Long-Lived Accurate Keypoint in Event Streams

Philippe Chiberre[1,2], Etienne Perot[1], Amos Sironi[1], and Vincent Lepetit[2]

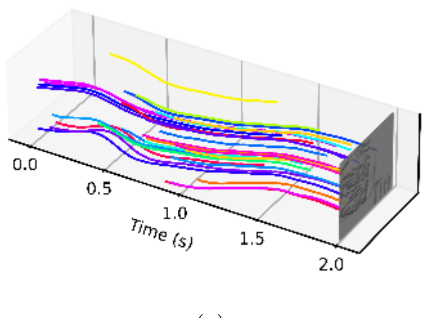
[1] Prophesee, Paris, France
[2] Ecole des Ponts, Univ Gustave Eiffel, CNRS, LIGM, F-77454 Marne-la-Vallée, France {pchiberre,asironi}@prophesee.ai vincent.lepetit@enpc.fr

Abstract. We present a novel end-to-end approach to keypoint detection and tracking in an event stream that provides better precision and much longer keypoint tracks than previous methods. This is made possible by two contributions working together. First, we propose a simple procedure to generate stable keypoint labels, which we use to train a recurrent architecture. This training data results in detections that are very consistent over time. Moreover, we observe that previous methods for keypoint detection work on a representation (such as the time surface) that integrates events over a period of time. Since this integration is required, we claim it is better to predict the keypoints' *trajectories* for the time period rather than single locations, as done in previous approaches. We predict these trajectories in the form of a series of heatmaps for the integration time period. This improves the keypoint localization. Our architecture can also be kept very simple, which results in very fast inference times. We demonstrate our approach on the HVGA ATIS Corner dataset as well as "The Event-Camera Dataset and Simulator" dataset, and show it results in keypoint tracks that are three times longer and nearly twice as accurate as the best previous state-of-the-art methods. We believe our approach can be generalized to other event-based camera problems, and we release our source code to encourage other authors to explore it.

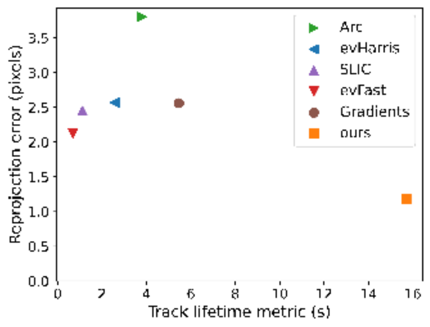
Keywords: Event-based cameras, keypoint detection.

1 Introduction

Keypoint detection is a fundamental problem of computer vision and a building block to a large number of applications such as Simultaneous Localisation and Mapping (SLAM), Structure-from-Motion (SfM), object recognition and tracking. Event-based cameras are recent sensors capturing the change in luminosity, asynchronously across the sensor [5, 13, 21]. They are able to capture high dynamic range scenes while maintaining a low-power consumption, making them particularly attractive for embedded systems [6]. Not surprisingly, many methods for keypoint detection for event-based cameras have already been proposed [1, 2, 8, 15, 16, 22].



(a)



(b)

Fig. 1: (a) Our tracks for the beginning of the Guernica sequence from the HVGA ATIS Corner dataset [15]. (b) We compare ourselves to Arc [1], evHarris [22], SILC [15], evFast [16], and the Gradients-based method of [2]. Our detector predicts well-localized keypoints that can be tracked reliably with a simple nearest-neighbor matching algorithm to obtain very long tracks. It nearly triples the track lifetime metric of the previous best method while reducing the reprojection error by more than 1 pixel.

However, the nature of the signal provided by event-based cameras is very different from the images captured by regular cameras. Event-based cameras send 'events', which signal a sudden change of light intensity at an image location. Alone, an event therefore provides very little information. Events are also very noisy, with many false positives and false negatives [3, 9]. This makes computer vision problems for event-based cameras particularly challenging.

As shown in Figure 1, we propose a novel event-based keypoint detector that significantly outperforms state-of-the-art methods, both in terms of stability as it provides much longer tracks, and accuracy as the keypoints are much better localized.

Our detector is based on a deep recurrent architecture. A first simple but critical contribution is how we generate training data: We generate video sequences of bitmap frames by applying homographies to images from the COCO dataset [14]. We transform these video sequences into event streams using an event-based camera simulator. To obtain keypoint location labels, we run the Harris corner detector [10] on the original image from COCO. We warp these locations using the homographies to obtain the keypoint locations over time. This generates much more stable labels than, for example, running the Harris detector on all the frames of the sequences.

While this procedure is simple, this results in keypoint detections that are much more consistent over time than previous methods. These detections can be linked by a simple nearest-neighbor matching procedure into keypoint tracks very reliably. This results in very long keypoint tracks, which are very important for many applications such as object tracking or Structure-from-Motion. This

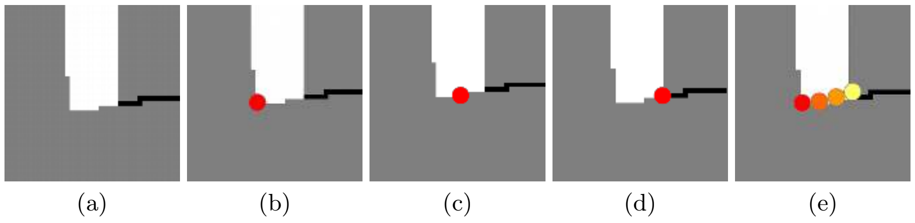


Fig. 2: **Keypoint detection as trajectory prediction.** (a) Buffer of accumulated events around a keypoint over a short period of time (20ms). Note that the keypoint is actually spread over several image locations. (b-d) Some of the possible locations for the keypoint, if one sticks to a single location for the buffer. (e) Instead, we predict multiple heatmaps each corresponding to a point in time (shown here with a gradient in color). This makes linking detections from several time periods much more reliable, and results in much longer keypoint tracks.

also provides very accurate locations. Note that while our architecture is trained on planar scenes (since we use homographies), it also works well on 3D scenes, which we show by evaluating on the dataset from The Event-Camera Dataset and Simulator [17]. Generalization of keypoints trained on planar scenes to 3D scenes was observed before for bitmap frames, for example in [4].

We also observed all previous event-based keypoint detection methods integrate events over a period of time in a 2D buffer. This integration period is required to gather enough information from the events, and compensates for noise. While this integration period is required, all previous methods for keypoint detection for event-based cameras still provide a single image location for each keypoint detected for this period. This is probably only because of the legacy of keypoint detection methods for bitmap images, in which it is assumed that the image information is captured instantly.

Instead, we predict multiple successive locations rather for a single integration period. This is illustrated in Figure 2: We predict a series of successive heatmaps for the entire frame. The local maxima of the heatmaps gives us the predicted keypoint locations. Predicting multiple heatmaps lets us handle different numbers of keypoints and the fact that keypoints can appear or disappear during the integration period. Because the predicted locations are spaced out by a very fine time step, we significantly improve the accuracy of our predictions.

An alternative would be to predict a single location per keypoint, but for many overlapping integration periods. However, this would result in significantly increased computation times. By contrast, our approach is very light and has a computational cost similar to previous methods.

Beyond keypoint detection, we believe that our first observation—predicting multiple successive estimates for the integration period rather than a single one—can be applied to other event-based cameras such as segment detection, depth prediction and object detection to improve their accuracy. Thus, we hope our work will encourage other researchers to explore this direction.

2 Related Work

We focus here on methods for keypoint detection in event streams. They can be classified into two categories: The first category is made of 'handcrafted' methods, in contrast to methods based on machine learning.

2.1 Handcrafted Methods

Multiple hand crafted methods for event-based are inspired or adapted from frame-based feature detection. The easiest approach is to create a 2D intermediate representation on which one can apply a feature detector. eHarris [22] creates a binary image from the latest events, with pixel locations set to 1 if an event has occurred during some integration period and set to 0 elsewhere. The Harris corner detector, originally defined for greylevel images [10], is then applied to this binary image. A modification of this approach has recently been proposed by Glover et al. with luvHarris [8]. They introduce a novel representation of events, named Threshold-ordinal Surface, replacing the binary image. This new representation makes the Harris detection more precise and more stable at the expense of a greater computational load: the representation is computed event-by-event, which is not parallelisable. The computational impact is lessened by only rendering the Harris score map "as-fast-as-possible" and achieving real-time detection.

eFAST [16] and ARC [1] on the other hand use a non-computationally intensive representation, named the Surface of Active Events (SAE). The SAE is defined for each pixel and polarity as the latest time an event has occurred:

$$\text{SAE}[x, y, t] \leftarrow t \, . \tag{1}$$

The SAE is also computed for an integration period since it contains the information of many preceding events. They then proceed to apply a local matching pattern for each event at its location in the SAE. [16] considers two circles around the latest event and compares their timestamps. In [1], Alzugaray and Chli also consider two concentric circles around the latest event but introduce new rules for the classification of an event as a corner. [1] achieves better results while at the same time requiring less computations. In all cases, the approach of computation event-by-event becomes quickly intractable as the computation cannot be done in parallel while at the same time, resolution of event-based sensors keep increasing [5] and event rates can increase dramatically.

Hand-crafted approaches are sensitive to the noise present in the event stream. This is why recently, several authors modeled the keypoint detection problem as a machine learning one.

2.2 Learned Methods

[15] introduced a novel event representation, called the Speed-Invariant Time Surface. It aims at removing the impact of motion speed on the differences

of appearance of the same keypoint in an event stream. Instead of applying a frame-based feature detector, they train a random forest to classify each event as a keypoint or a non-keypoint. Despite improving the quality of the keypoints, the computation of the representation and random forest make this approach unsuitable in practice.

In [19], the authors show that a gray-level image can be reconstructed from events by training a recurrent neural network. Standard frame-based keypoint detectors can be applied on the output of their method, leading to very accurate results. More recently, [2] observed that reconstructing a frame is unnecessary and adds computational complexity. In fact, the authors noticed that events are closely related to image gradients, and proposed to learn to predict them and use them to compute the Harris score. One advantage of this work is that the deep architecture used to predict the gradients is very light, and we use a similar architecture to predict the heatmaps.

As discussed in the introduction, all these works rely on a period of integration of events to output keypoint locations, but predict a single location per keypoint for this period. In this work, we propose to predict multiple successive locations, and show this leads to much more accurate keypoint localization.

3 Method

Our detector integrates the events sent by the camera over a time period into an 'event cube', and outputs a series of heatmaps for this time period, Figure 3. We detail below how we construct this event cube in practice, the nature of the heatmaps, the deep architecture we use, and how we generate training data to train it.

3.1 Input Event Representation

The input to our detector is a $H \times W \times B$ event tensor $E(x, y, t)$, where H, W are the image sensor height and width respectively and B is the number of temporal bins. In practice, we use $B = 10$. We use the method first proposed in [23] and describe it briefly below for completeness.

Each input event (x_i, y_i, t_i, p_i) received during the integration period ΔT contributes to E by its polarity p_i to the two closest temporal bins using a triangular kernel. Formally, E is computed as

$$E(x, y, t) = \sum_i p_i \max(0, 1 - |t - t_i^\star|) \, , \tag{2}$$

for all image locations x, y and all temporal bins t. The sum is over all the events such that $x_i = x$ and $y_i = y$. $t_i^\star$ is the normalized timestamp of the i^{th} event:

$$t_i^\star = \frac{(t_i - t_{\min})}{\Delta T}(B - 1) \, , \tag{3}$$

where $t_{\min}$ is the time at the beginning of the integration period.

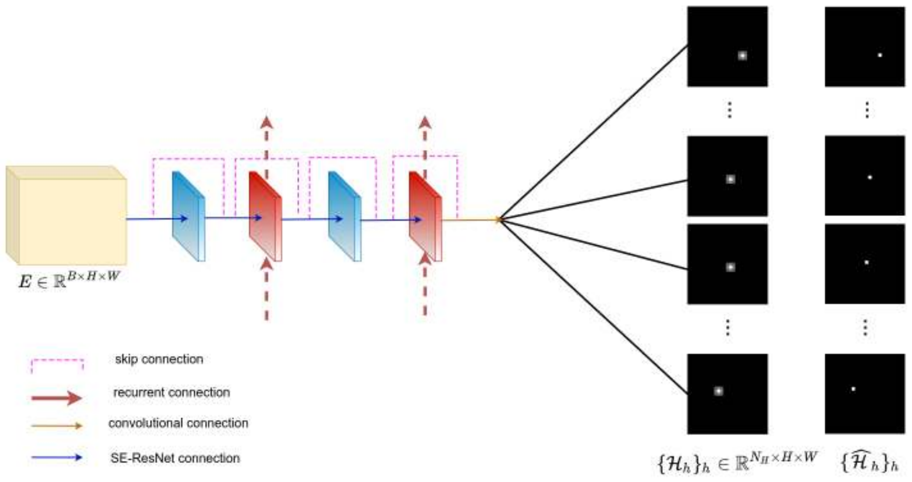

$E \in \mathbb{R}^{B \times H \times W}$
skip connection
recurrent connection
convolutional connection
SE-ResNet connection
$\{\mathcal{H}_h\}_h \in \mathbb{R}^{N_H \times H \times W}$
$\{\widehat{\mathcal{H}}_h\}_h$

Fig. 3: **Overview of our inference pipeline.** An event cube is build from the input events and fed to a recurrent neural network. We keep the architecture simple and efficient. We use the one introduced in [20] with the same modifications as in [2]. More precisely, the event cube is given as input to a Squeeze-and-Excite ResNet block, followed by a ConvLSTM with residual connection. This block of two layers is repeated once. Finally, a simple convolutional layer predicts the keypoints trajectory as a sequence of N_H heatmaps.

3.2 Predicting Heatmaps

From the event cube, our detector predicts a set of heatmaps. Heatmaps are convenient to predict the keypoints' locations as their number varies over time. In addition, predicting a dense tensor from a dense input like the event cube is standard and straightforward.

The local maxima of the heatmaps are expected to correspond the keypoints' locations. To do so, we rely on the cross-entropy between the predicted heatmaps and the labelled locations for the keypoints:

$$\mathcal{L} = \sum_{h \in [1;N_H]} \sum_{(x,y)} \mathrm{BCE}(\mathcal{H}_h(x,y), \widehat{\mathcal{H}}_h(x,y)) \,. \tag{4}$$

The first sum is over the N_H predicted heatmaps, in practice we use $N_H = 10$. The second sum is over the image locations (x, y). BCE denotes the binary cross-entropy.

The $\widehat{\mathcal{H}}_h$'s are labelled binary heatmaps for a given event cube: A location in $\widehat{\mathcal{H}}_h$ is set to 1 if there is a keypoint at this location at time step $[t_{\min} + (h-1)/N_H \times \Delta T, t_{\min} + (h)/N_H \times \Delta T[$ and set to 0 otherwise. The $\mathcal{H}_h$'s are the predicted heatmaps for the event cube. We guarantee that their values remain between 0 and 1 by applying the logistic function on the last

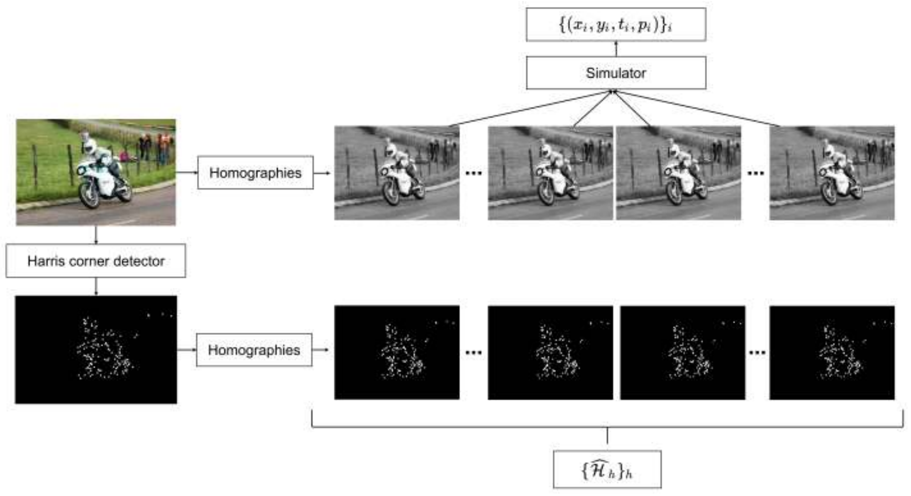


Fig. 4: **Generating training data.** Similarly to [2], given a still image, we apply homographies to warp it and generate a video sequence. However, by contrast with [2], we use the same homographies to warp the heatmap of Harris' corner of the still image to have a perfect match between the video frames and the corners. From the video sequence we then generate events using a simulator and then compute the event cubes [23] used as input to our network.

layer of our architecture. A similar loss function was used in [7] for example. It encourages the local maxima with large values in the predicted heatmaps to correspond to the locations of the keypoints.

3.3 Creating Training Data

To generate training data, we generate synthetic videos by applying homographies to grayscale images, and convert them into event streams using a simulator, as detailed below. The keypoint labels are obtained by applying the Harris corner detector to the original grayscale images. This procedure is also illustrated in Figure 4.

From an image to a video. More precisely, to generate a synthetic video, we select an image from the COCO dataset [14] at random. We apply the Harris corner detector to this image to obtain a set of keypoints. Then, as shown in Figure 4, we create a smooth video from homographies varying randomly to simulate complex camera motions in front of a planar scene.

To do so, we first generate a number of sine waves with differing periods and phases. We use this signal as a basis for a random yet continuous translation vector and a rotation vector. By combining the translation and rotation vectors with a random depth, we can obtain a homography matrix. By applying these homographies to the image, we obtain a smooth yet random video. More details are given in the supplementary material.

We generate a homography and thus a video frame every $\Delta T/N_H$ seconds, where ΔT is the integration period, and N_H the number of predicted heatmaps for this period.

From the video to the event stream. To generate events from the synthetic video, we apply our home-brewed simulator, which is mostly a combination of [18] and [3]. This simulator computes the difference between consecutive frames after applying a logarithm to the pixel intensities. It then generates events when the ON-or-OFF threshold is crossed by the log-difference, if the distance in time since the last event is greater than a hyperparameter for the refractory period. Multiple noise patterns are also added including random events firing, events firing multiple times and certain events always ON or OFF. The internal parameters of the simulator which define the noise are randomly selected for each video.

This finally gives us a stream of events $\{(x_i, y_i, t_i, p_i)\}_i$, from which we can build the event cubes.

Generating the keypoint labels. We simply apply the homographies already used to generate the video frames to the keypoint locations in the COCO images. From this, we can generate the $\widehat{\mathcal{H}}_h$ heatmaps needed to compute the loss function. The ablation study in Table 3 shows the beneficial influence of warping the keypoints locations as opposed to detecting them independently.

3.4 Training Details

We notice it is important to use hard negative mining during training. This is not surprising because of the imbalance between keypoint and non-keypoint pixels: Without mining, the network converges to the trivial solution of never predicting any keypoint, as this corresponds to a low value for the loss already.

At each iteration of the optimization, we select a labelled heatmap $\widehat{\mathcal{H}}_h$, and build a temporally consistent batch that mixes keypoint and hard non-keypoint pixels. This batch is made of all the positive locations in $\widehat{\mathcal{H}}_h$, *i.e.* the (x, y) such that $\widehat{\mathcal{H}}_h(x, y) = 1$ and the negative locations (x, y) for which the current predicted value $\mathcal{H}_h(x, y)$ is particularly wrong. More exactly, we use negative locations such that $\mathcal{H}_h(x, y) > \tau$ with threshold τ selected such that the number of negative locations in the batch is three times the number of positive locations.

We also use truncated-backpropagation-through-time of 10 time steps, detaching the state at each batch and never resetting for each video. We train for 30 epochs with a learning rate of $1e^{-4}$.

3.5 Inference and Tracking

At inference, we create the event cubes from incoming events over time periods. To define the time period, we use either a fixed length, or a fixed number of

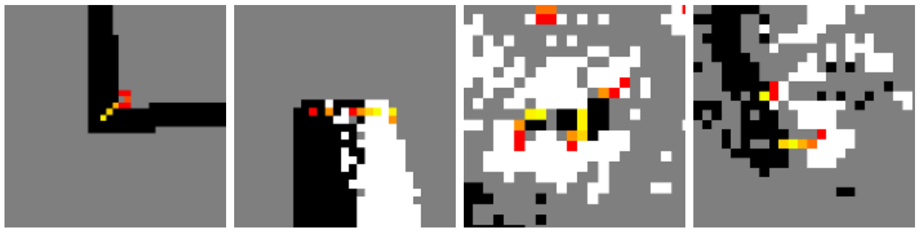


Fig. 5: **Predicting keypoints.** When events are integrated over a long time window, edges become thick and accurate keypoint localization becomes challenging. We therefore predict the keypoint's spatial position for multiple time steps. The color gradient from yellow to red represents the time from past to present. Our network is able to predict precise keypoint locations at every time step and predict their trajectory through time. (Best seen in colour)

events. We apply our network to each event cube once it is built. We obtain N_H heatmaps $\mathcal{H}_h$, to which we apply a local non-maximum suppression for every patch of 7×7 pixels and keep every local maximum over a threshold $\tau_{keypoint} = 0.2$ to be classified as a keypoint.

To obtain keypoint tracks, we rely on a simple nearest neighbor tracking algorithm: For each heatmap successively and for each keypoint in this heatmap, we look for a neighbor in a small region (we consider a 9×9 region) and a short time period in the past (we consider a 7ms period). If a neighbor keypoint is found, we associate the new keypoint to its track. If more than one neighbor keypoint are found, we consider the closest one only. If we do not find any neighbor, we start a new track with the new keypoint.

3.6 Architecture

We use a very simple recurrent neural network to predict heatmaps $\{\mathcal{H}_h\}_h$ from an event cube. Thanks to this simplicity, inference is very fast. We use an architecture similar to the one of [2] except we predict multiple heatmaps instead of two gradient maps as done in [2]. It is shown in Figure 3, and we describe it here for completeness. Also note that [2] still needs to compute the Harris score from the predicted gradients, while we directly predict keypoint heatmaps as output of the network.

Our architecture is a 5-layer fully convolutional network of 3×3 kernels. Each layer has 12 channels and residual connections [11]. The second and fourth layers are ConvLSTMs. The last layer, which predicts the heatmaps, is a standard convolutional layer, while the remaining feed-forward ones are Squeeze-Excite (SE) connections [12].

4 Experiments

In this section, we report our experimental comparison with previous event-based keypoint detectors, and an ablation study on the different aspects of our detector. We first introduce the datasets and metrics we consider, and then report the results of our experiments.

4.1 Datasets and Metrics

We consider the HVGA ATIS Corner dataset [15] build to evaluate event-based keypoint detectors. It consists of seven sequences with varying degrees of texture from a standard checkerboard to a complex natural image. These sequences were captured using an ATIS sensor with a resolution of 480×360 pixels. It only contains recordings of planar patterns.

Following the evaluation protocol defined in [2,15], we consider the following metrics:

– the δt-homography reprojection error $\text{er}_{\delta t}$, which depends on a time parameter δt:
$$\text{er}_{\delta t} = \frac{1}{N_K} \sum_t \sum_k \| W_t^{t+\delta t}(K_{t,k}) - K_{t+\delta t,k} \| \,, \tag{5}$$
where $K_{t,k}$ is a detected keypoint at time t, and $K_{t+\delta t,k}$ is the location of the keypoint at time $t+\delta t$ that belongs to the same track as keypoint $K_{t,k}$. If the track of $K_{t,k}$ ends before $t+\delta t$, it is ignored in the sum. $W_t^{t+\delta t}$ is the estimated homography (using $K_{t,k}$ and $K_{t+\delta t,k}$) that warps the view of the planar scene at time t to the one at time $t+\delta t$. N_K is the total number of terms in the sums, to compute the average of the distances.
– the average track lifetime of the longest 100 tracks over each of the seven sequence.

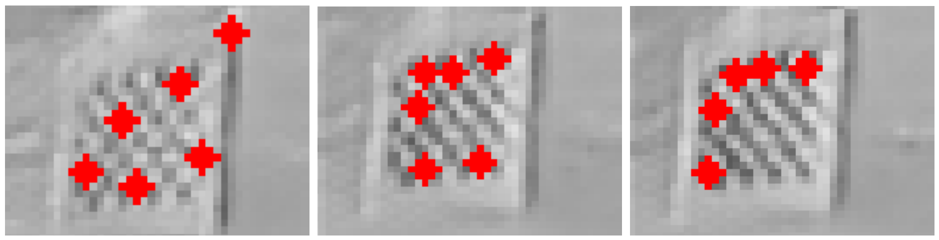


Fig. 6: **Ground truth inconsistencies for the Event-Camera Dataset [17].** Detecting ground-truth keypoint as Harris corners from consecutive event-to-video frames leads to unstable labels. (Best seen in colour)

We also evaluated our detector on The Event-Camera Dataset and Simulator [17]. The dataset is composed of multiple scenes with a variety of shapes and textures. Some scenes capture 3D and/or dynamic environments. This makes the computation of the homography reprojection error metric unfeasible. We therefore followed a similar protocol to [8] for evaluation: We run the event-to-video model from [19] to recreate frames from events and extract Harris corners on said frames to create a groundtruth. However, as can be seen in Figure 6, the groundtruth is inaccurate and unstable with time. Another flaw of this evaluation criterion is a positive bias towards keypoints detectors using, or based on, Harris corners. Nonetheless, for completeness, we provide an evaluation following this protocol for comparison with earlier methods as it was used in previous papers.

On this dataset, to be able to compare with already published results, we report P_{rel} and R_{rel}, respectively the precision and recall relative to eHarris. As in [8] results are computed on the sequences `boxes_6dof`, `dynamic_6dof`, `poster_6dof` and `shapes_6dof`.

4.2 Ablation Study

N_H	δt-reprojection Error (pixels)					Track Lifetime (s)
	$\delta t = 25$	50	100	150	200	
1	1.47	1.62	1.91	1.95	2.62	15.63
2	1.27	1.47	1.66	1.81	2.06	**16.70**
3	1.25	1.32	1.60	1.67	1.79	16.44
5	1.26	1.31	1.47	1.82	1.85	16.14
10	1.18	1.28	1.45	1.63	1.84	15.7
12	**1.15**	**1.24**	**1.35**	**1.53**	**1.68**	15.4

Table 1: **Influence of N_H, the number of predicted heatmaps.** $N_H = 1$ corresponds to the standard approach with a single prediction for the entire integration period. While $N_H = 1$ already performs well thanks to our training procedure, increasing N_H consistently improves accuracy.

Predicting a single heatmap vs multiple heatmaps, and the influence of N_H. We evaluated the performance of our detector on the HVGA ATIS Corner Dataset when varying N_H, the number of predicted heatmaps. Setting $N_H = 1$ allows us to also evaluate the influence of predicting multiple heatmaps compared to a single one, as it is more traditionally done.

Table 1 reports the results. Predicting a single heatmap already performs well thanks to our training data. However, accuracy keeps improving when N_H increases. As computation time also increases with N_H, there is still a balance to find. We chose $N_H = 10$ for all the other experiments reported in this paper.

Influence of ΔT, the length of integration period. Our network is dependent on the integration period. Table 2 reports the δt-reprojection error and the track liftetime metric for various values of ΔT. Reducing ΔT too much reduces the performance as not enough events are given as inputs, which puts too much strain on the memory of such a small network. On the other hand, setting ΔT too high also reduces the precision of the predictions. A good trade-off is $\Delta t = 5ms$, and it is the value we used for all other experiments.

Integration Period ΔT	δt-reprojection Error (pixels)					Track Lifetime (s)
	$\delta t = 25$	50	100	150	200	
2.5ms	1.19	1.33	1.72	2.15	3.11	12.8
5ms	**1.18**	**1.28**	**1.45**	1.63	1.84	15.7
7.5ms	1.31	1.37	1.52	**1.52**	**1.54**	**15.8**
10ms	1.39	1.50	1.51	1.55	1.56	14.8
25ms	1.46	1.50	1.62	1.98	2.86	11.7
50 ms	1.55	1.92	2.28	3.32	3.81	10.8

Table 2: **Influence of ΔT, the length of integration period.** We use $\Delta T =$ 5ms in all the other experiments.

Generation of Training Data	δt-reprojection Error (pixels)					Track Lifetime (s)
	$\delta t = 25$	50	100	150	200	
Detecting Corners	1.34	1.59	1.99	2.19	2.31	2.4
Warping Corners	**1.18**	**1.28**	**1.45**	**1.63**	**1.84**	**15.7**

Table 3: **Influence of detecting or warping keypoints during training.** We compare two networks: *Detecting Corners* corresponds to a network trained with keypoints detected independently in each frame; *Warping Corners* corresponds to training with keypoints detected in the reference frame and then warped using the simulated homographies, as explained in Sec. 3.3.

4.3 Comparison to the State-of-the-Art

We compare ourselves to [1,2,8,15,16,22]. We have only limited results for [2,15] as the authors did not make their models available.

Table 4 reports the comparison results on the HVGA ATIS Corner Dataset. Our detector is able to beat previous methods both in terms of accuracy as given by the δt-homography reprojection error and of track lifetime metrics. Moreover, Figure 7 provides visual comparisons of the track lengths for these methods.

Method	δt-reprojection Error (pixels)					Track Lifetime (s)
	$\delta t = 25$	50	100	150	200	
eHarris [22]	2.57	3.46	4.58	5.37	6.06	0.74
eFast [16]	2.12	2.63	3.18	3.57	3.82	0.69
Arc [1]	3.80	5.31	7.22	8.48	9.49	0.91
SILC [15]	2.45	3.02	3.68	4.13	4.42	1.12
gradients [2]	2.46	-	-	-	-	5.46
ours	**1.18**	**1.28**	**1.45**	**1.63**	**1.84**	**15.7**

Table 4: **Evaluation on the HVGA ATIS Corner Dataset.** Our detector outperforms all previous methods in terms of δt-homography reprojection error for various δt, and of track lifetime metric. The reprojection errors for Gradients are not available for $\delta t > 25$. We nearly reduce by half the reprojection error while at the same time triple the tracks lifetime.

Table 5 reports the comparison results on The Event-Camera Dataset. While, as discussed at the beginning of this section, the results can be considered as biased, they show that our detector performs well on this other dataset captured with a different sensor.

Method	P_{rel}	R_{rel}
eHarris [22]	1	1
eFast [16]	0.68	0.55
Arc [1]	0.84	0.59
luvHarris [8]	0.86	0.97
ours	**1.03**	**2.17**

Table 5: **Evaluation on The Event-Camera Dataset**. The dataset was captured with a different sensor than the HVGA ATIS Corner dataset, nevertheless our detector also outperforms earlier work. Following the protocol of [8], we report P_{rel} and R_{rel}, the precision and recall relative to eHarris.

4.4 Qualitative results

Figure 5 shows how our network is able to predict heatmaps through time which are coherent with one another. The keypoint is correctly predicted at multiple locations for the same volumetric input despite long accumulation time which make the edges become thicker. Keypoint are correctly detected for easy corners on the left as well as more textured keypoint. The consistency through time is a great advantage of our method and make it possible to track corners using a simple nearest neighbor algorithm.

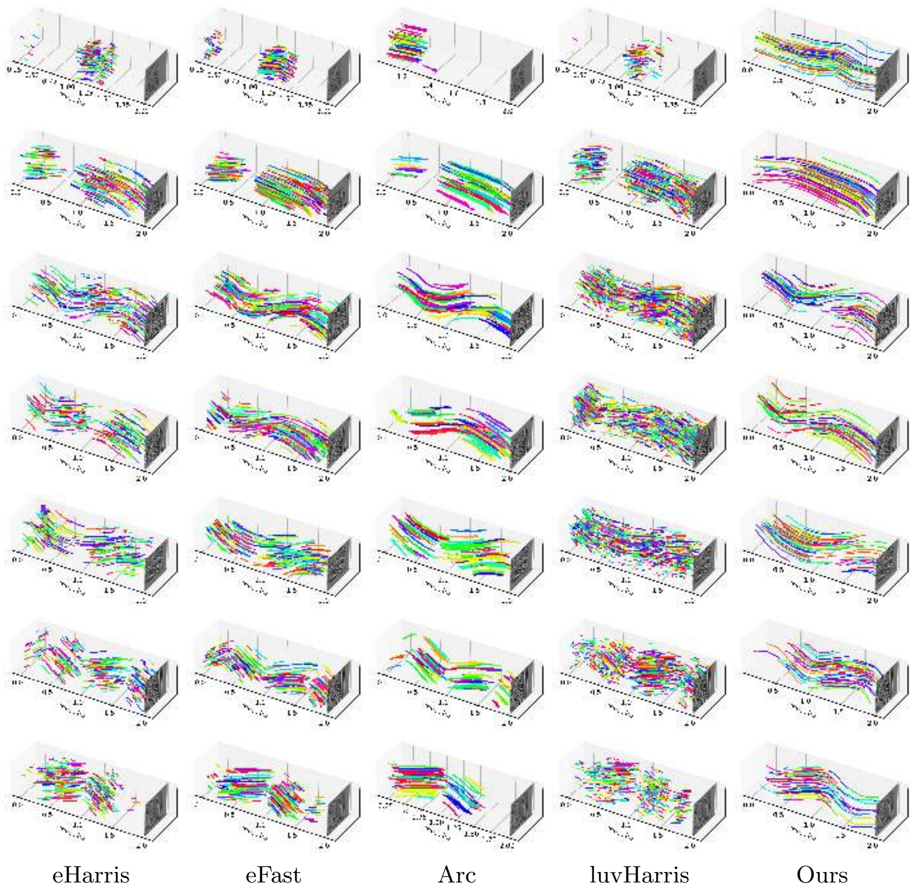


Fig. 7: **Visual comparisons of tracks obtained by different methods on the HVGA ATIS Corner Dataset.** The tracks obtained with our detector are significantly longer.

Similarly, Figure 7 demonstrate the homogeneous spatial distribution of our keypoints which create very long and smooth tracks. Our method helps with tracks continuity, they are seldomly lost or broken which is not the case for other methods such as eHarris, eFast, Arc or even luvHarris.

4.5 Computation Times and Memory Footprint

Our network has less than 26K parameters and runs in 7ms on a GTX 1080 GPU, for a HVGA input event sensor. The low footprint of the network coupled with the computation of the event cube enable running at arbitrarily high event rate. Our detector offers the possibility to reduce even more computations by augmenting the integration period and the number of predicted heatmaps at the cost of only minimal latency. This makes our novel approach very suitable for real-time applications.

5 Conclusion

We presented a novel keypoint detector in event streams, which provides long tracks of accurately localized keypoints. We will make our code available, and believe it will be useful for many downstream applications, such as SLAM and object tracking.

We also believe that predicting as we do multiple successive estimates rather than one for the integration period used by many methods is general and could be applied to many other computer vision problems on event-based cameras, such as segment detection,depth prediction and object detection.

References

1. Alzugaray, I., Chli, M.: Asynchronous corner detection and tracking for event cameras in real time. IEEE Robotics and Automation Letters **3**(4), 3177–3184 (2018)
2. Chiberre, P., Perot, E., Sironi, A., Lepetit, V.: Detecting Stable Keypoints from Events through Image Gradient Prediction. In: Conference on Computer Vision and Pattern Recognition Workshops (2021)
3. Delbruck, T., Hu, Y., He, Z.: V2E: From Video Frames to Realistic DVS Event Camera Streams. In: Conference on Computer Vision and Pattern Recognition Workshops (2021)
4. Detone, D., Malisiewicz, T., Rabinovich, A.: SuperPoint: Self-Supervised Interest Point Detection and Description. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops (2018)
5. Finateu, T., Niwa, A., Matolin, D., Tsuchimoto, K., Mascheroni, A., Reynaud, E., Mostafalu, P., Brady, F., Chotard, L., Legoff, F.: A 1280×720 Back-Illuminated Stacked Temporal Contrast Event-Based Vision Sensor with 4.86 μm Pixels, 1.066 GEPS Readout, Programmable Event-Rate Controller and Compressive Data-Formatting Pipeline. In: IEEE International Solid-State Circuits Conference-(ISSCC) (2020)
6. Gallego, G., Delbruck, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A., Conradt, J., Daniilidis, K., Scaramuzza, D.: Event-Based Vision: A Survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
7. Germain, H., Bourmaud, G., Lepetit, V.: S2DNet: Learning Image Features for Accurate Sparse-to-Dense Matching. In: European Conference on Computer Vision. pp. 626–643 (2020)
8. Glover, A., Dinale, A., Rosa, L.D.S., Bamford, S., Bartolozzi, C.: luvHarris: A Practical Corner Detector for Event-Cameras. IEEE Transactions on Pattern Analysis and Machine Intelligence (2021)
9. Graca, R., Delbruck, T.: Unraveling the Paradox of Intensity-Dependent DVS Pixel Noise. In: arXiv Preprint (Sep 2021)
10. Harris, C., Stephens, M.: A Combined Corner and Edge Detector. In: Alvey vision conference (1988)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep Residual Learning for Image Recognition. In: Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
12. Hu, J., Shen, L., Sun, G.: Squeeze-and-Excitation Networks. In: Conference on Computer Vision and Pattern Recognition. pp. 7132–7141 (2018)
13. Lichtsteiner, P., Posch, C., Delbruck, T.: A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. IEEE Journal of Solid-State Circuits pp. 566–576 (2008)
14. Lin, T.Y., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C.L., Dollár, P.: Microsoft COCO: Common Objects in Context (2014)
15. Manderscheid, J., Sironi, A., Bourdis, N., Migliore, D., Lepetit, V.: Speed Invariant Time Surface for Learning to Detect Corner Points with Event-Based Cameras. In: Conference on Computer Vision and Pattern Recognition. pp. 10245–10254, https://www.prophesee.ai/2019/06/05/hvga--atis--corner--dataset/ (2019)
16. Mueggler, E., Bartolozzi, C., Scaramuzza, D.: Fast Event-based Corner Detection. In: British Machine Vision Conference (2017)

17. Mueggler, E., Rebecq, H., Gallego, G., Delbruck, T., Scaramuzza, D.: The Event-Camera Dataset and Simulator: Event-Based Data for Pose Estimation, Visual Odometry, and SLAM. International Journal of Robotics Research pp. 142–149, https://rpg.ifi.uzh.ch/davis_data.html (2017)
18. Rebecq, H., Gehrig, D., Scaramuzza, D.: ESIM: An Open Event Camera Simulator. In: Conference on Robot Learning. pp. 969–982 (2018)
19. Rebecq, H., Ranftl, R., Koltun, V., Scaramuzza, D.: Events-To-Video: Bringing Modern Computer Vision to Event Cameras. In: Conference on Computer Vision and Pattern Recognition. pp. 3852–3861 (2019)
20. Scheerlinck, C., Rebecq, H., Gehrig, D., Barnes, N., Mahony, R., Scaramuzza, D.: Fast Image Reconstruction with an Event Camera. In: IEEE Winter Conference on Applications of Computer Vision. pp. 156–163 (2020)
21. Son, B., Suh, Y., Kim, S., Jung, H., Kim, J.S., Shin, C., Park, K., Lee, K., Park, J., Woo, J.: A 640$\times$ 480 Dynamic Vision Sensor with a 9μm Pixel and 300Meps Address-Event Representation. In: IEEE International Solid-State Circuits Conference (ISSCC) (2017)
22. Vasco, V., Glover, A., Bartolozzi, C.: Fast Event-Based Harris Corner Detection Exploiting the Advantages of Event-Driven Cameras. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 4144–4149 (2016)
23. Zhu, A.Z., Yuan, L., Chaney, K., Daniilidis, K.: Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion. In: Conference on Computer Vision and Pattern Recognition. pp. 989–997 (2019)