HAL
open science

Detecting Stable Keypoints from Events through Image Gradient Prediction

Philippe Chiberre, Etienne Perot, Amos Sironi, Vincent Lepetit

To cite this version:

Philippe Chiberre, Etienne Perot, Amos Sironi, Vincent Lepetit. Detecting Stable Keypoints from Events through Image Gradient Prediction. International Conference on Computer Vision and Pattern Recognition Workshop, Jun 2021, Online, United States. 10.1109/CVPRW53098.2021.00153 . hal-03482236

HAL Id: hal-03482236

https://hal.science/hal-03482236

Submitted on 15 Dec 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

This CVPR 2021 workshop paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Detecting Stable Keypoints from Events through Image Gradient Prediction

Philippe Chiberre[1,2] Etienne Perot[1] Amos Sironi[1] Vincent Lepetit[2]
[1]PROPHESEE Paris, France
[2] LIGM, Ecole des Ponts, Univ Gustave Eiffel, CNRS Marne-la-vallée, France
{pchiberre, eperot, asironi}@prophesee.ai vincent.lepetit@enpc.fr

Abstract

We present a method that detects stable keypoints from an event stream at high speed with a low memory footprint. Our key observation connects two points: It should be easier to reconstruct the image gradients rather than the image itself from the events, and the Harris corner detector, one of the most reliable keypoint detectors for short baseline regular images, depends on the image gradients, not the image. We therefore introduce a recurrent convolutional neural network to predict image gradients from events. As image gradients and events are correlated, this prediction task is relatively easy and we can keep this network very small. We train our network solely on synthetic data. Extracting Harris corners from these gradients is then very efficient. Moreover, in contrast to learned methods, we can change the hyperparameters of the detector without retraining. Our experiments confirm that predicting image gradients rather than images is much more efficient, and that our approach predicts stable corner points which are easier to track for a longer time compared to state-of-the-art event-based methods.

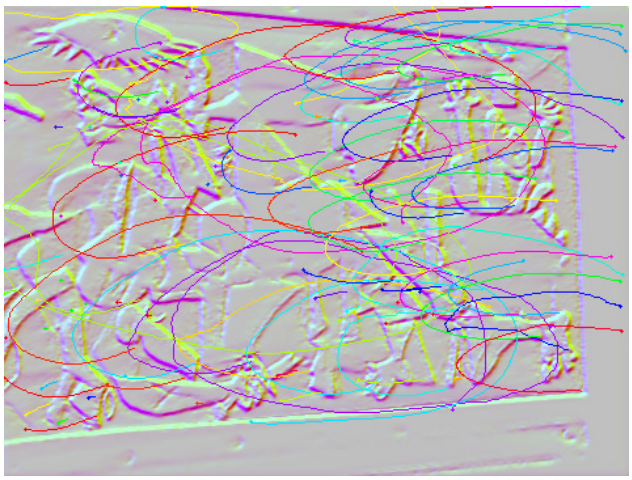


Figure 1. To detect keypoints in a stream of events, we propose to first predict the image gradients from the events using a small recurrent network. Then, we apply the Harris corner detector rule on the predicted gradients. Finally, we track the corners by simple nearest neighbor matching. The figure shows the predicted gradients with overlaid tracked corners (best seen in electronic format).

1. Introduction

Corner detection is an important computer vision problem, with downstream applications in Simultaneous Localization and Mapping (SLAM), Structure from Motion, object recognition and tracking [30, 40, 14, 37]. Event cameras [16, 35, 8] are sensors that capture visual information with high dynamic range and high temporal resolution, while maintaining power consumption and data rate acceptable for real time applications on embedded platforms [9]. As a consequence, reliable and efficient corner detection on event cameras enables vision pipelines to work in challenging illumination conditions and with very fast movements [39], situations where conventional frame-based sensors would require heavy computation and introduce latency.

For these reasons, corner detection has been among the first problems studied in even-based vision [4, 38, 21, 1]. Early approaches rely on hand-crafted rules inspired by frame-based corner detection algorithms [38, 21]. However, it is hard to define a single rule which is robust to the variability and noise patterns of event camera data. To overcome this, the authors of [20] propose a machine learning approach to classify corner events. This method is more robust than the hand-crafted ones and can still be applied event-by-event. However, to be real time for increasing input event rates, only a simple random forest model is used, which has limited generalization and expressive power. Moreover, it is not possible to change hyperparameters to control the keypoint detection: Hand-crafted methods for keypoint detections often depend on hyperparameters that are easy to change to adapt to the input images. Detectors using machine learning, random forest, or deep networks, do not have such hyperparameters that can be tuned, and multiple models would need to be trained instead.

Our approach to developing a keypoint detector for event streams that can be tuned while relying on the performance of deep learning is motivated by the recent work presented in [25]. [25] showed that it is possible to reconstruct high-quality gray-level images from events. They also show that applying standard gray-level visual-inertial odometry methods [23] on the images generated by their network gives more accurate results than state-of-the-art event-based methods [39]. The limitation of this approach is that they need a large neural network to reconstruct accurate gray-level images. Thus, their approach is not well adapted for low power and real-time applications—which are usually the motivation for using event-based cameras.

We observe that many gray-level corner detectors rely on image derivatives rather than intensity values [11, 33, 18, 19]. Moreover, since event cameras are sensitive to illumination changes, the events they produce are directly related to the spatio-temporal gradients of the scene.

We therefore propose to train a recurrent neural network to predict image gradients from events rather than intensity values, and apply the Harris detection using these gradients rather than gradients computed from image intensities (Figure 1). This approach has many advantages:

- Since predicting the gradients from events is a simpler task than predicting images, we can use a very light recurrent neural network.
- The Harris detector [11] remains a method of choice for short baseline matching, which is the target application of previous keypoint detection methods for event-based cameras. Note that the popular Good Features to Track method from [33] is essentially the same as the Harris detector. The only difference is that [33] relies on the eigenvalues of the auto-correlation matrix, while [11] proposes a score that avoids computing the eigenvalues, which can be relatively costly.
- By contrast with [20], we do not require a training set of event streams annotated with keypoints, which is difficult to build. We only need a video from which we can obtain events by using a standard simulator [24, 10].

We evaluate our approach on a standard event-based datasets [20], showing comparable accuracy than previous method. Our method can be combined with most event-based tracking algorithms, here we show that a very simple tracking rule, based on nearest neighbor matching, already gives much longer tracks compared to the state-of-the-art.

2. Related Work

2.1. Keypoint Detection in Images

Keypoint detection has a long history in computer vision, as it is very important for many downstream applications. Some detectors are designed for short baseline matching of images [34, 11, 33, 29], others for wide baseline matching [19, 2, 42, 7, 6], as the requirements are not the same depending on the exact target application.

2.2. Keypoint Detection in Event Streams

Handcrafted methods. [38] adapts the Harris corner detector to event streams by creating a binary image created by the last n events in the frame. This image is updated at every new event and the Harris score is computed on this binary image. There exists a number of other methods detecting points of interest via local pattern matching [4, 22, 15]. Unfortunately, these methods are very sensitive to noise and require careful tuning of the parameters.

Several existing detectors are based on a representation called the Surface of Active Events [3], which is constructed from past events and is structured as a pair of 2D images. Given the 2D location (x, y), polarity p, and time stamp of an upcoming event, the Surface of Active Events stores the time stamp at the location and polarity of the event:

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

This representation makes easier the adaptation of handcrafted corner detectors originally developed for regular cameras to event-based cameras: The Fast Event-based Corner Detection method [21] analyzes the distribution of timestamps around the latest event. If they are divided into two clear regions of old and new timestamps the event is considered a corner. [1] introduced an improved version of [21] changing the SAE to a "more restrictive SAE" by filtering redundant and noisy events. It improves the representation of high contrast regions and reduces the amount of computation of the following steps.

Those methods are computed for every new event and can therefore be a bottleneck when the event rate increases dramatically which can happen with new higher resolution sensors.

Learned methods. In contrast to image-based corner detection, to the best of our knowledge, there is only one method based on machine learning for corner detection in event streams: [20] relies on a Random Forest to learn to discriminate corners from non corners events. The Random Forest uses tests on a variant of the Time Surface that depends less of the firing time of the events compared to the standard Time Surface. This method suffers from the same problem as handcrafted methods with the added computation of the speed invariant time surface and the random forest making it too slow for real sensors.

2.3. Reconstructing Events from Images

Our approach is based on the reconstruction of the image gradients from the events. Several methods have already

been proposed for the reconstruction of the image itself. The first efforts in video and gradients reconstruction were made by [36] using a K-SVD algorithm to learn patch-based dictionary atoms. [31] introduced asynchronous spatial image convolutions to reconstruct an image from events. The linear spatial kernel acts as an internal state and reconstructs gray-level images. They also produce gradients and detect corners with the Harris score but do not show quantitative results on known datasets.

[25] on the other hand trained an architecture called E2VID inspired by U-Net [28] on a large amount of simulated data, which was later improved in [26]. E2VID was also revisited in FireNet [32] improving vastly on the speed and memory footprint of the model. We take inspiration from their model as well as their approach for our paper. However, as we do not require the image itself but the gradients which are closer in nature to the events, we can reduce even further the architecture to a minimal five layer convolutional recurrent neural network.

3. Method

Let us consider an event camera of $H \times W$ pixels. Let $L(x, y, t)$, be the light intensity at pixel (x, y) and time $t \geq 0$. The pixels in the event camera will not record absolute intensity, but will send an output *event* as soon as they detect a big enough change of L. Formally, given a contrast threshold θ, an event $e_i = (x_i, y_i, p_i, t_i)$ is generated for pixel (x_i, y_i) if

$$\left| \log \left(\frac{L(x_i, y_i, t_i)}{L(x_i, y_i, t_{i-1})} \right) \right| \geq \theta \,, \tag{2}$$

where t_{i-1} is the time of the last event at (x_i, y_i) and p_i, called polarity of the event, is the sign of the contrast change:

$$p_i = \text{sign} \left(\log \left(\frac{L(x_i, y_i, t_i)}{L(x_i, y_i, t_{i-1})} \right) \right) \,. \tag{3}$$

Let $\mathbf{e} = \{e_i\}_{i=1}^N$ be a generic sequence of events generated by the camera in time interval $[t_1, t_N]$. Our goal is to find a function $\mathcal{F}$ mapping an event sequence $\mathbf{e}$ to the corresponding gradient image $G \in \mathbf{R}^{H \times W \times 2}$ at time t_N, where $G(x, y, t_N)$ is given by

$$\begin{aligned} G(x, y, t_N) &= \nabla_{xy} L(x, y, t_N) \\ &= \left(L_x(x, y, t_N), L_y(x, y, t_N) \right)^\top \,. \end{aligned} \tag{4}$$

For simplicity of notation, we will omit the dependency on t_N in the following.

In [31], $\mathcal{F}$ is implemented as a model-based filtering method. This approach has the advantage of being easily interpretable. However, since it is a local method with a very limited memory mechanism, it can not reach accurate enough results when applied to noisy real world data. By contrast, we use a recurrent convolutional neural network which gives better results. The advantage of using a deep learning approach is that we can directly learn the best function $\mathcal{F}$ from the data being robust to its variability. Moreover, by using ConvLSTM layers [41], our model can handle long spatio-temporal dependencies on the events.

Once G is computed, it is straightforward to estimate corners locations using Harris rule. We first compute the structure tensor M as

$$M = w_\sigma * (G \cdot G^\top) \,, \tag{5}$$

where w_σ is a Gaussian kernel of standard deviation σ. For each image location, G is a 2D vector, so M can be seen as a 2×2 matrix for each image location. Following Harris detection method, we then compute the score map:

$$S = \det(M) - k \cdot \text{tr}(M)^2 \,. \tag{6}$$

Corners locations are given by the local maxima of S that are above a given threshold. Note that we can change hyperparameters σ and k to tune keypoint detection without having to retrain our network.

In the following, we detail our architecture for $\mathcal{F}$ and how we train it.

3.1. Architecture

As mentioned in the previous section, we learn the function $\mathcal{F}$, predicting the image gradients from events, as a recurrent neural network. The architecture we use is shown in Figure 2. It is a 5-layer fully convolutional network of 3×3 kernels. Each layer has 12 channels and residual connections [12]. The second and fourth layers are ConvLSTM, the last layer predicting the gradients is a standard convolutional layer, while for the remaining feed-forward ones we use Squeeze-Excite (SE) connections [13]. This results in a very efficient architecture, but still able to learn gradients from event data. In the following, we describe how we train our network and the events representation used as input.

3.2. Training

For training, we use a subset of the training images in the COCO dataset [17]. As shown in Figure 3, for each image, we create a smooth video sequence from random homographies simulating camera movement in front of a planar scene and simulate events from said video. We draw the contrast threshold from a uniform distribution in [0.01, 0.2]. Each image yields a 5000 frame video. Every 5 frames, we generate events with our own simulator, which is based on a combination of [24] and [5]. From the events, we build an event cube (see Section 3.4) composed of 5 channels. At each iteration, we use a batch made of 8 different sequences of 20 time bins. The input size

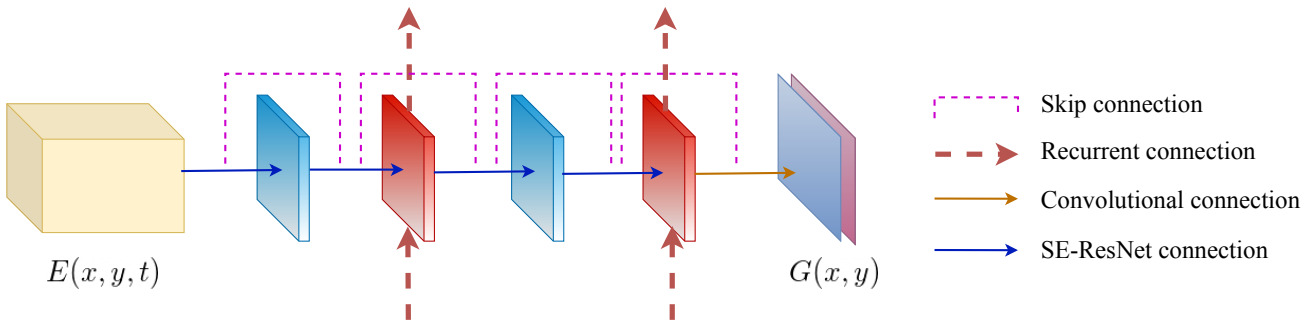


Figure 2. **Proposed Architecture.** Our architecture combines recurrent connections with convolutional and Squeeze-and-Excite ResNet connections. The input to the network is given by an event cube $E(x, y, t)$ [43], computed at every N events. The event cube is given as input to a Squeeze and Excite ResNet connection, followed by a ConvLSTM with residual connection. This block of two layers is repeated once. Finally, a simple convolutional layer is used to predict the gradients $G(x, y)$.

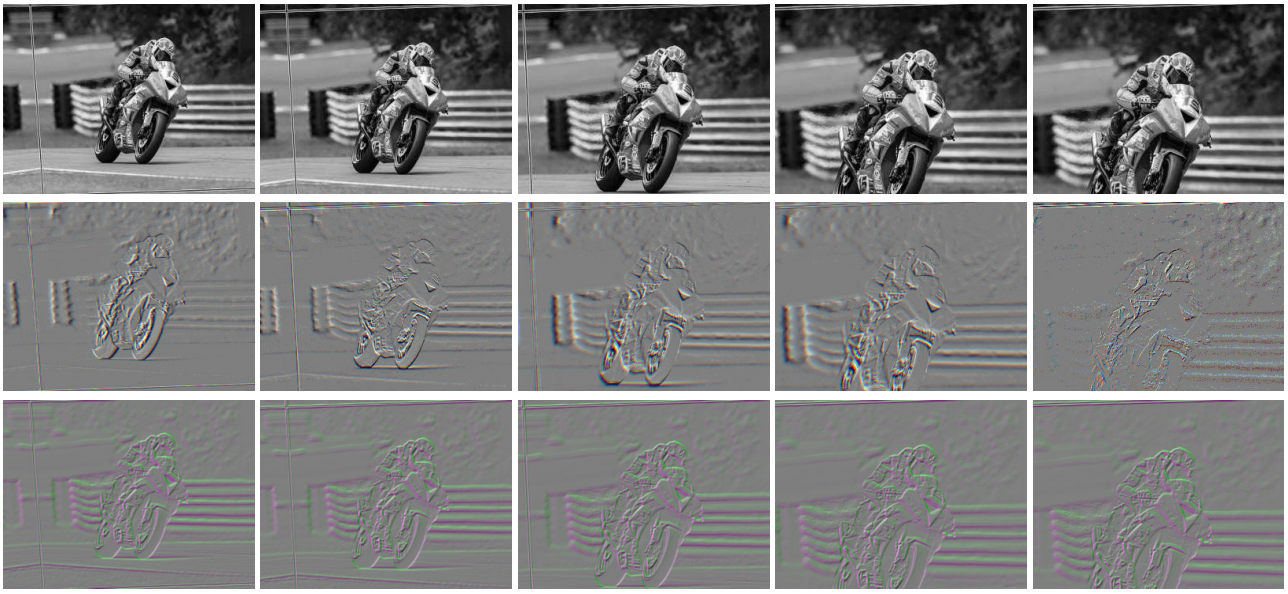
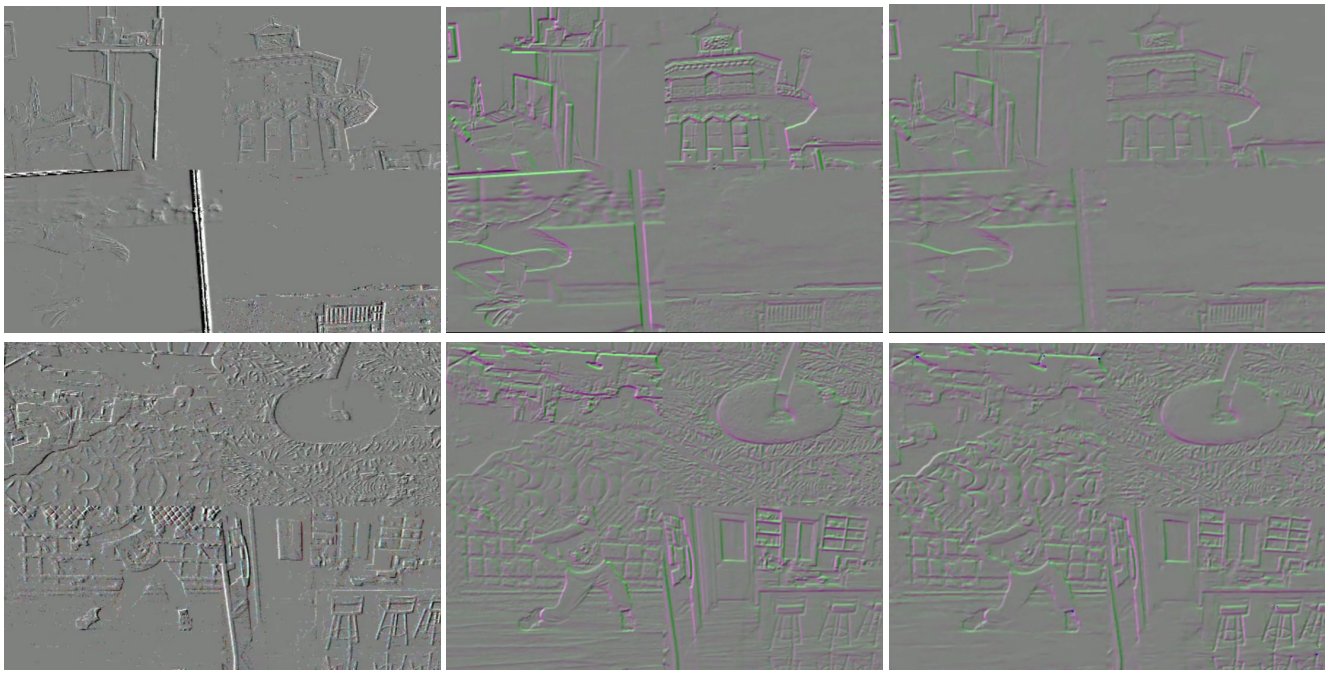


Figure 3. **Generating Training Data.** First row: Given a training image, we apply homographies to warp it and generate a video sequence. Second row: From the video sequence we generate events using a simulator and then compute the event cubes [43] used as input to our network. Third row: Ground-truth gradients of the video sequence. We train the architecture of Figure 2 to predict image gradients from events. Finally, at inference time, we compute the Harris score from the predicted gradients.

is $(T, Ba, B, H, W) = (20, 8, 5, 320, 240)$. For each input tensor, we have a corresponding image. We use it to compute its spatial gradient using kornia [27]. Our loss is simply the smooth L1 metric between the spatial gradient computed using the Sobel kernels and the 2 channels output of our network. We use truncated-backpropagation-through-time of 20 time steps, detaching the state at each batch never resetting for each video. We train for 10 epochs with a learning rate of $1e^{-4}$.

3.3. Inference

At inference, we build an event cube (see Section 3.4) and feed it to our network. The event cube depends on a hyperparameter N (the length of the sequence of events used to predict the image gradients), if it is large the quality of the tracks will improve, but the network might be a bottleneck. The network outputs a prediction of the image gradients. In practice, we notice that some values are erroneously large, and we clamp all values exceeding 3 times the standard deviation of the output values. Then, we compute the Harris score as given in Eq. (6) using the kornia [27] implementation. We simply extract the local maxima of this score map as keypoints. Example of predicted gradients and corresponding events and ground-truth are shown in Figure 4.



Input Event Cube | Ground-truth Image Gradients | Predicted Image Gradients

Figure 4. **Predicting Image Gradients.** Events from a camera are directly correlated to the spatio-temporal gradients of the scene. For this reason, in our method, we train a small neural network to predict image gradients from events. The gradients can be directly used to estimate corner locations using the Harris rule.

3.4. Input Representation

The input to our network is a $H \times W \times B$ event tensor $E(x, y, t)$, as proposed in [43]. H, W are the image sensor height and width respectively and B is the number of temporal bins. In the event tensor computation, each input event (x_i, y_i, t_i, p_i) contributes by its polarity to the two closest temporal bins using a triangular kernel. More formally, E is computed as

$$E(x, y, t_n) = \sum_i p_i \max(0, 1 - |t_n - t_i^\star|) \,, \qquad (7)$$

with

$$t_i^\star = \frac{(t_i - t_{min})}{(t_{max} - t_{min})}(B - 1) \,, \qquad (8)$$

and where n is the temporal bin index, p_i is the polarity, and $t_i^\star$ is the normalized timestamp of the ith event. In practice, we use $B = 5$.

At runtime, we select the N latest events to create the event cube. Running by N events enables us to follow the rate of the event stream naturally and avoid useless computation when there are no new incoming events.

4. Results

4.1. Quantitative Results

For evaluating our method, we consider the ATIS Corner Dataset [20], which is specifically designed to evaluate event-based corner detection and tracking methods. This dataset is composed of 7 sequences of planar scenes acquired with a HVGA event sensor. We use the same evaluation metrics as in [20], that is we compute reprojection error by estimating a homography form the tracked corners. As in [20], we also consider average track length.

For tracking, we use a very simple nearest neighbor rule: Two corners are assigned to the same track if their distance in pixel and in timestamp is lower than a threshold.

The results are shown in Table 1. As can be seen, our method has the best tracking length, almost 5 times longer than the second best method [20]. This is thanks to the memory of the recurrent layers which can stabilize the detection and also to the robustness of the Harris detector.

Our method has slightly worse reprojection error. This is probably due to the smoothing effect introduced by the network on the predicted gradients. The other methods suffer less from this problem, since they operate event by event.

4.2. Qualitative Results

Figure 4 compares ground truth gradients with gradients predicted by our architecture, and shows the predicted gradients are very similar to the real ones.

Figure 5 compares Harris keypoints extracted using gradients predicted with our recurrent architecture and Harris keypoints extracted using gradients estimated by the hand-crafted method [31]. Only the gradient computation is dif-

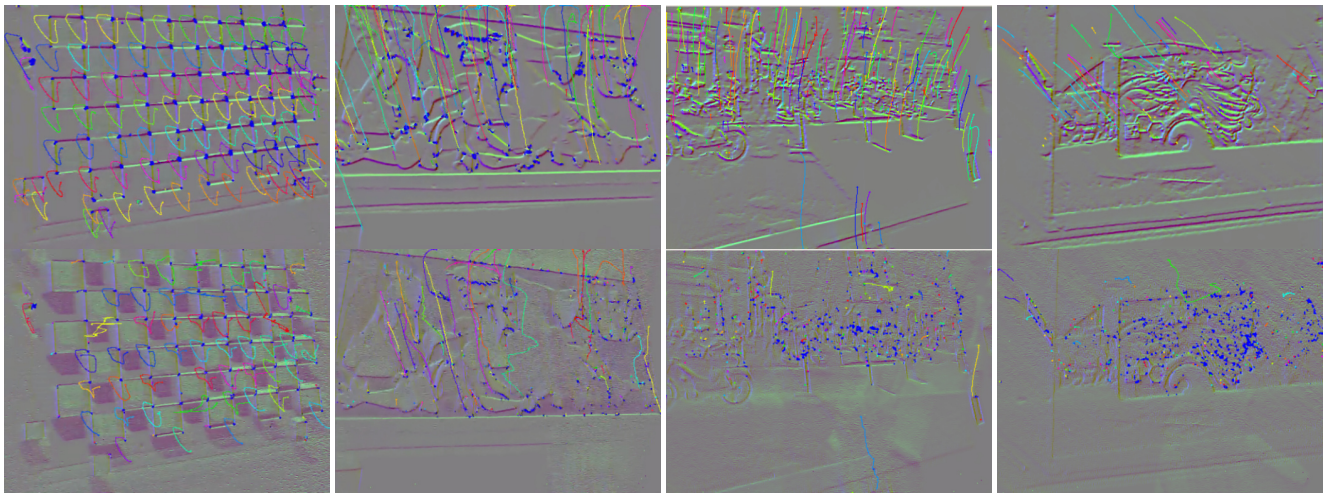


Figure 5. **Qualitative Results on the ATIS Corner Dataset.** First row: Gradients predicted by our network, with overlaid tracks returned by our method. Second row: Results obtained by replacing our network by the method of [31], which predicts gradients from events by using a model-based approach. As we can see, our gradients are better localized. Moreover, thanks to the recurrent layers of the network, we do not have a trailing effect on the gradients. As a consequence, we can track more corner points, more accurately.

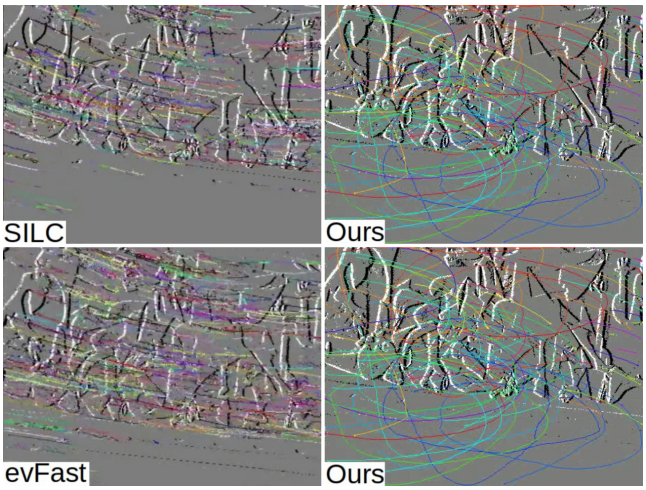


Figure 6. **Qualitative comparison to previous state-of-the-art methods.** First row: we visually compare our method with the previous state-of-the art SILC [20]. Second row: Our method is compared to evFast method [21]. Here we visualize the events with different polarities for an easier comparison but our method still computes corners on the predicted gradients (best seen in electronic format).

ferent, all the other computations are the identical. Our method predicts much more accurate gradients, yielding much more stable keypoints.

Figure 6 shows a visual comparison of our method with previous state-of-the art. Our stable gradient prediction enables the corner tracks to be longer and more accurate than other methods.

Table 1. Evaluation on the ATIS Corner Dataset [20] for $\Delta t = 25ms$. Our method has 5 times longer tracks, while maintaining similar reprojection error as the state-of-the-art.

	evHarris [38]	evFast [21]	Arc [1]	$SILC^*$ [20]	Ours
Reprj. error (pix)	2.57	**2.12**	3.8	2.45	2.56
Track length (sec)	0.74	0.69	0.91	1.12	**5.46**

4.3. 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. For comparison, the network of [25] reconstructing graylevel intensities runs in 35ms and has more than 5.1M parameters. This low footprint enables fast computation of corners and makes our approach suitable for real-time applications.

5. Conclusion

In this paper, we proposed to predict image gradients from events for keypoint detection, as it is a relatively easy task for a deep network and the Harris corner detector, an attractive option for short baseline keypoint detection, depends only on the image gradients rather than the image itself. More generally, many image analysis methods developed for regular cameras depend on image derivatives rather than the image intensities, such as the SIFT descriptor and the SIFT detector (the Difference-of-Gaussians in SIFT approximates the Laplacian-of-Gaussian *i.e.* the sum of the second order derivatives). We therefore believe that,

beyond the Harris detector, our approach could be extended to other image analysis tasks by relying on methods developed for regular cameras.

In fact, the first layer of a deep network trained on regular images often learns to extract oriented gradients from the input image. It should be possible to learn to predict such oriented gradients and reuse architectures trained on regular images. Thus we could reuse architectures trained on datasets of regular images by applying them from their second layer on the predicted gradients. This would make possible the use of the many datasets created for regular images and their annotations for applications using event-based cameras.

References

[1] Ignacio Alzugaray and Margarita Chli. Asynchronous Corner Detection and Tracking for Event Cameras in Real Time. *IEEE Robotics and Automation Letters*, 2018. 1, 2, 6

[2] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc J. Van Gool. SURF: Speeded Up Robust Features. *Computer Vision and Image Understanding*, 10(3):346–359, 2008. 2

[3] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-Based Visual Flow. *IEEE Transactions on Neural Networks and Learning Systems*, 25(2), 2014. 2

[4] Xavier Clady, Sio-Hoi Ieng, and Ryad Benosman. Asynchronous Event-Based Corner Detection and Matching. *Neural Networks*, 2015. 1, 2

[5] Tobi Delbruck, Yuhuang Hu, and Zhe He. V2E: From video frames to realistic DVS event camera streams. *arxiv*, June 2020. 3

[6] Daniel Detone, Tomasz Malisiewicz, and Andrew Rabinovich. SuperPoint: Self-Supervised Interest Point Detection and Description. In *Conference on Computer Vision and Pattern Recognition*, June 2018. 2

[7] Mihai Dusmanu, Ignacio Rocco, Tomas Pajdla, Marc Pollefeys, Josef Sivic, Akihiko Torii, and Torsten Sattler. D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2

[8] Thomas Finateu, Atsumi Niwa, Daniel Matolin, Koya Tsuchimoto, Andrea Mascheroni, Etienne Reynaud, Pooria Mostafalu, Frederick Brady, Ludovic Chotard, and Florian Legoff. 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. 1

[9] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Joerg Conradt, Kostas Daniilidis, and Davide Scaramuzza. Event-Based Vision: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 1

[10] Daniel Gehrig, Mathias Gehrig, Javier Hidalgo-Carrió, and Davide Scaramuzza. Video to Events: Bringing Modern Computer Vision Closer to Event Cameras. In *arXiv Preprint*, 2019. 2

[11] Chris Harris and Mike Stephens. A Combined Corner and Edge Detector. In *Alvey vision conference*, 1988. 2

[12] Kaiming He, X. Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Conference on Computer Vision and Pattern Recognition*, 2016. 3

[13] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-Excitation Networks. In *Conference on Computer Vision and Pattern Recognition*, 2018. 3

[14] Kanghun Jeong and Hyeonjoon Moon. Object Detection Using FAST Corner Detector Based on Smartphone Platforms. In *ACIS/JNU International Conference on Computers, Networks, Systems and Industrial Engineering*, 2011. 1

[15] Xavier Lagorce, Cédric Meyer, Sio-Hoi Ieng, David Filliat, and Ryad Benosman. Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking. *IEEE Transactions on Neural Networks and Learning Systems*, 26(8), 2015. 2

[16] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128× 128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor. *IEEE Journal of Solid-State Circuits*, 43(2), Feb. 2008. 1

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, Lubomir Bourdev, Ross Girshick, James Hays, Pietro Perona, Deva Ramanan, C. Lawrence Zitnick, and Piotr Dollar. Microsoft COCO: Common Objects in Context. 3

[18] Tony Lindeberg. Feature Detection with Automatic Scale Selection. *International Journal of Computer Vision*, 30(2), 1998. 2

[19] D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 20(2):91–110, 2004. 2

[20] Jacques Manderscheid, Amos Sironi, Nicolas Bourdis, Davide Migliore, and Vincent Lepetit. Speed Invariant Time Surface for Learning to Detect Corner Points with Event-Based Cameras. In *Conference on Computer Vision and Pattern Recognition*, 2019. 1, 2, 5, 6

[21] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast Event-Based Corner Detection. In *British Machine Vision Conference*, 2017. 1, 2, 6

[22] Zhenjiang Ni, Sio-Hoi Ieng, Christoph Posch, Stépahen Régnier, and Ryad Benosman. Visual Tracking Using Neuromorphic Asynchronous Event-Based Cameras. *Neural Computation*, 27(4), 2015. 2

[23] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator. *IEEE Transactions on Robotics*, 34(4), 2018. 2

[24] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. ESIM: An Open Event Camera Simulator. In *Conference on Robot Learning*, 2018. 2, 3

[25] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. Events-To-Video: Bringing Modern Computer Vision to Event Cameras. In *Conference on Computer Vision and Pattern Recognition*, 2019. 2, 3, 6

[26] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. High Speed and High Dynamic Range Video

with an Event Camera. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019. 3

[27] Edgar Riba, Dmytro Mishkin, Daniel Ponsa, Ethan Rublee, and Gary Bradski. Kornia: An Open Source Differentiable Computer Vision Library for PyTorch. In *Winter Conference on Applications of Computer Vision*, 2020. 4

[28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. *MICCAI*, May 2015. 3

[29] Edward Rosten, Reid Porter, and Tom Drummond. Faster and Better: A Machine Learning Approach to Corner Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32:105–119, 2010. 2

[30] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An Efficient Alternative to SIFT or SURF. In *International Conference on Computer Vision*, 2011. 1

[31] Cedric Scheerlinck, Nick Barnes, and Robert Mahony. Asynchronous Spatial Image Convolutions for Event Cameras. *IEEE Robotics and Automation Letters*, 4(2), 2019. 3, 5, 6

[32] Cedric Scheerlinck, Henri Rebecq, Daniel Gehrig, Nick Barnes, Robert Mahony, and Davide Scaramuzza. Fast Image Reconstruction with an Event Camera. In *IEEE Winter Conference on Applications of Computer Vision*, 2020. 3

[33] Jianbo Shi and Carlo Tomasi. Good Features to Track. In *Conference on Computer Vision and Pattern Recognition*, 1994. 2

[34] S. M. Smith and J. M. Brady. Susan – A New Approach to Low Level Image Processing. Technical report, Oxford University, 1995. 2

[35] Bongki Son, Yunjae Suh, Sungho Kim, Heejae Jung, Jun-Seok Kim, Changwoo Shin, Keunju Park, Kyoobin Lee, Jinman Park, and Jooyeon Woo. 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. 1

[36] Barua Souptik, Miyatani Yoshitaka, and Veeraraghavan Ashok. Direct face detection and video reconstruction from event cameras. In *2016 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 1–9, 2016. 3

[37] Prithiraj Tissainayagam and David Suter. Object Tracking in Image Sequences Using Point Features. *Pattern Recognition*, 38(1), 2005. 1

[38] Valentina Vasco, Arren Glover, and Chiara Bartolozzi. Fast Event-Based Harris Corner Detection Exploiting the Advantages of Event-Driven Cameras. In *International Conference on Intelligent Robots and Systems*, 2016. 1, 2, 6

[39] 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 Robotics and Automation Letters*, 3(2), 2018. 1, 2

[40] Matt Westoby, J. Brasington, Neil F. Glasser, Michael J. Hambrey, and John M. Reynolds. Structure-from-Motion Photogrammetry: A Low-Cost, Effective Tool for Geoscience Applications. *Geomorphology*, 179, 2012. 1

[41] S. H. I. Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems*, 2015. 3

[42] K. M. Yi, Eduard Trulls, Vincent Lepetit, and Pascal Fua. LIFT: Learned Invariant Feature Transform. In *European Conference on Computer Vision*, pages 467–483, 2016. 2

[43] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. Unsupervised Event-Based Learning of Optical Flow, Depth, and Egomotion. In *Conference on Computer Vision and Pattern Recognition*, 2019. 4, 5