Speed Invariant Time Surface for Learning to Detect Corner Points with Event-Based Cameras

Jacques Manderscheid, Amos Sironi, Nicolas Bourdis, Davide Migliore, Vincent Lepetit

Speed Invariant Time Surface for Learning to Detect Corner Points with Event-Based Cameras

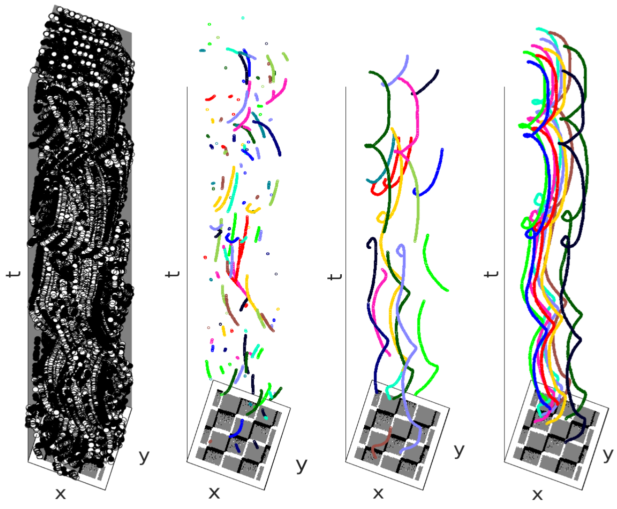
Jacques Manderscheid[1], Amos Sironi[1], Nicolas Bourdis[1], Davide Migliore[1] and Vincent Lepetit[2]
[1]Prophesee, Paris, France, [2]University of Bordeaux, Bordeaux, France
{jmanderscheid, asironi, nbourdis, dmigliore}@prophesee.ai, vincent.lepetit@u-bordeaux.fr

Abstract

We propose a learning approach to corner detection for event-based cameras that is stable even under fast and abrupt motions. Event-based cameras offer high temporal resolution, power efficiency, and high dynamic range. However, the properties of event-based data are very different compared to standard intensity images, and simple extensions of corner detection methods designed for these images do not perform well on event-based data. We first introduce an efficient way to compute a time surface that is invariant to the speed of the objects. We then show that we can train a Random Forest to recognize events generated by a moving corner from our time surface. Random Forests are also extremely efficient, and therefore a good choice to deal with the high capture frequency of event-based cameras—our implementation processes up to 1.6Mev/s on a single CPU. Thanks to our time surface formulation and this learning approach, our method is significantly more robust to abrupt changes of direction of the corners compared to previous ones. Our method also naturally assigns a confidence score for the corners, which can be useful for postprocessing. Moreover, we introduce a high-resolution dataset suitable for quantitative evaluation and comparison of corner detection methods for event-based cameras. We call our approach SILC, for Speed Invariant Learned Corners, and compare it to the state-of-the-art with extensive experiments, showing better performance.

1. Introduction

By capturing very efficiently local illuminance changes ('events'), event-based cameras [32, 46, 53] open the door to novel very fast and low-power computer vision algorithms able to deal with large dynamic ranges [6, 27, 48, 5]. However, because the events are created asynchronously, as shown in Fig. 1 (a), novel algorithms have to be developed to perform fundamental computer vision tasks that are typically performed on regular frame images.



(a) Events **(b) evFAST [39]** **(c) evHarris [61]** **(d) Ours**

Figure 1: **(a)** Stream of events generated by an event-based camera moving in front of a checkerboard pattern. Black dots represents events with a negative polarity, white dots events with a positive polarity. **(b-c)** Standard event-based corner detectors [39, 61] are not robust to direction changes of the camera, and the corners cannot be reliably tracked over time without a very complex tracking scheme. **(d)** By training a classifier to detect corners from event-based data, our method can reliably detect corners under even abrupt changes of direction. A simple nearest neighbor tracker produces continuous trajectories of the corners over time.

One of the main fundamental tasks is feature point detection, which is important for applications with very strong dynamics such as UAV navigation, where motion blur makes classical frame-based approaches less robust, or visual odometry in High Dynamic Range (HDR) conditions, among others. Inspired by the vast literature on frame-based feature point detection, some works adapted frame-based corner detector to event-based data. Typically, a local spatial descriptor is built around an event, for example by cumulating events in a given time window [61], or by considering the times of arrival of the events [39]. Then, a classical test, such as [26] can be applied to this 2D spatial neighborhood. However, the resulting detectors do not take into consideration the specific characteristics of event-based data, such as different noise patterns, responses to changes of direction, illumination changes, etc. Even if efforts have been made in order to design better tests for event-based cameras [39, 4], hand-crafted detectors remain unstable and corners can not be reliably detected over time Fig. 1 (b-c).

In this paper, we propose a learning approach to event-based feature detection. We train a classifier to label individual events as generated by a moving feature point or not. The main advantage of taking a learning approach is to obtain more stable corners: As shown in Fig. 1, a typical error made by previous detectors is that they are sensitive to changes of the apparent motion of the feature points. This is because corners in event-based cameras are not invariant under changes of direction, by contrast to corners in intensity images. Previous detectors also often erroneously detect points along edges because of noisy events, while such points cannot be detected in a stable way. Learning makes the detection more robust to motion changes and noise, without having to manually design an *ad hoc* method.

Our classification approach relies on a novel formulation of the Time Surface [7, 39], which is another contribution of this work. The Time Surface is a representation that accumulates the information from events over time, and is a common tool used in event-based vision, including to detect corner points. In our work, we also use a Time Surface as input to the classifier, but we show how to efficiently create a Time Surface that is invariant to the objects' speed. Previous work [3] already introduced a method for computing a time surface invariant to speed, however it is still too slow to compute, and incompatible with the high frequency of event-based cameras. The invariance to speed of our Time Surface is important both to achieve classification performance and also to keep the classifier small, which makes computation fast.

One critical aspect of our learning-based approach is indeed that classification must be performed extremely fast, otherwise we would lose the advantage of the capture efficiency of event-based cameras. We therefore chose to use a Random Forest, as Random Forests are very efficient without having to use a GPU (unlike Deep Networks), which would be counter-productive since we target low-power applications. In fact, parallelizing computation as done by GPUs is not well adapted to the sparse and asynchronous nature of the events. Our current implementation processes up to $1.6 \cdot 10^6$ events per second on a single CPU.

To evaluate the quality of our detector, we also release a new high-resolution benchmark dataset. We propose a metric which is independent of ground truth keypoints extracted from gray level images, which are often used but would introduce a strong bias in the evaluation. Specifically, we compare our approach to different detectors in combination with a simple nearest neighbor based tracking, showing that, thanks to the temporal continuity of the events, a very simple tracking rule can lead to state-of-the-art results.

2. Event-based cameras

Standard frame-based cameras capture visual information by acquiring snapshots of the observed scene at a fixed rate. This might result in motion blur for highly dynamic scenes and in redundant data generation for static ones. By contrast, event-based cameras [32, 46, 53], a relatively recent type of cameras, adaptively record information from a scene, depending on the content. More precisely, each pixel in an event-based sensor is independent from the rest of the pixel array. When a change in the log-luminosity intensity at a location surpasses a threshold, the corresponding pixel emits an *event*. An event contains the 2D location of the pixel, the timestamp of the observed change and its polarity, *i.e.* a binary variable indicating whether the luminosity increased or decreased. Thanks to these characteristics, event-based cameras have temporal resolution in the order of the microsecond, high dynamic range, and are well suited for low-power applications.

Some types of event-based cameras also provide gray-level measurements together with the change detection events. The DAVIS [10] sensor for example, is a 240×180 sensor able to output intensity frames at standard frame rates. The ATIS sensor [46] instead provides asynchronous intensity measurements in form of time differences at the same temporal resolution of the events. Our method can be applied to any type of event-based cameras and does not require gray-level information at run time. Intensity information has been used in this work only to build the dataset used to train our classifier, as explained in Section 4.4.

Finally, we note that the spatial resolution of event-based cameras, still limited to low resolution, is likely to soon reach standard resolutions of frame-based cameras [58]. As a consequence, event-based algorithms will have to process constantly increasing event rates. In this work, we use and release the first event-based dataset acquired with a HVGA (480×360) sensor, showing that our algorithm is suited to real time and high data rate applications.

3. Related work

In this section, we review related work on event-based feature detection, machine learning approaches applied to event-based cameras and to feature point detection.

Event-based Features Detection Event-based feature detection methods can be divided into two categories. The first class of methods [14, 43, 29] aims at detecting a particular pattern in the stream of events by applying local template matching rules. For example, a blob detector [29] can be obtained by using a gaussian correlation kernel. When enough events are received in the receptive field of the kernel, a feature event is generated. The method is generalized to generic shapes, like corners or T-junctions. In [14] instead, corners are identified as intersection of local velocity planes, which are obtained by fitting a plane on the spatio-temporal distribution of events. This class of methods is very efficient, but it is sensitive to noise and requires careful tuning of the model parameters.

The second class of methods instead, relies on the adaptation of classical frame-based corner detectors to event-based data. The basic principle behind these approaches is to build a local representation around an event and then apply classical corner tests on it. For example, in [61] the Harris corner detector is applied to images obtained by cumulating a fixed number of events in a local spatial window. The Surface of Active Events (or Time Surface, presented in Sec. 4.2) is a common representation used in event-based vision [7, 39, 65, 55]. It has been used in [39] for corner detection, where the FAST algorithm [50] is adapted to the pattern of Time Surface of a moving corner. In [4] this formulation has been extended to corners with obtuse angles. These methods lose accuracy mostly in case of changes of feature point motion direction, because event-based data do not share the same characteristics as standard intensity images. In particular, the corner appearance in a stream of events depends on its motion, as shown in Fig. 2. As a consequence, the constant appearance assumption made in frame-based vision does not hold in the event-based case.

To overcome this problem, some works rely on gray level images [60, 28, 34, 23]. In these approaches, feature points are detected in the images using standard techniques, and then tracked in between by using the event stream. These methods have the same limitation as frame-based approaches, namely they lose accuracy in presence of motion blur, HDR situations, and increase data rate and the power consumption. Moreover, they require either an event-based sensors able to acquire graylevel information, or a careful synchronization of event-based and frame-based sensors. By contrast, our method requires at run time only a single and purely event-based sensor. We claim that the time information carried by the events is sufficient to reliably detect feature points, without the need of intensity information. We propose to use a machine learning approach to learn the appearance of the features directly from the input data.

Finally, many works focus on the problem of event-based feature tracking [20, 42, 28, 24, 64, 3]. Our method can be combined with any of these tracking algorithms. However, thanks to the high temporal resolution of the event-based cameras, given our stable and efficient feature detector, the problem of data association is made much simpler and a simple nearest neighbor matching rule is sufficient in our experiments to obtain accurate results.

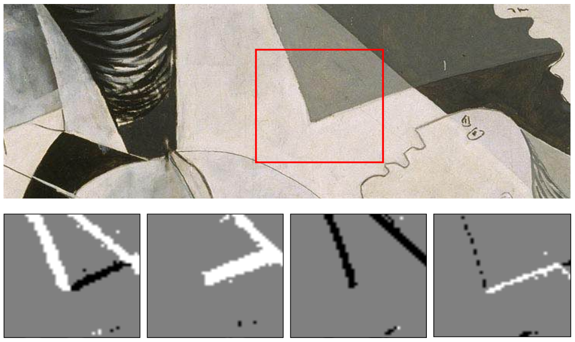


Figure 2: **(Top)** In a classical frame-based camera, the appearance of a corner, such as the one in the red square, is invariant under camera motions. **(Bottom)** In the case of an event-based camera, instead, the same corner can generate vert different pattern of the events depending on its motion. The four panels show 40ms of events generated by the corner on the top while rotating the camera at different speeds.

Learning from Events Machine learning approaches for event-based cameras can also be divided in two categories. In the first category [31, 45, 15, 55, 35, 36, 65, 12], the events from the camera are accumulated for a given period of time, or for a fixed number of events, to build a dense representation. This representation can then be effectively processed by standard Computer Vision techniques, like Convolutional Neural Networks (CNNs).

In [36] for example, the input events are summed into histograms and the obtained images are used to predict the steering angle of a car using a CNN. Similarly, in [65], histograms and time surfaces are used to predict the optical flow. In [15, 47, 55], dense descriptors are built by comparing and averaging the timings of the events in local windows. The descriptors are then passed to a classifier, such as an SVM or a neural network, for object recognition or other vision tasks. Even if these approaches reach good performance and are efficient, they increase the latency of the system by requiring an artificial accumulation time.

A second class of methods avoids this limitation by processing the input event-by-event [25, 33, 44, 41, 30]. The most common model used in event-based cameras are Spiking Neural Networks [44, 38, 8, 54, 37, 9, 51, 13], which have been proven to reach good accuracy for simple classification tasks. However, these approaches are difficult to scale because of the large event rate of the event-based cameras. Hardware implementations and neuromorphic architectures [52, 22, 1, 17] have been proposed to overcome this bottleneck. However, these architecture are not large enough to support large networks and high-resolution event-based cameras as input.

By contrast our classifier is applied event-by-event, keeping the original time resolution of the camera, while running on a standard CPU.

Learning Frame-based Keypoint Detection. A number of works in frame-based computer vision focus on the problem of learning feature points, as we do here for event-based computer vision. One of the first approaches using machine learning for corner detection is FAST [50], where a decision tree is used to generalize brightness tests on contiguous circular segments introduced in [49]. The motivation for using learning in FAST was to speed up the detection by early rejection of non-corner points. Other authors [57, 19] also showed how to learn a fast approximation of existing detectors. But other authors follow this direction by training efficient ensembles of decision trees. For example, in [59], the WaldBoost algorithm [56] is used to learn detectors for specific tasks.

More recent works rely on Deep Learning models [62, 2, 63]. For example, in [62], a regressor is trained to detect stable points under large illumination changes. In [18], a fully-convolutional model is trained with self-supervision to jointly predict interest points and descriptors.

4. Method

In this section, we first introduce our speed invariant formulation of the time surface, then we formalize the problem of learning a feature detector from the output of an event-based camera. Finally, we explain how to create a dataset for training a corner events detector. An overview of our method is given in Fig. 5.

4.1. Asynchronous Event-based Representation of a Visual Scene

As described in Section 2, the output of an event-based camera is an asynchronous stream of events $\{\mathbf{e}_i\}_{i\in\mathbb{N}}$. Each event $\mathbf{e}_i$ represents a contrast change at a given pixel and at a given time and can be formalized as

$$\mathbf{e}_i = (x_i, y_i, t_i, p_i) \;, \tag{1}$$

where (x_i, y_i) are the pixel coordinates of the event, $t_i \geq 0$ is the timestamp at which the event was generated, and $p_i \in \{-1, 1\}$ is the polarity of the event, with -1 and $+1$ meaning respectively that the luminosity decreased or increased at that pixel.

Given an input event $\mathbf{e}_i$, event-based corner detectors typically rely on a spatio-temporal neighborhood of events around $\mathbf{e}_i$ to build a local descriptor. The descriptor is then used to test the presence of a corner. In this work, we consider as in [39] the Time Surface [7] as local descriptor. However, we show how the variance of the standard Time Surface formulation is not suited for stable corner detections. We therefore introduce a normalization scheme to make the time surface invariant to speed. Moreover, we adopt a machine learning approach and use the invariant time surface as input to a classifier trained to discriminate corner events.

4.2. Speed Invariant Time Surface

A common representation used in event-based vision is the *Surface of Active Events* [7], also referred as *Time Surface* [30]. The Time Surface T at a pixel (x, y) and polarity p is defined as

$$T(x, y, p) \leftarrow t \;, \tag{2}$$

where t is the time of the last event with polarity p occurred at pixel (x, y). The Time Surface, besides being very efficient to compute, has been proved to be highly discriminative for several tasks [39, 55, 65]. However, we noticed that local patches of the Time Surface can have a very large variability. In fact, depending on the speed, the direction and the contrast of the corners, the aspect of the time surface can vary significantly. To keep the classification model compact and efficient, it is thus important to introduce some normalization of its input.

Inspired by [39], we notice that only relative timings, and not absolute ones, are relevant to determine the presence of a corner. We could therefore obtain invariance by normalizing the times of the events according to their local speed. However, normalization techniques suitable for intensity images cannot be applied to the time surface, as explained in [3]. [3] proposed to sort the times of the events in a local patch. However, sorting a local patch at every incoming event is still too expensive, and requires to store multiple time stamps per pixel.

Instead, we show how to obtain efficiently a Time Surface that is independent of the speed, by keeping a single value for each pixel location (x, y) and every polarity. We call this novel formulation *Speed Invariant Time Surface*.

The intuition for our new formulation goes as follows: Imagine the contour of an object moving in one direction. We would like to capture the 'wave' of events generated by this contour, but this wave should have the same profile whatever the speed of the contour. When a new event arrives, we store a large value at its pixel location in the Time Surface, and decrease the values for the surrounding locations. In this way, we progressively reduce the values for the events previously generated by the moving contour. Since we decrease the values by a constant factor, the slope of the 'wave' is independent of contour's speed.

More precisely, given a parameter r, we initialize the Speed Invariant Time Surface $S(x, y, p)$ to 0 for every pixel location (x, y) and every polarity p. Then, for every incoming event (x, y, p), we consider all pixel locations (x', y') in its neighborhood of size $(2r + 1) \times (2r + 1)$. If $S(x', y', p)$ is larger than $S(x, y, p)$, then we subtract 1 to $S(x', y', p)$. Finally, we set $S(x, y, p)$ to $(2r + 1)^2$. The pseudo-code of the algorithm is given in Algorithm 1.

Fig. 3 illustrates the application of this algorithm to several edges moving from left to right, each edge moving at a different speed. For the standard Time Surface, each edge

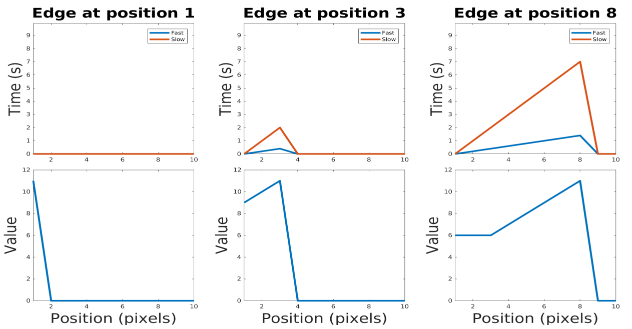


Figure 3: An edge moves from position 1 to 10 at two different speeds. It creates a slope on the Standard Time Surface T **(Top)** and on the corresponding Speed Invariant Time Surface S **(Bottom)**. S is identical for both speeds.

generates a different slope behind it. On the Speed Invariant Time Surface, the slope is the same for all the edges, and has a total length of 5 pixels which is equal to the parameters r used for this experiment. Also, the Speed Invariant Time Surface is constant after the slope with a value of $r+1$. Finally, it can be seen that the values in the Speed Invariant Time Surface remain between 0 and $2r+1$.

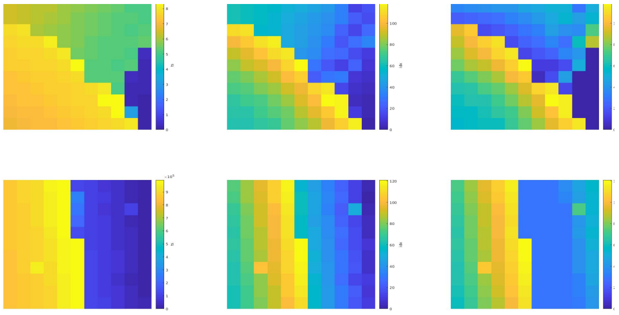
In Fig. 4, we compare the standard Time Surface, the sorting normalization method of [3], and our Speed Invariant Time Surface. The two normalization approaches achieve similar results for both examples with a high contrast around the edge in comparison with the standard Time Surface. Furthermore, our Speed Invariant Time Surface increases this contrast by reducing the values after the edge.

Algorithm 1 Speed Invariant Time Surface

```
Output: Speed Invariant Time Surface $S(x, y, p)$
Initialization: $S(x, y, p) \leftarrow 0$ for all $(x, y, p)$
For each incoming event $(x, y, p)$, update $S$:
for $-r \leq dx \leq r$ do
    for $-r \leq dy \leq r$ do
        if $S(x+dx, y+dy, p) \geq S(x, y, p)$ then
            $S(x+dx, y+dy, p) \leftarrow S(x+dx, y+dy, p) - 1$
$S(x, y, p) \leftarrow (2r+1)^2$
```

4.3. Learning to Detect Corner Points from Events

Previous corner detectors rely on hand-crafted rules which make them unstable when the input events do not follow their assumptions. For example, *evFast* is not able to detect corners during a change of motion. In the case of *Arc* , the detection of obtuse angles lead to a high number of false detection on edges. Finally, *evHarris* is less robust on fast motions and computationally too expensive. This is why we train a classifier to detect corner points from events. More exactly, given an incoming event $\mathbf{e}_i = (x_i, y_i, p_i, t_i)$, we first update our time surface S as explained in Section 4.2, and we extract from S a local patch $\mathbf{s}_i$ of size $n \times n$ centered on the event 2D location (x_i, y_i).



Standard Time Surface

Normalization method of [3]

Speed Invariant Time Surface

Figure 4: Different types of Time Surfaces for two different patches. The two last methods achieve similar results with a high contrast around the edge in comparison with the standard Time Surface. Our Speed Invariant Time Surface is much more efficient to compute and increases the contrast by reducing the values after the edge.

An ideal detector $\mathcal{F}^*$ taking $\mathbf{s}_i$ as input is defined as

$$\mathcal{F}^*(\mathbf{s}_i) = \begin{cases} 1 & \text{if } \mathbf{e}_i \text{ is a feature event, and} \\ 0 & \text{otherwise .} \end{cases} \quad (3)$$

In principle, any classifier could be used to implement $\mathcal{F}$, but in practice we chose a Random Forest [11] because of its efficiency. For the sake of completeness, in the following we briefly describe Random Forests. A Random Forest classifier $\mathcal{F}$ is given by an ensemble of decision trees F_l:

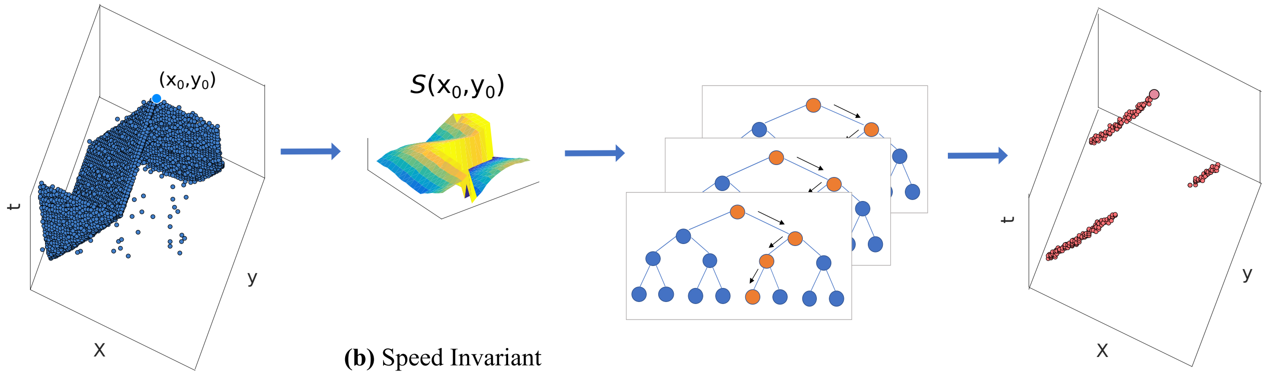
$$\mathcal{F}(\mathbf{s}) = \frac{1}{L} \sum_{l=1}^{L} F_l(\mathbf{s}) \; . \quad (4)$$

A decision tree $F_l(\mathbf{s})$ classifies a sample $\mathbf{s} = (s_1, \ldots, s_K)$ by recursively branching the nodes of the tree until the sample reaches a leaf node. In each node a binary split function is applied to the input to decide if the sample is sent to the left or the right child node.

In practice, we use decision stumps as split functions [16], where a descriptor dimension d_k is compared to a threshold th. This is computationally efficient and effective in practice [16]. The output of the tree is the prediction stored at the leaf reached by the sample, which in our case is the probability of the sample being a feature point.

During training, each tree is trained independently in a greedy manner, one node at the time. For a given node, let $\mathbf{D}_N$ be the training set of samples that reached this node. The optimal $\theta = (k, th)$ for the split are obtained by minimizing the Gini impurity index [21] by exhaustive search. The dataset is then split in left and right $\mathbf{D}_L$, $\mathbf{D}_R$ and the children nodes are trained recursively. Training stops when a maximum depth is reached, when a minimum number of samples is reached or if the impurity index is too low.

A Random Forest improves the accuracy of a single decision tree by training multiple trees. The trees need to be



(a) Input Events **(b)** Speed Invariant Time Surface **(c)** Random Forest **(d)** Corner Events

Figure 5: Overview of the proposed method. For each incoming event (x_0, y_0, t_0, p_0) in the input stream **(a)** we compute the Speed Invariant Time Surface **(b)**. The Speed Invariant Time Surface is used as input to a Random Forest trained to discriminate corner points **(c)**. If the probability returned by the Random Forest is above a threshold, the event is classified as a corner. This generates a sparse and stable stream of corner events **(d)** which can be used for further processing.

uncorrelated, and this is done by randomly sampling a subset of the training set for each tree and by randomly subsampling a subset of the descriptors dimensions at each node.

4.4. Building a Dataset for Event-based Corner Detection

To train our classifier $\mathcal{F}$, we need a labeled training set $\{(\mathbf{s}_j, c_j)\}$, where the binary class c_j of event $\mathbf{e}_j$ is 1 if $\mathbf{e}_j$ is a feature event, 0 otherwise. Building such a dataset in the case of event-based cameras is not trivial. Manual annotation is impractical, since it would require to label single events in a sequence of million of events per second.

We propose to leverage the graylevel measurement provided by a HVGA ATIS sensor, already discussed in Section 2. For every event, we apply the Harris corner detector to its location only, in the graylevel image. If a corner is detected at this location, we add the event to our training set as a positive sample, and as a negative sample otherwise. In practice, even the Harris detector applied to graylevel images can sometimes fail in presence of noise, and we therefore decided to acquire our dataset by recording well contrasted geometric patterns. Examples of this dataset are shown in the supplemen tary material The dataset can be downloaded at the following URL[1]. In this way, we greatly reduce the number of outliers in the dataset. As we will see in Section 6, even if the pattern we use is quite simple, the classifier learnt on such data can generalize well in more complex real situations. We also extend this protocol to a DAVIS sensor. In this case the corners are detected on entire frames at regular intervals and all the events falling within 2 pixels form a detected corner in a time interval of $5ms$ are marked as corner events.

5. Evaluation of Event-based Detectors

Event-based Datasets for Corner Detection We evaluate our method on two event-based datasets. The first is the commonly used Event-Camera dataset [40]. It is composed by recordings of different scenes taken with a DAVIS sensor. As done by previous works, we consider the `boxes`, `shapes`, `hdr` and `dynamic` subsets for evaluation, for which camera poses groundtruth are provided, but we keep the simplest `shapes` dataset to train our model.

The second dataset we use is a new dataset that we introduce for the first time in this paper. Compared to the Event-Camera dataset, ours was acquired using a higher resolution HVGA ATIS sensor, thus allowing better localization and finer feature detection, together with higher data rate. While the Event-Camera dataset is more generic and adapted to test visual odometry and SLAM applications, the intent of our dataset is to specifically evaluate the accuracy of event-based feature detection and tracking algorithms. We call this dataset the *HVGA ATIS Corner Dataset*. It consists of 7 sequences of increasing difficulty, from a standard checkerboard to a complex natural image (Fig. 6). We record planar patterns, so that ground truth acquisition is simple and the evaluation is less affected by triangulation errors.

Evaluation Metrics Previous approaches for event-based feature detection rely on Harris corners extracted from graylevel images [61, 64, 23]. This approach has the shortcoming of penalizing event-based features when not matched to any graylevel one, even if these features might correspond to stable points in the events stream. We notice that our approach, since it was trained starting from Harris corners would have an advantage when using this metric.

Other works [39] instead, consider a detected corner valid if, when associated with nearby corners, it forms a well localized track in space and time. This metric evaluates how easy it is to track a given feature. However, it does not take into account the correct trajectory of the feature.

[1] `http://prophesee.ai/hvga-atis-corner-dataset`

We also start from the observation that a stable detector, able to continuously identify a feature in the events stream, would remove the need of complex tracking and data association methods. Therefore, in order to assess also the quality of an event-based detector, we combine it with a simple tracking algorithm based on nearest neighbor matching in space and time. After tracking, we can evaluate the accuracy of the method by computing the reprojection error associated to the feature tracks.

In the case of the planar dataset, the reprojection error is computed by estimating a homography between two different timestamps. More precisely, given two timestamps t_1 and t_2, we collect all the features that fall within a $5ms$ timewindow from these times. We then consider the last feature for each track. This gives two sets of 2D correspondences that can be used to estimate the homography between the two views of the camera. We use RANSAC for a robust estimation. Once the homography is computed, we reproject points from time t_2 to the reference time t_1 and compute the average distance between the reference points and the projected ones. During this process, we exclude points detected outside the planar pattern.

In the case of the Event-Camera dataset, which contains 3D scenes, the reprojection error is computed by triangulating the tracked points, using the available 3d poses. We use the same protocol as in [3] and report also the percentage of tracks with an error smaller than 5 pixels. Finally we compare the methods in terms of computational time, as done in [23, 3]. All our experiments were implemented in C++ and run on a Xeon CPU E5-2603 v4 at 1.70GHz.

6. Experiments

Parameters and Baselines Our method depends on few parameters, namely the radius r used to compute the Speed Invariant Time Surface, the size n of the classifier input patch, and the parameters for the Random Forest. The parameters were optimized by cross-validation on the training set of Section 4.4 to minimize the misclassification error. Once the best parameters have been found, we fix them and use them for all the test sequences of Section 5. For the descriptor, we set $r = 6$ and $n = 8$. For the Random Forest, we use 10 trees. We stop growing the trees when there are less than 50 training samples in a branch.

We compare our method against previously published event-based feature detectors: the event-based Harris detector of [61], which we denote *evHarris* ; the event-based Fast of [39] *evFast* ; and its recent modification *Arc* [4]. For all of these methods we use the publicly available implementation provided by the authors.

As done in [4], we also apply an event-based 'trail' filter before each detector. This filter removes the multiple events generated by a contrast step in intensity which caused multiple threshold crossing. The timings of Table 3 are reported taking into account this filtering.

Ablation Study In the first set of experiments, we quantify the improvement brought by our Speed Invariant Time Surface formulation against the standard Time Surface.

We train two Random Forests, one using local patches extracted from the Time Surface of [6], and the second one on Speed Invariant Time Surface patches. Then, we apply these detectors to the HVGA dataset and track the features using nearest neighbor data association with a radius of 3 pixels and a temporal window of $10ms$. From the tracked features we estimate the homography using different time steps, as explained in Section 5. The corresponding reprojection errors are reported in Table 1. As we can see our method has lower reprojection error, the reason is that the detector trained on the Time Surface can not generalize well, and, a part for the simple chessboard pattern returns very noisy detections. We refer to the supplementary material for a visual comparison of the detected features.

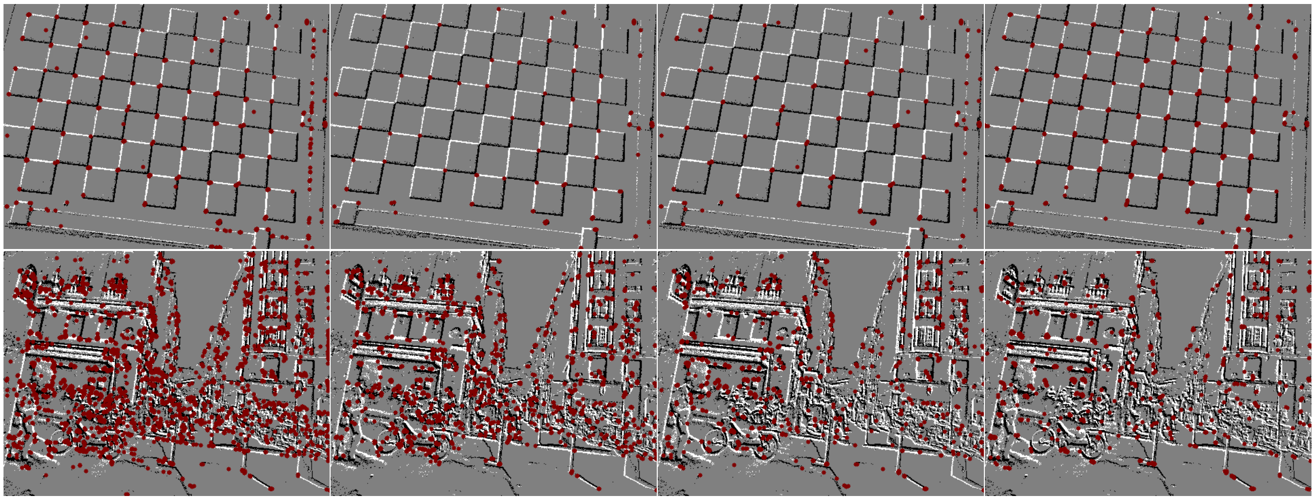
Table 1: Reprojection error on the HVGA Corner dataset for different values of Δt used to estimate the homography, when training a Random Forest on the Time Surface [6] or the proposed Speed Invariant Time Surface (SILC). SILC results in a more stable and accurate detections.

Random Forest	$\Delta t = 25ms$	$\Delta t = 50ms$	$\Delta t = 100ms$
Using T [6]	5.79	8.48	12.26
Using S (SILC)	**2.45**	**3.03**	**3.70**

Evaluation on the HVGA ATIS Corner Dataset In this section, we compare the stability of event-based feature detectors against the baseline methods using the homography reprojection error. Since this metric can be applied only to planar scenes, we use it for our dataset and not on the Event-Camera dataset. For nearest neighbor data association, we used a radius of 3 pixels and a temporal window of $10ms$. We also compute the lifetime of the tracked features, defined as the difference between the first and the last detection of the track. We then compute the average lifetime of the first 100 features.

We compute the homography at different time intervals, ranging from $25ms$ to $200ms$. The results for $100ms$ are shown in Fig. 7. Detailed values for other values and for each sequence are reported in the supplementary material, and show similar behavior. Our method is able to track a corner point longer while having a low reprojection error. The error for *evFast* is slightly lower, but the tracks for this method are significantly shorter. Finally, *evHarris* and *Arc* are very sensitive to noise and respond on edges, which make their tracks longer then *evFast* , but poorly reliable.

Qualitative results are shown in Fig. 7. A snapshot example can be found in Fig. 6. Corresponding video sequences are provided in the supplementary material.



Arc [4] *evHarris* [61] *evFast* [39] SILC

Figure 6: Comparison of event-based detectors on the HVGA ATIS Corner Dataset. Positive and negative events accumulated during 5ms showed in black and white with corresponding corner events in red. Our method is more robust to noise and can detect stable corners even in complex scenes.

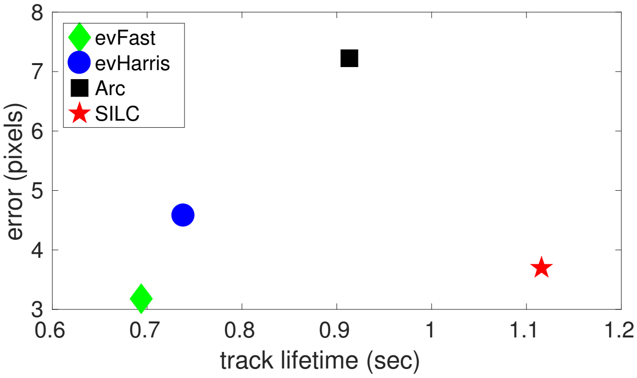


Figure 7: Comparison of detector performance on the HVGA ATIS Corner dataset. Our method is able to track a corner longer than the baselines while keeping a low reprojection error.

Evaluation on the Event-Camera Dataset We repeat similar experiments for the Event-Camera Dataset. Since the resolution of the camera is lower for this dataset, we use a larger temporal window, equal to 50ms, for the tracking. Because of the lower resolution and the higher level of noise, this dataset is more challenging.

The reprojection error and the percentage of valid tracks are given in Tab. 2. Detailed values are reported in the supplementary material. We notice that applying the model trained on the ATIS camera (SILC ATIS) generalizes well on the DAVIS, reaching similar performance than the baseline. Since the DAVIS camera provides gray-level images, we also retrained a Random Forest on the DAVIS data (SILC DAVIS). We use a small model composed of 10 trees of depth 10. We observe that the results obtained with our detector and a simple nearest neighbor tracking are comparable with results obtained with complex trackers [3].

Table 2: Evaluation on the Event-camera dataset. Our method has the lowest reprojection error and generalizes well across different sensors.

	evHarris	*evFast*	*Arc*	SILC	SILC
	[61]	[39]	[4]	ATIS	DAVIS
3D reprj. error (pix)	2.46	2.50	2.58	2.53	**2.16**
Valid tracks (%)	47.4	50.1	42.9	47.3	**65.3**

Table 3: Computational Time on the HVGA dataset. Our method is real time despite the high data rate of the sensor.

	evHarris	*evFast*	*Arc*	SILC
	[61]	[39]	[4]	
Event rate (Mev/s)	0.22	1.74	5.61	1.61
Real Time Factor	0.60	3.80	12.32	3.53

7. Conclusion and Future Work

We presented an efficient and accurate learning approach for event-based corner detection. Our method produces more stable and repeatable corners compared to the state of-the-art and when coupled with a simple tracking algorithm gives good accuracy. A key component for our approach is a novel formulation of the Time Surface, which provides a rich event-based representation which is invariant to the local speed of the object. In the future, we plan to apply the Speed Invariant Time Surface to other event-based vision tasks, such as low-latency object detection or relocalization.

Acknowledgement

Dr. Vincent Lepetit is a senior member of the *Institut Universitaire de France*.

References

[1] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, et al. Truenorth: Design and Tool Flow of a 65 Mw 1 Million Neuron Programmable Neurosynaptic Chip. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2015.

[2] H. Altwaijry, A. Veit, S. J. Belongie, and C. Tech. Learning to detect and match keypoints with deep architectures. In *BMVC*, 2016.

[3] I. Alzugaray and M. Chli. ACE: An Efficient Asynchronous Corner Tracker for Event Cameras. In *2018 International Conference on 3D Vision (3DV)*, 2018.

[4] I. Alzugaray and M. Chli. Asynchronous Corner Detection and Tracking for Event Cameras in Real Time. *IEEE Robotics and Automation Letters*, 2018.

[5] A. Andreopoulos, H. J. Kashyap, T. K. Nayak, A. Amir, and M. D. Flickner. A Low Power, High Throughput, Fully Event-Based Stereo System. In *CVPR*, 2018.

[6] R. Benosman, C. Clercq, X. Lagorce, S. Ieng, and C. Bartolozzi. Event-Based Visual Flow. *IEEE Trans. Neural Netw. Learning Syst.*, 2014.

[7] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan. Asynchronous Frameless Event-Based Optical Flow. *Neural Netw.x*, 2012.

[8] O. Bichler, D. Querlioz, S. J. Thorpe, J.-P. Bourgoin, and C. Gamrat. Extraction of Temporally Correlated Features from Dynamic Vision Sensors with Spike-Timing-Dependent Plasticity. *Neural Networks*, 2012.

[9] S. M. Bohte, J. N. Kok, and H. La Poutre. Error-Backpropagation in Temporally Encoded Networks of Spiking Neurons. *Neurocomputing*, 2002.

[10] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck. A 240× 180 130 dB 3 μs Latency Global Shutter Spatiotemporal Vision Sensor. *IEEE Journal of Solid-State Circuits*, 2014.

[11] L. Breiman. Random Forests. *Machine learning*, 2001.

[12] M. Cannici, M. Ciccone, A. Romanoni, and M. Matteucci. Event-Based Convolutional Networks for Object Detection in Neuromorphic Cameras. In *arXiv*, 2018.

[13] Y. Cao, Y. Chen, and D. Khosla. Spiking Deep Convolutional Neural Networks for Energy-Efficient Object Recognition. *IJCV*, 2015.

[14] X. Clady, S.-H. Ieng, and R. Benosman. Asynchronous Event-Based Corner Detection and Matching. *Neural Networks*, 2015.

[15] X. Clady, J.-M. Maro, S. Barré, and R. B. Benosman. A Motion-Based Feature for Event-Based Pattern Recognition. *Frontiers in neuroscience*, 2017.

[16] A. Criminisi, J. Shotton, E. Konukoglu, et al. Decision forests: A unified framework for classification, regression, density estimation, manifold learning and semi-supervised learning. *Foundations and Trends® in Computer Graphics and Vision*, 2012.

[17] M. Davies, N. Srinivasa, T.-H. Lin, G. Chinya, Y. Cao, S. H. Choday, G. Dimou, P. Joshi, N. Imam, S. Jain, et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro*, 2018.

[18] D. DeTone, T. Malisiewicz, and A. Rabinovich. Superpoint: Self-supervised interest point detection and description. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2018.

[19] P. Di Febbo, C. Dal Mutto, K. Tieu, and S. Mattoccia. Kcnn: Extremely-efficient hardware keypoint detection with a compact convolutional neural network. In *CVPR Workshop on Embedded Vision (EVW)*, 2018.

[20] D. Drazen, P. Lichtsteiner, P. Häfliger, T. Delbrück, and A. Jensen. Toward Real-Time Particle Tracking Using an Event-Based Dynamic Vision Sensor. *Experiments in Fluids*, 2011.

[21] J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*. Springer series in statistics New York, NY, USA:, 2001.

[22] S. B. Furber, F. Galluppi, S. Temple, and L. A. Plana. The Spinnaker Project. *Proceedings of the IEEE*, 2014.

[23] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza. Asynchronous, Photometric Feature Tracking Using Events and Frames. In *ECCV*, 2018.

[24] A. Glover and C. Bartolozzi. Robust Visual Tracking with a Freely-Moving Event Camera. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, 2017.

[25] R. Gütig and H. Sompolinsky. The Tempotron: A Neuron That Learns Spike Timing–Based Decisions. *Nature neuroscience*, 2006.

[26] C. Harris and M. Stephens. A Combined Corner and Edge Detector. In *Alvey vision conference*, 1988.

[27] H. Kim, S. Leutenegger, and A. J. Davison. Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera. In *ECCV*, 2016.

[28] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza. Low-Latency Visual Odometry Using Event-Based Feature Tracks. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 2016.

[29] X. Lagorce, C. Meyer, S.-H. Ieng, D. Filliat, and R. Benosman. Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking. *IEEE Transactions on Neural Networks and Learning Systems*.

[30] X. Lagorce, G. Orchard, F. Galluppi, B. E. Shi, and R. B. Benosman. Hots: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. *PAMI*, 2017.

[31] H. Li, G. Li, and L. Shi. Classification of Spatiotemporal Events Based on Random Forest. In *Advances in Brain Inspired Cognitive Systems: International Conference*, 2016.

[32] P. Lichtsteiner, C. Posch, and T. Delbruck. A 128x128 120db 15us Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid State Circuits*, 2008.

[33] B. Linares-Barranco, T. Serrano-Gotarredona, L. A. Camuñas-Mesa, J. A. Perez-Carrasco, C. Zamarreño-Ramos, and T. Masquelier. On Spike-Timing-Dependent-Plasticity, Memristive Devices, and Building a Self-Learning Visual Cortex. *Frontiers in neuroscience*, 2011.

[34] H. Liu, D. P. Moeys, G. Das, D. Neil, S.-C. Liu, and T. Delbrück. Combined Frame-And Event-Based Detection and Tracking. In *Circuits and Systems (ISCAS), 2016 IEEE International Symposium on*, 2016.

[35] W. Liu, H. Chen, R. Goel, Y. Huang, A. Veeraraghavan, and A. Patel. Fast Retinomorphic Event-Driven Represen-

tations for Video Recognition and Reinforcement Learning. In *arXiv*, 2018.

[36] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garcıa, and D. Scaramuzza. Event-Based Vision Meets Deep Learning on Steering Prediction for Self-Driving Cars. In *CVPR*, 2018.

[37] D. Martí, M. Rigotti, M. Seok, and S. Fusi. Energy-Efficient Neuromorphic Classifiers. *Neural computation*, 2016.

[38] T. Masquelier and S. J. Thorpe. Unsupervised Learning of Visual Features through Spike Timing Dependent Plasticity. *PLoS computational biology*, 2007.

[39] E. Mueggler, C. Bartolozzi, and D. Scaramuzza. Fast Event-Based Corner Detection. In *BMVC*, 2017.

[40] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. The Event-Camera Dataset and Simulator: Event-Based Data for Pose Estimation, Visual Odometry, and SLAM. *The International Journal of Robotics Research*, 2017.

[41] D. Neil, M. Pfeiffer, and S.-C. Liu. Phased LSTM: Accelerating Recurrent Network Training for Long or Event-Based Sequences. In *NIPS*, 2016.

[42] Z. Ni, A. Bolopion, J. Agnus, R. Benosman, and S. Regnier. Asynchronous Event-Based Visual Shape Tracking for Stable Haptic Feedback in Microrobotics. *IEEE Transactions on Robotics*, 2012.

[43] Z. Ni, S.-H. Ieng, C. Posch, S. Régnier, and R. Benosman. Visual Tracking Using Neuromorphic Asynchronous Event-Based Cameras. *Neural computation*, 2015.

[44] P. O'Connor, D. Neil, S.-C. Liu, T. Delbruck, and M. Pfeiffer. Real-Time Classification and Sensor Fusion with a Spiking Deep Belief Network. *Frontiers in neuroscience*, 2013.

[45] X. Peng, B. Zhao, R. Yan, H. Tang, and Z. Yi. Bag of Events: An Efficient Probability-Based Feature Extraction Method for AER Image Sensors. *IEEE transactions on neural networks and learning systems*, 2017.

[46] C. Posch, D. Matolin, and R. Wohlgenannt. A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor with Lossless Pixel-Level Video Compression and Time-Domain CDS. *Solid-State Circuits, IEEE Journal of*, 2011.

[47] B. Ramesh, H. Yang, G. Orchard, N. A. L. Thi, and C. Xiang. DART: Distribution Aware Retinal Transform for Event-Based Cameras. In *arXiv*, 2017.

[48] H. Rebecq, T. Horstschaefer, G. Gallego, and D. Scaramuzza. EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time. *IEEE Robotics and Automation Letters*, 2017.

[49] E. Rosten and T. Drummond. Fusing points and lines for high performance tracking. In *ICCV*. IEEE, 2005.

[50] E. Rosten and T. Drummond. Machine Learning for High-Speed Corner Detection. In *ECCV*, 2006.

[51] A. Russell, G. Orchard, Y. Dong, Ş. Mihalas, E. Niebur, J. Tapson, and R. Etienne-Cummings. Optimization Methods for Spiking Neurons and Networks. *IEEE transactions on neural networks*, 2010.

[52] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gómez-Rodríguez, L. Camuñas-Mesa, R. Berner, M. Rivas-Pérez, T. Delbruck, et al. CAVIAR: A 45k Neuron, 5M Synapse, 12G Connects/s AER Hardware Sensory–Processing–Learning–Actuating System for High-Speed Visual Object Recognition and Tracking. *IEEE Transactions on Neural Networks*, 2009.

[53] T. Serrano-Gotarredona and B. Linares-Barranco. A 128 X 128 1.5% Contrast Sensitivity 0.9% FPN 3 μs Latency 4 mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Preamplifiers. *Solid-State Circuits, IEEE Journal of*, 2013.

[54] S. Sheik, M. Pfeiffer, F. Stefanini, and G. Indiveri. Spatio-Temporal Spike Pattern Classification in Neuromorphic Systems. In *Biomimetic and Biohybrid Systems*. Springer, 2013.

[55] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman. HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification. In *CVPR*, 2018.

[56] J. Sochman and J. Matas. Waldboost-learning for time constrained sequential detection. In *CVPR*. IEEE, 2005.

[57] J. Šochman and J. Matas. Learning fast emulators of binary decision processes. *International Journal of Computer Vision*, 2009.

[58] B. Son, Y. Suh, S. Kim, H. Jung, J.-S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, et al. 4.1 a 640× 480 Dynamic Vision Sensor with a 9μm Pixel and 300meps Address-Event Representation. In *Solid-State Circuits Conference (ISSCC), 2017 IEEE International*, 2017.

[59] C. Strecha, A. Lindner, K. Ali, and P. Fua. Training for task specific keypoint detection. In *Joint Pattern Recognition Symposium*. Springer, 2009.

[60] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza. Feature Detection and Tracking with the Dynamic and Active-Pixel Vision Sensor (Davis). In *Event-based Control, Communication, and Signal Processing (EBCCSP), 2016 Second International Conference on*, pages 1–7, 2016.

[61] V. Vasco, A. Glover, and C. Bartolozzi. Fast Event-Based Harris Corner Detection Exploiting the Advantages of Event-Driven Cameras. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 2016.

[62] Y. Verdie, K. Yi, P. Fua, and V. Lepetit. Tilde: A temporally invariant learned detector. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[63] K. M. Yi, E. Trulls, V. Lepetit, and P. Fua. Lift: Learned invariant feature transform. In *European Conference on Computer Vision*. Springer, 2016.

[64] A. Z. Zhu, N. Atanasov, and K. Daniilidis. Event-Based Feature Tracking with Probabilistic Data Association. In *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 4465–4470, 2017.

[65] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. Ev-Flownet: Self-Supervised Optical Flow Estimation for Event-Based Cameras. In *arXiv*, 2018.

Appendix

	Gen3 CD	Gen3 ATIS
Supplier	Prophesee	Prophesee
Year	2017	2017
Resolution (pixels)	640x480	480x360
Latency (μs)	40 - 200	40 -200
Dynamic range (dB)	>120	>120
Min. contrast sensitivity (%)	12	12
Die power consumption (mW)	36 - 95	25 - 87
Camera Max. Bandwidth (Meps)	66	66
Chip size (mm^2)	9.6x7.2	9.6x7.2
Pixel size (μm^2)	15x15	20x20
Fill factor (%)	25	25
Supply voltage (V)	1.8	1.8
Stationary noise (ev=pix=s) at 25C	0.1	0.1
CMOS technology (μm)	0.18 1P6M CIS	0.18 1P6M CIS
Grayscale output	no	yes
Grayscale dynamic range (dB)	NA	>100
Max. framerate (fps)	NA	NA
IMU output	1 kHz	1 kHz

Table 4: Technical description of the cameras used for the paper. The Gen3 ATIS was used to generate the ground truth.