Détection de points d'intérêts avec des caméras neuromorphiques

Philippe Chiberre

École des Ponts ParisTech

THÈSE DE DOCTORAT
de l'École des Ponts ParisTech

Détection de points d'intérêts avec des caméras neuromorphiques

ED MSTIC n°532

Doctorat en informatique (Signal, Image, Automatique)

Thèse préparée au sein du laboratoire IMAGINE, LIGM

Thèse soutenue le 12/12/2022, par
Philippe CHIBERRE

Composition du jury :

Franck, DAVOINE Directeur de recherche, UTC Compiègne	*Président*
Guillermo, GALLEGO Professeur, Technische Universität Berlin	*Rapporteur*
Arren, GLOVER Chargé de recherche, Instituto Italiano di Tecnologia	*Examinateur*
Cornelia, FERMULLER Research Scientist, University of Maryland	*Examinatrice*
Vincent, LEPETIT Professeur, Ecole des Ponts ParisTech	*Directeur de thèse*
Amos, SIRONI Docteur, Prophesee	*Co-Directeur de thèse*

LEARNING TO DETECT KEYPOINTS WITH AN EVENT-BASED CAMERA

Philippe Chiberre

Doctoral thesis in the domain of signal and image processing supervised by Vincent LEPETIT and Amos SIRONI

Presented on 12/12/2022 to a committee consisting of:

Vincent Lepetit	École des Ponts ParisTech	Supervisor
Amos Sironi	Prophesee	Co-Supervisor
Franck Davoine	Université de technologie de Compiègne	Reviewer
Guillermo Gallego	Technische Universität Berlin	Reviewer
Arren Glover	Istituto Italiano di Tecnologia	Examiner
Cornelia Fermuller	University of Maryland	Examiner

Abstract

Keypoint detection is a building block of many computer vision applications, such as augmented or virtual reality and robotics. On the other hand, event cameras, or neuromorphic cameras, have numerous advantages for mobile platforms such as low power consumption, high dynamic range and very fine temporal resolution (in the order of the microsecond). The goal of this thesis is to develop keypoint detection algorithms for event cameras and enable multiple computer vision applications to be easily use with events.The first part of the thesis focuses on how data can be generated for any data driven approach when using event while still leveraging the vast computer vision literature for frames. The second part of the thesis introduces our approach of detecting keypoints using an intermediate representation. We predict image gradients from events with a convolutional recurrent neural network and detect keypoints using the Harris formula directly on the predicted gradients. Our approach maintains similar reprojection error to the state-of-the-art while improving significantly the track lifetime of the detected keypoints. The last contribution of this thesis is to predict keypoints directly from events without any intermediate step. We reduce the state-of-the-art reprojection error and triple the track lifetime using a data driven approach of predicting multiple heatmaps of keypoint localization's and improving our training pipeline for better coherence of the keypoints through time.

Résumé

Les points clés sont à l'origine de multiples applications de vision par ordinateur telles que la réalité virtuelle ou augmentée et la robotique. Les caméras par événements, ou caméras neuromorphiques, ont par ailleurs de nombreux avantages pour les plateformes mobiles : une très faible consommation d'énergie, une grande plage dynamique et une résolution temporelle très faible (de l'ordre de la microseconde). L'objectif de cette thèse et de développer des algorithmes de détection de points clés pour les caméras par événements et permettre le transfère des algorithmes de vision historiques pour les images vers cette nouvelle plateforme. La première partie de la thèse présente les différentes approches possibles pour générer des données de points clés dans des flux d'événements tout en utilisant la vaste littérature de vision par ordinateur existante pour les images. La deuxième partie de la thèse présente un nouvel algorithme permettant de détecter des points d'intérêts en utilisant une représentation intermédiaire. En effet nous utilisons un réseau de neurones convolutionels et récurrent pour prédire les gradients de l'image correspondante aux événements. Nous utilisons ensuite le score de Harris qui utilise les gradients de l'image pour détecter des points d'intérêts. Cet algorithme obtient une erreur de reprojection de l'ordre de l'état de l'art tout en augmentant le temps de vie des trajectoires. Notre dernière contribution est un algorithme qui prédit directement les points d'intérêts depuis les événements sans représentation intermédiaire. Cette méthode, basée sur les données, réduit l'erreur de rerojection de l'état de l'art tout en multipliant par trois la longueur des trajectoires. Ces améliorations proviennent de deux éléments clés: une amélioration de la plateforme d'entraînement pour une meilleure cohérence temporelle des points d'intérêts et une prédiction de plusieurs cartes de chaleur simultanément pour une meilleure précision.

Remerciements

Je souhaite tout d'abord remercier Vincent et Amos qui m'ont fait confiance pour cette thèse. Vos conseils, votre expérience et votre bienveillance m'ont permis d'achever ce doctorat en toute sérénité, et ce, malgré une pandémie mondiale. Merci d'avoir été disponible et d'avoir su m'accompagner dans les moments plus difficiles.

Je voudrais remercier Etienne qui a su, lors de nombreuses discussions, m'aider et me guider fort de son intuition et de sa curiosité. Ton travail et ta persévérance sont un exemple pour moi.

Merci à Stéphane Laveau, Stéphane Valente et Prophesee de m'avoir permis de faire cette thèse.

Merci, à David Picard pour ton écoute et ta gentillesse, surtout quand il s'agit du bien-être des doctorants.

Merci à tous les gens du laboratoire Imagine pour ces bons moments passés ensemble. Le sport et les sorties, mais aussi les nombreux repas et discussions partagés.

Je tiens à remercier ma mère et ma sœur, pour le support qu'elles m'ont apporté tout au long de cette thèse. Merci maman de m'avoir toujours encouragé à poursuivre mes passions.

Merci, à Martine et Jean-Michel de votre aide lors des confinements, vous avez rendu ces moments, qui auraient pu être bien compliqués, agréables. De façon plus générale, merci de votre accompagnement sur mon parcours professionnel.

Enfin, je remercie tout particulièrement Camille d'avoir été à mes côtés tous les jours de ce doctorat. Merci pour ta présence, ta bonne humeur, ton soutien et ton écoute. Merci d'être qui tu es. Merci de m'avoir accompagné dans cette aventure.

Contents

1 Introduction **7**
- 1.1 Goals . . . 7
- 1.2 Motivations . . . 8
- 1.3 Approach . . . 10
- 1.4 Challenges . . . 10
- 1.5 Contributions . . . 11
- 1.6 Outline . . . 12
- 1.7 Publications . . . 13

2 The Event Camera **14**
- 2.1 Principle . . . 14
- 2.2 Advantages . . . 16
- 2.3 Challenges . . . 17
- 2.4 Use-cases . . . 17
- 2.5 Overview of different devices . . . 18
- 2.6 Conclusion . . . 18

3 Related Work **19**
- 3.1 Reconstruction from Events . . . 19
  - 3.1.1 Updated event-by-event . . . 19
  - 3.1.2 Updated from a set of events . . . 28
  - 3.1.3 Conclusion . . . 34
- 3.2 Keypoint Detection . . . 36
  - 3.2.1 Frame-based Keypoint Detection . . . 36
  - 3.2.2 Event-based Keypoint Detection . . . 43
- 3.3 Convolutional Neural Networks . . . 50
  - 3.3.1 Squeeze and excite . . . 50
  - 3.3.2 Convolutional Recurrent Neural Networks . . . 51
  - 3.3.3 Convolutional Neural Networks in Event-Based Data . . . 53
- 3.4 Image and Gradient Prediction . . . 53

3.4.1 Handcrafted Methods . . . 54
3.4.2 Learned Methods . . . 55

4 Data Generation **60**
4.1 Introduction . . . 60
4.2 Real Data . . . 61
4.2.1 ATIS . . . 61
4.2.2 Alignment of Two Cameras . . . 61
4.2.3 Planar Surface . . . 63
4.3 Simulated Data . . . 64
4.4 Conclusion . . . 68

5 Detecting Stable Keypoints from Events through Image Gradient Prediction **69**
5.1 Introduction . . . 69
5.2 Method . . . 70
5.3 Results . . . 76
5.4 Conclusion . . . 79

6 Long-Lived Accurate Keypoints in Event Streams **80**
6.1 Introduction . . . 81
6.2 Method . . . 83
6.2.1 Input Event Representation . . . 83
6.2.2 Predicting Heatmaps . . . 84
6.2.3 Creating Training Data . . . 86
6.2.4 Training Details . . . 87
6.2.5 Inference and Tracking . . . 87
6.2.6 Architecture . . . 87
6.3 Experiments . . . 88
6.3.1 Baseline, Datasets and Metrics . . . 88
6.3.2 Ablation Study . . . 91
6.3.3 Comparison to the State-of-the-Art . . . 92
6.3.4 Qualitative results . . . 93
6.3.5 Computation Times and Memory Footprint . . . 95
6.4 Conclusion . . . 95

7 Conclusion **96**
7.1 Contributions . . . 97
7.1.1 Detecting Stable Keypoints from Events through Image Gradient Prediction . . . 97

7.1.2 Long-Lived Accurate Keypoints in Event Streams 97
7.2 Impact 98
7.3 Future work 98

Chapter 1

Introduction

1.1 Goals

Keypoint detection is a fundamental problem of computer vision since the 1980s and a building block to a large number of applications such as Simultaneous Localization and Mapping (SLAM), Structure-from-Motion (SfM), motion detection, image mosaicing/panorama stitching, image registration, object tracking, object recognition and 3D recognition.

We aim to develop and improve this very important tool of computer vision for a novel sensor: The Event Camera.

Event-based cameras (or neuromorphic camera) are recent sensors capturing the change in luminosity, asynchronously across the sensor [35, 68, 109]. They are able to capture high dynamic range scenes while maintaining a low-power consumption, making them particularly attractive for embedded systems [37]. Not surprisingly, many methods for keypoint detection for event-based cameras have already been proposed [4, 20, 43, 73, 77, 119].

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 noisy, with many false positives and false negatives [27, 45]. This makes computer vision problems for event-based cameras particularly challenging.

The goal of our work is therefore to build a very lightweight keypoint detection algorithm to be able to highlight important events and remove redundant or uninformative ones. This follows both the event camera mindset of keeping only relevant information and the need of many computer vision algorithms which use keypoints as their basis. For the algorithm to be useful, running in real time is paramount and

directs our choice of a minimal pipeline.

1.2 Motivations

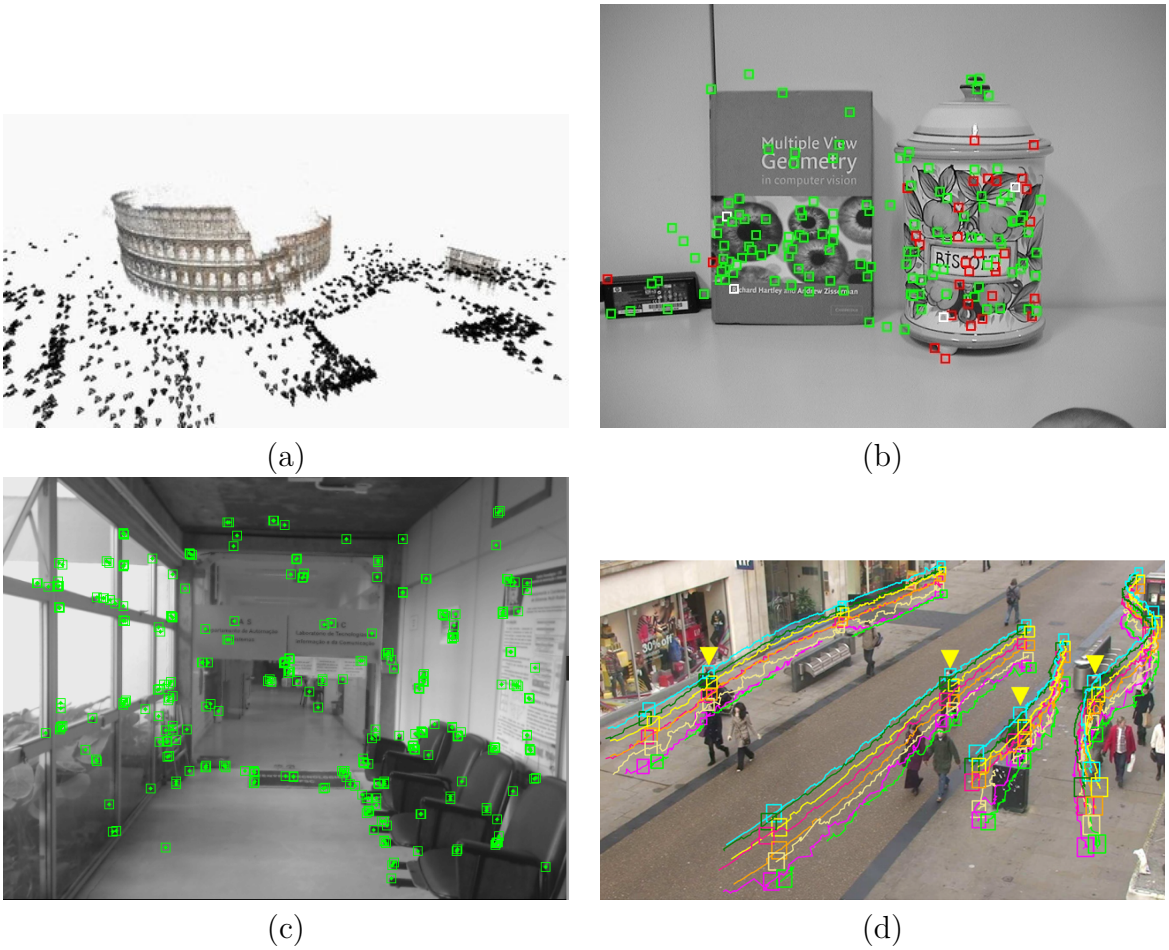
As of today there is not a single, generally accepted, approach to keypoint detection algorithm in event-based. Choosing between speed and quality is always a necessity, and even the best method's spatial precision leaves to be desired.
We want to make a keypoint detection algorithm, which is accessible to all, easy to use, and most importantly fast and very accurate. Many applications using events as their input would benefit from a robust keypoint detection algorithm. Other application using only frame-based data would also benefit from our algorithm as it would enable them to use event cameras for better dynamic range and lower latency without changing their pipeline as keypoints are universal and make the pipeline agnostic to the input data.

We introduce a multitude of applications where keypoints are proving themselves useful by reducing making a problem tractable or enabling for real time applications. Indeed comparing keypoints instead of pixels directly reduces computational needs and improve accuracy. The described applications are illustrated in Figure 1.1

Object Tracking Once a Deep Neural Network has been used to detect objects in an event stream such as [83] keypoint can prove useful to track said objects through time and reduce computation needs. This indeed avoids re-detecting the same objects over and over again. As improving efficiency of resources is at the core of event cameras tracking objects with minimal computation would be very valuable.

Object Recognition Object recognition tasks are very important in computer vision as they prove very useful to build some understanding of a scene. However object recognition algorithms are often using a lot of resources and can be quite time consuming. Simple object recognition or pattern detection with a template can be done with a keypoint detection algorithm in a very efficient way. Objects can also be detected when their motion is different from the scene or camera movement and keypoints can be grouped together to define an object.

AR/VR Augmented Reality is the way of overlaying generated graphics with the real world and Virtual reality is a computer generated environment where all is made to feel as real as possible and camera will follow the head movement of the person using the device to enhance the immersion. In Virtual Reality a precise and quick knowledge of the headset viewing direction is paramount and can be achieved with



(a) (b)

(c) (d)

Figure 1.1: (a) Keypoints thanks to their efficiency are used for large scale 3D reconstruction and camera pose estimation as done in [3] (b) Keypoints are useful for 3D object recognition [8] (c) SLAM pipeline also benefit from keypoints especially when real-time computation is needed (AR/VR, robot navigation) [47] (d) Another example of keypoints importance is for object tracking, pedestrians in this example. Illustration courtesy of `http://deepmachinelearningai.com/object-tracking-in-deep-learning/`

keypoints, while in Augmented Reality, the knowledge of the physical space around the user is also very important to be able to correctly position objects in said space.

SfM Structure from Motion is the act of recreating 3D from 2D input. From a movement and the motion parallax, the depth of keypoints can be inferred and 3D structure reconstructed. Inspired form how humans see the world this is very useful for 3D reconstruction of structure. Using keypoints can be an efficient way of recreating a sparse 3D cloud of an object or a scene.

SLAM Simultaneous Localisation And Mapping is another key application for keypoint detection. It needs accurate keypoints and is very useful for autonomous vehicles traversing unknown environment. It will enable the robot to navigate the environment while at the same time creating a 3D map of said environment. This application is one of the most complex as well as the most rewarding.

1.3 Approach

The asynchronous and sparse nature of the event stream entails that new algorithms specifically designed for event cameras need to be developed. One approach is to create novel dense representations of events and use the vast existing literature and algorithm of keypoint detection developed for frame-based applications. The other is to imagine novel algorithms and use machine learning to bridge the gap between event-based and frame-based computer vision. We have chosen the later approach in our work.

1.4 Challenges

The task of detecting keypoint in an event stream presents many challenges. The novelty, sparsity and asynchronicity of the inputs, the very high readout of novel sensors, combined with many appearance changes for the same objects in the event stream are problems we have faced in developing our keypoint detection algorithms.

Novel inputs As event cameras are relatively new sensors: 2000s, the modeling of the noise has been studied but is not as widely understood as the noise in frame-based sensors. As the noise is not yet fully understood adding hand-crafted heuristics to algorithms is often error prone. Preprocessing the events into a representation or letting a machine learning algorithm learn said representation is often a choice researchers are faced with. The asynchronous nature also has an impact on potential

algorithm creation. The design of the algorithm can use an event after the other or accumulate them using a fixed number or event or a predetermined time interval. The algorithm can use a dense representation or not, and can predict a stream or a dense map of keypoints.

Appearance changes depending on movements direction and speed In computer vision finding accurate, repeatable and local keyoint is already a challenge as the keypoint appearance can vary depending on lighting, scale and potential distortion. In an event stream the changes of appearance of the keypoints are drastic and vary with the direction of the movement. If the event are accumulated in a dense representation which is not invariant to speed (most are not) the appearance of the keypoint will also be dependent on the speed of the camera or of the object the keypoint is localized on. An illustration of this issue is shown in 3.5.

Balancing compute efficiency and precise temporal location of keypoints In the event cameras temporal precision is in the order of magnitude of the microsecond. However predicting all keypoint locations at the microsecond is intractable, especially as the sensors resolution augments [36]. There is therefore a balance to find between precise temporal localisation of keypoints and computational cost which comes with processing a large number of events. Indeed computations per event need to be kept to a minimum to achieve real-time performance without sacrificing accuracy.

Real-time keypoint prediction High temporal precision of event cameras combined with a high resolution sensor, a textured scene and high speed camera movement can lead to more than 1 gigaevents per seconds [36]. The more complex the computation per events are the more intractable the processing of events in real time becomes. Real-time prediction come at the cost of poor precision either spatial or temporal.

1.5 Contributions

The contributions of the thesis are:

- Two new algorithms for keypoint detection in an event stream
  - An algorithm in two steps: predicting gradients and detecting keypoints: Winner of the best paper award at the CVPR workshop on event-based computer vision in 2021.
  - An end-to-end keypoint prediction algorithm faster and more robust

- The source code used to train a gradient or keypoint prediction algorithm from an event stream.

1.6 Outline

We have sectioned this thesis in multiple chapters as follows:

Chapter 2: The Event Camera In this chapter we will introduce the theoretical aspects of the event camera, and how this novel sensor works. We will talk about the asynchronous and sparse nature of the pixels, and the link between the events and the image gradients. We will present the advantages of the camera: its speed, low power consumption and high dynamic range. We will also mention the challenges associated with using this sensor and the use-cases of the camera.

Chapter 3: Related work This chapter will consist of numerous reference to diverse works which have been an inspiration to build our algorithms. We present Convolutional Neural Networks, different dense representations for the events and finally keypoint detection algorithms, in frames and in events.

Chapter 4: Data Generation This chapter will present different strategies to generate data which can be used to supervise keypoint detection tasks in event-based. We will present both hardware and software based methods to obtain direct correspondence between real events and gray levels representation. In a second part we will describe different simulators used to simulate events from slow motion videos or synthesized videos. The videos can be created using either a rendering engine or as we do it using random yet temporally coherent homographies.

Chapter 5: Detecting Stable Keypoints from Events through Image Gradient Prediction This chapter consists of our first work done on events for keypoint prediction. We have used a lightweight recurrent convolutional neural network combined with a dense representation of events to predict image gradients and then compute a keypoint score using the Harris corner score formula. The theoretical link between events and image gradients make for an easy training of the network. The network is necessary to predict clean image gradients as the theoretical link between events and gradients does not account for noise which needs to be learned and modeled. We also introduce our training pipeline using simulated videos and events to

have a perfect match between events and images. This is useful in this work to predict image gradients and in the work presented in chapter 6.

Chapter 6: Long-Lived Accurate Keypoints in Event Streams In this chapter we refine our first approach to directly predict keypoints and avoid the intermediate step of the previous approach. We modify our training pipeline in two ways: instead of using the simulation to predict image gradients we use it to predict Harris corners and instead of recomputing keypoints as we did for image gradients we compute them once and reproject them using the known homographies. This gives us some added invariance to scale for the keypoints. Learning end-to-end enables a better spatial prediction of keypoints and a better temporal consistency. To improve even further the temporal consistency we also introduce a novel way to predict keypoints: instead of predicting keypoints at a single point in time, we predict multiple keypoints' positions in time corresponding to the accumulation time used to build the dense input of the network. It reduces computational needs and improves keypoint accuracy by making the tracking simpler for the nearest neighbor algorithm and therefore more robust.

Chapter 7: Conclusion In this last chapter we will reflect on our work and the state of keypoint detection in event cameras. We will give a summary of our contributions and offer some insights to build on our work.

1.7 Publications

Best paper award at CVPR 2021 Workshop on Event-based Vision:
Detecting Stable Keypoints from Events through Image Gradient Prediction,
P. Chiberre, E. Perot, A. Sironi, V. Lepetit

Long-Lived Accurate Keypoints in Event Streams, (submitted)
P. Chiberre, E. Perot, A. Sironi, V. Lepetit

Chapter 2

The Event Camera

We present in this chapter the event camera: its principle in Section 2.1, its advantages in Section 2.2 and challenges in Section 2.3. We also describe use-cases in Section 2.4 and give an overview of different devices in Section 2.5 before concluding in Section 2.6

2.1 Principle

The Event camera, also called neuromorphic camera, is a bio-inspired sensor registering asynchronously light intensity changes instead of images at fixed frame rate. Each of the pixel in the sensor, as the photoreceptors in the retina, stays silent as long as there is no change in the light intensity it measures. As soon as the intensity exceeds or is exceeded by a threshold, an event (x, y, p, t) is created, or "fired". It consists of the pixel coordinates (x, y), the polarity p: 1 when light increases and -1 when light decreases, and t the time at which the threshold was crossed. The sensor therefore produces a continuous stream of events $E(x_i, y_i, p_i, t_i)_{i\in\mathbb{N}}$. Figure 2.1 presents a dynamic and active pixel vision sensor (DAVIS) [15] which outputs asynchronous DVS events and synchronous global shutter frames concurrently. Another example of a sensor which outputs events coupled with grayscale measurements is the Asynchronous Time-based Image Sensor (ATIS) [86].

Event generation model We 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 C, an

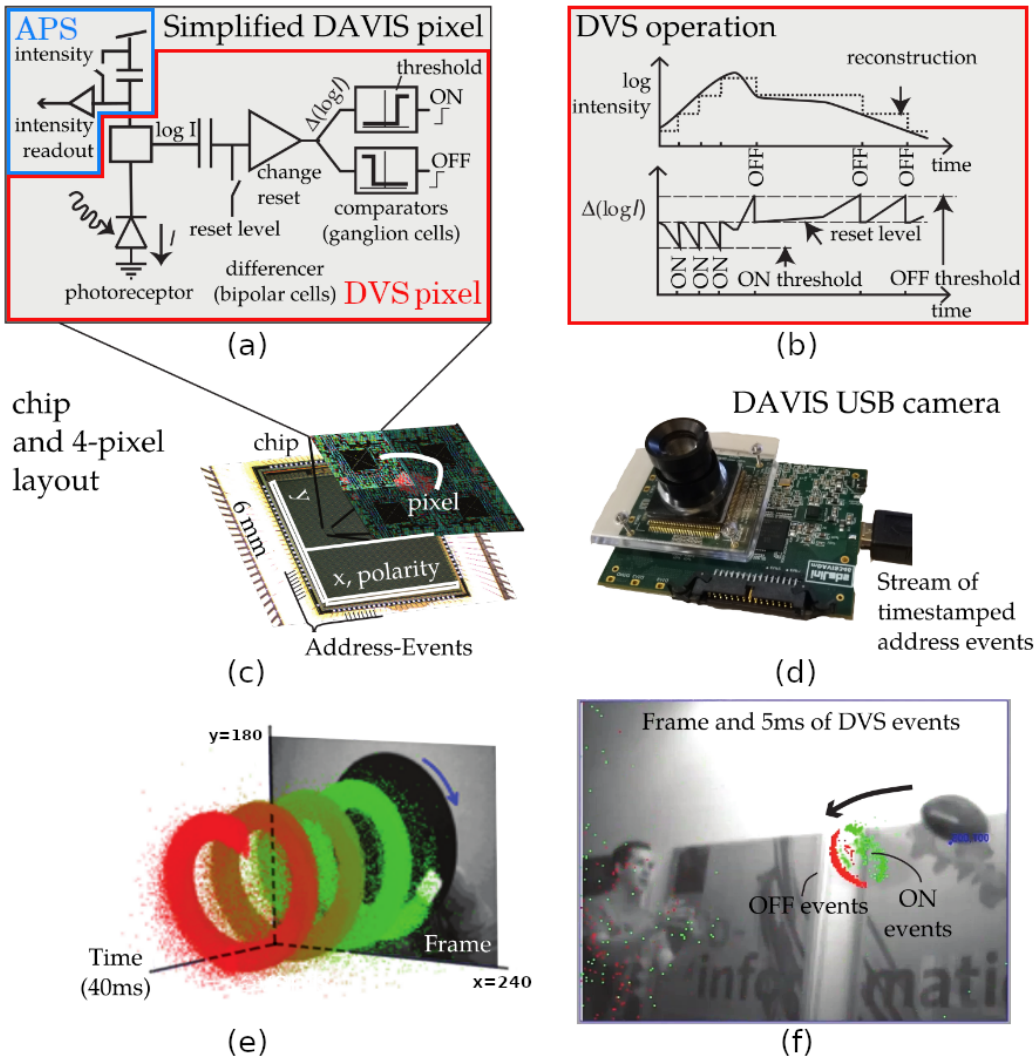


Figure 2.1: Summary of the DAVIS camera [15], comprising an event-based dynamic vision sensor (DVS [68]) and a frame-based active pixel sensor (APS) in the same pixel array, sharing the same photodiode in each pixel. (a) Simplified circuit diagram of the DAVIS pixel (DVS pixel in red, APS pixel in blue). (b) Schematic of the operation of a DVS pixel, converting light into events. (c)-(d) Pictures of the DAVIS chip and USB camera. (e) A white square on a rotating black disk viewed by the DAVIS produces grayscale frames and a spiral of events in space-time. Events in space-time are color-coded, from green (past) to red (present). (f) Frame and overlaid events of a natural scene; the frames lag behind the low-latency events (colored according to polarity). A more in-depth comparison of the DVS, DAVIS and ATIS pixel designs can be found in [87]. Figure courtesy of [38]

event $e_i = (x_i, y_i, p_i, t_i)$ is generated for pixel (x_i, y_i) if

$$\left| \log \left(\frac{L(x_i, y_i, t_i)}{L(x_i, y_i, t_{i-1})} \right) \right| = C \,, \tag{2.1}$$

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:

$$p_i = \text{sign} \left(\log \left(\frac{L(x_i, y_i, t_i)}{L(x_i, y_i, t_{i-1})} \right) \right) \,. \tag{2.2}$$

Defining:

$$\Delta log(L(x_i, y_i, t_i)) = log(L(x_i, y_i, t_i)) - log(L(x_i, y_i, t_i - \Delta t_i)) \tag{2.3}$$

combined with 2.1 and 2.2 gives us:

$$\Delta log(L(x_i, y_i, t_i)) = p_k * C \tag{2.4}$$

Where $\Delta t_i := t_i - t_{i-1}$ is the time since the last event at this position and p_i (the polarity) is the sign of the brightness change: $p_i \in \{1, -1\}$. In practice the contrast threshold can be selected and often set to C^+, C^-

Using the first order Taylor approximation and 2.3 with a small Δt_i we get:

$$\Delta log(L(x_i, y_i, t_i)) \approx \frac{\partial L}{\partial t}(x_i, y_i, t_i) * \Delta t_i \tag{2.5}$$

Which leads to the interpretation of events giving information of the temporal derivative of the log of the light intensity:

$$\frac{\partial log(L)}{\partial t}(x_i, y_i, t_i) \approx \frac{p_i * C}{\Delta t_i} \tag{2.6}$$

2.2 Advantages

Multiple advantages derive from the novel asynchronous registration of information.

Low latency. As the camera does not rely on a global clock, the temporal resolution of the event stream is very high. The temporal resolution is of the order of magnitude of microseconds [35].

High dynamic range. As the camera does no rely on a global sensitivity threshold, the dynamic range across the sensor is very enhanced, typically of the order of $120dB$ [69]

Low power. As the sensor registers information asynchronously and depending on the scene the consumption varies. Even for high resolution sensors (1280×720) the power consumption is minimal: 32mW (static) to 84mW at high activity (300ME/s) [35]

Low data. When the scene is static an event camera does not register any of the temporally redundant information, reducing both the data generated and post processing steps needed afterward [35].

These combined advantages make it a great sensor for robotics (drones in particular), wearable devices such as AR/VR headsets, and more generally: lightweight battery powered electronics in challenging scenarios for traditional cameras, where high dynamic range and low latency are expected.

2.3 Challenges

The event camera output is fundamentally different from the output of a standard camera. The common, dense and synchronous, output called frames are replaced by a sparse and asynchronous stream. Adapting frame-based algorithms to event-based cameras is therefore difficult.

The sensing also diverges from frame-based cameras. In event-based cameras the output is always relative to the past, which is fundamentally different from the absolute sensing of light intensity in standard cameras.

With the novelty of the sensor also comes the issue of noise modeling. Every physical sensor is noisy in its own way and diverges from the theoretical models. For now, and despite a lot of research, noise models for event cameras are still not perfect and do not allow direct frame reconstruction.

2.4 Use-cases

Algorithms for events cameras include, feature detection and tracking, optical-flow estimation, frame reconstruction, 3D reconstruction (mono and stereo), SLAM, segmentation and pattern recognition [5]. The low power consumption and low latency make the event camera a great tool for robotics [14, 24, 126, 127] and has been extensively tested on drones [28, 30, 34, 48, 79, 99, 112, 122]. The vast number and range of algorithms combined with the different nature of the event camera which outperforms standard cameras in challenging scenarios taking advantage of the low latency, and high dynamic range enables many applications such as: AR/VR, vibration monitoring, high speed object counting, self-driving cars, super slow motion and image

	Supplier	iniVation			Prophesee				Samsung			CelePixel		Insightness
	Camera model	DVS128	DAVIS240	DAVIS346	ATIS	Gen3 CD	Gen3 ATIS	Gen 4 CD	DVS-Gen2	DVS-Gen3	DVS-Gen4	CeleX-IV	CeleX-V	Rino 3
Sensor specifications	Year, Reference	2008 [68]	2014 [15]	2017	2011 [86]	2017 [1]	2017 [1]	2020 [35]	2017 [110]	2018 [57]	2020 [113]	2017 [46]	2019 [19]	2018 [2]
	Resolution (pixels)	128 × 128	240 × 180	346 × 260	304 × 240	640 × 480	480 × 360	1280 × 720	640 × 480	640 × 480	1280 × 960	768 × 640	1280 × 800	320 × 262
	Latency ()	12μs @ 1klux	12μs @ 1klux	20	3	40 - 200	40 - 200	20 - 150	65 - 410	50	150	10	8	125μs @ 10lux
	Dynamic range ()	120	120	120	143	> 120	> 120	> 124	90	90	100	90	120	> 100
	Min. contrast sensitivity ()	17	11	14.3 - 22.5	13	12	12	11	9	15	20	30	10	15
	Power consumption ()	23	5 - 14	10 - 170	50 - 175	36 - 95	25 - 87	32 - 84	27 - 50	40	130	-	400	20-70
	Chip size ()	6.3 × 6	5 × 5	8 × 6	9.9 × 8.2	9.6 × 7.2	9.6 × 7.2	6.22 × 3.5	8 × 5.8	8 × 5.8	8.4 × 7.6	15.5 × 15.8	14.3 × 11.6	5.3 × 5.3
	Pixel size ()	40 × 40	18.5 × 18.5	18.5 × 18.5	30 × 30	15 × 15	20 × 20	4.86 × 4.86	9 × 9	9 × 9	4.95 × 4.95	18 × 18	9.8 × 9.8	13 × 13
	Fill factor ()	8.1	22	22	20	25	20	> 77	11	12	22	8.5	8	22
	Supply voltage ()	3.3	1.8 & 3.3	1.8 & 3.3	1.8 & 3.3	1.8	1.8	1.1 & 2.5	1.2 & 2.8	1.2 & 2.8		1.8 & 3.3	1.2 & 2.5	1.8 & 3.3
	Stationary noise (ev/pix/s) at 25C	0.05	0.1	0.1	-	0.1	0.1	0.1	0.03	0.03		0.15	0.2	0.1
	CMOS technology ()	350	180	180	180	180	180	90	90	90	65/28	180	65	180
		2P4M	1P6M MIM	1P6M MIM	1P6M	1P6M CIS	1P6M CIS	BI CIS	1P5M BSI			1P6M CIS	CIS	1P6M CIS
	Grayscale output	no	yes	yes	yes	no	yes	no	no	no	no	yes	yes	yes
	Grayscale dynamic range ()	NA	55	56.7	130	NA	> 100	NA	NA	NA	NA	90	120	50
	Max. frame rate (fps)	NA	35	40	NA	NA	NA	NA	NA	NA	NA	50	100	30
Camera	Max. Bandwidth (eps)	1	12	12	-	66	66	1066	300	600	1200	200	140	20
	Interface	USB 2	USB 2	USB 3		USB 3	USB 3	USB 3	USB 2	USB 3	USB 3			USB 2
	IMU output	no	1	1	no	1	1	no	no	1	no	no	no	1

Table 2.1: Comparison of commercial or prototype event cameras. Values are approximate since there is no standard measurement testbed. Table courtesy of [38]

deblurring.

2.5 Overview of different devices

As can be seen in Table 2.1 the general tendency in event cameras development is to increase spatial resolution, increase readout speed, and add features, such as: gray level output (in ATIS and DAVIS). Only recently has the focus turned more towards the difficult task of reducing pixel size for economical mass production of sensors with large pixel arrays. In this respect, 3D wafer stacking fabrication has the biggest impact in reducing pixel size and increasing the fill factor.

2.6 Conclusion

We have presented the event camera, its principle, the challenges and advantages linked to the novel structure of information generated by the sensor. We will present in the next chapter the works which have influenced this thesis and on which we have build new ideas and algorithms.

Chapter 3

Related Work

Our work on keypoint detection for event cameras is related to many other preceding works. We use, and have been influenced by, a multitude of deep learning tools, different approaches of reconstructing a dense representation from events and other keypoint detection algorithms for both frame and event cameras. We will present works on dense reconstruction from events in Section 3.1, keypoint detection in Section 3.2, convolutional neural networks in Section 3.3 and, image and gradient prediction in Section 3.4.

3.1 Reconstruction from Events

The sparse nature of events and vast literature on dense representation (frames mostly) has led researchers to build a dense representation from events, enabling fast use of existing algorithms. Multiple dense representations of event streams have therefore been invented. We will detail each one bellow.

3.1.1 Updated event-by-event

A first class of dense representations is a tensor describing a state at time t and updated for each new event. It has the advantage of using and displaying all the available data while at the same time keeping the small temporal increments inherent in events. It is however demanding in compute resources and can be very imprecise if noisy events are not filtered when building said representations.

Membrane Potential Used to train spiking neural network the membrane potential [66, 76, 85] is a way to create a dense representation from a stream of events at

every available timestamp. A Leaky Integrate and Fire (LIF) [88] neuron is associated to every pixel and updated at every new event. Since the states of LIF neurons can be updated asynchronously based on the timing of input events, it is a very efficient model in terms of computational cost.

The Surface of Active Events (SAE) [13] define for each pixel and polarity the Surface of Active Events as the latest time an event has occurred (Equation 3.1):

$$\mathcal{SAE}(x, y, p) \rightarrow t\,. \tag{3.1}$$

To put some emphasis on the latest events, a regularization is introduced. [18, 23] introduce a representation named Leaky Surface. The surface increases at the position of the newest event and decreases everywhere else.

$$q(x, y) = \max(p(x, y) - \lambda * (t_i - t_{i-1}), 0), \tag{3.2}$$

$$p(x, y) = \begin{cases} q(x, y) + \Delta_{increment}, & \text{if } x_i = x \text{ and } y_i = y \\ p(x, y), & \text{otherwise} \end{cases} \tag{3.3}$$

where (x_i, y_i, p_i, t_i) is the latest event.

HOTS Regularization with an exponential decay are also considered and introduced in [63] named the time surface $\mathcal{S}_i$ for the i^{th} event. $\mathcal{T}_i$ is defined in Equation 3.4. We can see in Figure 3.1 the overview of the generation of the (regularized) time surface, and in Figure 3.2 the time surfaces for a couple of simple shapes and movements.

$$\mathcal{T}_i(u, p) = \max_{j \leq i} \{t_j | x_j = (x_i + u), p_j = p\}, \tag{3.4}$$

$$\mathcal{S}_i(u, p) = e^{-(t_i - \mathcal{T}_i(u,p))/\tau}, \tag{3.5}$$

where $u = [u_x, u_y]^T$ is such that $u_x \in \{-R, ..., R\}$ and $u_y \in \{-R, ..., R\}$

The time surface has the advantage of keeping values bounded and the parameters τ gives control over the order of magnitude of the accumulation of the events.

HATS In [107] the time surface is made more robust to noise by averaging both in the spatial and temporal dimension for $\mathcal{T}_i(u, p)$ As opposed to [63] where a time surface, defined for a spatial neighborhood $[-\rho, \rho]$ around an event $e_i = (\mathrm{x}_i, t_i, p_i)$ is defined as in Equation 3.6. This time surface can be visualized in Figure 3.3 (a).

$$\bar{\mathcal{T}}_{e_i} = \begin{cases} e^{\frac{t_i - t'(\mathrm{x_i}+z,q)}{\tau}} & \text{if } p_i = q \\ 0 & \text{otherwise.} \end{cases} \tag{3.6}$$

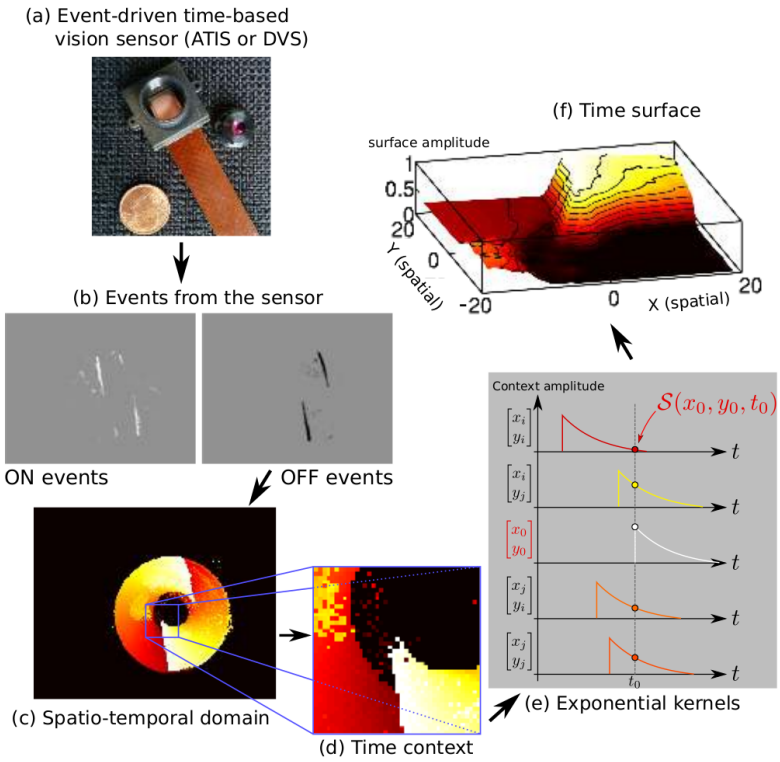


Figure 3.1: Definition of a time-surface from the spatio-temporal cloud of events. A time-surface describes the recent time history of events in the spatial neighborhood of an event. This figure shows how the time-surface for an event happening at pixel $\mathrm{x}_0 = [x_0, y_0]^T$ at time t 0 is computed. The event-driven time-based vision sensor (a) is filming a scene and outputs events shown in (b) where ON events are represented on the left hand picture and OFF events on the right hand one. For clarity, we continue by only showing values associated to OFF events. When an OFF event $ev_i = [x_0, t_i, 1]$ arrives, we consider the times of most recent OFF events in the spatial neighborhood (c) where brighter pixels represent more recent events. Extracting a spatial receptive field allows to build the event-context $\mathcal{T}_i(x, p)$ (d) associated with that event. Exponential decay kernels are then applied to the obtained values (e) and their values at t_i constitute the time-surface itself. (f) shows these values as a surface.

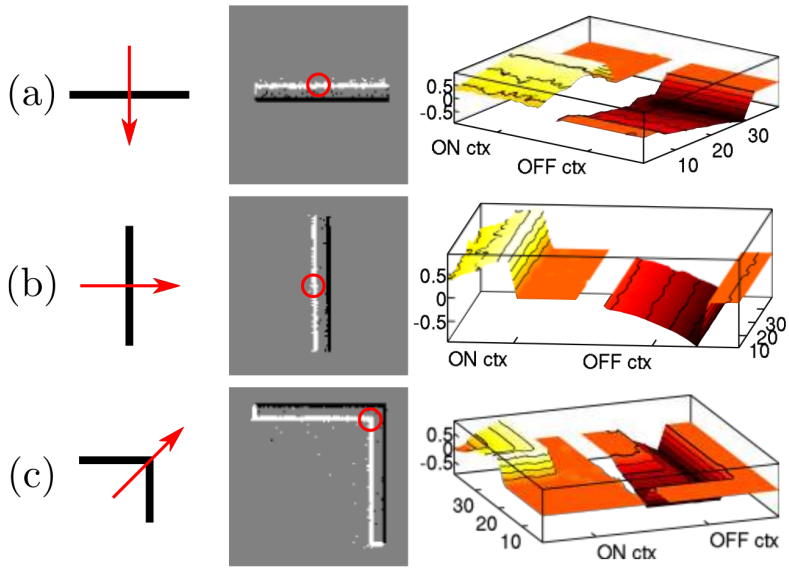


Figure 3.2: Example of some time-surfaces for simple movements of objects. First column shows a representation of the stimulus. The second column shows corresponding data from the ATIS sensor where white dots are ON events and black dots are OFF events. The third column shows the time-surface obtained from these events and computed for the event located in the center of the circle in the second column: the first, positive, half is obtained from the ON events and the second, negative, half is obtained from the OFF events. (a) A horizontal bar moving downwards. (b) A vertical bar moving rightward. (c) Corner moving to the top-right.

Where $t'(\mathrm{x}_i + \mathrm{z}, q)$ is the time of the last event with polarity q received from pixel $\mathrm{x}_i + \mathrm{z}$ and τ is a decay factor.

They introduce two new concepts, the Local Memory Time Surfaces and the Histograms of Averaged Time Surfaces. In the Local Memory Time Surfaces they avoid using only the latest event as in Equation 3.4 and use a sum of the latest events over a certain period of time (Equation 3.7), i.e. the sum is taken over a temporal window, making the time surface less sensitive to noisy pixels. This can be seen in Figure 3.3 (b).

$$\bar{\mathcal{T}}_{e_i}(\mathrm{z}, q) = \begin{cases} \sum_{e_j \in \mathcal{N}_{(\mathrm{z},q)(e_i)}} e^{\frac{t_i - t_j}{\tau}} & \text{if } p_i = q \\ 0 & \text{otherwise.} \end{cases} \tag{3.7}$$

Where $\mathcal{N}_{(\mathrm{z,q})}(e_i) = \{e_j : x_j = x_i + \mathrm{z}t_j \in [t_i - \Delta t, t_i), p_j = q\}$.

Another way to make the representation more robust is to average the time surface defined in Equation 3.7 spatially. The approach is inspired by [26], where adjacent pixels are grouped into cells $\{\mathcal{C}_l\}_{l=1}^{L}$ of size $K * K$. The sum is normalized by $|\mathcal{C}|$ the number of events in each cell. The Equation 3.8 formalizes the creation process for the averaged histogram and an example can also be seen in Figure 3.3 (c)

$$\mathbf{h}_{\mathcal{C}}(\mathrm{z}, q) = \frac{1}{|\mathcal{C}|} \sum_{e_i \in \mathcal{C}} \mathcal{T}_{e_i]}(\mathrm{z}, q). \tag{3.8}$$

Filtered Surface of Active Events Similarly, in [4] the authors present a way to remove noisy events for corner detection. They introduce the Filtered Surface of Active Events (FSAE) inspired from [63, 106]. The main idea is to remove events which occur rapidly after one another and only keep the first timestamp of the series. Intuitively rapid succession of events correspond to the same edge and are important only for gray level reconstruction as the number of events corresponds to the intensity difference between the two sides of the edge. As the intensity difference is not important for corner detection this approach (Equation 3.9) is beneficial to reduce noise. The approach is illustrated in Figure 3.4, where we can see the absence of noise for this linearly moving straight edge. Removing the events closely following the first event of the series can be seen as adding an artificial refractory period. This refractory period however needs to be tuned. A compromise needs to be found between removing noisy events and keeping informative events i.e. avoid removing important events which would be part of the underlying signal. Indeed during fast motions events can be close in time for a single pixel location yet carry important information. The tuning of τ^- in Equation 3.9 can be delicate.

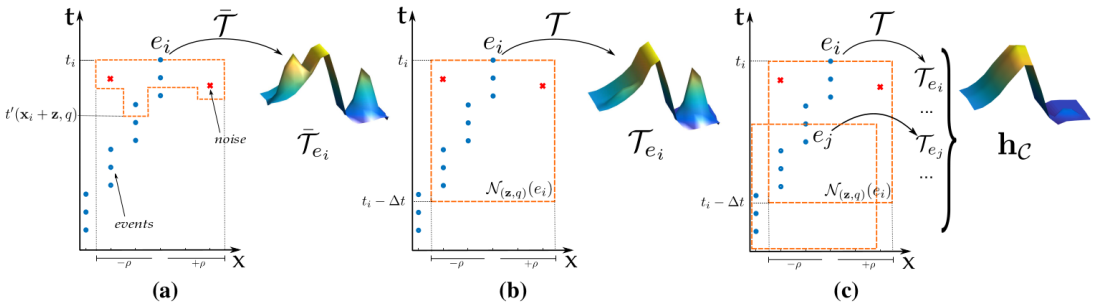


Figure 3.3: Time surface computation around an event e_i , in presence of noise. Noisy events are represented as red crosses, non-noisy events as blue dots. For clarity of visualization only the $x - t$ component of the event stream and a single polarity are shown. (a) In [63] the time surface $\bar{\mathcal{T}}_{e_i}$ (Equation 3.6) is computed by considering only the times $t'(\mathrm{x}_i + \mathrm{z}, q)$ of the last events in a neighborhood of e_i (orange dashed line). As a consequence, noisy events can have a large weight in $\bar{\mathcal{T}}_{e_i}$. This is visible from the spurious peaks in the surface $\bar{\mathcal{T}}_{e_i}$. (b) By contrast, the definition of Local Memory Time Surface $\mathcal{T}_{e_i}$ of Equation 3.7, considers the contribution of all past events in a spatio-temporal window $\mathcal{N}_{(z,q)}(e_i)$. In this way, the ratio of noisy events considered to compute $\mathcal{T}$ is smaller and the result better describes the real dynamics of the underling stream of events. (c) The time surface can be further regularized by spatially averaging the time surfaces for all the events in a neighborhood (Equation 3.8). Thanks to both the spatial and temporal regularization, the contribution of noise is almost completely suppressed.

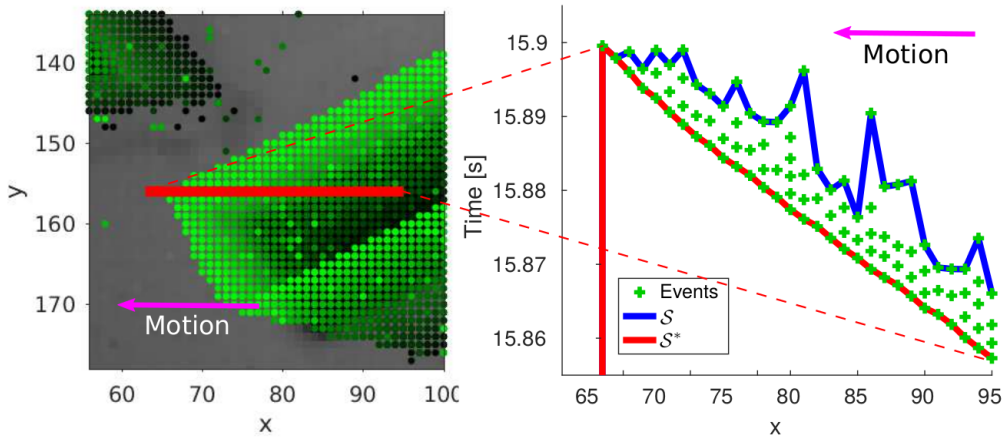


Figure 3.4: On the left, real events generated due to translation of a black rectangle are depicted (brighter green indicates newer events). On the right the timestamps of the latest events triggered in a small region (red segment) are illustrated. Although a single contrast change occurs, the significant magnitude of the change induces the triggering of multiple events in each of the pixels. The proposed filtered SAE S (red) captures the timestamps of the first event in each pixel, accurately representing the position of the rectangle at each time (note the constant velocity profile). A naive SAE S (blue) would capture the timestamp of consecutive latest events instead, which are subject to noise.

$$\mathcal{FSAE}(x,y,p) = \begin{cases} t, & \text{if } t - \mathcal{SAE}(x,y,p) \geq \tau^- \\ \mathcal{SAE}(x,y,p), & \text{otherwise.} \end{cases} \tag{3.9}$$

In all previously mentioned representations a change of speed or direction of the movement in the scene changes the visual aspect of the representation (see Figure 3.5). This proves detrimental for keypoint detection's robustness so [43, 73] introduce representations agnostic to speed and direction. [73] introduces the Speed Invariant Time Surface described in Algorithm 1. [43] also introduces a local neighborhood-based method called Threshold-Ordinal Surface (TOS), updating positions after each event relative to the surrounding values of the latest event as described in Algorithm 2.

Speed Invariant Time Surface As shown in Figure 3.5 the same corner will change appearance when aggregating events into frame representations such as an histogram. Surface of Active Events and Filtered Surface of Active Events suffer from the same issue. In [73] the authors present a novel dense representation for events named the Speed Invariant Time Surface. As the name suggests the representation is invariant to the speed of an object at the cost of added computations. The speed invariance is illustrated in Figure 3.6 for a one dimensional motion and the algorithm 1 explicitly details the computation for the creation of the representation.

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**
            $S(x + dx, y + dy, p) \leftarrow S(x + dx, y + dy, p) - 1$
$S(x, y, p) \leftarrow (2r + 1)^2$

Threshold Ordinal Surface [43] introduce the Threshold Ordinal Surface (TOS) which is illustrated in Figure 3.7. This representation is asynchronous and highly contrasted tailored to the Harris corner detection algorithm improving corner detection rate. The representation discards the polarity information of the events and is fully detailed in Algorithm 2. The main advantage of this representation is to have a constant minimum and maximum which is important to set a fix threshold for the Harris

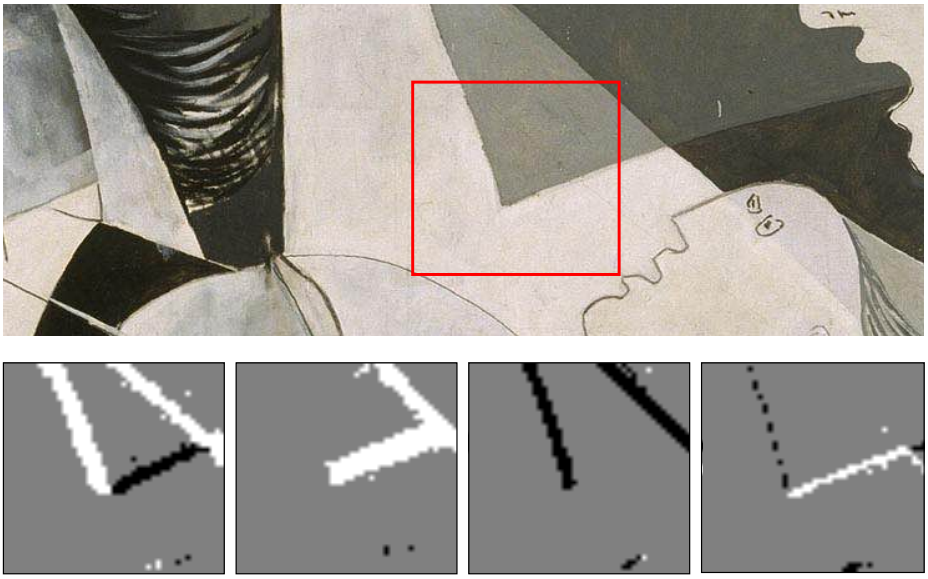


Figure 3.5: **(Top)** In a classical frame-based camera, the appearance of a corner, such as the one in the red square, is invariant under camera motions. **(Bottom)** In the case of an camera, instead, the same corner can generate vert different pattern of the events depending on its motion. The four panels show 40ms of events generated by the corner on the top while rotating the camera at different speeds.

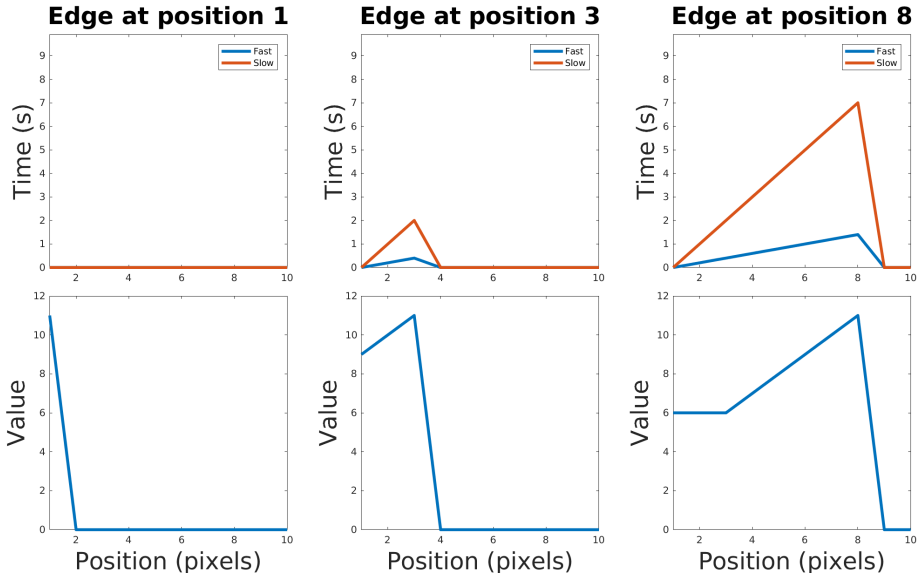


Figure 3.6: An edge moves from position 1 to 10 at two different speeds. It creates a slope on the Standard Time Surface T **(Top)** and on the corresponding Speed Invariant Time Surface S **(Bottom)**. S is identical for both speeds.

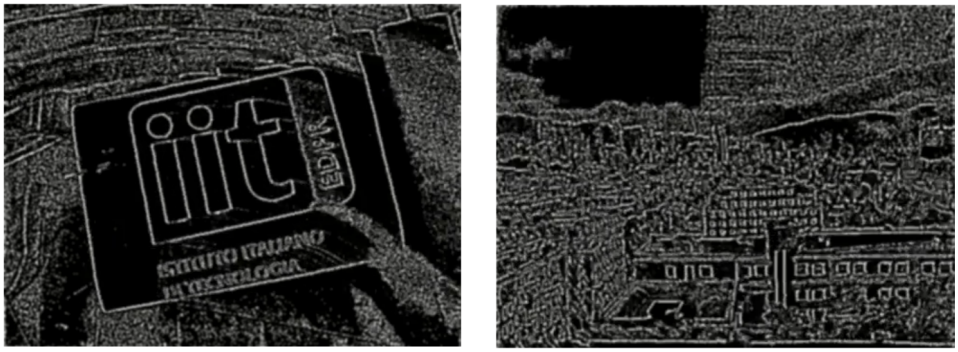


Figure 3.7: Illustration of the high contrast Threshold Ordinal Surface introduced in [43]. The Background is dark and uniform, it does not suffer from any ghosting effect.

corner detection algorithm. The hyperparameter used to create the binary histogram needed in eHarris [119] is also bypassed as the TOS is asynchronous and for each new event a new representation is computed. The computation, even if only a small region around every new event is affected, can quickly become intractable with high event rates. The approach also suffers from noise as events which are continuously and erroneously firing will have a very negative impact on the representation.

Algorithm 2 Event-by-event computation (q_1)

$v = \langle x, y, t \rangle$, TOS
for $x = v_x - k_{TOS} : v_x + k_{TOS}$ **do**
    for $y = v_y - k_{TOS} : v_y + k_{TOS}$ **do**
        $TOS_{xy} \leftarrow TOS_{xy} - 1$
        if $TOS_{xy} < 255 - T_{TOS}$ **then**
            $TOS_{xy} \leftarrow 0$
$TOS_{v_x v_y} \leftarrow 255$

3.1.2 Updated from a set of events

The representation methods detailed in the previous section all have something in common when creating a dense representation (with or without adding computations): the update is done for every new event. Even though every event brings some new information the extremely fine temporal granularity is not always needed and more general information can be extracted when considering a group of events instead of single events independently. The following methods we present in this section all use

of a set of event to create a dense representation. The set of events can be selected in two different ways: either by considering the last N events or the latest events such that $t_i \geq t - \delta_t$ with t being the timestamp of the latest event. The main issue with this approach is that both N and δ_t are hyperparameters which need to be fine-tuned depending on the use case. Indeed when considering only a fixed time frame and the speed of motion changes drastically appearance changes might become an issue. The same can be said when only considering a fixed number of events as the events can be uniformly distributed or very localized having a strong impact on the representation.

Flat representations

Binary Histogram The simplest dense representation from a set of events is a binary histogram as used in [119]. For the considered set of events E both the polarity and timestamps are discarded while only the positions (x_i, y_i) are used. For every (x_i, y_i) the binary histogram value is set to 1. It is set to 0 everywhere else. Binary histograms (BH) is formalized in Equation 3.10 and can be visualized and compared with histograms in Figure 3.8.

$$BH(x,y) = \begin{cases} 1, & \text{if } \exists e_i = (x_i, y_i, p_i, t_i) \in E | x_i = x \text{ and } y_i = y \\ 0, & \text{otherwise.} \end{cases} \tag{3.10}$$

Histogram A way to improve on the binary histogram described above is to count the number of events at each of the pixels location and create an histogram H. The values of each pixel (x, y) will correspond to the number of events with these coordinates in the set of considered events (see Equations 3.11 3.12 for formalization). The polarity of the events can be considered and two distinct histograms (HP) can be created (Equation 3.13 3.14). This would effectively create a two channels dense representation for the set of events. Another way to consider polarity is to create two histograms as before but aggregating them into a one channel representation using the difference operator (see Equation 3.15). Different histograms can be visualized in Figure 3.8

$$E_{(x,y)} = \{e_i = (x_i, y_i, p_i, t_i) \in E | x_i = x \text{ and } y_i = y\} \tag{3.11}$$

$$H(x,y) = \sum_{e_i \in E_{(x,y)}} 1, \tag{3.12}$$

$$E_{(x,y,p)} = \{e_i = (x_i, y_i, p_i, t_i) \in E \text{ such that } x_i = x, y_i = y \text{ and } p_i = p\} \tag{3.13}$$

$$HP(x, y, p) = \sum_{e_i \in E_{(x,y,p)}} 1, \tag{3.14}$$

$$H(x, y) = HP(x, y, 1) - HP(x, y, 0) \tag{3.15}$$

Inceptive Events Time-Surface The Inceptive Events Time-Surface introduced in [9] uses a more restrictive filtering than the FSAE introduced in [4]. Events are filtered considering both the preceding timestamps and following ones. The general idea is to average events within a certain time window. This methods, similar the FSAE, suffers from the same issues: the need to fine-tune parameters τ^- and τ^+. The Equations 3.16, 3.17, 3.18 give details of how the filtering is done. Let it be noted that this filtering is only possible for a set of events as timestamps from the past and the future of the events are considered which is not possible when working event by event. The visual comparison can be seen in Figure 3.9. Let E be the set of events:

$$T(x, y, p) = \{e_i \in E | x_i = x, y_i = y, p_i = p\} \tag{3.16}$$

$$\mathcal{IE}(x, y, p) = \{T(x, y, p) | t_i - t_{i-1} \geq \tau^- \wedge t_{i+1} - t_i \leq \tau^+\} \tag{3.17}$$

$$\mathcal{IETS}(x, y, p) = \text{mean}\{\mathcal{IE}(x, y, p)\} \tag{3.18}$$

Contrast Maximization Framework for Event Cameras In [39] the authors introduce a framework producing motion-corrected edge-like images with high dynamic range. Those images do not suffer from the motion blur in histograms and does not need any hyper parameter. The main idea of the method is to find the point trajectories on the image plane that are best aligned with the event data by maximizing an objective function: the contrast of an image of warped events. This framework has the advantage of being simple and can offer information on the optical-flow, the depth or the motion of the scene. As the events are aggregated and despite not suffering from motion blur the very fine temporal information of event-based sensors is lost in the process. the main issue however is the fact that the computational complexity of the image of warped events is linear on the number of events to be warped. This makes the framework unusable for lightweight and real time applications, especially as the number of events keep increasing with the higher resolution, novel sensors such as the one described in [36].

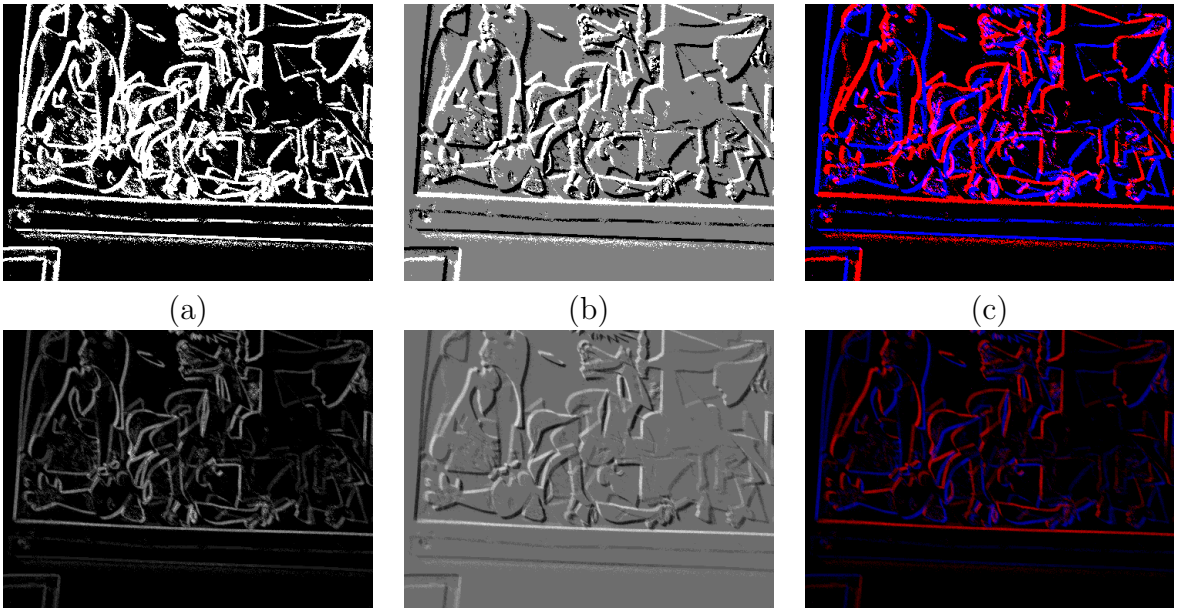
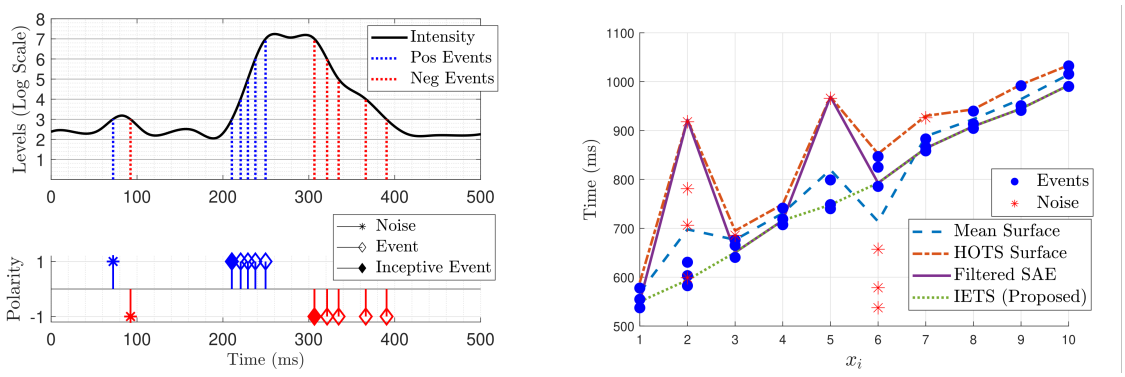
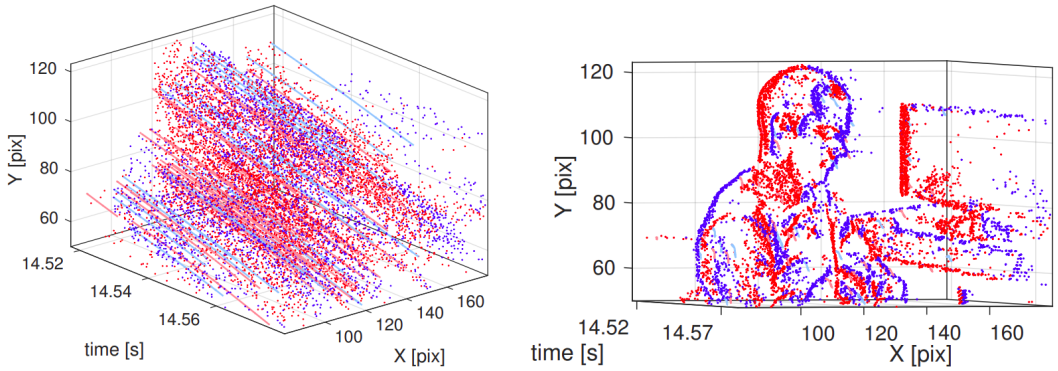


Figure 3.8: Different representations using histograms: (a) A simple binary histogram without considering the polarity of the events. It can be represented very efficiently with a single channel and a single byte. (b) and (c) consider the polarity of the events yet maintain a binary form of information when considering the set of events. The polarity can be represented with a single channel as a deviation from a standard value (b) or as a color when using multiple channels (c). (d)-(f) are the histograms counterparts of the first row. Each pixel value corresponds the amount of events with the same (x, y) coordinates. The histograms are a way to emphasize the pixels corresponding to signal and reduce the weight of noisy (isolated) events. The polarity is not considered in (d) while it is represented with a single channel in (e) as a deviation from a standard value or with two channels (one color per channel) in (f). (b) and (e) are also called difference of histograms as they are the difference of two histograms each corresponding to a polarity.



(a) (b)

Figure 3.9: Event Generation. **(a)** On a per pixel level, intensity variations trigger events at each log-scaled level crossing. The first event in a series of consecutive events is called an Inceptive Event. **(b)** IETS, FSAE, HOTS and the Mean Surface are compared in the presence of noisy events. IETS is the most stable approach and less sensitive to noisy events, both when leading and following signal events.



(a) (b)

Figure 3.10: motion-corrected image generation. **(a)** Events (dots) caused by a moving edge pattern and point trajectories in a space-time region of the image plane, colored according to event polarity **(b)** Visualization of the events along the direction of the point trajectories highlighted in (a). The approach works by maximizing the contrast of an image of warped events similar to (b).

Volumetric representation

The histograms are a simple and efficient way of representing a set of event in a dense manner. The issue however is the lost of details in the temporal dimension. The way to densify a set of events without the loss of information would be to have a volume where each channel would correspond to an event. The large number of events make this approach unfeasible in practice. The idea is therefore to find a compromise between having a single event per channel and aggregating all the events into the same channel. The Event Volume is here to fill this gap and keep a lot of temporal information. The events are not simply assigned to the closest channel i.e. they are not only contributing to a single channel but contributes to the two closest, weighted by the inverse of the distance to the channel, or bin. The notion of distance when saying closest is on the temporal direction: two events are close if the difference of their timestamp is close to zero.

Event Volume A more versatile approach to densify events is the event volume. It was first introduced in [129]. The event volume is populated using trilinear voting (interpolation) where each event (x_i, y_i, t_i, p_i) contributes its polarity to its two closest temporal bins according to:

$$E(x, y, t_n) = \sum_i p_i max(0, 1 - |t_n - t_i^\star|), \tag{3.19}$$

with:

$$t_i^\star = \frac{(t_i - t_{min})}{(t_{max} - t_{min})}(B - 1) \tag{3.20}$$

and where n is the temporal bin index, p_i is the polarity and $t_i^\star$ is the normalized timestamp of the i^{th} event. The hyperparameter B defines the number of channels of the volume. The linear voting in time make the Event Volume a very powerful representation for event. It is compact and descriptive without suffering of any motion blur. The hyperparameter B proves to be flexible and multiple values produce good results. We have chosen representation for our work due to its simpleness: no added computation apart from a simple linear interpolation, and descriptive power into a small tensor: we chose B to be of the order of magnitude of 10.

Discretized Event Spike Tensor In [41] a unifying framework is presented and a novel learnable representation is introduced. The main advantages of the method are two folds: minimize the information lost when densifying the events and having a representation which can be learned, each task can have an optimal representation

improving results on downstream tasks such as optical flow and object detection. The method works in three steps :

- The interpretation of the events as a succession of measurements of a function $f_\pm$ defined on the domain of the events (Equation 3.21).
- The convolution of the measurement field with a suitable aggregation kernel (Equation 3.22).
- The discretization of the signal into the Discretized Event Spike Tensor by sampling the convolved signal at regular intervals (Equation 3.23).

The final representation is created by projecting the Discretized Event Spike Tensor over the temporal axis or the polarities. The kernel can be handcrafted or learned to enable an end-to-end learnable representation. The learnable kernel is a small multilayer perceptron which takes the coordinates and time stamp of an event as input, and produces an activation map around it. The whole framework is more general than previous fixed representation but requires non negligible computational power.

$$S_\pm(x, y, t) = \sum_{e_k \in \mathcal{E}_\pm} f_\pm(x, y, t)\delta(x - x_k, y - y_k)\delta(t - t_k) \tag{3.21}$$

$$(k * S_\pm)(x, y, t) = \sum_{e_k \in \mathcal{E}_\pm} f_\pm(x_k, y_k, t_k)k(x - x_k, y - y_k, t - t_k) \tag{3.22}$$

$$S_\pm(x_l, y_m, t_n) = (k * S_\pm)(x_l, y_m, t_n) = \sum_{e_k \in \mathcal{E}_\pm} f_\pm(x_k, y_k, t_k)k(x_l - x_k, y_m - y_k, t_n - t_k) \tag{3.23}$$

3.1.3 Conclusion

We have seen in this section on overview of different ways to represent events bridging the gap between event-based data and vast frame-based algorithms and literature. The main takeaway is that most if not all methods who introduce some form of computation or who treat each event independently suffer from latency and become unusable when high event rates occur. The more simple representations such as the histograms, binary or not, are however too simplistic and discard too much of the temporal information for the use case of keypoint detection. The Event Volume proves to be a very good trade off between low latency and high spatio-temporal information. It is simple enough to be generated very quickly and retains a lot of the information thanks to the linear interpolation in the time direction. We have decided to use it as the input of our network and proved it's efficiency for the task of keypoint detection.

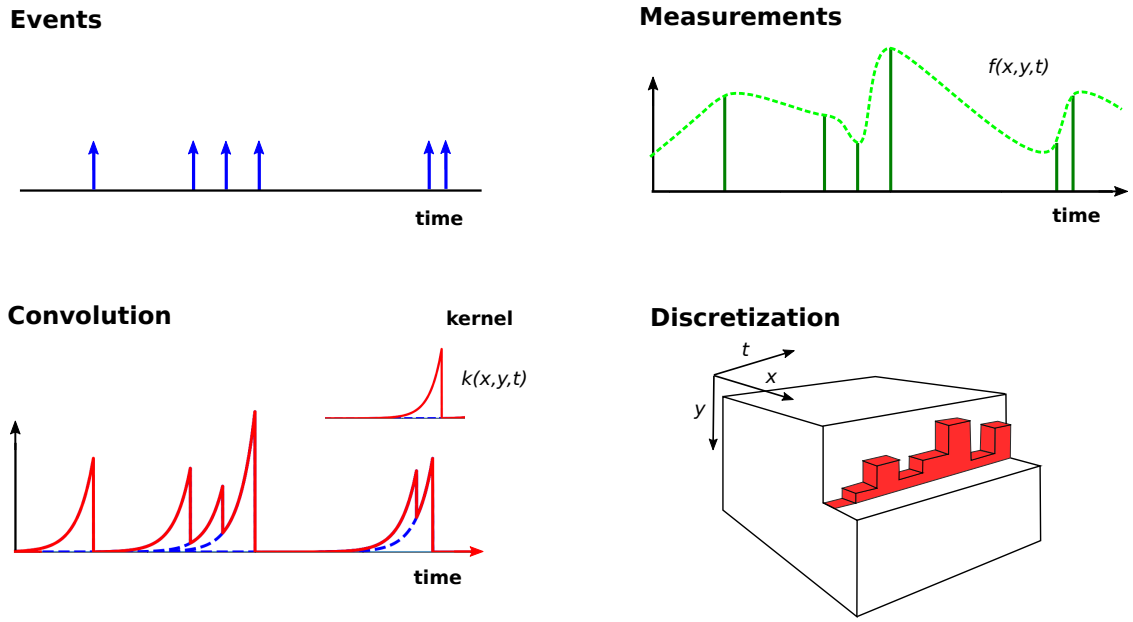


Figure 3.11: Overview of the method presented in [41]: Each event is associated with a measurement (green) which is convolved with a (possibly learnt) kernel. This convolved signal is then sampled on a regular grid. Finally, various representations can be instantiated by performing projections over the temporal axis or over polarities.

3.2 Keypoint Detection

Keypoints are sparse by nature. They are used as robust indicators and to reduce computations. For example when using keypoints to estimate a change of pose between two frames, one need only to consider a dozen (at most a couple hundreds) keypoints. This is both easier and more efficient than trying to match all (in the order of millions) pixels, in both frames. Keypoints are designed to be local, repeatable and accurate.

Local Keypoints are very local in the sense that they only consider a small pixel neighborhood (also called a "patch"). Ignoring global semantics of the image enables corner detection to dramatically reduce computations.

Repeatable Keypoints need to be as repeatable as possible. They should be invariant to rotations, translations, and zooms (scale). The translation invariance is achieved thanks to the local aspect of detectors. Rotation and aspect changes invariance need to be engineered into each method.

Accurate Most important, keypoints need to be very accurately localized. The best ones are sub-pixel accurate. As keypoint are used as a building block for downstream applications small errors can quickly accumulate and lead to algorithms failing. A comparison of keypoint detector with sub-pixel and pixel accuracy is done in [111].

A specific difficulty with events is that the same corner can appear differently depending on the direction and speed of the movement as can be seen in Figure 3.5. This change in appearance in event-based imply that repeatability is very difficult to achieve. While tracking can easily be done using the very high temporal definition of the sensor, tracking based on appearance is very difficult. We will present the historical keypoint detection methods in frame-based in section 3.2.1 and the keypoint detection methods developed for event-based data in section 3.2.2.

3.2.1 Frame-based Keypoint Detection

For Visual Odometry, Structure from Motion, Simultanous Localization and Mapping, and similar problems matching between images in a sequence need to be computed. Dense matching such as optical flow can be too expensive to compute which has led people to use only a subset of pixels to do the matching. We will see in the following subsection how those pixels are selected.

handcrafted Features

Corners are good candidates for keypoints as they are local, repeatable and accurate. Multiple method focus on corner detection such as the Harris corner detector [50], it's improvement Harris-Laplace [75] making it scale invariant and Features from Accelerated Segment Test (FAST) [98]. Another keypoint detection, focusing on blob-like areas: Scale-Invariant Feature Transform (SIFT) is also described in this section.

Harris corner detector One of the first successful attempts was the detection of corners and edges later referred to as the Harris corner detector [49]. The idea of the algorithm is to consider an image patch, compute it's cross-correlation and select local maximums as points of interests. The algorithm works as follow :

- convert and RGB image to a black and white image
- compute the spatial derivatives of the image (making it more robust to light intensity variations)
- consider a window such that $(x, y) \in W$, shift it by $(\Delta x, \Delta y)$ and compute the sum of square differences (see Equations 3.24 and 3.25).
- compute Harris score (see Equation 3.27). k is an hyperparameter determined by experiments $k \in [0.04, 0.06]$.

$$\begin{aligned} f(\Delta x, \Delta y) &= \sum_{(x,y)\ \in W} (I(x, y) - I(x + \Delta x, y + \Delta y))^2 \\ &\approx \sum_{(x,y)\ \in W} (I_x(x, y)\Delta x + I_y(x, y)\Delta y)^2 \end{aligned} \tag{3.24}$$

with I_x and I_y as the partial derivatives of I and using the first order Taylor expansion of f. in the matrix form, with M the structure tensor, we get

$$f(\Delta x, \Delta y) \approx (\Delta x \Delta y) M \begin{pmatrix} \Delta x \\ \Delta y \end{pmatrix} \tag{3.25}$$

$$M = \sum_{(x,y)\ \in W} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \tag{3.26}$$

$$S = det(M) - k * tr(M)^2 \tag{3.27}$$

The Harris corner detector is a way to measure how much an image patch is different from itself when moved horizontally and vertically. When the patch is very different the score is high and the point is considered as a corner, when the score is low however it corresponds to an edge or a flat region. In order to select the best corner in a local region, only local maximas of a small $3x3$ regions are considered. The Harris corner detector is highly repeatable and invariant to translation and in-plane rotation. However it fails for strong changes in view-point or scale.

Harris-Laplace To remove the variance of the Harris corner score when scale changes, [75] run the Harris corner detector at multiple scales. The best scale (characteristic scale) is subsequently selected using the Laplacian operator as proposed in [17]. The scale invariance however comes with a larger computational cost.

Distinctive Image Features from Scale-Invariant Keypoints (SIFT) [72] introduces features which are invariant to image scaling, rotation, partially invariant to change in illumination and 3D camera viewpoint. It works in multiple steps, we give them here as described in the original paper:

- **Scale-space extrema detection** : The first stage of computation searches over all scales and image locations. It is implemented efficiently by using a difference-of-Gaussian function to identify potential interest points that are invariant to scale and orientation. (Illustrated in Figure 3.12).
- **Keypoint localization** At each candidate location, a detailed model is fit to determine location and scale. Keypoints are selected based on measures of their stability. (Illustrated in Figure 3.13).
- **Orientation assignment**: One or more orientations are assigned to each keypoint location based on local image gradient directions. All future operations are performed on image data that has been transformed relative to the assigned orientation, scale, and location for each feature, thereby providing invariance to these transformations.
- **Keypoint descriptor** : The local image gradients are measured at the selected scale in the region around each keypoint. These are transformed into a representation that allows for significant levels of local shape distortion and change in illumination. (See Figure 3.14 for an illustration).

In addition to highly distinctive descriptors SIFT has the advantage of generating a large number of keypoints per image compared to other methods (in the order of 2000 stable keypoints for a 500x500 pixels image depending on the scene).

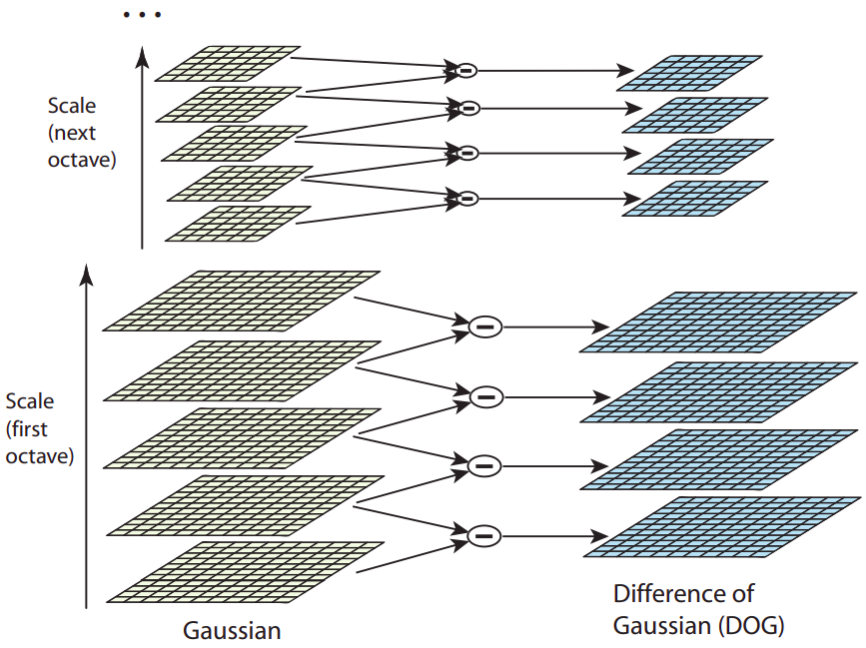


Figure 3.12: For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated.

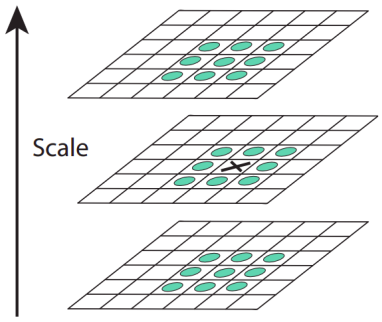


Figure 3.13: Maxima and minima of the difference-of-Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors in 3x3 regions at the current and adjacent scales (marked with circles).

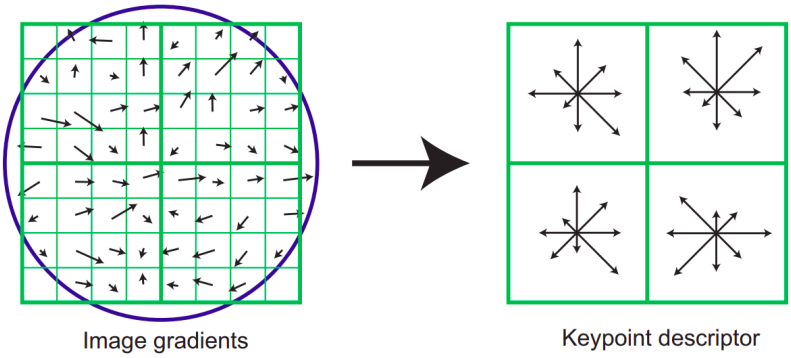


Figure 3.14: A keypoint descriptor is created by first computing the gradient magnitude and orientation at each image sample point in a region around the keypoint location, as shown on the left. These are weighted by a Gaussian window, indicated by the overlaid circle. These samples are then accumulated into orientation histograms summarizing the contents over 4x4 subregions, as shown on the right, with the length of each arrow corresponding to the sum of the gradient magnitudes near that direction within the region. This figure shows a 2x2 descriptor array computed from an 8x8 set of samples, whereas the experiments in the original paper use 4x4 descriptors computed from a 16x16 sample array.

FAST FAST [98] focuses on building a computationally efficient corner detection algorithm. For each pixel in the image the 16 pixels forming a circle around it is considered. The intensity of corners in the circle are compared to the one at its center. If a continuous arc of pixels are brighter or darker than the center pixel's intensity, the center pixel will be considered as a corner (illustration in Figure 3.15). The issue of duplicate detection in a small area is solved by using non maximum suppression. The proposed method is fast but suffers from noise as only pixels intensity values are considered.

Learned Features

With the development of machine learning many authors proposed methods where keypoints are learned using a dataset rather than setting handcrafted rules. Quickly after the development of ConvNets for image classification they had a high impact on the keypoint detection field. Improving robustness in keypoint detection is paramount and ConvNets can improve robustness both to noise and aspect changes of corners.

TILDE A Temporally Invariant Learned DEtector (TILDE) [121] introduces a piece-wise linear regressor based on Generalized Hinging Hyperplanes (GHH) [16]

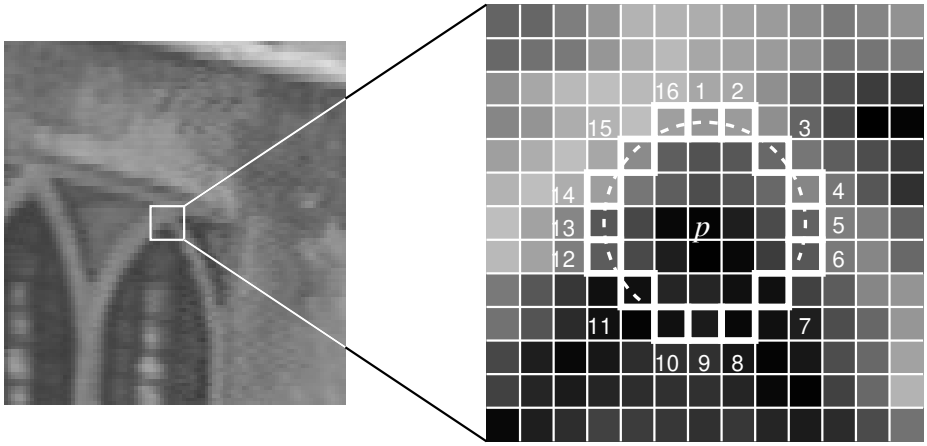


Figure 3.15: In FAST the candidate corner (pixel p) is is classified as a corner based on a 12 point segment test. The highlighted corners (forming a circle around the candidate) are considered, if a number of contiguous pixels in the circle are all brighter than p plus a threshold or dimmer than p minus a threshold, p is classified as a corner. Here 12 pixels are brighter hence the classification of p as a corner.

to detect corners invariant to strong illumination changes. The training is done with videos from a static webcam over long periods of time, learning both illumination and weather changes. Keypoint detectors are usually very sensitive to weather and lighting conditions. This approach however detects repeatable keypoints due to the training methodology of using the same point of view for day and night or different seasons altogether. The features used as input are the LUV components of the color space as well as the horizontal and vertical gradients of the image and the gradients magnitude.

Learned Invariant Feature Transform LIFT [125] are the first to learn the full feature point handling pipeline, that is, detection, orientation estimation, and feature description as is done in SIFT. They use three Convolutional Neural Networks (one for each component) who feed into each other (overview in Figure 3.16). The softargmax replaces standard non maximum suppression (NMS) to keep the pipeline differentiable. Although the end-to-end training is possible in theory it proves difficult to achieve in practice and the components are trained independently. The Descriptor is trained first, which is then used to train the Orientation Estimator, and finally the Detector, based on the already learned Descriptor and Orientation Estimator, differentiating through the entire network. At test time, they decouple the Detector, which runs over the whole image in scale space, from the Orientation Estimator and Descriptor, which process only the keypoints. The training is done using SIFT

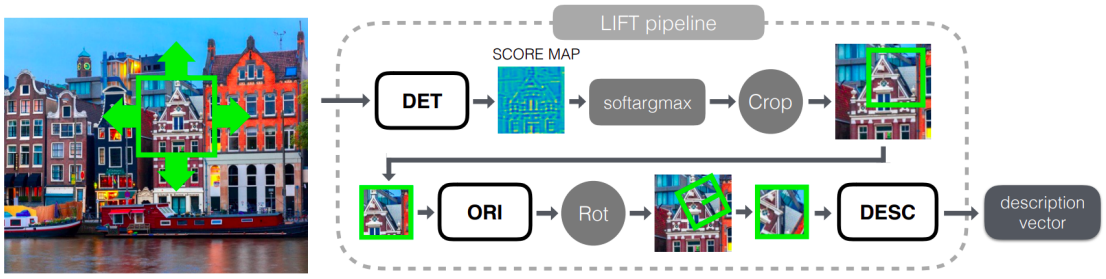


Figure 3.16: LIFT's integrated feature extraction pipeline. The pipeline consists of three major components: the Detector, the Orientation Estimator, and the Descriptor. They are tied together with differentiable operations to preserve end-to-end differentiability.

keypoints in association with a Structure-from-Motion model. This has proven to be an inspiration to our work: refining a handcrafted method with more modern and robust to noise algorithms. They manage to learn keypoints invariant to rotation, translation, scale and view-point changes, as in SIFT, and manage to outperform both SIFT and previous state-of-the-art baselines.

SuperPoint [29]Introduces two novel and important ideas to the keypoint detection field. They train their network with synthetic images for the first time using simple symbols with known corners as training examples and in a second step they refine the corners positions by training the network on natural images with and without some distortion from a known homography. The pipeline, described in Figure 3.17, has been another inspiration for us. The use of synthetic dataset as well as trying to refine existing keypoint information and improve them has been one of our main objective for event-based keypoint detection. Another advantage of SuperPoint is the fully convolutional architecture which takes as input the image and outputs interest points and descriptors without the need for multiple networks or a complicated pipeline at test time.

Others A number of other method use ConvNets to detect and describe keypoints in images which we cite here for completeness: D2-Net [32], R2D2 [96], D2D [116], S2D-Net [42] and DISK [118].

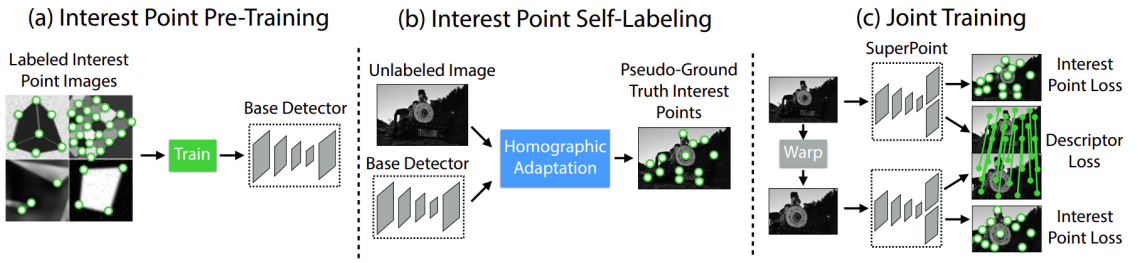


Figure 3.17: **Self-Supervised Training Overview of SuperPoint.** (a) An initial interest point detector is pre-trained on synthetic data. (b) A novel Homographic Adaptation procedure is applied to automatically label images from a target, unlabeled domain. (c) The generated labels are used to train a fully-convolutional network that jointly extracts interest points and descriptors from an image.

3.2.2 Event-based Keypoint Detection

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. As in other areas of computer vision handcrafted methods precede machine learning ones. Handcrafted methods are more interpretable, easier to implement (do not need any data) and can sometimes be very fast, however machine learning methods benefit from the large amount of data available and already existing handcrafted methods to train models which surpass previous handcrafted only methods. We have used and built on the best of both worlds, handcrafted for the interpretability and machine learning for its robustness to noise and generalization power.

Handcrafted Methods

The handcrafted methods for keypoint detection in event-based are inspired or adapted from frame-based keypoint detection. We will first present two approaches: eHarris and LuvHarris where the authors have directly applied the Harris corner detector on a dense representation built from the events. The aim is to create the best representation to be able to use a well known and robust corner detector which was developed for frames. Two other methods were inspired from keypoint detection in frames, and more specifically corner detection: eFast and ARC. eFast as in eHarris have created a dense representation on which they applied the FAST algorithm developed for frames. ARC introduced both a new representation and added some handcrafted rules to make eFast more robust to noise and detect more varied shapes of corners.

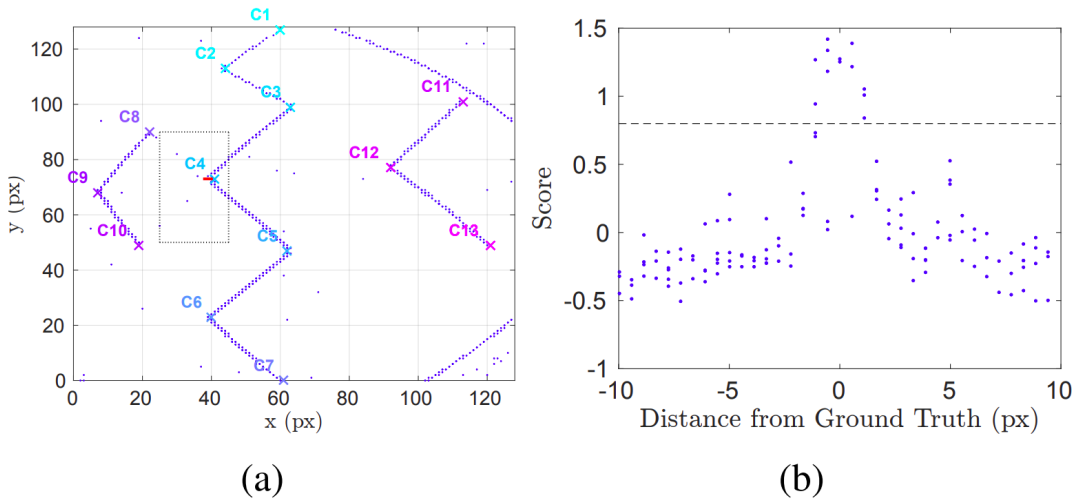


Figure 3.18: (a) Slice of 1000 events and ground truth corners and (b) distribution of the score compared to the distance from the ground truth. The threshold S is marked by the dashed black line. Blue dots represent the score around the ground truth (red line) of each event inside the dotted black square within the slice of 1000 events.

eHarris In [119] the authors creates a binary histogram from the latest N 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 grey level images [50], is then applied to this binary histogram. The method suffers from two drawbacks: without some kind of memory it is sensitive to speed of motion or scene complexity while the hyperparameter for the number of events needs to be tuned accordingly, secondly as can be seen in Figure 3.18 (b) the score for the corner is not accurately localized. Some post processing such as a Gaussian fitting needs to be done where scores are higher than a threshold to estimate the corner position in space. The estimation would however be too computationally expensive, which is why it was not implemented in the approach.

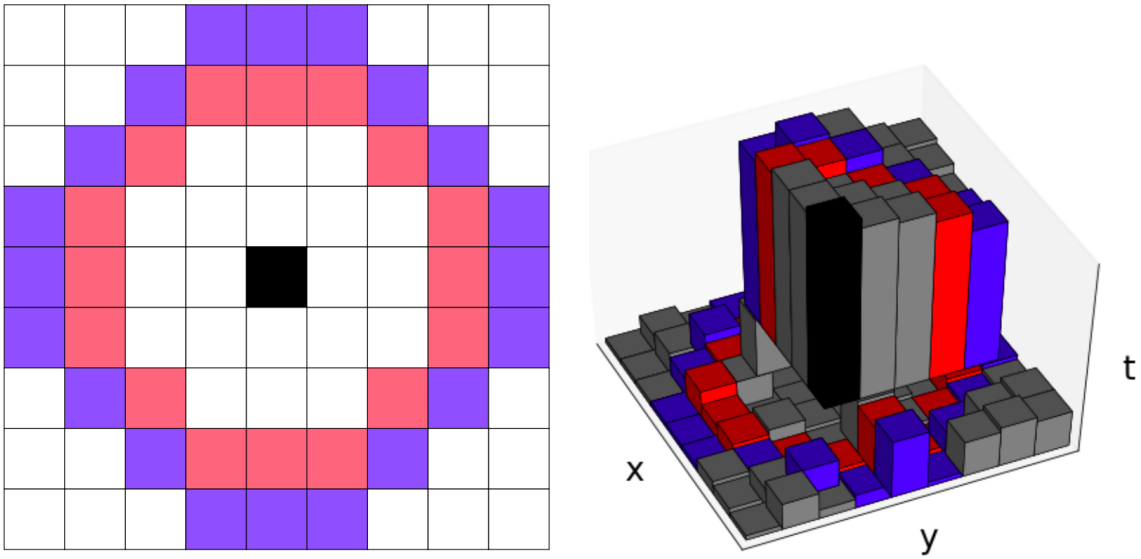
LuvHarris Look-up event-Harris [43] contributes in two ways to corner detection in event-based vision. They introduce a novel representation of events, named Threshold-ordinal Surface (TOS), replacing the binary image used in [50]. The TOS is a novel representation specially designed for Harris corner detection computation. It has an integrated memory while at the same time staying bounded. 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. It is described in more details in 3.1.1. The second contribution

is the introduction to computing the Harris corner score only as-fast-as-possible and completely decoupled from the data flow. When the Harris score cannot be computed due to computational constraints it is looked-up in a stored table, hence the name of the method. The computational impact of the Harris score map is therefore lessened and they manage spikes of up to 8.59 M events / s.

eFast In [77], the authors use a non-computationally intensive representation, named the Surface of Active Events (SAE). 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 new event at its location in the SAE. They considers two circles around the latest event of radius 3 and 4 respectively and compare the timestamps with respect to other timestamps along the circles. Similar to [98] they classify center pixels as corners if segments of contiguous pixels (arcs) are higher (i.e., they have a more recent timestamp) than all other pixels on the circle. The number of pixels should be between 3 and 6 for the inner circle and between 4 and 8 for the outer one. If such segments are found on both circles, the current event is classified as a corner event. The circles, the SAE and a classification example can be visualized in Figure 3.19.

Arc In [4], 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, enabling detection of corners with angles larger than 180 degrees. They introduce a new representation for events: the Filtered Surface of Active Events described above in 3.1.1. They introduce as well a significantly more efficient iterative algorithm that minimizes the number of operations to classify corner-events. The novel representation makes the algorithm more robust by reducing the amount of noisy events, the new classification rules augment the number of corners detected and the novel algorithm enables more than a 4× speed-up compared to eFast. As can be seen in Figure 3.21 the tracks detected by the algorithm are continuous and suffer from very few misclassifications. The corners are however not very precisely localized and the metrics used for validation do not really reflect it. Indeed they use a metric where a corner-event is labelled as true positive if it is closer than 3.5 pixels from an intensity-based track, i.e. a track generated from corners in gray level images. The method is therefore precise up to 3.5 pixels which is problematic in practice. This also explains why in Figure 3.21 we can see tracks which are not really straight (they seem to be jittering) while at the same time still being classified as corners.

In all those handcrafted methods some computation is done event-by-event, either for the dense representation or the classification of events as corners. It becomes quickly intractable as the resolution of event-based sensors keeps growing [35] and



(a) (b)

Figure 3.19: The proposed method compares the timestamps of the pixels on two circles (red and blue) around the current event (in black). (a): The inner (red) and outer (blue) circle around the current event (black). (b): Visualization of the Surface of Active Events (SAE) around the current event (black) and of the circles used for the timestamp comparison. In this example, the event under consideration (center pixel) is classified as corner.

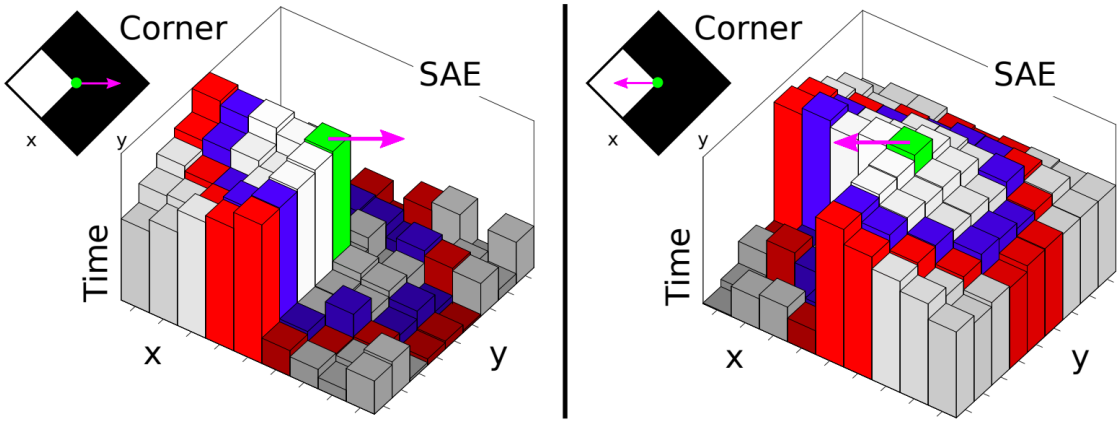


Figure 3.20: This figure illustrates an example in which the same corner under two different motion directions (Magenta arrows) may induce completely different SAEs (Height represents the timestamps of the latest event triggered in each location). Both algorithms, Arc*, and eFAST successfully detect the corner on SAE depicted on the left, as the set of newest elements in the inspected circles (Blue and red) are distributed continuously. However, only Arc* (and not eFAST) is able to detect the same corner from the SAE depicted on the right, where the angle of arc of newest elements is now over 180°.

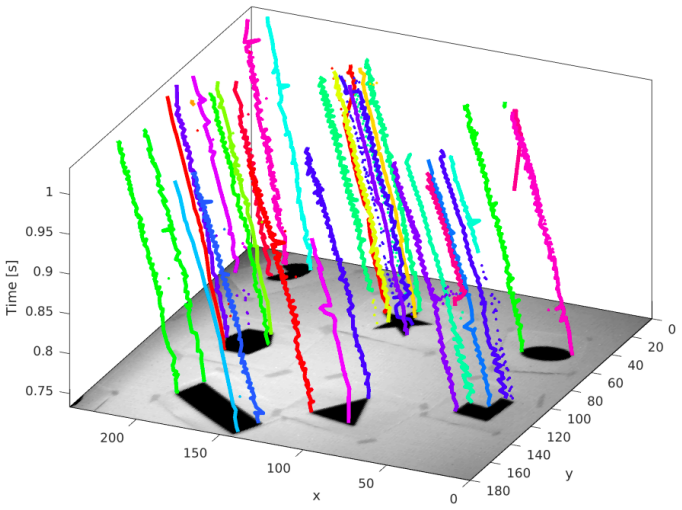


Figure 3.21: Despite the new Filtered Surface of Active Events and the new rules for corner detection, the Arc algorithm suffers from small (less than 3 pixels) yet noticeable local noise in the keypoint localization.

therefore event rates increase dramatically. The second issue is that the metrics for algorithm evaluation are not good enough when approaching the pixel level or even sub-pixel accuracy which is needed in many applications. The novel evaluation metric introduced in [73] described in Section 3.2.2 solves the issues regarding the metrics. The next section will focus on Learned methods which need training as well as data to perform well, i.e. be very robust to noise and generalize to many different scenes. The robustness to noise and generalization power are two very important aspects of a good keypoint detection algorithms.

Learned Methods

We have presented in the previous section handcrafted methods, they are a natural way to start a problem since learned methods require annotated data which is not easily attainable, specifically in a novel field. The handcrafted approaches however are sensitive to the noise present in the event stream which is an issue when trying to generalize the method to other scenes or sensors. More recently, despite the advance of deep learning only a single method is using machine learning for keypoint detection in event-based. Training a model is very challenging as coherent and accurate ground truth is very difficult to obtain. Only one paper [73] before us has used machine learning successfully to detect keypoints in an event stream.

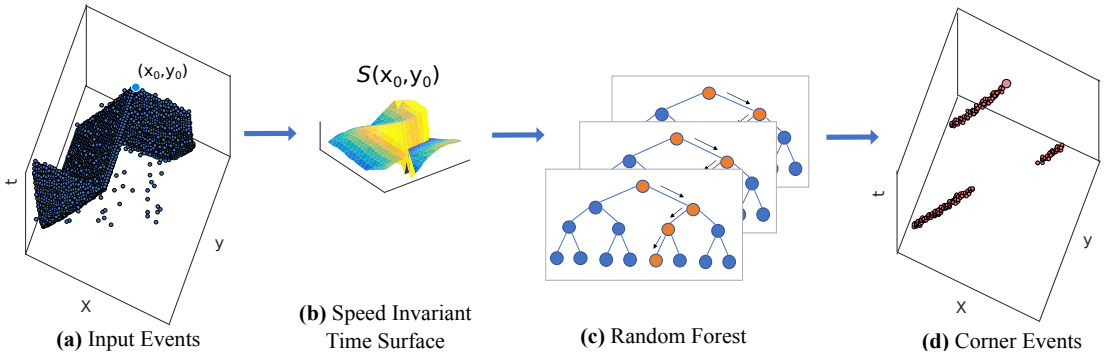
Speed Invariant Learned Corners (SILC) [73] made three significant contributions to the field (which are overviewed in Figure 3.22: the speed invariant time surface, the classification of patches as corners using a learned Random Forest and the introduction of a novel evaluation method for event-based corner detection algorithms on planar scenes with a novel dataset: The HVGA ATIS Corner dataset. We have already detailed the speed invariant time surface in 3.1.1. We focus in this paragraph on the machine learning and new dataset. As a reminder a Random Forest is a an ensemble of decision trees, and each tree is uncorrelated (by randomly sampling a subset of the training set). We detail here how the Random Forest is trained and how the evaluation for planar scenes can be defined to be more general and more precise. To train the Random Forest, a dataset (and therefore a ground truth) needs to be built. Ground truth for event-based corners is difficult to obtain, manual annotation is unfeasible and automatic annotation can prove difficult. They propose to leverage the gray level measurement provided by a HVGA ATIS sensor which provides asynchronous intensity measurements in form of time differences at the same temporal resolution of the events. For every event, they apply the Harris corner detector to its location only, in the graylevel image. If a corner is detected at this location, they add the event to the training set as a positive sample, and as a negative sample otherwise. The gray level are only used in training and are not needed at testing time.

The Harris detector applied to gray level images can sometimes fail in presence of noise, to avoid such an issue the dataset was acquired by recording well contrasted geometric patterns only. The use of simple geometric patterns reduces the amount of false negatives examples in the training set but reduces the generalization power of the method and fewer corners will therefore be detected at test time. Another issue of the method is the use of gray levels which is not available in all sensors. This limits the number of sensors on which this method can be applied.
The other important contribution they make is by introducing a new dataset and a new metric specifically designed to evaluate the accuracy of event-based feature detection and tracking algorithms. The dataset contains only records of planar patterns, so that ground truth acquisition is simple and the evaluation is less affected by triangulation errors. In order to assess the quality of an event-based detector, they combine it with a simple tracking algorithm based on nearest neighbor matching in space and time. After tracking, they evaluate the accuracy of the method by computing the reprojection error associated to the feature tracks (computed by estimating a homography between two different timestamps). The evaluation method is explained more formally in Section 6.3.1. As in [6] they are however obligated to use "trail filtering": removing the multiple events generated by a contrast step in intensity which causes multiple threshold crossing. The use of the filter shows the limitation of the method to generalize and the need for some tuning parameters. Despite improving the quality of the keypoints, the computation of the speed invariant time surface and random forest done event-by-event make this approach unsuitable for real time applications in practice.

Discussion

While algorithms treating events in their natural form, independently and asynchronously, are theoretically appealing, they are very sensitive to noise and are computationally intractable for certain high event-rate scenes in high resolution sensors such as the one described in [35]. Even when trying to treat each event independently every keypoint detection method uses a dense representation: binary histograms, Surface of Active Events, Time Surface, Threshold-Ordinal Surface, Speed Invariant Time Surface. Depending on the representation chosen issues with creating the surface arise when values are interdependent and cannot be updated directly (Threshold-Ordinal Surface and Speed Invariant Time Surface). Removing any processing step such as the binary histograms or Surface of Active Events ensures a quick generation of the dense representation. The downside of such methods is that they are very sensitive to noise as every event has the same weight in the representation. Some expensive post-processing would be needed to have precise keypoint detection algorithms.

To move away from such issues we have chosen to use a simple and direct dense



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

Figure 3.22: Overview of the SILC method. For each incoming event (x_0, y_0, t_0, p_0) in the input stream **(a)** the Speed Invariant Time Surface is computed **(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.

representation of the events: the event volume, and use a neural network to learn noise patterns from data directly, building a very stable feature representation for keypoint detection with minimal computational cost. The event volume maintains the benefit of event cameras which are high dynamic range and no motion blur while at the same time the shallow convolutional recurrent neural network limits computation and keeps the state of the scene into memory. This approach has given very precise and stable keypoint detection trough space and time.

3.3 Convolutional Neural Networks

Convolutional neural networks are an essential part of our approach. First introduced in [65] for image classification, they have not stopped gaining traction in computer vision problems and are still widely used for a multitude of tasks such as: image classification [51, 54, 104], object recognition [67, 71, 94, 95], semantic segmentation [114, 115] and many others. Both our approaches to keypoint predictions rely on ConvNets albeit with multiple modifications.

3.3.1 Squeeze and excite

The Squeeze and excite (SE) [55] block is the basis of the ILSVRC 2017 classification submission which won first place and reduced the top-5 error to 2.251%, surpassing the winning entry of 2016 by a relative improvement of $\approx 25\%$. The key idea is to

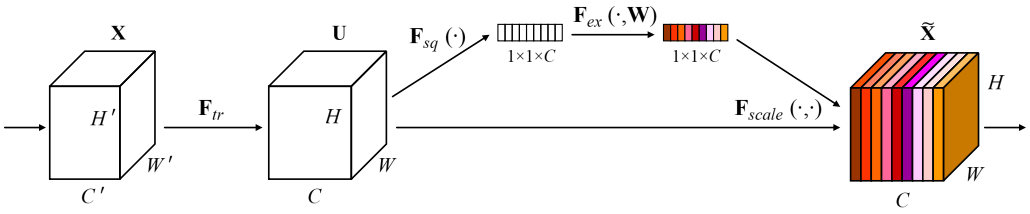


Figure 3.23: **Squeeze and Excitation (SE) block.** The main idea is to first Squeeze the features down to a channel descriptor by aggregating feature maps across their spatial dimensions. This descriptor, which allows the information from the global receptive field of the network to be used by all its layers, is followed by an excitation operation. It is a simple self-gating mechanism producing per-channel modulation weights. They are then applied to the initial feature map to generate the output of the block. As the dimension is unchanged, blocks can be stacked to create an SE network (SENet).

explicitly model interdependencies between channels with minimal additional computational cost. They propose a mechanism that allows the network to perform feature recalibration, through which it can learn to use global information to selectively emphasise informative features and suppress less useful ones. We can see in Figure 3.23 the overview of an SE block. In Figure 3.24 we can see how easily the resnet block can be adapted to use the Squeeze-and-Excitation block. They proved it to be greatly beneficial for a slight additional computational cost and this is the approach we have chosen and used in our small network when predicting gradients and later directly predicting keypoints.

3.3.2 Convolutional Recurrent Neural Networks

Shi et al. in [124], added a memory block to a convolutional neural network. The memory block, a Long Short Term Memory [53] was introduced to learn to store information over extended period of time and solving the issue with conventional Back Propagation Through Time (BPTT) which suffers from either a vanishing or exploding gradient when the sequence is too long. In the LSTM Equations 3.28 ($\circ$ denotes the Hadamard product), which can also be visualized in Figure 3.29, we see how every hidden state, and past state depends on all of the inputs. [124] combines a convolutional neural network for feature extraction to an LSTM memory both in the input-to-state and state-to-state. As the next state will depend only on previous states and local spatial neighbors in the input. The added convolutions (Equation 3.29), reduce the complexity and use the inductive bias of ConvNets. In Equation

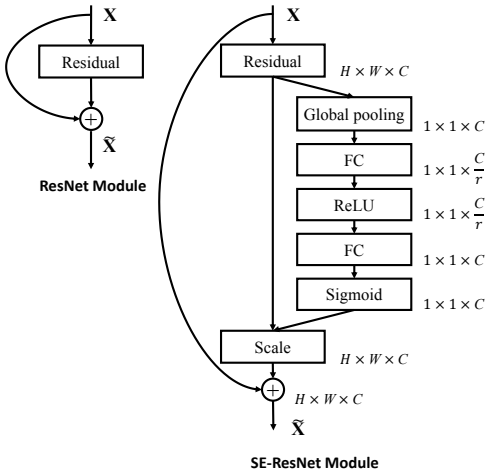


Figure 3.24: The schema of the original Residual module (left) and the SE-ResNet module (right).

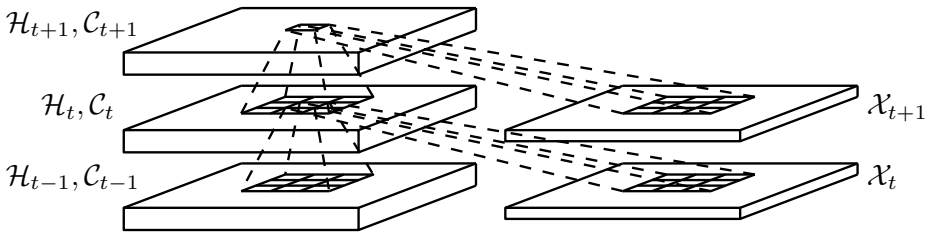


Figure 3.25: Inner structure of ConvLSTM

3.29 $*$ is a convolution operator and $\circ$ is the Hadamard product. The Equations 3.29 corresponds to the diagram in Figure 3.25. Their network captures spatio-temporal correlations better than competitors and achieves state of the art.

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci} \circ c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} \circ c_{t-1} + b_f) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co} \circ c_t + b_o) \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
\tag{3.28}
$$

$$
\begin{aligned}
i_t &= \sigma(W_{xi} * \mathcal{X}_t + W_{hi} * \mathcal{H}_{t-1} + W_{ci} \circ \mathcal{C}_{t-1} + b_i) \\
f_t &= \sigma(W_{xf} * \mathcal{X}_t + W_{hf} * \mathcal{H}_{t-1} + W_{cf} \circ \mathcal{C}_{t-1} + b_f) \\
\mathcal{C}_t &= f_t \circ \mathcal{C}_{t-1} + i_t \circ \tanh(W_{xc} * \mathcal{X}_t + W_{hc} * \mathcal{H}_{t-1} + b_c) \\
o_t &= \sigma(W_{xo} * \mathcal{X}_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \circ \mathcal{C}_t + b_o) \\
\mathcal{H}_t &= o_t \circ \tanh(\mathcal{C}_t)
\end{aligned}
\tag{3.29}
$$

3.3.3 Convolutional Neural Networks in Event-Based Data

ConvNets are widely used in event-based data for numerous applications. This has been an inspiration for our work.

Video Synthesis [92] Introduced the use of recurrent network for video synthesis from an event stream and surpasses other approaches by a large margin. Their approach was later improved in a follow-up paper [93]. [102] took inspiration from [59] for the architecture of the network. They used this shallow and lightweight convolutional recurrent network for the task of video reconstruction from events.

Object Detection In [83], the authors introduce a novel deep convolutional recurrent neural network to detect objects in an event stream without any reconstruction of intensity images.

Those approaches have shown the great potential of ConvNets, and more specifically convolutional recurrent neural networks, applied to event-based data. They enable more efficient computations, as well as more robustness to varied types of noises.

3.4 Image and Gradient Prediction

Events are a very novel way to represent visual information and none of the traditional (i.e. frame-based) computer vision algorithms are adapted to deal with such inputs. A natural reaction when dealing with a new representation is to try to recreate a known representation for which a large amount of literature and algorithms exists. Many papers work on creating dense representations for event-based data which we have presented in Section 3.1. We will focus in this part on methods who explicitly represent an event stream with images or image gradients. Some methods try to recreate images and gradients directly but suffer from multiple noise patterns in event cameras which degrades the reconstructions. Others use deep learning to learn a

mapping from events to images and successfully use neural network to learn and remove the noise coming from event sensors. When reconstructing images from events the benefits of neuromorphic cameras are not lost. The reconstructed images will not suffer from motion blur or loss of details in scenes with high dynamic range due to the nature of events.

3.4.1 Handcrafted Methods

Using the direct theoretical link between events and log intensity some research has been done in the direction of direct image or image gradient reconstruction from events.

Gradient prediction In 2011, Cook et al. [25] first introduce the idea of gradient and image reconstruction from an event stream. Their approach is however limited to rotation only camera movements. Building on this work [61], also relying on the known movement of the camera (a pure rotation), estimate at the same time the camera motion and the image gradients corresponding to such rotation. The gradients are a step in the process of reconstruction gray levels by solving a partial differential equation. In [62] the authors reconstruct gradients and intensity from a joint optimisation and estimation of the 3D motion of the camera, the gradients and the inverse depth of the scene per pixel. Everything is done pixel per pixel which can be intractable with modern cameras and high event rates. Another issue of the approach is the slow convergence of inverse depth estimates limiting the amplitude of the camera motion. In [91] the simultaneous tracking of the camera and semi-dense mapping of the scene give all the tools needed to estimate the gradients of the mapped scene and use said gradients to reconstruct the intensity using a Poisson's reconstruction. These estimations are all done under the assumption of constant brightness of the scene which limits the quality of the gradients and intensity reconstruction, as well as the generalization to dynamically lit scenes.

Direct intensity prediction In [11] the estimation of the intensity is done jointly with a dense optical flow field, using a variational optimisation problem. The optimisation is however very costly and the quality of the reconstructed intensity levels are not precise enough to detect keypoints. In [82] they reconstruct the intensity image from events via integration of the events similar to [11] they however introduce an event manifold for regularisation and they work on an event-by-event basis. A complementary filter is introduced in [101] to fuse events and frames, reconstructing high temporal resolution and high dynamic range images. The approach obtains better results than the ones in [11], by using frames in addition to events. When using only

events as inputs of the filter the results are not convincing and are very far from what would be needed to detect keypoints consistently.

Issues with handcrafted methods We have seen a number of different methods recreating directly or indirectly intensity images from an event stream. They do not model or learn noise patterns and except for a sliding window in [11] or the event manifold in [82] do not focus on this issue, creating a number of artifacts such as ghosting effects and bleeding edges in the reconstruction. The bleeding edges come from the fact that the contrast threshold (value of the minimum brightness change to create an event) is theorize to be constant over the pixels. In reality it is neither constant nor uniform across the pixels. The ghosting effect results from the nature of events being a relative measure. From a set of events it is possible to integrate them up to an unknown initial image. This initial image can remain visible over a period creating what is called a ghosting effect. As long as the noise in event cameras is not modeled correctly it will prevent any reconstruction using analytical methods as their basis to be perfect. Some methods therefore make use of machine learning to recreate intensity images from events using more complex models. We will present them in the next subsection.

3.4.2 Learned Methods

[12] were the first authors who learned gradients from event data. The gradients are obtained using a learned dictionary mapping between small patches of accumulated events and corresponding gradients. The correspondence between the events and the gradients is done using a simulator. They use the learned gradients as an intermediate representation to reconstruct an image using the Poisson reconstruction. Other learned methods focused on reconstructing images directly. Improving the image reconstruction quality is necessary for many tasks such as super slow motion, High Dynamic Range (HDR) video, deblurring etc.

cGAN Multiple paper use conditional Generative Adversarial Networks as a way to predict images from events. [58] introduce the idea of using a conditional GAN to predict images from events. They use a very sparse 3D volume as input to the generator network and train it using a mix of real data captured with the DAVIS camera and synthetic data using the Open Event Camera Simulator (ESIM) [89]. Their method is later adapter in [108] to predict the depth and the optical flow with minor modifications, proving the strength of the approach. [84] is able to synthesize RGB images from events and an initial or a periodic set of color key-frames. They use a combination of Generator and Discriminator as in [44] coupled with a convolutional

LSTM [124] to refine the RGB predictions and improve the temporal coherence. They also use high frame rate RGB datasets to create synthetic events. Using larger RGB datasets improves the model generalisation to novel scenes while using real events help the model to learn real noise distribution. The main issue with this method is the need for an hybrid sensor producing both frames and events which is not the case in most event cameras.

Event and frames fusion [123] have an analytical approach to frame and event fusion using a module named the Differential model-based reconstruction. The issue, as previously stated, with such a reconstruction is the noise of the event camera. Indeed, the noise when not fully taken into account in the model, has a negative impact on the frame reconstruction: it creates unwanted artifacts. To remove said artifacts, the authors model them as additive noise and remove them using deep residual learning. The residual learning is done with a deep convolutional neural network. The approach however, suffers from the same limitations, as [84], of needing a reference frame and inability to work using only events.

E2VID In 2019, [92] created a paradigm shift in image reconstruction from events. They introduced a novel recurrent network using only events as inputs. The network they use has an integrated recurrence (overview in Figure 3.26) enabling them to reuse previously reconstructed frames and improving temporal consistency. They train the network with data generated using a more complex and realistic simulator [90]. More specifically the simulator is based on the observation made in [68], that the contrast threshold is not constant, but rather normally distributed across the pixel array. [90] therefore sample the contrast threshold at every step of the simulation. They also differentiate the positive and negative contrast threshold which are sampled independently. They trained their network using the calibrated perceptual loss (LPIPS) from [128]. The loss consists in passing the ground truth image and the reconstructed frame through a VGG network [105] trained on ImageNet and measure the distance between features at multiple depths. Qualitative results can be seen in Figure 3.27 where we can observe the realism of reconstructed frames compared to previous methods.

E2VID+ In [93] the authors of E2VID changed the initial loss presented in [92]. They add some temporal consistency constraints to remove some of the flickering observed in their previous work. The loss they introduced (based on [64]) is using the known optical flow $\mathcal{F}$ from the simulated data. The equation for the temporal consistency (TC) loss is:

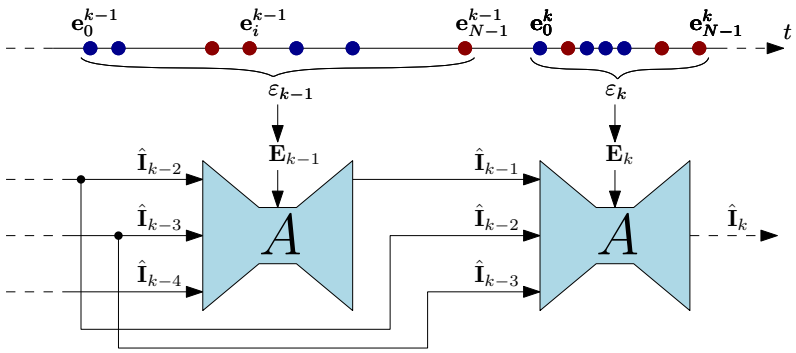
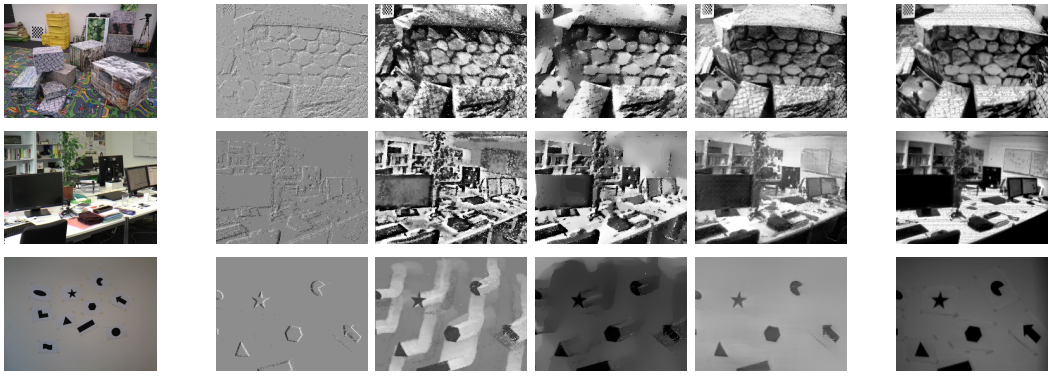


Figure 3.26: Overview of [92]'s approach. The event stream (depicted as red/blue dots on the time axis) is split into windows ε_k with N events in each. Each window is converted into a 3D event tensor $\mathbf{E}_k$ and passed through the network together with the last K reconstructed images to generate a new image reconstruction $\hat{\mathcal{I}}_k$. In this example, $K = 3$ and $N = 7$. Image courtesy of [92].



(a) Scene overview (b) Events (c) HF (d) MR (e) E@VID (f) Ground truth

Figure 3.27: Comparison of [92]'s method with Manifold Regularization (MR) and High-pass Filter (HF) on sequences from [81]. [92]'s network reconstruct fine details well (textures in the first row) compared to the competing methods, while avoiding their artifacts (e.g. the "bleeding edges" in the third row).

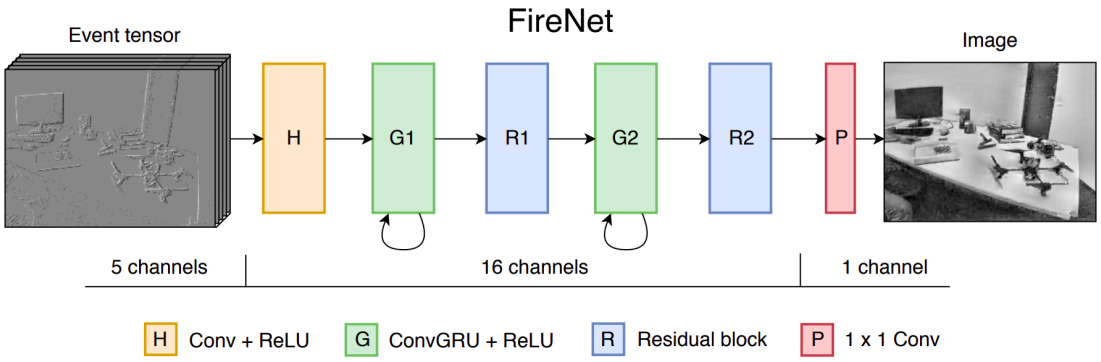


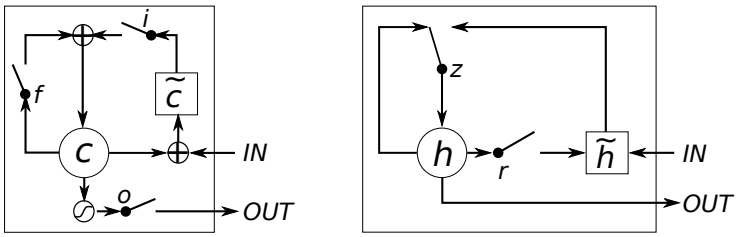
Figure 3.28: FireNet architecture. The input is an event tensor with 5 temporal bins. The network consists of convolutional layers (H, P), convolutional gated recurrent units (G1, G2) and residual blocks (R1, R2). Every layer uses ReLU activation except the final layer (P).

$$\mathcal{L}_k^{TC} = \mathcal{M}_k^{k-1} \|\hat{\mathcal{I}}_k - \mathcal{W}_k^{k-1}(\hat{\mathcal{I}}_{k-1})\|_1, \tag{3.30}$$

where $\mathcal{W}_k^{k-1}(\hat{\mathcal{I}}_{k-1})$ is the reconstructed image $\hat{\mathcal{I}}_{k-1}$ warped to $\hat{\mathcal{I}}_k$ using optical flow $\mathcal{F}_k^{k-1}$ and $\mathcal{M}_k^{k-1} = \exp(-\alpha \|\mathcal{I}_k - \mathcal{W}_k^{k-1}(\mathcal{I}_{k-1})\|_2^2)$ is a weighing term to account for occlusion. When the error between the images $\mathcal{I}_k$ and warped image $\mathcal{W}_k^{k-1}$ is high, the weight goes to zero and the error between reconstructed images is ignored. $\alpha = 50$ in their work.

Fast Image Reconstruction with an Event Camera In [102] the authors introduce a novel architecture inspired from [59] and manage to reduce the number of parameters from 10M to 38k and lower inference times from 30ms to 10ms compared to the state of the art [93]. The architecture can be seen in Figure 3.28. As opposed to [93] they use ConvGRU [10] instead of ConvLSTM [103]. As can be seen in Figure 3.29 (from [22]), the Gated Recurrent Unit (GRU) [21] is found to be comparable to the Long Short-Term Memory (LSTM) [53] unit while reducing computational and memory footprint. Indeed, the FireNet method performs three times faster than E2VID on GPU, and up to four times faster on CPU, requiring less than one tenth the number of FLOPs.

Conclusion In practice using image reconstruction methods for keypoint detection does not work well. Keypoints are not correctly localized spatially or temporally due to the imperfection in the reconstruction. The spatial precision is poor and detected keypoints tend to move around a point in space (jittering). The temporal stability is



(a) Long Short-Term Memory (b) Gated Recurrent Unit

Figure 3.29: Illustration of (a) LSTM and (b) gated recurrent units. (a) i, f and o are the input, forget and output gates, respectively. c and $\tilde{c}$ denote the memory cell and the new memory cell content. (b) r and z are the reset and update gates, and h and $\tilde{h}$ are the activation and the candidate activation.

also an issue. keypoints turn on and off alternatively without any major visual change of the scene (flickering). This was observed in [92] and improved in [93] yet results on keypoint detection leave to be desired when directly applied to reconstructions. When detecting keypoints using the Harris formula one needs to compute the image gradients first, it is therefore more natural to learn to predict gradients directly, rather than predict images followed by a computation of image gradients. In addition, the nature of events simplify the prediction of gradients as opposed to the prediction of images.

Chapter 4

Data Generation

In this chapter we will present how one can generate data for event-based vision applications. We introduce the notion of data generation for event-based in section 4.1, describe how hybrid senors, beam splitter or planar scenes can be used to generate ground truth data in Section 4.2, and present in Section 4.3 the work we have done and improvements we made to simulate events from high frame rate videos (natural or synthetic).

4.1 Introduction

From the beginning all our work has been focused on trying to train a network to predict keypoints. We could train our network in two ways: directly (supervised) or using a proxy task (self-supervised). We decided to focus our research on supervised training and therefore to remove the difficulty of finding a proxy task for self-supervision. It was difficult to find any kind of annotated data which we could use for keypoint detection, either directly or indirectly, as was done in automatic annotation of humans in videos [31]. In this work the authors benefited from the scripts of movies and used them as a supervision. Annotated data for keypoints in an event stream does not exist directly and is very difficult to create in practice. We will detail in the following sections some ways to create a dataset of keypoints with recorded data from a sensor: either aligning two cameras and syncing them using a one way mirror or as was done in [73] using a sensor where both gray levels and events are available. The other way to generate some ground truth data is by using a simulator. The simulator has multiple advantages: it will always have perfectly matching frames and events; it can generate gray levels at a very high (as high as needed) frame rate. Real data is useful when the sensor does not change between training and validation, and the noise levels are not previously known but need to be modeled. On the other

hand when trying to create an approach which can generalize to multiple sensors, a simulator is most useful. It enables us to randomly change the noise modelling and therefore avoid overfitting of the network to a specific noise model. To be able to model noise as truthfully as possible, however requires some detailed understanding and knowledge of noise in event sensors.

4.2 Real Data

4.2.1 ATIS

As was done in [73] a good way to get a correspondence between events and corners is to use the Asynchronous Time-based Image Sensor (ATIS) presented in Figure 4.1. It gives a direct correspondence between events and gray levels or grayscale measurement. This makes it possible to detect keypoints in the gray levels using traditional methods such as the Harris corner detector and classify corresponding events as keypoints or non keypoints. This approach was used successfully in [73]. However the method is dependent on gray levels which do not exist in most event-based sensors such as the one presented in [36]. As we tried to detect keypoints in higher resolution (ATIS is QVGA) we could not use this method. As [73] was state-of-the-art we tried to use the detected keypoints of the their method (SILC) on higher resolution sensors. The SILC method is very slow and unusable for high resolution sensors as then event rate can augment drastically and the Speed Invariant Time Surface cannot be computed in real time. We tried to use the SILC keypoints as ground truth for higher resolution, as the computation could have been done offline, and train a smaller and faster convolutional neural network for inference. However this knowledge distillation was not a success as despite good generalisation power of the SILC method to other sequences in the dataset, when using this method on another sensor with higher resolution such as the one presented in [36] the method failed to detect coherent and consistent keypoints both in space and time.

4.2.2 Alignment of Two Cameras

Another method to have a correspondence between events and gray levels or even RGB images is to use two cameras and a beam-splitter as was done in [74]. Their approach is illustrated in Figure 4.2. Some hardware synchronisation of the camera clocks is needed as well as some calibration of the cameras. The perfect pixel matching between the event camera and the RGB camera is not trivial to achieve and we decided not to pursue this idea. At the time of our research the dataset was not available. An issue of this method is the need to choose between a very high speed camera

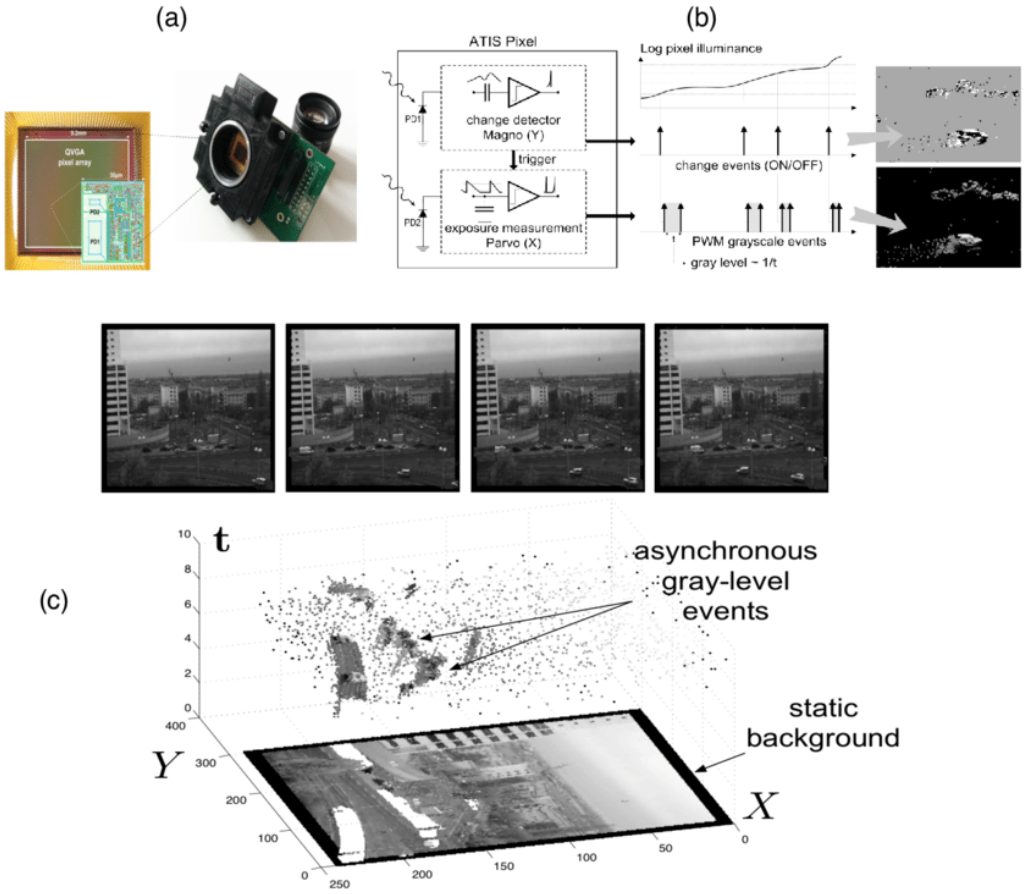


Figure 4.1: ATIS, Asynchronous Time-based Image Sensor: (a) The ATIS and its pixel array, made of 304x240 pixels (QVGA). PD1 is the change detector, PD2 is the grayscale measurement unit. (b) When a contrast change occurs in the visual scene, the ATIS outputs a change event (ON or OFF) and a grayscale event. (c) The spatio-temporal space of imaging events: static objects and scene background are acquired first. Then, dynamic objects trigger pixel-individual, asynchronous gray-level events after each change. Frames are absent from this acquisition process. Samples of generated images from the presented spatio-temporal space are shown in the upper part of the figure. Figure courtesy of [38]

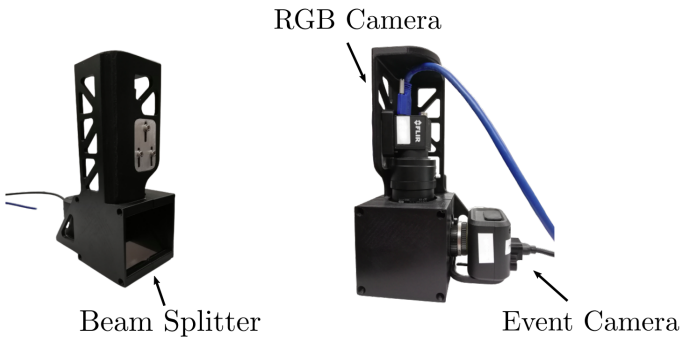
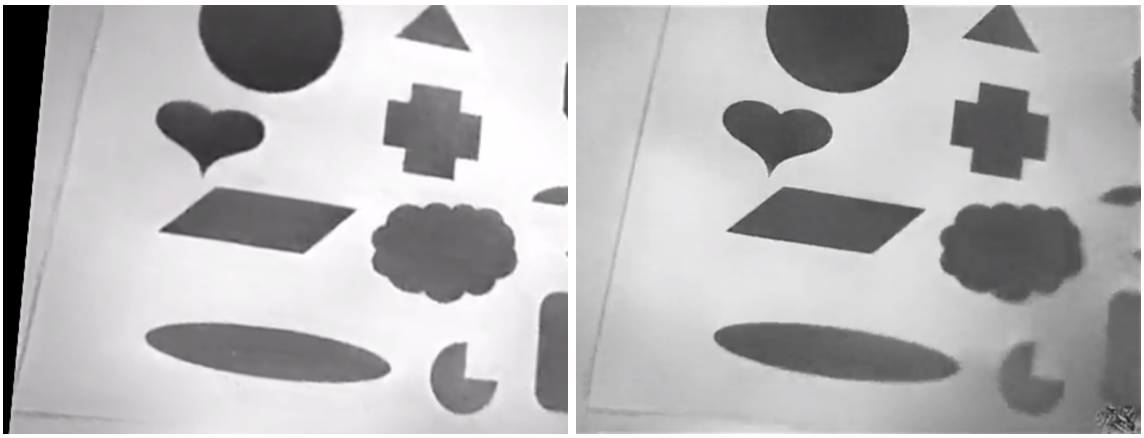


Figure 4.2: The beam-splitter setup used to record the new HDR-ERGB dataset. It combines an event and RGB camera by projecting the scene via a beam-splitter mirror to both cameras.

which can be quite expensive or settle for discrete keypoint locations in time. Indeed the position of the keypoint in time would only be measured when a frame from the standard camera is recorded. Interpolation could be done in between frames necessitating some tracking and potentially lowering the overall quality of the ground truth.

4.2.3 Planar Surface

When [73] released their dataset and novel evaluation metric using only planar scene, we had an idea to generate a very stable ground truth. Indeed when using the Harris corner detector multiple times for the same corner we can run into two issues. The first one is that the gray level reconstruction from ATIS is not perfect and noise can disturb the corner detection leading to instability of the detection in space and time. The second issue comes from the fact that the Harris corner detector is not scale invariant and depending on the level of movement towards or away from the image, Harris corners can be falsely classified as background. Using the fact that the scene was planar meaning an homography between the first and every image in the sequence exists we decided to explicit every homography corresponding to the sequence generation. The first step of the method was to recreate some gray levels using event integration (the reconstructed gray levels are noisy but this is not an issue). With the reconstructed gray levels for every timestep needed we are then able to search for the homography corresponding to the camera movement. Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization presented in [33] was used to find the homography. The method can be initialized manually if



(a) (b)

Figure 4.3: (a) The gray level with the least noise is used as a comparison to other gray levels and is warped using an homography to match other gray levels in the sequence (b) current gray level in the sequence to which the warped gray level ((a)) needs to match. The matching is done using Parametric Image Alignment Using Enhanced Correlation Coefficient Maximization [33].

needed, in practice we found manual initialization to be unnecessary when the gray levels did not contain too much noise. We show an example of the warped gray level corresponding to the current gray level in Figure 4.3. Using the now known homographies from the sequence we have to choices: either use the Harris corner detector on the first image of the sequence and reproject the corners to the whole sequence (i.e. to each subsequent frames using the corresponding homography) or, if the quality of the detector is not what we wish for, we can manually annotate the first frame and reproject corners. As the images correspond to events we have now a continuous set of events and corners to train a neural network or any other learned method.

4.3 Simulated Data

We also studied the possibility of using simulated data for our keypoint detection ground truth generation. The main advantage of simulation is having access to unlimited data both in term of the variability in appearance but also in the type of noise pattern helping ourselves to be robust to changes in appearance and changes in the amount and different types of noise. Another advantage of simulation is the ability to instantly label keypoints with pixel perfect accuracy and perfect time synchronization. The issue with simulated data is the absolute necessity to be as close

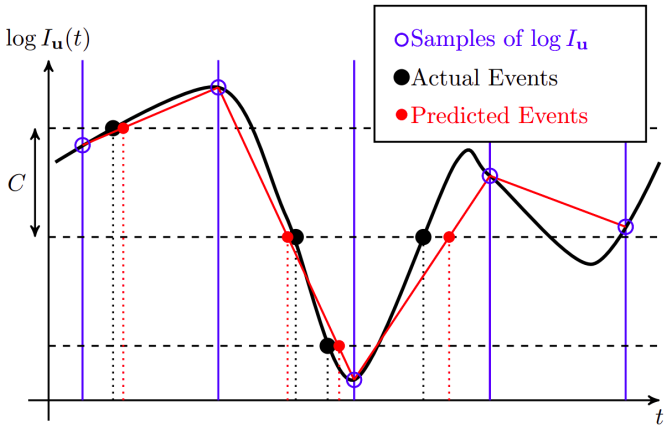
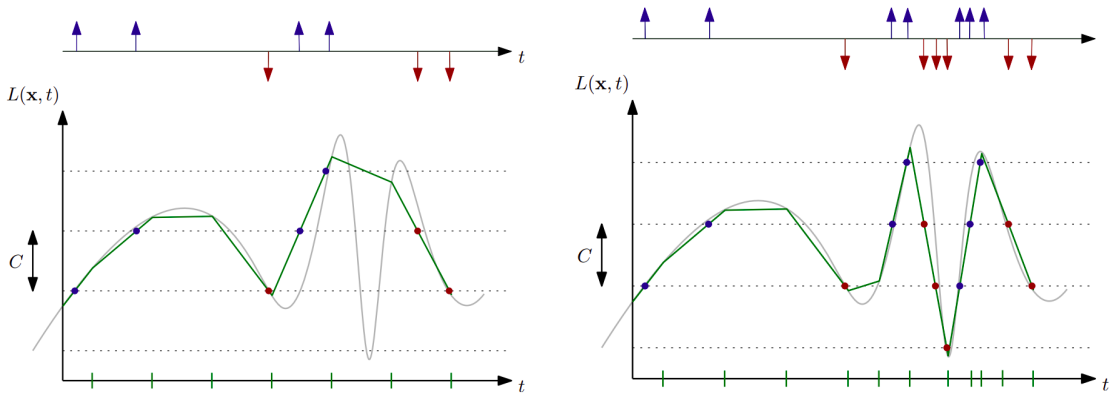


Figure 4.4: DAVIS Simulator. Per-pixel event generation using piecewise linear time interpolation of the intensities given by the rendered images. For simplicity, images were rendered at a fixed rate. Figure courtesy of [80]

as possible to real data. This proves to be very difficult to achieve in practice.

The concept was introduced in [80] where they used the computer graphics software Blender to generate thousands of rendered images along a specific trajectory. The images were generated to always have less than 1/3 of a pixel motion between consecutive frames. The authors create a Surface of Active Events and use time interpolation of the rendered image intensities to determine brightness changes. This interpolation removes the need to generate millions of images (up to microsecond-resolution) yet make it possible to reconstruct a piecewise linear approximation of the continuous underlying visual signal. The time interpolation approach is illustrated in Figure 4.4. The sampling strategy in [80] is uniform and bounded by the displacement of pixels, while in [89] a novel sampling strategy called the adaptive sampling strategy is introduced (see Figure 4.5 for comparison with the uniform sampling strategy of [80]). Adapting the sampling rate based on the predicted dynamics of the visual signal requires tight coupling between the rendering engine and the event simulator. The sampling can be done based on brightness change or pixel displacement. A second addition of [89] is the non constant contrast threshold. Indeed as originally observed in [68] the contrast threshold is normally distributed across the pixel grid. Therefore, at every simulation step the contrast is sampled according to $\mathcal{N}(C, \sigma_C)$, where σ_C controls the amount of noise and can vary. The mean C can be set independently for the positive (C_+) and negative contrast threshold (C_-) to simulate a real event camera more accurately.

Some additional improvement to the simulator have been introduce in [56] such



(a) Uniform Sampling [80] (b) adaptive Sampling [89]

Figure 4.5: Comparison of uniform sampling (a) versus adaptive sampling (b). The timestamps at which brightness samples are extracted are shown on the time axis as green markers. While both strategies produce similar events when the signal varies slowly (left part), the uniform sampling strategy fails to faithfully simulate events in the fast-varying part of the signal. By contrast, the adaptive sampling strategy extracts more samples in the fast-varying region, thus successfully simulating events. Figure courtesy of [89].

as finite intensity dependent bandwidth, and intensity-dependent noise. Since the real DVS pixel bandwidth is proportional to intensity [56] models this effect for each pixel by making the filter bandwidth (BW) increase monotonically with the intensity value. To avoid nearly zero bandwidth for small digital number pixels, an additive constant limits the minimum bandwidth to about 10% of the maximum value. They add multiple kind of noises:

- **Hot pixels** pixels who continuously fire events at a high rate even in the absence of input
- **Leak noise events** ON events emitted spontaneously called leak events
- **Temporal noise** To model shot noise they generates ON and OFF temporal noise events with lower probability in bright parts

On the other hand, [60] simplifies the latency and the noise models. In addition, to more closely model the behaviour of a real pixel, the readout circuitry is modelled, as this can strongly affect the time precision of events in complex scenes (shown in Figure 4.6).

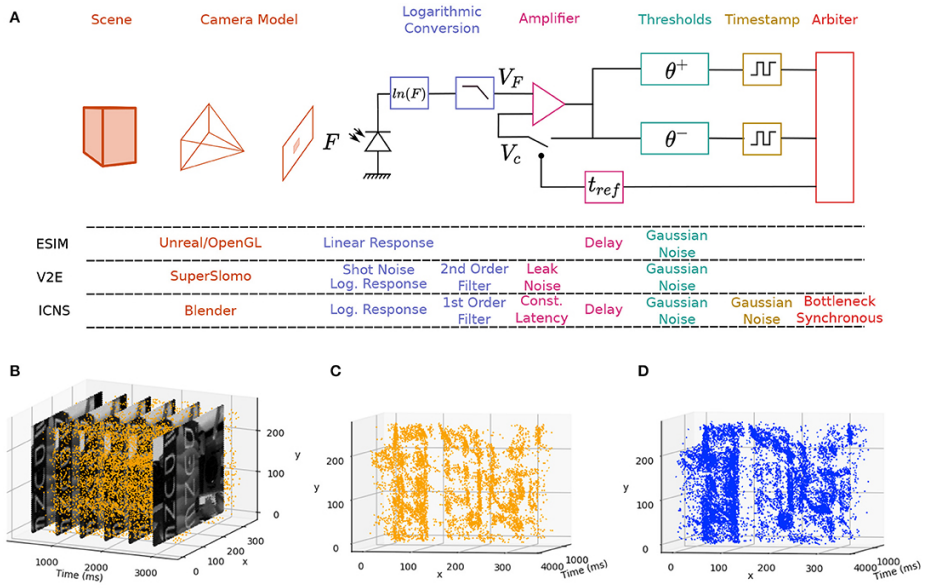


Figure 4.6: (A) Main blocks of an imaging system based on a DVS camera simulated by ESIM, V2E, and ICNS. The Scene and Camera Model are often part of the same rendering tool, but slow motion interpolation neural networks such as SuperSlomo can help provide better time precision. The Logarithmic Conversion block encapsulates the conversion of the light into an electrical signal before the Amplifier and the Thresholds. In the ICNS model, the noise is also injected before the Timestamp block and the noise distribution is a function of the light level. (B) Events and frames from the DAVIS and comparison between the original events (C) and the simulated events (D) using the ICNS model. Figure courtesy of [60].

We have decided to base our simulator on [40] and [56]. We kept the correspondence we had using planar scenes between real events and synthesized video to run our simulator. This made it possible to directly compare the noise levels and distribution between simulated events and recorded events. We simulated only planar sequences to be able to use the known homographies for ground truth reprojection instead of detecting keypoints repeatedly. This is a key part of our data generation pipeline as it creates a very stable ground truth which is paramount to train the network successfully. Another key advantage of using reprojected ground truth is that it enables us to use hard negative mining without emphasizing on possible errors in the ground truth during training. To ensure the most diversity possible in the training set we used the images from the COCO dataset [70] which is made of 200K images and more than 1.5 million object instances.

4.4 Conclusion

We have presented in this chapter the general approaches to generate ground truth to train algorithms for tasks in event-based vision. While using data recorded using a beam splitter can lead to impressive results [117] we have decided to work with simulated data for two key reasons:

- The noise modeling is good enough to enable neural network generalisation on real data when randomly sampling noise parameters of the simulator
- The pixel perfect alignment between events and frames enable us to be very accurate when generating the data, leading to very accurate predictions

Indeed to achieve similar robustness to noise using a beam splitter we would have needed to record multiple scenes with multiple sensors with very precise calibration to ensure pixel perfect alignment. This proves to be both impractical and costly.

We will present in the next chapter how we used our simulator to train a recurrent convolutional neural network to predict image gradients from events leading to accurate keypoint detection computing the Harris corner score directly on the predicted gradients.

Chapter 5

Detecting Stable Keypoints from Events through Image Gradient Prediction

We present in this Chapter our approach of predicting image gradients as an intermediate step for keypoint detection from events. We first introduce our method and explain the rational behind it in Section 5.1. We present our method in greater details in Section 5.2 and present our results in Section 5.3 before concluding the chapter in Section 5.4

5.1 Introduction

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.

5.2 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 C, an event $e_i = (x_i, y_i, p_i, t_i)$ is generated for pixel (x_i, y_i) if

$$\left| \log \left(\frac{L(x_i, y_i, t_i)}{L(x_i, y_i, t_{i-1})} \right) \right| \geq C \,, \tag{5.1}$$

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:

$$p_i = \text{sign} \left(\log \left(\frac{L(x_i, y_i, t_i)}{L(x_i, y_i, t_{i-1})} \right) \right) \,. \tag{5.2}$$

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

$$\begin{aligned} G(x, y, t_N) &= \nabla_{xy} L(x, y, t_N) \\ &= (L_x(x, y, t_N), L_y(x, y, t_N))^\top \,. \end{aligned} \tag{5.3}$$

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

In [100], $\mathcal{F}$ 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 [124], 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

$$M = w_\sigma * (G \cdot G^\top) \,, \tag{5.4}$$

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:

$$S = \det(M) - k \cdot \text{tr}(M)^2 \,. \tag{5.5}$$

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.

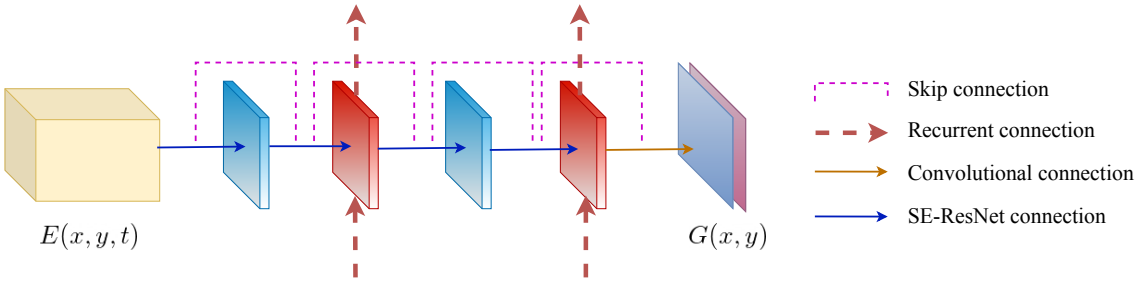


Figure 5.1: **Proposed Architecture.** Our architecture combines recurrent connections with convolutional and Squeeze-and-Excite ResNet connections. The input to the network is given by an event cube $E(x, y, t)$ [130], computed at every N events. The event cube is given as input to a Squeeze and Excite 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 $G(x, y)$.

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 5.1. It is a 5-layer fully convolutional network of 3×3 kernels. Each layer has 12 channels and residual connections [51]. 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 [55]. 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.

Training

For training, we use a subset of the training images in the COCO dataset [70]. As shown in Figure 5.2 and detailed in the Listing 5.1, 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

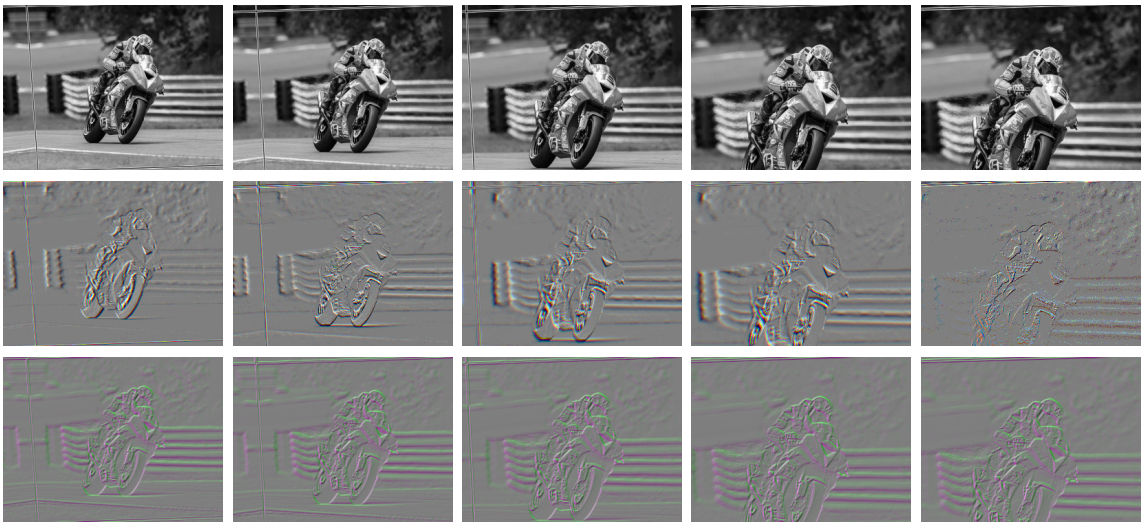


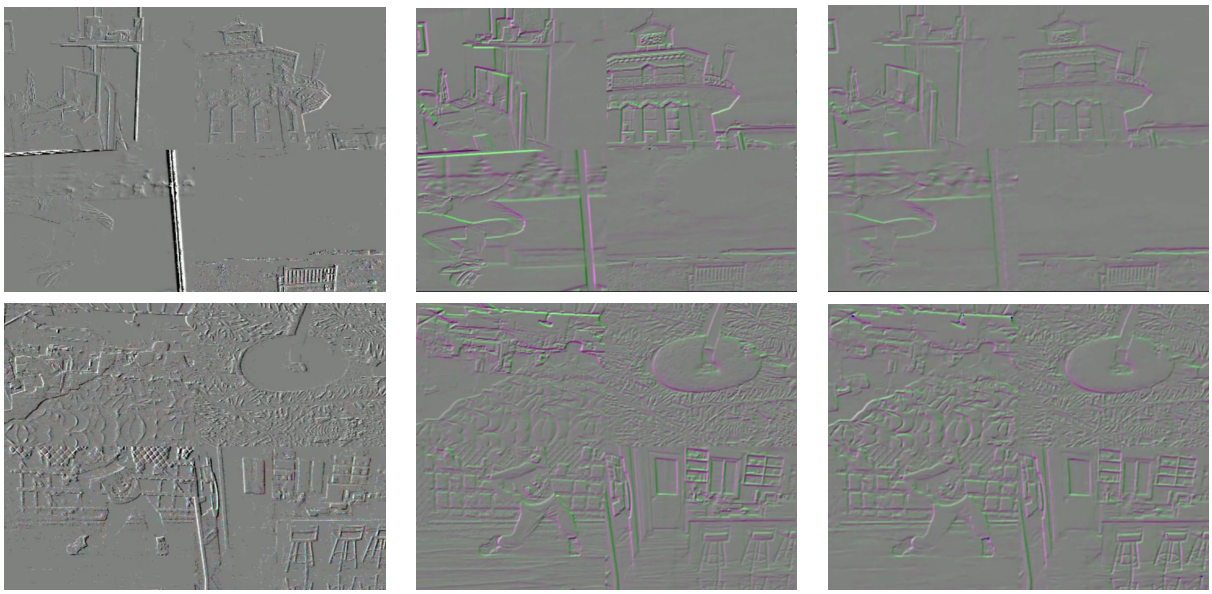
Figure 5.2: **Generating Training Data.** First row: Given a training image, we apply homographies to warp it and generate a video sequence. Second row: From the video sequence we generate events using a simulator and then compute the event cubes [130] used as input to our network. Third row: Ground-truth gradients of the video sequence. We train the architecture of Figure 5.1 to predict image gradients from events. Finally, at inference time, we compute the Harris score from the predicted gradients.

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 [90] and [27]. From the events, we build an event cube (see Section 5.2) composed of 5 channels. At each iteration, we use a batch made of 8 different sequences of 20 time bins. The input size 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 [97]. 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 $1e^{-4}$.

To generate a smooth video by applying homographies to an initial image, we 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, from which we can compute homographies. For completeness, our code is given in the Listing 5.1.

```
import numpy as np
import cv2

DTYPE = np.float32


def generate_homography(rvec, tvec, nt, depth):
    """
    Generates a single homography

    Args:
        rvec (np.array): rotation vector
        tvec (np.array): translation vector
        nt (np.array): normal to camera
        depth (float): depth to camera
    """
    R = cv2.Rodrigues(rvec)[0].T
    H = R - np.dot(tvec.reshape(3, 1), nt) / depth
    return H


def generate_image_homography(rvec, tvec, nt, depth, K, Kinv):
    """
    Generates a single image homography

    Args:
        rvec (np.array): rotation vector
        tvec (np.array): translation vector
        nt (np.array): normal to camera
        depth (float): depth
        K (np.array): intrisic matrix
        Kinv (np.array): inverse intrinsic matrix
    """
    H = generate_homography(rvec, tvec, nt, depth)
    G = np.dot(K, np.dot(H, Kinv))
    G /= G[2, 2]
    return G

def generate_smooth_signal(num_signals, num_samples,
                           min_speed=1e-4, max_speed=1e-1):
    """
    Generates a smooth signal
```

```
43
44      Args:
45          num_signals (int): number of signals to generate
46          num_samples (int): length of multidimensional signal
47          min_speed (float): minimum rate of change
48          max_speed (float): maximum rate of change
49      """
50      t = np.linspace(0, num_samples - 1, num_samples)
51
52      num_bases = 10
53      samples = 0 * t
54      samples = samples[None, :].repeat(num_signals, 0)
55      for i in range(num_bases):
56          speed =
57              np.random.uniform(min_speed,
58                                max_speed,
59                                (num_signals,))[:, None]
60          phase = np.random.uniform(np.pi, 10 * np.pi)
61          test = np.sin((speed * t[None, :]) + phase)
62          samples += test / num_bases
63
64      # final
65      speed =
66          np.random.uniform(min_speed,
67                            max_speed,
68                            (num_signals,))[:, None]
69      test = np.sin((speed * t[None, :]))
70      test = (test + 1) / 2
71      samples *= test
72      return samples.astype(DTYPE)
73
74
75  max_frames = 100 # number of homographies needed
76  signal = generate_smooth_signal(6, max_frames).T
77  rvecs = signal[:, :3]
78  tvecs = signal[:, 3:]
79  nt = np.array([0, 0, 1], dtype=DTYPE).reshape(1, 3)
80  depth = np.random.uniform(1.0, 2.0)
81  K = np.array(
82                [[width / 2, 0, width / 2],
83                 [0, height / 2, height / 2],
84                 [0, 0, 1]],
```



Input Event Cube Ground-truth Image Gradients Predicted Image Gradients

Figure 5.3: **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.

```
                dtype=DTYPE,
            )
Kinv = np.linalg.inv(K)

for frame_index in range(max_frame):
    rvec, tvec = rvecs[frame_index], tvec[frame_index]
    homography = generate_image_homography(rvec, tvec, nt,
                                           depth, K, Kinv)
```

Listing 5.1: Random Homography Generation

Inference

At inference, we build an event cube (see Section 5.2) 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. (5.5) using the kornia [97] 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 5.3.

Input Representation

The input to our network is a $H \times W \times B$ event tensor $E(x, y, t)$, as proposed in [130]. 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

$$E(x, y, t_n) = \sum_i p_i \max(0, 1 - |t_n - t_i^\star|) \,, \tag{5.6}$$

with

$$t_i^\star = \frac{(t_i - t_{min})}{(t_{max} - t_{min})}(B - 1) \,, \tag{5.7}$$

and where n is the temporal bin index, p_i is the polarity, and $t_i^\star$ is the normalized timestamp of the i^{th} 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.

5.3 Results

Quantitative Results

For evaluating our method, we consider the ATIS Corner Dataset [73], 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 [73], that is we compute reprojection error by estimating a homography form the tracked corners. As in [73], 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 5.1. As can be seen, our method has the best tracking length, almost 5 times longer than the second best method [73]. This is

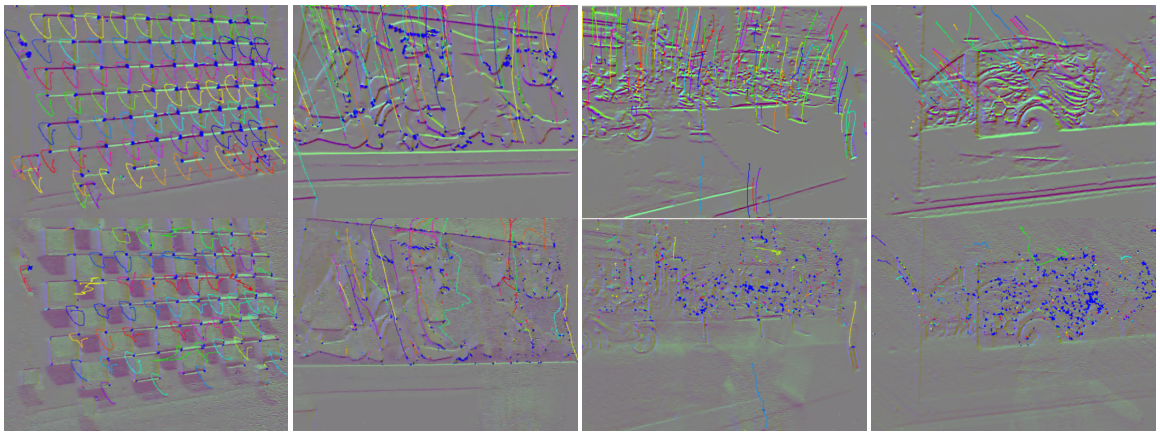


Figure 5.4: **Qualitative Results on the ATIS Corner Dataset.** First row: Gradients predicted by our network, with overlaid tracks returned by our method. Second row: Results obtained by replacing our network by the method of [100], which predicts gradients from events by using a model-based approach. 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.

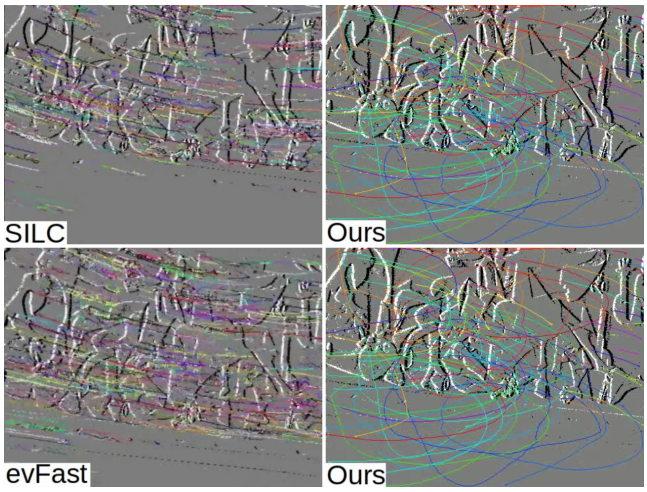


Figure 5.5: **Qualitative comparison to previous state-of-the-art methods.** First row: we visually compare our method with the previous state-of-the art SILC [73]. Second row: Our method is compared to evFast method [78]. Here we visualize the events with different polarities for an easier comparison but our method still computes corners on the predicted gradients (best seen in electronic format).

Table 5.1: Evaluation on the ATIS Corner Dataset [73] for $\Delta t = 25ms$. Our method has 5 times longer tracks, while maintaining similar reprojection error as the state-of-the-art.

	evHarris [120]	evFast [78]	Arc [7]	$SILC^*$ [73]	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**

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.

Qualitative Results

Figure 5.3 compares ground truth gradients with gradients predicted by our architecture, and shows the predicted gradients are very similar to the real ones.

Figure 5.4 compares Harris keypoints extracted using gradients predicted with our recurrent architecture and Harris keypoints extracted using gradients estimated by the handcrafted method [100]. Only the gradient computation is different, all the other computations are the identical. Our method predicts much more accurate gradients, yielding much more stable keypoints.

Figure 5.5 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.

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 [92] 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.

5.4 Conclusion

We have seen in this Chapter how image gradients can efficiently be used as an intermediate representation between events and keypoints. The direct computation of image gradients from events is too noisy and does not enable accurate keypoint detection. This has justified our choice to use a small recurrent convolutional neural network to predict image gradients from the stream of events. The predicted gradients contain less noise and are more consistent through time helping with the accuracy and the track lifetime. However we would like to avoid the intermediate representation to limit computations as well as remove scale dependence of keypoint detection inherent to the Harris corner detection operator.

Chapter 6

Long-Lived Accurate Keypoints in Event Streams

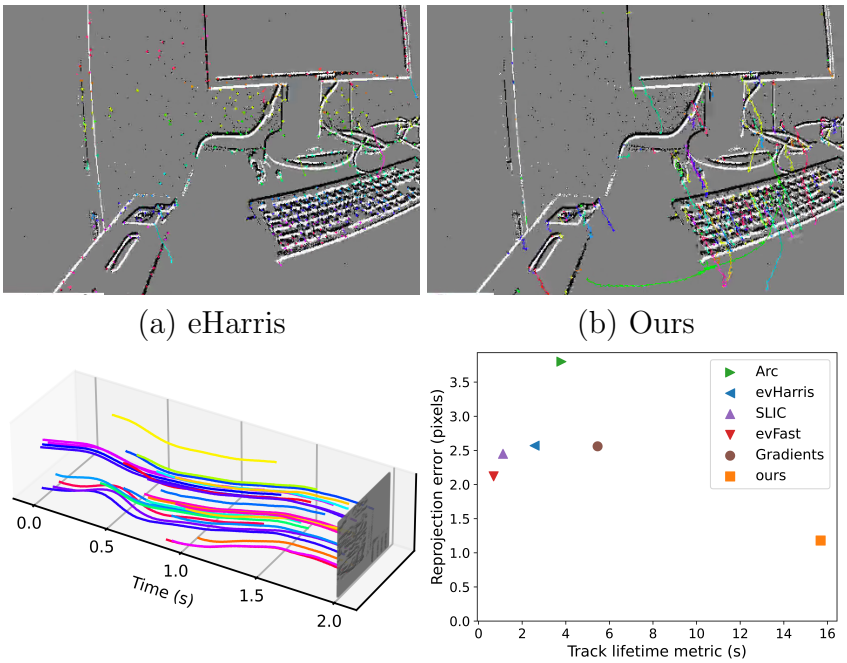


Figure 6.1: (a-b) Using our keypoint detection method on a real event streams of a 3D scene, we obtain tracks that are significantly longer than with previous methods. (c) Our tracks for the beginning of the Guernica sequence from the HVGA ATIS Corner dataset [73]. (d) We compare our method to Arc [4], eHarris [119], SILC [73], eFast [77], and the Gradients-based method seen in Chapter 5. Our detector predicts well-localized keypoints that can be tracked with a simple nearest-neighbor matching algorithm to obtain very long tracks. It nearly triples the track lifetime of the previous best method while reducing the reprojection error by more than 1 pixel.

This chapter will be sectioned as follow: we introduce the modifications to the keypoint detection pipeline in Section 6.1, we present our method in greater details in Section 6.2 and our experiments in Section 6.3. Section 6.4 is the section in which we present our conclusion regarding this novel approach.

6.1 Introduction

As shown in Figure 6.1, we propose a novel event-based keypoint detector that significantly outperforms previous methods, 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 [70]. 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 [50] 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 for previous methods. We link these detections using a simple nearest-neighbor matching procedure into tracks. This results in very long keypoint tracks, which are very important for many applications such as object tracking or Structure-from-Motion. This also provides very accurate locations.

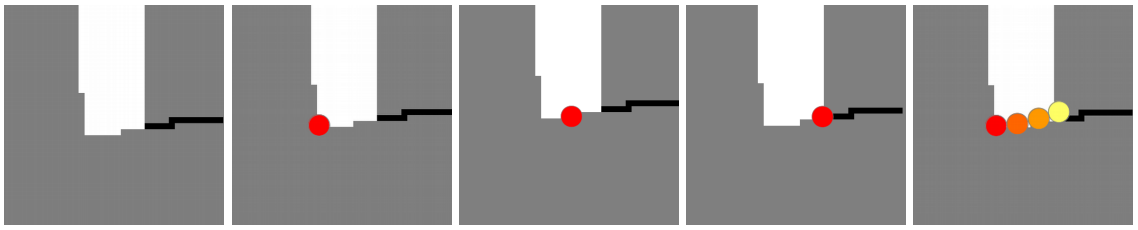
We train our architecture on synthetic planar scenes (since we use homographies), but it generalizes well on real data captured from 3D scenes, which we show by evaluating on both the HVGA ATIS Corner and The Event-Camera Dataset and Simulator real datasets [81] (cf. Figure 6.1). Generalization of keypoints trained on planar scenes to 3D scenes was observed before for bitmap frames, for example in [29], while generalization of networks trained on synthetic data and evaluated on real events was demonstrated in [92, 102].

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, as our second contribution, **we predict multiple successive locations rather for a single integration period**. This is illustrated in Figure 6.2: We predict a series of successive heatmaps for the entire frame. The local maxima of the heatmaps give 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



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

Figure 6.2: **Different keypoint location prediction strategies. (a)** Buffer of accumulated events around a keypoint over a short period of time (20ms). The keypoint is 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.

applied to other event-based camera tasks 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.

6.2 Method

Figure 6.3 summarizes our inference pipeline: 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. We detail below how we construct this event cube in practice, the nature of the heatmaps, the architecture we use, and how we generate our training data.

6.2.1 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 [130] 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

$$E(x, y, t) = \sum_i p_i \max(0, 1 - |t - t_i^\star|) \,, \tag{6.1}$$

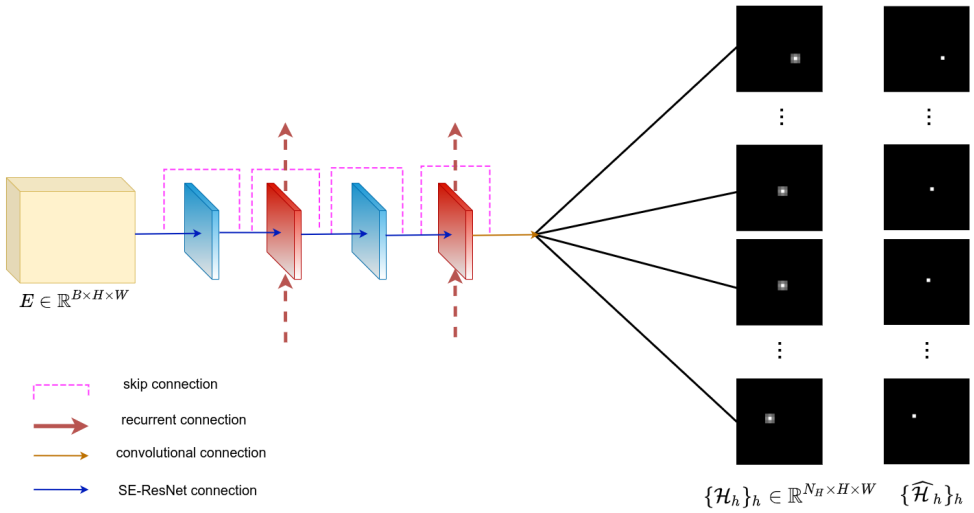


Figure 6.3: **Overview of our inference pipeline.** An event cube is built from the input events and fed to a recurrent neural network. We keep the architecture simple and efficient. We use the one introduced in Chapter 5. More precisely, the event cube is given as input to a Squeeze-and-Excite ResNet block, followed by a ConvLSTM with residual connection. This block of two layers is repeated once. Finally, a simple convolutional layer predicts the keypoints as a sequence of N_H heatmaps.

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:

$$t_i^\star = \frac{(t_i - t_{\min})}{\Delta T}(B-1)\,, \tag{6.2}$$

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

6.2.2 Predicting Heatmaps

From the event cube, our detector predicts a set of heatmaps. Heatmaps are convenient to predict the keypoints' locations as the number of keypoints 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

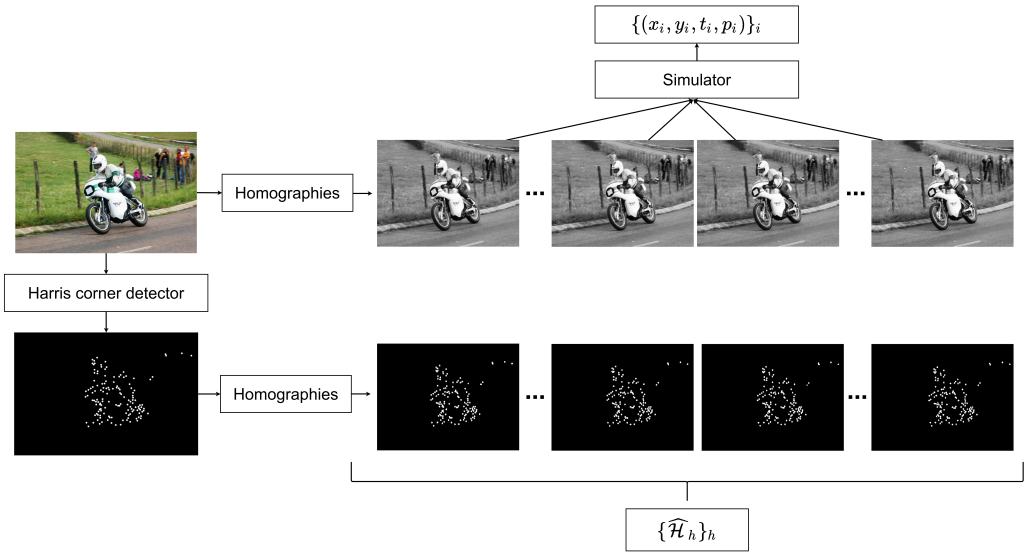


Figure 6.4: **Generating training data.** Similarly to Chapter 5, given a still image, we apply homographies to warp it and generate a video sequence. However, by contrast with Chapter 5, we use the same homographies to warp the heatmap of Harris' corner of the still image to have a perfect match between the video frames and the corners. From the video sequence we then generate events using a simulator and then compute the event cubes [130] used as input to our network.

and the labelled locations for the keypoints:

$$\mathcal{L} = \sum_{h \in [1; N_H]} \sum_{(x,y)} \text{BCE}(\mathcal{H}_h(x,y), \widehat{\mathcal{H}}_h(x,y)) \,. \tag{6.3}$$

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 layer of our architecture. A similar loss function was used in [42] for example. It encourages the local maxima with large values in the predicted heatmaps to correspond to the locations of the keypoints.

6.2.3 Creating Training Data

Our procedure to generate training data is illustrated in Figure 6.4. We generate synthetic videos by applying homographies to grayscale images, and convert them into event streams using a simulator. The keypoint labels are obtained using the Harris detector for grayscale images. We detail this procedure below.

From an image to a video. More precisely, to generate a synthetic video, we select an image from the COCO dataset [70] at random. We apply the Harris corner detector to this image to obtain a set of keypoints. Then, as shown in Figure 6.4, 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 Listing 5.1. 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 [90] and [27]. 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 6.3 shows the beneficial influence of warping the keypoints locations as opposed to detecting them independently.

6.2.4 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 $1e^{-4}$.

6.2.5 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 time delta, or a fixed number of 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 $\tau_{keypoint} = 0.3$ to be classified as a keypoint.

To obtain 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 (closest in space within the 7ms period), 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.

6.2.6 Architecture

We use a very shallow recurrent neural network to predict heatmaps $\{\mathcal{H}_h\}_h$ from an event cube. Thanks to this simplicity, inference is very fast. We rely on an architecture similar to the one presented in Chapter 5 with the exception that we predict multiple heatmaps instead of two gradient maps as it was previously done. This architecture is shown in Figure 6.3, and we describe it here for completeness.

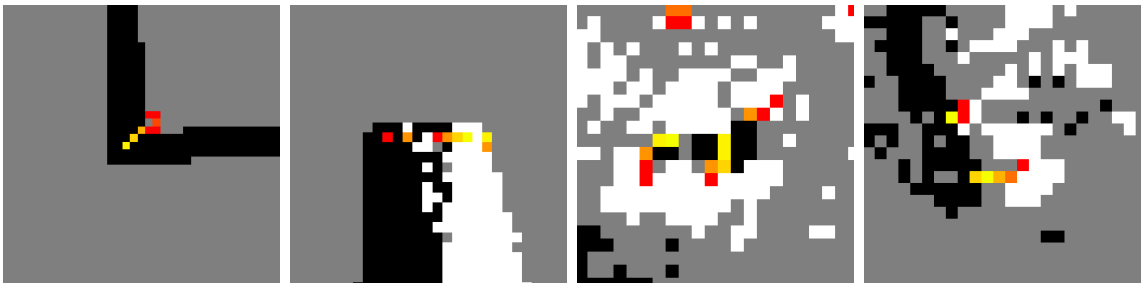


Figure 6.5: **Predicting keypoints.** When events are integrated over a long time window, edges become thick and accurate keypoint localization becomes challenging. We therefore predict the keypoint's spatial position for multiple time steps. The color gradient from yellow to red represents the time from past to present. Our network is able to predict precise keypoint locations at every time step. We then link these keypoint locations into trajectories using a simple nearest neighbor tracker. (Best seen in colour).

Also note that in Chapter 5 we still needed to compute the Harris score from the predicted gradients, while in this Chapter 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 [51]. 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 [55].

6.3 Experiments

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

6.3.1 Baseline, Datasets and Metrics

We used eHarris [119] as baseline for our experiments. Their approach has proven to be very accurate and very easy to implement. The combination of the efficiency and ease of execution make it the perfect baseline. We demonstrate with our experiments that our approach surpasses the baseline by a large margin.
We first consider the HVGA ATIS Corner dataset [73], which was created to evalu-

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 6.1: **Influence of N_H, the number of predicted heatmaps.** $N_H = 1$ corresponds to the standard approach with a single prediction for the integration period. While $N_H = 1$ already performs well thanks to our training procedure, increasing N_H consistently improves accuracy. Here, we use $\Delta T = 5$ms.

ate event-based keypoint detectors. It consists of seven real sequences with varying degrees of texture from a standard checkerboard to a complex natural image. These sequences were captured using an event camera (ATIS sensor) with a resolution of 480×360 pixels. It only contains recordings of planar patterns.

Following the evaluation protocol of [73], we consider the following metrics:

- the δt-homography reprojection error $\text{er}_{\delta t}$, which depends on a time parameter δt:

$$\text{er}_{\delta t} = \frac{1}{N_K} \sum_t \sum_k \| W_t^{t+\delta t}(K_{t,k}) - K_{t+\delta t,k} \| \,, \tag{6.4}$$

  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.

We also evaluated our detector on The Event-Camera Dataset and Simulator [81]. 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 [43] for evaluation: We run the event-to-video model from [92]

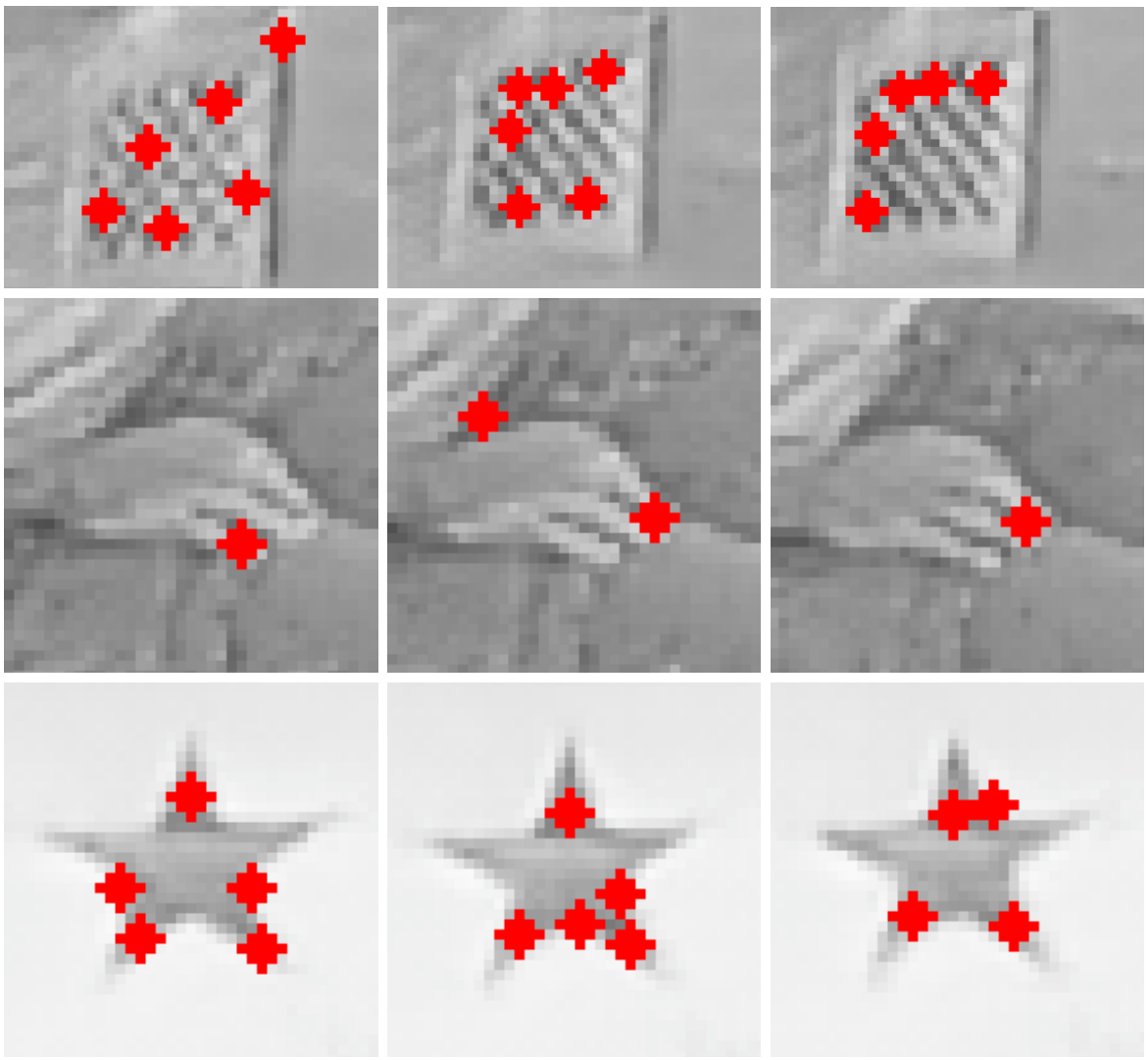


Figure 6.6: **Ground truth inconsistencies for the Event-Camera Dataset [81].** Ground truth keypoint locations for this dataset are sometimes quite inconsistent, making the comparison of methods on this data challenging. (Best seen in colour)

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	-	1.92	2.28	3.32	3.81	10.8

Table 6.2: **Influence of ΔT, the length of integration period.** We use $\Delta T =$ 5ms in all the other experiments.

to recreate frames from events and extract Harris corners on said frames to create groundtruth. However, as shown in Figure 6.6, the groundtruth is inaccurate and unstable over time. Another flaw of this evaluation criterion is a positive bias towards keypoints detectors relying 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 ($P_{\text{rel}} = P/P_{\text{Harris}}$ and $R_{\text{rel}} = R/R_{\text{Harris}}$). As in [43] results are computed on the sequences `boxes_6dof`, `dynamic_6dof`, `poster_6dof` and `shapes_6dof`.

6.3.2 Ablation Study

Predicting a single heatmap vs multiple heatmaps (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 6.1 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 6.2 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

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 6.3: **Evaluation of two strategies for generating training data: detecting or warping keypoints.** 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.

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.

Influence of the strategy for generating training data. In Table 6.3, we evaluate the strategy to generate keypoint labels described at the end of Section 6.2.3, where we detect a first set of keypoints in the initial grayscale image using Harris and warp their locations in the other images using the homographies already used to generate the video frames. It is compared with a more commonly approach where the keypoints are redetected in each frame independently. Generating training labels by warping results in a significantly more stable detector.

6.3.3 Comparison to the State-of-the-Art

We compare ourselves to [4, 43, 73, 77, 119] and our previously mentioned approach using image gradients, on two real event-based datasets.

Table 6.4 reports the comparison results, using the same tracking algorithm for all methods, 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. When comparing this novel approach to the one presented in Chapter 5, predicting image gradients first followed by a corner score computation, we can clearly see the benefit of predicting keypoints directly. Removing the intermediate representation improves accuracy as well as temporal coherence (measured with the track lifetime metric). Moreover, Figure 6.7 provides visual comparisons of the track lengths for these methods.

Table 6.5 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

Method	δt-reprojection Error (pixels)					Track Lifetime (s)
	$\delta t = 25$	50	100	150	200	
eHarris [119]	2.57	3.46	4.58	5.37	6.06	0.74
eFast [77]	2.12	2.63	3.18	3.57	3.82	0.69
Arc [4]	3.80	5.31	7.22	8.48	9.49	0.91
SILC [73]	2.45	3.02	3.68	4.13	4.42	1.12
gradients 5	2.46	4.26	7.25	11.98	8.89	5.46
ours	**1.18**	**1.28**	**1.45**	**1.63**	**1.84**	**15.7**

Table 6.4: **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. We nearly reduce by half the reprojection error while at the same time triple the tracks lifetime.

Method	P_{rel}	R_{rel}
eHarris [119]	1	1
eFast [77]	0.68	0.55
Arc [4]	0.84	0.59
luvHarris [43]	0.86	0.97
ours	**1.03**	**2.17**

Table 6.5: **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 [43], we report P_{rel} and R_{rel}, the precision and recall *relative* to eHarris ($P_{\text{rel}} = P/P_{\text{Harris}}$ and $R_{\text{rel}} = R/R_{\text{Harris}}$).

different sensor.

6.3.4 Qualitative results

Figure 6.5 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.
Similarly, Figure 6.7 demonstrate the homogeneous spatial distribution of our key-

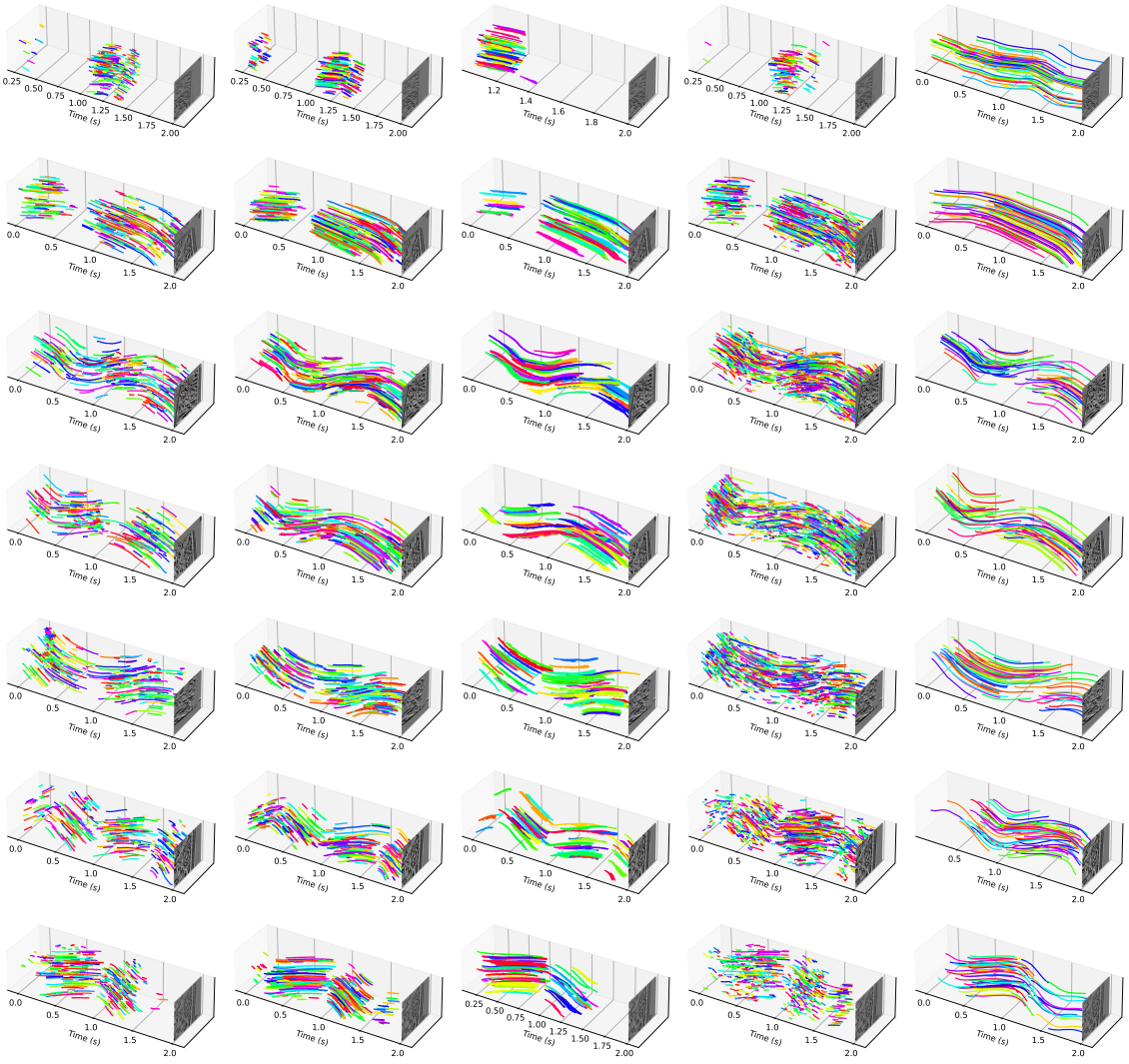


Figure 6.7: **Visual comparisons of tracks obtained by different methods on the HVGA ATIS Corner Dataset.** The tracks obtained with our detector are significantly longer.

points 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.

6.3.5 Computation Times and Memory Footprint

The network used in Chapter 5 has only been changed to ouptut more than two channels. The small number of parameters (27.5k) and minimal runtime (7ms on a GTX 1080 GPU for a HVGA input event sensor) make it the ideal candidate for real-time applications. The overall approach is very lightweight: The event cute have a low footprint, the network size is minimal and no post processing is needed for keypoint detection other than a non-maximal suppression. 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 if needed.

6.4 Conclusion

We have explained in this Chapter how we train a neural network to predict keypoints directly from an event cube without using an intermediate representation, such as the image gradients. Bypassing the intermediate representation speeds up the process as computing the Harris score becomes unnecessary. The prediction of a volume, through the prediction of multiple heatmaps, instead of predicting keypoints for a single timestamp, is another way to decrease computational needs while at the same time improving the accuracy of the keypoints. We manage to reach an accuracy in the range of the pixel for the reprojection error which is very precise and will enable future applications such as SLAM.

Chapter 7

Conclusion

To conclude our work we will discuss in this chapter the overall contributions of our work in Section 7.1, the impact it had in Section 7.2 and the potential future work directions in Section 7.3.

7.1 Contributions

The main objective of our work was to create a keypoint detection algorithm for event based sensors as computationally efficient as possible. The perfect keypoint detector is fast yet robust. Our first work used gradient prediction as a intermediate step between events and keypoints. In our second work we have removed this intermediate step and predicted keypoints directly. We have also introduced multiple predictions in time for keypoints, improving significantly the precision of keypoints in space and lowering the frequency of inference needed for a similar precision.

7.1.1 Detecting Stable Keypoints from Events through Image Gradient Prediction

In this paper the main contribution is to use the image gradients as a bridge between events and keypoints. The network used is very lightweight and enables very fast gradient predictions. The computation of the Harris score efficiently gives the classification of events as keypoints. The network was trained using a simulator to benefit from pixel perfect alignment and robustness to noise. The approach became the novel state-of-the-art in the field both in terms of keypoint precision and keypoints robustness (stability through time).

7.1.2 Long-Lived Accurate Keypoints in Event Streams

In this work we were able to bypass the intermediate representation of the image gradients to directly predict keypoints locations as heatmaps. We improved the robustness of the predictions and the track lifetime by reprojecting the keypoints during the generation of the ground truth. The good quality of the ground truth also enabled us to use hard negative mining during training helping with the imbalance between positive and negative classification of keypoints. The accuracy of the prediction was greatly improved by predicting multiple heatmaps instead of a unique one. This also made it possible to augment the time between inference and reduce the computational needs of the method.

We have released the code needed to train and validate the keypoint detection algorithm for event cameras here `https://www.prophesee.ai/metavision-intelligence/`. The code can be downloaded along with trained models.

7.2 Impact

The use of gradient as an in-between representation between events and images can be found in [52]. The authors compute the gradients of images and collect polarities pixelwise to produce a brightness increment image. They then proceed to compare both representations as part of their Direct Sparse Odometry pipeline.
The code has been publicly released for our the state-of-the-art keypoint detection algorithm and is now available to everyone. It will also be used internally for multiple projects at Prophesee.

7.3 Future work

Our work has been focused on keypoint detection i.e. classification of pixels as keypoints. The tracking has always been done with a simple nearest neighbor algorithm proving the robustness of our method and taking advantage of the very high temporal resolution of event sensors.

This approach has however some limitations, especially when considering loop closing in a SLAM or odometry setting. It is also an issue when trying to do some matching or relocalisation. This fallout can be avoided by computing some descriptors per pixel as is done in many other papers such as [29] (illustrated in Figure 7.1). The easiest way to compute descriptors would be to add a descriptor head to our network and introduce a descriptor loss. We could use the known homographies to train our descriptor head as was done in [29] using a hinge loss. The learning would also need to be balanced between detecting and describing keypoints. This addition would increase the computational needs of the method but would enable many other applications for the keypoint detection algorithm. This would also help to make the algorithm even more robust as we could use the descriptors to link feature tracks together instead of creating a novel track when some discontinuity or discrepancies arise.

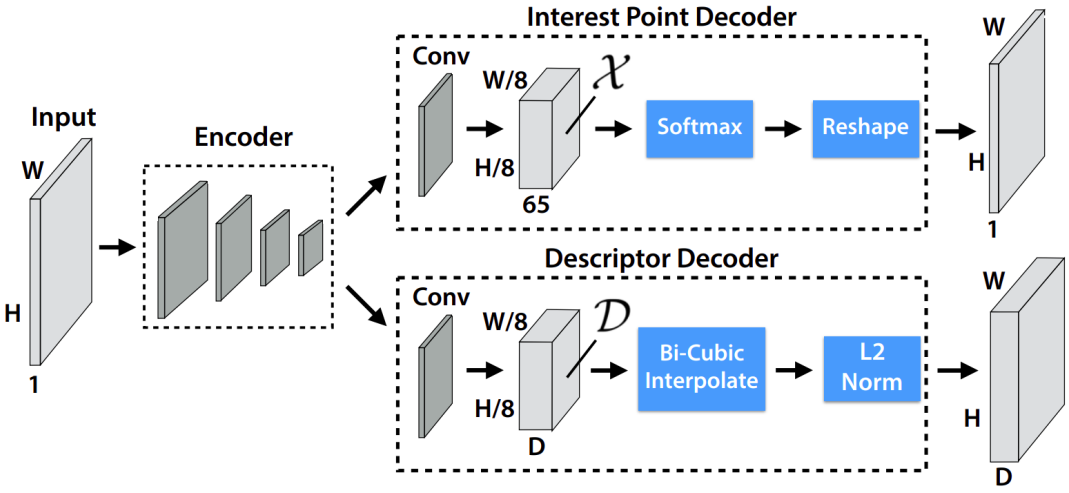


Figure 7.1: A simple way to add descriptors to a neural network is to add a description head in addition to the head for classification. Figure courtesy of [29]

Bibliography

[1] Prophesee evaluation kits. https://www.prophesee.ai/event-based-evk/

[2] Insightness event-based sensor modules. http://www.insightness.com/technology/ (2020)

[3] Agarwal, S., Snavely, N., Simon, I., Seitz, S.M., Szeliski, R.: Building rome in a day. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 72–79 (2009). https://doi.org/10.1109/ICCV.2009.5459148

[4] 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)

[5] Alzugaray, I.: Event-driven Feature Detection and Tracking for Visual SLAM. Ph.D. thesis, ETH Zurich, Zurich (2022). https://doi.org/10.3929/ethz-b-000541700

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

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

[8] Arnaud, É., Delponte, E., Odone, F., Verri, A.: Trains of keypoints for 3d object recognition. In: International Conference on Pattern Recognition. Honk Kong, Hong Kong SAR China (2006), https://hal.inria.fr/inria-00306707

[9] Baldwin, R.W., Almatrafi, M., Kaufman, J.R., Asari, V., Hirakawa, K.: Inceptive event time-surfaces for object classification using neuromorphic cameras. In: Karray, F., Campilho, A., Yu, A. (eds.) Image Analysis and Recognition. Springer International Publishing (2019)

[10] Ballas, N., Yao, L., Pal, C., Courville, A.C.: Delving deeper into convolutional networks for learning video representations. In: Bengio, Y., LeCun, Y. (eds.) 4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings (2016), http://arxiv.org/abs/1511.06432

[11] Bardow, P., Davison, A.J., Leutenegger, S.: Simultaneous optical flow and intensity estimation from an event camera. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 884–892 (2016). https://doi.org/10.1109/CVPR.2016.102

[12] Barua, S., Miyatani, Y., Veeraraghavan, A.: Direct face detection and video reconstruction from event cameras. In: 2016 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 1–9 (2016). https://doi.org/10.1109/WACV.2016.7477561

[13] Benosman, R., Ieng, S.H., Clercq, C., Bartolozzi, C., Srinivasan, M.: Asynchronous frameless event-based optical flow. Neural networks : the official journal of the International Neural Network Society **27**, 32–7 (11 2011). https://doi.org/10.1016/j.neunet.2011.11.001

[14] Blum, H., Dietmüller, A., Milde, M.B., Conradt, J., Indiveri, G., Sandamirskaya, Y.: A neuromorphic controller for a robotic vehicle equipped with a dynamic vision sensor. In: Robotics: Science and Systems (2017)

[15] Brandli, C., Berner, R., Yang, M., Liu, S.C., Delbruck, T.: A 240 × 180 130 db 3 µs latency global shutter spatiotemporal vision sensor. IEEE Journal of Solid-State Circuits **49**(10), 2333–2341 (2014). https://doi.org/10.1109/JSSC.2014.2342715

[16] Breiman, L.: Hinging hyperplanes for regression, classification, and function approximation. IEEE Trans. Inf. Theor. **39**(3), 999–1013 (may 1993). https://doi.org/10.1109/18.256506, `https://doi.org/10.1109/18.256506`

[17] Bretzner, L., Lindeberg, T.: Feature tracking with automatic selection of spatial scales. Computer Vision and Image Understanding **71**(3), 385–392 (1998). https://doi.org/https://doi.org/10.1006/cviu.1998.0650, `https://www.sciencedirect.com/science/article/pii/S1077314298906506`

[18] Cannici, M., Ciccone, M., Romanoni, A., Matteucci, M.: Asynchronous convolutional networks for object detection in neuromorphic cameras. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (2019)

[19] Chen, S., Guo, M.: Live demonstration: Celex-v: A 1m pixel multi-mode event-based sensor. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1682–1683 (2019). https://doi.org/10.1109/CVPRW.2019.00214

[20] 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)

[21] Cho, K., van Merrienboer, B., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder–decoder approaches. In: SSST@EMNLP (2014)

[22] Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS 2014 Workshop on Deep Learning, December 2014 (2014)

[23] Cohen, G.K.: Event-Based Feature Detection, Recognition and Classification. Theses, Université Pierre et Marie Curie - Paris VI ; University of Western Sydney (Sep 2016), https://tel.archives-ouvertes.fr/tel-01426001

[24] Conradt, J., Cook, M., Berner, R., Lichtsteiner, P., Douglas, R., Delbruck, T.: A pencil balancing robot using a pair of aer dynamic vision sensors. In: 2009 IEEE International Symposium on Circuits and Systems. pp. 781–784 (2009). https://doi.org/10.1109/ISCAS.2009.5117867

[25] Cook, M., Gugelmann, L., Jug, F., Krautz, C., Steger, A.: Interacting maps for fast visual interpretation. In: The 2011 International Joint Conference on Neural Networks. pp. 770–776 (2011). https://doi.org/10.1109/IJCNN.2011.6033299

[26] Dalal, N., Triggs, B., Schmid, C.: Human Detection Using Oriented Histograms of Flow and Appearance. In: European Conference on Computer Vision (2006)

[27] 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)

[28] Delmerico, J., Cieslewski, T., Rebecq, H., Faessler, M., Scaramuzza, D.: Are we ready for autonomous drone racing? the uzh-fpv drone racing dataset. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 6713–6719 (2019). https://doi.org/10.1109/ICRA.2019.8793887

[29] 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)

[30] Dimitrova, R., Gehrig, M., Brescianini, D., Scaramuzza, D.: Towards low-latency high-bandwidth control of quadrotors using event cameras. pp. 4294–4300 (05 2020). https://doi.org/10.1109/ICRA40945.2020.9197530

[31] Duchenne, O., Laptev, I., Sivic, J., Bach, F., Ponce, J.: Automatic annotation of human actions in video. In: 2009 IEEE 12th International Conference on Computer Vision. pp. 1491–1498 (2009). https://doi.org/10.1109/ICCV.2009.5459279

[32] Dusmanu, M., Rocco, I., Pajdla, T., Pollefeys, M., Sivic, J., Torii, A., Sattler, T.: D2-Net: A Trainable CNN for Joint Detection and Description of Local Features. In: Conference on Computer Vision and Pattern Recognition (2019)

[33] Evangelidis, G.D., Psarakis, E.Z.: Parametric image alignment using enhanced correlation coefficient maximization. IEEE Transactions on Pattern Analysis and Machine Intelligence **30**, 1858–1865 (2008)

[34] Falanga, D., Kleber, K., Scaramuzza, D.: Dynamic obstacle avoidance for quadrotors with event cameras. Science Robotics **5**(40), eaaz9712 (2020). https://doi.org/10.1126/scirobotics.aaz9712, `https://www.science.org/doi/abs/10.1126/scirobotics.aaz9712`

[35] 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)

[36] Finateu, T., Niwa, A., Matolin, D., Tsuchimoto, K., Mascheroni, A., Reynaud, E., Mostafalu, P., Brady, F., Chotard, L., LeGoff, F., Takahashi, H., Wakabayashi, H., Oike, Y., Posch, C.: 5.10 A 1280chr(195)chr(151)720 Back-Illuminated Stacked Temporal Contrast Event-Based Vision Sensor with 4.86μm Pixels, 1.066GEPS Readout, Programmable Event-Rate Controller and Compressive Data-Formatting Pipeline. In: 2020 IEEE International Solid- State Circuits Conference - (ISSCC) (2020)

[37] 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)

[38] Gallego, G., Delbrück, T., Orchard, G., Bartolozzi, C., Taba, B., Censi, A., Leutenegger, S., Davison, A.J., Conradt, J., Daniilidis, K., Scaramuzza, D.: Event-based vision: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence **44**(1), 154–180 (2022). https://doi.org/10.1109/TPAMI.2020.3008413

[39] Gallego, G., Rebecq, H., Scaramuzza, D.: A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3867–3876 (2018). https://doi.org/10.1109/CVPR.2018.00407

[40] Gehrig, D., Gehrig, M., Hidalgo-Carrió, J., Scaramuzza, D.: Video to Events: Bringing Modern Computer Vision Closer to Event Cameras. In: arXiv Preprint (2019)

[41] Gehrig, D., Loquercio, A., Derpanis, K.G., Scaramuzza, D.: End-to-end learning of representations for asynchronous event-based data. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 5632–5642 (2019)

[42] 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)

[43] 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)

[44] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K. (eds.) Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc. (2014), https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf

[45] Graca, R., Delbruck, T.: Unraveling the Paradox of Intensity-Dependent DVS Pixel Noise. In: arXiv Preprint (Sep 2021)

[46] Guo, M., Huang, J., Chen, S.: Live demonstration: A 768 × 640 pixels 200meps dynamic vision sensor. In: 2017 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 1–1 (2017). https://doi.org/10.1109/ISCAS.2017.8050397

[47] Haddadi, S.J., Castelan, E.B.: Evaluation of monocular visual-inertial slam: Benchmark and experiment. In: 2019 7th International Conference on Robotics and Mechatronics (ICRoM). pp. 599–606 (2019). https://doi.org/10.1109/ICRoM48714.2019.9071825

[48] Hagenaars, J.J., Paredes-Vallés, F., Bohté, S.M., de Croon, G.C.H.E.: Evolved neuromorphic control for high speed divergence-based landings of mavs. IEEE Robotics and Automation Letters **5**(4), 6239–6246 (2020). https://doi.org/10.1109/LRA.2020.3012129

[49] Harris, C., Stephens, M.: A combined corner and edge detector (1988)

[50] Harris, C., Stephens, M.: A Combined Corner and Edge Detector. In: Alvey vision conference (1988)

[51] 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)

[52] Hidalgo-Carrió, J., Gallego, G., Scaramuzza, D.: Event-aided direct sparse odometry. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2022)

[53] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)

[54] Howard, A.G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., Adam, H.: Mobilenets: Efficient convolutional neural networks for mobile vision applications. ArXiv **abs/1704.04861** (2017)

[55] Hu, J., Shen, L., Sun, G.: Squeeze-and-Excitation Networks. In: Conference on Computer Vision and Pattern Recognition. pp. 7132–7141 (2018)

[56] Hu, Y., Liu, S.C., Delbrück, T.: v2e: From video frames to realistic dvs events. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) pp. 1312–1321 (2021)

[57] Hyunsurk Eric, R.: Industrial DVS design; key features and applications. `http://rpg.ifi.uzh.ch/docs/CVPR19workshop/CVPRW19_Eric_Ryu_Samsung.pdf` (2019)

[58] Isfahani, S.M.M., Wang, L., Ho, Y.S., jin Yoon, K.: Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 10073–10082 (2019)

[59] Jadon, A., Omama, M., Varshney, A., Ansari, M.S., Sharma, R.: Firenet: A specialized lightweight fire & smoke detection model for real-time iot applications. arXiv preprint arXiv:1905.11922 (2019)

[60] Joubert, D., Marcireau, A., Ralph, N.O., Jolley, A., van Schaik, A., Cohen, G.: Event camera simulator improvements via characterized parameters. Frontiers in Neuroscience **15** (2021)

[61] Kim, H., Handa, A., Benosman, R., Ieng, S.H., Davison, A.: Simultaneous mosaicing and tracking with an event camera. In: Proceedings of the British Machine Vision Conference. BMVA Press (2014)

[62] Kim, H., Leutenegger, S., Davison, A.J.: Real-time 3d reconstruction and 6-dof tracking with an event camera. In: ECCV (2016)

[63] Lagorce, X., Orchard, G., Galluppi, F., Shi, B.E., Benosman, R.B.: Hots: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (2017)

[64] Lai, W.S., Huang, J.B., Wang, O., Shechtman, E., Yumer, E., Yang, M.H.: Learning blind video temporal consistency. In: European Conference on Computer Vision (2018)

[65] LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object Recognition with Gradient-Based Learning. In: Shape, Contour and Grouping in Computer Vision. p. 319 (1999)

[66] Lee, J.H., Delbruck, T., Pfeiffer, M.: Training deep spiking neural networks using backpropagation. Frontiers in Neuroscience **10** (2016). https://doi.org/10.3389/fnins.2016.00508, https://www.frontiersin.org/articles/10.3389/fnins.2016.00508

[67] Li, Y., Chen, Y., Wang, N., Zhang, Z.: Scale-aware trident networks for object detection (2019)

[68] 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)

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

[70] 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)

[71] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S.E., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV (1). Lecture Notes in Computer Science, vol. 9905, pp. 21–37. Springer (2016)

[72] Lowe, D.G.: Distinctive Image Features from Scale-Invariant Keypoints. Int. J. Comput. Vision pp. 91–110 (2004)

[73] 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)

[74] Messikommer, N., Georgoulis, S., Gehrig, D., Tulyakov, S., Erbach, J., Bochicchio, A., Li, Y., Scaramuzza, D.: Multi-bracket high dynamic range imaging with event cameras. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) pp. 546–556 (2022)

[75] Mikolajczyk, K., Schmid, C.: Indexing based on scale invariant interest points. In: Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001. vol. 1, pp. 525–531 vol.1 (2001). https://doi.org/10.1109/ICCV.2001.937561

[76] Mohemmed, A., Schliebs, S., Kasabov, N.K.: Span: A neuron for precise-time spike pattern association. In: ICONIP (2011)

[77] Mueggler, E., Bartolozzi, C., Scaramuzza, D.: Fast Event-based Corner Detection. In: British Machine Vision Conference (2017)

[78] Mueggler, E., Bartolozzi, C., Scaramuzza, D.: Fast Event-Based Corner Detection. In: British Machine Vision Conference (2017)

[79] Mueggler, E., Baumli, N., Fontana, F., Scaramuzza, D.: Towards evasive maneuvers with quadrotors using dynamic vision sensors. In: 2015 European Conference on Mobile Robots (ECMR). pp. 1–8 (2015). https://doi.org/10.1109/ECMR.2015.7324048

[80] 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. The International Journal of Robotics Research **36**(2), 142–149 (2017). https://doi.org/10.1177/0278364917691115

[81] 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)

[82] Munda, G., Reinbacher, C., Pock, T.: Real-time intensity-image reconstruction for event cameras using manifold regularisation. International Journal of Computer Vision **126**(12), 1381–1393 (2018). https://doi.org/10.1007/s11263-018-1106-2

[83] Perot, E., De Tournemire, P., Nitti, D., Masci, J., Sironi, A.: Learning to Detect Objects with a 1 Megapixel Event Camera. In: Advances in Neural Information Processing Systems. pp. 16639–16652 (2020)

[84] Pini, S., Borghi, G., Vezzani, R.: Learn to see by events: Color frame synthesis from event and rgb cameras. In: VISIGRAPP (2020)

[85] Ponulak, F.: Resume-new supervised learning method for spiking neural networks (2005)

[86] Posch, C., Matolin, D., Wohlgenannt, R.: A qvga 143 db dynamic range frame-free pwm image sensor with lossless pixel-level video compression and time-domain cds. IEEE Journal of Solid-State Circuits **46**(1), 259–275 (2011). https://doi.org/10.1109/JSSC.2010.2085952

[87] Posch, C., Serrano-Gotarredona, T., Linares-Barranco, B., Delbruck, T.: Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output. Proceedings of the IEEE **102**(10), 1470–1484 (2014). https://doi.org/10.1109/JPROC.2014.2346153

[88] Rast, A.D., Galluppi, F., Jin, X., Furber, S.: The leaky integrate-and-fire neuron: A platform for synaptic model exploration on the spinnaker chip. In: The 2010 International Joint Conference on Neural Networks (IJCNN). pp. 1–8 (2010). https://doi.org/10.1109/IJCNN.2010.5596364

[89] Rebecq, H., Gehrig, D., Scaramuzza, D.: ESIM: an open event camera simulator. Conf. on Robotics Learning (CoRL) (Oct 2018)

[90] Rebecq, H., Gehrig, D., Scaramuzza, D.: ESIM: An Open Event Camera Simulator. In: Conference on Robot Learning. pp. 969–982 (2018)

[91] Rebecq, H., Horstschaefer, T., Gallego, G., Scaramuzza, D.: Evo: A geometric approach to event-based 6-dof parallel tracking and mapping in real time. IEEE Robotics and Automation Letters **2**(2), 593–600 (2017). https://doi.org/10.1109/LRA.2016.2645143

[92] 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)

[93] Rebecq, H., Ranftl, R., Koltun, V., Scaramuzza, D.: High Speed and High Dynamic Range Video with an Event Camera. IEEE Transactions on Pattern Analysis and Machine Intelligence (2019)

[94] Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 779–788 (2016). https://doi.org/10.1109/CVPR.2016.91

[95] Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **39**(6), 1137–1149 (2017). https://doi.org/10.1109/TPAMI.2016.2577031

[96] Revaud, J., De Souza, C., Humenberger, M., Weinzaepfel, P.: R2D2: Reliable and Repeatable Detector and Descriptor. In: Advances in Neural Information Processing Systems (2019)

[97] Riba, E., Mishkin, D., Ponsa, D., Rublee, E., Bradski, G.: Kornia: An Open Source Differentiable Computer Vision Library for PyTorch. In: Winter Conference on Applications of Computer Vision (2020)

[98] Rosten, E., Drummond, T.: Machine Learning for High-Speed Corner Detection. In: European Conference on Computer Vision. pp. 430–443 (2006)

[99] Sanket, N., Parameshwara, C.M., Singh, C., Kuruttukulam, A.V., Fermüller, C., Scaramuzza, D., Aloimonos, Y.: Evdodgenet: Deep dynamic obstacle dodging with event cameras. pp. 10651–10657 (05 2020). https://doi.org/10.1109/ICRA40945.2020.9196877

[100] Scheerlinck, C., Barnes, N., Mahony, R.: Asynchronous Spatial Image Convolutions for Event Cameras. IEEE Robotics and Automation Letters pp. 816–822 (2019)

[101] Scheerlinck, C., Barnes, N., Mahony, R.: Continuous-time intensity estimation using event cameras. In: Jawahar, C., Li, H., Mori, G., Schindler, K. (eds.) Computer Vision – ACCV 2018. pp. 308–324. Springer International Publishing, Cham (2019)

[102] 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)

[103] Shi, X., Chen, Z., Wang, H., Yeung, D., Wong, W., Woo, W.: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In: Advances in Neural Information Processing Systems. pp. 802–810 (2015)

[104] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1409.1556

[105] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: Bengio, Y., LeCun, Y. (eds.) 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings (2015), http://arxiv.org/abs/1409.1556

[106] Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., Benosman, R.: HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification. In: Conference on Computer Vision and Pattern Recognition (2018)

[107] Sironi, A., Brambilla, M., Bourdis, N., Lagorce, X., Benosman, R.B.: Hats: Histograms of averaged time surfaces for robust event-based object classification. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 1731–1740 (2018)

[108] S.MohammadMostafavi, I., Wang, L., Yoon, K.J.: Learning to reconstruct hdr images from events, with applications to depth and flow prediction. International Journal of Computer Vision pp. 1–21 (2021)

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

[110] Son, B., Suh, Y., Kim, S., Jung, H., Kim, J.S., Shin, C., Park, K., Lee, K., Park, J., Woo, J., Roh, Y., Lee, H., Wang, Y., Ovsiannikov, I., Ryu, H.: 4.1 a 640×480 dynamic vision sensor with a 9µm pixel and 300meps address-event representation. In: 2017 IEEE International Solid-State Circuits Conference (ISSCC). pp. 66–67 (2017). https://doi.org/10.1109/ISSCC.2017.7870263

[111] Sroba, L., Ravas, R., Grman, J.: The influence of subpixel corner detection to determine the camera displacement. Procedia Engineering **100**, 834–840 (2015). https://doi.org/https://doi.org/10.1016/j.proeng.2015.01.438, `https://www.sciencedirect.com/science/article/pii/S1877705815004658`, 25th DAAAM International Symposium on Intelligent Manufacturing and Automation, 2014

[112] Stagsted, R., Vitale, A., Binz, J., Renner, A., Larsen, L., Sandamirskaya, Y.: Towards neuromorphic control: A spiking neural network based pid controller for uav (07 2020). https://doi.org/10.15607/RSS.2020.XVI.074

[113] Suh, Y., Choi, S., Ito, M., Kim, J., Lee, Y., Seo, J., Jung, H., Yeo, D.H., Namgung, S., Bong, J., Yoo, S., Shin, S.H., Kwon, D., Kang, P., Kim, S., Na, H., Hwang, K., Shin, C., Kim, J.S., Park, P.K.J., Kim, J., Ryu, H., Park, Y.: A 1280×960 dynamic vision sensor with a 4.95-m pixel pitch and motion artifact minimization. In: 2020 IEEE International Symposium on Circuits and Systems (ISCAS). pp. 1–5 (2020). https://doi.org/10.1109/ISCAS45731.2020.9180436

[114] Sun, K., Zhao, Y., Jiang, B., Cheng, T., Xiao, B., Liu, D., Mu, Y., Wang, X., Liu, W., Wang, J.: High-resolution representations for labeling pixels and regions. CoRR **abs/1904.04514** (2019)

[115] Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Chaudhuri, K., Salakhutdinov, R. (eds.) Proceedings of the 36th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 97, pp. 6105–6114. PMLR (09–15 Jun 2019)

[116] Tian, Y., Balntas, V., Ng, T., Barroso-Laguna, A., Demiris, Y., Mikolajczyk, K.: D2d: Keypoint extraction with describe to detect approach. In: Computer Vision – ACCV 2020: 15th Asian Conference on Computer Vision, Kyoto, Japan, November 30 – December 4, 2020, Revised Selected Papers, Part III. p. 223–240. Springer-Verlag, Berlin, Heidelberg (2020)

[117] Tulyakov, S., Bochicchio, A., Gehrig, D., Georgoulis, S., Li, Y., Scaramuzza, D.: Time Lens++: Event-based frame interpolation with non-linear parametric flow and multi-scale fusion. IEEE Conference on Computer Vision and Pattern Recognition (2022)

[118] Tyszkiewicz, M., Fua, P., Trulls, E.: Disk: Learning local features with policy gradient. Advances in Neural Information Processing Systems **33** (2020)

[119] 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)

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

[121] Verdie, Y., Yi, K.M., Fua, P.V., Lepetit, V.: Tilde: A temporally invariant learned detector. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) pp. 5279–5288 (2015)

[122] Vitale, A., Renner, A., Nauer, C., Scaramuzza, D., Sandamirskaya, Y.: Event-driven vision and control for uavs on a neuromorphic chip. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 103–109 (2021). https://doi.org/10.1109/ICRA48506.2021.9560881

[123] Wang, Z.W., Jiang, W., He, K., Shi, B., Katsaggelos, A., Cossairt, O.: Event-driven video frame synthesis. In: 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW). pp. 4320–4329 (2019). https://doi.org/10.1109/ICCVW.2019.00532

[124] Xingjian, S.H.I., Chen, Z., Wang, H., Yeung, D.Y., Wong, W.K., Woo, W.c.: Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In: Advances in Neural Information Processing Systems. pp. 802–810 (2015)

[125] Yi, K.M., Trulls, E., Lepetit, V., Fua, P.: LIFT: Learned Invariant Feature Transform. In: European Conference on Computer Vision. pp. 467–483 (2016)

[126] Youssef, I., Mutlu, M., Bayat, B., Crespi, A., Hauser, S., Conradt, J., Bernardino, A., Ijspeert, A.: A neuro-inspired computational model for a visually guided robotic lamprey using frame and event based cameras. IEEE Robotics and Automation Letters **5**(2), 2395–2402 (2020). https://doi.org/10.1109/LRA.2020.2972839

[127] Youssef, I., Mutlu, M., Bayat, B., Crespi, A., Hauser, S., Conradt, J., Bernardino, A., Ijspeert, A.: A neuro-inspired computational model for a visually guided robotic lamprey using frame and event based cameras. IEEE Robotics and Automation Letters **5**(2), 2395–2402 (2020). https://doi.org/10.1109/LRA.2020.2972839

[128] Zhang, R., Isola, P., Efros, A.A., Shechtman, E., Wang, O.: The unreasonable effectiveness of deep features as a perceptual metric. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 586–595 (2018)

[129] Zhu, A., Yuan, L., Chaney, K., Daniilidis, K.: Unsupervised event-based learning of optical flow, depth, and egomotion. In: 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 989–997. IEEE Computer Society, Los Alamitos, CA, USA (jun 2019). https://doi.org/10.1109/CVPR.2019.00108, `https://doi.ieeecomputersociety.org/10.1109/CVPR.2019.00108`

[130] 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)