event cameras and standard cameras are, thus, complementary, which calls for the development of novel algorithms capable of combining the specific advantages of both cameras to perform computer vision tasks with high temporal resolution. In fact, the Dynamic and Active-pixel Vision Sensor (DAVIS) [2] was recently introduced (2014) in that spirit. It is a sensor comprising an asynchronous event-based sensor and a standard frame-based camera in the same pixel array. A survey on event cameras, algorithms, and applications can be found in [3].

We tackle the problem of feature tracking using both events and frames, such as those provided by the DAVIS. Our goal is to combine both types of intensity measurements to maximize tracking accuracy, and for this reason we develop a maximum likelihood approach based on a generative event model.

Feature tracking is an important research topic in computer vision, and has been widely studied in the last decades. It is a core building block of numerous applications, such as object tracking [4] or Simultaneous Localization and Mapping (SLAM) [5–8]. While feature detection and tracking methods for frame-based cameras are well established, they cannot track in the blind time between consecutive frames, and are expensive because they process information from all pixels, even in the absence of motion in the scene. Conversely, event cameras acquire only relevant information for tracking and respond asynchronously, thus, filling the blind time between consecutive frames.

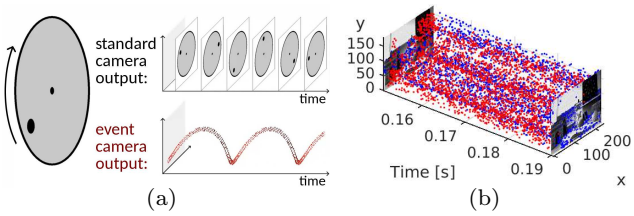


Fig. 1: Fig. 1(a): Comparison of the output of a standard frame-based camera and an event camera when facing a black dot on a rotating disk (figure adapted from [12], and animated here: https://youtu.be/LauQ6LWTkxM?t=25). The standard camera outputs frames at a fixed rate, thus sending redundant information when there is no motion in the scene. Event cameras respond to pixel-level *brightness changes* with microsecond latency. Fig. 1(b): A combined frame and event-based sensor such as the DAVIS [2] provides both standard frames and the events that occurred in between. Events are colored according to polarity: blue (brightness increase) and red (brightness decrease).

1.1 Contribution

In this work we present a feature tracker which we term EKLT, event-based Lucas-Kanade, that extracts features on frames and subsequently tracks them using only events. This allows us to take advantage of the asynchronous, high dynamic range and low-latency nature of the events to produce feature tracks with high temporal resolution. However, associating individual events coming from the same object is challenging due to the varying appearance of the events with respect to the motion of the object on the image plane, which is known as the data association problem. In contrast to previous works, which used heuristics to solve for data association, we introduce a feature tracker that combines events and frames in a way that (*i*) fully exploits the strength of the brightness gradients causing the events, (*ii*) circumvents the data association problem, and (*iii*) leverages a generative model to explain, in a maximum likelihood formulation, how events are related to brightness patterns (i.e., features) on the frames. We thoroughly evaluate the proposed tracker using sequences from publicly available datasets [9,10], and show its performance both on man-made environments with large contrast and on natural scenes.

This paper is based on our previous work [11], which we extend in several ways:

- We provide an interpretation of the method as an extension for event cameras of the popular Lucas-Kanade tracker (KLT) originally designed for frame-based cameras (Section 4.5).
- We provide additional experiments with sequences from driving and flying scenarios (Section 5).
- We compare our method against four baselines (Section 5.2), and show that our approach provides more accurate feature tracks.
- We carry out a quantitative analysis of the dependency of the proposed method on grayscale frames, by comparing against frames obtained by a state-of-the-art image reconstruction method for event cameras (Section 6).

The paper is organized as follows: Section 2 reviews prior work; Section 3 motivates why event-based feature tracking is challenging; Section 4 presents our solution; Sections 5 and 6 analyze the performance of the proposed method through extensive evaluation. Finally, Sections 7 and 8 briefly discuss future work and conclude the paper, respectively.

2 Related Work

Feature detection and tracking with event cameras is a major research topic [13–23], where the goal is to unlock the capabilities of event cameras and use them to solve these classical problems in computer vision in challenging scenarios inaccessible to standard cameras, such as low-power, high-speed and high dynamic range (HDR) scenarios. A good survey of algorithms for event cameras can be found in [3]. Recently, extensions of popular image-based keypoint detectors, such as Harris [24]

and FAST [25], have been developed for event cameras [21–23]. Detectors based on the distribution of optical flow [26] for recognition applications have also been proposed for event cameras [20]. Finally, most event-based trackers use binary feature templates, either predefined [15–17] or built from events [14], to which they align new events by means of iterative point-set–based methods, such as Iterative Closest Point (ICP) [27].

Our work is most related to [13], since it also combines frames and events for feature tracking. The approach in [13] detects patches of Canny edges around Harris corners in the grayscale frames and then tracks such local edge patterns using ICP on the event stream. Thus, the patch of Canny edges acts as a template to which the events are registered to yield tracking information. Under the simplifying assumption that events are mostly generated by strong edges, the Canny edgemap template is used as a proxy for the underlying grayscale pattern that causes the events. The method in [13] converts the tracking problem into a geometric, point-set alignment problem: the event coordinates are compared against the point template given by the pixel locations of the Canny edges. Hence, pixels where no events are generated are, efficiently, not processed. However, the method has two drawbacks: (*i*) the information about the strength of the edges is lost (since the point template used for tracking is obtained from a binary edgemap) (*ii*) explicit correspondences (i.e., data association) between the events and the template need to be established for ICP-based registration. The method in [14] can be interpreted as an extension of [13] with (*i*) the Canny-edge patches replaced by motion-corrected event point sets and (*ii*) the correspondences computed in a soft manner using Expectation-Maximization (EM). The methods in [13, 14], which process events as points sets, are inspired by prior event-based ICP trackers [15, 16].

Like [13,14], our method can be used to track generic features, as opposed to constrained (i.e., predefined) edge patterns. However, our method differs from [13,14] in that (*i*) we take into account the strength of the edge pattern causing the events and (*ii*) we do not need to establish correspondences between the events and the edgemap template. In contrast to [13, 14], which use a point-set template for event alignment, our method uses the spatial gradient of the raw intensity image, directly, as a template. Correspondences are implicitly established as a consequence of the proposed image-based registration approach (Section 4), but before that, let us motivate why establishing correspondences is challenging with event cameras.

3 The Challenge of Data Association for Feature Tracking

The main challenge in tracking scene features (i.e., edge patterns) with an event camera is that, because this sensor responds to temporal changes of intensity (caused by moving edges on the image plane), the appearance of the feature varies depending on its motion, and thus, it may continuously change in time (see Fig. 2). Feature tracking using events requires the establishment of correspondences between events at different times (i.e., data association), which is difficult due to the above-mentioned varying feature appearance (Fig. 2).

Instead, if additional information is available, such as the absolute intensity of the pattern to be tracked (i.e., a time-invariant representation or "map" of the feature), such as in Fig. 2(a), then event correspondences may be established indirectly, via establishing correspondences between the events and the intensity pattern. This, however, additionally requires to continuously estimate the motion (optic flow) of the pattern since it determines the appearance of the events. This is in fact an important component of our approach. As we show in Section 4, our method is based on a model to generate a prediction of the time-varying event-feature appearance using a given frame and an estimate of the optic flow. This generative model has not been considered in previous feature tracking methods, such as [13, 14, 28].

4 Methodology

An event camera has independent pixels that respond to changes in the continuous brightness signal[1] $L(\mathbf{u}, t)$. Specifically, an event $e_k = (x_k, y_k, t_k, p_k)$ is triggered at pixel $\mathbf{u}_k = (x_k, y_k)^\top$ and at time t_k as soon as the brightness increment since the last event at the pixel reaches a threshold $\pm C$ (with $C > 0$):

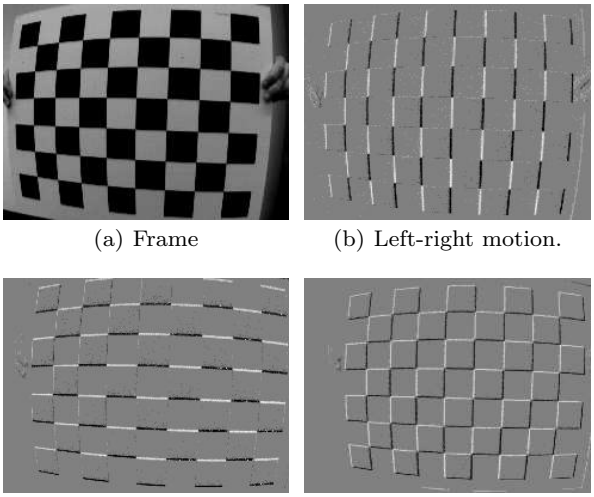
$$\Delta L(\mathbf{u}_k, t_k) \doteq L(\mathbf{u}_k, t_k) - L(\mathbf{u}_k, t_k - \Delta t_k) = p_k C, \quad (1)$$

where Δt_k is the time since the last event at the same pixel, and $p_k \in \{-1, +1\}$ is the event polarity (i.e., the sign of the brightness change). Eq. (1) is the event generation equation of an ideal sensor [29, 30].

4.1 Brightness-Increment Images from Events and Frames

Pixel-wise accumulation of event polarities over a time interval $\Delta\tau$ produces an image $\Delta L(\mathbf{u})$ with the amount

[1] Event cameras such as the DVS [1] respond to logarithmic brightness changes, i.e., $L \doteq \log I$, with brightness signal I, so that (1) represents logarithmic changes.



(a) Frame (b) Left-right motion. (c) Up-down motion. (d) Diagonal motion.

Fig. 2: Result of moving a checkerboard (a) in different directions in front of an event camera. (b)-(d) show brightness increment images (Eq. (2)) obtained by accumulating events over a short time interval. Pixels that do not change intensity are represented in gray, whereas pixels that increased or decreased intensity are represented in bright and dark, respectively. Clearly, (b) (only vertical edges), (c) (only horizontal edges), and (d) cannot be related to each other without the prior knowledge of the underlying photometric information provided by (a).

of brightness change that occurred during the interval (Fig. 3a),

$$\Delta L(\mathbf{u}) = \sum_{t_k \in \Delta\tau} p_k C\, \delta(\mathbf{u} - \mathbf{u}_k), \tag{2}$$

where δ is the Kronecker delta due to its discrete argument (pixels on a lattice).

For small $\Delta\tau$, such as in the example of Fig. 3a, the brightness increments (2) are due to moving edges according to the event generation model[2]:

$$\Delta L(\mathbf{u}) \approx -\nabla L(\mathbf{u}) \cdot \mathbf{v}(\mathbf{u}) \Delta\tau, \tag{3}$$

that is, increments are caused by brightness gradients $\nabla L(\mathbf{u}) = \left(\frac{\partial L}{\partial x}, \frac{\partial L}{\partial y}\right)^\top$ moving with velocity $\mathbf{v}(\mathbf{u})$ over a displacement $\Delta\mathbf{u} \doteq \mathbf{v}\Delta\tau$ (see Fig. 3b). As the dot product in (3) conveys, (i) if the motion is parallel to the edge ($\mathbf{v} \perp \nabla L$), the increment vanishes, i.e., no events are generated; (ii) if the motion is perpendicular to the edge ($\mathbf{v} \parallel \nabla L$) events are generated at the highest rate. From now on (and in Fig. 3b) we denote the modeled increment (3) using a hat, $\Delta\hat{L}$, and the frame by $\hat{L}$.

Remark. Despite the fact that brightness increment images (2) (Fig. 3a) have been used in the past for several tasks, such as segmentation [31] and stereo [32] or optical flow [33] computation, our contribution consists of using them in combination with a forward model (3) to solve for feature tracking. Specifically, as we show in Section 4.2, we use a generative model of the events (4) to predict brightness increments ("observations") from a few explanatory variables (feature brightness, warp and optic flow) characterizing the scene under investigation (Fig. 5). Then, we use an optimization framework to compute the unknown explanatory variables from the error between the observations and the predictions. This approach can be applied to solve other problems, such as camera tracking [34].

4.2 Optimization Framework

Following a maximum likelihood approach, we propose to use the difference between the observed brightness changes ΔL from the events (2) and the predicted ones $\Delta\hat{L}$ from the brightness signal $\hat{L}$ of the frames (3) to estimate the motion parameters that best explain the events according to an optimization score.

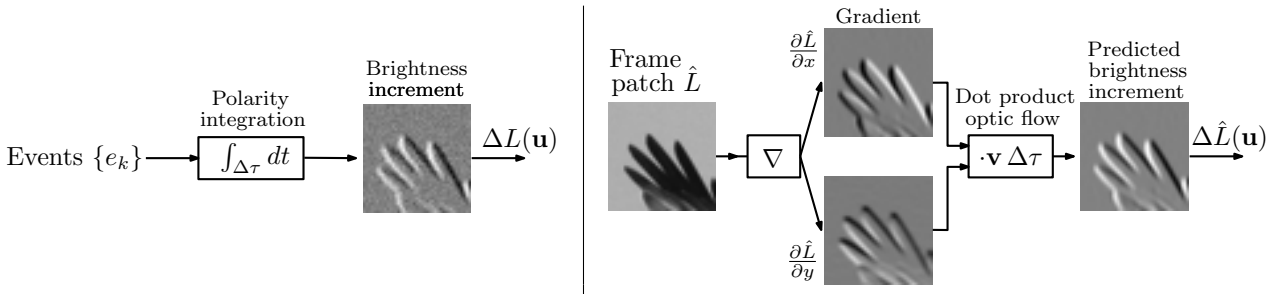
More specifically, we pose the feature tracking problem using events and frames as that of *image registration* [35, 36], between images (2) and (3). Effectively, frames act as feature templates with respect to which events are registered. As is standard, let us assume that (2) and (3) are compared over small patches ($\mathcal{P}$) containing distinctive patterns, and further assume that the optic flow $\mathbf{v}$ is constant for all pixels in the patch (same regularization as [35]).

Letting $\hat{L}$ be given by an intensity frame at time $t = 0$ and letting ΔL be given by events in a space-time window at a later time t (see Fig. 4), our goal is to find the registration parameters $\mathbf{p}$ and the velocity $\mathbf{v}$ that maximize the similarity between $\Delta L(\mathbf{u})$ and

$$\Delta\hat{L}(\mathbf{u}; \mathbf{p}, \mathbf{v}) \doteq -\nabla\hat{L}(\mathbf{W}(\mathbf{u}; \mathbf{p})) \cdot \mathbf{v}\Delta\tau, \tag{4}$$

where $\mathbf{W}$ is the warping map used for the registration. We explicitly model optic flow $\mathbf{v}$ instead of approximating it by finite differences of past registration parameters to avoid introducing approximation errors and to avoid error propagation from past noisy feature positions. A block diagram showing how both brightness increments are computed, including the effect of the warp $\mathbf{W}$, is given in Fig. 5.

[2] Eq. (3) can be shown [30] by substituting the brightness constancy assumption (i.e., optical flow constraint) $\frac{\partial L}{\partial t}(\mathbf{u}(t), t) + \nabla L(\mathbf{u}(t), t) \cdot \dot{\mathbf{u}}(t) = 0$, with image-point velocity $\mathbf{v} \equiv \dot{\mathbf{u}}$, in Taylor's approximation $\Delta L(\mathbf{u}, t) \doteq L(\mathbf{u}, t) - L(\mathbf{u}, t - \Delta\tau) \approx \frac{\partial L}{\partial t}(\mathbf{u}, t)\Delta\tau$.



(a) $\Delta L(\mathbf{u})$ given by the events (2).

(b) $\Delta\hat{L}(\mathbf{u})$ predicted using the frame (3).

Fig. 3: Brightness increments given by the events (2) vs. predicted from the frame and the optic flow using the generative model (3). Pixels of $L(\mathbf{u})$ that do not change intensity are represented in gray in ΔL, whereas pixels that increased or decreased intensity are represented in bright and dark, respectively.

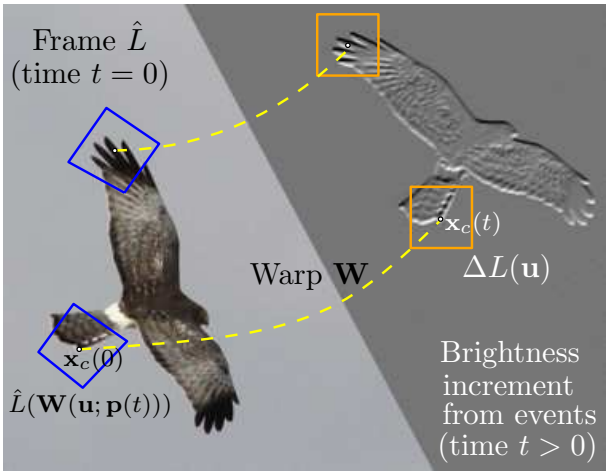


Fig. 4: Illustration of tracking for two independent patches. Events in a space-time window at time $t > 0$ are collected into a patch of brightness increments $\Delta L(\mathbf{u})$ (in orange), which is compared, via a warp (i.e., geometric transformation) $\mathbf{W}$ against a predicted brightness-increment image based on $\hat{L}$ (given at $t = 0$) around the initial feature location (in blue). Patches are computed as shown in Fig. 5, and are compared in the objective function (7).

Assuming that the difference $\Delta L - \Delta\hat{L}$ follows a zero-mean additive Gaussian distribution with variance σ^2 [1], we define the likelihood function of the set of events $\mathcal{E} \doteq \{e_k\}_{k=1}^{N_e}$ producing ΔL as

$$p(\mathcal{E} \,|\, \mathbf{p}, \mathbf{v}, \hat{L}) \propto e^{-\frac{1}{2\sigma^2}\int_{\mathcal{P}} \left(\Delta L(\mathbf{u}) - \Delta\hat{L}(\mathbf{u};\mathbf{p},\mathbf{v})\right)^2 d\mathbf{u}}. \tag{5}$$

Maximizing this likelihood with respect to the motion parameters $\mathbf{p}$ and $\mathbf{v}$ (since $\hat{L}$ is known) yields the minimization of the L^2 norm of the photometric residual,

$$\min_{\mathbf{p},\mathbf{v}} \|\Delta L(\mathbf{u}) - \Delta\hat{L}(\mathbf{u};\mathbf{p},\mathbf{v})\|^2_{L^2(\mathcal{P})} \tag{6}$$

where $\|f(\mathbf{u})\|^2_{L^2(\mathcal{P})} \doteq \int_{\mathcal{P}} f^2(\mathbf{u}) d\mathbf{u}$.

However, the objective function (6) depends on the contrast sensitivity C (via (2)), which is typically unknown in practice. Inspired by [36], we propose to minimize the difference between unit-norm patches:

$$\min_{\mathbf{p},\mathbf{v}} \left\| \frac{\Delta L(\mathbf{u})}{\|\Delta L(\mathbf{u})\|_{L^2(\mathcal{P})}} - \frac{\Delta\hat{L}(\mathbf{u};\mathbf{p},\mathbf{v})}{\|\Delta\hat{L}(\mathbf{u};\mathbf{p},\mathbf{v})\|_{L^2(\mathcal{P})}} \right\|^2_{L^2(\mathcal{P})}, \tag{7}$$

which cancels the terms in C and $\Delta\tau$, and only depends on the direction of the feature velocity $\mathbf{v}$. In this generic formulation, the same type of parametric warps $\mathbf{W}$ as for image registration can be considered (projective, affine, etc.). For simplicity, we consider warps given by rigid-body motions in the image plane,

$$\mathbf{W}(\mathbf{u};\mathbf{p}) = \mathtt{R}(\mathbf{p})\mathbf{u} + \mathbf{t}(\mathbf{p}), \tag{8}$$

where $(\mathtt{R}, \mathbf{t}) \in SE(2)$. The objective function (7) is optimized using the non-linear least squares framework provided in the Ceres software [37].

4.3 Discussion of the Approach

One of the most interesting characteristics of the proposed method (7) is that it is based on a generative model for the events (3). As shown in Fig. 5, the frame $\hat{L}$ is used to produce a registration template $\Delta\hat{L}$ that changes depending on $\mathbf{v}$ (weighted according to the dot product) in order to best fit the motion-dependent event data ΔL, and so our method not only estimates the warping parameters of the event feature but also its optic flow. This optic flow dependency was not explicitly modeled in previous works, such as [13, 14, 28]. Moreover, for the template, we use the full gradient information of the frame $\nabla\hat{L}$, as opposed to its Canny (i.e., binary-thresholded) version [13], which provides higher accuracy and the ability to track less salient patterns.

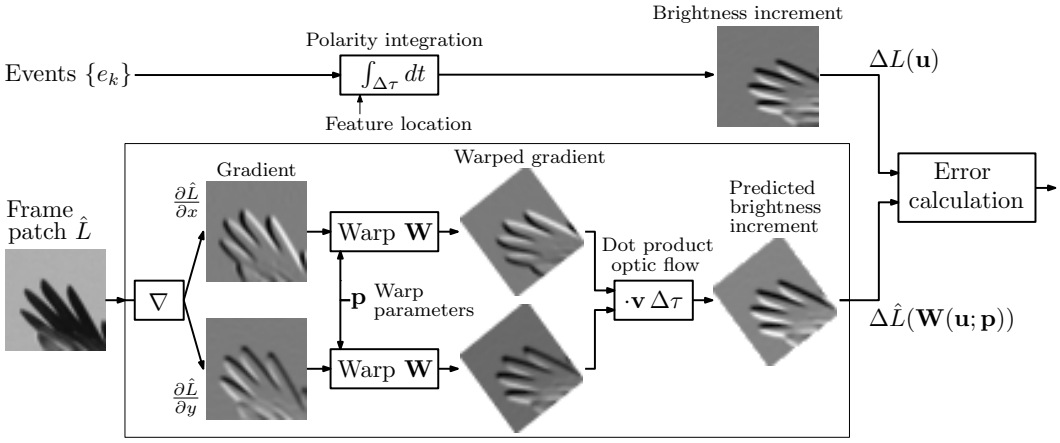


Fig. 5: Block diagram showing how the brightness increments being compared ($\Delta L, \Delta\hat{L}$) are computed for one of the patches in Fig. 4. The top of the diagram depicts the brightness increment obtained by event integration (2), whereas the bottom of the diagram shows the generative event model stemming from the frame (3).

Another characteristic of our method is that it does not suffer from the problem of establishing event-to-feature correspondences, as opposed to ICP methods [13, 14]. We borrow the implicit pixel-to-pixel data association typical of image registration methods by creating, from events, a convenient image representation. Hence, our method has smaller complexity (establishing data association in ICP [13] has quadratic complexity) and is more robust since it is less prone to be trapped in local minima caused by data association (as it is shown in Section A.1). As optimization iterations progress, all event correspondences evolve jointly as a single entity according to the evolution of the warped pixel grid.

Additionally, monitoring the evolution of the minimum cost values (7) provides a sound criterion to detect feature track loss and, therefore, initialize new feature tracks (e.g., in the next frame or by acquiring a new frame on demand).

Algorithm 1 Photometric feature tracking using events and frames

Feature initialization:
- Detect Harris corners [24] on the frame $\hat{L}(\mathbf{u})$, extract intensity patches around corner points and compute $\nabla\hat{L}(\mathbf{u})$.
- Set patches $\Delta L(\mathbf{u}) = 0$, set initial registration parameters $\mathbf{p}$ to those of the identity warp, and set the number of events N_e to integrate on each patch.

Feature tracking:
for each incoming event **do**
- Update the patches containing the event (i.e., accumulate polarity pixel-wise (2)).
for each patch $\Delta L(\mathbf{u})$ (once N_e events have been collected (2)) **do**
- Minimize the objective function (7), to get parameters $\mathbf{p}$ and optic flow $\mathbf{v}$.
- Update the registration parameters $\mathbf{p}$ of the feature patch (e.g., position).
- Reset the patch ($\Delta L(\mathbf{u}) = 0$) and recompute N_e.

4.4 Algorithm

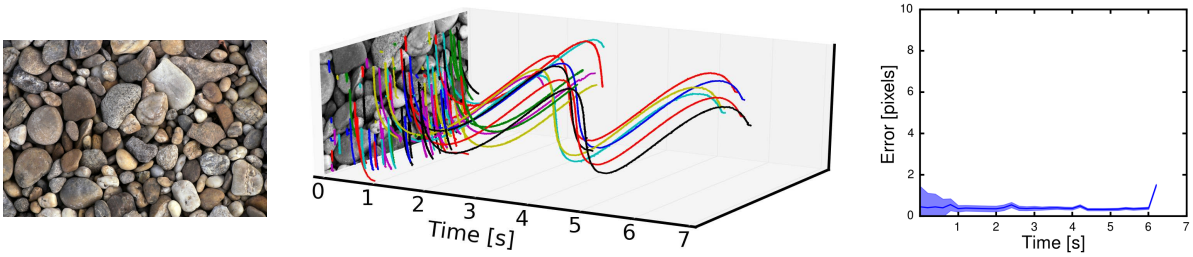
The steps of our asynchronous, low-latency feature tracker are summarized in Algorithm 1, which consists of two phases: (*i*) initialization of the feature patch and (*ii*) tracking the pattern in the patch using events according to (7). Multiple patches are tracked independently from one another. To compute a patch $\Delta L(\mathbf{u})$, (2), we integrate over a given number of events N_e [28,38–40] rather than over a fixed time $\Delta\tau$ [41, 42]. Hence, tracking is asynchronous, as soon as N_e events are acquired on the patch (2), which typically happens at rates higher than the frame rate of the standard camera (~ 10 times higher). Section 5.4 provides an analysis of the sensitivity of the method with respect to N_e and a formula to compute a sensible value, to be used in Algorithm 1.

4.5 Connection with the Lucas-Kanade Method

The approach (6) can be interpreted as an extension of the KLT tracker [35, 43] to the case of event-based cameras, where we estimate, in addition to the feature's warping parameters, its optical flow.

Lucas-Kanade. The goal of the Lucas-Kanade method [35, 43] is to minimize the photometric error, using the L^2 norm criterion, between an image I and another one T that are related via a geometric distortion described by a warp $\mathbf{W}(\mathbf{u};\mathbf{p})$ with parameters $\mathbf{p} \in \mathbb{R}^M$:

$$\min_{\mathbf{p}} \|(I \circ \mathbf{W})(\mathbf{u}) - T(\mathbf{u})\|^2_{L^2(\mathcal{P})}. \tag{9}$$



(a) (b) (c)

Fig. 6: Feature tracking results on simulated data. (a) Example texture used to generate synthetic events in the simulator [9]. (b) Qualitative feature tracks represented as curves in space-time. (c) Mean tracking error (center line) and fraction of surviving features (width of the band around the center line) as a function of time. Our features are tracked with 0.4 pixel accuracy on average.

This is a non-linear least-squares (NLLS) problem, since it can be written as the (squared) L^2 norm of a residual:

$$\min_{\mathbf{p}} \|r(\mathbf{p})\|^2_{L^2(\mathcal{P})}, \quad r(\mathbf{p}) \doteq (I \circ \mathbf{W})(\mathbf{u}) - T(\mathbf{u}), \tag{10}$$

where the pixel-wise dependency was omitted for simplicity of notation. Problem (10) is solved iteratively using Gauss-Newton's method, which consists of linearizing the residual using Taylor's expansion

$$r(\mathbf{p} + \Delta\mathbf{p}) \approx r(\mathbf{p}) + \nabla r(\mathbf{p}) \cdot \Delta\mathbf{p}, \tag{11}$$

where $\nabla r(\mathbf{p}) \doteq (\frac{\partial r}{\partial \mathbf{p}}(\mathbf{p}))^\top$, and finding the parameter update $\Delta\mathbf{p}$ that minimizes the norm of the linearized residual (a quadratic expression in $\Delta\mathbf{p}$). This yields a linear system of equations (the normal equations), $\mathtt{A}\,\Delta\mathbf{p} = \mathbf{b}$, with

$$\begin{aligned} \mathtt{A} \equiv \mathtt{A}(\mathbf{p}) &\doteq \int_{\mathcal{P}} \nabla r(\mathbf{p})\, (\nabla r(\mathbf{p}))^\top d\mathbf{u}, \\ \mathbf{b} \equiv \mathbf{b}(\mathbf{p}) &\doteq -\int_{\mathcal{P}} \nabla r(\mathbf{p})\, r(\mathbf{p})\, d\mathbf{u}. \end{aligned} \tag{12}$$

The parameters are iteratively updated, $\mathbf{p} \leftarrow \mathbf{p} + \Delta\mathbf{p}$, with $\Delta\mathbf{p} = \mathtt{A}^{-1}\mathbf{b}$, until convergence.

Our proposal. The approach (6) also minimizes a photometric error using the L^2 norm criterion, as in the KLT method (9). However, the goal of (6) is to register two brightness increment images (rather than "absolute" brightness ones), one rendered by accumulating events and the other one computed from the gradient of the brightness frame (Fig. 5). This requires to optimize for both, the geometric transformation $\mathbf{W}$ (as in original KLT) and the appearance of the feature, which depends on the motion. Since the motion (optical flow) is unknown, we *simultaneously* vary the warp parameters $\mathbf{p}$ and the optic flow $\mathbf{v}$ of the feature to find the geometric distortion *and* appearance of the predicted image (4) that best explains the image obtained by accumulating events.

Redefining the residual as $\Delta L - \Delta\hat{L}$, i.e.,

$$r(\tilde{\mathbf{p}}) \doteq \Delta L(\mathbf{u}) + \nabla\hat{L}(\mathbf{W}(\mathbf{u};\mathbf{p})) \cdot \mathbf{v}\Delta\tau \tag{13}$$

with respect to the augmented parameter vector $\tilde{\mathbf{p}} \doteq (\mathbf{p}^\top, \mathbf{v}^\top)^\top \in \mathbb{R}^{M+2}$, allows us to apply the Gauss-Newton method to minimize the photometric error (6). This yields a linear system $\tilde{\mathtt{A}}\,\Delta q = \tilde{\mathbf{b}}$, as in the Lucas-Kanade method (12), but now with $(M+2)$ unknowns instead of M (due to the two additional unknowns in $\mathbf{v}$). The two components of $\tilde{\mathbf{p}}$, namely the warp parameters $\mathbf{p}$ and the optic flow $\mathbf{v}$ of the feature, are thus jointly estimated. Alternatively, defining $r(\tilde{\mathbf{p}}) \doteq \Delta L(\mathbf{W}(\mathbf{u};\mathbf{p})) + \nabla\hat{L}(\mathbf{u}) \cdot \mathbf{v}\Delta\tau$ reduces the interaction between $\mathbf{p}$ and $\mathbf{v}$, yielding simpler derivatives than with (13). Due to this connection we term our method EKLT, event-based KLT.

5 Experiments

To illustrate the high accuracy of our method, we first evaluate it on simulated data (Section 5.1), where we can control scene depth, camera motion, and other model parameters. Then we test our method against four baseline trackers on real data consisting of high-contrast and natural scenes, with challenging effects such as occlusions, parallax and illumination changes (Section 5.2). We also analyze the sensitivity of our method with respect to the number of events used (Section 5.4) and the spatial size of the feature (Section 5.5). The robustness of the method to illumination changes and to low light conditions is also shown (Section 5.6). Additionally, we show that our tracker can operate using frames reconstructed from a set of events [44–46], which have higher dynamic range than those of standard cameras,

thus opening the door to feature tracking in high dynamic range (HDR) scenarios (Section 6.1). Moreover, we quantify the gap between using our method with regular frames from a standard camera and using reconstructed HDR frames from events (Section 6).

For all experiments we use patches $\Delta L(\mathbf{u})$ of 25×25 pixel size (as justified in Section 5.5) and the corresponding events falling within the patches as the features moved on the image plane. On the synthetic datasets, we use the 3D scene model and camera poses to compute the ground truth feature tracks. On the real datasets, we use KLT [35] as ground truth. Since our feature tracks are produced at a higher temporal resolution than the ground truth, interpolating ground truth feature positions may lead to wrong error estimates if the feature trajectory is not linear in between samples. Therefore, we evaluate the error by comparing each ground truth sample with the feature location given by linear interpolation of the two closest feature locations in time and averaging the Euclidean distance between ground truth and the estimated positions.

5.1 Simulated Data. Assessing Tracking Accuracy

By using simulated data we assess the accuracy limits of our feature tracker. To this end, we used the event camera simulator presented in [9] and 3D scenes with different types of texture, objects and occlusions (Fig. 6). The tracker's accuracy can be assessed by how the average feature tracking error evolves over time (Fig. 6(c)); the smaller the error, the better. All features were initialized using the first frame and then tracked until discarded, which happened if they left the field of view or if the registration error (7) exceeded a threshold of 1.6. We define a feature's age as the time elapsed between its initialization and its disposal. The longer the features survive, the more robust the tracker.

The results for simulated datasets are given in Fig. 6 and Table 1. Our method tracks features with a very high accuracy, of about 0.4 pixel error on average, which can be regarded as a lower bound for the tracking error (in noise-free conditions). The remaining error is likely due to the linearization approximation in (3). Note that feature age is just reported for completeness, since simulation time cannot be compared to the physical time of real data (Section 5.2).

Table 1: Average pixel error and average feature age for simulated data.

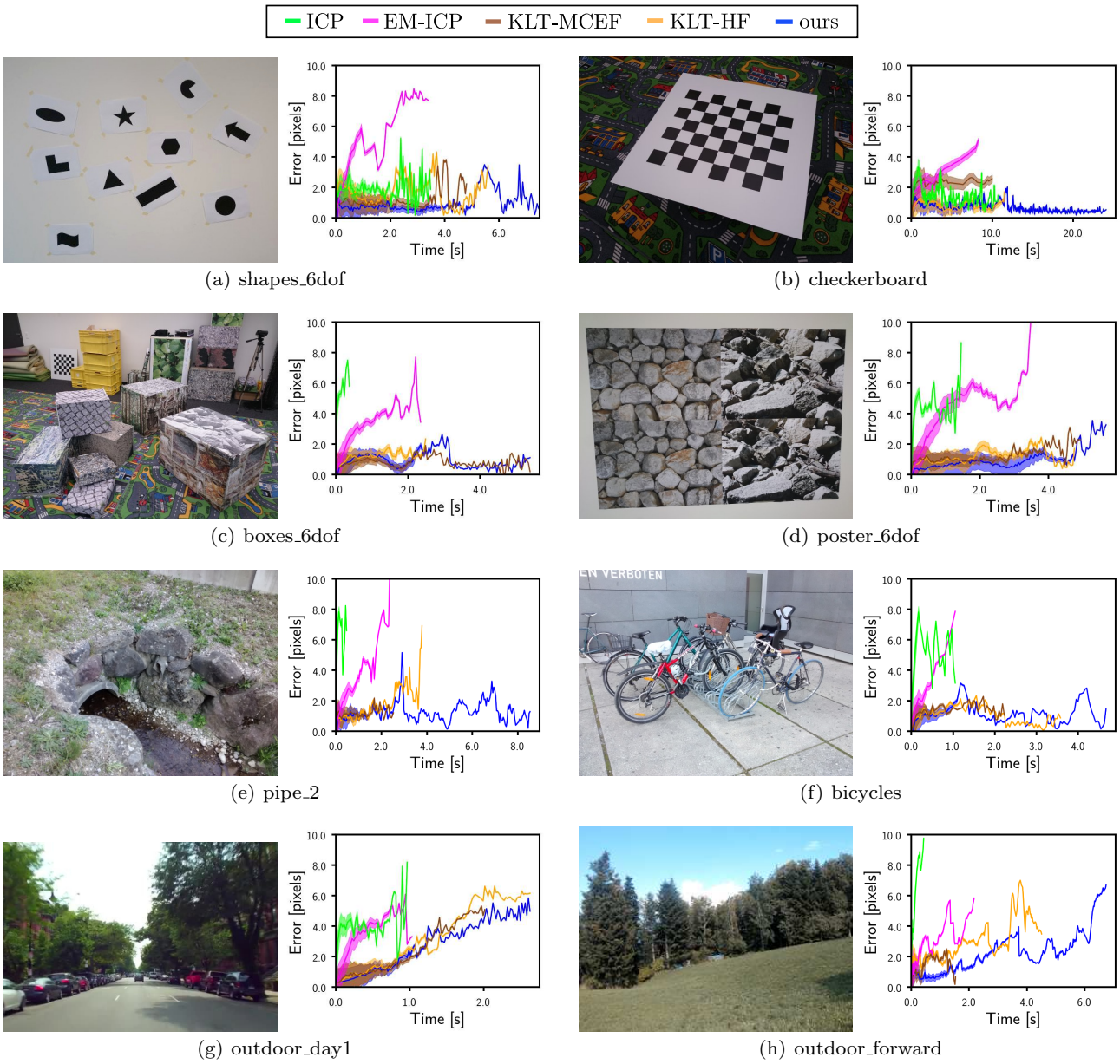
Datasets	**Error [px]**	**Feature age [s]**
sim_april_tags	0.20	1.52
sim_3planes	0.29	0.78
sim_rocks	0.42	1.00
sim_3wall	0.67	0.40

5.2 Real Data

5.2.1 Description of Baseline Methods

We compare the proposed feature tracker against four baselines, which we present next. In all cases, we use the same initial feature locations to compare the tracking results across methods on the same set of features. Thus, the methods mainly differ in two aspects: the way features are represented around the given location (i.e., the feature template), and the way tracking is performed. In all methods, tracking is done with respect to the feature template created at initial time (i.e., the time of the grayscale frame), as proposed in Algorithm 1, rather than with respect to the last frame (i.e., frame-to-frame tracking). Ground truth is provided by Lucas-Kanade tracking (KLT) on the DAVIS [2] frames.

- *Feature Tracking on Canny Point Sets (ICP).* The method in [13] represents features from the grayscale frame using point sets extracted from Canny edges. Tracking is performed by point set registration (ICP) between the feature and the incoming events.
- *Feature Tracking on Motion-compensated Point Sets (EM-ICP).* The method in [14] also represents features as point sets, but they are built from the events by means of motion compensation. Thus, we take the events around the given feature location to build a motion-compensated feature template. Tracking is performed by Expectation-Maximization (EM) and fuzzy ICP between the feature and the incoming events.
- *KLT Feature Tracking on Motion-compensated Event Frames (KLT-MCEF).* In [28], motion-compensated event images were built from the events, the estimated scene depth and the rotational motion provided by an inertial measurement unit (IMU). Then, a standard feature detector and tracker (FAST [25] and KLT [35], respectively) were used on these event images to track the feature's location. Inspired by [28], we build motion-compensated event images by fitting a homography to a temporal window of events [39] (thus, effectively assuming a quasi-planar scene), which avoids the need for an IMU and scene depth



(a) shapes_6dof (b) checkerboard

(c) boxes_6dof (d) poster_6dof

(e) pipe_2 (f) bicycles

(g) outdoor_day1 (h) outdoor_forward

Fig. 7: Feature tracking on all eight datasets reported in Table 2: simple black and white scenes (a)(b), highly textured scenes (c)(d) and natural scenes (e)-(h). Plots on the right of the image show the mean tracking error (center line) and fraction of surviving features (band around the center line) for our method and all four baselines in Table 2. We encourage the reader to watch the accompanying video for a visualization of the feature tracks. Figure best viewed in color.

estimation. The motion-compensated event image provides the feature templates used for tracking with the KLT method.

- *KLT Feature Tracking on Reconstructed Images (KLT-HF).* This is an approach inspired by the combination of recent developments in image reconstruction and classical feature tracking. In this approach, grayscale frames are built at the timestamps of the DAVIS frames using the image reconstruction method in [46]. Frames are reconstructed from all past events, by pixel-wise temporal integration and high-pass filtering (HF), without requiring estimation of the scene depth or the camera ego-motion. Additionally, we apply the Contrast Limited Adaptive Histogram Equalization (CLAHE) from OpenCV to improve the quality of the frames. Tracking is done using the KLT tracker on the reconstructed brightness frames (in

the same way as ground truth is obtained using the DAVIS frames).

5.2.2 Quantitative Comparison of Feature Trackers

The above-mentioned methods were evaluated on several datasets. The sequences used are "shapes_6dof", "checkerboard", "boxes_6dof" and "poster_6dof" from the Event Camera Dataset [9], "pipe_2" and "bicycles" from [29][3], "outdoor_day1" from the Multi-Vehicle Stereo Event Camera Dataset (MVSEC) [10] and "outdoor_forward5" from the UZH-FPV Drone Racing Dataset [47]. The results are reported in Figs. 7-8 and Tables 2-3. Sample feature tracks are visualized in Fig. 9. To take into account the tracking capabilities of KLT we normalize the feature age by the age of corresponding KLT feature track. In addition, to remove any bias from features that are discarded early we report the *track-normalized tracking error* for each dataset. This metric first computes the average tracking error over single tracks and then averages them over all features. More details about the evaluation can be found in our open source feature-tracking evaluation package[4].

The plots in Fig. 7 show the mean tracking error as a function of time (center line). The width of the colored band indicates the proportion of features that survived up to that point in time. The width of the band decreases with time as feature tracks are gradually lost. The wider the band, the more robust the feature tracker. Table 2 reports the tracking error of all compared methods, using the span of the features tracked by our method. This is so in order to compare accuracy before drift in other methods occurs. Table 3 reports the average feature age of the tracks.

Fig. 8 visually summarizes the values on Tables 2 and 3. Overall, our method outperforms all other methods in tracking accuracy. It also provides longer tracks than previous works ICP and EM-ICP, and comparable feature age to the newly desgined baselines KLT-MCEF and KLT-HF.

In simple, black and white scenes (Figs. 7(a) and 7(b)), such as those in [13], our method is, on average, twice as accurate and produces tracks that are twice longer than ICP. Compared to EM-ICP our method is also more accurate and robust. The method KLT-MCEF provides longer tracks, albeit it is not as accurate as our method. For highly textured scenes (Figs. 7(c) and 7(d)), our tracker maintains the accuracy even though many events are generated everywhere in the patch,

Table 2: Comparison of five different feature tracking methods on eight test sequences from real data. Average of the tracking error, normalized by the length of the tracks, for each combination of method and sequence.

Scene	Datasets	Track-normalized error [px]				
		Ours	ICP	EM-ICP	KLT-MCEF	KLT-HF
Black & White	shapes_6dof	**0.80**	1.49	2.31	0.94	2.43
	checkerboard	**1.21**	1.92	2.30	2.30	1.75
High Texture	poster_6dof	**0.64**	2.48	3.10	0.97	1.18
	boxes_6dof	**0.72**	4.59	1.60	0.80	1.24
Natural	bicycles	**0.76**	4.22	1.50	1.26	1.21
	pipe_2	**0.78**	4.90	1.63	1.04	1.06
	outdoor_day1	**0.71**	2.96	2.30	2.00	2.52
	outdoor_forward5	**0.80**	4.15	1.47	1.58	2.36

Table 3: Comparison of five different feature tracking methods on eight test sequences from real data. Relative feature age, normalized by the length of KLT tracks, for each combination of method and sequence.

Scene	Datasets	Relative feature age				
		Ours	ICP	EM-ICP	KLT-MCEF	KLT-HF
Black & White	shapes_6dof	0.54	0.28	0.21	**0.60**	0.53
	checkerboard	0.35	0.13	0.27	**0.45**	0.37
High Texture	poster_6dof	**0.45**	0.04	0.22	0.27	0.37
	boxes_6dof	0.54	0.09	0.27	0.55	**0.69**
Natural	bicycles	0.20	0.09	0.10	0.25	**0.27**
	pipe_2	0.34	0.10	0.25	**0.41**	0.37
	outdoor_day1	0.23	0.07	0.15	0.34	**0.48**
	outdoor_forward5	0.25	0.13	0.16	0.16	**0.30**

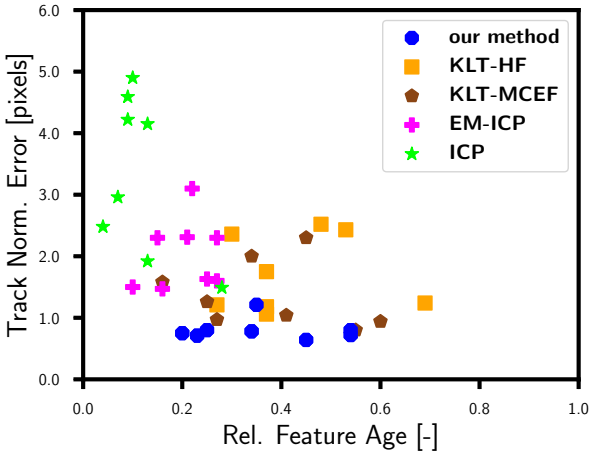


Fig. 8: Visualization of the values on Tables 2 and 3. Normalized Tracking Error and Relative Feature Age for five feature trackers on all eight test sequences. The smaller the error and the longer the feature age, the better.

which leads to significantly high errors in ICP and EM-ICP. Although our method and KLT-HF achieve similar feature age, our method is more accurate. Similarly, on natural scenes, our method is more accurate than the baselines. A more detailed comparison with [13] (ICP)

[3] The datasets are publicly available at: http://rpg.ifi.uzh.ch/direct_event_camera_tracking/

[4] Code can be found here: https://github.com/uzh-rpg/rpg_feature_tracking_analysis

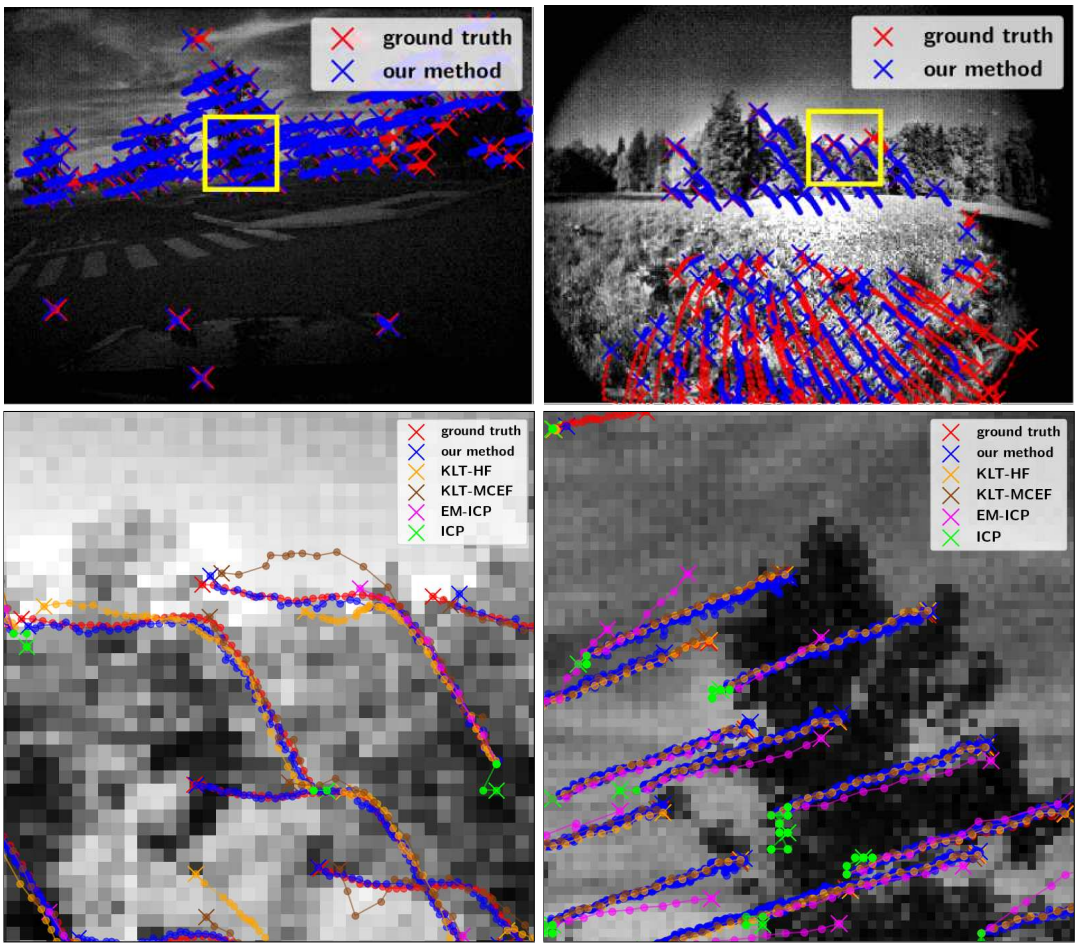


Fig. 9: Visualization of feature tracks. The sequence "outdoor_day1" [10] (top row) depicts the data acquired with a DAVIS camera installed on the windshield of a car, driving through streets in a city. The sequence "outdoor_forward5" (bottom row) shows the data acquired by a camera mounted on a drone, flying through a meadow and a forest. See also the accompanying video.

is further explored in Appendix A.1, where we show that our objective function is better behaved.

On average, high-contrast and high-texture scenes yield longer tracks than natural scenes (width of the band around the center line). In many cases, the average error decreases as time progresses since the features that survive longest are typically those that are most accurately tracked.

The tracking error of our method on real data is larger than that on synthetic data, which is likely due to modeling errors concerning the events, including noise and dynamic effects (such as unequal contrast thresholds for events of different polarity). Nevertheless, our tracker achieves subpixel accuracy and outperforms the baselines in terms of accuracy.

5.3 Computational Performance

Our non-optimized C++ implementation of the proposed approach is able to process about 17000 events per second (on an Intel i7 CPU, with 64 bits, 3.20 GHz, single-threaded), only counting events that fall within the domain of the tracked features. On the considered dataset (shapes_6dof), the real event rate reaches between 54 000–130 000 events per second. Since features can be tracked independently from one another, on the implementation side there is room for improvement using a more distributed, i.e., parallelized, platform.

5.4 Sensitivity with respect to the Number of Events in a Patch

As anticipated in Section 4.4 (Algorithm 1), we adaptively find the optimal number of events N_e integrated

in (2) to create a patch $\Delta L(\mathbf{u})$. Let us show how. As shown in (3), it is clear that $\Delta L(\mathbf{u})$ (thus N_e) depends on the scene texture as well as the motion. First, the larger the amount of texture (i.e., brightness gradients), the more events will be generated by the feature. Second, motion parallel to an edge prevents some events from being generated (Fig. 2).

Fig. 10 shows how the number of accumulated events N_e, which defines the appearance of the patch $\Delta L(\mathbf{u})$, affects the shape of the objective function (7), and, therefore, affects its minimizer. Using too few or too many events does not provide a reliable registration with respect to the frame template, either due to the fact that there is not enough information about the patch appearance conveyed by the events or because the information has been washed out by an excessive integration time. These are the left- and right-most plots in Fig. 10, respectively. Using an intermediate N_e gives an event-brightness patch that captures the underlying scene texture and produces a nicely-shaped objective function with the minimizer at the correct warp and flow parameters.

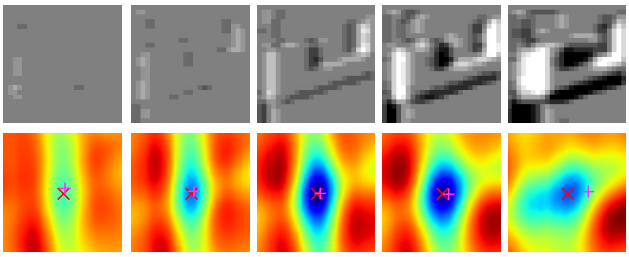
We propose a simple formula to compute N_e based on the the frame, $\hat{L}$, as follows. Stemming from (2), the amount of brightness change over a patch $\mathcal{P}$ is

$$\int_{\mathcal{P}} |\Delta L(\mathbf{u})| d\mathbf{u} = C\, N_e \tag{14}$$

assuming that no events of opposite polarity are triggered at the same pixel during the short integration time $\Delta\tau$. Then, assuming that (3) is a good approximation for the event patch gives $C\, N_e \approx \int_{\mathcal{P}} |\nabla \hat{L}(\mathbf{u}) \cdot \mathbf{v}\Delta\tau| d\mathbf{u}$. Finally, considering an integration time $\Delta\tau \approx 1/\|\mathbf{v}\|$ (so that the events correspond to a displacement of the pattern of $\|\mathbf{v}\|\Delta\tau \approx 1$ pixel) and a threshold in the order of $C \approx 1$ gives

$$N_e \approx \int_{\mathcal{P}} \left| \nabla \hat{L}(\mathbf{u}) \cdot \frac{\mathbf{v}}{\|\mathbf{v}\|} \right| d\mathbf{u}. \tag{15}$$

At each time step, the newly estimated unit vector $\mathbf{v}/\|\mathbf{v}\|$ is used to compute the optimal number of events to be processed. For Fig. 10, this value is approximately the number of events in the center plot. By adapting the number of events our method gains considerable accuracy as is highlighted in Fig. 11 and Table 4. In this experiment we compare the tracking accuracy against using a fixed number of events on "shapes_6dof". Note that the adaptive method used an average of 72 events per update step. We can clearly see that the tracking error is optimal if we use an adaptive number, while the feature age stagnates with higher numbers of events.



(a) $N_e = 10$ (b) 50 (c) 200 (d) 500 (e) 1000

Fig. 10: Effect of varying the number of events N_e accumulated in (2). *Top row*: brightness increment patches $\Delta L(\mathbf{u})$, of size 25×25 pixels. For simplicity, the feature moves horizontally. *Bottom row*: corresponding profiles of the function (7), represented as heat maps, along the x, y translation parameters (± 5 pixels from the minimizer of the function, indicated by a red cross ($\times$)). The magenta plus sign ($+$) indicates the ground truth warp parameters.

Table 4: Tracking error and feature age depending on the number of integrated events per patch evaluated on the *shapes_6dof dataset. The best tracking results are achieved when we adapt the number of events according to* (15) *which yields an average of 72 events per update.*

Number of events	10	30	72*	150	300
Error [px]	1.12	0.91	**0.72**	1.00	1.14
Feature Age [s]	0.86	1.81	1.84	1.94	**2.09**

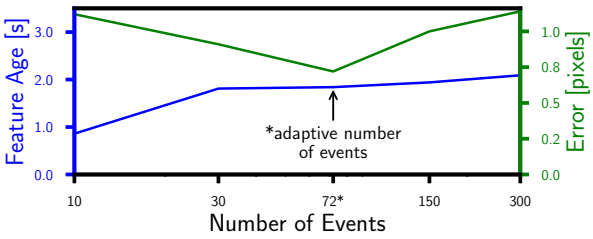
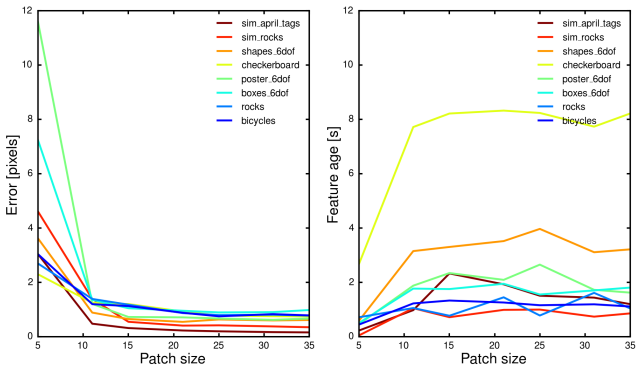


Fig. 11: Visualization of the values in Table 4. The best tracking results are achieved when we adapt the number of events according to (15) which yields an average of 72 events per update.

5.5 Influence of the Patch Size

As anticipated at the beginning of Section 5, we provide a justification of the choice of the patch size used in our method. Tables 5, 6 and Fig. 12 report the dependency of the tracking error and the feature age with respect to the size of the patches used, from 5×5 pixels to 35×35 pixels.



(a) (b)

Fig. 12: Visualization of the values on Tables 5 and 6. Panels (a) and (b) show, respectively, the evolution of the mean tracking error and feature age as a function of the patch size used.

In Tables 5 and 6 we highlighted in bold the best result per row. Better accuracy is achieved for larger patch sizes whereas longer feature tracks are achieved towards medium to smaller patch sizes. We chose a patch size of 25×25 pixels as a compromise between accuracy and robustness (feature age), and performed all other experiments in Section 5 with this value.

Table 5: Tracking error for different datasets and varying patch size (p).

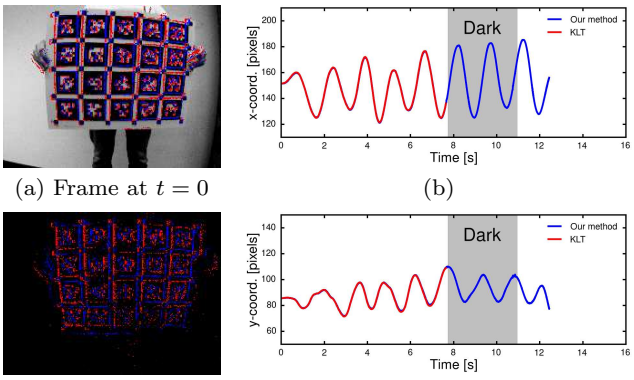
Sequences	**Error [px]**						
	$p = 5$	$p = 11$	$p = 15$	$p = 21$	$p = 25$	$p = 31$	$p = 35$
sim_april_tags	3.04	0.48	0.32	0.23	0.20	0.17	**0.16**
sim_rocks	4.61	1.39	0.55	0.41	0.42	0.38	**0.35**
shapes_6dof	3.62	0.89	0.65	**0.54**	0.64	0.6	0.62
checkerboard	2.30	1.25	1.20	0.93	0.78	**0.75**	**0.75**
poster_6dof	11.59	1.21	0.73	0.71	0.67	**0.62**	0.67
boxes_6dof	7.24	1.36	1.05	0.96	**0.89**	0.90	0.98
pipe_2	2.69	1.39	1.18	0.87	0.80	0.81	**0.77**
bicycles	3.04	1.20	1.13	0.88	**0.75**	0.83	0.78

Table 6: Feature age for different datasets and varying patch size (p).

Sequences	**Feature Age [s]**						
	$p = 5$	$p = 11$	$p = 15$	$p = 21$	$p = 25$	$p = 31$	$p = 35$
sim_april_tags	0.23	0.98	**2.33**	1.93	1.52	1.44	1.20
sim_rocks	0.05	**1.05**	0.72	0.99	1.00	0.74	0.86
shapes_6dof	0.59	3.15	3.31	3.52	**3.97**	3.11	3.21
checkerboard	2.68	7.72	8.21	**8.32**	8.24	7.74	8.22
poster_6dof	0.46	1.88	2.34	2.09	**2.65**	1.73	1.62
boxes_6dof	0.50	1.77	1.76	**1.95**	1.56	1.71	1.81
pipe_2	0.72	1.05	0.77	1.45	0.78	**1.62**	1.04
bicycles	0.44	1.22	**1.33**	1.26	1.16	1.19	1.11

5.6 Feature Tracking in Low Light and with Abrupt Light Changes

To further illustrate the robustness of our tracker, we performed additional experiments in low light and with abrupt changes of illumination, achieved by switching the lights on and off in the room. Results are displayed in Figs. 13, 14 and 15. In these experiments we show that our tracker can extract features from a standard frame and track them robustly through time, even when the light is off, thanks to the very high dynamic range of the event camera. Our method is also able to track after the light has been switched on again. By contrast, KLT [35] loses track immediately after switching the light off because the frames do not have a dynamic range as high as the events. We encourage the reader to watch the accompanying video, which shows the experiment in a better form than still images can convey.



(a) Frame at $t = 0$ (b)

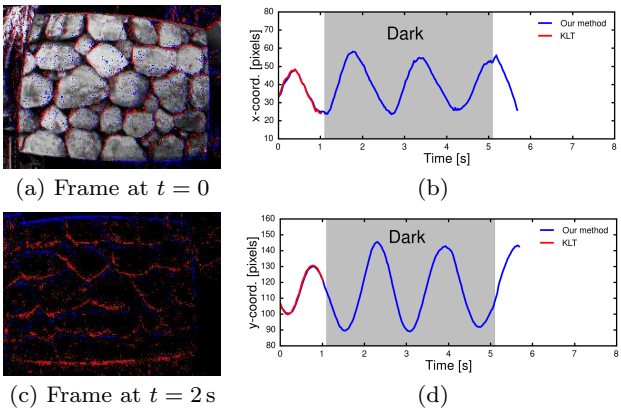
(c) Frame at $t = 9\,\mathrm{s}$ (d)

Fig. 13: Figs. (a) and (b) show the standard frames with the events superimposed, respectively when the light in the room is on or off. Figs. (b) and (d) show the evolution of the x and y coordinates of one feature tracked through time (red: KLT [35] on the frames, blue: our method). In contrast to KLT, our tracker maintains stable feature tracks even in the period when the light is off (marked in gray), and keeps tracking them when the light is on again.

6 Are Standard Camera Frames Needed?

6.1 Tracking using Frames Reconstructed from Events

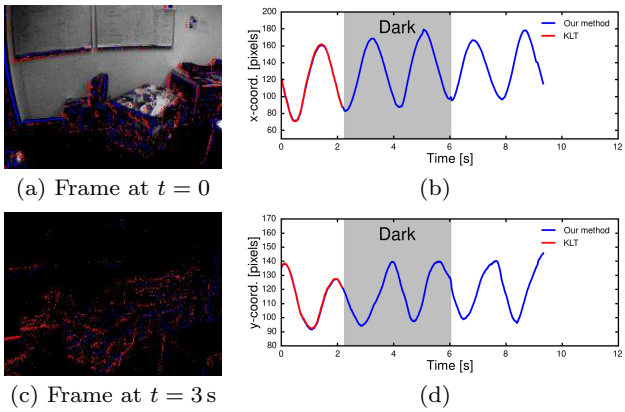
Recent research [44–46, 48, 49] has shown that events can be combined to reconstruct intensity frames that inherit the outstanding properties of event cameras (high dynamic range (HDR) and lack of motion blur). In the next experiment, we show that our tracker can be used



(a) Frame at $t = 0$ (b)

(c) Frame at $t = 2\,\text{s}$ (d)

Fig. 14: Feature tracking in low light and with abrupt illumination changes. Rocks scene. Same notation as in Fig. 13.



(a) Frame at $t = 0$ (b)

(c) Frame at $t = 3\,\text{s}$ (d)

Fig. 15: Feature tracking in low light and with abrupt illumination changes. Office scene. Same notation as in Fig. 13. See the multimedia material.

on such reconstructed images, thus removing the limitations imposed by standard cameras. As an illustration, we focus here on demonstrating feature tracking in HDR scenes (Fig. 16). However, our method could also be used to perform feature tracking during high-speed motions by using motion-blur–free images reconstructed from events.

Standard cameras have a limited brightness dynamic range (60 dB), which often results in under- or over-exposed areas of the sensor in scenes with a high dynamic range (Fig. 16(b)), which in turn can lead to tracking loss. Event cameras, however, have a much larger dynamic range (140 dB) (Fig. 16(b)), thus providing valuable tracking information in those problematic areas. Figs. 16(c)-(d) show qualitatively how our method can exploit HDR intensity images reconstructed from a set of events [46] to produce feature tracks in such difficult conditions. For example, Fig. 16(d) shows

Table 7: Performance of the proposed feature tracker using two different types of frames: the DAVIS frames [2], and frames reconstructed using [46]. The average pixel error and average feature age are reported for all eight test sequences.

Scene	Sequences	Track-norm. error [px]		Rel. feature age	
		DAVIS [2]	HF [46]	DAVIS [2]	HF [46]
Black & White	shapes_6dof	**0.80**	1.51	**0.54**	0.51
	checkerboard	1.21	**1.10**	**0.35**	0.21
High Texture	poster_6dof	**0.64**	0.67	**0.45**	0.23
	boxes_6dof	**0.72**	0.74	**0.54**	0.41
Natural	bicycles	0.76	**0.57**	**0.20**	0.16
	pipe_2	0.78	**0.55**	**0.34**	0.14
	outdoor_day_1	**0.71**	0.77	**0.23**	0.18
	outdoor_forward_5	**0.80**	1.14	**0.25**	0.17

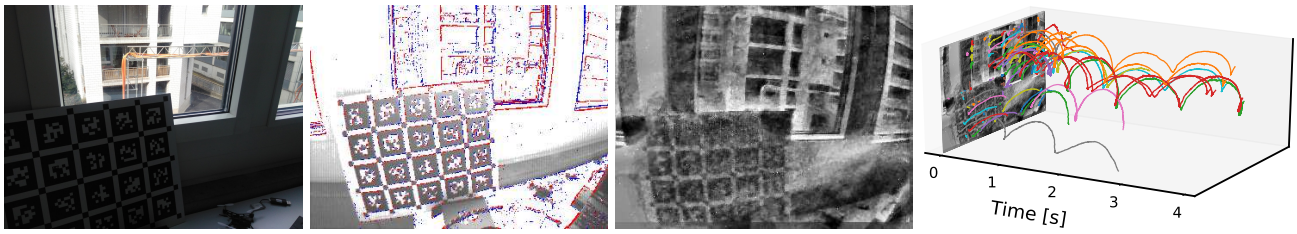
that some feature tracks were initialized in originally overexposed areas, such as the top right of the image (Fig. 16). A quantitative analysis of the difference between using DAVIS frames versus using intensity-reconstructed frames is provided in Section 6.

6.2 Quantitative Evaluation

In the experiment on Fig. 16, we showed that the proposed method is able to track even when the grayscale frame used is not produced by a frame-based sensor, but rather reconstructed from events. In this section, we analyze the dependency of our method with respect to the type of grayscale frame used. We compare the performance of our method using frames from the DAVIS camera, and frames reconstructed from the events using a state-of-the-art image reconstruction method.

More specifically, we perform image reconstruction using [46] at the time of a frame acquired by the DAVIS camera. We detect features on the DAVIS frame (e.g., Harris corners), and use them to initialize our feature tracker on the DAVIS frame as well as on the reconstructed frame, so that the comparison on these two different frames is carried out using the same tracked features. Fig. 17 and Table 7 summarize the results of the experiments carried out on eight test sequences (the same ones as in Section 5.2).

In general, we observe that the tracking results are very similar in terms of accuracy (about 1 pixel error), and the biggest differences occur in terms of feature age. In terms of feature age, the best results are obtained when our method uses the DAVIS frames. However, the tracking results on the reconstructed frames are also good, and they are solely based on events, that is, the proposed method (Algorithm 1) does not require a frame-based sensor co-located with the event-based sensor. Additionally, in challenging scenarios (HDR or high-speed) the DAVIS frames are impaired, whereas



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

Fig. 16: Our feature tracker is not limited to intensity frames from a real camera. In this example, we use an intensity image reconstructed from a stream of events [46] in a scene with high dynamic range (a). The DAVIS frame, shown in (b) with events overlaid on top, cannot capture the full dynamic range of the scene. By contrast, the reconstructed image in (c) captures the full dynamic range of the scene. Our tracker (d) can successfully use this image to produce accurate feature tracks everywhere, including the badly exposed areas of (b).

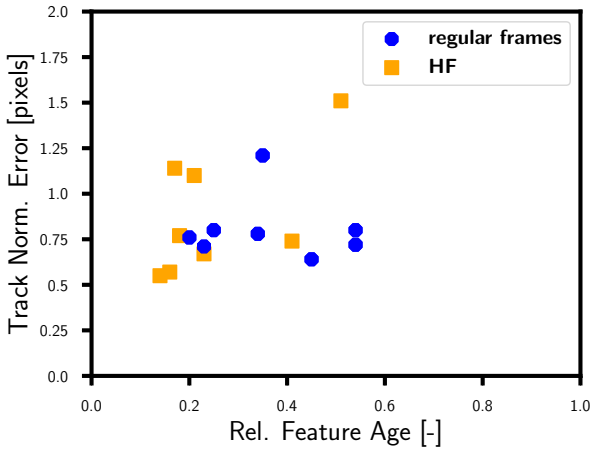


Fig. 17: Visualization of the values on Table 7. Accuracy and feature age for all eight test sequences and two different types of frames used in our method: regular frames from the DAVIS camera [2] and reconstructed brightness frames using [46]. The smaller the error and the longer the feature age, the better.

the reconstructed frames may be the only viable option since they inherit the lack of motion blur and HDR characteristics from the events.

Note that the proposed method (Algorithm 1) only requires a limited number of frames since features can be tracked for several seconds. This complements the computationally-demanding task of image reconstruction. Thus, it is sensible to reconstruct the frames at low rates (∼1 Hz) (or on demand) to initialize features and then track asynchronously (i.e., at high rate) with the events, as shown in the accompanying video.

7 Discussion

While our method advances event-based feature tracking in natural scenes, there remain directions for future research. For example, the generative model we use to predict events is an approximation that does not account for severe dynamic effects and noise. In addition, our method assumes uniform optical flow in the vicinity of features. This assumption breaks down at occlusions and at objects undergoing large flow distortions, such as motion along the camera's optical axis. Nevertheless, as shown in the experiments, many features in a variety of scenes and motions do not suffer from such effects, and are therefore tracked well (with sub-pixel accuracy). Finally, we demonstrated the method using a Euclidean warp since it was more stable than more complex warping models (e.g., affine). Future research includes ways to make the method more robust to sensor noise and to use more accurate warping models.

8 Conclusion

We presented a method that leverages the complementarity of event cameras and standard cameras to track visual features with low-latency. Our method extracts features on frames and subsequently tracks them asynchronously using events. To achieve this, we presented the first method that relates events directly to pixel intensities in frames via a generative event model. We thoroughly evaluated the method on a variety of scenes and against four baselines, showing that it produces feature tracks that are more accurate (subpixel accuracy) than the state-of-the-art. We also investigated the need for frames from the standard camera and concluded that they can be replaced with a similar signal: frames built from events by means of state-of-the-art image reconstruction methods. This removes the need

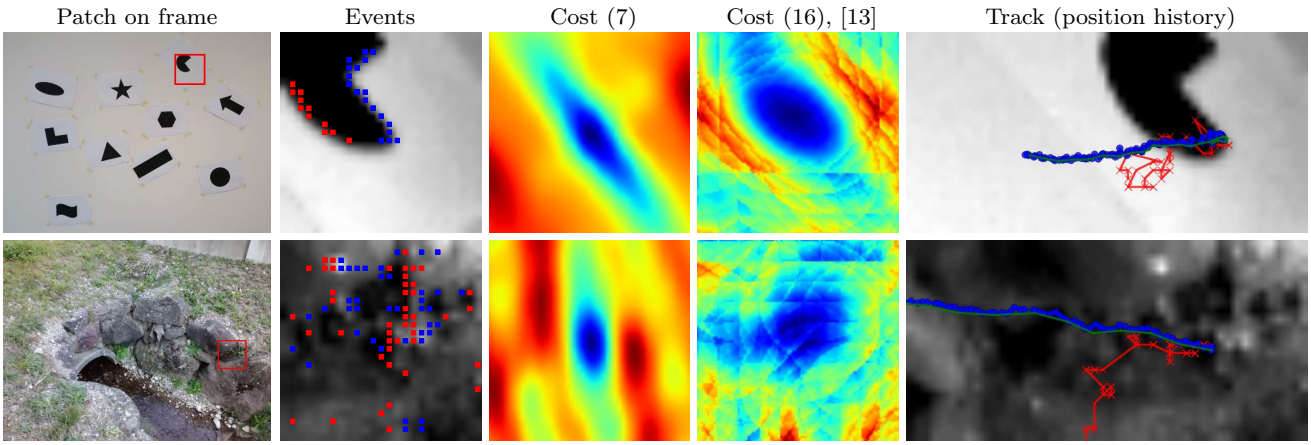


Fig. 18: Our cost function (7) is better behaved (smoother and with fewer local minima) than that in [13], yielding a better tracking (last column). The first two columns show the datasets and feature patches selected, with intensity (grayscale) and events (red and blue). The third and fourth columns compare the cost profiles of (7) and (16) for varying translation parameters in x and y directions (± 5 pixel around the best estimate from the tracker). The point-set–based cost used in [13] shows many local minima for more textured scenes (second row) which is not the case of our method. The last column shows the position history of the features (green is ground truth, red is [13] (ICP) and blue is our method).

for having a standard camera co-located with the event camera. We believe this work will open the door to unlock the outstanding properties of event cameras on various computer vision tasks that rely on accurate feature tracking.

A Appendix

A.1 Objective Function Comparison against ICP-based Method [13]

As mentioned in Section 4, one of the advantages of our method is that data association between events and the tracked feature is implicitly established by the pixel-to-pixel correspondence of the compared patches (2) and (3). This means that we do not have to explicitly estimate it, as was done in [13, 14], which saves computational resources and prevents false associations that would yield bad tracking behavior. To illustrate this advantage, we compare the cost function profiles of our method and [13] (ICP), which minimizes the alignment error (Euclidean distance) between two 2D point sets: $\{\mathbf{p}_i\}$ from the events (data) and $\{\mathbf{m}_j\}$ from the Canny edges (model),

$$\{\mathtt{R}, \mathbf{t}\} = \arg\min_{\mathtt{R},\mathbf{t}} \sum_{(\mathbf{p}_i, \mathbf{m}_i) \in \text{Matches}} b_i \left\| \mathtt{R}\mathbf{p}_i + \mathbf{t} - \mathbf{m}_i \right\|^2 . \qquad (16)$$

Here, $\mathtt{R}$ and $\mathbf{t}$ are the alignment parameters and b_i are weights. At each step, the association between events and model points is done by assigning each $\mathbf{p}_i$ to the closest point $\mathbf{m}_j$ and rejecting matches which are too far apart (> 3 pixel). By varying the parameter $\mathbf{t}$ around the estimated value while fixing $\mathtt{R}$ we obtain a slice of the cost function profile. The resulting cost function profiles for our method (7) and (16) are shown in Fig. 18.

For simple black and white scenes (first row of Fig. 18), all events generated belong to strong edges. In contrast, for more complex, highly-textured scenes (second row), events are generated more uniformly in the patch. Our method clearly shows a convex cost function in both situations. In contrast, [13] exhibits several local minima and very broad basins of attraction, making exact localization of the optimal registration parameters challenging. The broadness of the basin of attraction, together with the multitude of local minima can be explained by the fact that data association changes for each alignment parameter. This means that there are several alignment parameters which may lead to partial overlapping of the point-clouds resulting in a suboptimal solution.

To show how non-smooth cost profiles affect tracking performance, we show the feature tracks in the last column of Fig. 18. The ground truth derived from KLT is marked in green. Our tracker (in blue) is able to follow the ground truth with high accuracy. On the other hand [13] (in red) exhibits jumping behavior leading to early divergence from ground truth.

Acknowledgements This work was supported by the DARPA FLA program, the Swiss National Center of Competence Research Robotics, through the Swiss National Science Foundation, and the SNSF-ERC starting grant.

References

1. 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.
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.

3. Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-based vision: A survey. *arXiv e-prints*, abs/1904.08405, 2019.
4. Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using SIFT features and mean shift. *Comput. Vis. Image. Und.*, 113(3):345–352, 2009.
5. Georg Klein and David Murray. Parallel tracking and mapping on a camera phone. In *IEEE ACM Int. Sym. Mixed and Augmented Reality (ISMAR)*, 2009.
6. Christian Forster, Zichao Zhang, Michael Gassner, Manuel Werlberger, and Davide Scaramuzza. SVO: Semidirect visual odometry for monocular and multicamera systems. *IEEE Trans. Robot.*, 33(2):249–265, 2017.
7. Raúl Mur-Artal, José M. M. Montiel, and Juan D. Tardós. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. Robot.*, 31(5):1147–1163, 2015.
8. Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate SLAM? combining events, images, and IMU for robust visual SLAM in HDR and high speed scenarios. *IEEE Robot. Autom. Lett.*, 3(2):994–1001, April 2018.
9. 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*, 36(2):142–149, 2017.
10. Alex Zihao Zhu, Dinesh Thakur, Tolga Ozaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3D perception. *IEEE Robot. Autom. Lett.*, 3(3):2032–2039, July 2018.
11. Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. Asynchronous, photometric feature tracking using events and frames. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 766–781, 2018.
12. Elias Mueggler, Basil Huber, and Davide Scaramuzza. Event-based, 6-DOF pose tracking for high-speed maneuvers. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, pages 2761–2768, 2014. . Event camera animation: https://youtu.be/LauQ6LWTkxM?t=25.
13. 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, 2016.
14. Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based feature tracking with probabilistic data association. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4465–4470, 2017.
15. Zhenjiang Ni, Aude Bolopion, Joel Agnus, Ryad Benosman, and Stéphane Régnier. Asynchronous event-based visual shape tracking for stable haptic feedback in microrobotics. *IEEE Trans. Robot.*, 28(5):1081–1089, 2012.
16. Zhenjiang Ni, Sio-Hoï Ieng, Christoph Posch, Stéphane Régnier, and Ryad Benosman. Visual tracking using neuromorphic asynchronous event-based cameras. *Neural Computation*, 27(4):925–953, 2015.
17. 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.
18. Xavier Clady, Sio-Hoi Ieng, and Ryad Benosman. Asynchronous event-based corner detection and matching. *Neural Netw.*, 66:91–106, 2015.
19. 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)*, 2016.
20. Xavier Clady, Jean-Matthieu Maro, Sébastien Barré, and Ryad B. Benosman. A motion-based feature for event-based pattern recognition. *Front. Neurosci.*, 10, January 2017.
21. 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.
22. Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast event-based corner detection. In *British Mach. Vis. Conf. (BMVC)*, 2017.
23. Ignacio Alzugaray and Margarita Chli. Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robot. Autom. Lett.*, 3(4):3177–3184, October 2018.
24. Chris Harris and Mike Stephens. A combined corner and edge detector. In *Proc. Fourth Alvey Vision Conf.*, volume 15, pages 147–151, 1988.
25. Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *Eur. Conf. Comput. Vis. (ECCV)*, pages 430–443, 2006.
26. Rizwan Chaudhry, Avinash Ravichandran, Gregory Hager, and Rene Vidal. Histograms of oriented optical flow and Binet-Cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 1932–1939, 2009.
27. Paul J. Besl and Neil D. McKay. A method for registration of 3-D shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.
28. Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In *British Mach. Vis. Conf. (BMVC)*, 2017.
29. Guillermo Gallego, Jon E. A. Lund, Elias Mueggler, Henri Rebecq, Tobi Delbruck, and Davide Scaramuzza. Event-based, 6-DOF camera tracking from photometric depth maps. *IEEE Trans. Pattern Anal. Mach. Intell.*, 40(10):2402–2412, October 2018.
30. Guillermo Gallego, Christian Forster, Elias Mueggler, and Davide Scaramuzza. Event-based camera pose tracking using a generative event model. arXiv:1510.01972, 2015.
31. Francisco Barranco, Ching L. Teo, Cornelia Fermuller, and Yiannis Aloimonos. Contour detection and characterization for asynchronous event sensors. In *Int. Conf. Comput. Vis. (ICCV)*, 2015.
32. Jürgen Kogler, Christoph Sulzbachner, Martin Humenberger, and Florian Eibensteiner. Address-event based stereo vision with bio-inspired silicon retina imagers. In *Advances in Theory and Applications of Stereo Vision*, pages 165–188. InTech, 2011.
33. Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. Asynchronous frameless event-based optical flow. *Neural Netw.*, 27:32–37, 2012.
34. Samuel Bryner, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Event-based, direct camera tracking from a photometric 3D map using nonlinear optimization. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

35. Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. Artificial Intell. (IJCAI)*, pages 674–679, 1981.
36. Georgios D. Evangelidis and Emmanouil Z. Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 30(10):1858–1865, October 2008.
37. Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.
38. Guillermo Gallego and Davide Scaramuzza. Accurate angular velocity estimation with an event camera. *IEEE Robot. Autom. Lett.*, 2(2):632–639, 2017.
39. Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 3867–3876, 2018.
40. Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. EMVS: Event-based multi-view stereo—3D reconstruction with an event camera in real-time. *Int. J. Comput. Vis.*, 126(12):1394–1414, December 2018.
41. Ana I. Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. Event-based vision meets deep learning on steering prediction for self-driving cars. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 5419–5427, 2018.
42. Patrick Bardow, Andrew J. Davison, and Stefan Leutenegger. Simultaneous optical flow and intensity estimation from an event camera. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, pages 884–892, 2016.
43. Simon Baker and Iain Matthews. Lucas-kanade 20 years on: A unifying framework. *Int. J. Comput. Vis.*, 56(3):221–255, 2004.
44. Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew J. Davison. Simultaneous mosaicing and tracking with an event camera. In *British Mach. Vis. Conf. (BMVC)*, 2014.
45. 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(2):593–600, 2017.
46. Cedric Scheerlinck, Nick Barnes, and Robert Mahony. Continuous-time intensity estimation using event cameras. In *Asian Conf. Comput. Vis. (ACCV)*, 2018.
47. Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are we ready for autonomous drone racing? the UZH-FPV Drone Racing Dataset. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.
48. Christian Reinbacher, Gottfried Graber, and Thomas Pock. Real-time intensity-image reconstruction for event cameras using manifold regularisation. In *British Mach. Vis. Conf. (BMVC)*, 2016.
49. Gottfried Munda, Christian Reinbacher, and Thomas Pock. Real-time intensity-image reconstruction for event cameras using manifold regularisation. *Int. J. Comput. Vis.*, 126(12):1381–1393, July 2018.