University of Zurich[UZH]

Department of Informatics

Event Cameras: from SLAM to High Speed Video

Dissertation
submitted to the
Faculty of Business, Economics and Informatics
of the University of Zurich

to obtain the degree of
Doktor der Wissenschaften, Dr. sc.
(corresponds to Doctor of Science, PhD)

presented by

Henri Rebecq
from Sallanches, France

approved in February 2020 at the request of
Prof. Dr. Davide Scaramuzza, advisor
Prof. Dr. Andrew Davison, examiner
Prof. Dr. Tobi Delbruck, examiner
Prof. Dr. Bernt Schiele, examiner

The Faculty of Business, Economics and Informatics of the University of Zurich hereby authorizes the printing of this dissertation, without indicating an opinion of the views expressed in the work.

Zurich, 12.02.2020

Chairman of the Doctoral Board: Prof. Dr. Thomas Fritz

To my family and Anne.

Acknowledgements

First of all, I would like to thank my supervisor Prof. Davide Scaramuzza for accepting me as a PhD student in his lab. Davide provided me with many exciting opportunities, while also giving me the freedom and resources to pursue my own ideas.
This thesis would not have been possible without the help, fruitful discussions, and fun distractions from my colleagues. I therefore wish to express my gratitude to all the current and past members, visitors, and students. I would particularly like to thank Davide Falanga, Zichao Zhang, Titus Cieslewski, Elias Mueggler, Matthias Faessler, Christian Forster, Antonio Loquercio, Daniel Gehrig, Mathias Gehrig, Elia Kaufmann, Philipp Foehn, Manasi Muglikar, Yunlong Song, Guillermo Gallego, Jeff Delmerico, Dario Brescianini, Javier Hidalgo Carrio, Dimche Kostadinov, Julien Kohler, Alessandro Simovic, Michael Gassner, Thomas Längle, Manuel Sutter, Cedric Scheerlinck, Timo Stoffregen, Kosta Derpanis, Francisco Javier Perez Grau, Yi Zhou, Bianca Sangiovanni, Ruben Gomez Ojeda, Ana Maqueda, and Tamar Tolcachier. I also had the pleasure to work with great students, namely Timo Horstschäfer, Toni Rosinol Vidal, Samuel Bryner, Raffael Theiler, Dan Jia, Bojana Nenezic, Jihwan Youn. I am very grateful to Vladlen Koltun for accepting me as an intern at Intel Labs, Munich, Germany, and René Ranftl for his profitable mentorship during my stay there. I would like to thank the agencies funding my research, namely the National Centre of Competence in Research (NCCR) Robotics, the Swiss National Science Foundation, and the DARPA FLA Program. I would also like to thank Qualcomm for their support through the Qualcomm Innovation Fellowship and Daniel Fontijne for helpful pointers. I would like to thank Luca Longinotti and Federico Corradi from iniVation for their fast and effective support with event-camera software and hardware, as well as Samsung – in particular Eric Ryu, Yotam Ater and Lior Weintraub – for providing event-based sensors for my research and support with the drivers. I would like to thank the Sensors Group at the Institute of Neuroinformatics for developing and providing event-based hardware as well as plenty of support for my research. I would like to thank Prof. Andrew Davison, Prof. Tobi Delbruck and Prof. Bernt Schiele for accepting to review my thesis and for their valuable feedback.
Last but not least, I am very grateful to Anne, my family, and my friends who supported me at all times.

Zurich, March 2020 H. R.

Abstract

Cameras are appealing sensors because they are small, passive, inexpensive and provide rich information about the environment. For this reason, most computer vision research of the past fifty years has been devoted to them. As a result, frame-based cameras have become today the *de facto* sensor of choice to understand the world around us, and have found applications in numerous domains, such as the navigation of mobile robots (*e.g.* automotive), augmented and virtual reality, scene understanding, *etc*.

Yet, conventional cameras suffer from a number of shortcomings that limit their scope of application. In particular, they struggle in challenging scenarios such as high-speed motions (because of motion blur) and high dynamic range scenes. In addition, they have high latency; for example, a camera operating at 30Hz has a blind time of 33ms between each frame, which can be too high for some applications.

This thesis investigates *event cameras* as an alternative sensor to overcome the limitations of conventional cameras. These neuromorphic vision sensors work in a completely different way: instead of providing a sequence of images (*i.e.*, frames) at a constant rate, event cameras transmit only information from those pixels that undergo a significant brightness *change*. These pixel-level brightness changes, called *events*, are timestamped with *micro*-second resolution and transmitted asynchronously at the time they occur. Event cameras convey only non-redundant information, and are thus able to capture very high-speed motions. In addition, event cameras achieve a dynamic range of more than 140 dB, compared to about 60 dB of conventional cameras, because each pixel is autonomous and operates at its own set-point. However, since the output of an event camera is fundamentally different from that of conventional cameras, new algorithms that can deal with the absence of direct brightness measurement, exploit the high temporal resolution of the sensor, and leverage the asynchronous nature of event data are required to unlock their potential.

This thesis presents algorithms to perform robust perception using event cameras. It focuses on exploring the possibilities that event cameras bring to some fundamental problems in computer vision, both for low level tasks such as motion estimation, or high level tasks (*e.g.* object detection). Amongst others, this thesis provides contributions to solving the simultaneous localization and mapping problem, *i.e.*, use an event camera to simultaneously build a 3D map of the environment and estimate its location with respect to that map. It also explores the possibility of using event cameras in combination with complementary sensors (*e.g.* a conventional camera or an inertial measurement unit) for state estimation, and also presents contributions in the use of event cameras for robotic applications, *e.g.* quadrotor flight. This thesis also presents novel ways of synthesizing useful "image-like" representations of event data, in the

form of *motion-compensated* event frames, and *motion-invariant* image reconstructions, which open the door to bringing the mainstream of computer vision to event cameras, and to perform transfer learning from frame-based data to event data. Finally, this thesis also introduces a simulator for event cameras, as well as novel event camera datasets. The following is a list of contributions of this thesis:

- An accurate, efficient and versatile event camera simulator, which is made available open source to the community.
- The introduction of the event-based multi-view stereo problem (3D reconstruction from a moving event camera), and a method to solve it.
- A method to perform 6 Degrees-of-Freedom simultaneous localization and mapping (SLAM) with an event camera in natural scenes, which works in real-time, even on computationally constrained devices.
- A method to perform visual-inertial odometry in high-speed and high dynamic range scenarios, leveraging the complementary advantages of an event camera, an inertial measurement unit, and an optional conventional camera.
- The application of this method to demonstrate the first closed-loop flight of a quadrotor using an event camera for state estimation, allowing the quadrotor to fly in challenging conditions, such as a dark room.
- A novel recurrent neural network to synthesize high quality videos solely based on event data, providing state of the art image quality, and its application to high frame rate and high dynamic range video synthesis.
- The application of this network to several downstream applications (*e.g.* object detection, visual odometry, monocular depth prediction, *etc.*), demonstrating that transfer learning can be achieved from frames to events.

List of Contributions

Journal Publications

- **Henri Rebecq**, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. "High Speed and High Dynamic Range Video with an Event Camera". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).
  Links: Appendix F, PDF, Video, Code

- Daniel Gehrig, **Henri Rebecq**, Guillermo Gallego, and Davide Scaramuzza. "EKLT: Asynchronous, Photometric Feature Tracking using Events and Frames". In: *International Journal of Computer Vision* (2019). DOI: 10.1007/s11263-019-01209-w
  Links: PDF, Video

- **Henri Rebecq**, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. "EMVS: Event-based Multi-View Stereo – 3D Reconstruction with an Event Camera". In: *International Journal of Computer Vision* (2018). DOI: 10.1007/s11263-017-1050-6
  Links: Appendix B, PDF, Video, Code

- Antoni Rosinol Vidal, **Henri Rebecq**, Timo Horstschaefer, Guillermo Gallego, and Davide Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: *IEEE Robotics and Automation Letters (RA-L)* (2018). DOI: 10.1109/LRA.2018.2793357
  Links: Appendix E, PDF, Video

- Elias Mueggler, Guillermo Gallego, **Henri Rebecq**, and Davide Scaramuzza. "Continuous-Time Visual-Inertial Odometry for Event Cameras". In: *International Journal of Robotics Research* (2018). DOI: 10.1109/tro.2018.2858287
  Links: PDF

- **Henri Rebecq**, Timo Horstschaefer, Guillermo Gallego, and Davide Scaramuzza. "EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-time". In: *IEEE Robotics and Automation Letters (RA-L)* (2017). DOI: 10.1109/LRA.2016.2645143
  Links: Appendix C, PDF, Video

- Elias Mueggler, **Henri Rebecq**, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. "The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM". In: *International Journal of Robotics Research* (2017). DOI: 10.1177/0278364917691115
  Links: PDF, Video, Dataset

- Guillermo Gallego, Jon E. A. Lund, Elias Mueggler, **Henri Rebecq**, Tobi Delbruck, and Davide Scaramuzza. "Event-based, 6-DOF Camera Tracking from Photometric Depth Maps". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017). DOI: 10.1109/TPAMI.2017.2769655
  Links: PDF, Video

Peer-Reviewed Conference Papers

- **Henri Rebecq**, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. "Events-to-Video: Bringing Modern Computer Vision to Event Cameras". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).
  Links: PDF, Video

- Cedric Scheerlinck, **Henri Rebecq**, Timo Stoffregen, Nick Barnes, Robert Mahony, and Davide Scaramuzza. "CED: Color Event Camera Dataset". In: *IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)* (2019).
  Links: PDF, Video, Dataset

- Samuel Bryner, Guillermo Gallego, **Henri Rebecq**, and Davide Scaramuzza. "Event-based, Direct Camera Tracking from a Photometric 3D Map using Nonlinear Optimization". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2019).
  Links: PDF, Video, Project Webpage and Datasets

- Jeffrey Delmerico, Titus Cieslewski, **Henri Rebecq**, Matthias Faessler, and Davide Scaramuzza. "Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2019).
  Links: PDF, Video, Project Webpage and Datasets

- **Henri Rebecq**, Daniel Gehrig and Davide Scaramuzza. "ESIM: an Open Event Camera Simulator". In: *Conference on Robot Learning (CoRL)* (2018).
  Links: Appendix A, PDF, Video

- Daniel Gehrig, **Henri Rebecq**, Guillermo Gallego, and Davide Scaramuzza. "Asynchronous, Photometric Feature Tracking using Events and Frames". In: *European Conference on Computer Vision (ECCV)* (2018). DOI: 10.1007/978-3-030-01258-8_46
  Links: PDF, Video, Evaluation Code

- Yi Zhou, Guillermo Gallego, **Henri Rebecq**, Laurent Kneip, Hongdong Li, and Davide Scaramuzza. "Semi-Dense 3D Reconstruction with a Stereo Event Camera". In: *European Conference on Computer Vision (ECCV)* (2018). DOI: 10.1007/978-3-030-01246-5_15
  Links: PDF, Video

- Guillermo Gallego, **Henri Rebecq**, and Davide Scaramuzza. "A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2018). DOI: 10.1109/CVPR.2018.00407
  Links: PDF, Video

- **Henri Rebecq**, Timo Horstschaefer and Davide Scaramuzza. "Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization". In: *British Machine Vision Conference (BMVC)* (2017).
  Links: Appendix D, PDF, Video

- **Henri Rebecq**, Guillermo Gallego and Davide Scaramuzza. "EMVS: Event-based Multi-View Stereo". In: *British Machine Vision Conference (BMVC)* (2016).
  Links: PDF, Video

- Zichao Zhang, **Henri Rebecq**, Christian Forster, and Davide Scaramuzza. "Benefit of Large Field-of-View Cameras for Visual Odometry". In: *IEEE International Conference on Robotics and Automation (ICRA)* (2016).
  Links: PDF, Video, Research page

Open-source Software

- ESIM: an open event camera simulator
- EMVS: 3D reconstruction with an event camera
- High speed and high dynamic range video with an event camera
- Omnidirectional camera model for Blender

Awards

- **RA-L Best Paper Award (Honorable Mention), 2018.**
- **Qualcomm Innovation Fellowship Europe ($40.000), 2018**
- **Misha Mahowald Prize for Neuromorphic Engineering ($3.000), 2017**
- **BMVC Best Industry Paper Award, 2016**

Patents

- H. Rebecq, G. Gallego, D. Scaramuzza, **Simultaneous Localization and Mapping with an Event Camera**, US 2019/0197715 A1 . Issued on January 3, 2018.
- H. Rebecq, T. Horstschaefer, D. Scaramuzza, **Visual-Inertial Odometry with an Event Camera**, EU 17189223.5 - 1906 . Filed on November 6, 2017.

Contents

1 Introduction

This thesis presents algorithms to process data from event cameras. Event cameras are bio-inspired sensors that pose a paradigm shift in the way they acquire visual information. In contrast to conventional cameras that capture images at a fixed rate, event cameras have independent, smart pixels that measure brightness *changes* asynchronously, transmitting them as a stream of "events" which encode the time, location, and sign of the brightness change. Event cameras possess numerous advantages over conventional cameras: very high dynamic range (140dB vs. 60dB of conventional cameras), high temporal resolution (in the order of microseconds), no motion blur, and low power consumption. Hence, they have a large potential for computer vision and robotics in extreme scenarios (such as high-speed and high dynamic range), which are challenging for conventional cameras. However, because event cameras work in a fundamentally different way from conventional cameras, novel methods are required to process their output and unlock their potential.

This thesis is split into four parts. First, recognizing that high quality data is key to progress, I introduce an efficient and accurate event camera simulator. Second, I present algorithms to perform 3D reconstruction and 6-Degrees-of-Freedom (6 DOF) Simultaneous Localization and Mapping (SLAM) with an event camera. Third, I consider leveraging the complementarity between event cameras and additional sensors – such as an inertial measurement unit (IMU) or a conventional camera – to perform visual odometry in high speed and high dynamic range scenarios. I also demonstrate the applicability of this method for a challenging robotic application, namely autonomous flight of a quadrotor in the dark. Finally, I propose a recurrent neural network to reconstruct a video from event data. The video reconstructions achieve a quality on a par with conventional cameras, while having high frame rate and dynamic range. I also show the effectiveness of these reconstructions as an intermediate event representation which is *motion-invariant* and *transferrable*, allowing to bring the mainstream of conventional computer vision to event data for both low- and high-level tasks.

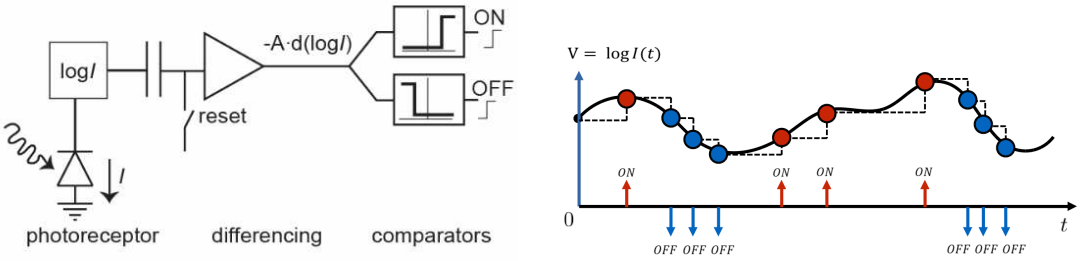
This thesis is structured in the form of a collection of papers. An introductory section that highlights the concepts and ideas behind the thesis is followed by self-contained publications in the appendix.

Section 1.1 presents the working principle, advantages and challenges of event cameras. Section 1.2 then states and motivates the research objectives of this dissertation, while placing them in the context of the related work. Chapter 2 summarizes the papers in the appendix and their connections with respect to each other. Finally, Chapter 3 provides future research directions.

1.1 Event Cameras for Computer Vision and Robotics

1.1.1 Principle of Operation

Event cameras have independent pixels that report local brightness changes asynchronously, at the time they occur. Fig. 1.1 provides a close look at a single pixel. The output of an event camera is a stream of events, with a new event triggered whenever the (logarithmic) brightness changes by a user-defined threshold (cf. Fig. 1.1b). To better highlight what happens across the entire sensor, the output of an event camera is compared to the one of a conventional camera in Fig. 1.2, and in a video[1]. Both cameras observe a rotating black disk. A conventional camera outputs a video, *i.e.* a sequence of frames at a fixed rate. By contrast, an event camera only reports the brightness changes.



(a) Analog Circuit of an Event Camera Pixel **(b)** Response to Intensity Profile

Figure 1.1 – How an event camera works. Figures adapted from [98].

1.1.2 Advantages

I now describe the main advantages of event cameras compared to conventional, frame-based cameras.

Low Latency and High Temporal Resolution. Event cameras measure brightness changes with a very high temporal resolution, and report them with very low latency (in the order of a few microseconds). The DVS [98] and DAVIS [18] have latencies of 15 µs and 3 µs, respectively. As comparison, the latency of conventional cameras is

[1] A video is also available at: https://youtu.be/LauQ6LWTkxM

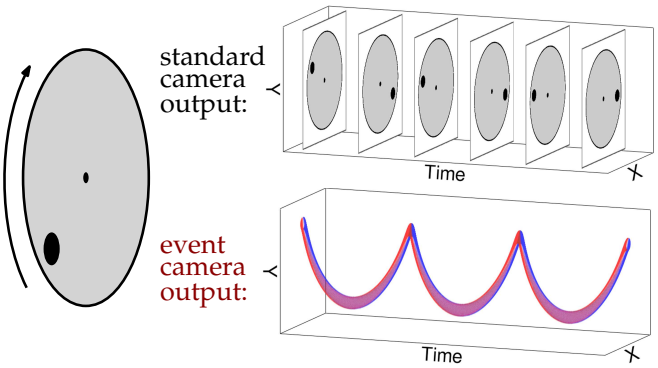


Figure 1.2 – Comparison of the output of a conventional camera and an event camera looking at a black disk on a rotating circle. While a conventional camera captures frames at a fixed rate, an event camera transmits the brightness changes continuously in the form of a spiral of events in space-time (red: positive events, blue: negative events). Figure inspired by [123].

bounded by their frame rate. For a frame-based camera operating at $f = 30\,\text{Hz}$, the temporal discretization can be as high as 33 ms [121] – four orders of magnitude higher as an event camera. Therefore, event cameras allow to perceive the world with much finer temporal detail, allowing for example to capture extremely fast motions (*e.g.* a bullet hitting an object), and also allow for faster control loops in robotic applications.

High Dynamic Range. Since each pixel of an event camera is independent (*i.e.* chooses its own set points), and responds to logarithmic changes of brightness, event cameras reach very high intra-scene dynamic ranges: the DVS [98], DAVIS [18], and ATIS [139] achieve dynamic ranges of 120 dB, 130 dB, and 143 dB, respectively. Conventional cameras typically achieve 60 dB, or more using different exposure time frames, at the expense of latency. Therefore, event cameras can still be used in scenes with very bright and very dark regions (see Fig. 1.3 for an extreme example).

Low Bandwidth. Because event cameras only report pixel-level brightness changes, no bandwidth is required if pixels do not change value, *i.e.*, if there is no relative motion between the scene and the camera and no illumination changes. Furthermore, the contrast sensitivity (relative brightness threshold above which an event is fired) can be adjusted for most event sensors. Setting the contrast threshold to a high value causes the events to mostly fire in salient regions (intensity edges), which is sufficient for some applications (*e.g.* 3D reconstruction [146]), thus reducing bandwidth even more.

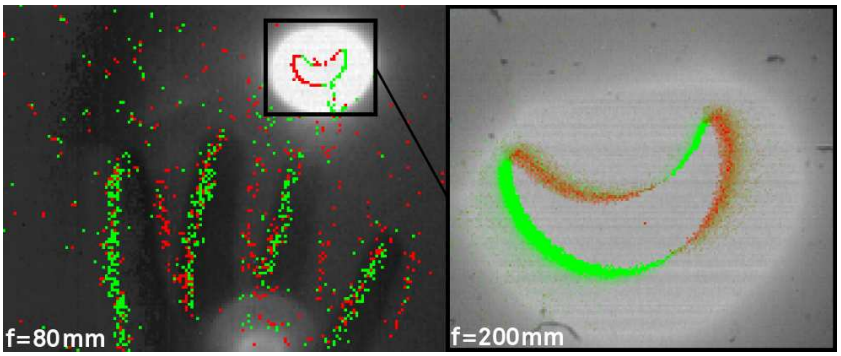


Figure 1.3 – Solar eclipse seen by a conventional and an event camera. While the image exhibits both overexposed (blooming) and underexposed areas, the events can detect the contours of the eclipse flawlessly. Image courtesy of Marc Osswald, Simeon Bamford and Tobi Delbruck.

Low Power. An event camera requires less power compared to a standard camera, because it only transmits non-redundant data, thus reducing the power necessary for acquisition and transmision [140]. However, the power consumption of the entire system (sensor and processing) must be considered. To make the most of the power savings that event cameras can bring, algorithms that can also operate with low power need to be designed. Since event cameras already compute temporal contrast, much computing power could potentially be saved.

Little Motion Blur. Conventional, global shutter cameras expose all their pixels simultaneously by collecting light during a given exposure time. This results in *motion blur* when a fast moving object is visible, or when the camera itself undergoes a fast motion. Motion blur reduces the amount of visual information available in an image, and can, it its most extreme form, even make it completely unexploitable. By contrast, all the pixels of an event camera are independent, removing the need for a global exposure time. Thus, event cameras are more robust to motion blur than conventional cameras.

1.1.3 Nonidealities

The last section provided a simplified view of the advantages of an ideal event camera compared to a conventional camera. These advantages need to be nuanced though, as event cameras possess some nonidealities as well.

Finite Bandwidth. Although event camera pixels are fast, they have a finite analog bandwidth [98], like any physical transducer. If the incoming light intensity varies too quickly, the front end photoreceptor circuits filter out the variations. More precisely, the pixel bandwidth of an event camera is proportional to the absolute light intensity

(when nothing else limits bandwidth). This means that the response to a step edge of intensity has a finite response time: a low contrast intensity step will result in a stream of events from the pixel that can extend over tens of milliseconds, as the event pixel photoreceptor changes its output to the final value after the step. Second, there is an event refractory period (during which the pixel is blind to change). Both of these affect the responses of the event camera pixel to small and fast moving objects, especially under lower lighting conditions, *e.g.* as quantified in [17].

Latency Considerations. As explained above, the output latency of an event camera depends on the absolute light intensity. Moreover, while most event cameras have digital timestamp of 1µs, the event timestamp jitter strongly depends on the absolute light intensity [98]. In addition, both USB transmission and processing threading also need to be taken into account. As a result, the latency at the computer can be much longer than 1µs at the computer [39, 98, 38].

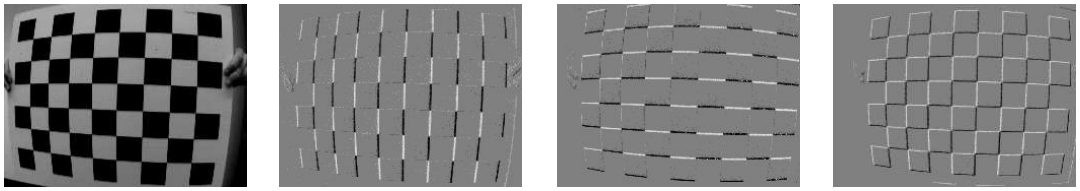
Coarse Brightness Quantization. Event cameras have a relatively coarse brightness quantization, of about 15% minimum temporal contrast. This may lead to practical limitations for scenes with low contrast and high speed. There are, however, prototype event sensors with much higher sensitivity. For example, [177] developed a sensor with a temporal contrast sensitivity of 1.5%.

Output Bus Saturation. When a scene is busy, the chip output bus may saturate, thus potentially dropping events. This is especially true as event camera sensor sizes keep increasing. For example, recent hardware such as the Samsung DVS [186] reports an output rate of up to 300Meps, a drastic increase from the 1Meps rate from an earlier sensor like the DVS128 [98].

1.1.4 Challenges

Despite their numerous advantages over conventional cameras, event cameras present a number of challenges that need to be overcome to unlock their full potential for computer vision and robotics. The main challenge arises from the nature of their output (a stream of asynchronous events that encode brightness *changes*), which differs fundamentally from that of a conventional camera (a synchronous sequence of frames that encode light intensity directly). In addition, contrarily to conventional sensors, whose bandwidth is constant, the event rate varies with the camera motion and scene texture. Thus, guaranteeing real-time performance is challenging – especially for algorithms which process the event stream in an *event-by-event* fashion, with constant computation time per event.

No Absolute Intensity – Data Association? Event cameras do not measure absolute brightness directly (with the exception of some sensors such as the ATIS [140]), but instead respond to brightness *changes* on the image plane. Hence, what events really measure is the *temporal derivative* of the brightness signal [64], which depends on the image gradient and the optical flow [10, 61]. As illustrated in Fig. 1.4, the same intensity pattern yields a different set of events depending on the relative motion between the scene and the camera. As a result, data association (*i.e.* matching corresponding sets of events) is challenging. Addressing this problem is one of the core endeavors of my thesis. I propose three different ways to address this problem: (*i*) by circumventing explicit data association (Papers B, C), (*ii*) by leveraging complementary sensors, such as an IMU or/and a conventional camera (Papers D, E), or finally (*iii*) by learning to synthesize a motion-invariant representation of event data (Paper F).



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

Figure 1.4 – Moving a checkerboard (a) in front of an event camera in different directions. (b)-(d) shows visualizations of the events obtained by binning events over a short time interval. Pixels that do not change intensity are represented in gray, whereas pixels that increased or decreased intensity are represented in bright and dark, respectively. Clearly, (b) (only vertical edges), (c) (only horizontal edges), and (d) can hardly be related to each other in the absence of underlying photometric information (a).

Asynchronous Output. Conventional cameras rely on an external clock to perform sampling, resulting in a series of synchronous measurements (frames); in contrast, event cameras perform level-crossing sampling, *i.e.*, based on brightness changes in the scene, which happen sparsely on the image plane. Hence, event cameras break the implicit assumption of most computer vision algorithms: the existence of frames.

Sensor Modelling and Noise. In the ideal event camera model, events fire as soon as the brightness change at a pixel exceeds a known threshold (user-defined). In reality, event cameras suffer from non-idealities due to shot noise in photons and transistor circuit noise. Event camera pixels also have a refractory period, which limits the rate at which events can be perceived and transmitted. Thus, the actual contrast thresholds depend on many factors, such as the brightness value and pixel location, temperature, and time (due to memory effects). While some simple models of event generation based on an ideal event camera with Gaussian noise have been proven useful for some tasks [80], they fail at capturing the full complexity of real event sensors [19].

Limited Amount of Event Data. The last few years have seen the formidable rise of machine learning techniques, and especially their incarnation in neural networks [86]. One crucial element in the success of these approaches for computer vision is *data*, in the form of large image datasets such as ImageNet [167], or large video databases such as YouTube [96], which have allowed to fuel neural network training while providing for a reliable way to benchmark progress in computer vision. Yet, because event cameras are emerging sensors, there is no such large database of event data – which slows down the progress of research. For this reason, I have largely contributed to recording and releasing a variety of novel event camera datasets during my thesis ([124, 172, 41], Section 2.1.2), to enable further research. In addition, an efficient and accurate event camera simulator is introduced in Paper A. Finally, the last part of this thesis (Paper F) examines the possibility of transferring knowledge learned from image data to the event domain – thus drastically reducing the demand for event data.

1.1.5 Sensor Fusion with Event Cameras

Event cameras excel at providing robust, low-latency measurements of brightness changes. Conversely, conventional cameras provide slow, but direct intensity measurements for every pixel. More generally, other sensors provide crucial complementary information (*e.g.* an inertial measurement unit (IMU) measures sensor acceleration and angular velocity, albeit with a large amount of noise). Computer vision systems may exploit this complementarity. In Section 2.3 of this thesis, I have specifically examined the task of camera pose estimation, considering an event camera, a conventional camera and an IMU, and strived to design algorithms capable of combining the specific advantages of each sensor, aiming at low-latency, robust and accurate pose estimates.

1.1.6 Event Sensors: Past, Present and Future

In the early 90's, Mischa Mahowald and Carver Mead were the first to develop sensors inspired by the human retina, which they called "silicon retinas" [112, 107, 106]. Their silicon retina mimicked the neural architecture of the eye, and ignited the emerging field of neuromorphic engineering. In 2008, the DVS [98] became the first commercially available event camera, developed in the context of the CAVIAR project [176], which aimed at building an event-based hardware system with sensing, processing, and learning using the Address-Event Representation (AER) communication framework. Since then, the commercial availability of the DVS as a prototyping and research platform has led to a large body of research on event cameras, allowing to tackle problems that are currently infeasible with conventional, frame-based image sensors. This has, in turn, attracted a lot of attention from the industry, resulting in the development of ever-improving event cameras, driven both by large sensor companies (such as Samsung) and startup companies (such as iniVation, Insightness,

CelePixel, and Prophesee). In addition to major improvements in event cameras, the last few years have seen the emergence of hybrid sensors that combine an event-based sensor with other, complementary sensors. The DAVIS [18], for example, can output synchronized events, grayscale frames and inertial measurements. In Section 2.3 of this thesis, I explore the benefits of sensor fusion with event cameras in the context of visual odometry. Table 1.1 summarizes the progress of event camera hardware over the last few years, showing the specifications of a few event cameras. While the first prototypes had low resolution (*e.g.* 128×128 for the DVS128 [98]), HD resolution sensors (1280×800) are now commercially available (*e.g.* CeleX-V). Event cameras are still expensive sensors (*e.g.* the CelePixel-V sensor costs $2500). However, with many industrial players in the field, event cameras are poised to enter mass production, which should result in a sharp decrease of price in the next few years. For example, Samsung recently released their first commercial product using a VGA event camera[2], which costs $130. The startup Prophesee also freshly announced the first industrial grade event camera[3]. There seems to be a big commercial interest in event cameras, particularly in the domains of industrial automation, mobile robotics, and augmented or virtual reality (AR/VR).

	DVS128 [98]	ATIS [139]	DAVIS240 [18]	DAVIS346 [33]	Gen3 CD [1]	DVS-Gen3 [169]	CeleX-V [25]	DVS-Gen4 [169]
Supplier	iniVation	Prophesee	iniVation	iniVation	Prophesee	Samsung	CelePixel	Samsung
Year	2008	2011	2014	2017	2017	2018	2018	2019
Resolution (pixels)	128×128	304×240	240×180	346×260	640×480	640×480	1280×800	1280×960
Dynamic range (dB)	120	143	120	120	> 120	> 90	> 120	> 90
Die power consumption (mW)	23	50 - 175	5 - 14	10 - 170	36 - 95	65	-	140
Camera Max. Bandwidth (Meps)	1	-	12	12	66	-	100	-
Pixel size (μm^2)	40×40	30×30	18.5×18.5	18.5×18.5	15×15	9×9	9.8×9.8	4.95×4.95
Grayscale output	no	yes	yes	yes	no	no	yes	no
IMU output	no	no	1 kHz	1 kHz	1 kHz	1 kHz	no	no

Table 1.1 – Evolution of event camera hardware in the last decade, including both commercial and research prototypes. In particular, the pixel size has drastically decreased, resulting in a large increase in sensor resolution. Note that this table is not exhaustive.

1.2 Related Work on Event Cameras

This section summarizes the state of the art in event-based vision. It also indicates how the work of this thesis relates to and improves on state-of-the-art methods. In addition, an excellent and comprehensive survey paper on event-based vision was recently published [57].

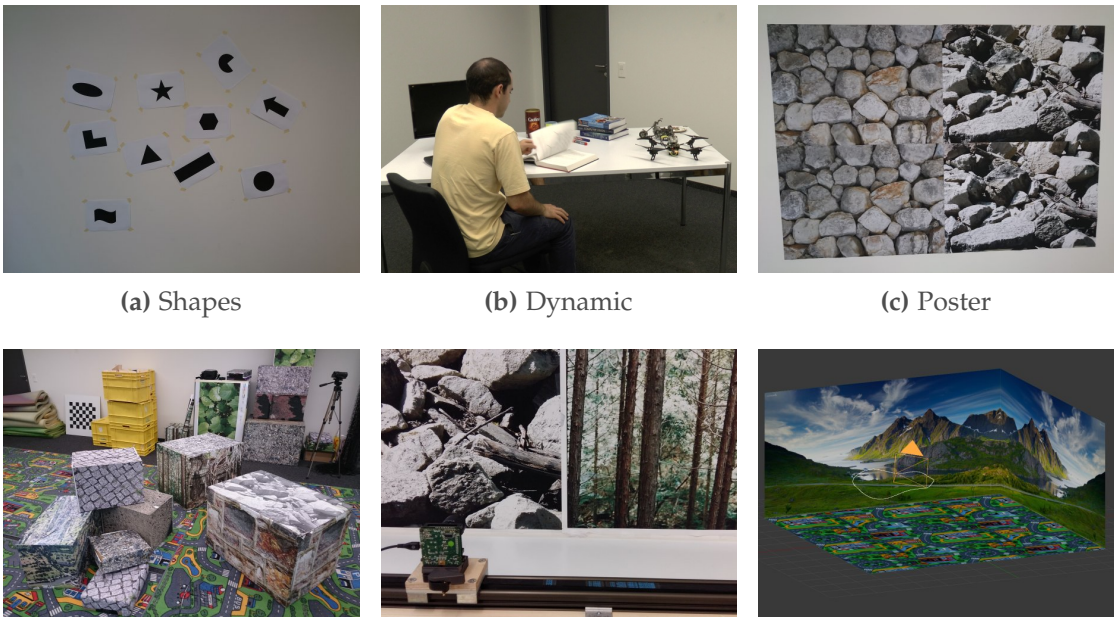
1.2.1 Event Camera Datasets and Simulators

Computer vision research has repeatedly benefitted from benchmark datasets and simulators, and event-based vision is no exception.

[2] https://www.samsung.com/se/smartthings/smartthings-vision-u999/
[3] https://www.prophesee.ai/2019/02/14/imago-prophesee/

Datasets. The range of tasks that have been addressed with event cameras can be roughly split into two: low-level vision tasks and high-level vision tasks. A number of event datasets has been specifically designed for low-level vision tasks such as visual (inertial) odometry [196, 124, 212], optic flow estimation [11, 166] or depth from stereo [202, 212]. The scope of the early datasets was severely restricted (for example, [11] contains a dozen sequences of a en event camera looking at a calibration pattern, each sequence during less than ten seconds.

In [124], a benchmark for event-based visual (inertial) odometry (Fig. 1.5) was introduced. It features 25 sequences recorded with a DAVIS240C sensor [18], and each sequence contains events, frames, and inertial measurements, as well as ground truth camera poses from a motion capture system. However, it does not contain ground truth depth or optic flow, and most of the scenes were captured indoors, targeting augmented and virtual reality (AR/VR) scenarios.



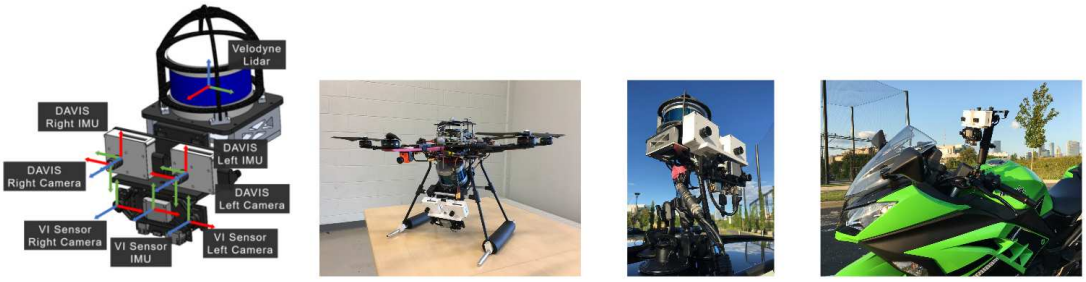
(a) Shapes **(b)** Dynamic **(c)** Poster

(d) Boxes **(e)** Linear Slider **(f)** Simulator

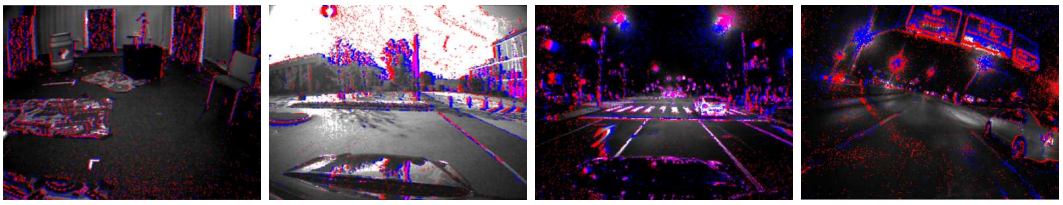
Figure 1.5 – Example Scenes from the Event Camera Dataset [124].

Shortly after, the MVSEC dataset [212] was introduced (Fig. 1.6). It contains 10 sequences from a stereo DAVIS346 sensor[4], which was mounted on various vehicles (car, quadcopter and motorbike). Images from a standard stereo camera setup, ground truth camera poses, depth maps (derived from a LIDAR) and optic flow maps [213] are also provided. However, the dataset essentially targets visual navigation scenarios for automotive, and features only a few sequences with fast motions (*i.e.* large optical flows).

[4] https://inivation.com/wp-content/uploads/2019/07/2019-07-09-DVS-Specifications.pdf



(a) Recording setup.



(b) Sample sequences (frames with events overlayed).

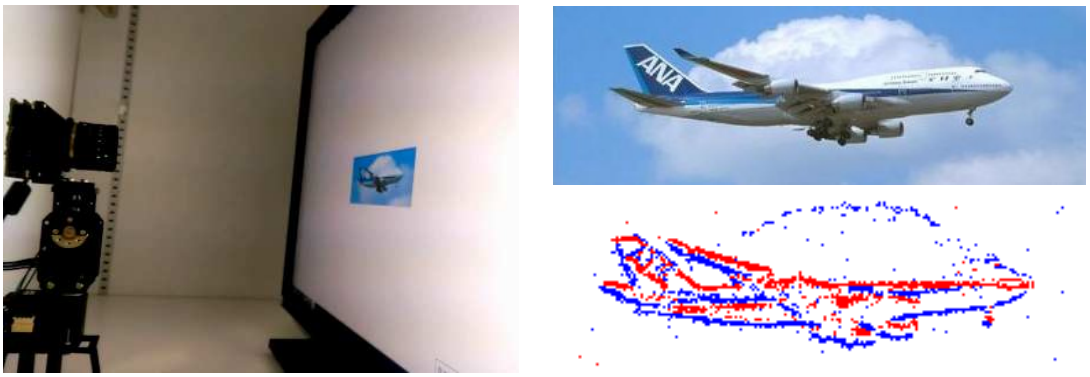
Figure 1.6 – The MVSEC dataset [212].

To push the limits in terms of speed, I contributed to recording a novel event dataset with a DAVIS346 camera placed onboard a drone racing quadcopter [41] flown at high speeds (up to 25m/s), both in indoor and outdoor environments (Section 2.1.2).

A number of datasets have also been released for high level tasks, most of which address object recognition [132, 178, 184], or gesture recognition [7]. In [132], an approach to convert frame-based datasets to event-based representations was presented, and used to generate semi synthetic, event-based versions of the MNIST [90] and Caltech101 datasets[5]. To achieve this, they mounted an event camera on a pan-tilt motor, facing a screen onto which images were projected, and recorded the events generated by nudging the sensor in different directions with the motor (Fig. 1.7). Finally, the recent DDD17 [15] and N-CARS [184] datasets target automotive applications. While N-CARS [184] specifically focuses on car recognition, DDD17 [15] provides the vehicle speed, GPS position, driver steering, throttle, and brake captured from the car's on-board diagnostics interface. However, it does not contain semantic labels, nor ground truth data for low-level vision tasks such as depth maps or optical flow.

All the datasets previously presented are limited to monochrome events. Yet, color information is known to be an important source of visual information [191], which has the potential to improve performance on many vision tasks (*e.g.* segmentation [110]). Taking advantage of the recent development of color event cameras, I contributed to collecting and releasing the first color event camera dataset (Section 2.1.2).

[5]These datasets can be found at: http://www.garrickorchard.com/datasets



(a) Recording setup. **(b)** Caltech101 frame (top) and events (bottom).

Figure 1.7 – The N-Caltech101 [132] dataset. Semi-synthetic events were obtained by moving an event camera looking at an image projected on a screen.

Simulators. There are only a few previous simulators for event cameras known to the author. A simple event simulator based on the Gazebo robot simulation framework [83] was introduced in [78]. It operates by thresholding the difference of the two latest images on a limited frame rate of approximately 60 Hz, and is therefore unable to provide precise timestamps of the events, which is crucial for most applications. Furthermore, neither are events generated for smooth gradients nor are several events generated for strong gradients that cross the threshold multiple times. A sensor-in-the-loop simulation framework is presented in [123], where a known video is shown to a calibrated event camera on a computer screen. This has the advantage that realistic sensor noise is contained in the data, but creates artifacts in the event data due to the limited display rate of the screen, which is much slower than the temporal resolution of the event camera [132]. I thus developed the first event camera simulator [124] that implements the true operating principle of event cameras[6] (*i.e.* providing accurate and independent timestamps for every simulated event). To simulate an event camera pixel, my simulator relied on rendering images from a 3D scene at a very high frame rate using Blender[7], and interpolated linearly the visual signal at each pixel using the two closest rendered intensities. However, because it relied on rendering frames at a high rate to approximate the visual signal, it was very inefficient, and failed when the brightness signal varied more rapidly than the arbitrarily chosen rendering frame rate.

To address these shortcomings, I proposed a novel adaptive sampling scheme to simulate event data accurately and efficiently, by only rendering new frames when needed (Paper A). Additionally, I released an efficient event camera simulator implemented in C++, which can simulate large amounts of accurate event data, while providing ground truth camera poses, depth maps, optical flow maps, and other sensing modalities, *e.g.* a frame-based sensor or an inertial measurement unit.

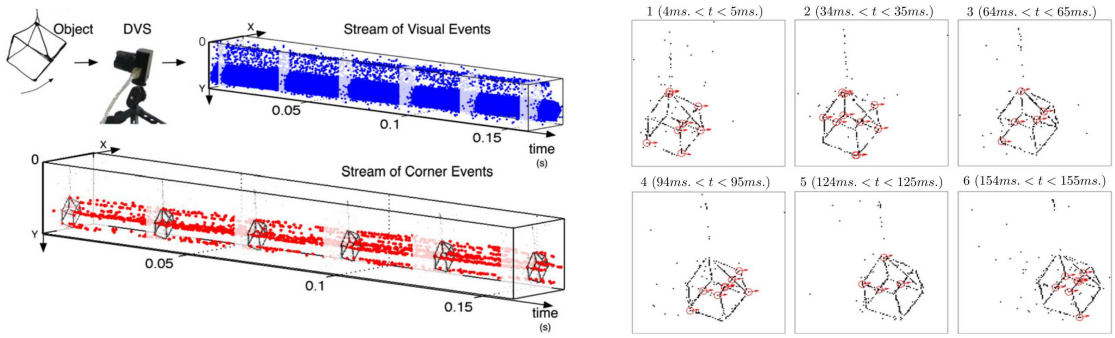
[6] Available at: https://github.com/uzh-rpg/rpg_davis_simulator
[7] https://www.blender.org

1.2.2 Feature Detection and Tracking, Optical Flow

Feature detection and tracking on the image plane are fundamental building blocks of many vision tasks such as visual odometry, object segmentation and scene understanding. Event cameras have the potential to enable low latency, asynchronous feature tracking, thus this topic has been widely studied.

The first event-based feature tracking methods were very simple and focused on demonstrating the low-latency and low-processing advantages of event-driven vision systems. Hence, they assumed the camera to be static and tracked moving objects by finding clusters of moving events [100, 101, 100, 39, 45, 38, 137], or assuming known object shapes, such as circles [131] or lines [29].

Later methods reduced the necessity for *a priori* knowledge or user input to perform tracking, and instead extract keypoints, or "corners" to track, directly from the event stream. For example, [27] computed corners as the intersection of two moving edges obtained by fitting planes in the space-time stream of events (Fig. 1.8), and extensions of frame-based keypoint detectors, such as Harris [69] and FAST [165], have been developed for event cameras [193, 120, 6]. Finally, methods based on machine learning have also emerged [108]. These methods, however, have mostly been demonstrated on artificial scenes with a large amount of texture and strong edges, making detection and tracking easier.



(a) Stream of events in response to a swinging 3D cube (top), and detected corner events (bottom)

(b) Example corner events detected in the event stream.

Figure 1.8 – Event-based corner tracking [27].

To address this problem and allow for feature detection and tracking in natural scenes, distinctive features need to be extracted from the event stream. As shown in [190, 87], such features can be obtained using small patches of intensity gradients (*i.e.*, edges) extracted from frames. Extending this idea, [210] used an expectation-maximization scheme (EM) to estimate an "edge template" (*i.e.* a point set) from events only, to which new events are registered. While this approach allows to track features in natural scenes without prior knowledge nor conventional frames, the algorithm relies on an

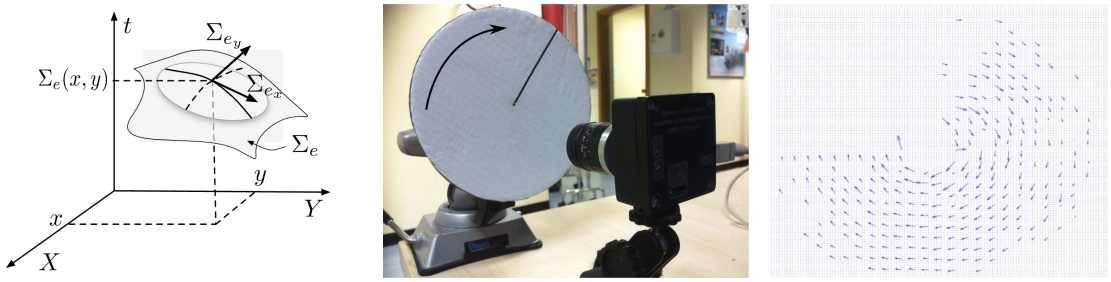
expensive, iterative EM scheme that can hardly cope with real-time constraints, and lacks robustness.

In Paper D, I proposed an efficient method to detect and tracking natural features from a moving event camera, in the context of visual-inertial odometry. To achieve this, I synthesize *motion-compensated* event frames and apply traditional feature detectors [165] and trackers [103] on those. Hence, motion-compensated events provide a useful representation of edge patterns; however, such representations depend on the apparent motion, and therefore, may introduce some tracking drift as the edge appearance changes over time.

To remove drift, events can be combined with a *motion invariant* representation of the scene, which may either be provided by a conventional camera [65], or built from past events (see Section 1.2.4).

Optical Flow. The problem of computing the velocity of objects on the image plane (without prior knowledge about the scene geometry or camera motion) is referred to as optical flow estimation. Computing flow from events is attractive because of the fine timing information from the events that allows measuring high speed flow, at a potentially low computational cost. Because of these promises, numerous methods to estimate optical flow from event data have been presented.

Early works [36, 14, 13] estimated normal optical flow, *i.e.* the component of the normal flow perpendicular to the edge (parallel to the brightness gradient). Normal flow is relatively straightforward to compute using an image containing the timestamp of the last event at each pixel (known as the *surface of active events* [36], Fig. 1.9) [36, 13].



(a) (b) (c)

Figure 1.9 – Event-based visual flow [13]. Fig. 1.9a: principle of flow computation – the surface of active events Σ_e is derived to provide an estimation of orientation and amplitude of motion. Fig. 1.9b: experimental setup consisting of a rotating black bar. Fig. 1.9c: normal optic flow computed from the events (cumulated for a period of 5ms).

More recent methods [10, 102, 62, 213] estimate the full optical flow (*i.e.* both tangential and normal components). While the full optical flow contains more information

than normal flow, it is considerably harder to compute and requires some form of regularization, in the form of priors, which can be either be handcrafted or learned.

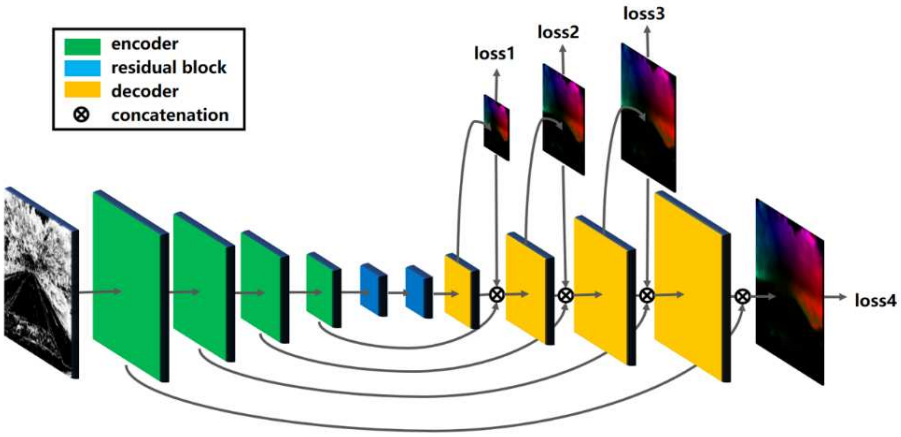
In particular, [10] showed how to estimate optical flow jointly with image intensity, for arbitrary scenes (including dynamic and deformable objects) and camera motion (Fig. 1.18). They used a variational optimisation framework to jointly estimate dense optical flow and image intensity over a sliding window of events, combining an event data term with spatial and temporal regularizers. However, optimisation was carried out using a primal-dual algorithm, thus requiring a GPU to run in near real-time. In addition, due to the handcrafted regularizers used, the quality of the optical flow (and image reconstructions) was limited.

Very recently, machine learning approaches [213] have shown impressive results for dense optical flow estimation from event data. The seminal work, EV-FlowNet [213], used a neural network operating on an image representation of the event stream, combined with a self-supervised photometric consistency loss computed from adjacent grayscale frames of the DAVIS [18] sensor (Fig. 1.10). This was later extended to fully unsupervised learning of optical flow from event data [214, 204], using a focus loss inspired by the contrast maximization approach proposed in [62, 59].

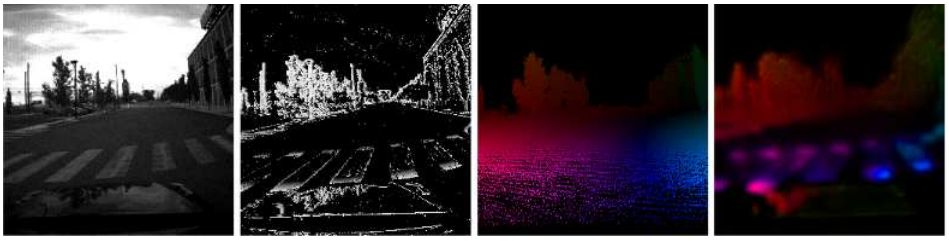
1.2.3 Visual Odometry and SLAM

Visual Odometry (VO), or visual *Simultaneous Localization and Mapping* (SLAM) is the process of simultaneously estimating the pose of a moving camera as well as a three dimensional map of the environment, using visual input (frames or events) from the camera.

With a frame-based camera. SLAM with conventional cameras is a well-studied problem, and has greatly evolved over the last two decades: from the first systems that demonstrated monocular SLAM in real-time [34, 82] to mature algorithms that work robustly in real-time [34, 47, 126, 54]. Most SLAM approaches can be split in two categories: *feature-based* and *direct* methods. Feature-based methods [34, 126] extract a sparse set of salient image features (*e.g.* points, lines) in each image; match them in successive frames using invariant feature descriptors; recover both camera motion and structure using epipolar geometry; finally, refine the pose and structure through reprojection error minimization. By contrast, direct methods [46, 47, 54] use intensity values in the image directly to estimate structure and motion; both arise as the result of an optimization problem minimizing the *photometric* reprojection error (as opposed to the geometric reprojection error used in feature-based methods). Thus, the pixel correspondence problem is solved implicitly through optimisation instead of explicitly through feature descriptors. Regardless of the approach used, the SLAM problem is



(a) Network architecture.



(b) Results. From left to right: frame, events, ground truth flow, estimated semi-dense flow.

Figure 1.10 – EV-FlowNet [213].

usually split into two sub-tasks running in parallel: a *localization* thread that estimates the camera pose (assuming a 3D map of the environment is available), and a *mapping* thread that builds and continuously refines the 3D map (assuming that the camera poses are known).

SLAM with an Event Camera

Solving the SLAM problem with an event camera in its most general setting (6-DOF motion and natural 3D scenes) with a single event camera is a challenging problem. For this reason, previous works addressed easier instances of the problem, *e.g.* localizing with respect to a given 3D map [123, 197, 61, 20], performing 2D SLAM [30, 198, 79], or using additional visual sensors such as a frame-based camera [24, 87] or a depth sensor [196]. I now review a few important related works.

Localization. Event-based localization with respect to a map was first proposed by [123] (see Fig. 1.11a). They used a map consisting of a set of lines, where every incoming

event was used to update a given line track, and thus, the pose estimate. The camera pose was obtained by minimizing the point-to-line distance between the reprojection of each line and the events belonging to it. The approach could track a quadrotor performing a flip (albeit, not in real-time nor in closed-loop). However, the approach was limited to very simple environments (a set of lines with known position).

A probabilistic filter framework to track an event camera with respect to a known map (Fig. 1.11b) was presented in [61]. The algorithm allowed for 6-DOF event camera tracking in natural environments, while providing pose updates upon the arrival of each event. However, it required a 3D photometric map of the environment.

Recently, [130] presented a method to relocalize the 6-DOF pose of an event camera. To achieve this, they trained a stacked spatial LSTM [72] network to learn the camera pose from a given training dataset.

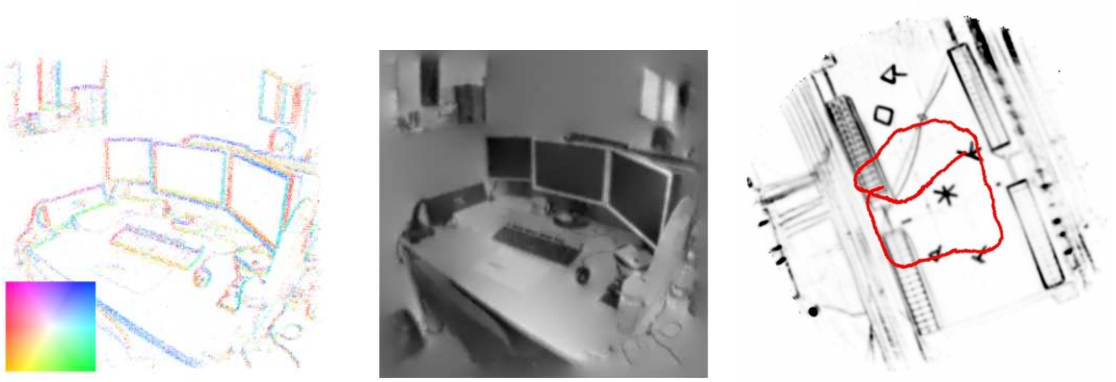
(a) Line-based tracking [123]. **(b)** Photometric tracking [61].

Figure 1.11 – Localization with an event camera.

2D SLAM. Rotation-only SLAM was presented in [30, 79]. In particular, Kim *et al*. [79] used parallel Kalman filters to estimate the 3D orientation of an event camera while generating a high resolution panorama of a natural scene (Fig. 1.12a).

An event-based 2D SLAM system was presented in [198] (Fig. 1.12b). However, the system design was limited to planar motions (*i.e.*, 3-DOF) and planar scenes parallel to the plane of motion consisting of artificial black-and-white line patterns.

Additional Visual Sensing. The 2D method above [198] was extended to 3D in [196], but relied on an external RGB-D sensor attached to the event camera for depth estimation (Fig. 1.13a). The depth sensor introduced bottlenecks, which deprived the system

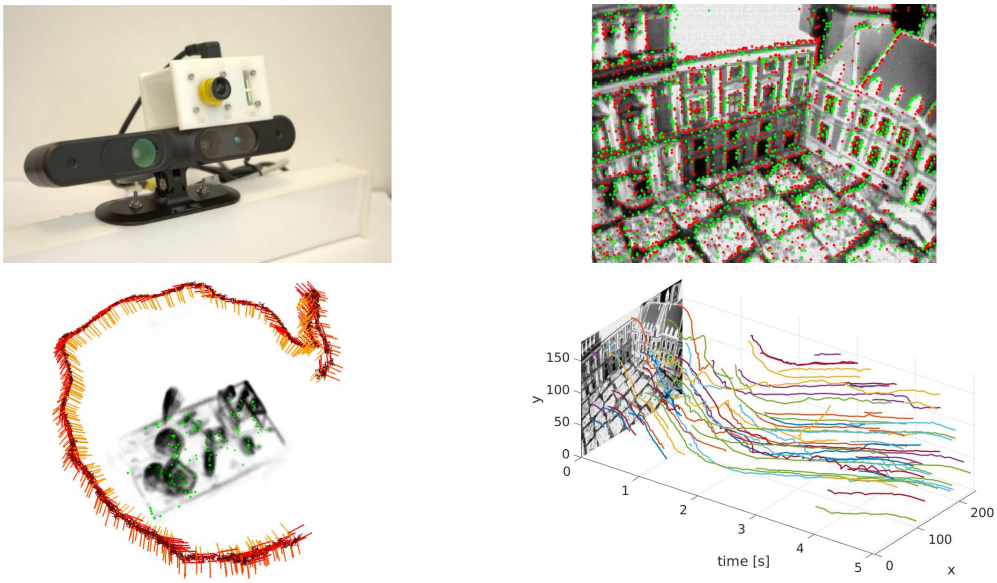


(a) Mosaicing and Tracking [79] **(b)** SLAM in planar scenes [198]

Figure 1.12 – 2D SLAM with an event camera.

of the low latency and high-speed advantages of event cameras.

A visual odometry system operating in a parallel tracking-and-mapping manner was presented in [87]. The system recovered 6-DOF motions in natural scenes by tracking a sparse set of features using the event stream. However, the system required intensity images to first detect the features.



(a) SLAM with depth augmented sensor [196]. **(b)** Odometry with events and frames [87].

Figure 1.13 – Event-based SLAM with additional visual sensing. Fig. 1.13a: setup with a combined event and depth sensor used in [196] (top), estimated trajectory and 3D map (bottom). Fig. 1.13b: frame and superimposed events used in [87] (top), feature tracks from the event stream (bottom).

3D, 6-DOF SLAM The first system to show 6-DOF SLAM with an event camera was [80], which proposed a system with three interleaved probabilistic filters to perform pose tracking as well as depth and intensity estimation in natural scenes (Fig. 1.14). However, the system was computationally expensive, requiring a GPU to reconstruct a regularized photometric depth map in real time. Thus, it was not suitable for computationally limited platforms.

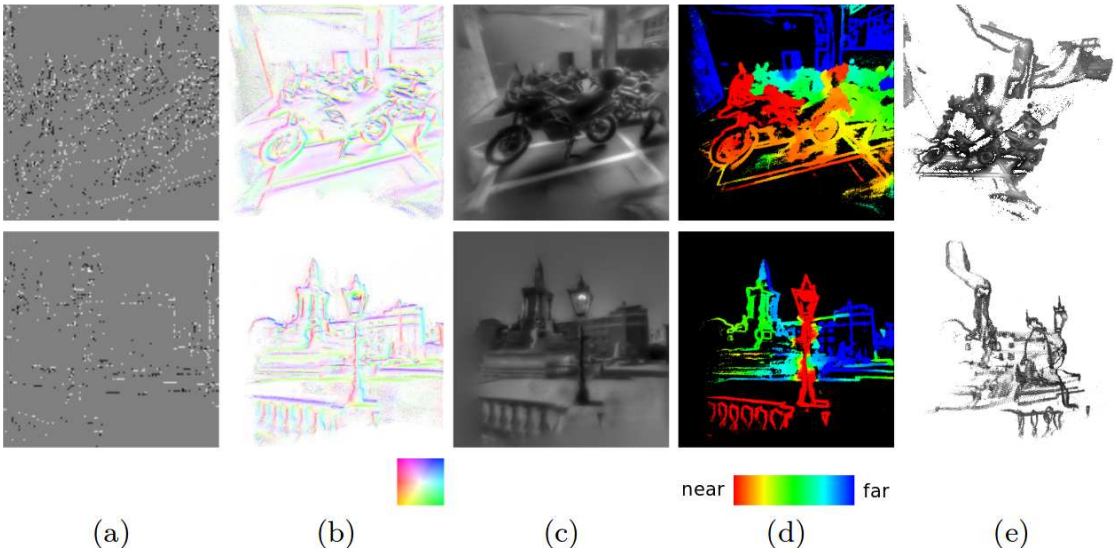


Figure 1.14 – 3D Reconstruction and 6-DoF Tracking [80]. (a) event visualization, (b) gradient estimation, (c) intensity reconstruction, (d) depth estimation, and (e) semi-dense point cloud.

In this thesis, I present EVO (Paper C), a system that performs 3D, 6-DOF SLAM in natural scenes, which was developed at the same time as [80]. One of the key ingredients of EVO is the event-based, 3D reconstruction approach also developed in this thesis (Paper B). Contrarily to [80], EVO uses a geometric approach which is computationally efficient, and can run in real-time on a CPU in case of moderate motions. However, unlike [80], it does not process the event stream in an event-by-event fashion. Instead, EVO uses instead batches of events, which incurs some latency.

Finally, very recent works [214, 204] have applied deep learning techniques to estimate depth and egomotion (relative camera pose) from event data (Fig. 1.15); however, both of these approaches were trained on driving event data from the MVSEC dataset [212], and fail to generalize to different environments not seen at training time.

Visual-inertial Odometry. The robustness of event-based visual odometry and SLAM systems can be improved by sensor fusion, *e.g.*, by combining an event camera with an inertial measurement unit (IMU) rigidly attached. For this and other reasons, some event cameras have an integrated IMU (see Table 1.1). Only few works have considered using an event camera with an IMU.

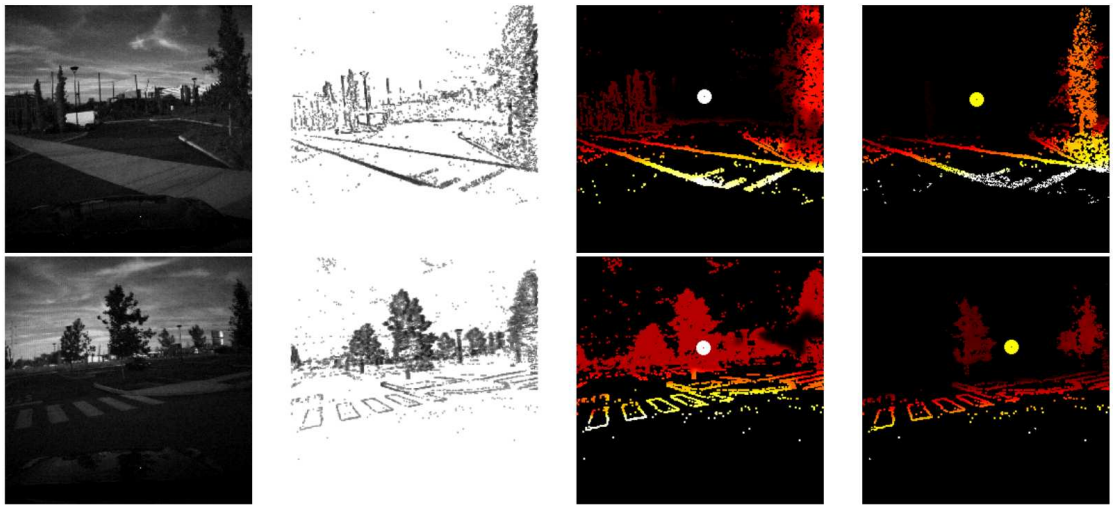
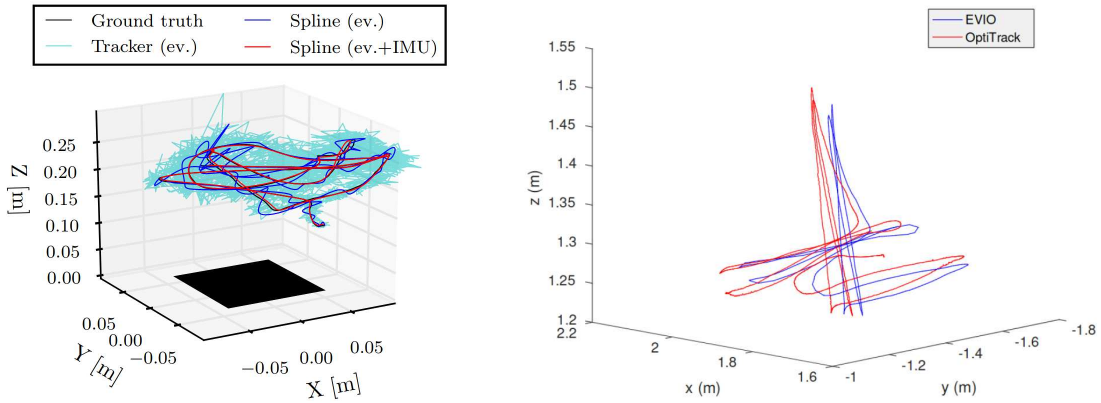


Figure 1.15 – Sample results from [214], which proposed to learn depth and egomotion from event data. From left to right: Grayscale image, event image, depth prediction with heading direction, groundtruth with heading direction. Heading direction is drawn as a circle.

A continuous time framework [135] to fuse events and inertial measurements was presented in [122]. Their method optimizes a combined objective functional with inertial- and event-reprojection error terms over a segment of the camera trajectory, in the style of visual-inertial bundle adjustment. However, their approach is not suited for real-time usage because of the expensive optimization required to update the spline parameters upon receiving every event.

Recently, [211] proposed a feature-based visual-inertial odometry algorithm for event cameras (Fig. 1.16b) that works in real-time (albeit, for limited motion speeds, and number of features). To achieve this, they track a set of features in the event stream using [210] (which jointly solves for the feature appearance and optical flow). The feature tracks are then fed to an extended Kalman filter [119] to produce new pose estimates.

In this thesis, I introduce a novel event-based visual-inertial odometry algorithm (Paper D) which follows a similar principle. My method tracks a set of visual features on *motion-corrected* event frames, which are computed efficiently from the inertial measurements, camera motion, and estimated scene structure. These visual features are then fused with the inertial measurements using a nonlinear optimization framework [94], thus providing pose estimates that are significantly more accurate than [211]. In addition, I extend the above-mentioned algorithm in Paper E to leverage conventional frames as a supplementary input [164], which yields a boost in robustness and accuracy. Finally, I also demonstrate the system on a computationally constrained quadrotor platform, enabling it to fly closed-loop in low light and HDR scenarios by exploiting the advantages of the event camera.



(a) Continuous-time framework [122]. **(b)** Filter-based approach [211].

Figure 1.16 – Event-based visual-inertial odometry. Fig. 1.16a: spline trajectory (in red) estimated using events and inertial measurements within the continuous-time framework [122]. In this example, the scene consists of a single black square. Fig. 1.16b: trajectory estimated by EVIO [211] (in blue) compared to ground truth from a motion capture system (in blue), on a sequence from [124].

1.2.4 Image and Video Reconstruction

In theory, the stream of events from an event camera encodes the entire visual signal – in a highly compressed form – and could thus be decompressed to recover a video with high frame rate and dynamic range. Events represent brightness changes, and so, in ideal conditions (noise-free scenario, perfect sensor response, etc.), integration of the events should yield "absolute" brightness. However, real event cameras are noisy and differ from the ideal camera model, making the reconstruction problem ill-posed. Thus, a few authors have investigated the problem of image reconstruction with events.

Interestingly, the first evidence that it is possible to recover intensity information from event data was shown in the context of motion estimation [30, 79, 80] (it should be noted, however, that image reconstruction was not the end goal of these work, but merely an intermediate representation). In particular, [30] used a distributed message-passing algorithm consisting of local gradient descent operations to jointly estimate the image gradient, camera velocity and optical flow from a large set of events collected by a rotating event camera (Fig. 1.17). However, these systems had severe limitations: [30, 79] assumed a rotating event camera, while [80] only worked in a static scene.

Bardow *et al.* [10] proposed to estimate intensity and optical flow *jointly* from sliding windows of events, through a variational energy minimization framework. They showed the first general video reconstruction framework from events that is applicable to dynamic scenes. However, their energy minimization framework employed multiple hand-crafted regularizers, which could result in severe loss of detail in the reconstructions (Fig. 1.18).

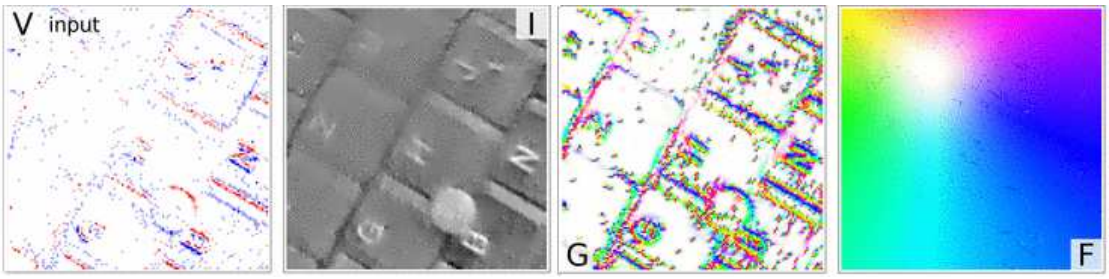


Figure 1.17 – Interacting Maps for Fast Visual Interpretation [30]. From left to right: input events (V), intensity reconstruction (I), gradient map (G), optic flow map (F).

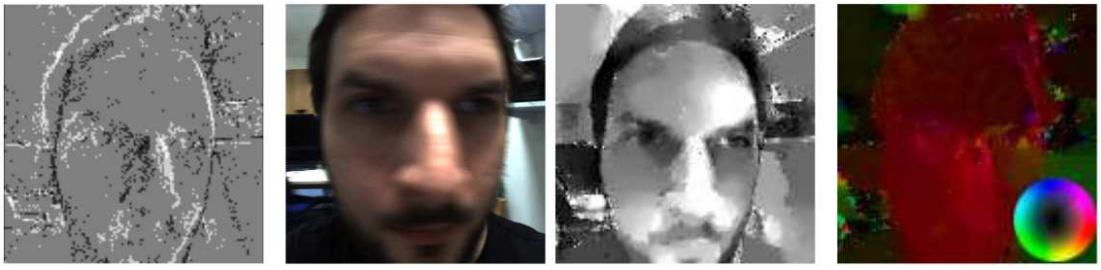


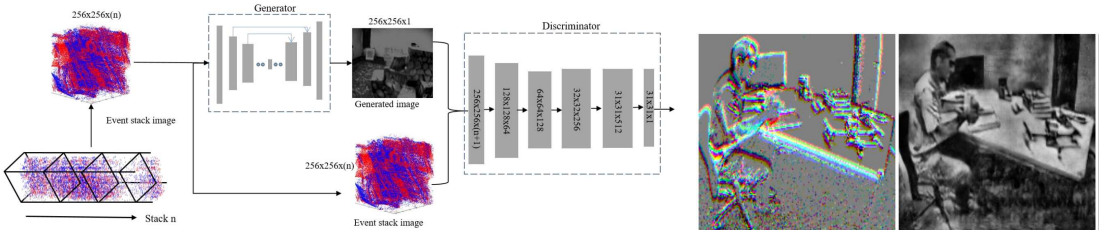
Figure 1.18 – Simultaneous optical flow and intensity reconstruction [10]. From left to right: Events, frame, intensity reconstruction, and estimated optical flow map.

Recently, methods based on direct event integration (*i.e.* without assumption on the scene or motion) have emerged. Munda *et al.* [125] cast intensity reconstruction as an energy minimization problem defined on a manifold induced by the event timestamps (Fig. 1.19), and Scheerlinck *et al.* proposed to filter the events with a high-pass filter prior to integration. These approaches defined the state-of-the-art prior to this thesis. Yet, both suffer from artifacts ("bleeding edges") inherent to direct event integration.

Figure 1.19 – Intensity reconstruction using manifold regularization [125]. From left to right: raw events, reconstruction result, and visualization of the *surface of active events*, which records the time of the last event at each pixel.

Machine learning approaches have also been explored, starting with Barua *et al.* [12], who used K-SVD [3] on simulated data to learn a dictionary mapping small patches

of integrated events to an image gradient (later converted to an intensity image via Poisson integration). [118] used a generative adversarial network (GAN) [67] to learn image reconstruction from events in a supervised fashion, however the reconstructed images suffer from many artefacts and are not temporally consistent (Fig. 1.20). Bardow *et al*. [9] showed that image reconstruction can also be learned in an unsupervised fashion using a GAN.



(a) Approach. **(b)** Example result.

Figure 1.20 – Learning supervised intensity reconstruction using a GAN [118]. Fig. 1.20a illustrates the reconstruction approach. Fig. 1.20b shows a sample reconstruction result (left: input events, right: reconstruction).

In Paper F1 of this thesis, I introduce a recurrent neural network to reconstruct videos from a stream of events, and train it on large amount of simulated event data generated with my simulator ESIM (Paper A). The network reaches unprecedented video reconstruction quality (improvement of $>$ 20% with respect to previous works), and can synthesize high frame rate ($>$ 5,000 frames per second) and HDR videos.

Finally, in Paper F2, I demonstrate the effectiveness of the reconstructions as an intermediate representation for event data, which is *motion invariant* and *transferrable*. In other words, off-the-shelf computer vision algorithms can be applied to the reconstructions for a variety of tasks, which opens the door to transfer learning from frames to events.

2 Contributions

This chapter summarizes the key contributions of the papers that are reprinted in the appendix. It further highlights the connections between the individual results and refers to related work and video contributions.

In total, this research has been published in 4 peer-reviewed conference publications and 4 journal publications (one in the *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, one in the *International Journal of Computer Vision (IJCV)*, and two in the *Robotics Automation Letters (RA-L)*.

These works led to several research awards, patents, and open-source software.

Awards.

- RA-L Best Paper Award (Honorable Mention), 2018
- Qualcomm Innovation Fellowship Europe, 2018
- Misha Mahowald Prize for Neuromorphic Engineering, 2017
- BMVC Best Industry Paper Award, 2016

Patents.

- H. Rebecq, G. Gallego, D. Scaramuzza, **Simultaneous Localization and Mapping with an Event Camera**, US 2019/0197715 A1 . Issued on January 3, 2018.
- H. Rebecq, T. Horstschaefer, D. Scaramuzza, **Visual-Inertial Odometry with an Event Camera**, EU 17189223.5 - 1906 . Filed on November 6, 2017.

Software.

- ESIM: an open event camera simulator
- EMVS: 3D reconstruction with an event camera
- High speed and high dynamic range video with an event camera
- Omnidirectional camera model for Blender

2.1 Infrastructure for Event Cameras: Simulator and Datasets

In contrast to conventional cameras, that have existed for decades, event cameras are a relatively novel class of sensors. Thus, at the time of writing they are still scarce and expensive to get, slowing down progress of the research community. As a result, there is a huge demand for high-quality, labeled event datasets for algorithm prototyping and benchmarking, as well as for training supervised machine learning algorithms.

The work conducted during this thesis contributes to addressing this issue in a number of ways. First, I developed a theoretically sound event camera simulator, which is efficient, accurate and versatile (Paper A). The simulator is released open-source. I also contributed in collecting, labeling and releasing high quality datasets to the public in order to foster progress in the community (Section 2.1.2).

2.1.1 Paper A: ESIM: an Open Event Camera Simulator

(P1) H. Rebecq, D. Gehrig, and D. Scaramuzza. "ESIM: an Open Event Camera Simulator". In: *Conf. on Robotics Learning (CoRL)*. 2018

Despite fast progress in hardware development, event cameras are – to this date – still scarce and expensive sensors, which slows down progress of the research community. Thus, there is demand for cheap, high-quality synthetic event datasets for algorithm prototyping, deep learning and benchmarking. The development of an event camera simulator, however, is not trivial since event cameras work fundamentally differently from frame-based cameras, measuring changes of light asynchronously as they occur on the image plane. The main contribution of this work is to present the first event camera simulator that can generate a large amount of reliable event data. The contributions are two-fold. First, a theoretically sound, adaptive rendering scheme is introduced, that only samples frames when necessary, through a tight coupling between the rendering engine and the event simulator. Second, an efficient, C++ implementation of the simulator is released as open source to the community.

Related Software

(S1) https://github.com/uzh-rpg/rpg_esim

Related Videos

(V1) https://youtu.be/ytKOIX_2clo

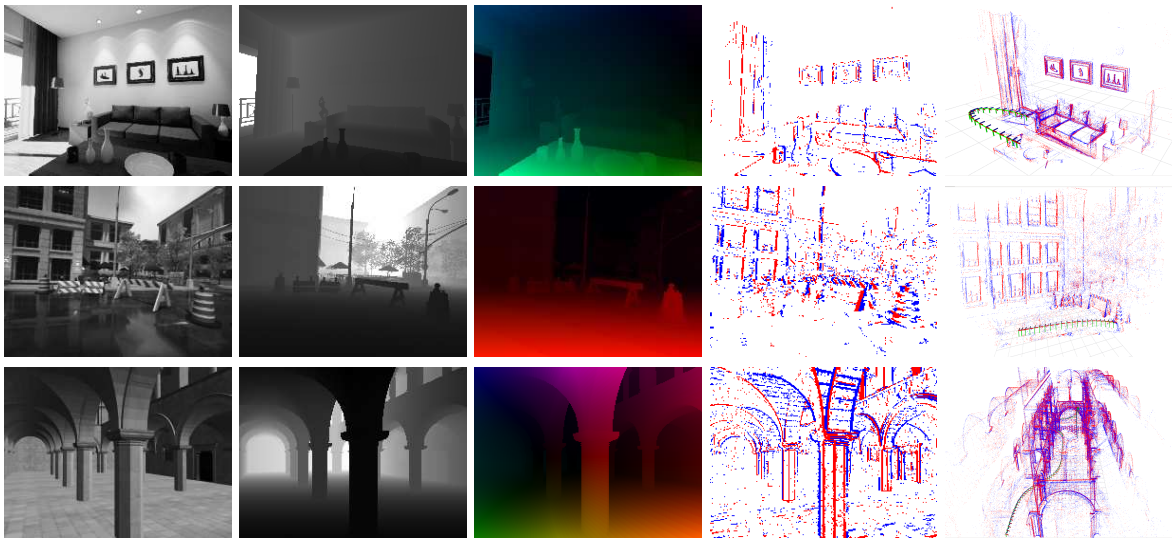


Figure 2.1 – Example outputs of the event simulator ESIM.(a) Image. (b) Depth map. (c) Optical Flow (Color-coded). (d) Events (Positive and Negative). (e) Point cloud (3D events) + Camera trajectory

2.1.2 Novel Event Camera Datasets

Besides the main contributions presented in this thesis, I contributed to collecting high quality event camera datasets, which are briefly presented below for completeness.

Drone Racing Dataset for High Speed State Estimation

(A1) J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza. "Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2019

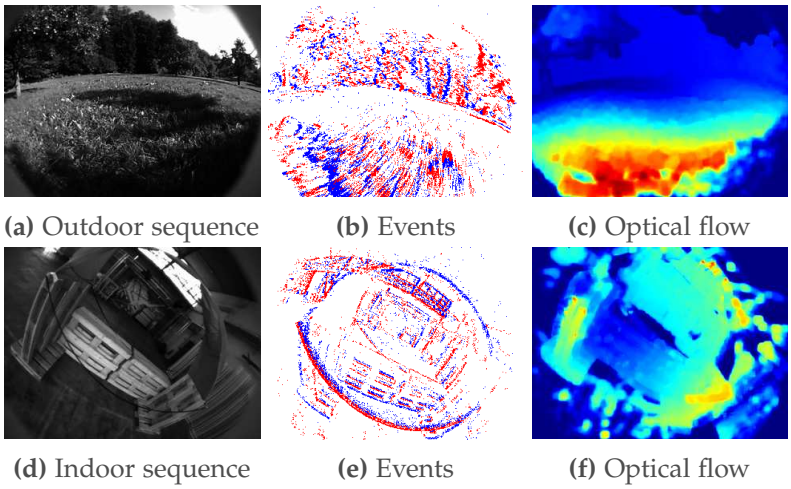
Aggressive trajectories feature large accelerations and rotational motions, which induce large apparent motions in vision sensors, all of which increase the difficulty of state estimation. Existing benchmark datasets do not address these types of trajectories, instead focusing on slow speed or constrained trajectories. In this paper, we introduced the UZH-FPV Drone Racing dataset, consisting of over 27 sequences, with more than 10 km of flight distance, captured on a first-person-view (FPV) racing quadrotor flown by an expert pilot. The dataset features event data, camera images, inertial measurements, and precise ground truth poses. Our goal is to enable advancement of the state of the art in motion estimation with event cameras by providing a dataset that is beyond the capabilities of existing state estimation algorithms.

Related Videos

(V2) https://youtu.be/G5w4ZcEzvoo

Related Datasets

(D1) http://rpg.ifi.uzh.ch/uzh-fpv.html



(a) Outdoor sequence **(b)** Events **(c)** Optical flow

(d) Indoor sequence **(e)** Events **(f)** Optical flow

Figure 2.2 – The drone racing dataset contains synchronized events, frames and IMU data.

Color Event Camera Dataset.

(A2) C. Scheerlinck, H. Rebecq, T. Stoffregen, N. Barnes, R. Mahony, and D. Scaramuzza. "CED: Color Event Camera Dataset". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2019

Most diurnal animals have some form of color vision, and most conventional cameras offer color sensitivity. However, until recently, event cameras have been limited to outputting events in the intensity channel. Recent advances have resulted in the development of color event cameras, such as the Color-DAVIS346 [116, 115]. In this work, we present and release the first *Color Event Camera Dataset (CED)*, containing 50 minutes of footage with both color frames and events. CED features a wide variety of indoor and outdoor scenes, which we hope will help drive forward event-based vision research.

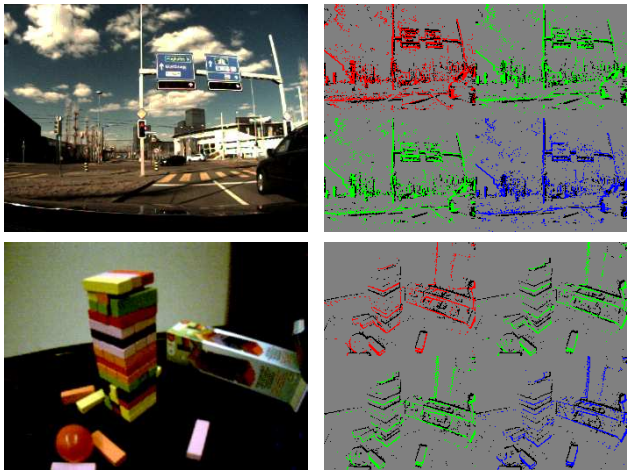
Related Videos

(V3) https://youtu.be/R9BiRN7f7uY

Related Datasets

(D2) http://rpg.ifi.uzh.ch/CED

(a) Color-DAVIS346



(b) Sample sequences from the CED dataset.

Figure 2.3 – Our Color Event Camera Dataset (CED) features diverse sequences, and provides color images (Fig. 2.3b, left column) and color events (Fig. 2.3b, right row) from the Color-DAVIS346 (Fig. 2.3a).

2.2 Direct SLAM with an Event Camera

This section explores how to perform SLAM using only event data from a moving event camera. A core challenge is that of *data association*: since events do not provide direct brightness measurements, establishing direct correspondences between events triggered at different points in time is hard. The core contribution of this section is a method to perform multi-view stereo from a moving event camera ("EMVS", Paper B), which does not require data association. Paper C complements EMVS with a novel, direct pose tracker to obtain an efficient 6-DOF SLAM system for event cameras.

2.2.1 Paper B: Multi-View Stereo with a Moving Event Camera

(P2) H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza. "EMVS: Event-based Multi-View Stereo—3D Reconstruction with an Event Camera in Real-Time". In: *Int. J. Comput. Vis.* 126.12 (Dec. 2018), pp. 1394–1414. DOI: 10.1007/s11263-017-1050-6

This paper introduces the problem of "event-based multi-view stereo" (EMVS), *i.e.* 3D reconstruction from an event camera moving along a known trajectory. Unlike traditional multi-view stereo methods, which rely on photometric information to establish correspondences between images taken from different viewpoints, this method does not require explicit data association. Instead, it elegantly exploits two essential properties of event cameras: (*i*) their ability to respond to salient edges in the scene – thus providing semi-dense geometric information without pre-processing, and (*ii*) the fact that events provide continuous measurements as the sensor moves. This results in a simple algorithm, able to produce accurate, semi-dense depth maps.

Related Software

(S2) https://github.com/uzh-rpg/rpg_emvs

Related Videos

(V4) https://youtu.be/EFpZcpd9XJ0

(a) EMVS illustration. **(b)** Scene preview. **(c)** Reconstructed 3D map.

Figure 2.4 – As the camera moves, events are triggered on the image plane. To each observed event corresponds a ray that spans the possible 3D-structure locations. The areas of high ray density indicate the locations of the 3D points, and are discovered as the sensor moves.

2.2.2 Paper C: 6 DOF SLAM with an Event Camera

(P3) H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza. "EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-Time". In: *IEEE Robot. Autom. Lett.* 2.2 (2017), pp. 593–600. DOI: 10.1109/LRA.2016.2645143

This work builds on top of the 3D reconstruction algorithm presented above, and introduces EVO ("Event-based Visual Odometry"): an algorithm capable of tracking the 6 DOF pose of an event camera from events only, in real-time on a CPU. Inspired by direct visual odometry methods, the tracking module of EVO estimates the current camera pose by solving for a 6 DOF transformation that minimizes the alignment error between the current semi-dense 3D map and an *event image* obtained by accumulating a small number of events. The local 3D map is continuously updated using the EMVS method presented in Paper B, as new camera poses are provided by the tracking thread. EVO is accurate (accumulated drift is around 1% of the trajectory) and can work for high-speed motions or high dynamic range scenarios.

Related Videos

(V5) https://youtu.be/bYqD2qZJlxE

Related Demonstrations

(D1) H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza. "EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-Time". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2018

(D2) H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza. "EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-Time". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018

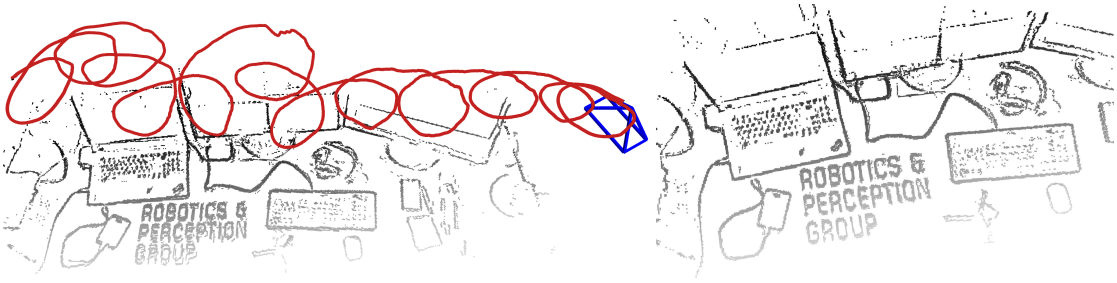


Figure 2.5 – Example of trajectory and 3D map estimated by EVO [150] using only events.

2.3 Sensor Fusion for Event-based Visual Odometry

As highlighted in the previous section, the main challenge for motion estimation with event cameras is that of data association. The works presented in the previous section have essentially bypassed that problem using direct methods that do not require explicit data association. While this has allowed to establish that event data alone contains sufficient information to allow for 6-DOF SLAM in natural environments, and conditions in which conventional cameras fail, these methods are nor highly efficient nor very robust. Indeed, the cost for building a *motion-invariant* representation – *e.g.* a 3D map in EVO (Paper C) – is time: building a reliable 3D map requires a large number of events (about one second of event data), during which the tracker is blind, making the system brittle.

To address this problem, I now explore the benefits of using an event camera in combination with additional sensors in order to allow for explicit data association. In Paper D, I complement an event camera with an inertial measurement unit (IMU) and show that it allows to build a robust and efficient visual-inertial odometry algorithm. The key contribution here is the concept of *motion-compensated event frames*, and an efficient way to synthesize those in the context of visual-inertial odometry. These event frames can be viewed as a useful, "image-like" representation of event data upon which traditional feature tracking algorithms can be applied for short time scales. This system is further extended in Paper E to use (optional) standard frames, through a loosely coupled approach that tracks features independently on event frames and standard frames. In addition, I also demonstrate the applicability of the method for a challenging robotic application, namely closed-loop, autonomous quadrotor flight in a dark room.

2.3.1 Papers D, E: "Ultimate SLAM" with Events, Frames and IMU

(P4) H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization". In: *British Mach. Vis. Conf. (BMVC)*. 2017

(P5) A. Rosinol Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: *IEEE Robot. Autom. Lett.* 3.2 (Apr. 2018), pp. 994–1001. DOI: 10.1109/LRA.2018.2793357

Paper D presents an approach to solving the problem of event-based visual-inertial odometry, *i.e.* pose tracking using an event camera and a synchronised IMU. To achieve this, I introduce the concept of *motion-compensated event frames* and propose an efficient way to synthesize those, based on inertial measurements plus the currently estimated camera motion and scene structure. These motion-compensated event frames can be seen as a representation of event data, upon which a conventional feature detector [165] and tracker [103] can be successfully applied. Finally, the feature tracks are fused with a keyframe-based, visual-inertial approach based on nonlinear optimization [94] to yield pose estimates. The system is evaluated on the publicly available event camera dataset [124], demonstrating state of the art accuracy. Moreover, I show qualitatively that the system can track extremely fast motions (*e.g.* an event camera spinning on a leash, Fig. 2.6), which were previously inaccessible to conventional cameras.

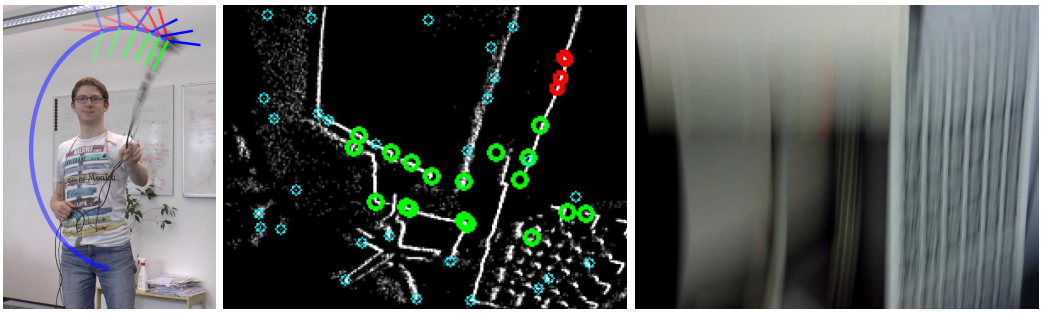


Figure 2.6 – Left: Spinning an event camera attached to a leash. My algorithm estimates the camera trajectory (superimposed on the image) using events and inertial measurements from the spinning sensor. Middle: motion-corrected event frame with feature tracks superimposed. Right: frame from a standard camera spinning at the same speed, with severe motion blur.

In Paper E, I show how to complement the previous algorithm with a conventional camera to further increase the accuracy and robustness of state estimation. In addition to tracking features in motion-compensated event frames, the system now also tracks features independently in conventional frames. Both both sources of feature tracks are then fused with the IMU with a traditional visual-inertial odometry algorithm. A thorough analysis of the method shows an improvement of 130% in accuracy over the *event-only* system [149]. In addition, I demonstrate the applicability of this method

for robotic applications. The algorithm is run onboard a quadrotor to provide state estimation in real-time, and used to demonstrate the first closed-loop control of a quadrotor (Fig. 2.7) using an event camera for state estimation, allowing it to fly safely in challenging environments, such as in a dark room.

Related Videos

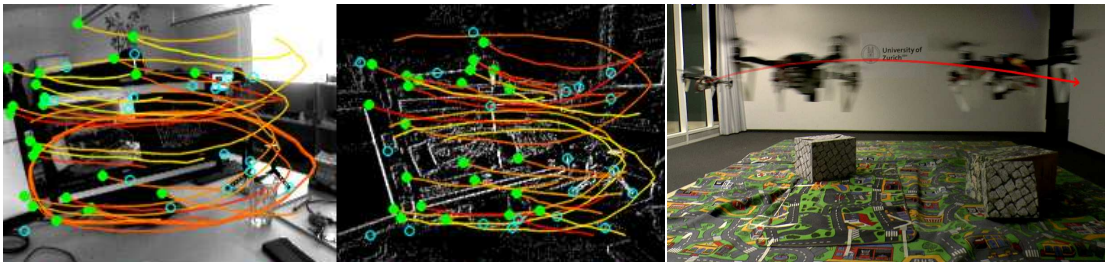
(V6) https://youtu.be/F3OFzsaPtvI

(V7) https://youtu.be/jIvJuWdmemE

Related Demonstrations

(D3) H. Rebecq, A. Rosinol Vidal, T. Horstschäfer, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2018

(D4) H. Rebecq, A. Rosinol Vidal, T. Horstschäfer, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018



(a) **(b)**

Figure 2.7 – (a) Feature tracks on standard frames (left) and event frames (right). (b): The system was used to fly an autonomous quadrotor fast, in a low-lit room, using only onboard computing.

2.4 Video Reconstruction from Event Data and Applications

While the main focus of the previous chapters was on motion estimation, in this chapter I examine a more general problem, with applications to both low-level tasks (*e.g.* motion estimation) and high level tasks (*e.g.* object detection). Specifically, I investigate the possibility of reconstructing high quality videos from event data only. My goal in this endeavor is two-fold: (*i*) revealing the astounding amount of information that is encoded in the event stream, and (*ii*) synthesizing an event representation with two desirable properties: *motion invariance* and *transferrability*. Motion invariance means that the appearance of the reconstructed images does not depend on the motion direction. Transferrability means that the reconstructions possess natural image statistics, thus conventional computer vision algorithms can be applied to them, opening the door to transfer learning from frames to events.

In Paper F1, I introduce a novel network to reconstruct high quality videos using event data only, train it on synthetic event data from my simulator ESIM (Paper A), and demonstrate very good generalization to real event data. Then, in Paper F2, I explore different applications of the reconstructions, and show that they allow – to some extent – to bring the mainstream of classical computer vision to event cameras.

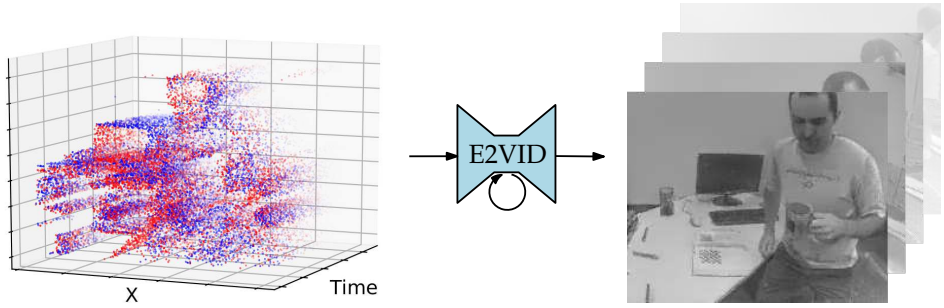


Figure 2.8 – The network presented in Paper F1 converts a spatio-temporal stream of events with microsecond temporal resolution into a high-quality video.

2.4.1 Paper F1: High Speed, HDR Video with an Event Camera

(P6) H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. "High Speed and High Dynamic Range Video with an Event Camera". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2019). DOI: 10.1109/TPAMI.2019.2963386

This work proposes to reconstruct videos from event data. Instead of relying on any hand-crafted priors, I propose a novel recurrent network to learn video reconstruction from events directly using a large amount of simulated event data obtained with my simulator ESIM (Paper A). This network surpasses state-of-the-art reconstruction methods by a significant margin in terms of image quality ($>20\%$), while comfortably running in real-time. I demonstrate high frame rate video synthesis ($>$ 5000 frames per second) of high-speed phenomena, such as a bullet hitting an object (Fig. 2.9), as well as high dynamic range video.

Related Publications

(R1) H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. "Events-to-Video: Bringing Modern Computer Vision to Event Cameras". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019

Related Software

(S3) https://github.com/uzh-rpg/rpg_e2vid

Related Videos

(V8) https://youtu.be/eomALySSGVU

Related Demonstrations

(D5) H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. "Events-To-Video: Real-Time Image Reconstruction With an Event Camera". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019

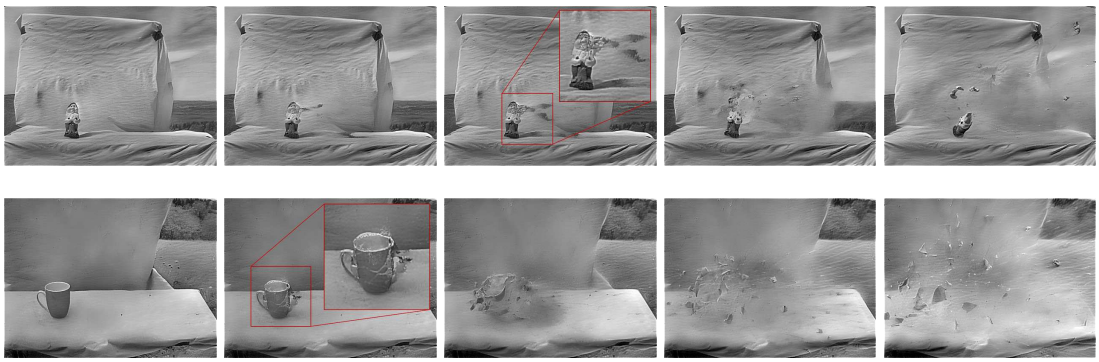


Figure 2.9 – Video reconstructions of high speed physical phenomena, synthesized at $>$ 5,000 FPS with my network. They reveal details invisible to a consumer camera.

2.4.2 Paper F2: Downstream Applications of Video Reconstruction

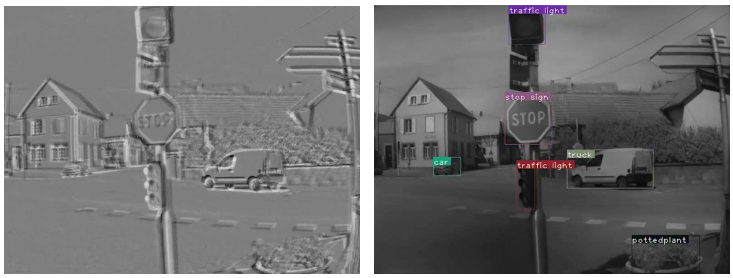
(P6) H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. "High Speed and High Dynamic Range Video with an Event Camera". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2019). DOI: 10.1109/TPAMI.2019.2963386

Since the output of an event camera is fundamentally different from natural images, existing computer vision techniques cannot be directly applied to event data. However, can they be applied to image reconstructions instead? I demonstrate here that the image reconstructions can be viewed as a representation for event data. Unlike other event representations, reconstructed images are *motion invariant*, *i.e.* their appearance does not depend on the motion direction. In addition, they are *transferrable*, *i.e.* any standard computer vision algorithm can be applied to them, enabling the application of pre-existing vision algorithms to event data. A quantitative analysis is performed on two different downstream tasks: object classification from events and visual-inertial odometry. I also show the method allows to address problems that had not been solved with event data, *e.g.* monocular depth prediction or object detection (Fig. 2.10).

Related Videos

(V8) https://youtu.be/eomALySSGVU

(a) Monocular depth from events. Left to right: events, reconstruction, predicted depth [97].



(b) Object detection from events. Left: events. Right: reconstruction and predicted labels [158].

Figure 2.10 – Image reconstructions from events can be used as input to off-the-shelf computer vision algorithms, allowing to transfer existing neural networks weights to the event domain.

2.5 Additional Contributions

I now present additional contributions of this thesis, which consist of a set of robotic demonstrators, as well as a publication unrelated to the main topic of this thesis.

2.5.1 Quadrotor Demonstrators for DARPA FLA

I believe that live demonstrations are one of the best way to communicate research results, while pushing to develop systems that are lightweight and work in the real world. My lab participated in the DARPA FLA (Fast, Lightweight, Autonomy) program [117] from 2016 to 2018. The contribution of this thesis to the project was to explore and highlight the potential of event cameras for fast, autonomous quadrotor flight, with a focus on state estimation. In this context, I built three demonstrators of my research, showing its applicability to quadrotor flight. Each demonstration was run on the field, and presented each time to a group of senior investigators from DARPA. In all three cases, the demonstrator was running in closed-loop (tight coupling between the perception and the control loop), *i.e.* all the computations were run onboard the quadrotor in real-time. In this context, I also contributed to recording a drone racing dataset with event data from an onboard event camera [41].

(a) Quadrotor hovering. **(b)** Quadrotor flight. **(c)** Fast obstacle avoidance.

Figure 2.11 – Quadrotor flight demonstrations given in the context of the DARPA FLA project.

First Quadrotor Hovering Demo. I demonstrated the first closed-loop quadrotor hovering using an event camera for state estimation in January 2017, at a DARPA training site in Florida. The demonstrator consisted of a quadrotor equipped with a downward-facing event camera, hovering over a carpet featuring salient features (Fig. 2.11a). I implemented my event-based visual odometry algorithm EVO [150] on the quadrotor's computer and used it to perform state estimation in real-time, and further used to compute control inputs to stabilize the quadrotor.

Autonomous Quadrotor Flight with Events, Frames and IMU. On November 2017, I demonstrated the first closed-loop complex quadrotor flight using an event camera for state estimation. I implemented my visual-inertial odometry algorithm for event cameras [149, 164] to provide state estimates in real-time. In contrast to the first,

hovering demo, the quadrotor was now flying in a natural environment (an army training base in Pennsylvania), with very few features to track. Nevertheless, the quadrotor was able to fly a (predefined) square trajectory of about 50 m, with a total drift of about 2 m (Fig. 2.11b).

Event-based, Moving Obstacle Avoidance with a Quadrotor. In June 2018, I presented a system allowing a quadrotor to detect fast moving obstacles using an event camera, to trigger an avoidance maneuver with low latency (Fig. 2.11c). I implemented a simple system to perform obstacle detection, inspired by [114] – replacing the costly optimization scheme used for motion compensation in [114] by a more efficient scheme using the quadrotor gyroscope to compensate for rotational motion[1]. More details are available in the publication that stemmed from that demonstration [49].

2.5.2 Unrelated Contributions

(U1) Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza. "Benefit of Large Field-of-View Cameras for Visual Odometry". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2016

In this paper, we investigated the impact of the camera field of view (FoV) – given a fixed sensor size – on the accuracy of visual odometry. We found that the optimal field of view depends on the type of scene: navigation in confined environments benefits from large FoV, while navigation in an urban canyon (driving) will benefit from a smaller FoV. We also released the MultiFOV synthetic datasets, containing images and depth maps from multiple FoV (synthetic) cameras moving in different environments, along the same trajectories. Finally, I developed and released a patch for Blender to allow it to simulate wide FoV cameras such as omnidirectional cameras[2].

Figure 2.12 – Images from our MultiFOV synthetic datasets [207]. From left to right: simulated pinhole, fisheye and catadioptric lenses.

[1] A video of the system running is available at: https://youtu.be/qwRZWyGXEnU.
[2] Available at: https://github.com/uzh-rpg/rpg_blender_omni_camera

3 Future Directions

Event-based vision is an emerging technology in the era of mature frame-based camera hardware and software. Research on event cameras is still in early stages and has not reached yet the same maturity level as conventional computer vision. Yet, considerable progress has been made in the last few years, both in terms of hardware and software, which has clearly shown the potential of event cameras to overcome some of the limitations of frame-based cameras, reaching new scenarios previously inaccessible.

While event cameras have numerous advantages (high temporal resolution, high dynamic range, low latency and low power), this thesis has mostly focused on exploiting the ability of event cameras to better perceive the environment (*i.e.* high temporal resolution and dynamic range), thus not fully exploiting the low latency and low power that these sensors offer. In particular, one important challenge is to develop systems that are natively event-based, thus allowing to fully exploit the low latency of event cameras. I now outline a few promising research directions.

Asynchronous Neural Networks and Hardware. Because events are generated asynchronously, Spiking Neural Networks (SNNs) [176] seem like a natural design choice for event-based processing. In these architectures, neurons do not fire at each propagation cycle, but only when the neuron state reaches a specific value, asynchronously. However, due to the binary response function (spike generation) modelling their dynamics, it is still not clear how to perform backpropagation for these networks [182]. Nevertheless, dedicated hardware for spiking neural network inference is being developed, such as CAVIAR [176], SpiNNaker [55], IBM TrueNorth [5], and DYNAP [142]. While providing massive parallel compute power and low latency updates, these systems require very little power. Another line of work has proposed to relax the assumptions on SNNs' binary dynamics and apply conventional machine learning architectures to asynchronous data [201, 128, 170]. Asynchronous neural networks have achieved some promising, early success in synthetic and/or simple scenarios [7]. However, it is still not clear to what extent they can be applied to more challenging, real-world tasks.

Better sensor modelling. All vision sensors are noisy because of the inherent shot noise in photons and from transistor circuit noise, and they also have non-idealities. This situation is especially true for event cameras, where the process of quantization of brightness change information is complex and has not been completely characterized. While machine learning techniques can help deal with the noise implicitly [155], we are far from having a model that can predict event camera noise statistics under arbitrary illumination and biasing conditions. Solving this challenge would lead to better estimation methods – and in particular would help improve sensor fusion methods between event cameras and other sensors.

More data. Machine learning techniques have enabled considerable progress in event processing, and its benefits scale with the amount of training data. Despite recent efforts to increase this amount (including the contributions of this thesis), it still falls far short of the quantity of visual data from frame-based sensors. I envision a few ways this issue could be addressed. First, learning from simulated events would allow to massively increase the amount of training data. Despite some early success in simulation-to-real transfer (Paper F1), it is not clear in which conditions and tasks this is possible. Second, developing efficient transfer learning techniques could allow to reuse knowledge distilled from frame-based data and apply it to the event domain, thus reducing the amount of event data necessary. Early works in that direction include [109, 155], but there is considerable room for further exploration. Finally, self-supervised learning (*e.g.* using a photometric consistency loss from grayscale frames [213]), and unsupervised learning (*e.g.* using a loss based on contrast maximization [214, 59, 204]) are promising research directions, which could allow to reduce the demand for labels, which are particularly difficult to collect for event data.

Novel applications. While event cameras have been applied successfully to a large variety of tasks [57], I believe that their unique principle of operation and outstanding properties could open the door to novel exciting problems, maybe outside of the traditional computer vision community. For example, [26] recently applied event cameras to the problem of star tracking with low latency and low power, [127] used them for tactile sensing, and the startup Prophesee applied them to high speed counting of objects and visual detection of vibrations[1].

Beyond event cameras. Event cameras can be seen as an entry point for more efficient, near-sensor processing, such that only high-level, non-redundant information is transmitted, thus reducing bandwidth, latency and power consumption. This could be done by pairing an event camera with hardware on the same sensor device (*e.g.* the

[1] https://www.prophesee.ai/2019/02/14/imago-prophesee/

Speck sensor[2]), in a similar way than ganglion cells in the brain are directly connected to the retina. Alternative bio-inspired imaging sensors have also recently emerged, such as cellular processor arrays (SCAMP [23]), in which every pixel has a processor that allows to perform several types of computations with the brightness of the pixel and its neighbors. While these sensors can be viewed in principle as a generalization of event cameras, both the hardware and software for these sensors is significantly less mature than event cameras, and it is not clear yet whether they will live up to their promises. As Andrew Davison put it in his "FutureMapping" paper [35]: *the key to efficient processing which is both fast and consumes little power is to divide computation between a large number of relatively low clock-rate or otherwise simple cores, and to minimise the movement of data between them*. Cellular processor arrays, coupled with specialized hardware such as graph processors[3], thus might be an exciting step in that direction.

[2] https://www.speck.ai/

[3] for example, Graphcore: https://www.graphcore.ai/

A ESIM: an Open Event Camera Simulator

Reprinted, with permission, from:

H. Rebecq, D. Gehrig, and D. Scaramuzza. "ESIM: an Open Event Camera Simulator". In: *Conf. on Robotics Learning (CoRL)*. 2018

ESIM: an Open Event Camera Simulator

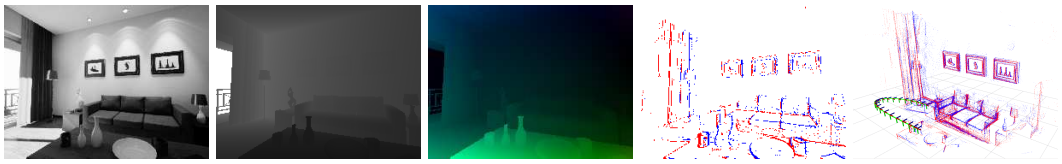
Henri Rebecq, Daniel Gehrig and Davide Scaramuzza

Abstract — Event cameras are revolutionary sensors that work radically differently from standard cameras. Instead of capturing intensity images at a fixed rate, event cameras measure changes of intensity *asynchronously*, in the form of a stream of *events*, which encode per-pixel brightness changes. In the last few years, their outstanding properties (asynchronous sensing, no motion blur, high dynamic range) have led to exciting vision applications, with very low-latency and high robustness. However, these sensors are still scarce and expensive to get, slowing down progress of the research community. To address these issues, there is a huge demand for cheap, high-quality synthetic, labeled event datasets for algorithm prototyping, deep learning and algorithm benchmarking. The development of such a simulator, however, is not trivial since event cameras work fundamentally differently from frame-based cameras. We present the first event camera simulator that can generate a large amount of reliable event data. The key component of our simulator is a theoretically sound, adaptive rendering scheme that only samples frames when necessary, through a tight coupling between the rendering engine and the event simulator. We release an open source implementation of our simulator.

We release ESIM as open source: http://rpg.ifi.uzh.ch/esim.

A.1 Introduction

Vision sensors, such as cameras, have become the sensor of choice for robotic perception. Yet, despite numerous remarkable achievements in the last few years, our robots are still limited by their sensing capabilities. In particular, low-latency perception, HDR (High Dynamic Range) scenes, and motion blur remain a profound issue for perception



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

Figure A.1 – An example of the output of our simulator in a room environment. Our simulator also provides inertial measurements (not shown here). (a) Image. (b) Depth map. (c) Motion field (Color-coded). (d) Events (Positive and Negative). (e) Point cloud (3D events) + Camera trajectory

systems that rely on standard cameras. Event cameras [98] are novel sensors that work fundamentally differently from standard cameras. Instead of capturing intensity images *synchronously*, i.e. at regular time intervals, event cameras measure changes of intensity *asynchronously*, in the form of a stream of *events*, which encode per-pixel brightness changes, with very low latency (1 microsecond). Because of their outstanding properties (low latency, HDR, no motion blur), event cameras promise to unlock robust and high-speed perception in situations that are currently not accessible to standard cameras: for example, balancing a pen [29], tracking features in the blind time between two frames [64], or flying a quadcopter in a dark room [164].

Research on event cameras is still in its infancy and we believe that the progress in this area has been slowed down by several issues. First, event cameras are rare and expensive sensors: to this date, there is only one company selling event cameras [1] and a recent DAVIS346 sensor [18] costs six thousand dollars, which only a tiny fraction of research groups can afford. Second, commercially available event camera hardware is still at the protoype level and suffers from many practical limitations, such as low resolution (346×260) for the DAVIS346), poor signal-to-noise ratio and complex sensor configuration, which requires expert knowledge. These issues make it difficult to understand the true potential of event cameras for robotics.

In parallel, the rise of deep learning has led to an explosion of the demand for data. To meet this demand, an unprecendented number of simulators (CARLA [44], Microsoft Airsim [179], UnrealCV [144]) has been proposed for standard cameras. Yet, the community is still missing an accurate, efficient, and scalable event camera simulator. Building such a simulator is not trivial because event cameras work fundamentally differently from conventional cameras. While standard camera samples the (time-varying) visual signal at a discrete, fixed sampling rate (in a simulator, this corresponds to *rendering* an image), event cameras perform some form of level-crossing sampling: an event is generated when the signal itself reaches some intensity bounds. As a consequence, the amount of data to be simulated is proportional to the amount of

[1] https://inivation.com/buy/

motion in the scene. This calls for a novel simulator architecture to accurately simulate this varying data rate, which is precisely the goal of this work. Specifically, our contributions are:

- a novel event camera simulator architecture that tightly couples the event simulator and the rendering engine to allow for an accurate simulation of events through a novel, adaptive sampling scheme (Section A.3),
- a quantitative evaluation of our proposed adaptive sampling scheme against fixed-rate sampling, both in accuracy and efficiency (Section A.4),
- validation of the simulated events by training a neural network for optic flow prediction, which generalizes well to real environments (Section A.5),
- an efficient, open-source implementation of our proposed event camera simulator.

A.2 Related Work

In the last few years, a number of event camera datasets and simulators have been introduced [2]. In this section, we briefly review the most important ones and their specific application scenarios. We then turn to the works on simulation of an event camera.

A.2.1 Event Camera Datasets

The range of tasks that have been addressed with event cameras can be roughly split into two: low-level vision tasks and high-level vision tasks.

Low-level Vision A number of event datasets has been specifically designed for low-level vision tasks such as visual (inertial) odometry [196, 124, 212], optic flow estimation [11, 166] or depth from stereo [202, 212]. While the scope of the early datasets was severely restricted (for example, [11] contains a dozen sequences of a en event camera looking at a calibration pattern, each sequence during less than ten seconds), two recent datasets have recently stood out. [124] was introduced as a benchmark for event-based visual (inertial) odometry. It features 25 sequences recorded with a DAVIS240C sensor [18], and each sequence contains events, frames, and inertial measurements, as well as ground truth camera poses from a motion capture system. However, it does not contain ground truth depth or optic flow, and most of the scenes were captured indoors, targeting AR/VR scenarios. The MVSEC dataset [212] contains 10 sequences from a stereo DAVIS346 sensor, which was mounted on various vehicles (car, quadcopter and

[2] An extensive list can be found at: https://github.com/uzh-rpg/event-based_vision_resources#datasets-and-simulators-sorted-by-topic

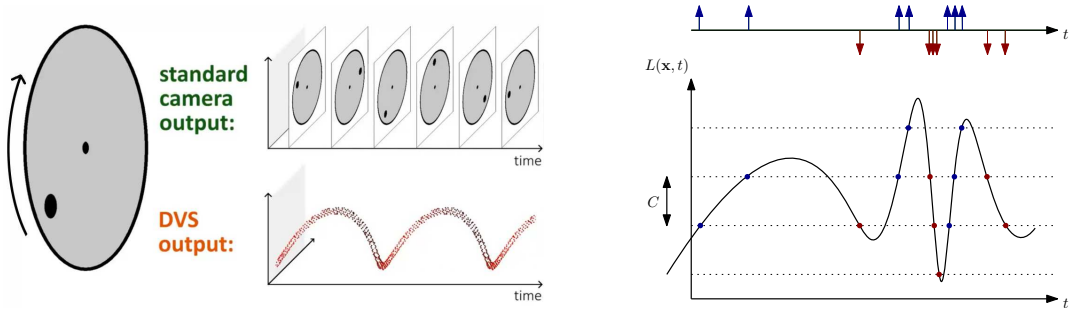
motorbike). Images from a standard stereo camera setup, ground truth camera poses, depth maps (derived from a LIDAR) and optic flow maps [213] are also provided. However, the dataset essentially targets visual navigation scenarios for automotive, and the different sensing modalities are only roughly calibrated together (sometimes manually).

Recognition and Scene Understanding In parallel, using event cameras for high-level tasks has motivated the release of several datasets. Most of these datasets target the task of object recognition [132, 178, 184], or gesture recognition [7]. In particular, [132] presented semi synthetic, event-based versions of the MNIST [90] and Caltech101 datasets. To convert these frame-based datasets to events, they mounted an event camera on a pan-tilt motor, facing a screen displaying standard frames, and recorded the events generated by moving the sensor slightly in different directions using the motor. Finally, the recent DDD17 [15] and N-CARS [184] datasets target automotive applications. While N-CARS [184] specifically focuses on car recognition, DDD17 [15] provides the vehicle speed, GPS position, driver steering, throttle, and brake captured from the car's on-board diagnostics interface. However, it does not contain semantic labels such as object bounding boxes, nor ground truth data for low-level vision tasks such as depth maps or optical flow.

A.2.2 Event Camera Simulators

A simple event camera simulator was presented in [78] and implemented as a plugin for the Gazebo simulation software [83]. However, as acknowledged by the authors, this simulator does not implement the principle of operation of an event camera. Instead, it merely thresholds the difference between two successive frames to create edge-like images that resemble the output of an event camera. The asynchronous and low-latency properties of the sensor is not simulated, as all the events from a pair of images are assigned the same timestamp. To the best of the author's knowledge, [124] and [95] are the only simulators that attempt to simulate an event camera accurately. To simulate an event camera pixel, both works rely on rendering images from a 3D scene at a very high framerate, using either Blender [3] [124], or a custom rendering engine [95]. While this approach allows to mimic the asynchronous output of an event camera, it cannot simulate event data reliably when the brightness signal varies more rapidly than the arbitrarily chosen rendering framerate can handle. A comparison between our approach and this fixed-rate sampling approach [124, 95] is given is Section A.4.

[3] https://www.blender.org/



(a) (b)

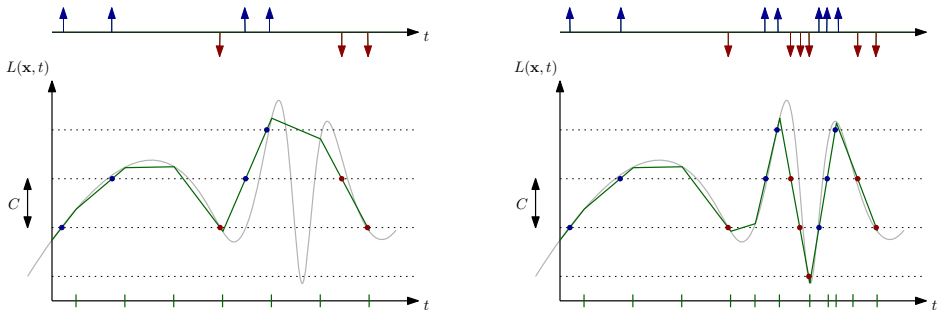
Figure A.2 – Principle of operation of an event camera. Fig. A.2a: comparison of the output of a standard camera and an event camera when viewing a spinning disk with a black circle. The standard camera outputs frames at a fixed rate, thus sending redundant information when no motion is present in the scene. In contrast, event cameras are data-driven sensors that respond to brightness changes with microsecond temporal resolution. Fig. A.2b: Zoom on a pixel $\mathbf{x}$: a positive (resp. negative) event (blue dot, resp. red dot) is generated whenever the (signed) brightness change exceeds the contrast threshold C. Observe how the event rate grows when the signal changes rapidly.

A.3 Proposed Event Simulation Strategy

Unlike standard cameras which capture intensity information from the scene synchronously, in the form of frames, event cameras sample the visual signal asynchronously, i.e., independently for each pixel. Specifically, each pixel of an event sensor will produce an event if the brightness change since the last event fired has reached a given threshold C. This principle is illustrated in Fig. A.3. At this point, it is worth clarifying what we mean exactly by "brightness". Vision sensors measure some function of the radiant flux (or intensity) of light falling per unit area of the sensor, which is referred to as the *irradiance* E. Event cameras operate in the log domain, which allows them to achieve a high dynamic range: instead of measuring changes of irradiance E, they measure changes of log-irradiance $\log E$. Throughout the rest of the paper, we will use the term *brightness* to denote the log-irradiance, and denote it: $\mathcal{L} = \log E$.

A.3.1 Event Simulation from Adaptive Sampling

To simulate an event camera, one would need access to a continuous representation of the visual signal at every pixel, which is not accessible in practice. To circumvent this problem, past works [124] have proposed to sample the visual signal (i.e. sample frames) synchronously, at a very high framerate, and perform linear interpolation between the samples to reconstruct a piecewise linear approximation of the continuous underlying visual signal, which is used to emulate the principle of operation of an event camera (Fig. A.3a). We take the same general approach for simulating events, sampling



(a) Uniform Sampling [124]. **(b)** Adaptive Sampling (Proposed).

Figure A.3 – Comparison of uniform sampling (Fig. A.3a) versus adaptive sampling (Fig. A.3b). 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, our proposed adaptive sampling strategy extracts more samples in the fast-varying region, thus successfully simulating events (as can be seen by comparing the simulated events with the ground truth events shown in Fig. A.2b).

the visual signal (by means of *rendering* images along the camera trajectory), with a key difference: instead of choosing an arbitrary rendering framerate, and sampling frames uniformly across time at the chosen framerate, we propose to sample frames *adaptively*, adapting the sampling rate based on the predicted dynamics of the visual signal.

An adaptative sampling of the visual signal requires a tight integration between the rendering engine, and the event simulator. In the next section, we describe our simulator architecture (Section A.3.2) and detail the adaptive sampling scheme we use to ensure that events are generated accurately (Section A.3.3).

A.3.2 Simulator Architecture

The architecture of ESIM is illustrated in Fig. A.4. It tightly couples the rendering engine and the event simulator, which allows the event simulator to adaptively query visual samples (i.e. frames) based on the dynamics of the visual signal (Section A.3.3). Below, we introduce the mathematical formalism we will use to describe, without loss of generality, the various components involved in our simulator. The details of possible implementations of each of these components are given in the appendix (Sections A.6 and A.6).

Sensor Trajectory The sensor trajectory is as a smooth function $\mathcal{T}$ that maps every time t to a sensor pose, twist (angular and linear velocity), and acceleration. Following the notation of [52], we denote the pose of the sensor expressed in some inertial frame W by $T_{WB} \in SE(3)$, where $SE(3)$ denotes the group of rigid motion in 3D, the twist by $\xi(t) = ({}_W\mathbf{v}(t), {}_B\boldsymbol{\omega}_{WB}(t))$, and its acceleration (expressed in the inertial frame W) by

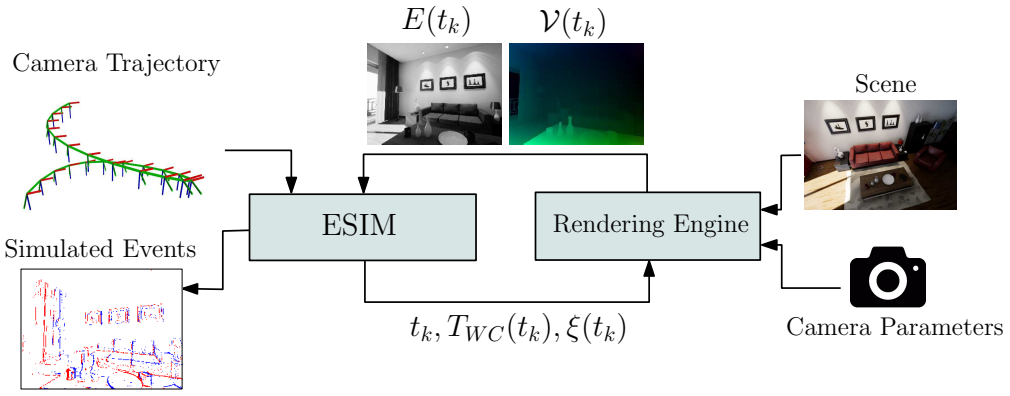


Figure A.4 – ESIM relies on a tight coupling with the rendering engine to generate events accurately. At time t_k, ESIM samples a new camera pose $T_{WC}(t_k)$ and camera twist $\xi(t_k)$ from the user-defined trajectory and passes them to the rendering engine, which, in turn, renders a new irradiance map $E(t_k)$ and a motion field map $\mathcal{V}(t_k)$. The latter are used to compute the expected brightness change (Eq. (A.1)), which is used to choose the next rendering time t_{k+1}.

${}_B\mathbf{a}(t)$.

Rendering Engine The renderer is a function $\mathcal{R}$ which maps any time t to a rendered image (or irradiance E) of the scene at the current sensor pose. The renderer is parameterized by the environment $\mathcal{E}$, the sensor trajectory within this environment $\mathcal{T}$, and the sensor configuration Θ. $\mathcal{E}$ controls the geometry of the scene, as well as its dynamics (it can be for example a still room, a city with moving vehicles, etc.). $\mathcal{T}$ represents the camera trajectory in the environment, which may be generated online based on a dynamical model of the robot and a set of control inputs, or precomputed offline. Θ represents the configuration of the vision sensor simulated, which includes the camera intrinsics (sensor size, focal length, distortion parameters) and the camera extrinsics (pose of the camera with respect to the sensor body, T_{BC}). It also includes sensor-specific parameters (for example, the contrast threshold C for an event camera, the exposure time for a standard camera, etc.). For reasons that will appear clearly in Section A.3.3, the renderer additionally provides the motion field $\mathcal{V}(t)$. Figure A.1 shows an example of the output of our simulator. Additional examples can be found in the supplementary material.

A.3.3 Adaptive Sampling Strategies

Based on Brightness Change Under the assumption of lambertian surfaces, a first-order Taylor expansion of the brightness constancy assumption [188] yields:

$$\frac{\partial \mathcal{L}(x;t_k)}{\partial t} \simeq -\left\langle \nabla \mathcal{L}(x;t_k), \mathcal{V}(x;t_k) \right\rangle \tag{A.1}$$

where $\nabla\mathcal{L}$ is the gradient of the brightness image, and $\mathcal{V}$ the motion field. In other words, the expected (signed) brightness change at pixel $\mathbf{x}$ and time t_k during a given interval of time Δt is: $\Delta\mathcal{L} \simeq \frac{\partial\mathcal{L}(x;t_k)}{\partial t}\Delta t$. We want to ensure that for every pixel $\mathbf{x} \in \Omega$, $|\Delta\mathcal{L}| \leq C$ (i.e. the brightness change is bounded by the desired contrast threshold C of the simulated event camera). It can be shown easily from Eq. A.1 that choosing the next rendering time t_{k+1} as follows allows for this:

$$t_{k+1} = t_k + \lambda_b C \left|\frac{\partial\mathcal{L}}{\partial t}\right|_m^{-1} \tag{A.2}$$

, where $\left|\frac{\partial\mathcal{L}}{\partial t}\right|_m = \max_{x\in\Omega}\left|\frac{\partial\mathcal{L}(x;t_k)}{\partial t}\right|$ is the maximum expected rate of brightness change across the image plane Ω, and $\lambda_b \leq 1$ is a parameter that controls the trade-off between rendering speed and accuracy. Essentially, Eq. (A.2) imposes that the rendering framerate grows proportionally to the maximum expected absolute brightness change in the image. Note that this criterion was derived from Eq. (A.1), whose correctness depends on the validity of the assumed lambertian surfaces, brightness constancy, and linearity of local image brightness changes. Therefore, it does not provide a strict guarantee that the signal was correctly sampled. To account for these unmodeled non-linear effects, we can enforce a slightly stricter criterion than the one imposed by Eq.(A.2) by choosing $\lambda_b < 1$ in Eq.(A.2). In our experiments, we used $\lambda_b = 0.5$.

Based on Pixel Displacement An alternative, simpler strategy to perform adaptive sampling is to ensure that the maximum displacement of a pixel between two successive samples (rendered frames) is bounded. This can be achieved by choosing the next sampling time t_{k+1} as follows:

$$t_{k+1} = t_k + \lambda_v \left|\mathcal{V}(x;t_k)\right|_m^{-1} \tag{A.3}$$

, where $|\mathcal{V}| = \max_{x\in\Omega}|\mathcal{V}(x;t_k)|$ is the maximum magnitude of the motion field at time t_k, and $\lambda_v \leq 1$. As before, we can choose λ_v to mitigate the effect of unmodeled non-linear effects. In our experiments, we used $\lambda_v = 0.5$.

A.3.4 Noise simulation and Non Idealities

We also implement a standard noise model for event cameras, based on an observation originally made in [98] (Fig. 6): the contrast threshold of an event camera is not constant, but rather normally distributed. To simulate this, at every simulation step, we sample the contrast according to $\mathcal{N}(C;\sigma_C)$, where σ_C controls the amount of noise, and can be set by the user. Furthermore, we allow setting different positive (respectively

negative) contrast thresholds C_+ (respectively C_-) to simulate a real event camera more accurately, where these values can be adapted independently with the right set of electronic biases. Additional noise effects, such as spatially and temporally varying contrast thresholds due to electronic noise, or limited bandwidth of the event pixels, are still open research questions.

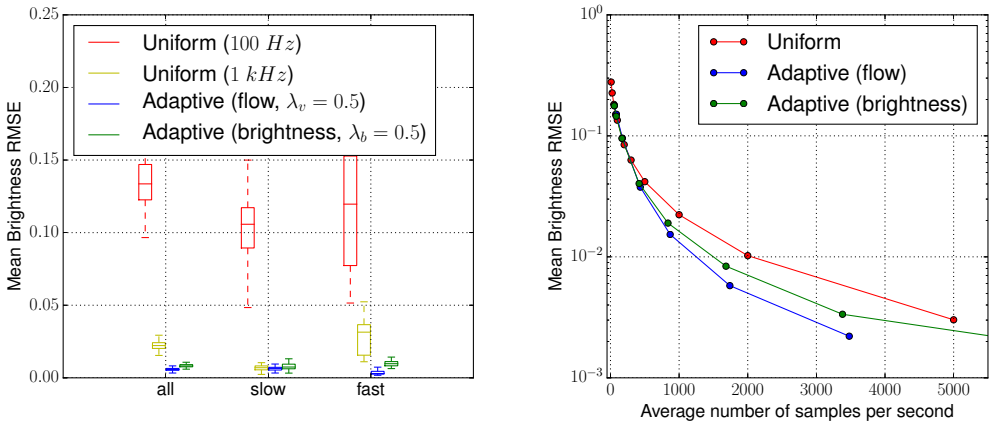
A.4 Experiments

A.4.1 Accuracy and Efficiency of Adaptive Sampling

In this section, we show that our adaptive sampling scheme allows achieving consistent event generation, independently of the speed of the camera. Furthermore, we show that the proposed adaptive sampling scheme is more efficient than a fixed-rate sampling strategy [124, 95]. Finally, we compare the two variants of adaptive sampling proposed in Section A.3.3, through Eqs. (A.2), (A.3).

Experimental Setup To quantify the influence of the proposed adaptive sampling scheme, compared to a uniform sampling strategy, we simulate the same scene (Figure A.8), under the same camera motion, with different sampling strategies, and compare the results. To quantify the accuracy of event simulation, one might directly compare the generated events and compare them against "ground truth" events. However, directly comparing event clouds is an ill-posed problem: there exists no clear metric to compute the similarity between two event clouds. Instead, we evaluate the accuracy of a sampling strategy by comparing the estimated brightness signal with respect to a ground truth brightness signal obtained by oversampling the visual signal (at 10 kHz). In order to obtain simulation data that matches the statistics of a real scene, we simulate random camera rotations in a scene simulated using a real panoramic image taken from the Internet, which features various textures, including low-frequency and high-frequency components.We simulated a 5 second sequence of random camera rotation at various speeds, up to 1,000 deg/s, yielding a maximum motion field of about 4,500 px/s on the image plane.

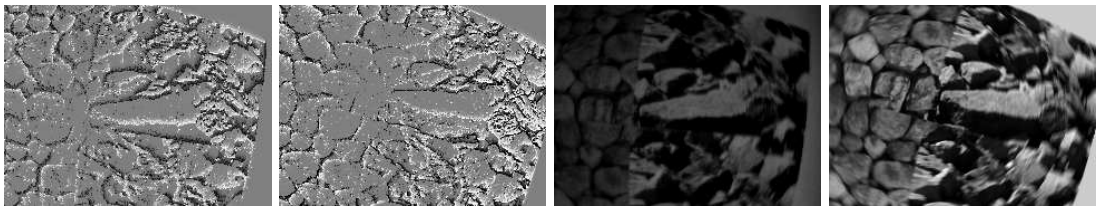
Evaluation Modalities We simulate the exact same scene and camera motion, each time using a different sampling scheme. For each simulated sequence, we consider a set of $N = 100$ pixel locations, randomly sampled on the image plane (the same set is used for all runs). We split the evaluation sequence in three subsets: a 'slow' subset (mean optic flow magnitude: 330 px/s), a 'fast' subset (mean optic flow magnitude: 1,075 px/s), and a subset 'all', containing the entire sequence (mean optic flow magnitude: 580 px/s). For every pixel in the set, we perform linear interpolation between the brightness samples, and compute the RMSE of the reconstructed brightness signal with respect to the ground truth.



(a) **(b)**

Figure A.5 – Evaluation of the accuracy and efficiency of the proposed sampling scheme against a fixed-rate sampling scheme. Left: mean RMSE of the reconstructed brightness signal as a function of the number of samples for each sampling strategy, computed on the entire evaluation sequence. Right: mean RMSE of the reconstructed brightness signal as a function of the number of samples for each sampling strategy, computed on the entire evaluation sequence.

Results Figure A.5a shows the distribution of the RMSE values for three different sampling schemes: uniform sampling at respectively 100 Hz and 1 kHz, and our proposed adaptive sampling schemes (flow-based and brightness-based) with $\lambda = 0.5$. Sampling at a "low" framerate (100 Hz) yields the highest error, even for the 'slow' sequence. Increasing the sampling rate to 1 kHz leads to a significant error reduction on the 'slow' sequence, but is not sufficient for faster motions such as those found in the 'fast' subsequence, yielding high error overall. Our proposed adaptive schemes do not require any parameter tuning, and perform consistently well on all three subsets, outperforming the uniform sampling strategy by a large margin, especially during fast motions. Figure A.5b shows the mean RMSE as a function of the number of samples (i.e. the number of frames rendered). It shows that, for a given computation budget, the proposed adaptive sampling strategies have a lower error. For example, the adaptive sampling strategy based on optic flow requires on average 1,260 samples per second to achieve an RMSE ≤ 0.01 while the uniform sampling strategy requires on average 2,060 samples per second to achieve the same error bound, which leads to a 60% decrease of the simulation time. Overall, these experiments show that the adaptive sampling strategy based on maximum optic flow achieves a higher accuracy than the one based on brightness for the same computational budget. Following this analysis, we now use the adaptive sampling strategy based on maximum optic flow with $\lambda = 0.5$ in all our next experiments.



(a) Real Events **(b)** Simulated Events **(c)** Real Frame **(d)** Simulated Frame

Figure A.6 – Qualitative comparison of real events and frames (Figs. A.6a, A.6c) with simulated events and frames (Figs. A.6b, A.6d) generated during a high-speed motion. While some differences are visible between the real and simulated events, the locations of the simulated events mostly correspond to the real events. A video version of this comparison is available at: https://youtu.be/VNWFkkTx4Ww

A.4.2 Qualitative Comparison with Real Data

To show qualitatively that our simulated event camera output resembles that of a real event camera, we show a side-by-side comparison of a real dataset from the Event Camera Dataset [124], and a simulated reproduction of the dataset. Details on how we reproduced the scene in simulation are provided in the appendix. Fig. A.6 shows a side-by-side comparison of simulated data (events and frames) with real data. The events are visualized using an exponential time surface [88] with exponential decay $\tau_c = 3.0\ ms$.

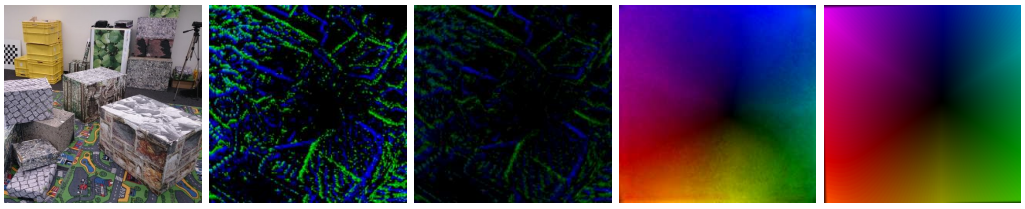
Discussion As Fig. A.6 shows, the main moving structures are correctly captured in the simulated events. However, small differences between the real and simulated event streams show that the simulation is not perfect. There are two main sources of error. First, the simulated and real setup do not match exactly: (i) the contrast thresholds and noise level of the real event camera are not known and were only guessed from data, and (ii) the poster position and texture do not exactly match the real poster used in the dataset. Second, our simulator features a simple noise model (A.3.4), which does not account for more complex noise sources in a real event camera coming from non-idealities in the electronics. For example, at high speeds, it has been observed that the refractory period of the event pixels may cause the loss of events [19], which is not modelled by our implementation.

A.5 Example Application: Learning Optic Flow

Affine Optical Flow Dataset In this section we explore an application of the event camera simulator which is supervised learning of globally affine optic flow. We generate a large dataset (10,000 sequences with 0.04 s duration each) of event data simulated by applying random affine transformations to a set of real images (from the Internet). As

in [43], we select the random affine parameters such that the resulting flow distribution matches that of the Sintel dataset. We train a neural network that regresses the dense motion field from the events, using the ground truth optic flow maps as labels. Our network architecture is the same as FlowNet Simple [43] and we train the network by minimizing the end-point-error (EPE) between predicted and ground truth optic flow. Similarly, we perform online geometric augmentation (ie, random rotation and scaling) during training. As event-based input to the network, we convert each event sequence to a tensor with 4 channels as proposed in [213] and report the validation and test EPE only on pixels where events were fired. Comparisons of predicted and ground truth optic flow from the validation set are shown in Fig. A.7 and Fig. A.10 in the appendix. The network achieves a final EPE of about 0.3 pixels on the validation set.

Sim2Real Transfer We evaluated the trained network directly on *real event data* without any fine-tuning. Specifically, we use the *boxes_rotation* sequence from the Event Camera Dataset [124], for which the optic flow can be approximated as an affine flow field. We use the IMU data to measure angular velocity which allows us to compute the ground truth motion field for testing. The network performs well, both qualitatively and quantitatively: the EPE being close to 4.5 pixels on the real event dataset. A comparison of predicted and ground truth optic flow can be found in Fig. A.7 (bottom row) and A.10 in the appendix. It is important to stress that the network was not trained on real event data, nor trained on scenes that look similar to the scene used for testing. We therefore showed that our proposed simulator is useful to train networks that will perform relatively well in the real world. In practice, such a network would be trained on a large amount of synthetic data and fine-tuned on a smaller amount of real event data.



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

Figure A.7 – Results from learning globally affine flow from simulated event data and ground truth optic flow. The top row shows the network output on the validation set (simulated) and the bottom row on the test set (real). The columns show a preview of the scenes (a), inputs to the network, comprising event time surfaces and event count frames (b-c) [213], and the predicted (d) and ground truth (e) optic flow.

A.6 Conclusions

We presented the first event camera simulator that can simulate events reliably and efficiently, and that can simulate arbitrary camera trajectories in arbitrary 3D scenes. The key component of our simulator is a theoretically sound, adaptive rendering scheme that only samples frames when necessary, through a tight coupling between the rendering engine and the event simulator. We release an open source implementation of our simulator. We believe our simulator will be of considerable interest for the vision community; allowing, for the first time, to collect large amounts of event data with ground truth, opening the door to accurate benchmarking of existing computer vision methods that operate on event data, as well as to supervised learning approaches with event cameras.

Appendix: Implementation Details

This section describes the implementation choices we made in the open source implementation of ESIM, available at: http://rpg.ifi.uzh.ch/esim.

Sensor Trajectory

To implement the trajectory function $\mathcal{T}$, we use a continuous time representation for the camera trajectory, based on splines in SE(3) ([56]). The user can either decide to generate a random trajectory or load a trajectory from a file containing a discrete set of camera poses. In the first case, we sample randomly a fixed number of camera poses in SE(3) and fit a spline to the poses. In the second case, a spline is fitted to the waypoint poses provided by the user. The spline representation allows us to sample poses continuously in time. Furthermore, splines provide analytical formulas for the sensor twist ξ and acceleration ${}_B\mathbf{a}$ along the trajectory, from which we can compute the motion field (A.6) and simulate inertial measurements (A.6).

Rendering Engines

Our open-source implementation comes with multiple rendering engines that have different characteristics. A full list is available on the simulator website [4]. Here, we briefly describe the two most important ones. The first rendering backend is based on pure OpenGL code, and is designed to simulate a large amount of data fast, at the price of simulating only simple 3D scenes with textures. The second is based on Unreal Engine, and is designed to simulate photorealistic data; however, it it relatively slow compared to the former.

OpenGL Rendering Engine The first available rendering engine is based on pure OpenGL, i.e. it is based on rasterization, implemented on the GPU. A vertex shader projects a user-provided 3D point cloud in the desired camera frame, and projects it on the image plane. A fragment shader evaluates the intensity to assign to each pixel in the image plane, by sampling textures (provided UV mappings) provided with the 3D model. The current implementation supports loading textured 3D models from OBJ files or directly from Blender, and also supports loading camera trajectories directly from Blender. A rendering pass is extremely fast (1ms on a laptop), which allows the event simulation to run in real-time in most scenarios. An example of the output of this renderer can be found in the third row of Fig. A.9.

[4] http://rpg.ifi.uzh.ch/esim

Photorealistic Rendering Engine We additionally provide a rendering engine based on Unreal Engine [5], through the UnrealCV [144] project. We implemented a C++ client that interacts with an UnrealCV server and queries images (which we assimilate to irradiance maps) and depth maps along camera poses sampled along a user-defined trajectory. Furthermore, we supply a user interface which allows users to navigate through a scene interactively, set up waypoints by clicking on the screen, and then simulate a camera following an interpolated trajectory passing through these waypoints offline. While the generated event data is of very high quality (photorealistic), the event generation is quite slow (about 2 min computing time for one second of simulated data on a laptop computer). Example outputs of this renderer can be found in the first and second rows of Fig. A.9.

Computation of Motion Field

As anticipated in Section A.3, our adaptive sampling scheme requires the computation of the motion field on the image plane. Here we briefly describe how to this from the camera twist ξ and depth map of the scene Z. The motion field $\mathcal{V}$ is the sum of two contributions: one term comes from the camera egomotion itself (apparent motion of the scene with respect to the camera) $\mathcal{V}_{ego}$, and the second arises from moving objects in the scene $\mathcal{V}_{dyn}$. It follows that $\mathcal{V} = \mathcal{V}_{ego} + \mathcal{V}_{dyn}$. Below we give a closed-form expression for the computation of $\mathcal{V}_{ego}$. A similar expression can be derived for $\mathcal{V}_{dyn}$. $\mathcal{V}_{ego}$ can be computed from a depth map of the scene Z and the camera twist vector $\xi(t) = ({}_W\mathbf{v}(t), {}_B\boldsymbol{\omega}_{WB}(t))$ as follows [58]:

$$\mathcal{V}_{ego}(\mathbf{x}, t) = B(\mathbf{x}, t)\xi(t) \tag{A.4}$$

where

$$B = \begin{bmatrix} -Z^{-1} & 0 & uZ^{-1} & uv & -(1+u^2) & v \\ 0 & -Z^{-1} & vZ^{-1} & 1+v^2 & -uv & -u \end{bmatrix}$$

is the interaction matrix, with $Z := Z(\mathbf{x}, t)$ denoting the depth at pixel $\mathbf{x}$, time t, and $\mathbf{x} = (u, v)$ denoting a pixel location in the image plane, expressed in calibrated coordinates.

[5] https://www.unrealengine.com/en-US/what-is-unreal-engine-4

Additional Sensors

Our open-source simulator can also simulate additional sensors besides an event camera, namely a standard camera, and an inertial measurement unit. We now briefly describe how we simulate both of these sensors.

Standard Camera Our camera sensor takes as an input a sequence of irradiance maps rendered from the rendering engine $\mathcal{R}$, and produces a simulated camera image, by integrating the sequence of irradiance samples (with times t_k) collected over a user-defined exposure time ΔT. Specifically, a camera image I is generated as follows:

$$I = f\left(\sum_{t_k \geq t-\Delta t/2}^{t_k \leq t+\Delta t/2} E(t_k)(t_{k+1} - t_k)\right)$$

where f is the camera response function that maps integrated irradiance values to pixel intensities [129]. In our implementation, we used $f = Id$ which matches the linear response of the DAVIS active pixel sensor.

Inertial Measurement Unit We simulate an inertial measurement unit using additive Gaussian noise on the ground truth angular velocity ${}_B\boldsymbol{\omega}_{WB}$ (respectively acceleration ${}_B\mathbf{a}$) values sampled at 1 kHz from the trajectory $\mathcal{T}$. Additionally, we offset the angular velocity (and acceleration) measurements with a gyroscope and accelerometer bias. In real sensors the dynamics of these biases are well characterized by random walks. To simulate them we generate random walks at 200 Hz by adding Gaussian noise and fit a spline through these points to provide continuous-time values.

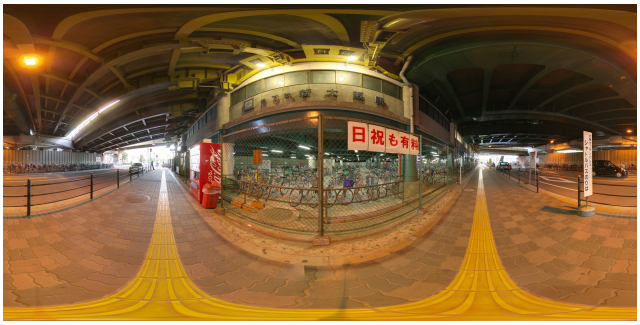
Appendix: Additional Details on the Experiments

Quantitative Comparison

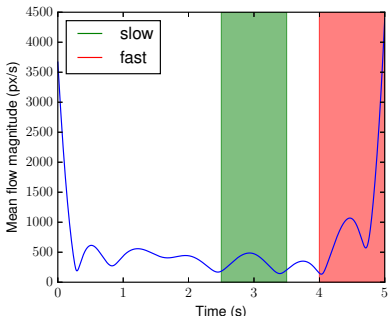
Figure A.8 shows a preview of the simulation scenario used for the quantitative evaluation of the adaptive sampling strategy proposed in Section A.4.1 the paper.

Qualitative Comparison

For the qualitative comparison with real event data presented in Section A.4.2, we replicated the *poster_6dof* sequence from the Event Camera Dataset [124], which consists of a planar poster attached to a wall. We recreated the scene in 3D by downloading high-resolution pictures of the original poster, and converting them from sRGB to a



(a) Panoramic map used to generate synthetic evaluation data.



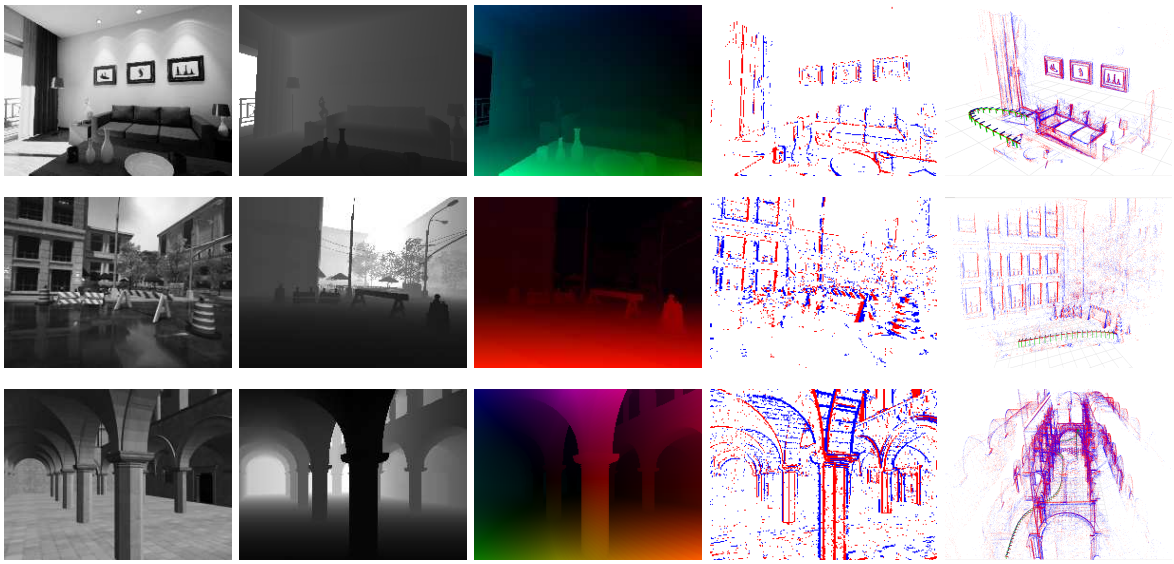
(b) Profile of the optic flow magnitude.

Figure A.8 – Simulation scenario used for quantitative evaluation of the adaptive sampling strategy proposed. Fig. A.8a: Real panoramic image that was used to simulate an event camera performing random rotations at various speeds. This scene was chosen because of it features both mostly uniform regions, but also regions with much finer, high frequency details such as the grillage close to the bikes. Fig. A.8b: average magnitude of the optic flow as a function of time. The red and green shaded areas mark the boundaries of the "slow" and "fast" subsets used mentioned in Section A.4.1.

linear color space. Furthermore, we used the provided camera calibration and trajectory (the trajectory was recorded with a motion capture system) to simulate the camera motion in the scene. Since neither the contrast thresholds of the event camera, nor the exposure time of the standard camera used in the dataset were disclosed, we guessed these values based on observing the event output, and chose respectively $C_+ = 0.55$, $C_- = 0.41$, with Gaussian noise on the thresholds $\sigma_C = 0.021$. For the exposure time of the standard camera simulated, we used $\Delta t = 7.5\ ms$.

Additional qualitative results for predicting affine flow

Figure A.10 shows additional qualitative examples of the machine learning application presented in Section A.5. We show results for both synthetic sequences from the evaluation set, and real event data.



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

Figure A.9 – (a) Image. (b) Depth map. (c) Optical Flow (Color-coded). (d) Events (Positive and Negative). (e) Point cloud (3D events) + Camera trajectory

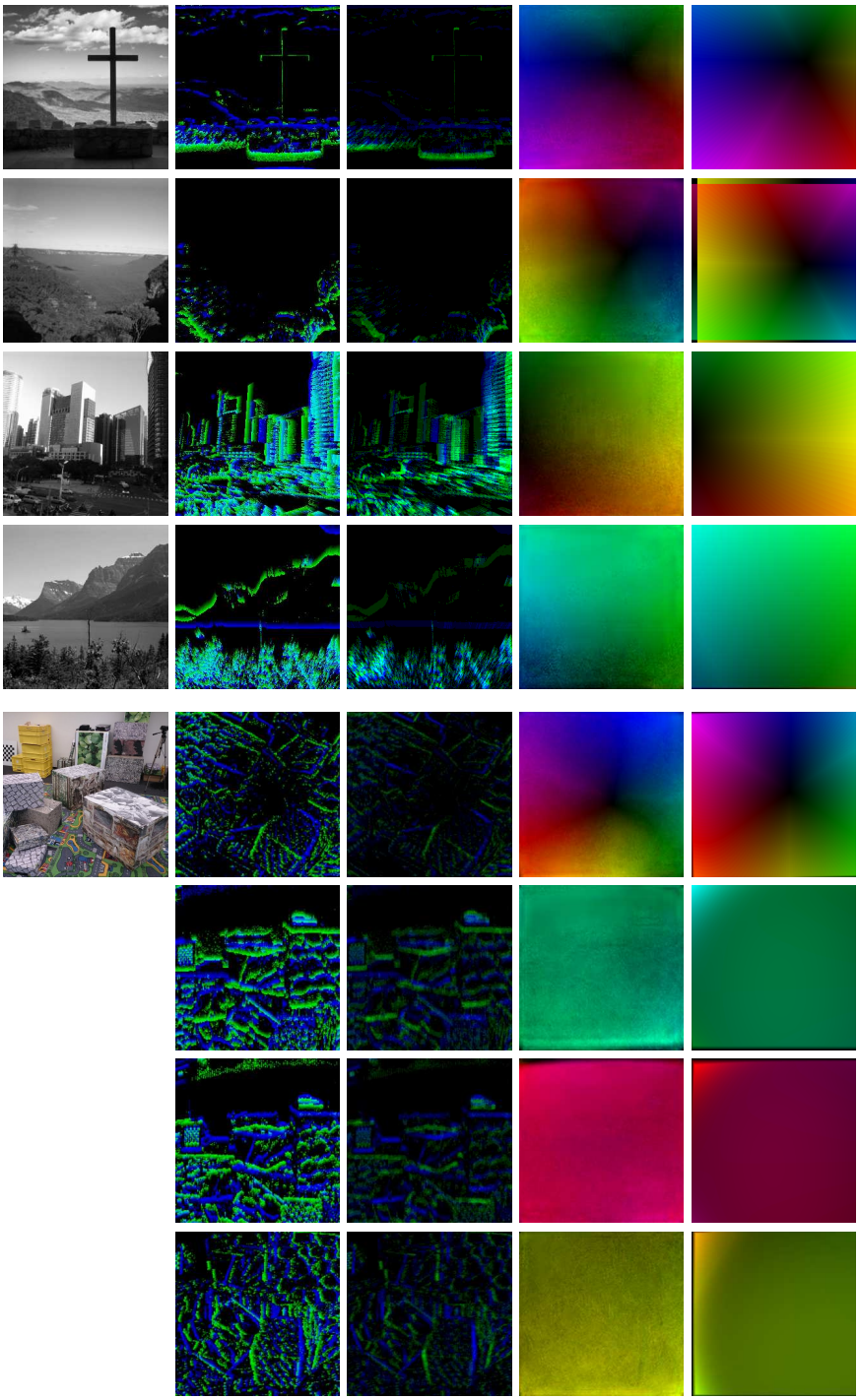


Figure A.10 – Results from learning globally affine flow from simulated event data and ground truth optic flow. The top 4 rows show network outputs on the validation set (simulated) and the bottom rows on the test set (real). The columns show a preview of the scenes (a), inputs to the network, comprising event time surfaces and event count frames (b-c) [213], and the predicted (d) and ground truth (e) optic flow.

B EMVS: Event-based Multi-View Stereo

Reprinted, with permission, from:

H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza. "EMVS: Event-based Multi-View Stereo—3D Reconstruction with an Event Camera in Real-Time". In: *Int. J. Comput. Vis.* 126.12 (Dec. 2018), pp. 1394–1414. DOI: 10.1007/s11263-017-1050-6

EMVS: Event-based Multi-View Stereo – 3D Reconstruction with an Event Camera in Real-Time

Henri Rebecq, Guillermo Gallego, Elias Mueggler and Davide Scaramuzza

Abstract — Event cameras are bio-inspired vision sensors that output pixel-level brightness changes instead of standard intensity frames. They offer significant advantages over standard cameras, namely a very high dynamic range, no motion blur, and a latency in the order of microseconds. However, because the output is composed of a sequence of asynchronous events rather than actual intensity images, traditional vision algorithms cannot be applied, so that a paradigm shift is needed. We introduce the problem of Event-based Multi-View Stereo (EMVS) for event cameras and propose a solution to it. Unlike traditional MVS methods, which address the problem of estimating *dense* 3D structure from a set of known viewpoints, EMVS estimates *semi-dense* 3D structure from an event camera with known trajectory. Our EMVS solution elegantly exploits two inherent properties of an event camera: (*i*) its ability to respond to scene edges—which naturally provide semi-dense geometric information without any pre-processing operation—and (*ii*) the fact that it provides continuous measurements as the sensor moves. Despite its simplicity (it can be implemented in a few lines of code), our algorithm is able to produce accurate, semi-dense depth maps, without requiring any explicit data association or intensity estimation. We successfully validate our method on both synthetic and real data. Our method is computationally very efficient and runs in real-time on a CPU.

Multimedia Material

A supplemental video for this work is available at
https://youtu.be/EFpZcpd9XJ0

B.1 Introduction

An event camera, such as the Dynamic Vision Sensor (DVS) [98], works very differently from a traditional camera. It has *independent* pixels that only send information (called "events") in presence of brightness changes in the scene at the time they occur. Thus, the output is not an intensity image but a stream of asynchronous events at microsecond resolution, where each event consists of its space-time coordinates and the *sign* of the brightness change (i.e., no intensity). Since events are caused by brightness changes over time, an event camera naturally responds to edges in the scene in presence of relative motion.

Event cameras have numerous advantages over standard cameras: a latency in the order of microseconds, low power consumption, and a very high dynamic range (130 dB compared to 60 dB of standard cameras). These properties make the sensors ideal in all those applications where fast response and high efficiency are crucial and also in scenes with wide variations of illumination. Additionally, since information is only sent in presence of brightness changes, the sensor removes all the inherent redundancy of standard cameras, thus requiring a very low data rate (kilobytes vs Megabytes). However, since event cameras became commercially available only recently [98], little related work exists, and, because their output is significantly different from that of standard cameras, traditional vision algorithms cannot be applied, which calls for new methods to process the data from these novel cameras, and therefore be able to unlock their potential.

Contribution

In this paper, we address the problem of structure estimation (i.e., 3D reconstruction) with a *single* event camera by introducing the concept of Event-based Multi-View Stereo (EMVS) (Section B.4), and we propose an algorithm to solve this problem.

Our approach (Sections B.5 to B.7) follows a Space-Sweep [28] voting and maximization strategy to estimate semi-dense depth maps at selected viewpoints, and then we merge the depth maps to build larger 3D models. We evaluate the method on both synthetic and real data (Section B.8). The results are analyzed and compared with ground truth, showing the successful performance of our approach.

This paper is based on our previous work [147], which we extend in several ways:

- We provide a justification of the choice of perspective sampling of space by analyzing the operation of event back-projection (Section B.6).
- We show how event back-projection can be efficiently implemented and parallelized using homographies to enable real-time performance, and we quantify the computational performance of our method (Section B.7).
- We improve structure estimation by means of simple processing techniques, such as bilinear voting in the Disparity Space Image (Section B.7.1) and median filtering of the semi-dense depth map (Section B.5.2).
- We include additional experiments (Section B.8), showing the applicability of our method.

B.2 Event Cameras and Applications

Event cameras are biologically inspired sensors that present a new paradigm on the way that dynamic visual information is acquired and processed. Each pixel of an event camera operates independently from the rest, continuously monitoring its intensity level and transmitting only information about brightness changes of given size ("events") whenever they occur, asynchronously, with microsecond resolution. Specifically, if $L(\mathbf{u}, t) \doteq \log I(\mathbf{u}, t)$ is the logarithmic brightness or intensity at pixel $\mathbf{u} = (x, y)^\top$ in the image plane, an event camera such as the DVS [98] (see Fig. B.1) generates an event $e_k \doteq \langle x_k, y_k, t_k, p_k \rangle$ if the change in logarithmic brightness at pixel $\mathbf{u}_k = (x_k, y_k)^\top$ reaches a threshold C (typically 10–15% relative brightness change):

$$\Delta L(\mathbf{u}_k, t_k) \doteq L(\mathbf{u}_k, t_k) - L(\mathbf{u}_k, t_k - \Delta t) = p_k C, \tag{B.1}$$

where t_k is the timestamp of the event, Δt is the time since the previous event at the same pixel $\mathbf{u}_k$, and $p_k = \pm 1$ is the polarity of the event (the sign of the brightness change). A comparison between the outputs of a standard and an event camera is shown in Fig. B.2.

Therefore, visual information is no longer acquired based on an external clock (e.g., global shutter); instead, each pixel has its own sampling rate, based on the visual input: event cameras are data-driven sensors. This different paradigm of acquiring visual information, i.e., reporting temporal contrast, offers significant advantages over that of standard cameras, namely redundancy removal, a very high dynamic range, no motion blur, and a latency in the order of microseconds. However, new computer vision algorithms that exploit the high temporal resolution and the asynchronous nature of the sensor are required to cope with this unfamiliar representation of the visual information.

Event cameras find applications in real-time interaction systems such as robotics or

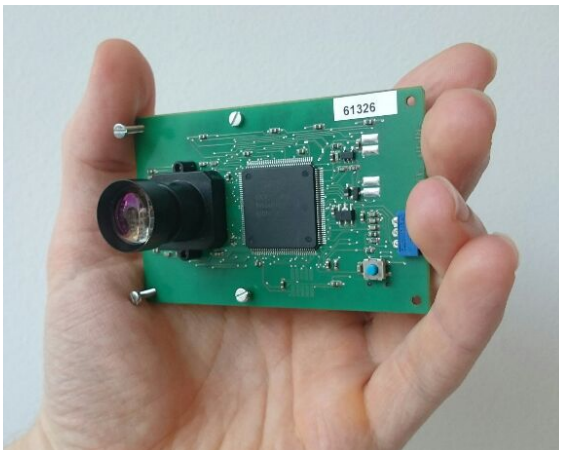


Figure B.1 – The event camera "eDVS" produced by iniLabs.

wearable electronics [37], where operation under uncontrolled lighting conditions, latency, and power are important. Event cameras have been used for object tracking [39, 45, 38], surveillance and monitoring [100, 137], object recognition [199, 133, 88] and gesture control [91]. They have also been used for stereo depth estimation [162, 136] (see also related work in Section B.3), 3D panoramic imaging [173], structured light 3D scanning [111], optical flow estimation [14, 13, 166, 10], high dynamic range (HDR) image reconstruction [30, 160], mosaicing [79] and video compression [19]. In ego-motion estimation, event cameras have been used for pose tracking [197, 123, 60], and visual odometry and Simultaneous Localization and Mapping (SLAM) [198, 24, 87, 80, 150]. Event-based vision is a growing field of research, and many more applications are expected to appear as event cameras become widely spread.

B.3 Related Work on Event-Based Depth Estimation

The majority of works on event-based depth estimation tackle the 3D reconstruction problem by using *two or more* event cameras that are rigidly attached (i.e., with a fixed baseline) and share a *common clock*. These methods follow a two-step approach: first they solve the event correspondence problem across image planes and then triangulate the location of the 3D point. Events are matched in two ways: either using traditional stereo methods on artificial frames generated by accumulating events over time [174, 85], or exploiting simultaneity and temporal correlations of the events across sensors [84, 162, 92, 22].

The event-based depth estimation problem that we address is entirely different: (*i*) we consider a *single* camera and (*ii*) we do not require simultaneous event observations.

Depth estimation from a single event camera is more challenging because we cannot

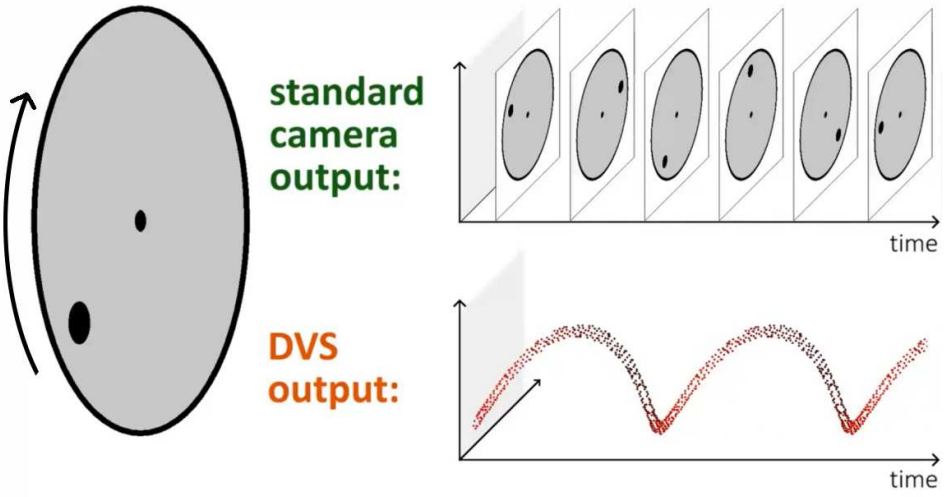


Figure B.2 – Comparison of the output of a standard camera and an event camera (DVS) when viewing a spinning disk with a black circle. The standard camera outputs frames at a fixed rate, thus sending redundant information when no motion is present in the scene. In contrast, event cameras are data-driven sensors that output pixel-level brightness changes with microsecond temporal resolution. Therefore, they do not suffer from motion blur and produce no output if there is no visual change in the scene. An animated version can be found here: https://youtu.be/LauQ6LWTkxM.

exploit temporal correlation between events across multiple image planes. Notwithstanding, we show that a single event camera suffices to estimate depth, and, moreover, that we are able to do it without solving the data association problem, as opposed to event-based stereo-reconstruction methods.

Since the publication of our monocular event-based depth estimation method [147], another solution has been proposed in [80]. Their method is part of a pipeline that uses three filters operating in parallel to jointly estimate the motion of the event camera, a 3D map of the scene, and the intensity image. Their depth estimation approach requires using an additional quantity—the intensity image—to solve for data association (events corresponding to the same 3D point have the same image intensity under the Lambertian hypothesis). Intensity estimation and depth regularization are carried out using dedicated hardware (a GPU) to achieve real-time performance. In contrast, our approach [147] leverages directly the sparsity of the event stream to perform 3D reconstruction (it does not need to recover the intensity image to estimate depth), and is computationally efficient, running in real-time on the CPU. In our most recent article [150], we address the problem of parallel tracking and mapping with an event camera; notably, we show how the 3D reconstruction method proposed in the present paper can be combined with an event-based pose tracking algorithm to yield both trajectory estimates as well as semi-dense 3D maps.

B.4 The Event-based Multi-View Stereo Problem

MVS with traditional cameras addresses the problem of 3D structure estimation from a collection of images taken from known viewpoints [188] of an intrinsically calibrated camera. Our Event-based MVS (EMVS) shares the same goal; however, there are some key differences:

1. Traditional MVS algorithms work on full images, so they cannot be applied to the stream of asynchronous events provided by the sensor. EMVS must take into account the *sparse* and *asynchronous* nature of the events.
2. Because event cameras do not output data if both the sensor and the scene are static, any event-driven algorithm, such as EMVS, requires the sensor to be *moved* in order to acquire visual content. In traditional MVS, the camera does not need to be in motion to acquire visual content.
3. Because events are caused by intensity edges, the natural output of EMVS is a *semi-dense* 3D map, as opposed to the dense maps of traditional MVS.

Hence, the EMVS problem consists of obtaining the 3D reconstruction of a scene from the sparse asynchronous streams of events acquired by moving event cameras with known viewpoints. Without loss of generality, it suffices to consider the case of one event camera.

To solve the EMVS problem, classical MVS approaches cannot be directly applied since they work on intensity images. Nevertheless, our event-based approach builds upon previous works on traditional MVS [175]. In particular, we follow (in Section B.5) the solving strategy of Scene Space MVS methods [175], which consist of two main steps: computing an aggregated consistency score in a discretized volume of interest (the Disparity Space Image (DSI)) by warping image measurements, and then finding 3D structure information in this volume. The term DSI [189] is interchangeably used to refer to the projective sampling of the volume (i.e., discretized volume) or to the scalar function defined in it (i.e., the score). Just by considering the way that visual information is provided, we can point out two key differences between the DSI approaches in MVS and EMVS:

1. In classical MVS, the DSI is *densely* populated using pixel intensities. In EMVS, the DSI may have holes (voxels with no score value), since warped events are also *sparse*.
2. In classical MVS, scene objects are obtained by finding an optimal surface in the DSI. By contrast, in EMVS, finding *semi-dense* structures (e.g., points, curves) is a better match to the sparsity of the DSI.

B.5 Event-Based Space-Sweep Method

Our method to solve the EMVS problem is similar to Collin's Space-Sweep approach for MVS [28], which shows how sparsity can be leveraged to estimate 3D structures without the need for explicit data association or photometric information. We generalize the Space-Sweep approach for the case of a moving event camera by building a virtual camera's DSI [189] containing only geometric information of edges and finding 3D points in it.

First, we review the classical Space-Sweep method for standard cameras (Section B.5.1), and then we describe our generalization to a moving event camera (Section B.5.2), showing that the continuous stream of events produced by the sensor is specially relevant to recover 3D structure.

B.5.1 Classical Space-Sweep Method

In contrast to most classical MVS methods, which rely on pixel intensity values, the Space-Sweep method [28] relies solely on binary edge images (e.g., Canny) of the scene from different viewpoints.

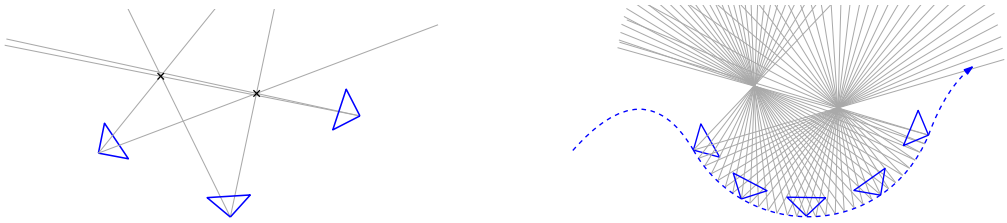
Thus, it leverages the sparsity or semi-density of the view-point dependent edge maps to determine 3D structure.

More specifically, the method consists of three steps: (1) warping (i.e., back-projecting) image features as rays through a DSI, (2) recording the number of rays that pass through each DSI voxel, and (3) determining whether or not a 3D point is present in each voxel. The DSI score measures the geometric consistency of edges in a very simple way: each pixel of a warped edge-map onto the DSI votes for the presence or absence of an edge. Then, the DSI score is thresholded to determine the scene points that most likely explain the image edges.

B.5.2 Event-Based Space-Sweep Method

In this section, we extend the Space-Sweep algorithm in Section B.5.1 to solve EMVS. Notice that the stream of events provided by event cameras is an ideal input to the Space-Sweep algorithm because (*i*) event cameras naturally highlight edges in hardware, and (*ii*) edges trigger events from *many* consecutive viewpoints rather than a few sparse ones (cf. Fig. B.3).

Next we detail the three steps of the event-based Space-Sweep method: back-projection (Section B.5.2), ray-counting (Section B.5.2), and determining the presence of scene structure (Section B.5.2). Then, we also discuss how to merge depth maps from multiple



(a) Classical (frame-based) Space-Sweep: only a fixed number of views is available. Two points of an edge map are visible in each image. The intersections of rays obtained by back-projecting the image points are used as evidence for detection of scene features (object points).

(b) Event-Based Space-Sweep: as the event sensor moves, events are triggered on the sensor. To each observed event corresponds a ray (through back-projection), that spans the possible 3D-structure locations. The areas of high ray density correspond to the locations of the two points, and are progressively discovered as the sensor moves.

Figure B.3 – Comparison of the back-projection step in classical Space-Sweep and Event-Based Space-Sweep. This is a 2D illustration with the scene consisting of two points.

viewpoints (Section B.5.2), and how to improve the quality of the reconstruction with simple post-processing techniques (Section B.5.2).

Feature-Viewing Rays by Event Back-Projection

Let us formally define an event $e_k = (x_k, y_k, t_k, p_k)$ as a tuple containing the pixel position (x_k, y_k), timestamp t_k, and polarity p_k (i.e., sign) of the brightness change. We extend the Space-Sweep method to the event-based paradigm by using the event stream $\{e_k\}$ output by the event camera as the input point-like features that are warped into the DSI. Each event e_k is back-projected according to the viewpoint of the event camera at time t_k, which is known according to the assumptions of MVS.

From a geometric point of view, we compare the back-projection step in the classical frame-based and the event-based settings using Fig. B.3. Observe that in frame-based MVS the number of viewpoints is small compared to that in the highly sampled trajectory of the event camera (at times $\{t_k\}$). This higher abundance of measurements and viewpoints in the event-based setting generates many more viewing rays than in frame-based MVS, and therefore, it facilitates the detection of scene points by analyzing the regions of high ray density.

A major advantage of our method is that no explicit data association is needed. This is the main difference between our method and existing event-based depth estimation methods (Section B.3). While other works essentially attempt to estimate depth by first solving the stereo correspondence problem in the image plane (using frames of accumulated events [174, 85], reconstructed intensity [80], temporal correlation of events [84, 162, 92, 22], etc.), our method works directly in 3D space.

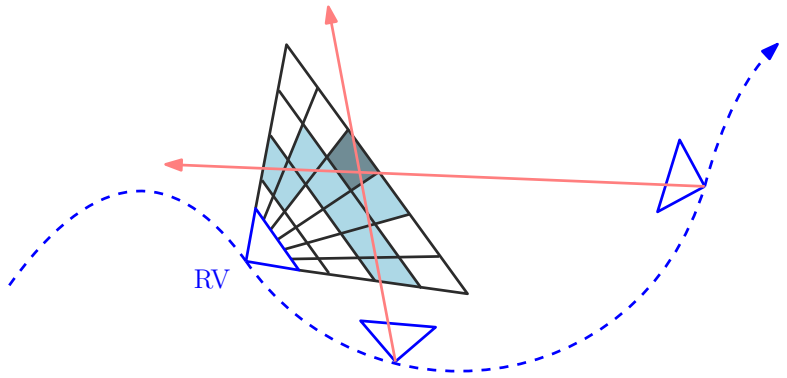


Figure B.4 – The DSI ray counter is centered at a virtual camera in a reference viewpoint (RV) and its shape is adapted to the perspective projection. Every incoming viewing ray from a back-projected event (in red) votes for all the DSI voxels (in light blue) which it traverses.

This is illustrated in Fig. B.3b: there is no need to associate an event to a particular 3D point to be able to recover its 3D location.

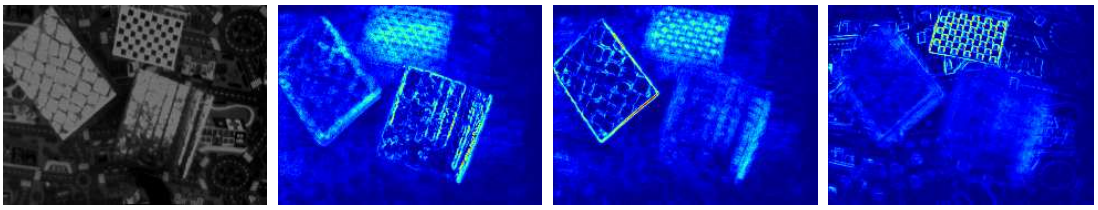
Volumetric Ray Counting. Creating the Disparity Space Image (DSI)

In the second step of Space-Sweep, we discretize the volume containing the 3D scene and count the number of viewing rays passing through each voxel using a DSI. To allow for the reconstruction of large scenes in a scalable way, we split the 3D volume containing the scene into smaller 3D volumes along the trajectory of the event camera, compute local 3D reconstructions, and then merge them, as will be explained in Section B.5.2.

For now, let us focus on computing a local 3D reconstruction of the scene from a subset of events. For this task, we create a virtual camera located at a reference viewpoint that is chosen among those event camera viewpoints associated to the subset of events, and then define a DSI in a volume V adapted to the field of view and perspective projection of the event camera, as illustrated in Fig. B.4 (see [189]). The DSI is defined by the event camera pixels and a number N_z of depth planes $\{Z_i\}_{i=1}^{N_z}$, i.e., it has size $w \times h \times N_z$, where w and h are the width and height of the event camera, respectively. The score stored in the DSI

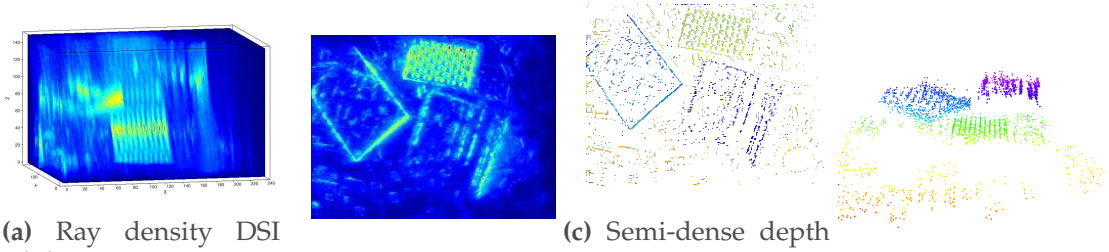
$$f(\mathbf{X}) : V \subset \mathbb{R}^3 \to \mathbb{R}^+ \tag{B.2}$$

is the number of back-projected viewing rays passing through each voxel with center $\mathbf{X} = (X, Y, Z)^\top$, as shown in Fig. B.4. We show in Section B.7.1 how to efficiently compute the ray-voxel intersections using a two-step approach, allowing for real-time performance on a single CPU.



(a) Image at virtual camera. **(b)** DSI slice at *close* depth. **(c)** DSI slice at *middle* depth. **(d)** DSI slice at *far* depth.

Figure B.5 – (a) Scene with the event camera moving above three textured planes located at different depths (close, middle, far). We build the ray density DSI $f(\mathbf{X})$ as described in Section B.5.2 and show the effect of slicing it at different depths, (b)–(d), as simulating a plane sweeping through the DSI. When the sweeping plane coincides with an object plane, the latter appears very sharp while the rest of the scene is "out of focus".



(a) Ray density DSI $f(\mathbf{X})$. **(b)** Confidence map. **(c)** Semi-dense depth map. **(d)** 3D point cloud.

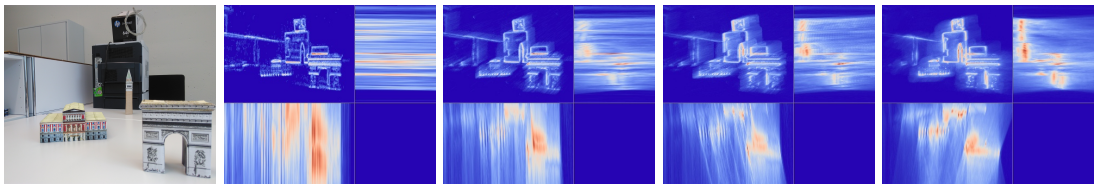
Figure B.6 – Our method builds the ray density DSI (a), from which a confidence map (b) and a semi-dense depth map (c) are extracted in a virtual camera. The semi-dense depth map gives a point cloud of scene edges (d). Same dataset as in Fig. B.5.

Detection of Scene Structure by Maximization of Ray Density

In the third step of Space-Sweep, we obtain a semi-dense depth map in the virtual camera by determining whether or not a 3D point is present in each DSI voxel. The decision is taken based on the ray density function stored in the DSI, $f(\mathbf{X})$.

Rephrasing the assumption of the Space-Sweep method [28], scene points are likely to occur at regions where several viewing rays nearly intersect (see Fig. B.3b), which correspond to regions of high ray density. Hence, scene points are likely to occur at *local maxima* of the ray density function. Fig. B.5 shows an example of slicing the DSI in Fig. B.6a, from a real dataset, at different depth planes; the presence of local maxima of the ray density function is evidenced by the in-focus areas. Additionally, Fig. B.7 shows the emergence of high ray-density regions in the DSI as the sensor moves and more events are observed.

We detect the local maxima of the DSI $f(\mathbf{X})$ following a two-step procedure: we first generate a dense depth map $Z^*(x, y)$ in the virtual camera and an associated confidence



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

Figure B.7 – Evolution of the DSI as the event camera moves. Figure (a) shows a preview of the scene, while figures (b) to (e) show the successive projections of the DSI along its three axes (top-left inset: front view, top-right inset: side-view, bottom-left inset: top-view). As more events are observed, areas of high ray density (in red) start appearing and the uncertainty in depth decreases in all directions. In this example, the DSI is sampled uniformly in inverse depth.

map $c(x,y)$ by recording the location and magnitude of the best local maximum of the DSI $f(X(x), Y(y), Z^*) =: c(x,y)$ along the row of voxels in the viewing ray of each pixel (x,y). Then, we select the most confident pixels in the depth map by thresholding the confidence map, yielding a semi-dense depth map (Fig. B.6c). We use Adaptive Gaussian Thresholding: a pixel (x,y) is selected if $c(x,y) > T(x,y)$, with $T(x,y) = c(x,y) * G_\sigma(x,y) - C$. In practice, we use a 5×5 neighborhood in G_σ and $C = -10$. The adaptive approach yields better results than global thresholding [28]. A summary of the main elements of our DSI approach is given in Fig. B.6.

Merging Depth Maps from Multiple Reference Viewpoints

So far, we have shown how to reconstruct the structure of scene corresponding to a subset of the events around a reference view. As pointed out in Section B.5.2, motivated by a scalable design, this operation is carried out on subsets of the event stream, thus recovering semi-dense depth maps of the scene at multiple *key* reference views. More specifically, we select a new *key* reference view as soon as the distance to the previous *key* reference view exceeds a certain percentage of the mean scene depth (typically a number between 15% and 40%), and use the subset of events until the next *key* reference view to estimate the corresponding semi-dense depth map of the scene. The depth maps are then converted to point clouds, cleaned from isolated points (those whose number of neighbors within a given radius is less than a threshold) and merged into a global point cloud using the known positions of the virtual cameras. Other depth map fusion strategies could be implemented. However, such a research topic is out of the scope of this paper. In practice, our approach shows compelling large-scale 3D reconstruction results even without the need for complex fusion methods or regularization.

Map Cleaning

To further enhance the quality of the 3D reconstruction, we use a median filter on the semi-dense depth maps obtained in Section B.5.2. Specifically, we consider only the converged pixels, i.e., the remaining pixels after the Adaptive Gaussian Thresholding, as input to the median filter. This allows removing outliers while preserving depth discontinuities.

Additionally, we also apply a radius filter [168] to the final point cloud, which discards the points whose number of neighbors within a given radius is less than a threshold. This helps remove isolated points, which are most likely outliers.

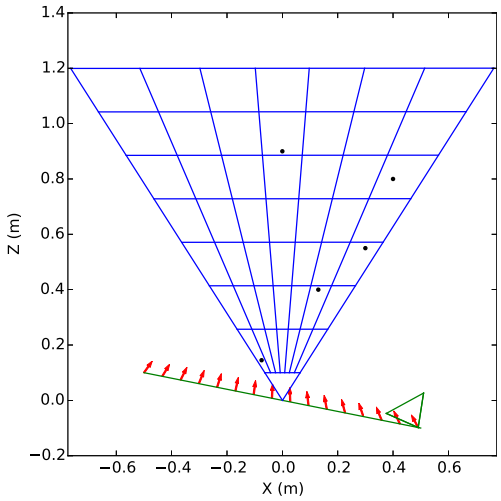
B.6 Sampling the DSI: Uniform vs. Projective

In this section we justify our choice of using a projective sampling of the DSI volume, i.e., a projective voxel grid, instead of using a uniform sampling (as originally proposed in [28]). The reader who is not interested in this explanation can jump to Section B.7.

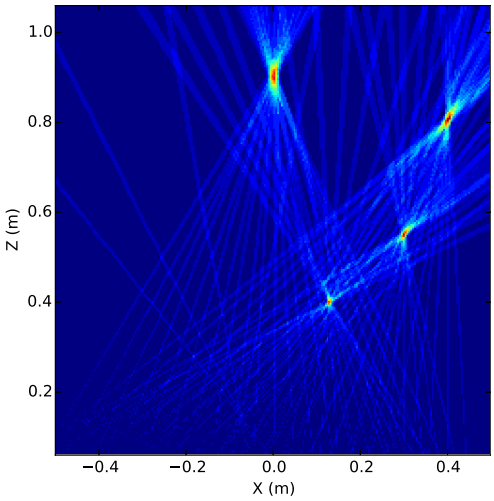
We compare both sampling strategies (uniform and projective) by means of a simple experiment in 2D, illustrated in Fig. B.8, and support the comparison by means of well-grounded mathematical results.

Let us consider a 2D scene consisting of a moving event camera and a few set of points with large contrast so that they generate events (Fig. B.8a).

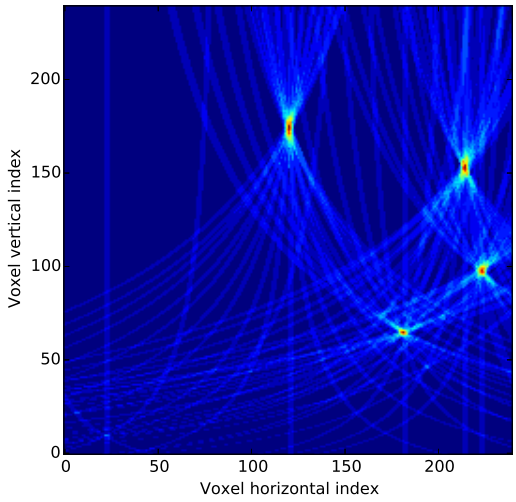
For simplicity, and since our method does need the event polarity, we model the event camera as a sensor that outputs a binary value describing whether a scene point is visible by a specific camera pixel. This is only an approximate model; for example, an event camera moving forward towards a point in the center of the image plane would not trigger events (since the brightness of this pixel does not change), but in this model we consider that for every visible scene point an event is generated at each camera pose. Nevertheless, this is a good geometric model that provides insight into the EMVS problem and our proposed solution. We use this model to compute the DSI ray density function on a region of the XZ space, and sample it in two different ways: (*i*) using a uniform grid along both X and depth Z axes (i.e., on a Cartesian grid), as shown in Fig. B.8b, and (*ii*) using a projective grid (as in Fig. B.4) that mimics the perspective operation of a camera located somewhere along the event camera trajectory, as shown in Fig. B.8a. Approach (*i*) corresponds to the one originally proposed in [28]. We are interested in comparing the effect of both sampling strategies on the shape and size of the rays, more correctly "cones", obtained by back-projecting events, that is, we consider that pixels are not just points but have a finite extent.



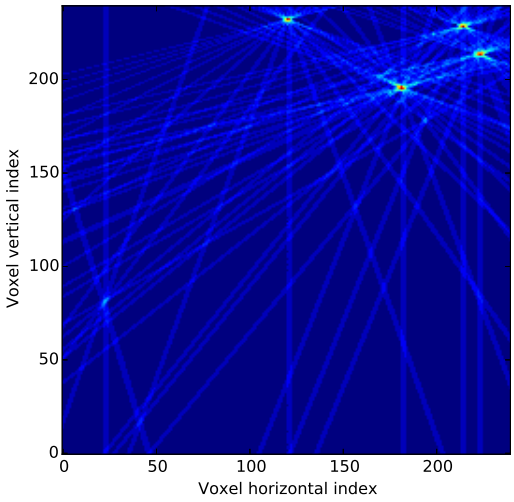
(a) 2D scene geometry featuring five points, the camera trajectory (in green) and optical axis direction (in red), and the projective voxel grid (in blue).



(b) Ray density in Euclidean space (uniform voxel grid). The width of each ray grows with the depth.



(c) Ray density using projective voxel grid, equispaced in depth (voxel vertical index). The width of each ray is constant along the depth.



(d) Ray density using projective voxel grid, equispaced in inverse depth (voxel vertical index). As in Fig. B.8c, the width of each ray is constant along the depth.

Figure B.8 – Illustration of uniform vs. projective sampling of the DSI using a 2D example. Ray density plots are pseudo-colored, from dark blue (small density) to red (high density). Figure generated with 50 camera poses, a camera FOV of 75 degrees, and image resolution of 100 pixels (along the camera's X axis). The voxel grid has a resolution of 240 pixels (along the X axis) and 240 depth planes.

B.6.1 Shape of the Back-Projected Rays

First, let us analyze the shape, i.e., ignoring the finite extent of the pixel. Later, we will analyze the effect of the finite pixel size on the back-projection operation to create the

DSI. The ray back-projected from a point $\mathbf{u}$ in the camera is a line in Euclidean space. Using calibrated coordinates, and assuming that $\mathtt{P} = (\mathtt{R}|\mathbf{t})$ is the projection matrix of the camera, the ray is given by the line joining two points [70, p.162]: the optical center of the camera $\mathbf{C} = -\mathtt{R}^\top \mathbf{t}$ and the point at infinity $(\mathbf{D}^\top, 0)^\top$, with $\mathbf{D} = \mathtt{R}^\top \mathbf{u}$, projecting on $\mathbf{u}$. A point on the ray has Euclidean coordinates

$$\mathbf{X} = \rho \mathbf{D} + \mathbf{C}. \tag{B.3}$$

These are the parametric equations of the line, with depth parameter ρ. The uniform sampling strategy preserves the straight nature of the back-projected rays, as shown in Fig. B.8b. In contrast, the rays are no longer straight in the case of the projective sampling (Fig. B.8c). In the projectively sampled space, a Euclidean point $\mathbf{X} \doteq (X, Y, Z)^\top$ is described by coordinates

$$\mathbf{X}_p = \left(\frac{X}{Z}, \frac{Y}{Z}, Z \right)^\top \doteq (x, y, Z)^\top. \tag{B.4}$$

Letting $\mathbf{C} = (C_i)$ and $\mathbf{D} = (D_i)$, $i = 1, \ldots, 3$, we combine (B.3) and (B.4) to obtain the parametric equations

$$\mathbf{X}_p = \left(\frac{\rho D_1 + C_1}{\rho D_3 + C_3}, \frac{\rho D_2 + C_2}{\rho D_3 + C_3}, \rho D_3 + C_3 \right)^\top. \tag{B.5}$$

Let us show that (B.5) explains the curved shapes of back-projected rays observed in Fig. B.8c. For depth values $\rho \gg 1$, the points on the ray follow the curve $\mathbf{X}_p \approx (D_1/D_3, D_2/D_3, 0)^\top + \rho\,(0, 0, D_3)^\top$, which is a line with direction vector $(0, 0, D_3)$, i.e., a line parallel to the Z-axis. This is observed in the top part of Fig. B.8c. For small depth values ($\rho \to 0$), the points on the ray approach the optical center of the camera, as expected, $\mathbf{X}_p \approx (C_1/C_3, C_2/C_3, C_3)^\top$, so we look at the way that they approach this point by computing the tangent:

$$\frac{d\mathbf{X}_p}{d\rho} \overset{\text{(B.5)}}{=} \left(\frac{D_1 C_3 - D_3 C_1}{(\rho D_3 + C_3)^2}, \frac{D_2 C_3 - D_3 C_2}{(\rho D_3 + C_3)^2}, D_3 \right)^\top. \tag{B.6}$$

The plots in Fig. B.8c where generated with a moving camera with $C_3 \ll D_3$, and so, for small depth values, $d\mathbf{X}_p/d\rho \approx ((-C_1/(D_3\rho^2), -C_2/(D_3\rho^2), D_3)^\top$. In the 2D example (only considering X and Z coordinates), as $\rho \to 0$ the tangent is dominantly along the X axis, which agrees with Fig. B.8c. In summary, when going from zero to infinite depth, the tangent changes from being parallel to the X axis to being parallel to the Z axis, and so, the tangent varies (smoothly) between these two directions, as shown in the curved shapes of Fig. B.8c.

Finally, consider what happens when the DSI is sampled projectively and equispaced in inverse depth instead of depth: the curved shapes analyzed in Fig. B.8c become

almost straight, as shown in Fig. B.8d. This is similar to the effect of representing the function $y = e^x$ in logarithmic scale: $\log(y)$ becomes a line. The curve represented by the X and Z coordinates of (B.5) is the parametric curve $(x(r), r)$, with $x(r) = D_1/D_3 + (C_1 - (C_3D_1)/D_3)r^{-1}$ and the change of variables $r = \rho D_3 + C_3$. Thus, $x(r)$ is a line when using the parameter $r^{-1} = (\rho D_3 + C_3)^{-1}$, which is approximately inverse depth. Fig. B.8d was generated with $C_3 \ll D_3$, and so the ray $(x(r), r)$ is indeed almost straight when the Z axis is given in inverse depth.

B.6.2 Size of the Back-Projected Cones

We now consider that pixels have a non-zero area and study how a back-projected event contributes to the DSI depending on the sampling scheme.

A pixel collects the light in a fixed, small angle around a given direction. This angle correspond to different object sizes depending on the distance of the object to the camera. This idea is roughly expressed by the formula of the area A of a sphere patch seen by a central solid angle Ω: $A = \Omega r^2$, where r is the radius of the sphere. Thus, the same pixel angle Ω covers an area A at a distance r and an area four-times larger $4A$ at double the distance $2r$. Hence, the back-projection of a pixel into space generates a cone whose base area A grows quadratically with the distance to the camera.

In a uniform sampling of the DSI, where all voxels have the same size, a pixel back-projects into a cone that will cover more voxels the farther they are from the camera. In contrast, using a projective sampling of the DSI, we compensate for the perspective effect of the camera by making the size of the voxel increase with the distance of the voxel to the virtual camera defining the projective grid, so that a pixel back-projected into space will cover always roughly the same number of voxels: one. This comparison can be observed in Figs. B.8b and B.8c. In Fig. B.8b we can identify the cones, whose apexes lie on the event camera trajectory. In Fig. B.8c, the cones are represented by curves of approximately constant width (perpendicular to the depth axis). This constant width is also appreciated in Fig. B.8d, where the cones become "cylinders".

Let us mathematically support the previous statements. Fig. B.8a illustrates the geometry of the projective sampling considered. The projective DSI is defined by a virtual camera with projection matrix $\mathtt{P}_v = (\mathtt{I}|0)$, in calibrated coordinates. At the time of the current event $e = (u, v, t, p)$, the event camera is described by projection matrix $\mathtt{P}_e = (\mathtt{R}|\mathbf{t})$. The pixel where the event has been triggered is back-projected into points of the form (B.4) in the projective DSI. Each depth plane $Z = Z_i$ induces a planar homography between the image plane of the event camera and the image plane of the virtual camera, by mapping the event coordinates $(u, v)^\top$ to the first two coordinates of (B.4), $(x, y)^\top$. We use this planar homography to measure the area in the virtual camera (i.e., the area perpendicular to the depth axis in the projective grid) that is due

to the pixel that triggered the event. The relation between the area elements in both cameras is given by the determinant of the Jacobian of the homography:

$$dxdy = \det\left(\frac{\partial(x,y)}{\partial(u,v)}\right) dudv. \tag{B.7}$$

The planar homography $\mathtt{H}_{Z_i} : (u,v) \mapsto (x,y)$ from the event camera to the virtual camera, induced by the plane $Z = Z_i$ (with coordinates $\boldsymbol{\pi} = (\mathbf{e}_3^\top, -Z_i)^\top$, $\mathbf{e}_3 = (0,0,1)^\top$), is given by the inverse of the homogeneous matrix (see (B.24))

$$\mathtt{H}_{Z_i}^{-1} \sim \mathtt{R} + \frac{1}{Z_i}\mathbf{t}\mathbf{e}_3^\top. \tag{B.8}$$

The Jacobian in (B.7) can be computed applying Result 2 in the Appendix to (B.8) and the fact that the Jacobian of $\mathtt{H}_{Z_i}$ is the inverse of the Jacobian of $\mathtt{H}_{Z_i}^{-1}$:

$$\det\left(\frac{\partial(x,y)}{\partial(u,v)}\right) = \left(\frac{Z_i'}{Z_i}\right)^3 \left(1 - \frac{C_z}{Z_i}\right)^{-1}, \tag{B.9}$$

where Z_i' is the depth of the point $\mathbf{X} \in \boldsymbol{\pi}$ with respect to the event camera $\mathtt{P}_e$, and C_z is the third coordinate of the optical center of $\mathtt{P}_e$. Therefore, the conversion factor between areas in the image planes is a function of the ratio of depths of the scene point with respect to both cameras and the ratio of depths C_z/Z_i. Assuming that the scene point is equally far away from both cameras (i.e., $Z_i' \approx Z_i$) and that the amount of forward motion of the event camera is negligible compared to Z_i, $C_z \ll Z_i$, the conversion factor (B.9) in (B.7) becomes approximately 1, that is, a pixel maps to an area (perpendicular to the depth axis) of 1 pixel in the projective grid; this is the area of a cross-section of a voxel, hence for each depth plane $Z = Z_i$, a pixel in the event camera votes for 1 voxel in the projective grid.

To summarize, we have shown that the projective sampling of the DSI is a better choice than the uniform sampling because a back-projected event will vote for approximately one grid cell per depth plane instead of multiple cells (in case of uniform sampling) whose number would grow quadratically with depth. This property (area conversion factor ≈ 1) is not only advantageous when creating the DSI (only one vote needed per depth plane), but also when extracting the scene edges from it. Indeed, areas at different depth planes of the virtual camera are comparable when using the projective DSI, thus enabling the use of a fixed-size adaptive-threshold mask in all depth planes to extract clusters of high ray density along the viewing rays of the virtual camera. In contrast, with a uniform voxel grid, the size of the clusters depends on the depth, which means that the mask size of the adaptive threshold itself would have to be dependent on the depth plane.

Remark. The previous analysis used calibrated coordinates. If, instead, we use pixel

coordinates, with $\mathtt{K}_v$ and $\mathtt{K}_e$ being the intrinsic parameter matrices of the DSI virtual camera and the event camera, respectively, it is easy to show, using an argument on how area elements transform (B.7), that (B.9) will become

$$\det\left(\frac{\partial(x,y)_{\text{pixel}}}{\partial(u,v)_{\text{pixel}}}\right) = \frac{\det(\mathtt{K}_v)}{\det(\mathtt{K}_e)}\left(\frac{Z_i'}{Z_i}\right)^3\left(1-\frac{C_z}{Z_i}\right)^{-1}, \tag{B.10}$$

that is, the ratio of the focal lengths of the cameras can be used to modify the number of voxels that each event votes for. However, a typical design choice is $\det(\mathtt{K}_v) = \det(\mathtt{K}_e)$ so that such a number is 1, as analyzed above.

Other compelling reasons to choose a local projective DSI over a global, uniform DSI are that: (*i*) for a given amount of memory, it is better to maintain a local map since it allows for higher resolution, and (*ii*) for some applications, such as visual odometry (without loop closure), it suffices to provide a local 3D map.

B.7 Algorithmic Considerations for Real-Time Performance

The goal of this section is two-fold: (*i*) describe the two-step approach that is used to accelerate computations and (*ii*) quantitatively measure the computational performance of the method (e.g., in number of events processed per second).

B.7.1 Efficient Event Back-Projection onto the DSI

Following [28], we populate the DSI using a space-sweep strategy. However, our approach differs from his in the fact that we use a projective DSI instead of a uniform one and we keep the entire DSI in memory, not just a slice of it, for later processing.

The approach is summarized in Algorithm 1. The main idea behind the approach is that to compute the back-projection locations corresponding to the depth plane $Z = Z_i$ it is more efficient to do it in two steps (back-projecting via a depth plane $Z = Z_0$ and then modifying the point locations to take into account the change in Z value) than it is to apply the homography to the original points. This is illustrated in Fig. B.9.

The homography to transfer points from the event camera to points on the virtual camera of the DSI via a plane Z_0 is $\mathtt{H}_{Z_0}$, used in step 1 of Algorithm 1:

$$(x(Z_0), y(Z_0), 1)^\top \sim \mathtt{H}_{Z_0}(u, v, 1)^\top, \tag{B.11}$$

where we explicitly wrote the dependency of the transferred point $(x(Z_0), y(Z_0))$ with respect to the plane used $Z = Z_0$. Points transferred via another plane, $Z = Z_i$, can be

Algorithm 1 Efficient event back-projection

Goal: back-project events positions $\{(u_j, v_j)\}$ to the projective DSI.
Input: a projective DSI defined by a virtual camera $\mathtt{P} = (\mathtt{I}|\mathbf{0})$ and N_z depth planes $Z = Z_i$; points $\{(u_j, v_j)\}$ at the current location of the event camera $\mathtt{P}_e = (\mathtt{R}|\mathbf{t})$.
Procedure:

1. Map points from the event camera to the virtual camera via a canonical plane $Z = Z_0$, according to homography $\mathtt{H}_{Z_0}$ (see (B.8)), and store the transferred points $\{(x_j(Z_0), y_j(Z_0))\}$ with full precision.
2. For each depth plane $Z = Z_i$:
   (a) Map points from the event camera to the virtual camera via the plane $Z = Z_i$ using the homography $h_{i0} \equiv \mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1}$ on the stored points: $(x_j(Z_i), y_j(Z_i)) = h_{i0}((x_j(Z_0), y_j(Z_0)))$. See (B.15).
   (b) Vote for the DSI voxels at positions $\{(x_j(Z_i), y_j(Z_i), Z_i)\}$.

written in terms of the points transferred using $Z = Z_0$ as follows:

$$(x(Z_i), y(Z_i), 1)^\top \sim \mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1}\,(x(Z_0), y(Z_0), 1)^\top, \tag{B.12}$$

where the homography $\mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1}$ has a very simple structure: a similarity without rotation. Let us show this. Using the matrix inversion lemma on the first term of

$$\mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1} \overset{(B.8)}{=} \left(\mathtt{R} + \frac{1}{Z_i}\mathbf{t}\mathbf{e}_3^\top\right)^{-1}\left(\mathtt{R} + \frac{1}{Z_0}\mathbf{t}\mathbf{e}_3^\top\right), \tag{B.13}$$

and the equation of the optical center of the event camera, $(C_x, C_y, C_z)^\top \doteq \mathbf{C} = -\mathtt{R}^\top\mathbf{t}$, we obtain

$$\mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1} \sim \mathtt{I} + \frac{Z_0 - Z_i}{Z_0(Z_i - C_z)}\mathbf{C}\mathbf{e}_3^\top. \tag{B.14}$$

Dividing the homogeneous matrix (B.14) by its last entry and writing (B.12) in expanded form gives

$$\begin{aligned} x(Z_i) &= \frac{Z_0}{Z_i}\delta\; x(Z_0) + \frac{1}{Z_i}(1-\delta)C_x, \\ y(Z_i) &= \frac{Z_0}{Z_i}\delta\; y(Z_0) + \frac{1}{Z_i}(1-\delta)C_y, \end{aligned} \tag{B.15}$$

where $\delta = (Z_i - Z_0)/(Z_0 - C_z)$. Hence, the transformation $\mathtt{H}_{Z_i}\mathtt{H}_{Z_0}^{-1}$ in (B.15) is very simple and fast to compute. This is the advantage of the two-step approach. These equations are similar to the equations in [28], except for the additional multiplicative factors Z_0/Z_i and $1/Z_i$.

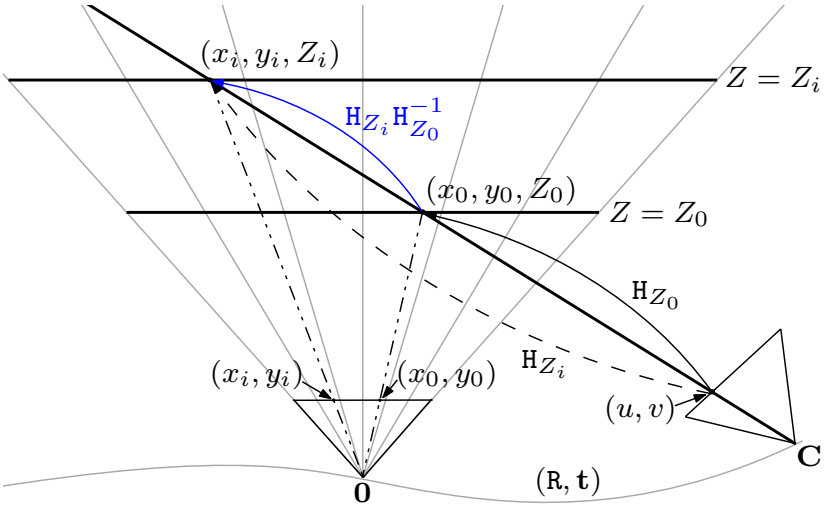


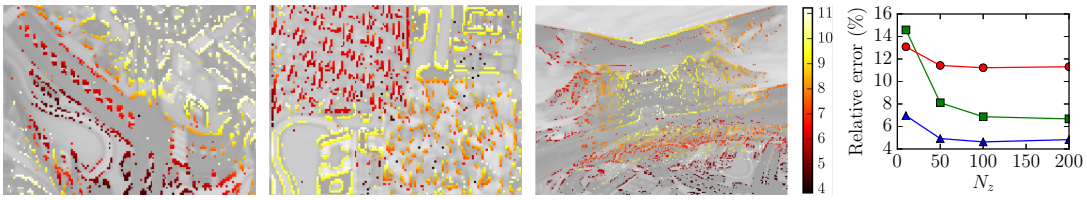
Figure B.9 – *Efficient event back-projection in Algorithm 1.* An event with coordinates (u, v) is mapped onto the depth plane $Z = Z_i$ of the projective DSI in two steps: first, it is mapped to the depth plane $Z = Z_0$ via H_{Z_0} and then it is mapped to $Z = Z_i$ via the similarity $\mathrm{H}_{Z_i}\mathrm{H}_{Z_0}^{-1}$ in (B.15). In the figure, the notation $x_i = x(Z_i)$ and $y_i = y(Z_i)$ is used for brevity.

Accumulating votes in the DSI (line 2b of Algorithm 1) is a process known as forward mapping in image processing [200, ch.3], and it can be done in different ways. The simplest one is nearest neighbor: point $(x_j(Z_i), y_j(Z_i))$ votes for a single cell of the depth plane $Z = Z_i$. A better strategy because it mitigates the grid discretization effect is bilinear voting: point $(x_j(Z_i), y_j(Z_i))$ votes for its four nearest cells on the depth plane $Z = Z_i$, splitting the vote according to the distances of $(x_j(Z_i), y_j(Z_i))$ to the integer cell locations, similarly to bilinear interpolation.

B.7.2 Computational Performance of the Method

The algorithm can be parallelized in a multi-core architecture by making each thread work on a different group of depth planes so that there are no race conditions during voting.

The two-step approach in Algorithm 1 is efficient if events are processed in groups or batches. Theoretically, each event has a different camera pose $\mathrm{P}_e(t)$, but using a different pose to process each event would make any algorithm terribly inefficient. For example, just the simple operation of pose interpolation along the camera trajectory becomes an expensive operation when it is done at the event rate (in the order of 10^5 to 10^6 events per second). In practice, it is sensible to assume that events, which have microsecond resolution, can be grouped in time so that they are assigned the same camera pose and processed together (i.e., they share the same homography H_{Z_0}, which is the most expensive part to compute). We typically use batches containing a small,



(a) Dunes **(b)** 3 planes **(c)** 3 walls **(d)** Depth error.

Figure B.10 – Synthetic experiments: estimated semi-dense depth maps overlayed over screenshots of the scene, in three datasets (a)-(c). Depth is colored, from close (red) to far (yellow). Our EMVS algorithm successfully recovers most edges, even without regularization or outlier filtering. (d): Relative depth error as a number of depth planes N_z, in all three datasets: Dunes (blue), 3 planes (red), and 3 walls (green).

fixed number of events (typically, 256 events). The corresponding time interval depends on the event rate (hence the camera motion), but it is typically very small (in the order of 1 ms or less).

The number of operations required to compute the DSI grows linearly with the number of depth planes in the voxel grid. Moreover, as explained in Section B.6.2, for the choice $\det(\mathrm{K}_v) = \det(\mathrm{K}_e)$, the complexity does not depend on the spatial resolution of the depth planes, because in that case only one vote is necessary per depth plane.

Finally, for an efficient implementation with real cameras, it is a good practice to use a look-up-table of undistorted calibrated coordinates (u, v) of the event camera and to use SIMD instructions for matrix multiplications in Algorithm 1.

Quantitative Evaluation. We measured the speed of our implementation on a Lenovo W541 laptop computer containing an Intel Core i7-4810MQ @2.80 GHz quad-core processor (consuming about 47W), and a scene recorded in a typical office environment (similar to the first row in Fig. B.15) with the DAVIS camera (240×180 resolution). The event rate in the scene varied between 250 000 events/s and 900 000 events/s. We used 100 depth planes in the voxel grid, and a batch size of 256 events. On a single core, our implementation can process on average 1.2 million events per second (which is higher than the maximum event rate in the scene, thus running faster than real-time), and on average 4.7 million events/s with the multi-core implementation (using 4 cores). Note that this cost does not include the cleaning steps described in Section B.5.2; however, their running time is usually negligible compared to the voxel grid voting steps. The memory usage is dominated by the voxel grid storage in memory (votes are stored as 32-bit floating point values), and can be significant depending on the sensor resolution and number of depth planes used. For example, for a sensor resolution of 240×180 and 100 depth planes, it would be 17.3 MB.

Table B.1 – Depth estimation accuracy in the synthetic datasets ($N_z = 100$)

	Dunes	3 planes	3 walls
Depth range	3.00 m	1.30 m	7.60 m
Mean error	0.14 m	0.15 m	0.52 m
Relative error	4.63 %	11.31 %	6.86 %

B.8 Experiments

We now evaluate the performance of our event-based Space Sweep Method, on both synthetic and real datasets.

B.8.1 Synthetic Data

We generated three synthetic datasets with ground truth information by means of an event camera simulator [124]. We set the spatial resolution to 240×180 pixels, as that of commercial event sensors. The datasets also contain intensity images along the event camera viewpoints. However, these are not used in our EMVS algorithm; they are solely shown to aid the visualization of the semi-dense depth maps obtained with our method. The datasets exhibit various depth profiles and motions: *Dunes* consists of a smooth surface (two dunes) and a translating and rotating camera in two degrees of freedom (DOF), *3 planes* shows three planes at different depths (i.e., discontinuous depth profile with occlusions) and a linear camera motion; finally, *3 walls* shows a room with three walls (i.e., a smooth depth profile with sharp transitions) and a general, 6-DOF camera motion.

Our EMVS algorithm was executed on each dataset. First, we evaluated the sensitivity of our method with respect to the number of depth planes N_z used to sample the DSI. In this experiment, the planes in the DSI were equispaced in depth (as opposed to inverse depth) since it provided better results in scenes with finite depth variations. Fig. B.10d shows, as a function of N_z, the relative depth error, which is defined as the mean depth error (between the estimated depth map and the ground truth) divided by the depth range of the scene. As expected, the error decreases with N_z, but it stagnates for moderate values of N_z. Hence, from then on, we fixed $N_z = 100$ depth planes. Table. B.1 reports the mean depth error of the estimated 3D points, as well as the relative depth error for all three datasets. Depth errors are small, in the order of 10 % or less, showing the good performance of our EMVS algorithm and its ability to handle occlusions and a variety of surfaces and camera motions.

Table B.2 – Depth estimation accuracy in the HDR experiment (no post-processing)

	Close (distance: 23.1 cm)		Far (distance: 58.5 cm)	
Illumination	Mean error	Relative error	Mean error	Relative error
∘ constant	1.22 cm	5.29 %	2.01 cm	4.33 %
∘ HDR	1.21 cm	5.25 %	1.87 cm	3.44 %

B.8.2 Real Data

We also evaluated the performance of our EMVS algorithm on datasets from a DAVIS sensor [18]. The DAVIS outputs, in addition to the event stream, intensity frames as those of a standard camera, at low frame rate (24 Hz)[1]. However, our EMVS algorithm does not use the frames; they are displayed here only to illustrate the semi-dense results of the method.

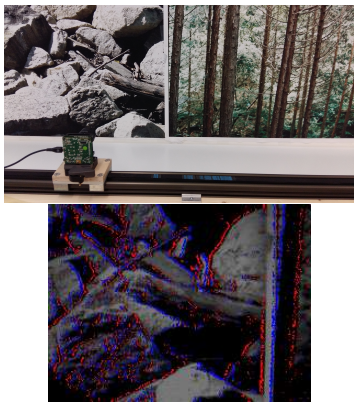
We considered two methods to provide our EMVS algorithm with camera pose information: a motorized linear slider or a visual odometry algorithm on the DAVIS frames. We used the motorized slider to analyze the performance in controlled experiments (since it guarantees very accurate pose information) and a visual odometry algorithm (SVO [53]) to show the applicability of our method in hand-held (i.e., unconstrained) 6-DOF motions.

High Dynamic Range and High-Speed Experiments

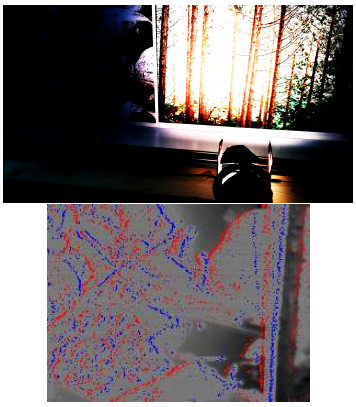
In this section, we show that our EMVS algorithm is able to recover accurate semi-dense structure in two challenging scenarios, namely (i) high-dynamic-range (HDR) illumination conditions and (ii) high-speed motion. For this, we place the DAVIS on the motorized linear slider, facing a textured wall at a known constant depth from the sensor. In both experiments, we measure the accuracy of our semi-dense maps against ground truth and demonstrate compelling depth estimation accuracy, in the order of 5 % of relative error, which is very good, especially considering the low resolution of the sensor (only 240×180 pixels). In order to provide a fair measurement of the raw accuracy of our approach, we did not perform any additional post-processing or map cleaning (Section B.5.2) for these quantitative experiments.

High Dynamic Range Experiment. We recorded two datasets under the same acquisition conditions except for illumination (Fig. B.11): first with constant illumination throughout the scene and, second, with a powerful lamp illuminating only half of the scene. In the latter case, a standard camera cannot cope with the wide intensity

[1]The DAVIS comprises both a frame camera and an event sensor (DVS) in the same pixel array of size 240×180. The frames may be used to simplify intrinsic camera calibration, by applying standard algorithms [206]. Otherwise, tailored event-based algorithms, such as [123], may be applied.



(a) Constant illumination setup. Events on a frame.



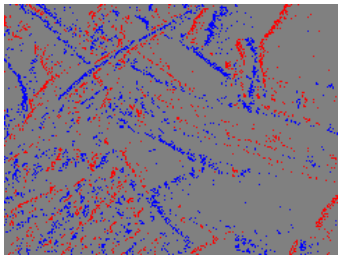
(b) HDR illumination setup. Events on a frame.

(c) Constant illum. 3D points: front and top views.

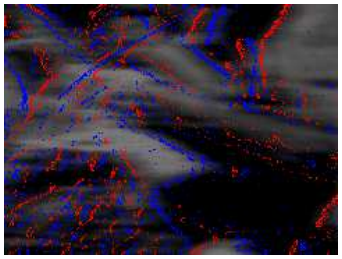
(d) HDR illum. 3D points: front and top views.

Figure B.11 – *HDR experiment*: Top: Scene and illumination setups, with the DAVIS on the motorized linear slider (a) and a lamp (b). Sample frames show under- and over-exposed levels in HDR illumination (b). By contrast, the events (overlayed on the frames) are unaffected, due to the high dynamic range of the event sensor. Bottom: reconstructed point clouds.

(a) Frame (motion blur).



(b) Events ($\Delta t = 2ms$).

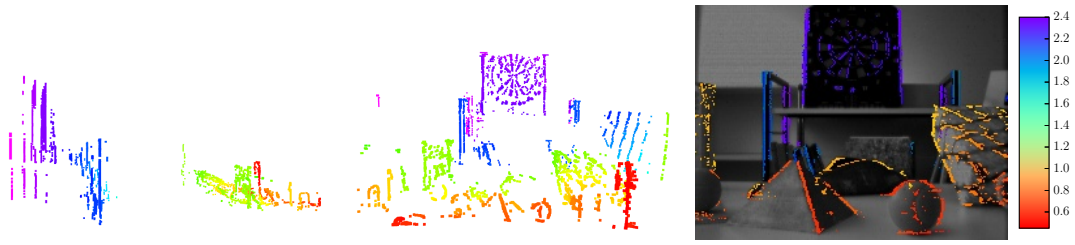


(c) Frame and events.

Figure B.12 – *High-speed experiment*. Frame and the events from the DAVIS at 376 pixels/s. The frame suffers from motion blur, while the events do not, thus preserving the visual content.

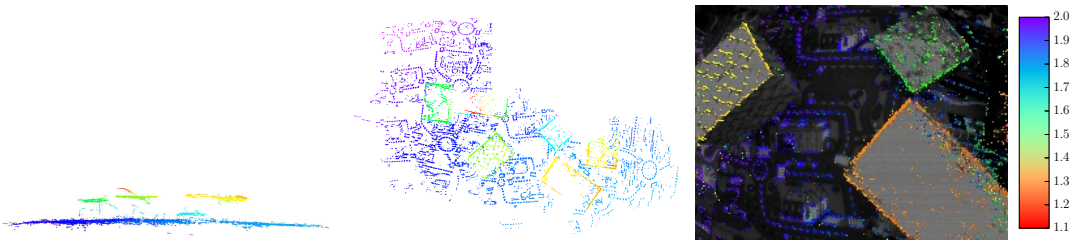
variation in the middle of the scene since some areas of the images are under-exposed while others are over-exposed. We performed the HDR experiment with two different wall distances (close and far).

The results of our EMVS algorithm are given in Fig. B.11 and Table B.2. Observe that



(a) Side view. **(b)** Front view. **(c)** Projection on a frame.

Figure B.13 – *Desk dataset*: scene with objects and occlusions.



(a) Side view. **(b)** Top view. **(c)** Projection on a frame.

Figure B.14 – *Boxes dataset:* large-scale semi-dense 3D reconstruction with a hand-held DAVIS.

the quality of the reconstruction is unaffected by the illumination conditions. In both cases, the EMVS method has a very high accuracy (mean relative error $\approx 5\,\%$), and also in spite of the low spatial resolution of the sensor or the lack of regularization. Moreover, observe that the accuracy is not affected by the illumination conditions. Hence, we unlocked the high-dynamic range capabilities of the sensor to demonstrate successful HDR depth estimation.

High-Speed Experiment. To show that we can exploit the high-speed capabilities of the event sensor for 3D reconstruction, we recorded a dataset with the DAVIS at 40.5 cm from the wall and moving at 0.45 m/s. This translated into an apparent speed of 376 pixels/s in the image plane, which caused motion blur in the DAVIS frames (Fig. B.12). The motion blur makes the images unintelligible. By contrast, the high temporal resolution of the event stream still accurately captures the edge information of the scene. Our EMVS method produced a 3D reconstruction with a mean depth error of 1.26 cm and a relative error of 4.84 %. The accuracy is consistent with that of previous experiments ($\approx 5\,\%$), thus supporting the remarkable performance of our method and its capability to exploit the high-speed characteristics of the event sensor.

Three-dimensional Scenes

All previous experiments were carried out with nearest-neighbor DSI voting (Section B.7.1), and lacked structure post-processing (no median or radius filters were applied). The following experiments were performed with bilinear DSI voting and structure post-processing (Section B.5.2).

Figs. B.13 and B.14 show some results obtained by our EMVS method on non-flat scenes. We show both the semi-dense point cloud and its projection on a frame (for better understanding). To ease the visualization, depth is colored from red (close) to blue (far).

In Fig. B.13, the DAVIS was moved in front of a scene containing various objects with different shapes and at different depths. In spite of the large occlusions of the distant objects, generated by the foreground objects, our EMVS algorithm was able to recover the structure of the scene reliably. Fig. B.14 shows the result of our EMVS algorithm on a larger scale dataset. The sensor was hand-held moved in a big room featuring various textured boxes. Multiple local point clouds were estimated along the trajectory, which were then merged into a global, large-scale 3D reconstruction.

Finally, Fig. B.15 shows qualitative results of our approach in various natural environments (both indoors and outdoors) and depth ranges. For each scene, we moved the event camera in a circular fashion, in order to generate events from edges in all directions. We used a visual odometry algorithm [53] on the DAVIS frames to estimate the camera motion, and used linear interpolation to provide the camera pose for each event. The DSI was sampled uniformly in inverse depth (as in Fig. B.8d) to cope with large depth variations, using between 100 and 150 depth planes. The minimum and maximum depth values were set manually, differently for each experiment to adapt better to the depth range in the scene. We used a median filter of size 15 pixels in the semi-dense depth maps. Then, in the point clouds, we used a radius filter of size equal to 5 % of the mean scene depth, and a minimum number of neighbors of $N = 4$ to remove isolated points.

Effect of Dynamic Objects

In this section, we show that the proposed method is robust to the presence of moving objects in the scene. In Fig. B.16, we compare two 3D reconstructions obtained by our method, with and without the presence of a moving, occluding object in front of the sensor, and show that they are qualitatively equivalent. Indeed, the moving object does not generate votes with a spatial persistence in the DSI, and so the votes are treated as noise and are filtered out by the Adaptive Gaussian Thresholding. In both cases, the length of the sequence of events used for reconstruction was the same, and the camera motion was very similar.

Effect of Light Changes

Due to the fact that the event camera reacts to light changes, one might think that strong temporal light changes would perturb the performance of the algorithm. In Fig. B.17, we show that this is not the case, e.g., the proposed approach is robust to strong light changes. The reason of this robustness is two-fold: (*i*) the sensor itself, thanks to its high dynamic range, is to a large extent invariant to illumination conditions (Fig. B.17b), and (*ii*) strong light changes generate a burst of events across the whole sensor, which results in simply adding a constant offset to the DSI, which does not affect the adaptive thresholding step.

B.9 Discussion

This work has focused on multi view stereo with a single moving event camera. Our goal was to show that 3D reconstruction with a single event camera is possible, and that we do not need to solve the data association problem or estimate image intensity. The results showed that (*i*) the method provides accurate results, being able to unlock the capabilities of the sensor in challenging scenarios (HDR and high-speed) where standard cameras fail, (*ii*) the method can handle inaccurate poses (the experiments with poses provided by a frame-based visual odometry algorithm show visually appealing results, which suggests that the method is robust to pose uncertainty), and (*iii*) the method is computationally efficient and can run on the CPU, without additional dedicated hardware.

The applicability of multi view stereo depends on the availability of pose information, which in the experiments was provided by an external tracking algorithm or system. However, this is not a limitation, since the method can be used in combination with an event-based motion estimation algorithm, as shown in [150], thus removing the need for an external pose estimator.

The major limitation of the proposed approach is that it provides depth values on a discrete set, thus the resolution is limited by the number of depth planes used, N_z. The computational complexity of the method is linear in the number of depth planes, $O(N_z)$, while the discretization error is proportional to $1/N_z$. Hence, there is an accuracy vs. computation effort tradeoff. However, increasing N_z does not improve the total accuracy, as shown in Fig. B.10d, since the accuracy also depends on the triangulation uncertainty. The discretization effect has also an undesirable influence when merging point clouds from different keyframes: the same 3D point may be extracted from two different DSIs, but the 3D positions may not agree since they are rounded to the position of the center of a voxel. A continuous formulation, in the form of depth filters [194, 138], where depth can have any positive real value, would be more desirable, and it is a line of future work.

Investigating methods to regularize semi-dense depth maps is also interesting and of large applicability since semi-dense depth maps are used not only in our method but also in state-of-the-art visual odometry algorithms for standard cameras, such as LSD-SLAM [46] and DSO [47]. We showed how simple processing techniques, such as median filtering, are effective tools to improve the quality of the reconstructions, but more principled methods would also be desirable.

B.10 Conclusion

We introduced the EMVS problem, and provided a simple and elegant solution to it that exploits the natural strengths of the sensor, and runs in real-time on a CPU. We validated our algorithm on both synthetic and real data, for various motions and scenes, showing very accurate 3D reconstructions (relative depth error of 5%) in spite of the low resolution of the sensor and the high amount of noise typical of event cameras. We believe this work is a major step towards building 3D reconstruction algorithms robust to speed (the events do not suffer from motion blur), and HDR illumination. This paper further highlights the potential of event cameras and the astounding possibilities it opens to computer vision.

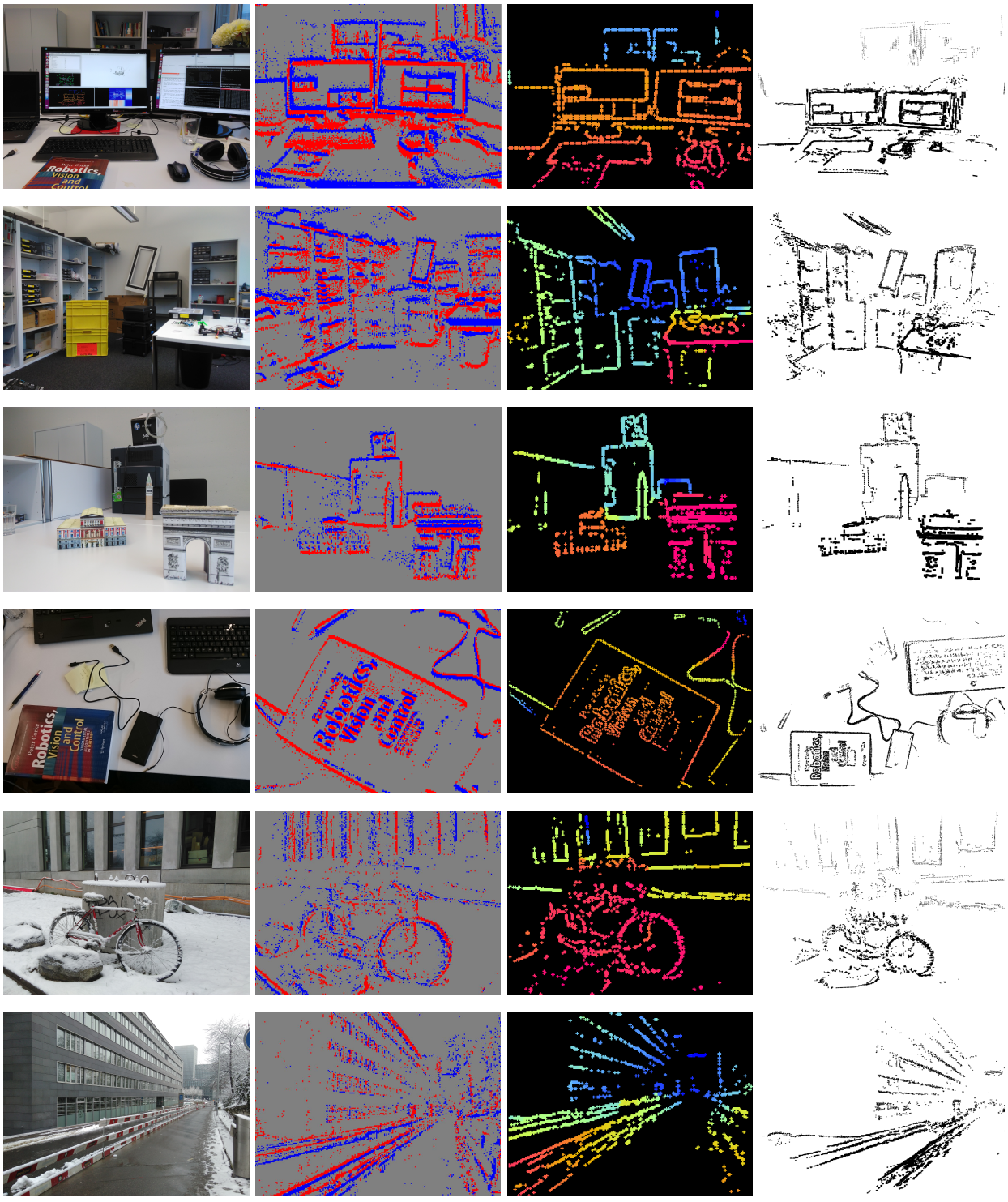
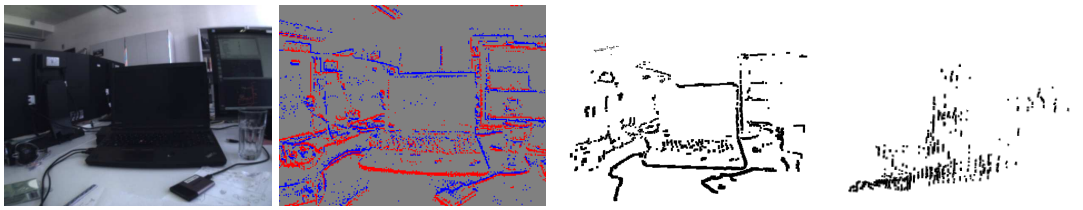
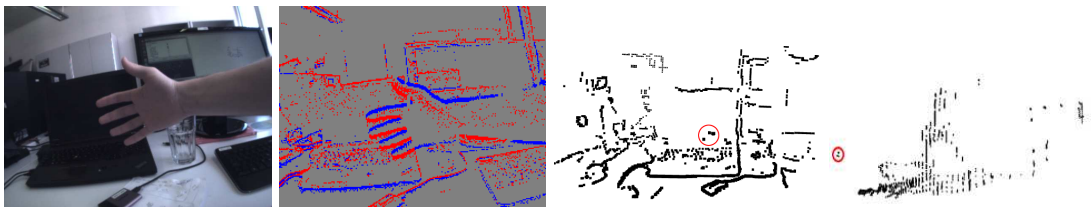


Figure B.15 – Semi-dense 3D reconstructions of several scenes with a hand-held DAVIS. (a) Scene. (b) Events (positive and negative). (c) Semi-dense depth map, pseudo-colored from red (close) to blue (far). (d) Point cloud.



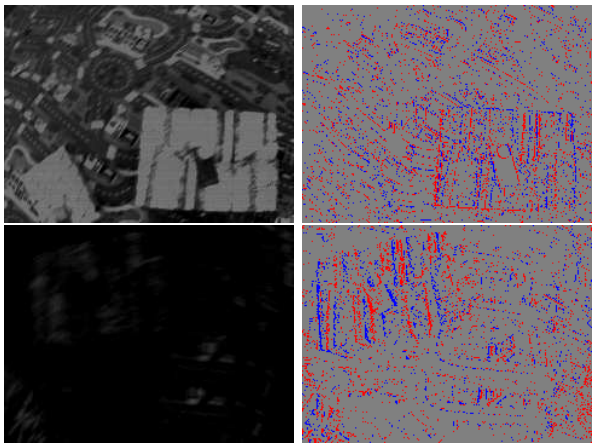
(a) Static scene. From left to right: Preview image; Preview of the events; 3D reconstruction (front view and side view).



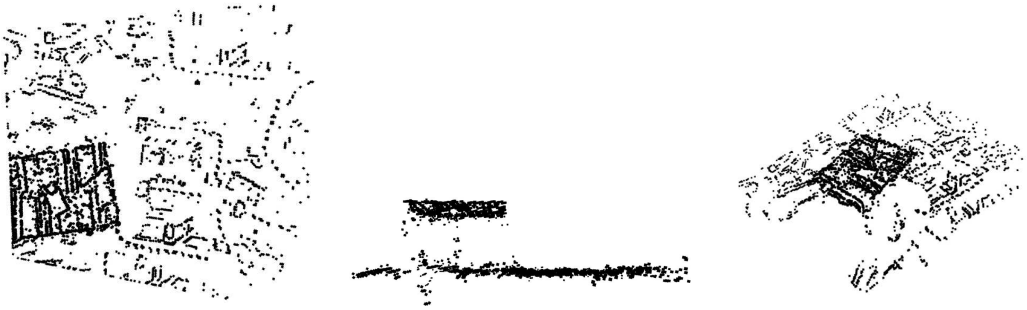
(b) Dynamic scene with hand continuously waving in front of the sensor. Apart from a small number of outlier 3D points generated by the moving hand (circled in red), our algorithm is able to reconstruct the scene as well as in the static case.

Figure B.16 – *Effect of a Dynamic Scene:* The same scene and camera motions were used to create the 3D reconstructions shown in Figs. B.16a and B.16b. However, in Fig. B.16b, a hand was continuously waived in front of the sensor, generating a large number of outlier events. Nonetheless, our algorithm is barely affected and both 3D reconstructions are similarly good.

(a) Preview of the scene.



(b) Visualization of frames from a standard camera, compared to the events. Top row: light ON; Bottom row: light OFF. The events are unaffected by the strong light change.



(c) From left to right: Top view, side view, and perspective view of the reconstructed 3D scene.

Figure B.17 – *Effect of strong light changes:* Despite switching off the light in the middle of the sequence (Fig. B.17b), the obtained 3D reconstruction remains unaffected and of high quality (Fig. B.17c).

Appendix: Relation of Area Elements due to a 2D Homography

This section provides a useful result on how a 2D transformation given by a homography affects the area element.

Result 1 (Jacobian of a Homography)**.** *Let* $\mathtt{H}$ *be a 2D homography transforming points* $\mathbf{x} \doteq (x, y, 1)^\top$ *to points* $\mathbf{x}' \doteq (x', y', 1)^\top$ *in homogeneous coordinates:* $\mathbf{x}' \sim \mathtt{H}\mathbf{x}$*, where* $\sim$ *means equality up to a non-zero scale factor. The determinant of the Jacobian of the transformation* $(x, y) \overset{\mathtt{H}}{\mapsto} (x', y')$ *(in Euclidean coordinates),*

$$\mathtt{J} \doteq \frac{\partial(x', y')}{\partial(x, y)} = \begin{pmatrix} \frac{\partial x'}{\partial x} & \frac{\partial x'}{\partial y} \\ \frac{\partial y'}{\partial x} & \frac{\partial y'}{\partial y} \end{pmatrix} \tag{B.16}$$

is

$$\det(\mathtt{J}) = \frac{\det(\mathtt{H})}{(\mathbf{e}_3^\top \mathtt{H}\mathbf{x})^3}, \tag{B.17}$$

where $\mathbf{e}_3 = (0, 0, 1)^\top$ *is the 3-rd vector of the canonical basis in* $\mathbb{R}^3$*.*

The determinant of the Jacobian (B.17) provides the relation between the area elements in (x, y) and in (x', y') according to the geometric transformation given by the homography $\mathtt{H}$,

$$dA' \doteq dx'dy' = \det(\mathtt{J})\, dxdy = \det(\mathtt{J})\, dA, \tag{B.18}$$

as illustrated in Fig. B.18.

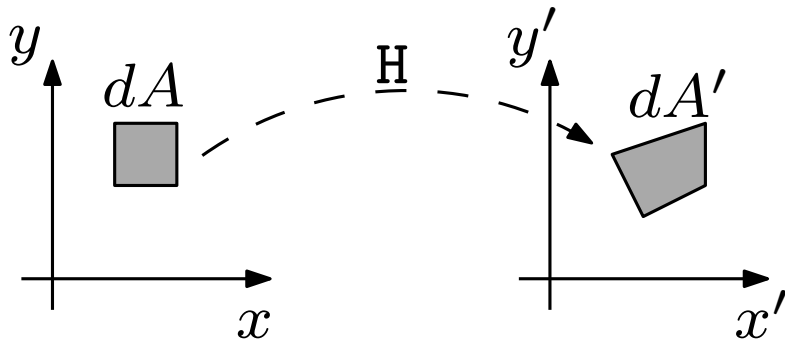


Figure B.18 – *Result 1.* A homography $\mathtt{H}$ maps points to points and lines to lines. Area elements are transformed according to $dA' = |\mathtt{J}|dA$, where $\mathtt{J}$ is the Jacobian of the homography $\mathtt{H}$.

Proof. Let $\mathtt{H} = (h_{ij})$ be the homogeneous matrix of the homography, and let $\mathbf{h}_3^\top \doteq \mathbf{e}_3^\top \mathtt{H}$

be its third row. Writing out explicitly the transformed variables

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}, \quad y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}, \tag{B.19}$$

we may compute the four elements of the Jacobian matrix (B.17):

$$\mathtt{J} = \frac{1}{\mathbf{h}_3^\top \mathbf{x}} \begin{pmatrix} h_{11} - x'h_{31} & h_{12} - x'h_{32} \\ h_{21} - y'h_{31} & h_{22} - y'h_{32} \end{pmatrix} \tag{B.20}$$

Next, we compute the determinant of this matrix. Noting that $(h_{11} - x'h_{31})(h_{22} - y'h_{32}) - (h_{12} - x'h_{32})(h_{21} - y'h_{31}) = \mathbf{x}' \cdot ((\mathtt{H}\mathbf{e}_1) \times (\mathtt{H}\mathbf{e}_2))$ is a mixed product in terms of the first two columns of $\mathtt{H}$, with $\mathbf{e}_1 = (1,0,0)^\top$ and $\mathbf{e}_2 = (0,1,0)^\top$, gives

$$\det(\mathtt{J}) = \frac{1}{(\mathbf{h}_3^\top \mathbf{x})^2} \mathbf{x}' \cdot ((\mathtt{H}\mathbf{e}_1) \times (\mathtt{H}\mathbf{e}_2)). \tag{B.21}$$

Substituting $\mathbf{x}' = \mathtt{H}\mathbf{x}/(\mathbf{h}_3^\top \mathbf{x})$ in the mixed product $\mathbf{x}' \cdot ((\mathtt{H}\mathbf{e}_1) \times (\mathtt{H}\mathbf{e}_2)) = \det(\mathbf{x}', \mathtt{H}\mathbf{e}_1, \mathtt{H}\mathbf{e}_2)$ and using the properties of the determinant, $\det(\mathtt{H}\mathbf{x}, \mathtt{H}\mathbf{e}_1, \mathtt{H}\mathbf{e}_2) = \det(\mathtt{H}) \det(\mathbf{x}, \mathbf{e}_1, \mathbf{e}_2) = \det(\mathtt{H})$, gives the desired result (B.17):

$$\det(\mathtt{J}) \overset{(B.21)}{=} \frac{\det(\mathtt{H}\mathbf{x}, \mathtt{H}\mathbf{e}_1, \mathtt{H}\mathbf{e}_2)}{(\mathbf{h}_3^\top \mathbf{x})^3} = \frac{\det(\mathtt{H})}{(\mathbf{e}_3^\top \mathtt{H}\mathbf{x})^3}. \tag{B.22}$$

□

Planar Homography

Next, we particularize the previous general Result 1 to the case of a planar homography induced by a plane in space.

Let us consider (*i*) two finite cameras (i.e., whose optical centers are not at infinity) with projection matrices given by $\mathtt{P} = (\mathtt{I}|\mathbf{0})$ and $\mathtt{P}' = (\mathtt{R}|\mathbf{t})$ in calibrated coordinates, and (*ii*) a plane not passing through the optical centers of the cameras, with homogeneous coordinates $\boldsymbol{\pi} = (a,b,c,d)^\top = (\mathbf{n}^\top, d)^\top$, where $\mathbf{n}$ is the unit normal to the plane. The optical centers of $\mathtt{P}$ and $\mathtt{P}'$ are $\mathbf{0}$ and $\mathbf{C} = -\mathtt{R}^\top\mathbf{t}$, respectively. The planar homography from the image plane of $\mathtt{P}$ to the image plane of $\mathtt{P}'$ via the plane $\boldsymbol{\pi}$, such that $\mathbf{x}' \sim \mathtt{H}\mathbf{x}$, is

$$\mathtt{H}_\pi(\mathtt{P}, \mathtt{P}') \sim \mathtt{R} - \frac{1}{d}\mathbf{t}\mathbf{n}^\top = \mathtt{R}\left(\mathtt{I} + \frac{1}{d}\mathbf{C}\mathbf{n}^\top\right), \tag{B.23}$$

where $\mathtt{I}$ is the identity matrix.

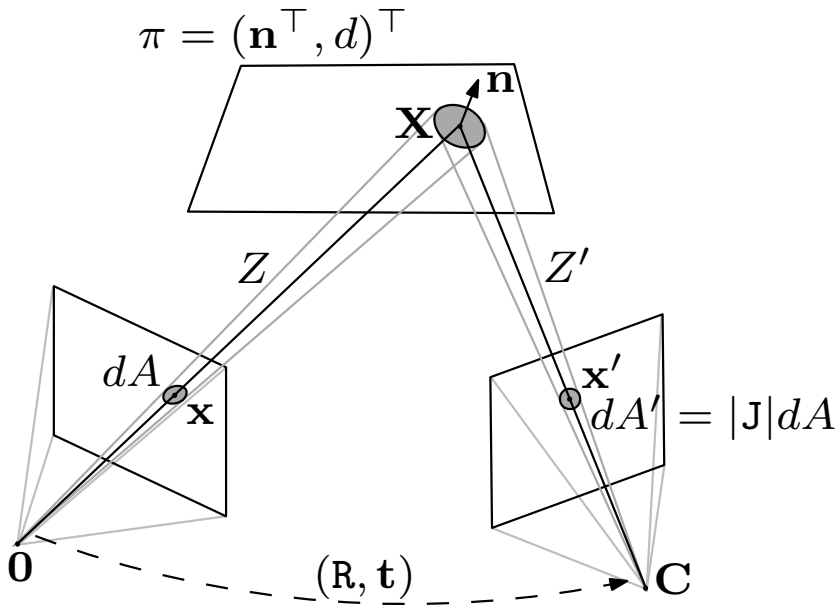


Figure B.19 – *Result 2.* Relation of area elements induced by a planar homography: $dA' = |\mathtt{J}|dA$, where $\mathtt{J}$ is the Jacobian of the planar homography, and Z, Z' are the depths of the scene point $\mathbf{X}$ with respect to the two cameras, respectively.

The planar homography from $\mathtt{P}'$ to $\mathtt{P}$ via the plane π is given by the inverse of (B.23):

$$\mathtt{H}_\pi(\mathtt{P}', \mathtt{P}) = \mathtt{H}_\pi^{-1}(\mathtt{P}, \mathtt{P}') \sim \left(\mathtt{I} - \frac{1}{d + \mathbf{n}^\top \mathbf{C}} \mathbf{C}\mathbf{n}^\top\right) \mathtt{R}^\top. \tag{B.24}$$

Result 2 (Jacobian of a Planar Homography)**.** *For a planar homography* (B.23)*, Result 1 becomes*

$$\det(\mathtt{J}) = \left(\frac{Z}{Z'}\right)^3 \left(1 + \frac{\mathbf{C} \cdot \mathbf{n}}{d}\right), \tag{B.25}$$

where Z and Z' are the depths of the point $\mathbf{X} \in \pi$, projecting on $\mathbf{x}$ and $\mathbf{x}'$, with respect to cameras $\mathtt{P}$ and $\mathtt{P}'$, respectively. This is illustrated in Fig. B.19

Proof. Let us compute the numerator and denominator of (B.17). Applying $\det(\mathtt{R}) = 1 = \det(\mathtt{I})$ and the matrix determinant lemma to (B.23) gives

$$\det(\mathtt{H}) = \det(\mathtt{R}) \det\left(\mathtt{I} + \frac{1}{d}\mathbf{C}\mathbf{n}^\top\right) = 1 + \frac{1}{d}\mathbf{n}^\top \mathbf{C}. \tag{B.26}$$

A point $\mathbf{X} \doteq (X, Y, Z)^\top$ lies on the plane π if it satisfies

$$\mathbf{n}^\top \mathbf{X} + d = 0. \tag{B.27}$$

The point $\mathbf{X}$ expressed in the frame of $\mathtt{P}'$ becomes

$$(X', Y', Z')^\top \doteq \mathbf{X}' = \mathtt{R}\mathbf{X} + \mathbf{t} = \mathtt{R}(\mathbf{X} - \mathbf{C}). \tag{B.28}$$

Since $\mathbf{x}Z = (x, y, 1)^\top Z = \mathbf{X}$ and

$$\mathtt{H}\mathbf{X} \overset{(B.23)}{=} \mathtt{R}\left(\mathbf{X} + \frac{\mathbf{X}\cdot\mathbf{n}}{d}\mathbf{C}\right) \overset{(B.27)}{=} \mathtt{R}(\mathbf{X} - \mathbf{C}) \overset{(B.28)}{=} \mathbf{X}', \tag{B.29}$$

the denominator of (B.17) is given in terms of $\mathbf{e}_3^\top \mathtt{H}\mathbf{x} \overset{(B.29)}{=} \mathbf{e}_3^\top \mathbf{X}'/Z$. Substituting this result and (B.26) in (B.17) gives (B.25). □

Acknowledgements

This research was supported by the National Centre of Competence in Research Robotics (NCCR) and the UZH Forschungskredit.

C EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-time

Reprinted, with permission, from:

H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza. "EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-Time". In: *IEEE Robot. Autom. Lett.* 2.2 (2017), pp. 593–600. DOI: 10.1109/LRA.2016.2645143

EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-time

Henri Rebecq, Timo Horstschaefer, Guillermo Gallego and Davide Scaramuzza

Abstract — We present EVO, an Event-based Visual Odometry algorithm. Our algorithm successfully leverages the outstanding properties of event cameras to track fast camera motions while recovering a semi-dense 3D map of the environment. The implementation runs in real-time on a standard CPU and outputs up to several hundred pose estimates per second. Due to the nature of event cameras, our algorithm is unaffected by motion blur and operates very well in challenging, high dynamic range conditions with strong illumination changes. To achieve this, we combine a novel, event-based tracking approach based on image-to-model alignment with a recent event-based 3D reconstruction algorithm in a parallel fashion. Additionally, we show that the output of our pipeline can be used to reconstruct intensity images from the binary event stream, though our algorithm does not require such intensity information. We believe that this work makes significant progress in SLAM by unlocking the potential of event cameras. This allows us to tackle challenging scenarios that are currently inaccessible to standard cameras.

Supplementary Material

A video showing the performance of our method in several sequences is available at: https://youtu.be/bYqD2qZJlxE

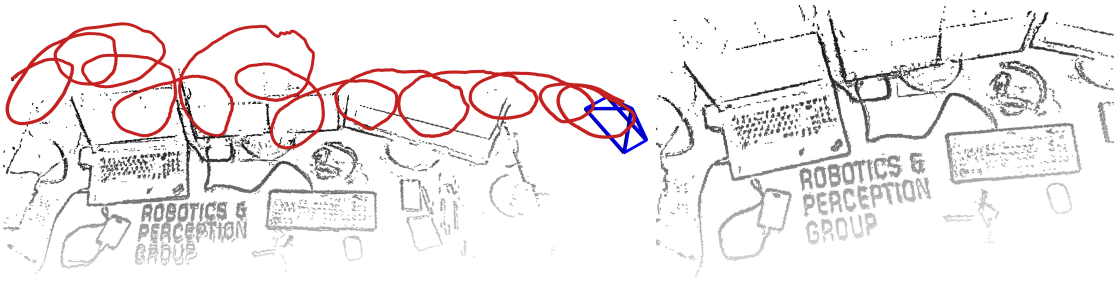


Figure C.1 – Example of trajectory and 3D map estimated by EVO using only events.

C.1 Introduction

An event camera, such as the Dynamic Vision Sensor (DVS) [98], works very differently from a traditional camera. It has independent pixels that send information (called "events") only in presence of brightness changes in the scene at the time they occur. Thus, the output is not an intensity image but a *stream of asynchronous events* at microsecond resolution. Each event consists of its space-time coordinates $< x, y, t, p >$, where p denotes the polarity of the brightness change. Since events are caused by brightness changes over time, an event camera naturally responds to edges in the scene in presence of relative motion.

Event cameras bring the following advantages to one of the core problems in robotics, that is, Simultaneous Localization and Mapping (SLAM): *low-latency, robustness, and efficiency*. (i) Notably, event cameras have negligible latency (microseconds), and so they have the potential to enable fast maneuvers of robotic platforms [123, 121], which are currently not possible with standard cameras because of the high latency of the sensing and processing pipeline (in the order of tens of milliseconds). (ii) Event cameras confer robustness to vision-based navigation in challenging conditions for standard cameras, such as high-speed motions and very high dynamic range (HDR) scenes (up 130 dB vs 60 dB of standard cameras). (iii) Event cameras have low power consumption characteristics (20 mW vs 1.5 W of standard cameras) and low bandwidth (100 kB/s on average vs 10 Mb/s for a standard camera), producing an event stream that is sparse and has no redundant data. An ideal event-based visual navigation algorithm would not process redundant data, thus it would be computationally very efficient, allowing on-board processing in real-time. However, unlocking the advantages of event cameras for SLAM and other standard vision problems is very challenging. This is due to the fact that the output of event cameras is fundamentally different from that of standard cameras, so that traditional vision algorithms cannot be applied; thus, new methods to process the data from these novel cameras must be investigated.

Reference	2D/3D	Tracking	Depth	Scene type	Event-only	Additional requirements
Cook et al. 2011 [30]	2D	✓	✗	natural	✓	rotational motion only
Weikersdorfer et al. 2013 [198]	2D	✓	✗	B&W, lines	✓	scene parallel to the plane of motion
Kim et al. 2014 [79]	2D	✓	✗	natural	✓	rotational motion only
Censi et al. 2014 [24]	3D	✓	✗	B&W	✗	requires depth from attached RGB-D sensor
Weikersdorfer et al. 2014 [196]	3D	✓	✓	natural	✗	requires depth from attached RGB-D sensor
Mueggler et al. 2014 [123]	3D	✓	✗	B&W, lines	✓	requires 3D map of lines
Gallego et al. 2016 [60]	3D	✓	✗	natural	✗	requires 3D map of the scene
Rebecq et al. 2016 [147]	3D	✗	✓	natural	✓	requires pose information
Kueng et al. 2016 [87]	3D	✓	✓	natural	✗	requires intensity images
Kim et al. 2016 [80]	3D	✓	✓	natural	✓	requires intensity reconstruction and GPU
This work	3D	✓	✓	natural	✓	

Table C.1 – Literature review on event-based methods for pose tracking and/or mapping with a single event camera. The type of motion is noted with labels "2D" (3-DOF motions, such as planar or rotational) and "3D" (free 6-DOF motion in 3D space). Note that only [80] and this work address the most general scenario ("3D" using only events).

Contribution In this paper, we tackle the problem of performing real-time, 6-DOF parallel tracking and mapping with an event camera in natural scenes. Over the past few years, several works have made contributions towards this goal; some approaches studied motion estimation from known 3D maps, while others worked on 3D mapping from known poses (see more in Section C.2). Preliminary work to solve this very challenging problem was recently proposed in [80] using Bayesian filtering and variational methods. However, the approach required the estimation of the image intensity and regularization of the depth, thus demanding dedicated hardware, such as a GPU, in order to run in real time. Additionally, no quantitative evaluation of the method was provided. The method we propose—which we call EVO (Event-based Visual Odometry)—is purely geometric and, in contrast to [80], does not require the estimation of the image intensity, thus (*i*) it prevents the propagation of errors from such estimation and (*ii*) it is computationally efficient (it can run in real time on a standard CPU). Our contributions are: (*i*) a novel event-based tracking approach based on image-to-model alignment using edge maps (Sections C.3.1, C.4) and (*ii*) its integration with a recent event-based 3D reconstruction algorithm [147] to produce the first parallel tracking and mapping pipeline for event cameras that runs in real-time on the CPU. EVO can estimate up to several hundreds of poses per second while recovering a semi-dense, 3D map of the environment. As a by-product, we also show that the output of our algorithm can be used, if desired, to recover image intensity. Additionally, we provide a quantitative evaluation in challenging sequences, both indoors and outdoors, which show how we unlock the potential of event cameras.

C.2 Related Work

Solving the event-based SLAM problem in its most general setting (6-DOF motion and natural 3D scenes) with a single event camera is a challenging problem. Historically, this problem has been addressed step-by-step in scenarios with increasing complexity.

Three complexity axes can be identified: dimensionality of the problem, type of motion and type of scene. The literature is dominated by methods that address the localization subproblem first (i.e., motion estimation) since it has fewer degrees of freedom to estimate (i.e., lower dimensionality) and data association is easier than in the mapping subproblem or the combined tracking and mapping problem. Regarding the type of motion, solutions for constrained motions, such as rotational or planar (both being 3-DOF), have typically been investigated before addressing the most complex case of a freely moving camera (6-DOF). Solutions for artificial scenes in terms of photometry (high contrast) and/or structure (line-based or 2D maps) have been proposed before focusing on the most difficult cases: natural scenes (3D and with arbitrary photometric variations). Finally, some methods are not solely based on events but require additional sensing (e.g., grayscale or RGBD) to reduce the complexity of the problem. This, however, has the drawback of introducing the same bottlenecks that exist in standard frame-based systems (e.g, latency and motion blur). Table C.1 classifies the related work according to the complexity axes mentioned.

Let us focus on references that address the tracking-and-mapping problem. Cook et al. [30] proposed a generic message-passing algorithm within an interacting network to jointly estimate ego-motion, image intensity and optical flow from events. However, the system was restricted to rotational motion and did not account for translation and depth.

An event-based 2D SLAM system was presented in [198]. However, the system design was limited to planar motions (i.e., 3-DOF) and planar scenes parallel to the plane of motion consisting of artificial B&W line patterns. The method was extended to 3D in [196] but relied on an external RGB-D sensor attached to the event camera for depth estimation. The depth sensor introduces bottlenecks, which deprives the system of the low latency and high-speed advantages of event cameras.

A filter-based system to estimate the 3D orientation of an event camera while generating high-resolution panoramas of natural scenes was presented in [79]. The system was limited to rotational motion, thus ignoring translation and depth.

A visual odometry system operating in a parallel tracking-and-mapping manner was presented in [87]. The system recovered 6-DOF motions in natural scenes by tracking a sparse set of features using the event stream. However, the system required intensity images to first detect the features. Finally, [80] proposed a system with three interleaved probabilistic filters to perform pose tracking as well as depth and intensity estimation in natural scenes. However, evaluations were only qualitative and the system required the estimation of the intensity. Moreover, it was computationally expensive, requiring a GPU to reconstruct the image intensity and regularize the depth in real time. Thus, it is not suitable for computationally limited platforms.

The system we present in this paper, EVO, tackles the event-based 3D SLAM problem for 6-DOF motions and natural scenes, and does not rely on any external sensor. That is, we address the most general scenario, as [80] (see Table C.1). However, contrarily to [80], our geometric approach is computationally efficient (runs in real-time on the CPU in case of moderate motions), and does not need to recover image intensity to estimate depth, which prevents propagation of errors from such estimation (although we show that intensity images can be recovered, if desired, using the output of our algorithm).

C.3 Event-based Parallel Tracking and Mapping

The core of EVO consists of two interleaved tracking and mapping modules (blocks marked with dashed lines in Fig. C.2), thus following the separation principle of SLAM systems, such as PTAM [82]. The tracking module estimates the 6-DOF pose of the event camera using the event stream, assuming that a semi-dense 3D map of the environment is given. The mapping module expands the semi-dense 3D map as new events are triggered, assuming that pose information is given. Both modules operate in parallel, each relying on the output of the other. We first present each module separately and then describe the way in which they are combined, as in Fig. C.2.

C.3.1 Pose Tracking

Our tracking module relies on image-to-model alignment, which is also used in frame-based, direct VO pipelines [53, 46]. In these approaches, a 3D rigid body warp is used to register each incoming intensity image to a keyframe. They minimize the photometric error on a set of selected pixels whose 3D correspondences in the scene have already been established.

We follow the same global image alignment strategy, but, since event cameras naturally respond to edges in the scene, we replace the photometric error by a geometric alignment error between two edge images (see Eq. (C.1)). The two images involved in the registration process are (see Fig. C.3): an *event image* I, obtained by aggregating a small number of events into an edge map, and a *template* M, which consists of the projected semi-dense 3D map of the scene according to a known pose of the event camera.

Registration is done using the inverse compositional Lucas-Kanade (LK) method [8, 32], by iteratively computing the incremental pose $\Delta \mathrm{T}$ that minimizes

$$\sum_{\boldsymbol{u}} \Big(M\Big(\mathbf{W}(\boldsymbol{u}; \Delta \mathrm{T})\Big) - I\Big(\mathbf{W}(\boldsymbol{u}; \mathrm{T})\Big)\Big)^2, \tag{C.1}$$

and then updating the warp $\mathbf{W}$, which leads to the following update of the rigid-body

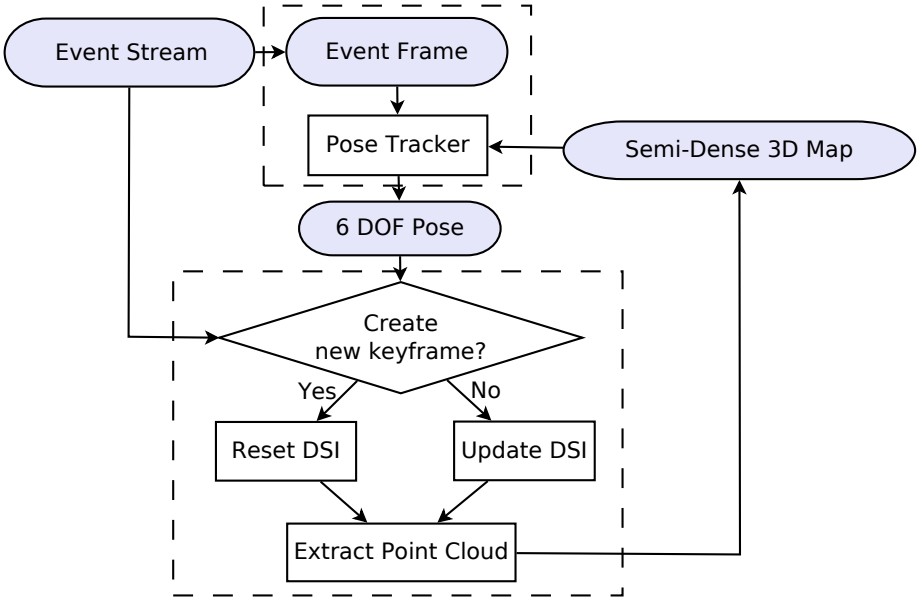


Figure C.2 – Block diagram of our event-based parallel tracking and mapping method, EVO. "DSI" denotes the Disparity Space Image, as described in Sec. C.3.2.

transformation $\mathtt{T}$ from the frame of M to the frame of I:

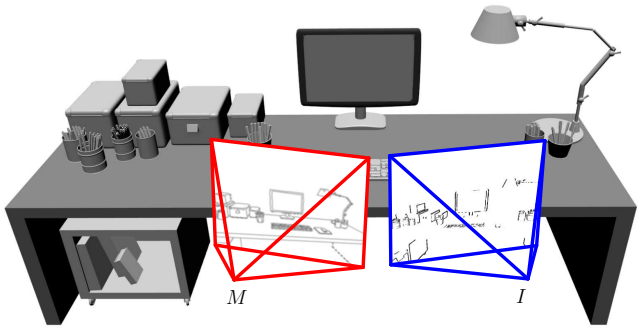
$$\mathtt{T} \leftarrow \mathtt{T} \cdot (\Delta \mathtt{T})^{-1}. \tag{C.2}$$

In the inverse approach (C.1), the projected map M is warped until it is aligned with the warped event image given by the current estimate of the registration transformation $\mathtt{T}$. The 3D rigid-body warp $\mathbf{W}$ is defined by

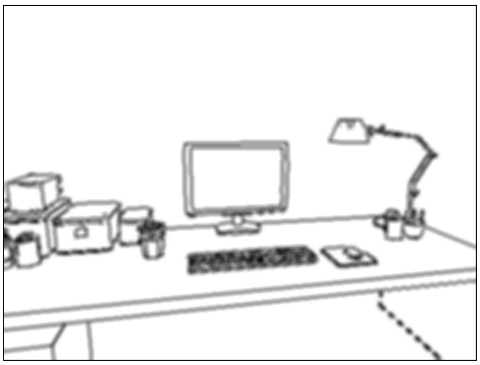
$$\mathbf{W}(\boldsymbol{u}; \mathtt{T}) := \boldsymbol{\pi}(\mathtt{T} \cdot \boldsymbol{\pi}^{-1}(\boldsymbol{u}, d_{\boldsymbol{u}})), \tag{C.3}$$

where $\boldsymbol{u}$ is a point in the image plane of M, $\mathtt{T}$ is a rigid-body transformation, $\boldsymbol{\pi}$ and $\boldsymbol{\pi}^{-1}$ denote the camera projection and inverse projection, respectively, and $d_{\boldsymbol{u}}$ is the known depth of the 3D point projecting on pixel $\boldsymbol{u}$. Hence, the sum in (C.1) is over all candidate pixels $\boldsymbol{u}$ in the domain of M for which there is an associated depth estimate $d_{\boldsymbol{u}}$. The 3D rigid-body warp is defined so that $\mathbf{W}(\boldsymbol{u}; \mathtt{Id}) = \boldsymbol{u}$ is the identity, as required in [8]. Rigid-body transformations are parametrized using twist coordinates [105]: $\boldsymbol{\xi} \in \mathbb{R}^6$, with $\mathtt{T} = \exp(\hat{\boldsymbol{\xi}}) \in SE(3)$ and Lie algebra element $\hat{\boldsymbol{\xi}} \in \mathfrak{se}(3)$.

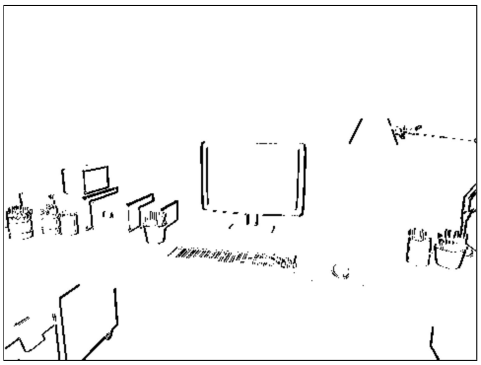
Since both I and M carry information about edges, the objective function (C.1) can be interpreted as a measure of the registration error between two edge maps: the measured one using the events and the predicted one from the projection of the 3D edge map. Due to the principle of operation of the event camera, the event image I captures all edges except those parallel to the apparent motion.



(a) 3D scene and poses involved in the registration process.



(b) Projected semi-dense map M



(c) Event image I

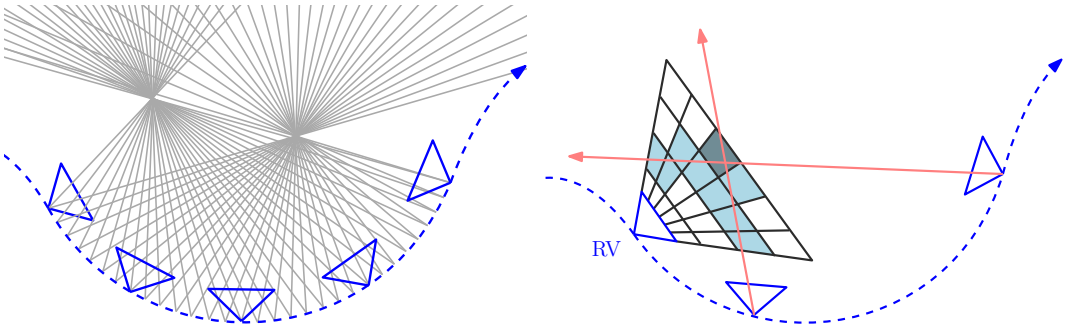
Figure C.3 – *Pose Tracking* computes the pose of the camera with respect to a reference pose by aligning the event image I with the projected semi-dense map M. Edges parallel to the camera motion are not captured by the event sensor.

The inverse compositional LK method has the advantage of low computational complexity with respect to other LK formulations [8]: the derivatives that depend on M can be pre-computed since M remains constant during the iteration. Additionally, these computations can be re-used for aligning multiple event images I with respect to the same M.

For efficiency, we use analytical derivatives of the error function (C.1), which involve, by the chain rule, computing the gradient ∇M and the derivative of the warping function with respect to the exponential coordinates of the unknown incremental pose $\Delta \mathrm{T}$. Using calibrated coordinates and assuming that lens distortion has been removed, $\boldsymbol{x} = (u, v)^\top \equiv \boldsymbol{K}^{-1}\boldsymbol{u}$, the latter derivative is given by the interaction matrix [31]

$$\mathbf{W}' = \begin{pmatrix} \frac{-1}{d_u} & 0 & \frac{u}{d_u} & uv & -(1+u^2) & v \\ 0 & \frac{-1}{d_u} & \frac{v}{d_u} & 1+v^2 & -uv & -u \end{pmatrix}. \tag{C.4}$$

Finally, the poses T obtained upon convergence of the LK method (C.2) are filtered using an average filter to get a smoother trajectory of the event camera.



(a) As the event sensor moves, events are triggered by edges in the scene. Back-projection of the events provides rays through space. The regions of high ray density mark the candidate locations of 3D edges.

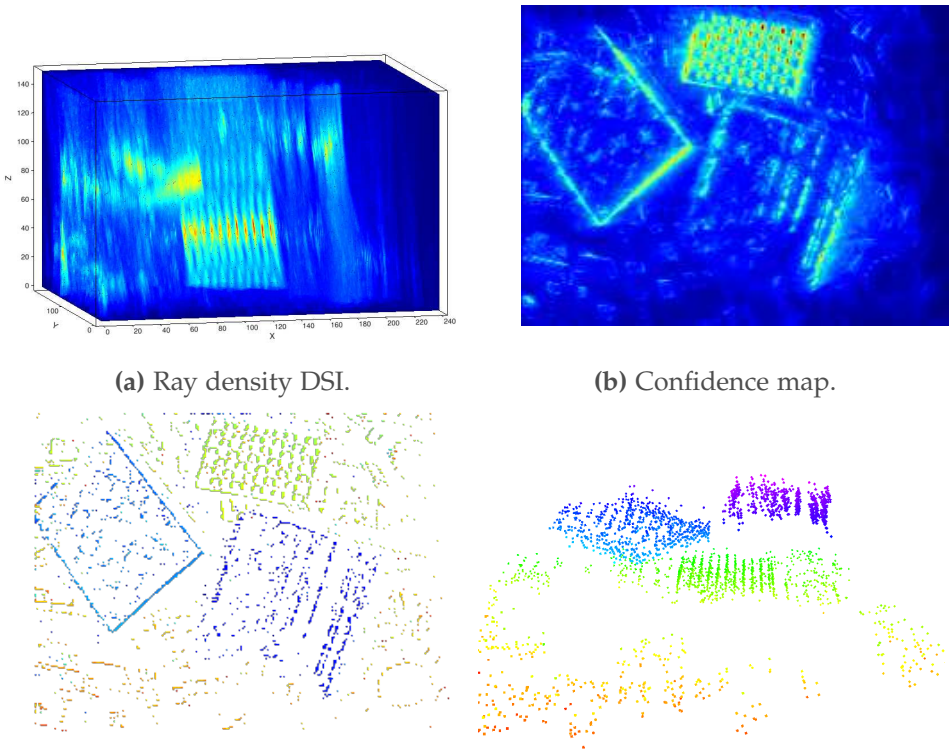
(b) Space is discretized using a voxel grid (DSI) centered at a virtual camera in a reference viewpoint (RV). Each voxel value (in blue) is the number of back-projected events (red rays) traversing it.

Figure C.4 – *Mapping*: the main idea of the EMVS [147] method used in our mapping module of Fig. C.2. Images courtesy of [147].

C.3.2 Mapping

EVO's mapping module uses the events and their corresponding camera poses to update a local semi-dense 3D map (see Fig. C.2). Our approach is based on the Event-based Multi-View Stereo (EMVS) method recently proposed in [147], which is a purely geometric approach to 3D reconstruction with a single event camera. It leverages the fact that event cameras naturally respond to edges to recover semi-dense 3D information from the event stream, without requiring intensity information or explicit data association. The main idea behind this method is illustrated in Fig. C.4a: the rays back-projected from the events highlight the regions of space where 3D edges are likely to occur. Poses for event back-projection are obtained by interpolating the tracked poses at the timestamps of the events.

The EMVS method discretizes space into a projective voxel grid (called Disparity Space Image–DSI) centered at a chosen reference viewpoint and counts the number of rays intersecting each voxel in the grid (Fig. C.4b). The local maxima of the DSI yield a point cloud of locations where 3D edges are most likely to occur. The point cloud is extracted in two steps (Fig. C.5): first, the DSI (Fig. C.5a) is collapsed into a depth map and an associated 2D confidence map (Fig. C.5b), and second, the confidence map is adaptively thresholded to keep the most confident local maxima of the DSI, yielding a semi-dense depth map (Fig. C.5c) that is finally converted into a point cloud (Fig. C.5d).



(a) Ray density DSI. **(b)** Confidence map.

(c) 2D semi-dense depth map. **(d)** 3D point cloud.

Figure C.5 – *Mapping*: the EMVS method [147] builds the ray density DSI (a), from which a confidence map (b) and a semi-dense depth map (c) are extracted in a virtual camera. The semi-dense depth map gives a point cloud of scene edges (d). Images courtesy of [147].

C.3.3 Parallel Tracking and Mapping

Let us describe how the tracking and mapping modules are interleaved in EVO to track the camera pose while continuously updating and expanding the map of the environment.

We follow a keyframe-based approach, where keyframe (KF) poses are selected along the event camera trajectory. A new local map of the scene that is used for tracking is built upon the creation of a keyframe (Fig. C.2). Hence, a KF should be created when the map is required to be expanded, that is, when the existing map turns insufficient for tracking. Our system creates a KF whenever the distance between the current camera pose and the last KF, divided by the mean scene depth, reaches a threshold (e.g., 15%). This ensures that the map is updated regularly as the event camera moves.

The creation of a KF triggers the creation of an associated local map from that viewpoint. The mapping thread processes the request ("Reset DSI" in Fig. C.2) while the tracking thread continues to track using the map from the previous KF, and switches to the new

map as soon as it is available. The DSI is re-computed using the last 2 million events. The map associated to a KF is refined as new events are triggered: every incoming event is used to update the current KF DSI, and every 100 thousand events a refined point cloud that replaces the current one for tracking purposes is extracted from the DSI ("Update DSI" in Fig. C.2).

C.3.4 Bootstrapping

To bootstrap the algorithm, we estimate an initial trajectory and 3D map during the first τ seconds as follows: first, we run the pose tracker up to time τ assuming that the local scene is planar and fronto-parallel to the sensor, and then we run the mapper to compute an initial 3D map using all the events and poses estimated by the tracker up to time τ. We observed that this strategy, despite its simplicity, is often sufficient to correctly bootstrap the system.

C.3.5 Intensity Image Reconstruction

EVO does not require intensity reconstruction to operate. Nevertheless, we show how the output of EVO can be used to generate intensity images of the mapped scene. To this end, we extended the intensity reconstruction algorithm from [79] (originally developed to reconstruct a panoramic image from purely rotational motions) to work with 3D scenes and 6-DOF motions. In essence, intensity reconstruction is based on the linearized generative model for the event camera, which combines the constant brightness assumption with the operating principle of event cameras (an event is triggered when the change in log-intensity L reaches the contrast threshold, $\Delta \log L = C$):

$$-\langle \nabla L, \dot{\mathbf{u}} \Delta t \rangle = C, \tag{C.5}$$

where all the quantities needed are known except for the gradient of the image (∇L) at a given point. Once the gradient image ∇L has been estimated, Poisson reconstruction is used to obtain the intensity image L. We refer to [79, 58] for more details. Note that the resulting images exhibit super-resolution and high-dynamic range (HDR) properties, and do not suffer from motion blur.

C.4 Implementation Details

In this section, we describe the implementation details of the proposed pipeline, EVO. The reader who is not interested in the details can jump directly to Section C.5.

C.4.1 Pose Tracking

Creation of alignment images M and I

The projection of the semi-dense, 3D map onto the reference pose yields a binary image M: pixels where map points project are set to 1; the rest, to zero. To use it in the LK method and increase the basin of attraction of the minimizer of the error function, we smooth M using a Gaussian filter with a small standard deviation $\sigma = 0.8$ pixels. The smoothing of the projected map image gives an intensity distribution that approximates the distance function to the actual projected edge map.

The event image I is binary, built by collecting a varying number of events N_e: pixels where events fired are set to 1; otherwise they are set to zero. We adapt the size N_e of the observation window to the scene: N_e is a fraction (e.g., 70 %) of the number of points in the current 3D map. This gives an estimate of the number of events that need to be considered to produce a reasonably complete edge map of the scene while collecting no more than one event per pixel. On average, we accumulate about 2000 events, which translates into a time span of 0.6–2 ms, given an event rate of 1–3 million events/s.

Moreover, we use a sliding window on the event stream to build the event image I. The shift S_e (in number of events) between the first events of adjacent windows allows us to control the rate of the computed poses: we may reuse events for several pose estimates or skip events to speed up tracking.

Tracking Improvements

To speed up tracking, we sub-sample the number of candidate pixels $\boldsymbol{u}$ in (C.1) in each iteration. We use the *stochastic gradient descent*, which means we divide the set of pixels $\{\boldsymbol{u}\}$ into random subsets of a given *batch size*. Then, for each iteration of the LK method we only process a single batch instead of the complete set $\{\boldsymbol{u}\}$. The batch size is around 300–500 pixels. A typical reference image M has around 10 k non-zero pixels, which gives 20–30 batches. To increase tracking precision, we use up to 5 optimization iterations (called *epochs*), meaning that we repeat the described process several times. For comparison, SVO [53] uses up to 30 iterations. This allows us to compute more than 500 poses per second on a single CPU. Since the temporal resolution of the event camera is very high and few events are accumulated per event image I, two consecutive event images I_k, I_{k+1} are very close in time and alike. In fact, the displacement between corresponding points in the images is typically less than one pixel, hence the pose of I_{k+1} is very close to the known pose of I_k, and therefore, there is no need to use a coarse-to-fine alignment approach, which is typically used for larger inter-image motions.

(a) *desk* dataset **(b)** *multi-keyframe* dataset

Figure C.6 – Impressions of the datasets recorded with the motion capture system

C.4.2 Mapping

Inverse depth

The main difference with respect to the original method of [147] is that, to allow for mapping far away objects, we discretize the DSI volume using depth planes uniformly spaced in inverse depth. We typically use a DSI with 50 depth planes, and adapt the depth range (minimum and maximum depths) to the characteristics of the scene.

Noise reduction

We apply a median filtering (typical size: 15×15 pixels) on the resulting semi-dense depth map (we only consider pixels with associated depth values in the computation of the median) to reduce noise while keeping valuable 3D structure. We also apply a radius filter [168] to the resulting point cloud to remove isolated 3D points, which most likely correspond to noise.

C.5 Experiments

In this section, we assess the accuracy of EVO both quantitatively and qualitatively on different challenging sequences. The results show that EVO produces reliable and accurate pose tracking even in conditions where VO with standard cameras fails. The event camera used to acquired the datasets is the DAVIS [18], which has a spatial resolution of 240×180 pixels, a temporal resolution in the order of microseconds and a very high dynamic range (130 dB). The DAVIS combines in the same pixel array an event sensor and a standard, frame-based sensor. EVO uses only the event stream; the frames of the DAVIS are shown in the following figures only for illustration purposes.

C.5.1 Accuracy Evaluation

To illustrate the robustness and accuracy of EVO, we evaluate it on two different sequences with ground truth recorded using a motion capture system (Optitrack): the first one features a single-keyframe trajectory characterized by fast, aggressive motion (up to 780 °/s) in an office environment (Fig. C.6a); the second sequence (Fig. C.6b) features a multi-keyframe trajectory. Both trajectories show strong illumination changes generated by switching the room lights off and on.

The two datasets we used to evaluate the performance are:

Single-keyframe trajectory, aggressive motion

This sequence contains aggressive 6-DOF motion. The results are presented in Fig. C.7. The times at which the room lights are switched off and on again are marked. EVO is able to track the whole sequence with high accuracy (2° rotation error and 2 cm translation error on average, over a 35 m trajectory, that is 0.057 % relative position error).

Multi-keyframe trajectory

The camera is moved along a long trajectory over a scene containing several boxes. The results are presented in Figs. C.8 and C.9. The times at which the room lights are switched off and on again are marked. As observed, EVO tracks the whole sequence with remarkable accuracy (6 cm drift in translation, and a few degrees (3°) in rotational drift, over a 30 m trajectory, that is, 0.2 % relative position error). Note that we do not perform any map or pose refinement (e.g., bundle adjustment); doing so would further reduce the drift.

C.5.2 Computational Performance

Since event cameras respond to visual changes in the scene, their output rate is not constant but rather a complex function of the apparent motion of the scene, the amount of texture, the sensor parameters, etc. Table C.2 reports the event rate and EVO performance analysis in two typical scenarios: moderate- and high-speed motion on the desk sequence. On a standard laptop (Intel Core i7-4810MQ CPU @ 2.80GHz), our implementation of EVO is able to process 1.5 million events/s on average, hence it can handle sequences with moderate speed in real time (see attached video). More precisely, we define the "real-time factor" (RTF) as the number of events processed per second divided by the incoming event rate (so that a factor above 1 means that the algorithm is faster than real-time). As shown in Table C.2, EVO is 1.25 to 3 times faster

	Moderate speed	High speed
Event rate (million ev/s)	$0.5 - 1.2$	$2 - 3$
Linear velocity (m/s)	$0.0 - 0.5$	$0.1 - 2.5$
Angular velocity (deg/s)	$0.7 - 155$	$20 - 780$
Real-time factor (RTF)	$1.25 - 3$	$0.5 - 0.75$

Table C.2 – Performance of EVO on a laptop computer in two typical scenarios: moderate and high speed, on the desk sequence (mean depth: 1 m). Our implementation can process approximately 1.5 million events/s.

than real time for moderate speed scenarios while it is, at worse, two times slower than real time for high-speed motions.

C.5.3 Experiments in Outdoor Environment

Outdoor sequence with aggressive motion

This sequence was recorded outdoors, in a busy street and features aggressive motions. Additionally, there are several moving elements in the scene (pedestrians, bicycles, etc.) generating outlier events. The sequence is shown in Fig. C.11. For illustration, we also display an image reconstructed using the output of our pipeline. See also the video attached to this paper for further impressions of this dataset. Since a motion-capture system is not available outdoors, we used a state-of-the-art VO method (SVO [53]) on the intensity frames of the DAVIS for comparison (Fig. C.10).

Outdoor sequence, pointing at the sun

To highlight further the potential of EVO, we show another outdoor sequence where we pointed the event camera directly at the sun, and used a road sign to cover/uncover the sun multiple times as the camera moved (Fig. C.12). EVO was able to track the motion successfully despite the considerable dynamic range of the scene (see attached video).

C.5.4 Discussion

Our method provides joint estimation of depth and 6-DOF motion with pure event data, in natural scenes. EVO is very accurate, even in very challenging scenarios, currently inaccessible to VO algorithms for standard cameras (e.g., aggressive motion, abrupt changes of illumination, high dynamic range scenes). It is lightweight enough to run in real-time on computationally constrained platforms.

Our method is purely geometric (with building blocks such as event back-projection

and edge-map registration) and exploits the natural strengths of event cameras as *moving edge detectors*, both in tracking (edge-map registration), and in mapping (structure extraction by edge back-projection from multiple viewpoints). Hence, intensity reconstruction is not needed for accurate tracking and mapping. This eliminates a source of errors and increases the speed of map convergence.

Unlike traditional visual odometry systems, EVO does not explicitly solve the data association problem, neither in the tracking nor in the mapping part. In the tracking part, we solve the data association indirectly in three steps: (*i*) we create an intermediate representation in the form of an edge-like image by accumulating events; (*ii*) we borrow the implicit pixel-to-pixel data association typical of photometric image alignment methods; (*iii*) we sparsify the representation using random sampling (inspired by stochastic gradient descent). This increases the speed of the tracker and improves the robustness to occlusions.

While the speed of convergence of the map is relatively fast (it typically takes ~ 2 million events, which corresponds roughly to 1-4 seconds, depending on the event rate), it is still slow compared to the velocities that the tracker can cope with, and thus limits the speed of the whole VO pipeline in a scenario where multiple keyframes would be required. This is however not a problem for many applications.

C.6 Conclusions

We have presented EVO, an event-based visual odometry pipeline that successfully leverages both the high temporal resolution and high dynamic range capabilities of event cameras. We have shown that by extracting only geometric information from the event stream we are able to compute the position and orientation of the camera with high precision ($\leq 0.2\,\%$ position error and $\leq 3^\circ$ orientation error), as well as obtain a semi-dense 3D map of the environment. The method is very efficient and runs in real-time on the CPU of a conventional laptop or, with reduced precision, even on mobile platforms as commonly seen on MAVs.

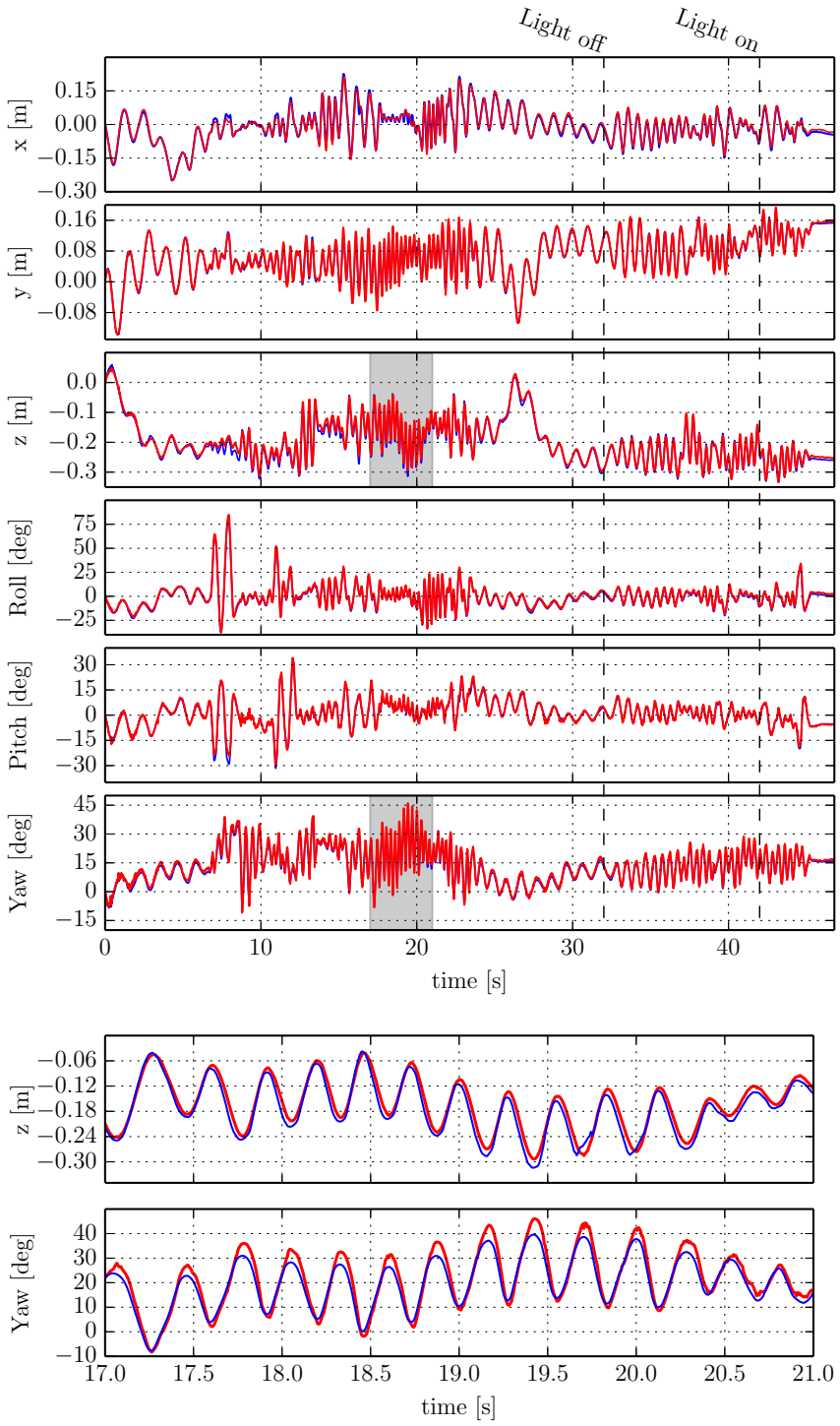


Figure C.7 – *Single-keyframe trajectory, aggressive motion. Top:* Estimated trajectory (blue) compared against ground truth (red): EVO can track aggressive motion with remarkable accuracy. The dashed lines mark the times at which the room lights were switched off and on again in order to generate strong illumination changes. *Bottom:* Zoom on the highlighted regions of z and *yaw*.

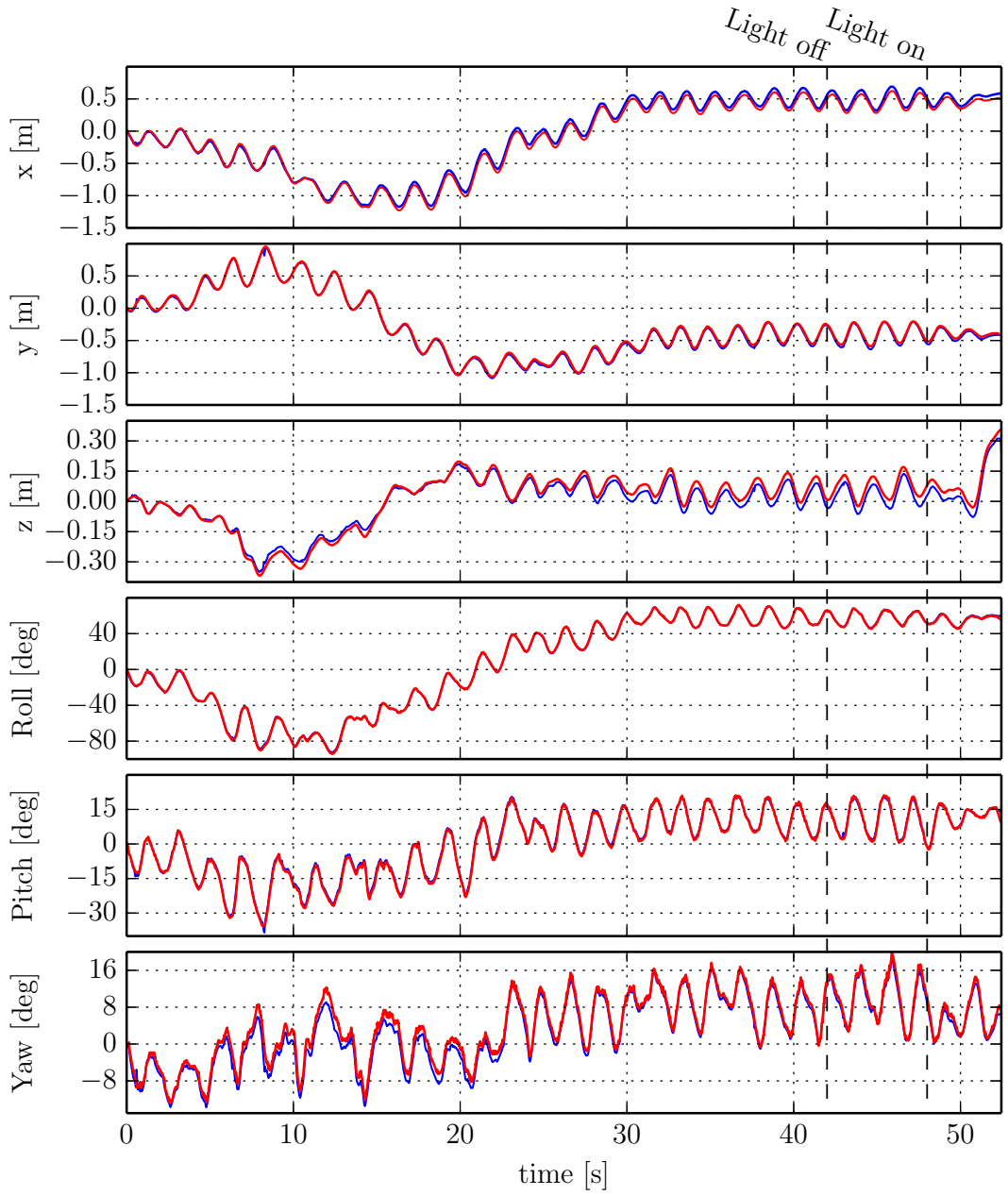


Figure C.8 – *Multi-keyframe trajectory.* Estimated trajectory (blue) compared against ground truth (red): EVO can track long trajectories without much drift. The average drift over the 30 m trajectory is 6 cm. The dashed lines mark the times at which the room lights were switched off and on again in order to generate strong illumination changes.

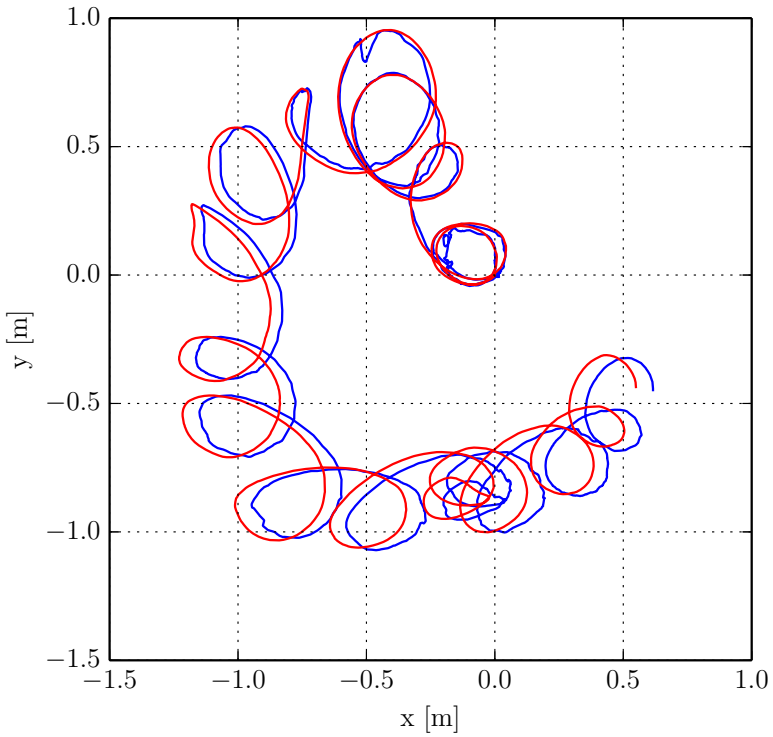


Figure C.9 – *Multi-keyframe trajectory*. Top view of the estimated trajectory (blue) against ground truth (red).

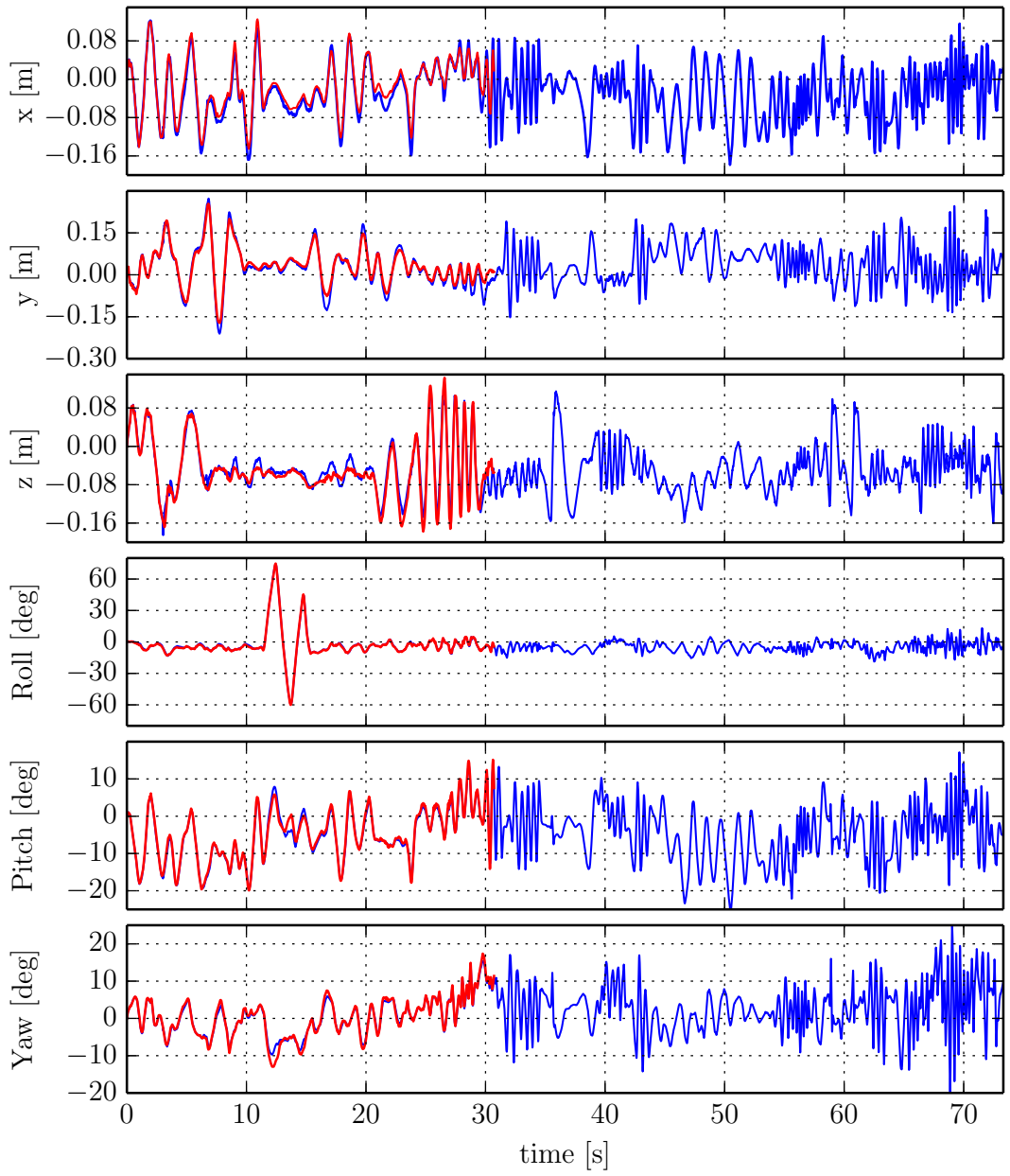


Figure C.10 – EVO (blue) vs. SVO (red) on *outdoor* dataset. At $t \approx 30\,\text{s}$, the frame-based VO system (SVO) fails because of the aggressive motion while EVO keeps tracking until the end of the trajectory.

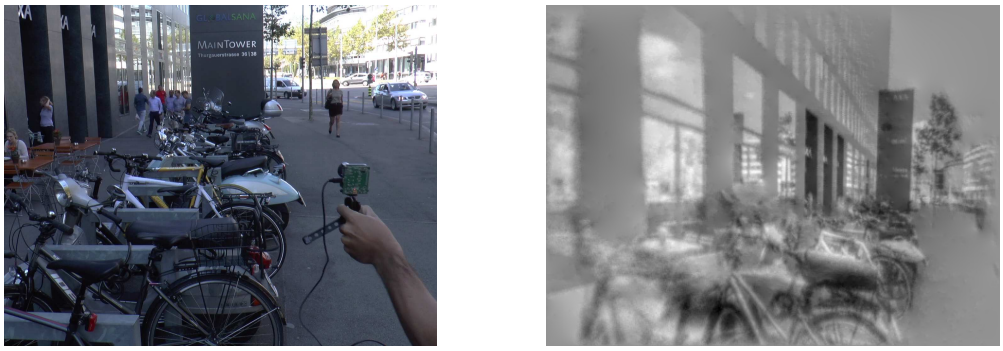


Figure C.11 – Impressions of the *outdoor* dataset. Left: image from the DAVIS. Right: reconstructed image using the output of EVO. Observe that the reconstructed image is not over- or under-exposed.

Figure C.12 – Outdoor sequence, pointing at the sun. Left: View from a standard camera, showing severe under-exposure on the foreground. Middle: Frame from the DAVIS, showing severe under- and over-exposed areas. Right: HDR image reconstructed using the output of EVO.

D Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization

Reprinted, with permission, from:

H. Rebecq, T. Horstschaefer, and D. Scaramuzza. “Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization”. In: *British Mach. Vis. Conf. (BMVC)*. 2017

Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization

Henri Rebecq, Timo Horstschaefer and Davide Scaramuzza

Abstract — Event cameras are bio-inspired vision sensors that output pixel-level brightness changes instead of standard intensity frames. They offer significant advantages over standard cameras, namely a very high dynamic range, no motion blur, and a latency in the order of microseconds. We propose a novel, accurate tightly-coupled visual-inertial odometry pipeline for such cameras that leverages their outstanding properties to estimate the camera ego-motion in challenging conditions, such as high-speed motion or high dynamic range scenes. The method tracks a set of features (extracted on the image plane) through time. To achieve that, we consider events in overlapping spatio-temporal windows and align them using the current camera motion and scene structure, yielding motion-compensated event frames. We then combine these feature tracks in a keyframe-based, visual-inertial odometry algorithm based on nonlinear optimization to estimate the camera's 6-DOF pose, velocity, and IMU biases. The proposed method is evaluated quantitatively on the public Event Camera Dataset [124] and significantly outperforms the state-of-the-art [211], while being computationally much more efficient: our pipeline can run much faster than real-time on a laptop and even on a smartphone processor. Furthermore, we demonstrate qualitatively the accuracy and robustness of our pipeline on a large-scale dataset, and an extremely high-speed dataset recorded by spinning an event camera on a leash at 850 deg/s.

Multimedia Material

Multimedia Material. A supplemental video for this work is available: https://youtu.be/F3OFzsaPtvI

D.1 Introduction

Event cameras, such as the Dynamic Vision Sensor (DVS) [98], work very differently from a traditional camera. They have *independent* pixels that only send information (called "events") in presence of brightness changes in the scene at the time they occur. Thus, the output is not an intensity image but a stream of asynchronous events at microsecond resolution, where each event consists of its space-time coordinates and the *sign* of the brightness change (i.e., no intensity). Event cameras have numerous advantages over standard cameras: a latency in the order of microseconds, low power consumption, and a very high dynamic range (130 dB compared to 60 dB of standard cameras). Most importantly, since all the pixels are independent, such sensors do not suffer from motion blur.

The task of estimating a sensor's ego-motion from a combination of images and measurements from an Inertial Measurement Unit (IMU), called Visual-Inertial Odometry (VIO), has important applications in various fields, for example augmented/virtual reality (AR/VR) applications. VIO has been thoroughly studied in the past decades, and is today a mature research field [21]. State-of-the-art VIO pipelines have shown impressive large-scale tracking results, with an overall drift below 0.5 % of the travelled distance ([94], [51]). However, VIO still fails in a number of situations, such as high-speed motions or high-dynamic range scenes. In the first case, large amounts of motion blur on the images spoil the visual information, forcing the pipeline to rely on integration of the IMU, resulting in large amounts of accumulated drift[1]. In the second case, due to the limited dynamic range of standard cameras, large regions on the image are either over-, or under-exposed, which reduces drastically the amount of information exploitable. It is in these challenging scenarios that the above-mentioned advantages of event cameras could be exploited to yield accurate and robust ego-motion estimation.

In this paper, we present a novel visual-inertial odometry (VIO) algorithm for event cameras. Our algorithm takes as input a stream of events and inertial measurements, and outputs camera poses at a rate proportional to the camera velocity. To achieve this, we track a set of features in the events, and fuse these feature tracks with the IMU measurements using a keyframe-based visual-inertial pipeline that uses nonlinear optimization.

[1] http://www.vectornav.com/support/library/imu-and-ins , see the first table under Case 1

Contribution. Our main contribution is a tightly-coupled visual-inertial odometry pipeline for event cameras that is significantly more accurate than the state-of-the-art [211], while being more efficient. More precisely, our contributions include:

- A novel feature tracker for event cameras that works on event frames, synthesized by fusing the events in a spatio-temporal window using the current estimate of the camera motion and the scene structure.
- The integration of these feature tracks in a keyframe-based, visual-inertial pipeline based on nonlinear optimization, yielding a robust and accurate VIO pipeline for event cameras.
- A quantitative evaluation of our pipeline compared to the state-of-the-art [211] on the public Event Camera Dataset [124], and some qualitative results on a large scale and a high-speed sequence.

D.2 Related Work

In the past decade, many works have considered to use event cameras for ego-motion estimation. Early works focused on addressing restricted, and easier instances of the problem: [30], [79], [63] and [161] showed how to do rotation-only (3 DOF) pose estimation, and [198] proposed a simultaneous tracking and mapping algorithm for event cameras that works for planar (2D) motion and planar scenes. Other authors have used complementary sensing modalities, additionally to an event camera: [196] used an event camera equipped with a depth sensor to jointly estimate the camera pose and 3D scene structure, and [87] proposed a low-latency, feature-based 6 DOF visual odometry pipeline that uses a frame-based sensor, where features are detected in the frames and tracked in the event stream. Event-based, 6-DOF visual odometry (using only an event camera) has been first shown only very recently: [80] proposed three parallel filters that jointly estimate the camera pose, 3D map of the scene, and image intensity, and [150] proposed a geometric approach that combines a global image alignment technique with an event-based reconstruction algorithm [147] to estimate the camera pose and 3D map of the scene without requiring image reconstruction. Few works have considered using an event camera with an IMU (the problem of event-based, visual-inertial odometry). [122] showed how to fuse events and inertial measurements into a continuous time framework. However, their approach is not suited for real-time usage because of the expensive optimization required to update the spline parameters upon receiving every event. Very recently, [211] proposed an event-based visual-inertial odometry algorithm, EVIO, that works in real-time (albeit, for limited motion speeds, and number of features). They proposed to track a set of features in the event stream using an iterative Expectation-Maximization scheme, that jointly solves for the feature appearance and optical flow. The feature tracks are then fed to an EKF filter to produce new pose estimates. EVIO is the closest approach to this work. In Section D.5, we

compare our approach to [211] in terms of accuracy, and show significant improvements compared to it.

D.3 Preliminaries

In this section, we introduce the notation that we will use throughout the rest of the paper. We also introduce the IMU model used, and provide formulas for discrete integration of the equations of motion.

Coordinate Frame Notation. A point P represented in a coordinate frame A is written as position vector ${}_A\mathbf{r}_P$. A transformation between frames is represented by a homogeneous matrix $\mathbf{T}_{AB}$ that transforms points from frame B to frame A. Its rotational part is expressed as a rotation matrix $\mathbf{R}_{AB} \in SO(3)$. Our algorithm uses a sensor composed of an event camera and an IMU rigidly mounted together. The sensor body is represented relative to an inertial world frame W. Inside it, we distinguish the camera frame C and the IMU-sensor frame S. To obtain $\mathbf{T}_{SC}$, an extrinsic calibration of the camera + IMU system must be performed, using for example the Kalibr toolbox [56].

IMU Model and Motion Integration. An IMU usually includes a 3-axis accelerometer and a 3-axis gyroscope, and allows measuring the rotational rate and the acceleration of the sensor with respect to an inertial frame. The measurements, ${}_S\tilde{\mathbf{a}}(t)$ and ${}_S\tilde{\boldsymbol{\omega}}(t)$, are affected by additive white noise $\boldsymbol{\jmath}$ and slowly varying sensor biases $\mathbf{b}$:

$$ {}_S\tilde{\boldsymbol{\omega}}(t) = {}_W\boldsymbol{\omega}(t) + \mathbf{b}_g(t) + \boldsymbol{\jmath}_g(t), \qquad {}_S\tilde{\mathbf{a}}(t) = \mathbf{R}_{WS}^T(t)\,({}_W\mathbf{a}(t) - {}_W\mathbf{g}) + \mathbf{b}_a(t) + \boldsymbol{\jmath}_a(t), \tag{D.1} $$

where ${}_W\mathbf{g}$ is the gravity vector in world coordinates.

Denoting a position vector and velocity, respectively as ${}_A\mathbf{r}$ and ${}_A\mathbf{v}$, the equations of motion can be numerically integrated as follows [51]:

$$ \begin{aligned} \mathbf{R}_{WS}(t+\Delta t) &= \mathbf{R}_{WS}(t)\exp\left({}_S\tilde{\boldsymbol{\omega}}(t) - \mathbf{b}_g(t) - \boldsymbol{\jmath}_{gd}(t)\Delta t\right) \\ {}_W\mathbf{v}(t+\Delta t) &= {}_W\mathbf{v}(t) + {}_W\mathbf{g}\Delta t + \mathbf{R}_{WS}(t)\left({}_S\tilde{\mathbf{a}}(t) - \mathbf{b}_a(t) - \boldsymbol{\jmath}_{ad}(t)\right)\Delta t \\ {}_W\mathbf{r}(t+\Delta t) &= {}_W\mathbf{r}(t) + {}_W\mathbf{v}(t)\Delta t + \tfrac{1}{2}{}_W\mathbf{g}\Delta t^2 + \tfrac{1}{2}\mathbf{R}_{WS}(t)\left({}_S\tilde{\mathbf{a}}(t) - \mathbf{b}_a(t) - \boldsymbol{\jmath}_{ad}(t)\right)\Delta t^2) \end{aligned} \tag{D.2} $$

where $\exp : se(3) \to SE(3)$ denotes the exponential map, and $\boldsymbol{\jmath}_{ad}$, $\boldsymbol{\jmath}_{gd}$ are the noise variables.

Event Data. Let us denote the set of events observed as $\epsilon = \{e_k\}$. The k^{th} event is represented as a tuple $\mathbf{e}_k = (\mathbf{x}_k, t_k, p_k)$, where $\mathbf{x}_k = (x_k, y_k)$ is the event location on the image plane, t_k its timestamp, and p_k its polarity.

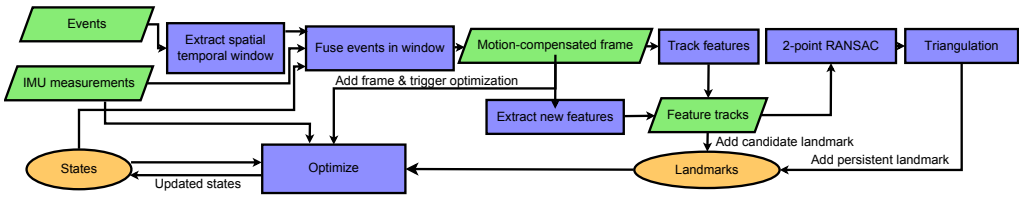


Figure D.1 – Overview of the proposed pipeline: (i) Events are grouped in spatio-temporal windows, and fused to build motion-compensated frames. (ii) New features are extracted if necessary; all feature tracks are updated, and outliers filtered using 2-point RANSAC. (iii) Feature tracks that can be triangulated are converted to persistent, and added to the map. The remaining ones are kept as candidate tracks. (iv) Selected frames are added to the optimizer, and trigger a global optimization.

D.4 Visual-Inertial Odometry with an Event Camera

Our visual-inertial odometry pipeline is classically composed of two parallel threads:

- the front-end (Section D.4.1) takes a stream of events as input. It establishes feature tracks and triangulates landmarks, both of which are passed to the back-end.
- the back-end (Section D.4.2) fuses the feature tracks, landmarks, and IMU measurements to continuously update the current and past sensor states.

Figure D.1 gives an overview of the modules involved and their interactions. The rest of this section is organized as follows. Section D.4.1 describes how we partition the event stream in spatio-temporal windows and synthesize motion-corrected event images, Section D.4.1 provides details about feature tracking and landmark triangulation, and Section D.4.1 gives additional implementation details. The back-end of our algorithm, a keyframe-based nonlinear optimization algorithm, is described in Section D.4.2.

D.4.1 Front-end

Our pipeline takes a stream of events as input, and produces a set of motion-corrected event images (Section D.4.1) that are fed to a visual odometry front-end (Sections D.4.1 and D.4.1).

Synthesis of Motion-Corrected Event Frames

Spatio-temporal Windows of Events. The set of observed events ϵ is split in a set of overlapping spatio-temporal windows $\{W_k\}$ (Fig. D.2). The k^{th} window is defined as the set of events $W_k = \{e_{kS}, ..., e_{kS+N-1}\}$, where N is the window size parameter, and S

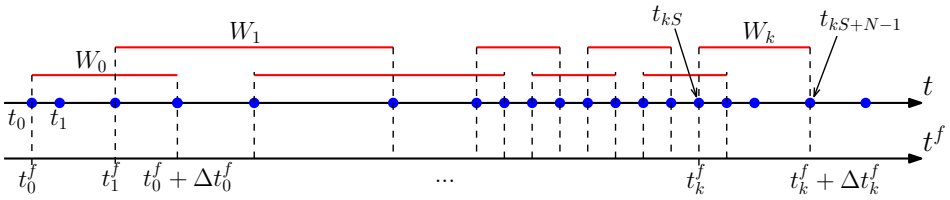


Figure D.2 – We split the event stream in a set of overlapping spatio-temporal windows. Events are depicted as blue dots on the timeline. The windows $\{W_k\}$ are marked in red ($N = 4$, $S = 2$ here). Note that the temporal size of each window is automatically adapted to the event rate.

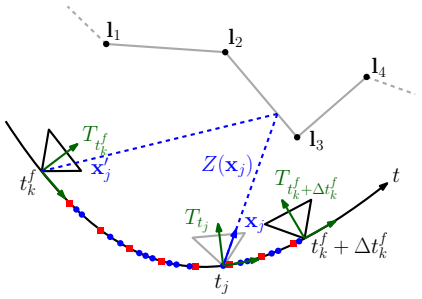


Figure D.3 – Motion correction: the inertial measurements in $[t_k^f, t_k^f + \Delta t_k^f]$ (red squares) are integrated to compute the relative transformation $T_{t_k^f, t_k^f + \Delta t_k^f}$. Each event (blue dot) $\mathbf{e}_j$ is reprojected to camera frame $C_{t_k^f}$ using the linearly interpolated pose T_{t_j} and the linearly interpolated depth $Z(\mathbf{x}_j)$.

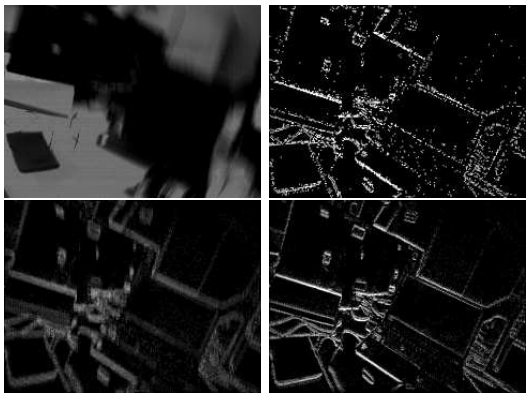


Figure D.4 – Synthesized event frames. From top left to bottom right: standard camera image; 3 000 events (noisy information); 30 000 events (motion-blurred image); 30 000 events with motion-correction.

a step size parameter that controls the amount of overlap between successive windows. Note that the start time $t_k^f := t_{kS}$ and duration of each window Δt_k^f are controlled by the events, which preserves the data-driven nature of the sensor. With this notation, W_k spans the time interval $[t_{kS}, t_{kS+N-1}] := [t_k^f, t_k^f + \Delta t_k^f]$.

From Events to Event Frames. A naive way to synthesize an event image from a window of events W_k would be to accumulate them as follows: $I_k(\mathbf{x}) = \sum_{e_j \in W_k} \delta\left(\mathbf{x} - \mathbf{x}_j\right)$, i.e., the intensity I at pixel $\mathbf{x}$ is simply the sum of the events that fired at the pixel location $\mathbf{x} = \mathbf{x}_j$. However, this yields event images that are not usable for reliable feature detection or tracking, as illustrated in Fig. D.4: small window sizes do not convey enough information, while large window sizes induce motion blur.

Inspired by [40, 63], we propose to locally correct the motion of each event according to its individual time stamp. This allows to synthesize motion-corrected event frames, used subsequently to establish feature tracks.

Motion Compensation of Events. We synthesize a motion-corrected event image I_k (see Figs. D.3 and D.4) as follows: $I_k(\mathbf{x}) = \sum_{e_j \in W_k} \delta(\mathbf{x} - \mathbf{x}'_j)$, where $\mathbf{x}'_j$ is the *corrected* event position, obtained by transferring event $\mathbf{e}_j$ to the reference camera frame $C_{t_k^f}$:

$$\mathbf{x}'_j = \boldsymbol{\pi}\left(T_{t_k^f, t_j}(Z(\mathbf{x}_j)\boldsymbol{\pi}^{-1}(\mathbf{x}_j)) \right), \tag{D.3}$$

where $\boldsymbol{\pi}(.)$ is the camera projection model, obtained from prior intrinsic calibration.

Adopting the short-hand notations: $T_{t_j} := T_{WC(t_j)}$, and $T_{t_i,t_j} := T_{C(t_i^f)C(t_j^f)}$, the incremental transformation $T_{t_k^f, t_k^f + \Delta t_k^f}$ is obtained through integration of the IMU measurements in $[t_k^f, t_k^f + \Delta t_k^f]$ using (D.2). The necessary starting pose $T_{t_k^f}$, and the IMU biases $\mathbf{b}_g, \mathbf{b}_a$ are known from the state estimation thread. The remaining quantities required to evaluate (D.3) are:

- $T_{t_k^f, t_j}$, which we linearly interpolate from $T_{t_k^f}$ and $T_{t_k^f + \Delta t_k^f}$, in the space of rigid-body motion transformations $SE(3)$.
- $Z(\mathbf{x}_j)$, which we estimate using 2D linear interpolation (on the image plane) of the current landmarks $\{\mathbf{l}_j\}$, reprojected on the current camera frame C_{t_j}.

In practice, we observed that using the median depth of the current landmarks instead of linearly interpolating the depth gives satisfactory results, at a lower computational cost.

We acknowledge that [40] were the first to show how a stream of events can be derotated on an event-by-event basis using rotational information from an IMU. However, by contrast to [40], we do not only correct for rotation, but we also model translational optical flow using the estimated camera velocity and scene depth. In addition, our motion compensation scheme operates in the current camera local frame (*i.e.* suppresses motion blur), while [40] derotates events in a global, gravity-aligned inertial frame (*i.e.* akin to image stabilization).

Feature Detection and Tracking. Landmark Triangulation

Feature Detection. New features are detected whenever the number of feature tracks falls below a certain threshold, or if the current frame is selected as a keyframe (see Section D.4.1). We use the FAST corner detector [165] on a motion-compensated event frame. We use a bucketing grid to ensure that the features are evenly distributed over the image.

Feature Tracking and Landmark Triangulation. We maintain two sets of landmarks: candidate landmarks, and persistent landmarks, whose 3D position in space has

been successfully triangulated. Newly extracted features are initialized as candidate landmarks, and are tracked across event frames. As soon as a candidate landmark can be reliably triangulated, it is converted to a persistent landmark, and added to the local map.

Both types of landmarks are tracked from I_k to I_{k+1} using pyramidal Lukas-Kanade tracking [8]. The incremental transformation $T_{t_k^f, t_{k+1}^f}$ (integrated from the IMU measurements in $[t_k^f, t_{k+1}^f]$) is used to predict the feature position in I_{k+1}. The patches around each features are warped through an affine warp, computed using $T_{t_k^f, t_{k+1}^f}$, prior to pyramidal alignment. Landmarks that are not successfully tracked in the current frame are discarded immediately.

Outlier Filtering. We use two-point RANSAC [192] (using the relative orientation between the current frame and the last keyframe) to further filter out outlier feature tracks.

Additional Implementation Details

Keyframe Selection. A new keyframe is selected either when the number of tracked features falls below a threshold, or when the distance to the last keyframe (scaled by the median scene depth) reaches a minimum threshold.

Initialization. To initialize our pipeline we add the first frames to the back-end without initializing any feature track. The back-end in turn estimates the initial attitude of the sensor by observing the gravity direction. The displacement between the following frames is then estimated by integrating IMU measurements.

D.4.2 Back-end

In this section, we describe how we fuse feature tracks from the event stream obtained in Section D.4.1 to update the full sensor state over time.

As opposed to the EKF-based filtering employed in [211], we prefer to rely on a full smoothing approach based nonlinear optimization on selected keyframes. Indeed, such approaches have been shown to outperform pipelines in terms of accuracy [187]. This has recently been made computationally tractable by the development of the pre-integration theory [104], [52], that consists of combining many inertial measurements between two keyframes into a single relative motion constraint, thus avoiding to reintegrate inertial measurements in each step of the optimization.We base our back-end implementation on OKVIS [94]. For space reasons, we omit the details of the

pipeline and refer the reader to the original publications [93], [94].

The visual-inertial localization and mapping problem is formulated as a joint optimization of a cost function that contains weighted reprojection errors $\mathbf{e}_r$ and inertial error terms $\mathbf{e}_s$:

$$J := \sum_{k=1}^{K} \sum_{j \in \mathcal{J}(k)} \mathbf{e}^{j,k^T} \mathbf{W}_r^{j,k} \mathbf{e}^{j,k} + \sum_{k=1}^{K-1} \mathbf{e}_s^{k^T} \mathbf{W}_s^k \mathbf{e}_s^k$$

where k denotes the frame index, and j denotes the landmark index. The set $\mathcal{J}(k)$ contains the indices of landmarks visible in the k^{th} frame. Additionally, $W_r^{j,k}$ is the information matrix of the landmark measurement $\mathbf{l}_j$, and W_s^k that of the k^{th} IMU error. The optimization is carried out, not on all the frames observed, but on a bounded set of frames composed of M keyframes (selected by the front-end, see Section D.4.1), and a sliding window containing the last K frames. In between frames, the prediction for the sensor state is propagated using the IMU measurements that fall in between the frames. We employ the Google Ceres [2] optimizer to carry out the optimization.

Reprojection Error. $\mathbf{e}_r^{j,k} = \mathbf{z}^{j,k} - \boldsymbol{\pi}\left(\mathbf{T}_{CS}^k \mathbf{T}_{SW}^k \mathbf{l}^j\right)$ where $\mathbf{z}^{j,k}$ is the measured image coordinate of the j^{th} landmark on the k^{th} frame.

IMU Measurement Error Term. We use the IMU kinematics and biases model introduced in (D.2) to predict the current state based on the previous state. Then, the IMU error terms are simply computed as the difference between the prediction based on the previous state and the actual state. For orientation, a simple multiplicative minimal error is used.

Keyframe Marginalization. Keeping all keyframes in the Gauss-Newton system state quickly becomes untractable. However, simply discarding measurements from past keyframes neglects valuable information. To overcome this problem we partially marginalize out old keyframes using the Schur complement on the corresponding observations. This turns old measurements into a prior for our system, represented as summands in the construction of the Schur complement. Details are provided in [94].

D.5 Evaluation

For all the experiments presented below, we used the DAVIS [18] sensor, which embeds a 240×180 pixels event camera with a 1 kHz IMU. In addition to the event stream and IMU measurements, the sensor provides standard images, which are *not* used by our pipeline.

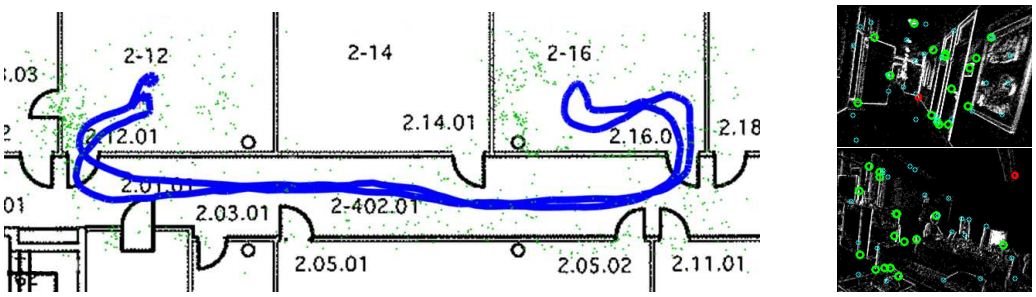


Figure D.5 – `corridor_dataset`. Left: trajectory and point cloud estimated by our pipeline, overlaid with a map of the building. Right: Two motion-corrected event frames used by our pipeline, with overlaid persistent landmarks (green) and candidate landmarks (blue). Red circles are RANSAC outliers.

D.5.1 Quantitative: Accuracy and Performance

We use the Event Camera Dataset [124] to evaluate quantitatively the accuracy of our pipeline. The dataset features various scenes with ground truth tracking information. In particular, it contains extremely fast motions and scenes with very high-dynamic range.

Evaluation Metrics. The estimated and ground truth trajectories were aligned with a 6-DOF transformation (e.g. in SE3), using the subset $[5-10s]$. We computed the mean position error (Euclidean distance) and the yaw error as percentages of the total travelled distance. Due to the observability of the gravity direction, the error in pitch and roll direction is constant and comparable between our approach and EVIO. Thus we omit them for compactness. Additionally, we use the relative error metrics proposed in [66], which evaluate the relative error by averaging the drift over trajectories of different lengths (Fig. D.6).

Parameters. The window size N was manually selected for all datasets, always in the range of $[10^4 - 10^5]$ events, except for the `shapes_translation` and `shapes_6dof` for which we used $N = 3000$ (since the global event rate is much lower in those). This translates to a temporal window size of about 5 to 10 milliseconds. We used $S = N$ for all the experiments, e.g., no overlap between successive windows. The patch size used for feature tracking was 32×32 pixels, with 2 pyramid levels.

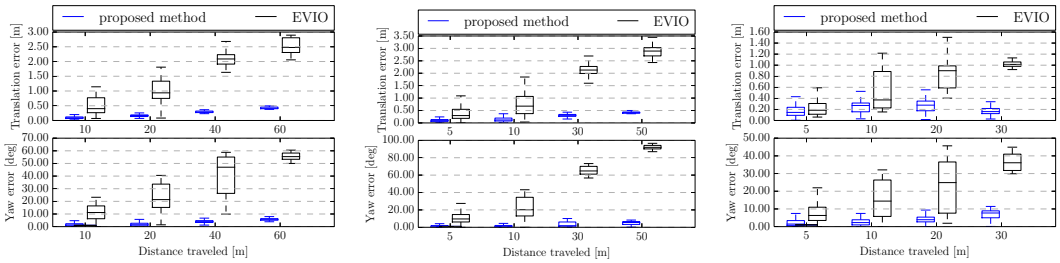
Accuracy and Performance. Table D.1 and Fig. D.6 demonstrate the remarkable accuracy of our pipeline compared to EVIO [211], the state-of-the-art. We ran the same evaluation code (alignment and computation of error metrics) for our method and EVIO, using raw trajectories provided by the authors.

Sequence	Our proposed method		EVIO [211] (CVPR'17)	
	Mean Position* Error (%)	Mean Yaw Error (deg/m)	Mean Position Error (%)	Mean Yaw Error (deg/m)
boxes_6dof	**0.69**	**0.09**	4.13	0.92
boxes_translation	**0.57**	**0.04**	3.18	0.67
dynamic_6dof	**0.54**	**0.26**	3.38	1.20
dynamic_translation	**0.47**	**0.11**	1.06	0.25
hdr_boxes	**0.92**	**0.01**	3.22	0.15
hdr_poster	**0.59**	**0.09**	1.41	0.13
poster_6dof	**0.82**	**0.11**	5.79	1.84
poster_translation	**0.89**	**0.03**	1.59	0.38
shapes_6dof	**1.15**	**0.08**	2.52	0.61
shapes_translation	**1.28**	**0.41**	4.56	2.60

Table D.1 – Accuracy of the proposed approach against EVIO [211], the state-of-the-art.

	Time (ms)
synthesize event frame	4.23
feature detection	0.69
feature tracking	0.90
two-point RANSAC	0.08
add frame to back-end	1.47
wait for back-end	1.29
total time	**8.23**

Table D.2 – Time spent in different modules, for a single spatio-temporal window.



(a) `boxes_6dof` **(b)** `poster_6dof` **(c)** `dynamic_6dof`

Figure D.6 – Comparison of the proposed approach versus EVIO on three datasets from [124]. Relative errors are measured over different segments of the trajectory. Additional plots for all the datasets are provided in the supplementary material.

Our method runs on average 50% faster than real-time on a laptop, even for fast motions that yield very high event rates. For example the `boxes_6dof` dataset was processed in 41.7 s, which corresponds to 3.2 million events/s or 1.45 times faster than real-time. Table D.2 shows the time spent per spatio-temporal window on an Intel Core i5-4278U@2.60GHz. The total time per event frame is 8.23 ms. We also run our algorithm on a smartphone CPU (Odroid-XU4@2GHz) and we measured a total time of 20ms.

D.5.2 Qualitative Results

To further demonstrate the capabilities of our method, we present two additional datasets: `corridor`, and `spinning_leash`. The `corridor` dataset was recorded by walking in our building with a DAVIS sensor, bringing it back to its exact start position. Fig. D.5 shows a top-view of the estimated trajectory and the accumulated landmarks. In the absence of ground truth, we estimate the accumulated drift as the distance

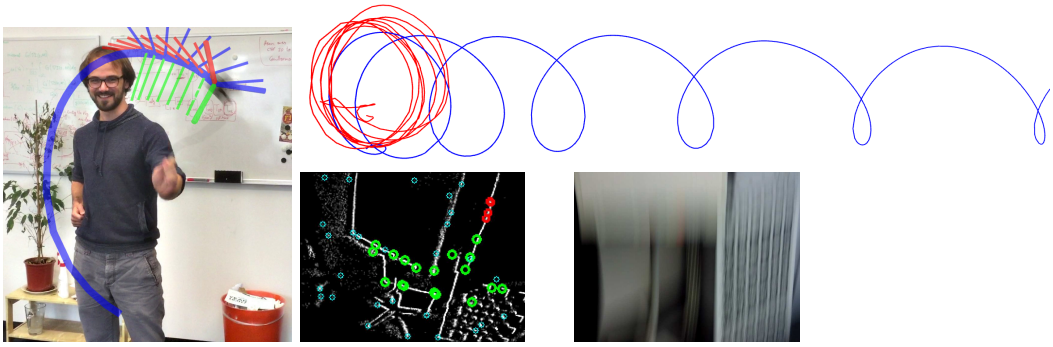


Figure D.7 – `spinning_leash` dataset. Left: Person spinning an event camera attached to a leash. The camera is barely visible due to motion blur. The trajectory estimated in real-time by our algorithm is superimposed on the image. Top-right: Trajectory estimated by our method (red) and plain IMU integration (blue). Bottom-right: motion-corrected event frame (same legend as Fig. D.5) compared to an image obtained by spinning a standard camera at the same speed.

between the first position and the last position: about 50 cm for a 55 m trajectory, i.e. less than 1 % drift.

The `spinning_leash` dataset was recorded by spinning really fast a DAVIS camera, attached to a leash, in our office (Fig. D.7). Despite the extreme velocity of the motion, our pipeline successfully tracks the camera pose with low drift.

D.5.3 Discussion

Our outstanding results compared to [211] can be explained in part because we use a nonlinear optimization approach, as opposed to a filtering approach, whose accuracy is known to quickly deteriorate due to the accumulation of linearization errors.

Due to its simplicity, our feature tracker can be implemented efficiently, making real-time tracking possible on the CPU. By contrast, the feature tracker used in [211] relies on an expensive, iterative Expectation-Maximization scheme which severely throttles the speed of the overall pipeline.

D.6 Conclusion

We presented a novel, tightly-coupled visual-inertial odometry pipeline for event cameras. Our method significantly outperforms the state-of-the-art [211] on the challenging Event Camera Dataset [124], while being computationally more efficient; it can run on average 50 % faster than real-time on a laptop. We also demonstrated qualitatively the accuracy and robustness of our pipeline on a large-scale dataset, and an extremely

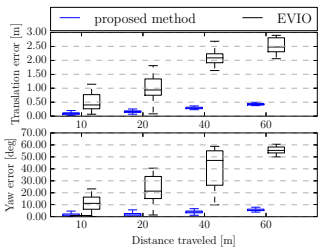
high-speed dataset. We believe this work makes a significant step towards the use of event cameras for high-impact applications, such as the navigation of mobile robots, or AR/VR applications.

Appendix

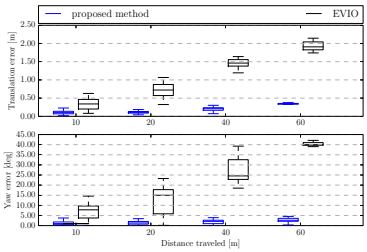
D.7 Acknowledgments

We thank Guillermo Gallego for valuable discussions and suggestions, and the authors of [211] for sharing the raw trajectories of their algorithm on the Event Camera Dataset.

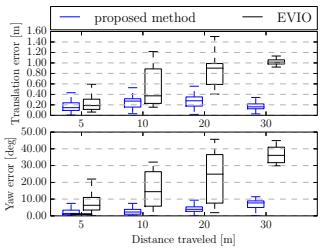
This research was funded by the DARPA FLA Program, the National Center of Competence in Research (NCCR) Robotics through the Swiss National Science Foundation and the SNSF-ERC Starting Grant.



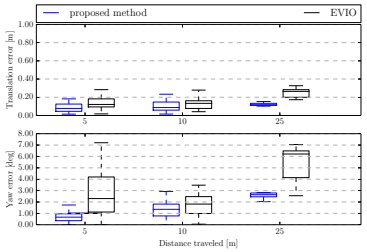
(a) `boxes_6dof`



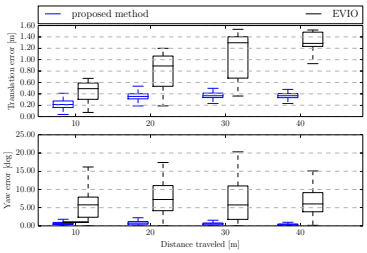
(b) `boxes_translation`



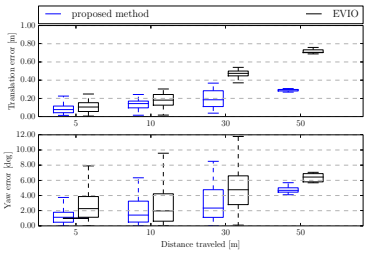
(c) `dynamic_6dof`



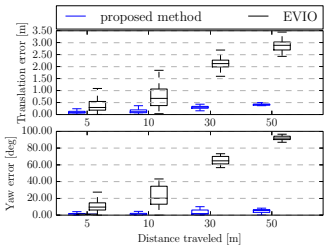
(d) `dynamic_translation`



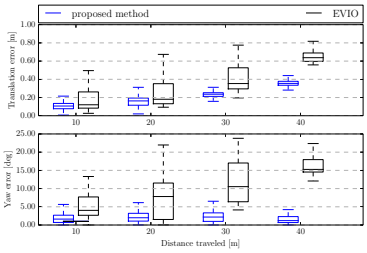
(e) `hdr_box`



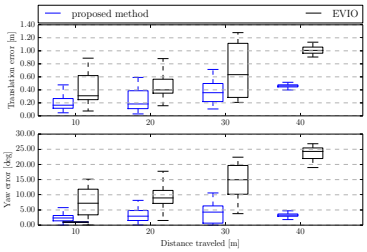
(f) `hdr_poster`



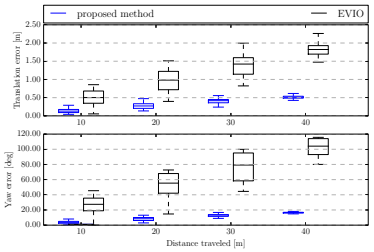
(g) `poster_6dof`



(h) `poster_translation`



(i) `shapes_6dof`



(j) `shapes_translation`

Figure D.8 – Comparison of our method against EVIO [211] on the Event Camera Dataset.

E UltimateSLAM? Combining Events, Images, and IMU for Robust Visual SLAM

Reprinted, with permission, from:

A. Rosinol Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: *IEEE Robot. Autom. Lett.* 3.2 (Apr. 2018), pp. 994–1001. DOI: 10.1109/LRA.2018.2793357

Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios

Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer and

Davide Scaramuzza

Abstract — Event cameras are bio-inspired vision sensors that output pixel-level brightness changes instead of standard intensity frames. These cameras do not suffer from motion blur and have a very high dynamic range, which enables them to provide reliable visual information during high speed motions or in scenes characterized by high dynamic range. However, event cameras output only little information when the amount of motion is limited, such as in the case of almost still motion. Conversely, standard cameras provide instant and rich information about the environment most of the time (in low-speed and good lighting scenarios), but they fail severely in case of fast motions, or difficult lighting such as high dynamic range or low light scenes. In this paper, we present the first state estimation pipeline that leverages the complementary advantages of these two sensors by fusing in a tightly-coupled manner events, standard frames, and inertial measurements. We show on the publicly available Event Camera Dataset that our *hybrid* pipeline leads to an accuracy improvement of 130% over *event-only* pipelines, and 85% over *standard-frames-only* visual-inertial systems, while still being computationally tractable. Furthermore, we use our pipeline to demonstrate—to the best of our knowledge—the first autonomous quadrotor flight using an event camera for state estimation, unlocking flight scenarios that were not reachable with traditional visual-inertial odometry, such as low-light environments and high-dynamic range scenes.

E.1 Introduction

The task of estimating a sensor's ego-motion has important applications in various fields, such as augmented/virtual reality or autonomous robot control. In recent years, great progress has been achieved using visual and inertial information ([119, 94, 52]). However, due to some well-known limitations of traditional cameras (motion blur and low dynamic-range), these Visual Inertial Odometry (VIO) pipelines still struggle to cope with a number of situations, such as high-speed motions or high-dynamic range scenarios.

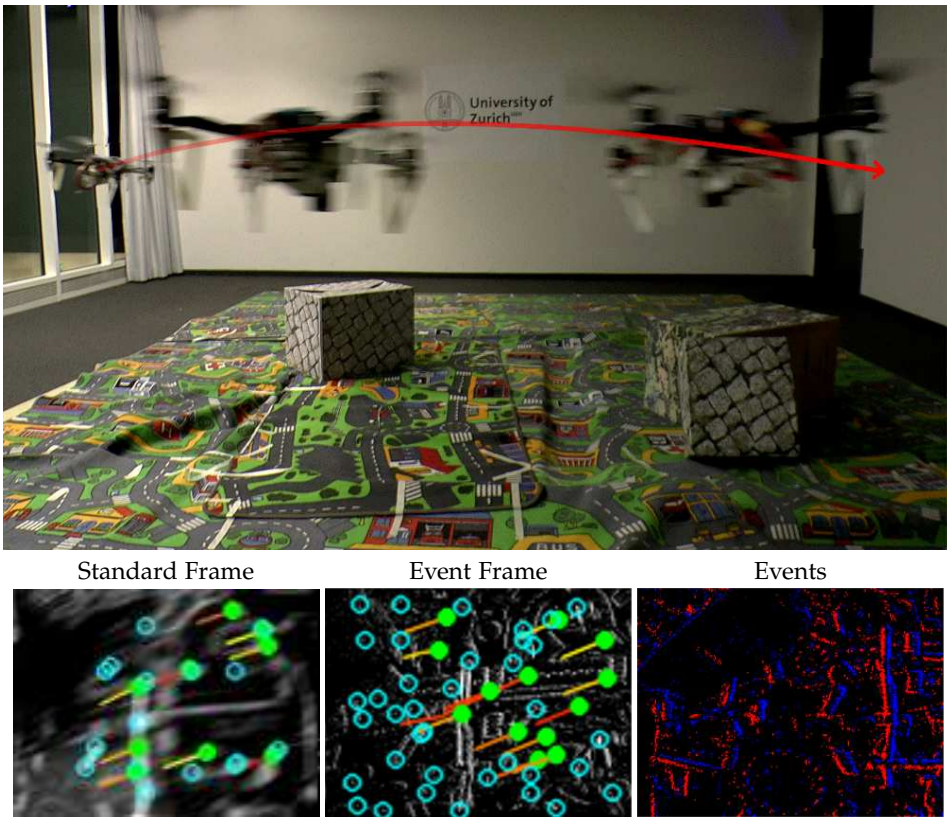


Figure E.1 – Our state estimation pipeline combines events, standard frames, and inertial measurements to provide robust state estimation, and can run onboard an autonomous quadrotor with limited computational power. Bottom Left: Standard frame, Bottom Middle: Virtual event frame, Bottom Right: Events only (blue: positive events, red: negative events).

Novel types of sensors, called event cameras, offer great potential to overcome these issues. Unlike standard cameras, which transmit intensity frames at a fixed framerate, event cameras, such as the Dynamic Vision Sensor (DVS) [98], only transmit *changes of intensity*. Specifically, they transmit per-pixel intensity changes at the time they occur, in the form of a set of asynchronous *events*, where each event carries the space-time

coordinates of the brightness change, and its sign.

Event cameras have numerous advantages over standard cameras: a latency in the order of microseconds and a very high dynamic range (140 dB compared to 60 dB of standard cameras). Most importantly, since all the pixels capture light independently, such sensors do not suffer from motion blur.

Event cameras transmit, in principle, all the information needed to reconstruct a full video stream [30, 10, 160], and one could argue that an event camera alone is sufficient to perform state estimation. In fact, this has been shown recently in [150] and [80]. However, to overcome the lack of intensity information, these approaches need to reconstruct, in parallel, a consistent representation of the environment (a semi-dense depth map in [150] or a dense depth map with intensity values in [80]), by combining—in one way or another—information from a large number of events to recover most gradients in the scene.

Conveniently, standard cameras provide direct access to intensity values, but do not work in low-light conditions, suffer from motion blur during fast motions (due to the synchronous exposure on the whole sensor), and have a limited dynamic range (60 dB), resulting in frequent over- or under-exposed areas in the frame.

Observing this complementarity, in this paper we propose a pipeline that leverages the advantages of both sensing modalities in combination with an inertial measurement unit (IMU) to yield a robust, yet accurate, state estimation pipeline.

While there is a considerable body of literature investigating the use of standard cameras with an IMU to perform state estimation, as well as recent work using an event camera with an IMU, combining all three sensing modalities is yet an open problem. Additionally, in the core application that we envision—flying autonomously a quadrotor with an event camera—there is no specific literature, although attempts to use an event camera for quadrotor flight can be traced to a single paper [73], which is currently limited to vertical landing maneuvers.

In this work, we propose—to the best of our knowledge—the first state estimation pipeline that fuses all three sensors, and we build on top of it to propose the first quadrotor system that can advantageously exploit this hybrid sensor combination to fly in difficult scenarios, using only onboard sensing and computing (Fig. E.1).

Contributions

A frontal comparison with state-of-the-art, commercial visual-inertial pipelines (like for example the ones used for the Snapdragon flight [185]) is not our goal in this work. Indeed, such solutions typically use one or more high quality cameras with a much

higher resolution than the sensor we used, and are carefully engineered to work well in the most common consumer situations. Instead, in this work, we focus on difficult scenarios, and show, for the first time, that (i) it is possible to run state estimation with an event camera onboard a computationally limited platform, and (ii) we show that it can unlock, in a set of difficult scenarios, the possibility for autonomous flight where even commercial systems would struggle.

Specifically, our contributions in this paper are three-fold:

- We introduce the first state estimation pipeline that fuses events, standard frames, and inertial measurements to provide robust and accurate state estimation. While our pipeline is based on [149], we extend it to include standard frames as an additional sensing modality, and propose several improvements to make it usable for real-time applications, with a focus on mobile robots.
- We evaluate quantitatively the proposed approach and show that using standard frames as an additional modality improves the accuracy of state estimation while keeping the computational load tractable.
- We show that our method can be applied for state estimation onboard an autonomous quadrotor, and demonstrate in a set of experiments that the proposed system is able to fly reliably in challenging situations, such as low-light scenes or fast motions.

Our work aims at highlighting the potential that event cameras have for robust state estimation, and we hope that our results will inspire other researchers and industries to push this work forward, towards the wide adoption of event cameras on mobile robots.

The rest of the paper is organized as follows: section E.2 reviews related literature on event-based ego-motion estimation methods, particularly those involving event cameras. In section E.3, we present our hybrid state estimation pipeline that fuses events, standard frames and inertial measurements in a tightly-coupled fashion, and evaluate it quantitatively on the publicly available Event Camera Dataset [124]. Section E.4 describes how the proposed approached can be used to fly a quadrotor autonomously, and demonstrate in a set of real-life experiments that it unlocks challenging scenarios difficult to address with traditional sensing E.4.2. Finally, we draw conclusions in section E.5.

E.2 Related Work

Using visual and inertial sensors for state estimation has been extensively studied over the past decades. While the vast majority of these works use standard cameras together

with an IMU, a recent parallel thread of research that uses event cameras in place of standard cameras has recently flourished.

Visual-inertial Odometry with Standard Cameras The related work on visual-inertial odometry (VIO) can be roughly segmented into three different classes, depending on the number of camera poses that are used for the estimation. While full smoothers (or batch nonlinear least-squares algorithms) estimate the complete history of poses, fixed-lag smoothers (or sliding window estimators) consider a window of the latest poses, and filtering approaches only estimate the latest state. Both fixed-lag smoothers and filters marginalize older states and absorb the corresponding information in a Gaussian prior. More specifically:

- Filtering algorithms enable efficient estimation by restricting the inference process to the latest state of the system. A example approach of a filter-based visual-inertial odometry system is [77].
- Fixed-lag smoothers estimate the states that fall within a given time window, while marginalizing out older states, as for example, [94].
- Full smoothing methods estimate the entire history of the states (camera trajectory and 3D landmarks), by solving a large nonlinear optimization problem. A recent approach in this category was proposed by [52].

Visual-inertial Odometry with Event Cameras Since the introduction of the first commercial event camera in 2008 [98], event cameras have been considered for state estimation by many different authors. While early works focused on addressing restricted and easier instances of the problem, like rotational motion estimation ([30], [79], [63], [161]), or Simultaneous Localization and Mapping (SLAM) in planar scenes only [198], it has been shown recently that 6-DOF pose estimation using only an event camera is possible ([150, 80]).

In parallel, other authors have explored the use of complementary sensing modalities, such as a depth sensor [196], or a standard camera ([24], [87]). However, (i) none of these image-based pipelines make use of inertial measurements, and (ii) both of them use the intensity of the frames as a template, to which they align the events. Therefore, these approaches work only when the standard frames are of good quality (sharp and correctly exposed); they will fail in those particular cases where the event camera has an advantage over a standard camera (high-speed motions, and HDR scenes).

Using an event camera and an IMU has only been explored very recently. [122] showed how to fuse events and inertial measurements into a continuous time framework, but

their approach is not suited for real-time usage because of the expensive optimization required to update the spline parameters upon receiving every event. [211] proposed to track a set of features in the event stream using an iterative Expectation-Maximization scheme that jointly refines each feature's appearance and optical flow, and then fuse these tracks using an Extended Kalman Filter to yield an event-based visual-inertial odometry pipeline. Unfortunately, due to the expensive nature of their feature tracker, the authors of [211] reported that their pipeline cannot run in real-time in most scenarios.

In [149], we proposed an accurate event-based visual inertial odometry pipeline that can run in real-time, even on computationally limited platforms, such as smartphone processors. The key of this approach was to estimate the optical flow generated by the camera's rigid body motion by exploiting the current camera pose, scene structure, and inertial measurements. We then efficiently generated virtual, motion-compensated event frames using the computed flow [40, 62], and further tracked visual features across multiple frames. Those feature tracks were finally fused with inertial information using keyframe-based nonlinear optimization, in the style of [94] and [52]. While our proposed state estimation approach is strongly inspired by this work (i.e., [149]), we extend it by allowing it to additionally work with frames from a standard camera, and propose several changes to the pipeline to adapt it to run onboard a flying robot.

Quadrotor Control with an Event Camera Although the research on robot control with event cameras is still in its infancy, previous work has demonstrated possible interesting applications. [123] mounted a DVS sensor on a quadrotor and showed that it can be used to track the 6-DOF motion of a quadrotor performing a high speed flip maneuver, although the tracker only worked for an artificial scene containing a known black square painted over a white wall. Also, the state estimation was performed offline, and therefore not used for closed-loop control of the quadrotor. More recently, [73] showed closed-loop take-off and landing of a quadrotor using an event camera. Their system, however, relied on computing optical flow and assumed the flow field to be divergent, thus it cannot be used for general 6-DOF control of a quadrotor, unlike our approach.

E.3 Hybrid State Estimation Pipeline

Our proposed state estimation pipeline is largely based on [149]. However, while [149] used only an event camera combined with an IMU, we propose to allow for an additional sensing modality: a standard camera, providing intensity frames at a fixed framerate. For this reason, we focus below on describing the differences between our approach and [149] in order to also consider standard frames. Finally, we evaluate the improved pipeline on the Event Camera Dataset [124] and show evidence that

incorporating standard frames in the pipeline leads to an accuracy boost of 130% over a pipeline that uses only events plus IMU, and 85% over a pipeline that uses only standard frames plus IMU.

E.3.1 Overview

[149] can be briefly summarized as follows. The main idea is to synthesize virtual frames (*event frames*) from spatio-temporal windows of events, and then perform feature detection and tracking using classical computer vision methods, namely the FAST corner detector [165] and the Lucas-Kanade tracker [103]. Feature tracks are used to triangulate the 3D locations of the corresponding landmarks whenever it can be done reliably. Finally, the camera trajectory and the positions of the 3D landmarks are periodically refined by minimizing a cost function involving visual terms (reprojection error) and inertial terms, thus effectively fusing visual and inertial information.

In this paper, we propose to not only maintain feature tracks from virtual event frames, but to also maintain, in parallel, feature tracks from standard frames as well. We then feed the feature tracks coming from these two heterogeneous sources (virtual event frames and standard frames) to the optimization module, thus effectively refining the camera poses using the events, the standard frames, and the IMU.

Coordinate Frame Notation

A point P represented in a coordinate frame A is written as position vector ${}_A\mathbf{r}_P$. A transformation between frames is represented by a homogeneous matrix $\mathbf{T}_{AB}$ that transforms points from frame B to frame A. Its rotational part is expressed as a rotation matrix $\mathbf{R}_{AB} \in SO(3)$. Our algorithm uses a hybrid sensor composed of an event camera, a standard camera, and an IMU rigidly mounted together. The sensor body is represented relative to an inertial world frame W. Within the sensor body, we distinguish the event camera frame C_0, the standard camera frame C_1 and the IMU-sensor frame S. To obtain $\mathbf{T}_{SC_0}$ and $\mathbf{T}_{SC_1}$, an extrinsic calibration of the {event camera + standard camera + IMU system} must be performed.

Spatio-temporal Windows of Events

We synchronize the spatio-temporal windows of events on the timestamps of the standard frames. Upon reception of each standard frame at time t_k, a new spatio-temporal window of events W_k is created (Fig. E.2). The k^{th} window is defined as the set of events $W_k = \left\{e_{j(t_k)-N+1}, ..., e_{j(t_k)}\right\}$, where $j(t_k)$ is the index of the first event whose timestamp $t_j < t_k$, and N is the window size parameter. Note that the duration of each window is inversely proportional to the event rate.

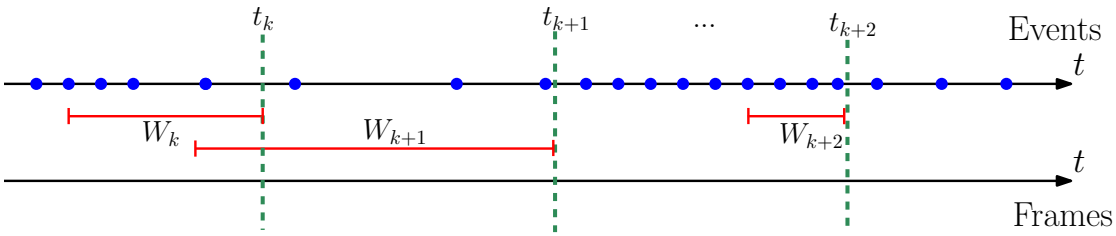


Figure E.2 – Upon receiving a new frame from the standard camera at time t_k, we select a spatio-temporal window of events W_k, containing a fixed number of events ($N = 4$ in this example). Note that the temporal size of each window is automatically adapted to the event rate. Blue dots correspond to events, and the dashed green lines correspond to the times at which standard frames are received. The bounds of the spatio-temporal windows of events considered are marked in red.

Synthesis of Motion-Compensated Event Frames

As in [149], we then collapse every spatio-temporal window of events to a synthetic event frame I_k by drawing each event on the image plane, after correcting for the motion of each event according to its individual timestamp.

Let $I_k(\mathbf{x}) = \sum_{e_i \in W_k} \delta(\mathbf{x} - \mathbf{x}'_i)$, where function $\delta(\mathbf{x})$ is the Kronecker delta, $\mathbf{x}'_i$ is the *corrected* event position, obtained by transferring event e_i to the reference event camera frame C_{0k}:

$$\mathbf{x}'_i = \boldsymbol{\pi}_0(T_{t_k,t_i}(Z(\mathbf{x}_i)\boldsymbol{\pi}_0^{-1}(\mathbf{x}_i))), \tag{E.1}$$

where $\mathbf{x}_i$ is the pixel location of event e_i, $\boldsymbol{\pi}_0(.)$ the event camera projection model, obtained from prior intrinsic calibration, and T_{t_l,t_m} the incremental transformation between the camera poses at times t_l and t_m, obtained through integration of the inertial measurements (we refer the reader to [149] for details). $Z(\mathbf{x}_i)$ is the scene depth at time t_i and pixel $\mathbf{x}_i$, which we can estimate using 2D linear interpolation (on the image plane) of the landmarks reprojected on the current camera frame C_{0i}. In practice, as in [149], we observed that using the median depth of the landmarks in the field of view instead of linearly interpolating the depth gives satisfactory results at a lower computational cost. The quality of the motion compensation depends on the quality of the 3D landmarks available, therefore the quality of the event frames improves when also using the landmarks from the standard frames.

The number of events N in each spatio-temporal window is a parameter that needs to be adjusted depending on the amount of texture in the scene. As an example, for the quadrotor experiments presented in section E.4, we used $N = 20\,000$ events per frame.

Feature Tracking

We use the FAST corner detector to extract features [165], both on the virtual event frames, and the standard camera frames. Those features are then tracked independently across standard frames and event frames using the KLT tracker [103] (see Fig. E.1). This yields two sets of independent features tracks $\{\mathbf{z}^{0,j,k}\}$, $\{\mathbf{z}^{1,j,k}\}$ (where j is the feature track index, and k is the frame index). For each sensor, each feature is treated as a *candidate* feature, and tracked over multiple frames. Once a feature can be triangulated reliably, the corresponding 3D landmark is triangulated through linear triangulation [70], and converted to a *persistent* feature which will be further tracked across the next frames. We re-detect features on each sensor as soon as the number of tracked features falls below a threshold. We used the same detection and tracking parameters for the motion-compensated event frames and for the standard frames. The FAST threshold we used was 50. We used a a pyramidal implementation of KLT with 2 pyramid levels, and a patch size of 24×24 pixels. Additionally, we used a bucketing grid (where each grid cell has size 32×32 pixels) to ensure that features are evenly distributed in each sensor's image plane.

Visual-inertial Fusion through Nonlinear Optimization

The visual-inertial localization and mapping problem is formulated as the joint optimization of a cost function that contains three terms: two weighted reprojection errors corresponding respectively to the observations from the event camera and the standard camera, plus an inertial error term $\mathbf{e}_s$:

$$J = \sum_{i=0}^{1} \sum_{k=1}^{K} \sum_{j \in \mathcal{J}(i,k)} \mathbf{e}^{i,j,k\,T} \mathbf{W}_r^{i,j,k} \mathbf{e}^{i,j,k} + \sum_{k=1}^{K-1} \mathbf{e}_s^{k\,T} \mathbf{W}_s^k \mathbf{e}_s^k$$

where i denotes the sensor index, k denotes the frame index, and j denotes the landmark index. The set $\mathcal{J}(i,k)$ contains the indices of landmarks maintained in the k^{th} frame by sensor i. Additionally, $W_r^{i,j,k}$ is the information matrix of the landmark measurement $\mathbf{l}_{i,j}$, and W_s^k that of the k^{th} IMU error. The reprojection error is:

$$\mathbf{e}_r^{i,j,k} = \mathbf{z}^{i,j,k} - \boldsymbol{\pi}_i \left(\mathbf{T}_{C_i S}^k \mathbf{T}_{SW}^k \mathbf{l}^{i,j} \right)$$

where $\mathbf{z}^{i,j,k}$ is the measured image coordinate of the j^{th} landmark on the i^{th} sensor at the k^{th} frame. We use standard IMU kinematics and biases model (see [52] for example) to predict the current state based on the previous state. Then, the IMU error terms are computed as the difference between the prediction based on the previous state and the actual state. For orientation, a simple multiplicative minimal error is used. For details, we refer the reader to [94].

The optimization is carried out not on all the frames observed but on a bounded set

Sequence	**Proposed (Fr + E + I)**		**E + I**		**Fr + I**	
	Mean Position Error (%)	Mean Yaw Error (deg/m)	Mean Position Error (%)	Mean Yaw Error (deg/m)	Mean Position Error (%)	Mean Yaw Error (deg/m)
boxes_6dof	**0.30**	**0.04**	0.44	0.05	0.30	0.06
boxes_translation	0.27	**0.02**	0.76	0.05	**0.17**	0.03
dynamic_6dof	**0.19**	0.10	0.38	**0.06**	0.62	0.10
dynamic_translation	**0.18**	**0.15**	0.59	0.16	0.67	0.26
hdr_boxes	**0.37**	**0.03**	0.67	0.09	0.78	0.17
hdr_poster	0.31	0.05	0.49	**0.04**	**0.28**	0.08
poster_6dof	**0.28**	**0.07**	0.30	0.08	0.59	0.11
poster_translation	**0.12**	0.04	0.15	**0.04**	0.23	0.08
shapes_6dof	**0.10**	**0.04**	0.48	0.06	0.17	0.05
shapes_translation	**0.26**	0.06	0.41	**0.04**	0.29	0.11

Table E.1 – Accuracy of the proposed approach using frames (Fr), events (E) and IMU (I), against using events and IMU, and using frames and IMU.

of frames composed of M keyframes (we use the same keyframe selection criterion as [149]), and a sliding window containing the last K frames. In between frames, the prediction for the sensor state is propagated using the IMU measurements. We employ the Google Ceres [2] optimizer to carry out the optimization.

Notice that with this formulation we avoid an explicit switching policy between standard and event camera: the optimization naturally uses the best sensing modalities available.

Additional Implementation Details

Initialization We assume that the sensor remains static during the initialization phase of the pipeline, during one or two seconds. We collect a set of inertial measurements and use them to estimate the initial attitude (pitch and roll) of the sensor, as well as to initialize the gyroscope and accelerometer biases.

No-Motion Prior for Almost-Still Motions When the sensor is still, no events are generated (except noise events). To handle this case in our pipeline, we add a strong zero velocity prior to the optimization problem whenever the event rate falls below a threshold, thus forcing the sensor to be still. We used a threshold in the order of 10^3 events/s in our experiments, and measured the event rate using windows of 20 ms.

E.3.2 Evaluation

We evaluate the proposed pipeline quantitatively on the Event Camera Dataset [124], which features various scenes with ground truth tracking information. In particular, it contains extremely fast motions and scenes with very high dynamic range, recorded with the DAVIS[1] [18] sensor. As in [211], we only use the datasets from the Event

[1] https://inilabs.com/products

Sequence	**Proposed (Fr + E + I)**		**State-of-the-art (E + I) [149]**	
	Mean Position Error (%)	Mean Yaw Error (deg/m)	Mean Position Error (%)	Mean Yaw Error (deg/m)
boxes_6dof	**0.30**	**0.04**	0.36	0.11
boxes_translation	**0.27**	**0.02**	0.31	0.08
dynamic_6dof	**0.19**	**0.10**	0.56	0.41
dynamic_translation	**0.18**	0.15	0.39	**0.06**
hdr_boxes	**0.37**	**0.03**	0.59	0.20
hdr_poster	**0.31**	**0.05**	0.33	0.19
poster_6dof	**0.28**	**0.07**	0.40	0.16
poster_translation	**0.12**	**0.04**	0.46	0.10
shapes_6dof	**0.10**	**0.04**	0.42	0.18
shapes_translation	**0.26**	**0.06**	0.50	0.13

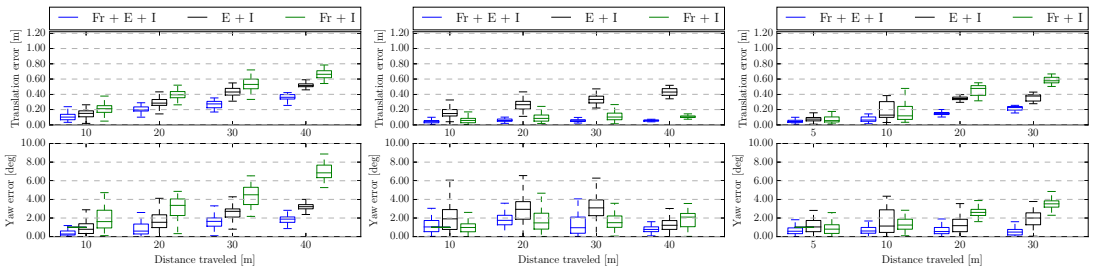
Table E.2 – Accuracy of the proposed approach using frames (Fr), events (E) and IMU (I), against [149], which uses events and IMU.

Camera Dataset that are relevant for Visual-Inertial Odometry. Specifically, we exclude the rotational only datasets, as well as the datasets without inertial measurements.

The DAVIS sensor embeds a 240×180 pixels event camera with a 1kHz IMU and also delivers standard frames at 24Hz. Events, standard frames, and IMU measurements are synchronized on hardware. The IMU is delayed by a constant time offset in the order of 2.5ms compared to the events and standard frames (because of the low-pass filter of the IMU). We estimated this delay using Kalibr [56].

To evaluate the results, the estimated and ground truth trajectories are aligned with a 6-DOF transformation in SE3, using 5 seconds of the trajectory (starting at second 3 and ending at second 8). Then, we compute the mean position error (Euclidean distance) and the yaw error as percentages of the total traveled distance. Due to the observability of the gravity direction, the error in pitch and roll is constant and comparable for each pipeline. Thus we omit them for compactness.

Table E.1 shows the results obtained when running the pipeline in its proposed mode, using standard frames (Fr), events (E), and IMU (I). To further quantify the accuracy gained by using events and frames (plus IMU), compared to using only events or only frames (plus IMU), we run our proposed pipeline using the three different combinations, and report the results in Table E.1. Additionally, in Fig. E.3, we use the relative error metrics proposed in [66], which evaluate the relative error by averaging the drift over trajectories of different lengths. Using jointly standard frames, events and IMU leads to an average position accuracy improvement of 85% compared to using frames and IMU only, and 130% against using events and IMU only. Notice that the Event Camera Dataset was made to showcase the situations where an event camera would be more



(a) `hdr_boxes` **(b)** `shapes_6dof` **(c)** `dynamic_6dof`

Figure E.3 – Comparison of the proposed approach, using frames (Fr), events (E), and IMU (I), on three datasets from the Event Camera Dataset [124]. The graphs show the relative errors measured over different segments of the trajectory as proposed in [66]. Additional plots for all the datasets are provided in the supplementary material.

useful. Nevertheless, datasets like boxes_translation and shapes_6dof show that using standard frames might still be advantageous compared to using only events, as can be seen in Table E.1 and the detailed analysis in the supplementary material.

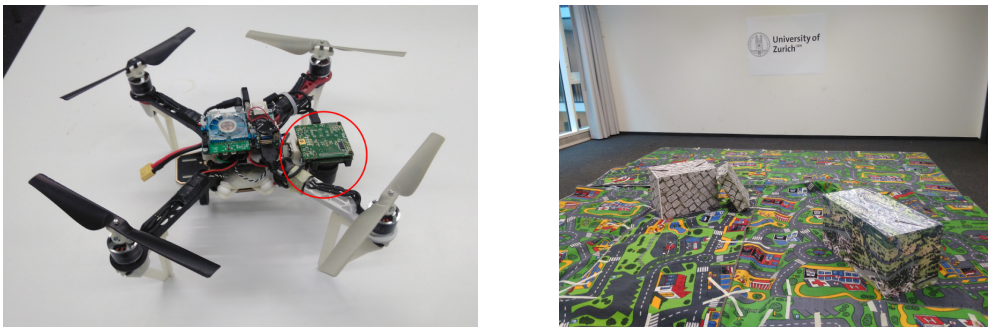
Table E.2 provides a comparison between our approach and the state-of-the-art [149].

The events plus IMU pipeline in Table E.1 is not the same as [149] in Table E.2; the former generates event frames at a fixed rate, while the latter generates them at a rate that depends on the event rate, we refer the reader to [149] for further details. Moreover, the parameters both pipelines share do not necessarily have the same values. These reasons account for the different results in Table E.1 (E + I) and Table E.2 (state-of-the-art E + I).

To the best of our knowledge, we are the first to report results on the Event Camera Dataset using all three sensor modalities. It can be seen that our approach, that uses frames and events, is better in terms of accuracy on almost all the datasets.

E.4 Quadrotor flight with an event camera

In order to show the potential of our hybrid, frame-and-event–based pipeline in a real scenario, we ran our approach onboard an autonomous quadrotor and used it to fly autonomously in challenging conditions. We first start by describing in detail the quadrotor platform we built (hardware and software) in section E.4.1 before turning to the specific in-flight experiments (section E.4.2).



(a) Quadrotor platform used. **(b)** Preview of the flying room.

Figure E.4 – Quadrotor platform used for the flight experiments, and preview of the room.

E.4.1 Aerial Platform

Platform

We built our quadrotor from selected off-the-shelf components and custom 3D printed parts (Fig. E.4). Our quadrotor relies on a DJI frame, with RCTimer motors and AR drone propellers. The electronic parts of our quadrotor comprise a PX4FMU autopilot [113]. In addition, our quadrotor is equipped with an Odroid XU4 computer, which contains a 2.0 GHz quad-core processor running Ubuntu 14.04 and ROS [145]. Finally, a DAVIS 240C sensor, equipped with a 70° field-of-view lens, is mounted on the front of the quadrotor, looking downwards. The sensor is connected to the Odroid computer via an USB 2.0 cable, and transmits events, standard frames, and inertial measurements, which we use to compute the state estimate on the Odroid using our proposed pipeline. Since the available ROS driver for the DAVIS did not come with an auto-exposure for the standard camera, we implemented an auto-exposure algorithm and made it available open-source for the community to use.[2] It is based on a simple proportional controller that controls the mean image intensity to a desired value (we used a value of 70 in our experiments).

Control

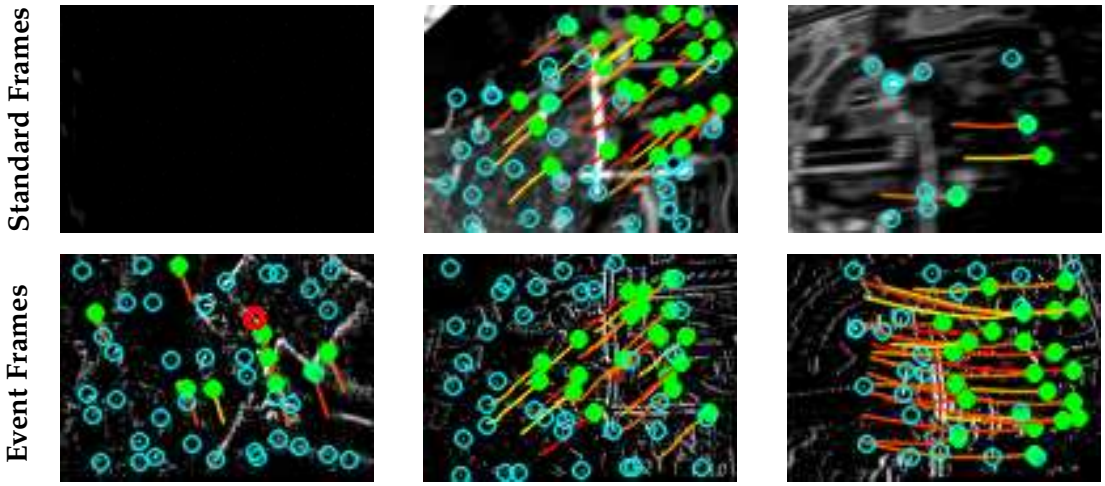
To follow reference trajectories and stabilize the quadrotor, we use the cascaded controllers presented in [48]. The high-level controller running on the Odroid includes a position controller and an attitude controller, while the low-level controller on the PX4 contains a body rate controller. The high-level controller takes a reference trajectory as input and computes desired body rates that are sent to the low-level controller. The low-level controller, in turn, computes the desired rotor thrusts using a feedback linearizing control scheme with the closed-loop dynamics of a first-order system. Details of the controllers can be found in [48].

[2] Available in the DAVIS ROS driver: https://github.com/uzh-rpg/rpg_dvs_ros

E.4.2 Flight Experiments

We present three flight experiments that demonstrate that our system is able to fly a quadrotor in challenging conditions: (i) flying indoors while switching on and off the light (which is challenging because of the abrupt large change of illumination caused by the switching of the light and the very low light present in the room after the artificial light is turned off); (ii) while performing fast circles in a low-lit room; (iii) hovering at the same position (which is challenging for the event camera because close to no motion). In the first case, when the light is off, the standard frames are completely black. In the second one, the speed of the quadrotor induces severe motion blur on the standard frames. Nevertheless, in both cases, the events are left unaffected and our pipeline is able to successfully exploit them to provide robust state estimation. In the third case, instead, there is almost no motion, which makes it difficult for the event camera to track reliable features. Nevertheless, the frames are left unaffected and our pipeline is able to successfully exploit them to provide robust state estimation.

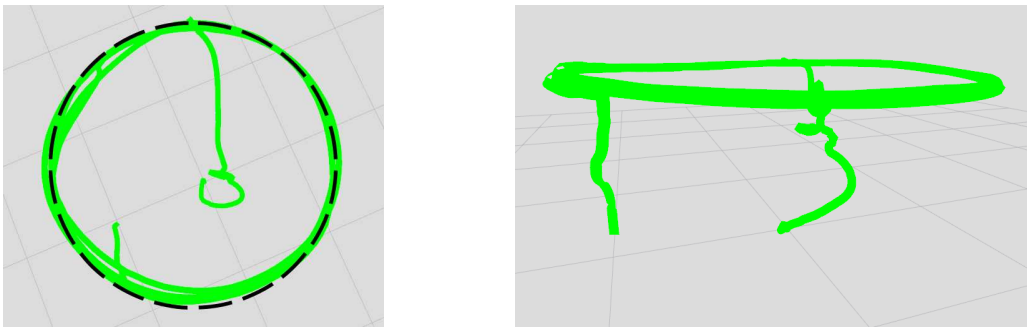
These three experiments are best appreciated in video attachment.[3]



(a) Low-light. **(b)** Good lighting, moderate speed. **(c)** Motion blur.

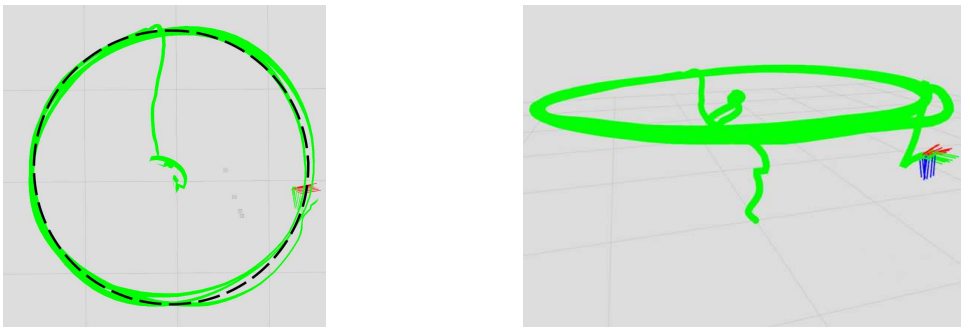
Figure E.5 – Example feature tracks in various conditions, on the standard frames (top row) and the virtual event frames (bottom row). Every column corresponds to the same timestamp, a frame from the top row has a corresponding event frame on the bottom row. The green solid dots are persistent features, and the blue dots correspond to candidate features. The tracks are shown as colored lines.

[3] http://rpg.ifi.uzh.ch/ultimateslam.html



(a) Top view. **(b)** Perspective view.

Figure E.6 – Experiment 1: switching the light off and on. The trajectory estimated by our pipeline is the green line. The commanded trajectory is the superimposed black dashed line.



(a) Top view. **(b)** Perspective view.

Figure E.7 – Experiment 2: Fast circles in a low-lit room. The trajectory estimated by our pipeline is the green line. The commanded trajectory is the superimposed black dashed line.

Switching the light off and on, in flight

In this experiment, we pushed our pipeline to the limit by outright switching the room light off while autonomously flying in circles. The only remaining light was residual light coming from the windows (very little light, but still enough for the event camera to work). The standard frames become completely black when the light goes off (top frame in Fig. E.5a), making them useless for state estimation. By contrast, the events still carry enough information (albeit noisier) to allow reasonable feature tracks (bottom frame Fig. E.5a). Switching the light off effectively forces the pipeline to rely only on events and inertial measurements. Note that the abrupt illumination change caused by switching the lights on and off makes almost every pixel fire events. Although we do not explicitly handle this particular case, in practice we observed no substantial decrease in accuracy when this occurs as features are quickly re-initialized.

The trajectory flown by the quadrotor is shown in Fig. E.6.

Fast Circles in a Low-lit Room

In this experiment, the quadrotor autonomously flies a circular trajectory with increasing speed in a closed room with little light (Fig. E.1); we carried this experiment during the night and set a low lighting in the room. The circular trajectory commanded to the quadrotor is parametrized by its radius and the desired angular velocity. We set the angular velocity to 1.4 rad/s on a circle of 1.2 m radius, corresponding to a top linear velocity of 1.68 m/s. The circle height was 1.0 m. At this speed and height, the optical flow generated on the image plane amounts to approximately 340 pixels/s.

While the speed remains moderate at the beginning of the trajectory (below 1.2 m/s), standard frames do not suffer from motion blur and our pipeline indeed tracks features in both the standard frames and the event frames (cf. top and bottom frames in Fig. E.5b, respectively). Nevertheless, as soon as the speed increases, the standard frames start to suffer from severe motion blur, as shown in the top frame of Fig. E.5c, and the number of features tracked in the standard frames significantly decreases. Conversely, the events allow synthesizing motion-free virtual event frames, which, in turn, allow keeping reliable feature tracks (bottom frame in Fig. E.5c).

In Fig. E.7, both the desired and estimated trajectories are shown for comparison. Interestingly, the right side of the trajectory is slightly noisier than the left side. This turns out to match well with the light configuration in the room: the left side of the room was indeed more illuminated than the right side (visible in Fig. E.1). This is coherent with the quantitative experiments presented in section E.3.2: the increase of the quality of the standard frames on the room side with more light correlates directly to an increase of accuracy of the pipeline.

Hovering

We also provide qualitative experiments to show how our pipeline performs close to no-motion conditions, typically encountered when a drone is hovering.

First, we command the drone to hover while using the events-only pipeline and observe that the state estimate drifts. We then command the drone to hover while using both the images and the event frames, and observe that the drone successfully keeps its position with no noticeable drift.

The difference lies in that features are tracked successfully on the standard frames, while they are lost on the event frames. This is because, unlike standard cameras, the appearance of the features tracked by the event camera may change "drastically" with the direction of the motion, which is exactly what happens when hovering: vibrations induce frequent changes of motion direction, which reduce the length of the feature tracks from the event streams, leading to increased drift.

E.5 Conclusions

We introduced the first hybrid pipeline that fuses events, standard frames, and inertial measurements to yield robust and accurate state estimation. We also reported results using these three sensing modalities on the Event Camera Dataset [124] and demonstrated an accuracy boost of 130 % compared to using only events plus IMU, and a boost of 85 % compared to using only standard frames plus IMU. Furthermore, we successfully integrated the proposed pipeline for state estimation onboard a computationally-constrained quadrotor and used it to realize, to the best of our knowledge, the first closed-loop flight of a quadrotor using an event camera. Finally, in a set of specific experiments, we showed that our hybrid pipeline is able to leverage the properties of the standard camera and the event camera to provide robust tracking when flying in multiple conditions, such as hovering, flying in fast circles or flying in a low-lit room.

F High Speed and High Dynamic Range Video with an Event Camera

This work first appeared as a conference paper:

H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. "Events-to-Video: Bringing Modern Computer Vision to Event Cameras". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019

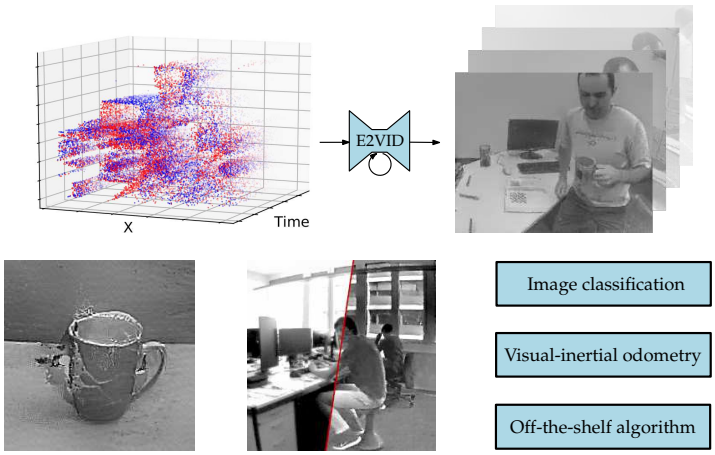
It was later extended and published in Transactions on Pattern Analysis and Machine Intelligence (T-PAMI). The version presented here is reprinted, with permission, from:

H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. "High Speed and High Dynamic Range Video with an Event Camera". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2019). DOI: 10.1109/TPAMI.2019.2963386

High Speed and High Dynamic Range Video with an Event Camera

Henri Rebecq, Rene Ranftl, Vladlen Koltun and Davide Scaramuzza

Abstract — Event cameras are novel sensors that report brightness changes in the form of a stream of asynchronous "events" instead of intensity frames. They offer significant advantages with respect to conventional cameras: high temporal resolution, high dynamic range, and no motion blur. While the stream of events encodes in principle the complete visual signal, the reconstruction of an intensity image from a stream of events is an ill-posed problem in practice. Existing reconstruction approaches are based on hand-crafted priors and strong assumptions about the imaging process as well as the statistics of natural images. In this work we propose to learn to reconstruct intensity images from event streams directly from data instead of relying on any hand-crafted priors. We propose a novel recurrent network to reconstruct videos from a stream of events, and train it on a large amount of simulated event data. During training we propose to use a perceptual loss to encourage reconstructions to follow natural image statistics. We further extend our approach to synthesize color images from color event streams. Our quantitative experiments show that our network surpasses state-of-the-art reconstruction methods by a large margin in terms of image quality ($>20\%$), while comfortably running in real-time. We show that the network is able to synthesize high framerate videos ($>$ 5,000 frames per second) of high-speed phenomena (*e.g.* a bullet hitting an object) and is able to provide high dynamic range reconstructions in challenging lighting conditions. As an additional contribution, we demonstrate the effectiveness of our reconstructions as an intermediate representation for event data. We show that off-the-shelf computer vision algorithms can be applied to our reconstructions for tasks such as object classification and visual-inertial odometry and that this strategy consistently outperforms algorithms that were specifically designed for event data.



(a) High framerate video (b) HDR video (c) Downstream applications

Figure F.1 – Our network converts a spatio-temporal stream of events with microsecond temporal resolution (top left) into a high-quality video (top right). This enables synthesis of videos of high-speed phenomena such as a bullet piercing a mug (a), or scenes with high dynamic range (b). The reconstructions can also be used as input to off-the-shelf computer vision algorithms, thereby serving as an intermediate representation between event data and mainstream computer vision (c). The images in the figure were produced by the presented technique.

Multimedia Material

A video of the experiments, as well as the reconstruction code and a pretrained model are available at: http://rpg.ifi.uzh.ch/E2VID.

F.1 Introduction

Event cameras are bio-inspired vision sensors that work radically differently from conventional cameras. Instead of capturing intensity images at a fixed rate, event cameras measure *changes* of intensity asynchronously at the time they occur. This results in a stream of *events*, which encode the time, location, and polarity (sign) of brightness changes (Fig. F.2 - top). Event cameras such as the Dynamic Vision Sensor (DVS) [98] possess outstanding properties when compared to conventional cameras. They have a very high dynamic range (140 dB versus 60 dB), do not suffer from motion blur, and provide measurements with very high temporal resolution. Event cameras thus provide a viable alternative (or complementary) sensor in conditions that are challenging for conventional cameras.

In theory, the stream of events contains the entire visual signal – in a highly compressed

form – and could thus be decompressed to recover a video with arbitrarily high framerate and high dynamic range. However, real event cameras are noisy and differ significantly from the ideal camera model, which renders the reconstruction problem ill-posed. Naive integration of the event stream leads to very fast degradation of image quality due to accumulating noise. As a remedy, earlier works have proposed handcrafted image priors to constrain the problem [10, 12, 125, 171]. However, these priors make strong assumptions about the statistics of natural images, leading to unrealistic reconstructions and artifacts. As a result, high-quality video reconstruction from event data has so far not been convincingly demonstrated.

In this work, we propose to bridge this gap by learning high-quality video reconstruction from sparse event data using a recurrent neural network. In contrast to previous image reconstruction approaches [10, 125, 171], we do not embed handcrafted smoothness priors into our reconstruction framework. Instead, we learn video reconstruction from events using a large amount of simulated event data, and encourage the reconstructed images to have natural image statistics through a perceptual loss that operates on mid-level image features. Our network outperforms prior methods in terms of image quality by a large margin (>20% improvement), demonstrating for the first time event-camera-based synthesized video sequences that are qualitatively on par with conventional cameras in terms of visual appearance. Our approach opens the door to a variety of applications, some of which we examine in this paper.

We explore the possibility of using an event camera to capture videos in scenarios that are challenging for conventional cameras. First, we show that our network can leverage the high temporal resolution of event data to synthesize high framerate (> 5,000 frames per second) videos of high-speed physical phenomena (Section F.5.1). The resulting videos reveal details that are beyond the grasp of the naked eye or conventional cameras, which operate at a few hundred frames per second at best. Second, we show that our reconstructions preserve the high dynamic range of event cameras (Section F.5.2), thus offering a viable alternative to conventional sensors in high dynamic range settings. We additionally present a simple strategy to synthesize color videos using a recent color event camera [116, 115], without the need to retrain the network (Section F.5.3).

Beyond pure imaging, we also consider using our approach for downstream applications. Since the output of an event camera is an asynchronous stream of events (a representation that is fundamentally different from natural images), existing computer vision techniques cannot be directly applied to this data. As a consequence, a number of algorithms have been specifically tailored to leverage event data, either processing the event stream in an event-by-event fashion [29, 30, 13, 123, 80, 61], or by building intermediate, "image-like" representations from event data [88, 184, 213, 209, 213]. In the spirit of the second category of methods, we explore the use of our image reconstructions as a novel representation for event data in Section F.5.4. Specifically, we apply existing computer vision algorithms to images reconstructed from event data.

We focus on object classification and visual-inertial odometry with event data, and show that this strategy consistently yields state of the art results in terms of accuracy. This suggests that high-quality reconstructions can be used as a bridge that brings the main stream of computer vision research to event cameras: mature algorithms, modern deep network architectures, and weights pretrained from large natural image datasets.

In summary, the contributions of this work are:

- A novel recurrent network that reconstructs video from a stream of events and outperforms the state of the art in terms of image quality by a large margin.
- We establish that networks trained from simulated event data generalize remarkably well to real events.
- Qualitative results showing that our method can be used in a variety of settings, such as high framerate video synthesis of high-speed phenomena (Section F.5.1), reconstruction of high dynamic range video (Section F.5.2), and reconstruction of color video (Section F.5.3).
- Application of our method to two downstream problems: object classification and visual-inertial odometry from event data. Our method outperforms state-of-the-art algorithms designed specifically for event data in both applications.

F.2 Related Work

Because of its far reaching applications, events-to-video reconstruction is a popular topic in the event camera literature. The first evidence that it is possible to recover intensity information from event data was provided by [30] and [79], in the context of rotation estimation with an event camera. They showed how to reconstruct a single image from a large set of events collected by an event camera moving through a static scene and exploited the fact that every event provides one equation relating the intensity gradient and optic flow through brightness constancy [64]. Specifically, Cook *et al*. [30] used bio-inspired, interconnected networks to simultaneously recover intensity images, optic flow, and angular velocity from an event camera undergoing small rotations. Kim *et al*. [79] developed an Extended Kalman Filter to reconstruct a 2D panoramic gradient image (later upgraded to a full intensity frame by 2D Poisson integration) from a rotating event camera. They later extended their approach to static 3D scenes and 6 degrees-of-freedom (6DOF) camera motion [80]. Bardow *et al*. [10] proposed to estimate optic flow and intensity *simultaneously* from sliding windows of events through a variational energy minimization framework. They showed the first video reconstruction framework from events that is applicable to dynamic scenes. However, their energy minimization framework employs multiple hand-crafted regularizers, which can result in severe loss of detail in the reconstructions.

Recently, methods based on direct event integration have emerged. These approaches do not rely on any assumption about the scene structure or motion dynamics, and can naturally reconstruct videos at arbitrarily high framerates. Munda *et al*. [125] cast intensity reconstruction as an energy minimization problem defined on a manifold induced by the event timestamps. They combined direct event integration with total variation regularization and achieved real-time performance on the GPU. Scheerlinck *et al*. [171] proposed to filter the events with a high-pass filter prior to integration. They demonstrated video reconstruction results that are qualitatively comparable with [125] while being computationally more efficient. While these approaches currently define the state-of-the-art, both suffer from artifacts which are inherent to direct event integration. The reconstructions suffer from "bleeding edges" caused by the fact that the contrast threshold (the minimum brightness change of a pixel to trigger an event) is neither constant nor uniform across the image plane. Additionally, pure integration of the events can in principle only recover intensity up to an unknown initial image $\mathcal{I}_0$ which causes "ghosting" effects where the trace of the initial image remains visible in the reconstructed sequence.

Barua *et al*. [12] proposed a learning-based approach to reconstruct intensity images from events. They used K-SVD [3] on simulated data to learn a dictionary that maps small patches of integrated events to an image gradient and used Poisson integration to recover the intensity image. In contrast, we do not reconstruct individual intensity images from small windows of events, but synthesize a temporally consistent video from a long stream of events (several seconds) using a recurrent network. Instead of mapping event patches to a dictionary of image gradients, we learn pixel-wise intensity estimation directly.

Finally, some methods alternatively proposed to consider both events and conventional frames to perform video reconstruction, *i.e*. they do not synthesize video from events only, but rather temporally upsample a stream of images using the events in between images [19, 171, 180]. While these methods can potentially generate higher quality videos thanks to the use of frames directly, they fail to produce high quality outputs in high speed and high dynamic range scenarios, for which the conventional frames are spoilt. For this reason, we purposefully choose to use only event data in this work.

Despite the body of work on events-to-video reconstruction, downstream vision applications based on the reconstructions have, to the best of our knowledge, never been demonstrated prior to our work.

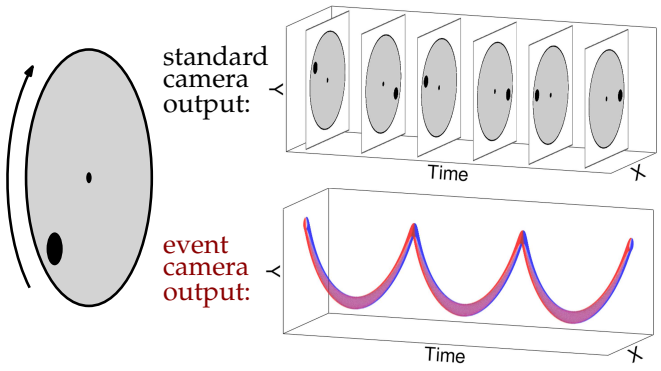


Figure F.2 – Comparison of the output of a conventional camera and an event camera looking at a black disk on a rotating circle. While a conventional camera captures frames at a fixed rate, an event camera transmits the brightness changes continuously in the form of a spiral of events in space-time (red: positive events, blue: negative events). Figure inspired by [123].

F.3 Video Reconstruction

An event camera consists of independent pixels that respond to changes in the spatio-temporal brightness signal $L(\mathbf{x}, t)$[1] and transmit the changes in the form of a stream of asynchronous events (Fig. F.2). For an ideal sensor, an event $\mathbf{e}_i = (\mathbf{u}_i, t_i, p_i)$ is triggered at pixel $\mathbf{u}_i = (x_i, y_i)^T$ and time t_i when the brightness change since the last event at the pixel reaches a threshold $\pm C$. However, C is in reality neither constant nor uniform across the image plane. Rather, it strongly varies depending on factors such as the sign of the brightness change [61], the event rate (because of limited pixel bandwidth) [19], and the temperature [203]. Consequently, events cannot by directly integrated to recover accurate intensity images in practice.

F.3.1 Overview

Our goal is to translate a continuous stream of events into a sequence of images $\{\hat{\mathcal{I}}_k\}$, where $\hat{\mathcal{I}}_k \in [0,1]^{W \times H}$. To achieve this, we partition the incoming stream of events into sequential (non-overlapping) spatio-temporal windows $\varepsilon_k = \{\mathbf{e}_i\}$, for $i \in [0, N-1]$, each containing a fixed number N of events. The reconstruction function is implemented by a recurrent convolutional neural network, which maintains and updates an internal state $\mathbf{s}_k$ through time. For each new event sequence ε_k, we generate a new image $\hat{\mathcal{I}}_k$ using the network state $\mathbf{s}_{k-1}$ (see Fig. F.3) and update the state $\mathbf{s}_k$. We train the network in supervised fashion, using a large amount of simulated event sequences with corresponding ground-truth images.

[1] Event cameras respond in fact to logarithmic brightness changes, i.e. $L = \log E$ where E is the *irradiance*.

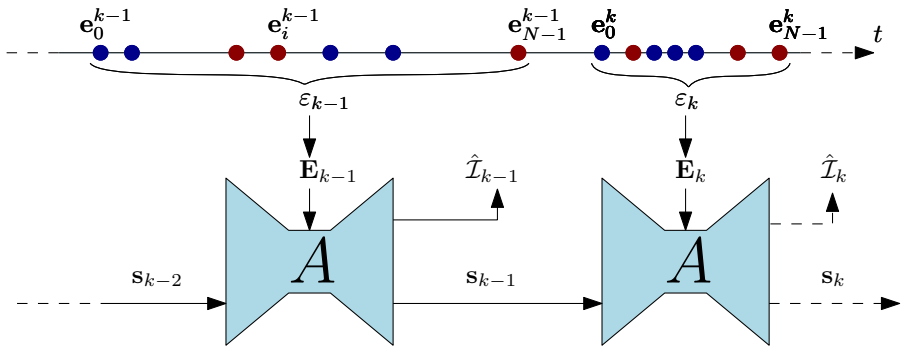


Figure F.3 – Overview of our approach. The event stream (depicted as red/blue dots on the time axis) is split into windows ε_k containing multiple events. Each window is converted into a 3D event tensor $\mathbf{E}_k$ and passed through the network, together with the previous state $\mathbf{s}_{k-1}$ to generate a new image reconstruction $\hat{\mathcal{I}}_k$ and updated state $\mathbf{s}_k$. In this example, each window ε_k contains a fixed number of events $N = 7$.

F.3.2 Event Representation

In order to be able to process the event stream using the convolutional recurrent network, we need to convert ε_k into a fixed-size tensor representation $\mathbf{E}_k$. A natural choice is to encode the events in a spatio-temporal voxel grid [215]. The duration $\Delta T = t_{N-1}^k - t_0^k$ spanned by the events in ε_k is discretized into B temporal bins. Every event distributes its polarity p_i to the two closest spatio-temporal voxels as follows:

$$\mathbf{E}(x_l, y_m, t_n) = \sum_{\substack{x_i = x_l \\ y_i = y_m}} p_i \max(0, 1 - |t_n - t_i^*|), \tag{F.1}$$

where $t_i^* \triangleq \frac{B-1}{\Delta T}(t_i - t_0)$ is the normalized event timestamp. We use $B = 5$ temporal bins.

F.3.3 Training Data

Our network requires training data in the form of event sequences with corresponding ground-truth image sequences. However, there exists no large-scale dataset with event data and corresponding ground-truth images. Furthermore, images acquired by a conventional camera would provide poor ground truth in scenarios where event cameras excel, namely high dynamic range and high-speed scenes. For these reasons, we propose to train the network on synthetic event data, and show subsequently (in Section F.4) that our network generalizes to real event data.

We use the event simulator ESIM [148], which allows simulating a large amount of event data reliably. ESIM renders images along the camera trajectory at high framerate, and interpolates the brightness signal at each pixel to approximate the continuous intensity

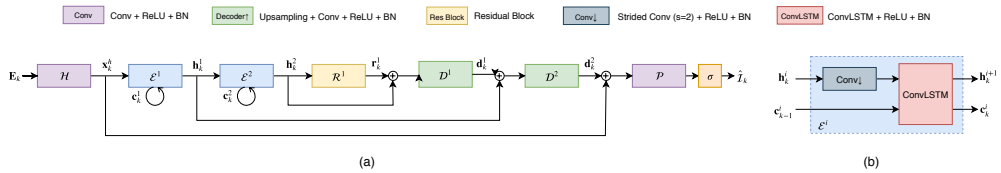


Figure F.4 – We use a fully convolutional, UNet-like [163] architecture (a), composed of N_E recurrent encoder layers (b), followed by N_R residual blocks and N_E decoder layers, with skip connections between symmetric layers. Encoders are composed of a strided convolution (stride 2) followed by a ConvLSTM [181]. Decoder blocks perform bilinear upsampling followed by a convolution. ReLU activations and batch normalization [75] are used after each layer (except the last prediction layer, for which a sigmoid activation is used). In this diagram, $N_E = 2$ and $N_R = 1$.

signal needed to simulate an event camera. Consequently, ground-truth images $\mathcal{I}$ are readily available. We map MS-COCO images [99] to a 3D plane and simulate the events triggered by random camera motion within this simple 3D scene. Using MS-COCO images allows capturing a much larger variety of scenes than is available in any existing event camera dataset. We set the camera sensor size to 240×180 pixels (to match the resolution of the DAVIS240C sensor used in our evaluation [18]). Note that inference can be performed at arbitrary resolutions since we will use a fully-convolutional network. Examples of generated synthetic event sequences are presented in the supplement.

We further enrich the training data by simulating a different set of positive and negative contrast thresholds for each simulated scene (sampled according to a normal distribution with mean 0.18 and standard deviation 0.03, values based on [79]). This data augmentation prevents the network from learning to naively integrate events, which would work well on noise-free, simulated data, but would generalize poorly to real event data (for which the assumption of a fixed contrast threshold does not hold).

We generate 1,000 sequences of 2 seconds each, which results in approximately 35 minutes of simulated event data. Note that the simulated sequences contain only globally homographic motion (i.e. there is no independent motion in the simulated sequences). Nevertheless, our network generalizes surprisingly well to scenes with arbitrary motions, as will be shown in Sections F.4 and F.5.

F.3.4 Network Architecture

Our neural network is a recurrent, fully convolutional network that was inspired by the UNet [163] architecture. An overview is shown in Fig. F.4. It is composed of a head layer ($\mathcal{H}$), followed by N_E recurrent encoder layers ($\mathcal{E}^i$), N_R residual blocks ($\mathcal{R}^j$), N_E decoder layers ($\mathcal{D}^l$), and a final image prediction layer ($\mathcal{P}$). Following [213], we use skip connections between symmetric encoder and decoder layers. The number of output

channels is N_b for the head layer $\mathcal{H}$, and is doubled after each encoder layer (thus, the final encoder has $N_b \times 2^{N_E}$ output channels). The prediction layer performs a depthwise convolution (one output channel, kernel size 5), followed by a sigmoid layer to produce an image prediction. Encoder layers $\mathcal{E}^i$ (Fig. F.4(b)) consist of a 2D downsampling convolution (kernel size: 5, stride: 2) followed by a ConvLSTM [181], with a kernel size of 3, and whose number of input and hidden layers is the same as the preceding downsampling convolution. Each encoder maintains a state $\mathbf{c}_k^i$ which is updated at every iteration, and initialized to zero at the first iteration ($k = 0$). The intermediate residual blocks [71] use a kernel size of 3. Each decoder layer consists of bilinear upsampling followed by a convolution with kernel size 5. Finally, we use the ReLU activation (for every layer except the final prediction) and batch normalization [75]. In total, the network has about 10.7 million parameters, which corresponds to a size of 42.9 MB on disk.

We used $N_E = 3$, $N_R = 2$, $N_b = 32$ and element-wise sum for the skip connection. In Section F.6.1, we motivate these choices of hyperparameters as the result of a search over multiple network architectures.

During training we unroll the network for L steps. We use $L = 40$. Note that this is in contrast to [153] which trains on significantly shorter event sequences ($L = 8$). The architecture in [153] is based on a vanilla recurrent (RNN) architecture, which suffers from vanishing gradients during back propagation through time on long sequences. By contrast, our network uses stacked ConvLSTM gates which prevent these issues and allows us to train on longer sequences. In Section F.6, we show that our architecture based on LSTM improves the temporal stability of the network. We trained the network for 300 epochs, which took about 40 hours on an NVIDIA RTX 2080Ti GPU.

F.3.5 Loss

We use a combination of an image reconstruction loss and a temporal consistency loss.

Image Reconstruction Loss. The image reconstruction loss ensures that the reconstructed image is similar to the target image. While a direct pixel-wise loss such as the mean squared error (MSE) could be used, such losses are known to produce blurry images [76]. Instead, we use a perceptual loss (specifically, the calibrated perceptual loss LPIPS [205]). The perceptual loss passes the reconstructed image and the target image through a VGG network [183] that was trained on ImageNet [167], and averages the distances between VGG features across multiple layers. By minimizing LPIPS, our network effectively learns to endow the reconstructed images with natural statistics (i.e. with features close to those of natural images). Our reconstruction loss is computed as $\mathcal{L}_k^R = d(\hat{\mathcal{I}}_k, \mathcal{I}_k)$, where d denotes the LPIPS distance [205].

Temporal Consistency Loss. In our previous work [153], the network relied on the recurrent connection to naturally enforce temporal consistency between successive reconstructions. However, some temporal artifacts remained, notably some slight blinking that was especially noticeable in homogeneous image regions. To address this issue, we introduce an explicit temporal consistency loss, the beneficial effect of which will be demonstrated in Section F.6. Our temporal consistency loss is based on [89]. Given optical flow maps $\mathcal{F}_{k-1}^{k}$ between successive frames, the temporal loss is computed as the warping error between two successive reconstructions:

$$\mathcal{L}_k^{TC} = M_{k-1}^k \|\hat{\mathcal{I}}_k - \mathcal{W}_{k-1}^k(\hat{\mathcal{I}}_{k-1})\|_1, \tag{F.2}$$

where $\mathcal{W}_{k-1}^k(\hat{\mathcal{I}}_{k-1})$ is the result of warping the reconstruction $\hat{\mathcal{I}}_{k-1}$ to $\hat{\mathcal{I}}_k$ using the optical flow $\mathcal{F}_{k-1}^k$, and $M_{k-1}^k = \exp(-\alpha \|\mathcal{I}_k - \mathcal{W}_{k-1}^k(\mathcal{I}_{k-1})\|_2^2)$ is a weighting term that helps to mitigate the effect of occlusions (this term is small when the warping error in the ground truth images $\mathcal{I}_k$ is high, which happens predominantly at occlusions). We set $\alpha = 50$ in our experiments.

Note that the optical flow maps are only required at training time, but not at inference time. The final loss is a weighted sum of the reconstruction and temporal losses:

$$\mathcal{L} = \sum_{k=0}^{L} \mathcal{L}_k^R + \lambda_{TC} \sum_{k=L_0}^{L} \mathcal{L}_k^{TC}, \tag{F.3}$$

where $\lambda_{TC} = 5$ (this value was chosen empirically to balance the range of values taken by both losses), and $L_0 = 2$ (the first few samples of each sequence are ignored in the computation of the temporal loss to leave time for the reconstruction to converge).

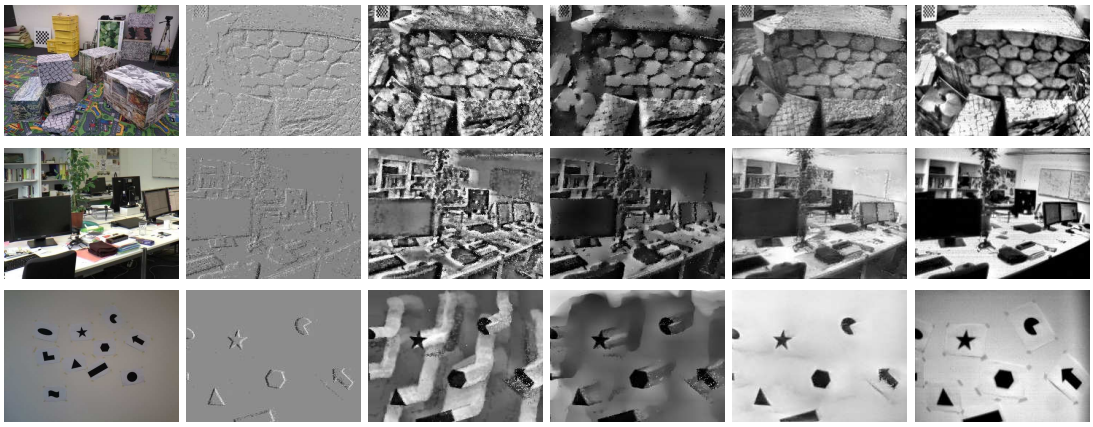
F.3.6 Training Procedure

We split the synthetic sequences into 950 training sequences and 50 validation sequences. The input event tensors are normalized such that the mean and standard deviation of the nonzero values in each tensor is 0 and 1, respectively. We augment the training data using random 2D rotations (in the range of ± 20 degrees), horizontal and vertical flips, and random cropping (with a crop size of 128×128).

We implement our network using PyTorch [134] and use ADAM [81] with a learning rate of 0.0001. We use a batch size of 2 and train for 160 epochs (320,000 iterations).

F.3.7 Post-processing

While the sigmoid activation guarantees that the resulting image prediction $\hat{\mathcal{I}}$ takes values between 0 and 1, we observe that the range of output values often does not



(a) Scene (b) Events (c) HF (d) MR (e) Ours (f) Ground truth

Figure F.5 – Comparison of our method with MR and HF on sequences from [124]. Our network is able to reconstruct fine details well (textures in the first row), while avoiding common artifacts (e.g. the "bleeding edges" in the third row).

span the entire range, i.e. the reconstructions can have low contrast. To remedy this, we rescale the image intensities using robust min/max normalization to get a final reconstruction $\hat{\mathcal{I}}^f$:

$$\hat{\mathcal{I}}^f = \frac{\hat{\mathcal{I}} - m}{M - m}, \tag{F.4}$$

where m and M are the 1% and 99% percentiles of $\hat{\mathcal{I}}$. Finally, $\hat{\mathcal{I}}^f$ is clipped to the range $[0, 1]$.

F.4 Evaluation

In this section, we present both quantitative and qualitative results on the fidelity of our reconstructions, and compare to recent methods [10, 125, 171]. We focus our evaluation on real event data. An evaluation on synthetic data can be found in supplementary material.

We use event sequences from the Event Camera Dataset [124]. These sequences were recorded using a DAVIS240C sensor [18] moving in various environments. It contains events as well as ground-truth grayscale frames at a rate of 20 Hz. We remove redundant sequences (e.g. ones captured in the same scene) and those for which the frame quality is poor, leaving seven sequences in total that amount to 1,670 ground-truth frames. For each sequence, we reconstruct a video from the events with our method and each baseline. For each ground-truth frame, we query the reconstructed image with the closest timestamp (tolerance of ± 1 ms).

Each reconstruction is then compared to the corresponding ground-truth frame according to several quality metrics. We apply local histogram equalization to every ground-truth frame and reconstructed frame prior to computing the error metrics (this way the intensity values lie in the same intensity range and are thus comparable). Note that the camera speed gradually increases in each sequence, leading to significant motion blur on the ground-truth frames towards the end of the sequences; we therefore exclude these fast sections in our quantitative evaluation. We also omit the first few seconds from each sequence, which leaves enough time for the baseline methods that are based on event integration to converge. Note that this works in favor of the baselines, as our method converges almost immediately (more details in Section F.6).

We compare our approach against several state-of-the-art methods: [10] (which we denote as SOFIE for "Simultaneous Optic Flow and Intensity Estimation"), [171] (HF for "High-pass Filter"), and [125] (MR for "Manifold Regularization"), both in terms of image reconstruction quality and temporal consistency. For HF and MR, we used the code that was provided by the authors and manually tuned the parameters on the evaluated sequences to get the best results possible. For HF, we also applied a bilateral filter to the reconstructed images (with filter size $d = 5$ and $\sigma = 25$) in order to remove high-frequency noise, which improves the results of HF in all metrics. For SOFIE, we report qualitative results instead of quantitative results since we were not able to obtain satisfying reconstructions on our datasets using the code provided by the authors. We report three image quality metrics: mean squared error (MSE; lower is better), structural similarity (SSIM; higher is better) [195], and the calibrated perceptual loss (LPIPS; lower is better) [205]. In addition, we measure the temporal consistency of the reconstructed videos using the temporal loss introduced in Eq. (F.2). Note that computing the temporal loss requires optical flow maps between successive DAVIS frames, which we obtain with FlowNet2 [74].

Results and Discussion. The main quantitative results are presented in Table F.1, and are supported by qualitative results in Figs. F.5 and F.6. Additional results are available in the supplementary material. We also encourage the reader to watch the supplementary video, which conveys these results better than still images.

Our reconstruction method outperforms the state of the art by a large margin, with an average 24% increase in SSIM and a 22% decrease in LPIPS. Qualitatively, our method reconstructs small details remarkably well compared to the baselines (see the boxes in the first row of Fig. F.5, for example). Furthermore, our method does not suffer from "ghosting" or "bleeding edges" artifacts that are present in other methods (particularly visible in the third row of Fig. F.5). These artifacts result from (i) incorrectly estimated contrast thresholds and (ii) the fact that these methods can only estimate the image intensity up to some unknown initial intensity $\mathcal{I}_0$, the ghost of which can remain visible. We also compare our method to HF, MR, and SOFIE qualitatively using datasets and image reconstructions directly provided by the authors of [10], in Fig. F.6.

(a) Events (b) SOFIE (c) HF (d) MR (e) Ours

Figure F.6 – Qualitative comparison on the dataset introduced by [10]. Our method produces cleaner and more detailed results.

Dataset	MSE			SSIM			LPIPS		
	HF	MR	Ours	HF	MR	Ours	HF	MR	Ours
dynamic_6dof	0.10	**0.05**	0.14	0.39	**0.52**	0.46	0.54	0.50	**0.46**
boxes_6dof	0.08	0.10	**0.04**	0.49	0.45	**0.62**	0.50	0.53	**0.38**
poster_6dof	0.07	**0.05**	0.06	0.49	0.54	**0.62**	0.45	0.52	**0.35**
shapes_6dof	0.09	0.19	**0.04**	0.50	0.51	**0.80**	0.61	0.64	**0.47**
office_zigzag	0.09	0.09	**0.03**	0.38	0.45	**0.54**	0.54	0.50	**0.41**
slider_depth	0.06	0.07	**0.05**	0.50	0.50	**0.58**	0.50	0.55	**0.44**
calibration	0.09	0.07	**0.02**	0.48	0.54	**0.70**	0.48	0.47	**0.36**
Mean	0.08	0.09	**0.05**	0.46	0.50	**0.62**	0.52	0.53	**0.41**

Table F.1 – Comparison to state-of-the-art image reconstruction methods on the Event Camera Dataset [124]. Our approach outperforms prior methods on almost all datasets and metrics by a large margin, with an average 24% increase in structural similarity (SSIM) and a 22% decrease in perceptual distance (LPIPS) compared to the best prior methods.

Once again, our network generates higher quality reconstructions, with finer details and less noise. In Section F.5, we provide many more qualitative reconstruction results, and in particular show that our network is able to leverage the outstanding properties of events to reconstruct images in high-speed and high dynamic range scenarios.

Finally, Table F.2 shows the temporal error (lower error means higher temporal consistency) for all methods, as well as for the ground-truth sequences for reference. Note that the temporal loss is greater than zero on the ground-truth sequences because of small errors in optical flow estimation and occlusions. (It is still significantly lower than for all reconstruction methods.) Our method outperforms the competing approaches in terms of temporal consistency. We attribute this mostly to the temporal loss introduced in Section F.3.5. In Section F.6, we evaluate the effect of the temporal loss in an ablation study.

Dataset	Temporal Error			
	HF	MR	Ours	Ground truth
dynamic_6dof	3.32	1.91	**1.64**	0.53
boxes_6dof	3.37	1.79	**1.32**	0.59
poster_6dof	3.63	2.15	**1.77**	0.57
shapes_6dof	3.50	1.80	**1.48**	0.89
office_zigzag	3.18	1.58	**1.38**	0.48
slider_depth	2.14	1.62	**1.37**	0.70
calibration	2.72	1.52	**1.02**	0.62
Mean	3.12	1.77	**1.43**	0.63

Table F.2 – Comparison of the temporal error (Eq. (F.2), lower is better) between HF, MR and our method. Our video reconstructions have higher temporal consistency than the baselines.

F.5 Applications

We now present some applications of our method. We synthesize high framerate videos of fast physical phenomena (Section F.5.1), high dynamic range videos (Section F.5.2), and color video (Section F.5.3). Finally, we evaluate the effectiveness of our reconstructions as an intermediate representation that enables direct application of conventional vision algorithms to event data (Section F.5.4).

F.5.1 High Speed Video Reconstruction

Event cameras have a higher temporal resolution than conventional sensors (µs vs ms). In addition, the event stream is sparse by nature, which saves bandwidth. We now show that our method can decompress the event stream to reconstruct videos of fast motions with high framerate (thousands of frames per second).

Datasets. Since there exists no public event dataset containing fast physical phenomena, we recorded our own. We used the Samsung DVS Gen3 sensor [186], with VGA resolution. We recorded four sequences at daytime under bright sunlight (Fig. F.7). The first two feature objects (plaster garden gnome, ceramic mug) being shot with a rifle (approximate muzzle velocity: 376 m/s). The last two sequences feature two balloons (filled with water and air, respectively) being popped with a needle. In order to reconstruct the background, each sequence starts with the camera being moved slightly, after which it is kept steady. We additionally recorded the first two sequences with a high-end mobile phone camera (Huawei P20 Pro) operating at 240 FPS[2].

High Framerate Video Synthesis from Events. We used a fixed number of events

[2]While the Huawei P20 Pro can in principle record at 960 FPS, it can only do so for a very short amount of time (0.2 s), which made a synchronized recording impractical.

$N \simeq 10^4$ per window (exact values in Fig. F.8), resulting in video reconstructions at only a few hundred FPS. While it would be possible to retrain the network with smaller window sizes to address the specific case of extremely high framerate video synthesis, it is in fact possible to arbitrarily increase the output framerate without retraining. To achieve that, we run multiple reconstructions in parallel, introducing a slight temporal shift (of D events) between each. We thus obtain a set of videos with different temporal offsets, then merged by reordering the frames from the individual videos. This yields a video with an arbitrarily high framerate, reaching multiple thousand FPS (Fig. F.8). This temporal upsampling process may introduce some slight flickering, which is easily reduced using a simple filter [159].

Results and Discussion. In the supplementary video, we show the synthesized, high framerate videos and compare them with the 240 FPS reference videos. Fig. F.7 shows a few still frames from each sequence to convey the motion. At these extreme speeds, the event sensor is pushed to its limits: the events suffer from high noise and have many artefacts (e.g. readout artefacts). Nonetheless, our network performs well, revealing details that are invisible to the naked eye or a consumer camera. The output framerate (which varies with the event rate) is shown in Fig. F.8, and consistently stays in the range of thousands of FPS: at least an order of magnitude above conventional consumer cameras.

F.5.2 High Dynamic Range Reconstruction

Event cameras react to changes in log intensity [98], which endows them with much higher dynamic range than conventional cameras (140 dB versus 60 dB). The videos that our method synthesizes preserve the high dynamic range of the events. In Fig. F.9, we support this claim by showing qualitative reconstruction results in a variety of challenging HDR scenes and compare our reconstructions to the corresponding frame from a conventional camera.

Datasets. There are many publicly available event camera datasets featuring HDR scenes [124, 212, 171]. However, the reference images in these datasets were taken by the DAVIS sensor [18], the quality of which falls short of state-of-the-art consumer cameras. To support a fair and up-to-date comparison, we recorded our own sequences in HDR scenes (indoors and outdoors), using a recent event camera (Samsung DVS Gen3) and a high-end smartphone for reference (Huawei P20 Pro). The two sensors were rigidly mounted to each other for recording, and the corresponding footage was geometrically and temporally aligned manually in post processing.

Results and Discussion. A comparison of our reconstruction results from event data and the frames from the conventional camera is presented in Fig. F.9. The phone camera provides color images, which we converted to grayscale for easier visual comparison

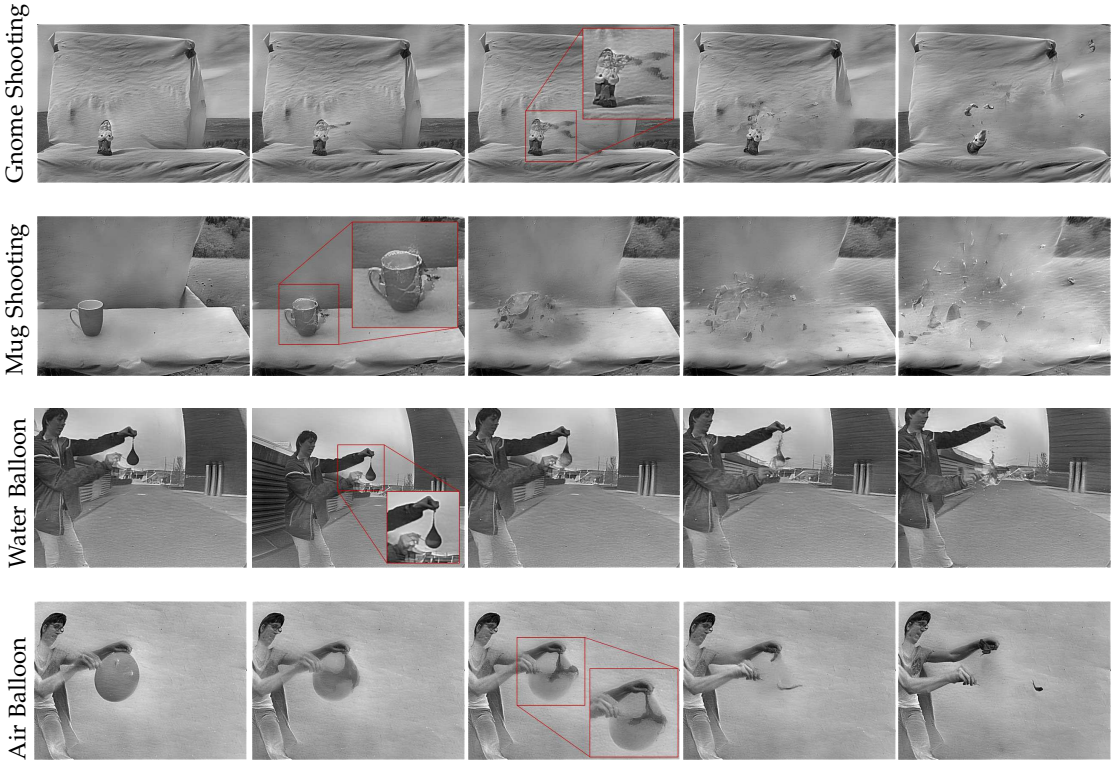
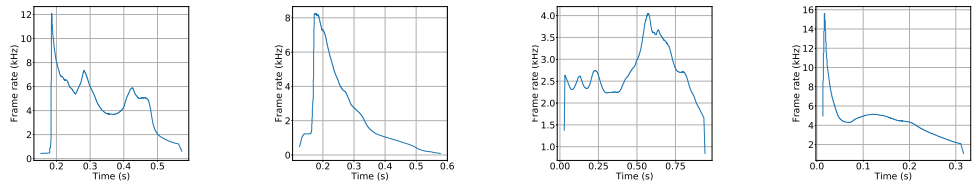


Figure F.7 – Video reconstructions of high speed physical phenomena, synthesized at $> 5{,}000$ FPS with our approach. First two rows: shooting a garden gnome and a mug with a rifle. Last two rows: popping a water balloon and an air balloon with a needle. Our reconstructions reveal details invisible to the naked eye or a conventional consumer camera. In the first two rows, the trace of the bullet is clearly visible (the bullet itself was too fast for the event sensor to catch), and cracks in both objects are visible before the pieces fly apart. In the last two rows, the membrane of the balloons contracting away from the point where the needle hit is clearly visible.



(a) Gnome $N = 10{,}000$, $D = 500$ (b) Mug $N = 12{,}000$, $D = 1000$ (c) Water balloon $N = 12{,}000$, $D = 1000$ (d) Air balloon $N = 20{,}000$, $D = 500$

Figure F.8 – Reconstruction framerate for the high-speed sequences. The output framerate grows with the event rate and value of D, and varies between 1 kHz and 15 kHz.

with our reconstructions. The first row of Fig. F.9 shows a "selfie" sequence, recorded indoors with the sensors hand-held. While the window behind the main subject appears severely overexposed in the conventional frame (b), the events (a) capture the full dynamic range, which our network successfully leverages to reconstruct the entire scene. In addition, because of the camera shaking induced by the hand-held motion, the phone frame suffers from motion blur, which is not present in our reconstruction. The second row shows a driving sequence, recorded with both sensors placed on the windshield of a car driving out of a tunnel. Once again, the area outside of the tunnel is saturated in the conventional frame, while the events capture details both indoors and outdoors, which our reconstructions recover. Finally, the third row shows an outdoor example recorded with the sensors pointing directly at the sun on a bright day. In this extreme case, the events suffer from unusually high levels of noise (flickering events), which cause reconstruction artefacts such as the dark stain around the sun. Nonetheless, our reconstruction does not suffer from glare, and reveals the circular shape of the sun (lost in the frame). The video sequences, and additional results on sequences from previously released datasets [171, 212], are available in the supplementary material.

F.5.3 Color Video Reconstruction

Until recently, event cameras were mostly monochrome, producing events based on the variations in luminance [98], and discarding color information. This has changed with the recent introduction of a sensor that can perceive "color events", the Color-DAVIS346 [116, 115]. The Color-DAVIS346 consists of an 8×6mm CMOS chip equipped with a color array filter (CFA), forming an RGBG filter pattern. The pixels in the CFA are sensitive to the variation of their specific color filter, producing events that encode color information. Color reconstruction from such events was first shown by [116], where a single color image was recovered from a large set of events using a method similar to [79]. Later, [172] adapted existing monochrome video reconstruction methods [125, 153] to color by reconstructing the individual color channels independently, resulting in videos that have a quarter of the resolution of the sensor.

We now describe a simple method to perform color reconstruction from color event data at full resolution with our network, and then present qualitative results. Following [172], we reconstruct the four color channels independently at quarter resolution (Fig. F.10(a)), upsample with bicubic interpolation, and recombine them into a low-quality color image (Fig. F.10(b)). We then combine the latter with a full-resolution grayscale image obtained by running our network on all the events (ignoring the CFA). To do this, we project the (upsampled) color image in the LAB colorspace, and replace the luminance channel with the high-quality grayscale reconstruction (Fig. F.10(d)). This exploits the human visual system's lower acuity to color differences than to luminance, a widely known phenomenon commonly used to compress videos (chroma subsampling [141]).

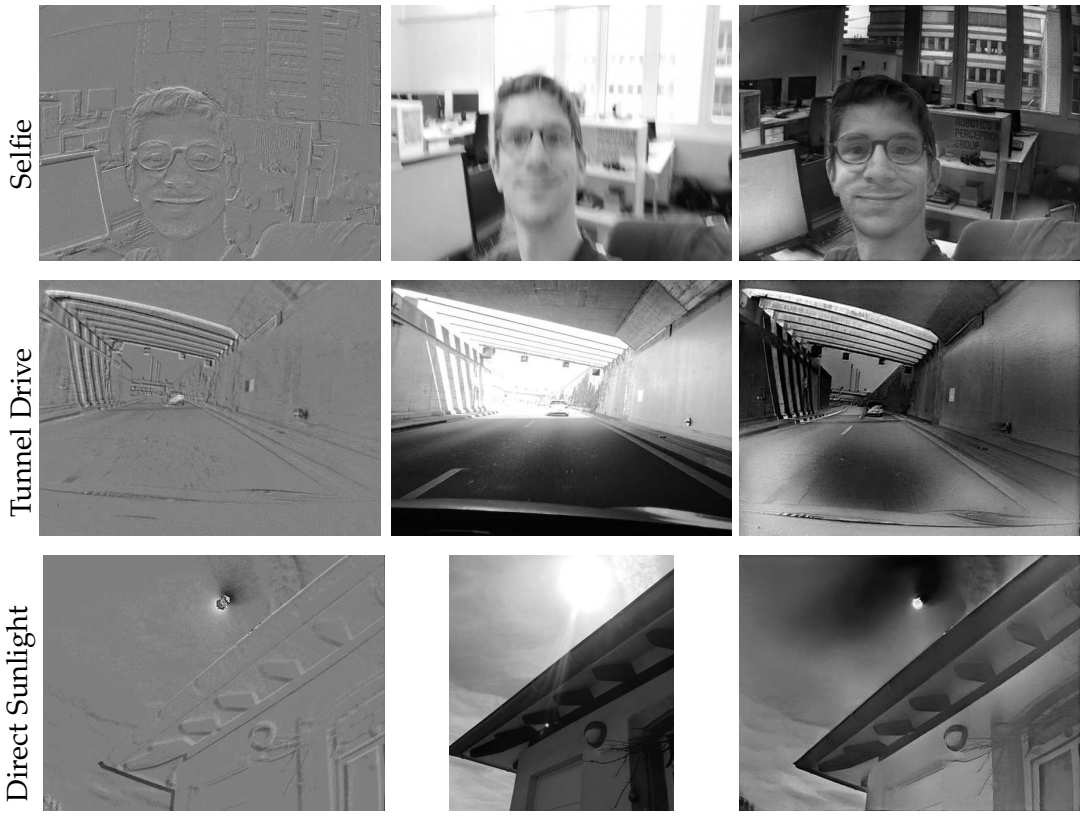
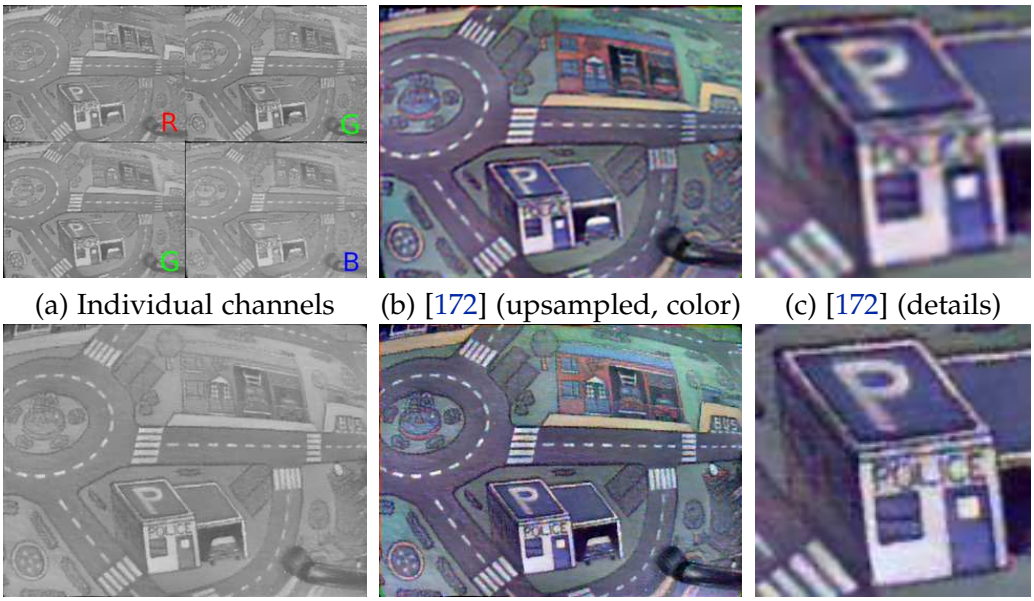


Figure F.9 – Video reconstruction under challenging lighting. First row: hand-held, indoor "selfie" sequence. Second row: driving sequence recorded while driving out of a tunnel. Third row: outdoor sequence recorded with the sensors pointing directly at the sun on a bright day. The frames from the consumer camera (Huawei P20 Pro) (b) suffer from under- or over-exposure, while the events (a) capture the whole dynamic range of the scene, which our method successfully recovers (c).



(a) Individual channels (b) [172] (upsampled, color) (c) [172] (details)

(d) Grayscale (full res) (e) Ours (full res, color) (f) Ours (details)

Figure F.10 – Our color reconstruction approach. Color channels are reconstructed independently at quarter resolution (a), then upsampled and recombined into a low-quality color image (b,c). The latter is combined with a high-quality grayscale image (d) reconstructed using all the events (ignoring the CFA). The resulting color image (e,f) preserves fine details that are lost in the quarter resolution reconstruction [172] (compare (c) and (f)).

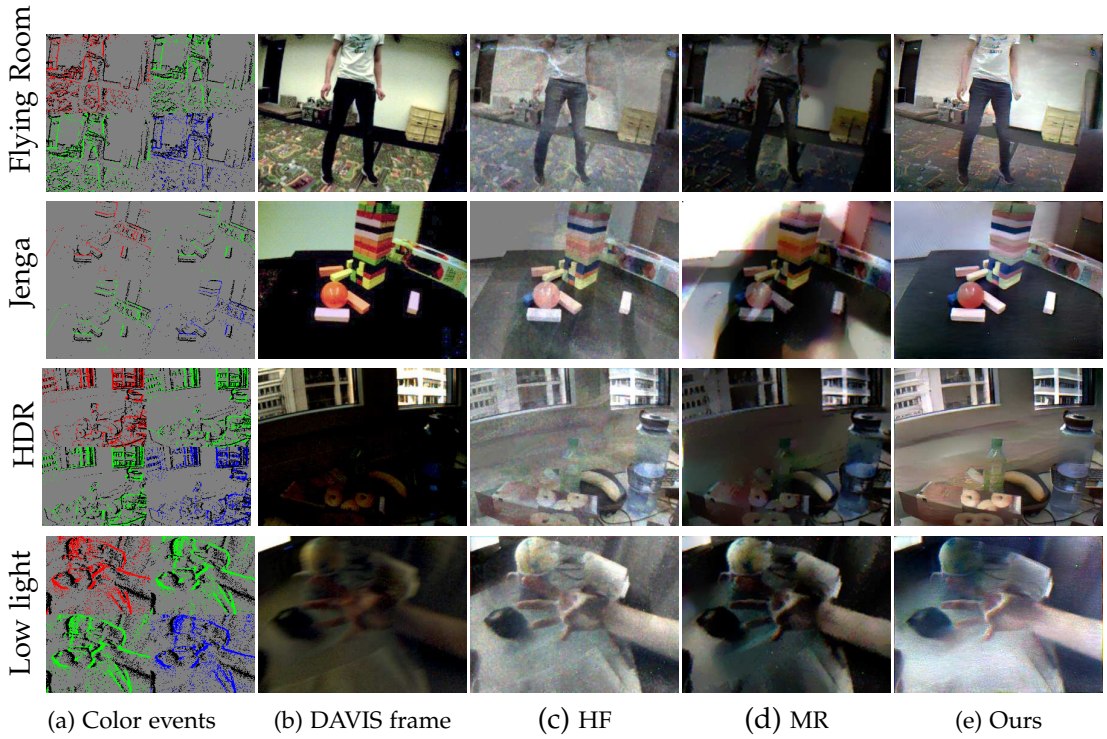


Figure F.11 – Color reconstruction results from color events (a) using datasets from [172], comparing a conventional frame (b) with multiple reconstructions from events only: HF (c), MR (d), and ours (d). For visualization, the color events (a) are split into each color channel. Positive (ON) events are colored by the corresponding filter color, and negative (OFF) events are black.

Fig. F.11 shows color reconstruction results from our method, compared to HF and MR. Unlike MR and our method, HF preserves the Bayer pattern, thus color images can simply be obtained by applying a demosaicing algorithm to raw image reconstructions. For MR, we applied the same technique as our method to obtain color reconstructions. Qualitatively, the results are consistent with those obtained for grayscale reconstructions (Fig. F.5): our method produces cleaner reconstructions than HF (less noise and "bleeding edges" artifacts), and richer reconstructions than MR which tends to smooth out details. The last two rows of Fig. F.11 show two scenarios with challenging lighting conditions: an HDR scene and a low-light scene.

F.5.4 Downstream Applications

We now investigate the possibility of using our reconstructions as an intermediate representation that facilitates direct application of conventional computer vision algorithms to event data.

Representations for Event Data. Because the output of an event camera is an asynchronous stream of events (a representation that is fundamentally different from images), existing computer vision techniques cannot be directly applied to events. To address this problem, many algorithms have been specifically tailored to leverage event data for a wide range of applications (a good survey is provided by [57]). Despite strong differences between these methods, we argue that they follow the same paradigm, relying on two ingredients: (i) a mechanism to build an internal representation of past event data, and (ii) an inference mechanism to decode new events given the current internal representation. For example, [88, 184] tackle the task of object classification from event data as follows. As internal representation, they use a *time surface*: essentially an image recording the timestamp of the last event fired at each pixel (with some temporal decay to increase the influence of recent events), which is updated with every event. They train a supervised classifier (linear SVM) to predict an object class from this surface. Finally, object prediction (the inference mechanism) consists in evaluating the classifier, which can either be done with every new event (if fast enough), or at regular intervals [184].

The image reconstructions from our method can also be viewed as a representation for event data. Similarly to other representations, our network uses, at any given time, all past events to produce an image (in other words, to update the representation). Unlike other representations, however, our image reconstructions live in the space of natural images. As such, they are *transferrable*: any computer vision algorithm operating on regular images can be used as the inference mechanism, enabling the application of pre-existing vision algorithms to event data. We acknowledge, however, that reconstructing an image with our network is slow compared to other methods that use simpler and more efficient representations that may be tailored to a task (*e.g.* time surfaces for

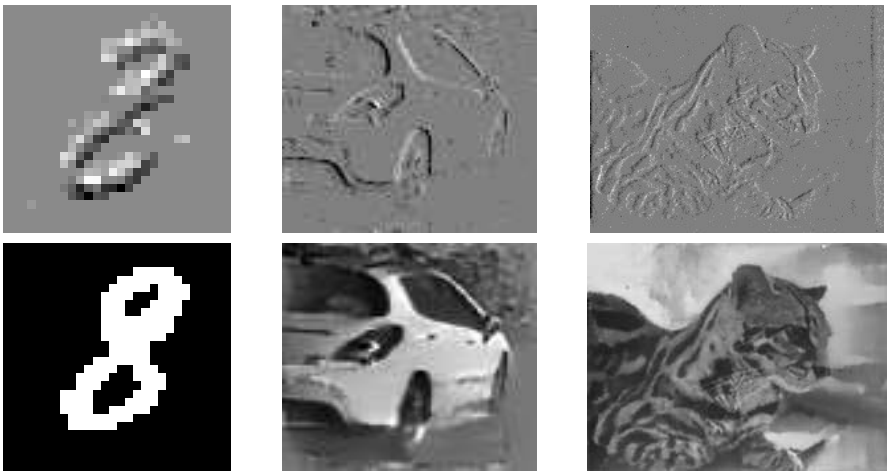
optic flow estimation [13]). Nevertheless, for the remainder of this section, we will show the effectiveness of our reconstructions as event representations on two different downstream tasks: object classification from events and camera pose estimation with events and inertial measurements. In both cases, we achieve state-of-the-art accuracy.

Object Classification. Pattern recognition from event data is an active research topic. While one line of work focuses on spiking neural architectures (SNNs) to recognize patterns from a stream of events with minimal latency (H-FIRST [133]), conventional machine learning techniques combined with novel event representations such as time surfaces (HOTS [88]) have shown the most promising results so far. Recently, HATS [184] addressed the problem of object classification from a stream of events. They proposed several modifications to HOTS, and achieved major improvements in classification accuracy, outperforming all prior approaches by a large margin.

We propose an alternative approach to object classification based on a stream of events. Instead of using a hand-crafted event representation, we directly apply a classification network (trained on image data) to images reconstructed from events. We compare our approach against several recent methods: HOTS, and the state-of-the-art HATS, using the datasets and metric (classification accuracy) used in the HATS paper. The N-MNIST (Neuromorphic-MNIST) and N-Caltech101 datasets [132] are event-based versions of the MNIST [90] and Caltech101 [50] datasets. To convert the images to event sequences, an event camera was placed on a motor, and automatically moved while pointing at images from MNIST (respectively Caltech101) that were projected onto a white wall. The N-CARS dataset [184] proposes a binary classification task: deciding whether a car is visible or not using a 100 ms sequence of events. Fig. F.12 shows a sample event sequence from each of the three datasets.

Our approach follows the same methodology for each dataset. First, for each event sequence in the training set, we use our network to reconstruct an image from the events (Fig. F.12, bottom row). We then train an off-the-shelf CNN for object classification using the reconstructed images from the training set. For N-MNIST, we use a simple CNN (details in supplementary material) and train it from scratch. For N-Caltech101 and N-CARS, we use ResNet-18 [71], initialized with weights pretrained on ImageNet [167], and fine-tune the network for the dataset at hand. Once trained, we evaluate each network on the test set (images reconstructed from the events in the test set) and report the classification accuracy. Furthermore, we perform a transfer learning experiment for the N-MNIST and N-Caltech101 datasets (for which corresponding images are available for every event sequence): we train the CNN on the conventional image datasets, and evaluate the network directly on images reconstructed from events without fine-tuning.

For the baselines, we directly report the accuracy provided in [184]. To make the comparison with HATS as fair as possible, we also provide results of classifying HATS features with a ResNet-18 network (instead of the linear SVM that was used



(a) N-MNIST (b) N-CARS (c) N-Caltech101

Figure F.12 – Samples from each dataset used in the evaluation of our object classification approach based on events (Section F.5.4). Top: preview of the event sequence. Bottom: our image reconstruction.

	N-MNIST	N-CARS	N-Caltech101
HOTS	0.808	0.624	0.210
HATS/linear SVM	**0.991**	0.902	0.642
HATS/ResNet-18	n.a.	0.904	0.700
Ours (transfer learning)	0.807	n.a.	0.821
Ours (fine-tuned)	0.983	**0.910**	**0.866**

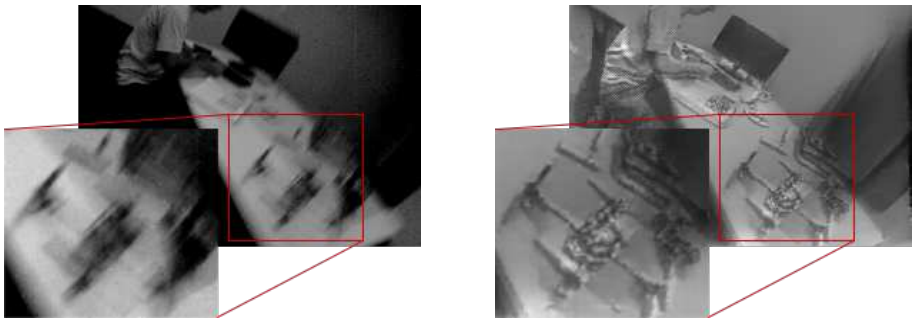
Table F.3 – Classification accuracy compared to recent approaches, including HATS [184], the state-of-the-art.

originally). The results are presented in Table F.3, where the datasets are presented in increasing order of difficulty from left to right. Despite the simplicity of our approach, it outperforms all baselines. The gap between our method and the state-of-the-art increases with the difficulty of the datasets. While we perform slightly worse than HATS on N-MNIST (98.3% versus 99.1%), this can be attributed to the synthetic nature of N-MNIST, for which our approach does not bring substantial advantages compared to a hand-crafted feature representation such as HATS. Note that, in contrast to HATS, we did not perform any hyperparameter tuning. On N-CARS (binary classification task with natural event data), our method slightly outperforms the baseline (91% versus 90.4% for HATS). However, N-CARS is almost saturated in terms of accuracy.

On N-Caltech101 (the most challenging dataset, requiring classification of natural event data into 101 object classes), our method outperforms HATS by a large margin (86.6% versus 70.0%). This significant gap can be explained by the fact that our approach leverages decades of computer vision research and datasets. Lifting the event stream into the image domain with our events-to-video approach allows us to use a mature CNN architecture that was pretrained on existing labeled datasets. We can thus use powerful hierarchical features learned on a large body of image data – something that is not possible with event data, for which labeled datasets are scarce. Strikingly, our approach, in a pure transfer learning setting (i.e. feeding images reconstructed from events to a network trained on real image data) performs better than all other methods, while not using the event sequences from the training set. To the best of our knowledge, this is the first time that direct transfer learning between image data and event data has been achieved.

While the proposed approach reaches state-of-the-art accuracy, alternative approaches such as HATS are computationally more efficient, mostly because updating the internal representation (time surface) requires less operations than generating a new image with our neural network. Nonetheless, our approach is real-time capable. On N-Caltech101, end-to-end classification takes less than 10 ms (sequence reconstruction: $\leq$ 8 ms, object classification: $\leq$ 2 ms) on an NVIDIA RTX 2080 Ti GPU. More details can be found in Section F.6.

Visual-Inertial Odometry. The task of visual-inertial odometry (VIO) is to recover the 6-degrees-of-freedom (6-DOF) pose of a camera from a set of visual measurements (images or events) and inertial measurements from an inertial measurement unit (IMU) that is rigidly attached to the camera. Because of its importance in augmented/virtual reality and mobile robotics, VIO has been extensively studied in the last decade and is relatively mature today [119, 94, 16, 52, 143]. Yet systems based on conventional cameras fail in challenging conditions such as high-speed motions and high-dynamic-range environments. This has recently motivated the development of VIO systems with event data (EVIO) [211, 149, 164].



(a) DAVIS frame (b) Our reconstruction

Figure F.13 – Comparison of DAVIS frames and reconstructed frames on a high-speed portion of the 'dynamic_6dof' sequence. Our reconstructions from events do not suffer from motion blur, which leads to increased pose estimation accuracy (Table F.4).

The state-of-the-art EVIO system, UltimateSLAM [164], operates by independently tracking visual features from pseudo-images reconstructed from events using motion compensation [149] (i.e. the internal representation) plus optional images from a conventional camera, and fusing the tracks with inertial measurements using an existing optimization backend [94].

Here, we go one step further and directly apply an off-the-shelf VIO system (specifically, VINS-Mono [143], which is state-of-the-art [42]) to videos reconstructed from events using either our approach, MR, or HF, and evaluate against UltimateSLAM. As is standard [211, 149, 164], we use sequences from the Event Camera Dataset [124], which contain events, frames, and IMU measurements from a DAVIS240C [18] sensor. Each sequence is 60 seconds long, and contains data from a hand-held event camera undergoing a variety of motions in several environments. All sequences feature extremely fast motions (angular velocity up to 880 °/s and linear velocity up to 3.5 m/s), which leads to severe motion blur on the frames (Fig. F.13). We compare our approach against the two operating modes of UltimateSLAM: UltimateSLAM (E+I) which uses only events and IMU, and UltimateSLAM (E+F+I) that uses the events, the IMU, and additional frames. We run a publicly available VIO evaluation toolbox [208] on raw trajectories provided by the authors of UltimateSLAM, which ensures that the trajectories estimated by all methods are evaluated in the exact same manner. For completeness, we also report results from running VINS-Mono directly on the frames from the DAVIS sensor.

Table F.4 presents the mean translation error of each method, for all datasets (additional results are presented in the supplement). First, we note that our method performs better than UltimateSLAM (E+I) on all sequences, with the exception of the 'shapes_6dof' sequence. This sequence features a few synthetic shapes with very few features (≤ 10), which cause VINS-Mono to not properly initialize, leading to high error (note that this is a problem with VINS-Mono and not our image reconstructions). Overall, the

Inputs	Ours E+I	U.SLAM E+I	U.SLAM E+F+I	HF E+I	MR E+I	VINS-Mono F+I
shapes_translation	0.18	0.32	**0.17**	failed	2.00	0.93
poster_translation	**0.05**	0.09	0.06	0.49	0.15	failed
boxes_translation	**0.15**	0.81	0.26	0.70	0.45	0.22
dynamic_translation	**0.08**	0.23	0.09	0.58	0.17	0.13
shapes_6dof	1.09	0.09	**0.06**	failed	3.00	1.99
poster_6dof	**0.12**	0.20	0.22	0.45	0.17	1.99
boxes_6dof	0.62	0.41	**0.34**	1.71	1.17	0.94
dynamic_6dof	0.15	0.27	**0.11**	failed	0.55	0.76
hdr_boxes	0.34	0.44	0.37	0.64	0.66	**0.32**
Mean	0.31	0.32	**0.19**	0.76	0.92	0.91
Median	**0.15**	0.27	0.17	0.61	0.55	0.84

Table F.4 – Mean translation error (in meters) on the sequences from [124]. Our method outperforms all other methods that use events and IMU, including UltimateSLAM (E+I). Surprisingly, it even performs on par with UltimateSLAM (E+F+I), while not using additional frames. Methods for which the mean translation error exceeds 5 m are marked as "failed".

median error of our method is 0.15 m, which is almost half the error of UltimateSLAM (E+I) (0.27 m) which uses the exact same data. Indeed, while UltimateSLAM (E+I) uses coarse pseudo-images created from a single, small window of events, our network is able to reconstruct images with finer details and higher temporal consistency – both of which lead to better feature tracks and thus better pose estimates. Even more strikingly, our approach performs on par with UltimateSLAM (E+F+I), while the latter requires additional frames which we do not need. The median error of both methods is comparable (0.15 m for ours versus 0.17 m for UltimateSLAM (E+F+I)).

Finally, we point out that running the same VIO (VINS-Mono) on competing image reconstructions (MR and HF) yields significantly larger tracking errors (e.g. median error three times larger for MR), which further highlights the superiority of our image reconstructions for downstream vision applications. We acknowledge that our approach is not as fast as UltimateSLAM. Since the main difference between both approaches is how they build the internal event representation, a rough estimate of the performance gap can be obtained by comparing the time it takes for each method to synthesize a new image. UltimateSLAM takes about 1 ms on a CPU whereas our method takes ≤ 4 ms on a high-end GPU. Nevertheless, our events-to-video network allows harnessing the outstanding properties of events for VIO and exceeds the accuracy of state-of-the-art EVIO algorithms that were designed specifically for event data.

F.6 Analysis

In this section, we provide a thorough analysis of our network. First, we measure the computational efficiency of our approach and compare it against several baselines. Second, we perform some ablation studies to justify the choice of some components. Finally, we analyze some of the interesting properties of our network and contrast these with prior work.

F.6.1 Computational Efficiency

Here we analyze the performance (i.e. computational efficiency) of the network. We also justify the specific hyperparameters of our architecture as the result of a simple search over these parameters.

We compare the performance of our method against HF [171] and MR [125]. Due to fundamental differences in the way each of these methods processes event data, it is difficult to provide a direct and fair performance comparison. HF processes the event stream in an event-by-event fashion, providing (in theory) a new image reconstruction with every incoming event. However, the raw image reconstructions from HF need to be filtered (for example, using a bilateral filter) to obtain results with reasonable quality. While MR can in principle also operate in an event-by-event fashion, its best quality results are obtained when it processes batches of events. Our method also operates on batches of events. In Table F.5, we report the mean "frame synthesis time", which we define as the time it takes to process $N = 10{,}000$ events (this number was chosen because it yields good-quality images for all three methods). We ran our method and MR on an NVIDIA GeForce RTX 2080 Ti GPU, and HF on an Intel Core i9-9900K @ 3.60 GHz CPU. Finally, we also report the number of floating point operations (FLOPs) at different resolutions in Table F.6, which is also related to power consumption.

	Frame synthesis time (ms)
HF	0.70 / 1.45*
MR	0.84
Ours	5.53

Table F.5 – Frame synthesis time (defined as the time it takes to process a batch of $N = 10{,}000$ events here) for our method, HF, and MR. HF works best when some filtering (e.g. a bilateral filter) is applied (*), which increases the per-frame computation time.

Discussion. Our method is not as fast as HF or MR, which can process event data roughly 5 times faster. While high performance is not the main focus of the present work, our network can nonetheless easily run in real-time, providing state-of-the-art reconstructions in terms of video quality. We believe there is ample room for performance improvements. First, the performance of our approach may improve

	FLOPs
240×180	21.2 G
346×260	44.5 G
640×480	147.2 G
1280×720	441.7 G
Per pixel	486.0 k

Table F.6 – Computational cost. We report the number of FLOPs for a single forward-pass at common sensor resolutions. The last row provides the FLOPs normalized by the sensor resolution, *i.e.* the number of floating point operations per pixel.

significantly when implemented on hardware specifically optimized to perform fast and efficient inference in neural networks. Second, exploiting the sparsity of the event tensors (most values of which are zeros) could additionally improve the computational efficiency by a large margin. One promising direction in that regard would be to use sparse convolutions [68] or hardware accelerators designed to efficiently process sparse inputs [4]. Finally, we believe one of the most alluring characteristics of our method is its ability to summarize a large number of events into one high-quality image. Since the network is robust to the number of events in each window (Section F.6), it can be used with large windows of events when online operation is required (for example, for generating a live video preview), and run again offline with smaller windows to generate a high framerate video a posterori (as shown for example in Section F.5.1). Alternatively, our network could also be used to generate high-quality images at low framerates (i.e. using large batches of events), which could be fused with event data using a complementary filter [171] to synthesize high-quality videos at very high framerate more efficiently. It should also be noted that the usage of a high-end GPU (*e.g.* GTX 2080Ti) for fast image reconstruction comes at a price. First, the cost of the GPU itself (> 1,000$), which reflects the huge silicon area and high speed stacked DRAM memory used by this GPU. Second, high-end GPUs incur a high power consumption: about 250W of power for the GTX2080Ti, to which the additional cost of the host PC (about 200W) should be added.

Searching for a Lightweight Architecture. In this section, we show that our choice of network architecture parameters provides a good trade-off between quality and performance. We performed a simple architecture search over important hyperparameters:

- number of encoders N_E in the range $\{2, 3, 4\}$,
- number of residual blocks N_R (in $\{0, 1, 2\}$),
- type of skip connection $\oplus$ (concatenation or sum),
- number of feature channels N_b ($\{8, 16, 32, 64\}$).

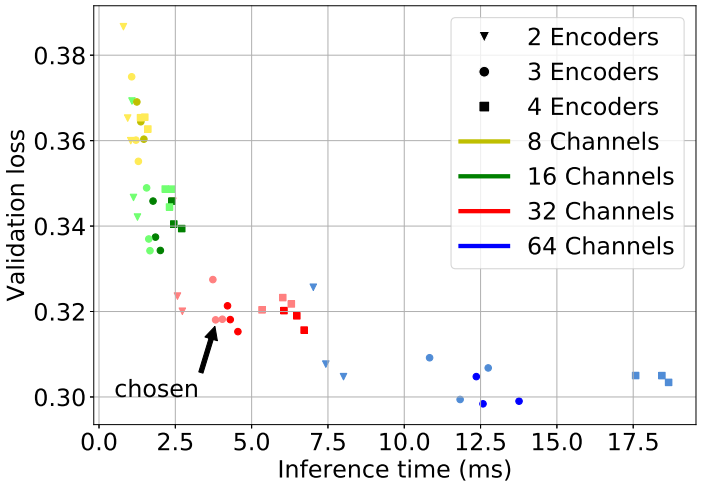


Figure F.14 – Identifying a good trade-off between accuracy and efficiency. Each point in this graph corresponds to a different architecture variant. The color intensity indicates the type of skip connection: light colors correspond to element-wise sum and normal colors to concatenation. The distribution forms an "elbow" that suggests a good trade-off between inference time and reconstruction quality.

We then trained a simplified version of the network (with the recurrent connection disabled) for each parameter combination (72 combinations in total) on a small subset of the full dataset to convergence (10 epochs), monitoring the validation loss as well as the mean inference time per event tensor (evaluated on the full network). The results are presented in Fig. F.14. Notably, the distribution of the architectures form an "elbow", which indicates that there exists a good trade-off between model complexity (i.e. inference time) and reconstruction quality (validation loss). Based on these results, we choose $N_E = 3$, $N_R = 2$, $N_b = 32$, and element-wise sum for the skip connection.

F.6.2 Ablation Studies

We now present some ablation studies to highlight the impact of some of the key features of our architecture.

Temporal Loss. To measure the impact of the temporal loss (Section F.3.5, Eq. (F.2)), we trained the network without it (using $\lambda_{TC} = 0$). Table F.7 compares the resulting network with the full network, in terms of temporal consistency and image reconstruction quality. We use the same sequences as in our quantitative evaluation (Section F.4), and report the mean values over all datasets, for each metric. Unsurprisingly, the network that was trained with the temporal loss achieves better temporal consistency (31% decrease of the temporal error on average). More interestingly, the full network

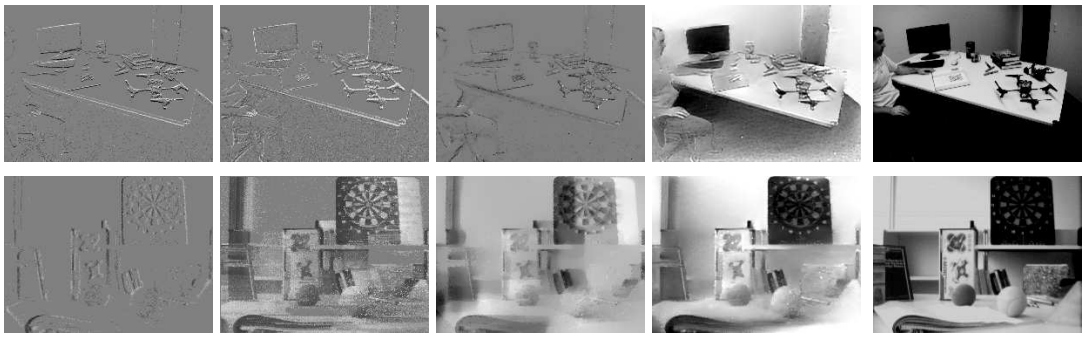
	MSE	SSIM	LPIPS	Temporal Error
w/o temporal loss	0.07	0.59	0.43	2.08
w/ temporal loss	**0.05**	**0.62**	**0.41**	**1.43**

Table F.7 – Ablation study: effect of the temporal loss. The temporal loss improves the temporal consistency as well as the image quality.

also performs better in terms of image quality overall (average improvement of about 5%), suggesting that the temporal loss also acts as a regularizer, driving the optimizer to converge to a better local optimum.

Recurrent Connection. Table F.8 compares the image quality and temporal consistency when the recurrent connection of our network is removed. It highlights the role that the recurrent connection plays in achieving good video reconstruction quality. The recurrent connection increases temporal video consistency by a large margin (65% decrease in temporal error), removing the high-frequency blinking present in the reconstructions produced without the recurrent connection (see supplementary video). The recurrent connection also increases the image quality (15%), suggesting that the network can effectively leverage its short-term memory to reconstruct accurate images.

ConvLSTM vs. vanilla RNN. We now compare our network (based on stacked ConvLSTMs [181]) to the preliminary version of this work [153] based on a vanilla RNN, focusing on the generalization ability of both networks to the duration of the event windows used (or, equivalently, the number of events in each window). We train our network and [153] with the same training data, and evaluate both networks at two different inference rates, feeding non-overlapping event windows of a fixed duration τ. In the first experiment, we use $\tau = 50\,\text{ms}$, which is close to the average duration of each event tensor in the training data. In the second experiment, we use $\tau = 5\,\text{ms}$ to assess the generalization ability of both networks to a window size that is significantly different from the training data. Note that the second experiment is harder since the networks see a much smaller number of events at each time step (i.e. incomplete information), and thus must rely more strongly on their internal memory to recall the missing data necessary to produce good quality reconstructions. The results are reported in Table F.9. When the window length is close to the training conditions ($\tau = 50\,\text{ms}$), the networks perform similarly. However, with $\tau = 5\,\text{ms}$, the image quality drops drastically for the vanilla RNN [153], while degrading only slightly with our network (6% decrease in SSIM). Our network is thus more robust to varying window sizes, maintaining intensity forward in time in a more stable fashion.



(a) Events (b) HF (c) MR (d) Ours (e) Ground truth

Figure F.15 – Analysis of the initialization phase (reconstruction from few events). This figure shows image reconstructions from each method, 0.5 seconds after the sensor was started. HF [171] and MR [125], which are based on event integration, cannot recover the intensity correctly, resulting in "edge" images (first and second row) or severe "ghosting" effects (third row, where the trace of the dartboard is clearly visible). In contrast, our network successfully reconstructs most of the scene accurately, even with a low number of events.

(a) Initial time (b) + 1 iteration (c) + 3 iterations (d) + 10 iterations (e) + 20 iterations

Figure F.16 – In this experiment, the events are artificially stopped at some time t, i.e. the network is fed empty event tensors in all subsequent iterations. The correct thing to do would be to simply copy the first image to all subsequent predictions. The network has instead learned to gradually decay the image.

	MSE	SSIM	LPIPS	Temporal Error
w/o recurrent	0.08	0.54	0.47	4.00
w/ recurrent	**0.05**	**0.62**	**0.41**	**1.43**

Table F.8 – Ablation study: effect of the recurrent connection. The quality and temporal consistency improve when using the recurrent connection, validating that our network is able to successfully propagate information through time.

	MSE	SSIM	LPIPS
RNN ($\tau = 50$ ms)	0.05	0.61	**0.40**
RNN ($\tau = 5$ ms)	0.14	0.27	0.71
Ours ($\tau = 50$ ms)	**0.05**	**0.62**	0.41
Ours ($\tau = 5$ ms)	0.06	0.58	0.44

Table F.9 – Reconstruction quality with $\tau = 50$ ms versus $\tau = 5$ ms windows. The RNN [153] does not generalize to $\tau = 5$ ms, yielding poor quality reconstructions (high MSE and LPIPS, low SSIM). In contrast, the reconstruction quality of our network degrades only slightly ($\leq$6% decrease in SSIM), thanks to its more stable recurrent connection.

F.6.3 Edge Cases

We now present two interesting edge cases that shed light on some characteristics of our approach. We first analyze the initialization phase, when few events have been observed. We then perform a simple experiment to estimate the effective size of our network's memory, to better understand its behavior in regions with low event rates.

Initialization. By analyzing the initialization phase (i.e. when few events have been previously triggered) we gain interesting insight into how our network operates. We see significantly different behaviour when compared to prior approaches that are based on direct event integration. Fig. F.15 compares image reconstructions from our approach, HF, and MR during the initialization phase. We specifically examine the interval from 0 s to 0.5 s from the beginning of capture. HF and MR, which rely on event integration, can only recover the intensity up to the initial (unknown) image $\mathcal{I}_0$ (i.e. they can only recover $\hat{\mathcal{I}} \approx \mathcal{I} - \mathcal{I}_0$), which results in an "edge" image which does not capture the appearance of the scene correctly. In contrast, our method successfully leverages deep priors to reconstruct the scene despite the low number of events.

Network Memory. For how long can our network remember intensity information? To answer this question – in other words, to measure the effective size of the temporal receptive field of our network – we perform a simple experiment. We take a given sequence of events and artificially stop the events at a fixed time t (this is achieved in practice by zeroing out all the event tensors with timestamps $\geq t$). We feed the resulting empty event tensors to our network and present the evolution of the reconstructions as a function of the number of iterations in Fig. F.16. In the complete absence of events,

the current reconstruction should be left untouched, i.e. the network should simply copy all the pixel values forward. As shown in Fig. F.16, this is in fact what the network has learned to do in the first few iterations, although this was never hardcoded in the network design; rather, the recurrent network discovered this pattern from the training data. However, as can be observed in Fig. F.16, the network gradually decays the image intensity when forced to perform inference with tensors containing no events. Interestingly, the image decay is not isotropic: the regions with high contrast (e.g. the dart board in the first row) tend to be preserved for a higher number of iterations. While this experiment may seem artificial – such a situation cannot happen in practice since no event tensor is generated when no events fire – it sheds light on the behavior of the network in regions where very few events are fired. We believe the length of the memory is strongly tied to the distribution of optical flows in the training data. Indeed, our synthetic training datasets (aimed at general-purpose reconstruction) contain few "pauses" (i.e. regions with low event rate), thus our network does not need to retain information for long time periods. This suggests that our network could be further improved for specific applications by generating training data with a similar distribution of optical flow than the target application. In autonomous driving for example, the network may learn to retain intensity information for a long time in the center of the image (focus of expansion, low event rate), and for a shorter amount of time on the sides of the image (higher event rate).

F.7 Conclusion

We presented a novel events-to-video reconstruction framework based on a recurrent convolutional network trained on simulated event data. In addition to outperforming state-of-the-art reconstruction methods on real event data by a large margin ($>20\%$ improvement), we showed the applicability of our method to synthesize high framerate, high dynamic range, and color video reconstructions from event data only. Finally, we demonstrated the effectiveness of our reconstructions as as intermediate representation that bridges event cameras and mainstream computer vision.

Appendix: Supplementary Video & Code

As the main focus of the present work is video reconstruction, we strongly encourage the reader to view the supplementary video, which contains:

- Video reconstructions from our method on various event datasets, with a visual comparison to several state of the art methods (Section F.4).
- High framerate videos of the high speed experiments (Section F.5.1).
- High dynamic range video reconstructions (Section F.5.2).
- Color video reconstructions (Section F.5.3).
- Video illustrations of the ablation studies (Section F.6).
- Video of the VINS-Mono visual-inertial odometry algorithm [143] running on a video reconstruction from events (Section F.5.4).
- Qualitative results on two additional downstream applications that were not presented in the main paper: object detection (based on YOLOv3 [158]), and monocular depth prediction (based on MegaDepth [97]). We point out that neither of these tasks have ever been shown with event data before this work.

Code Release. To spur further research, we release the reconstruction code and a pretrained model at: http://rpg.ifi.uzh.ch/E2VID.

Appendix: Formal Network Description

An overview of the network architecture is presented in Fig. F.4. Given an event tensor $\mathbf{E}_k$ (at time step k) and the previous network state, defined as $\mathbf{s}_k = \left\{\mathbf{c}_{k-1}^1, ..., \mathbf{c}_{k-1}^{N_E}\right\}$, our network performs the following sequence of operations (omitting ReLU and batch normalization):

$$\mathbf{x}_k^h = \mathcal{H}(\mathbf{E}_k) \tag{F.5}$$

$$\mathbf{h}_k^i, \mathbf{c}_k^i = \mathcal{E}^i(\mathbf{h}_k^{i-1}, \mathbf{c}_{k-1}^i) \tag{F.6}$$

$$\mathbf{r}_k^j = \mathcal{R}^j(\mathbf{r}_k^{j-1}) \tag{F.7}$$

$$\mathbf{d}_k^l = \mathcal{D}^l(\mathbf{d}_k^{l-1} \oplus \mathbf{h}_k^{N_E-l+1}) \tag{F.8}$$

$$\hat{\mathcal{I}}_k = \sigma\left(\mathcal{P}(\mathbf{d}_k^{N_E} \oplus \mathbf{x}_k^h)\right) \tag{F.9}$$

where $1 \leq i \leq N_E$, $1 \leq j \leq N_R$ and $1 \leq l \leq N_E$, $\mathbf{h}_k^0 = \mathbf{x}_k^h$, $r_k^0 = \mathbf{h}_k^{N_E}$ and $\mathbf{d}_k^0 = \mathbf{r}_k^{N_R}$. At the first iteration ($k = 0$), the hidden states for each encoder layer are initialized to zero, *i.e.* $\mathbf{c}_0^i = 0$, for $1 \leq i \leq N_E$. The $\oplus$ operator denotes the skip connection function (element-wise sum), and σ the sigmoid function.

Appendix: Why Use Synthetic Training Data?

Here, we expand on the reasons that motivated us to train our reconstruction network using synthetic event data. First, simulation allows to capture a large variety of scenes and motions at very little cost. Second, a conventional camera (even a high quality one) would provide poor ground truth in high-speed conditions (motion blur) and HDR scenes, which are the conditions in which event sensors excel; by contrast, synthetic data does not suffer from these issues. Last but not least, simulation allows to randomize the contrast thresholds of the event sensor, which increases the ability of the network to generalize to different sensor configurations (contrast sensitivity). To illustrate this last point, we show in Fig. F.17 (left) what happens when training the network on real event data from an event camera (specifically, the sequences from the Event Camera Dataset [124] already presented in the main paper, which were recorded with a DAVIS240C sensor), and evaluating the trained network on data coming from a different event sensor (specifically, the 'outdoors_day1' sequence from the MVSEC dataset [212], which was recorded with a mDAVIS346 sensor): the reconstruction suffers from many artefacts. This can be explained by the fact that the events from the mDAVIS346 sensor have statistics that are quite different from the training events (DAVIS240C): the set of contrast thresholds are likely quite different between both sensors, and the illumination conditions are also different (outdoor lighting for the MVSEC dataset versus indoor lighting for the training event data). By contrast, the network trained on simulated event data (Fig. F.17, right) generalizes well to the event data from the mDAVIS346, producing a visually pleasing image reconstruction.

Appendix: Additional Results

Results on Synthetic Event Data

We show a quantitative comparison of the reconstruction quality of our method as well as MR and HF on synthetic event sequences in Table F.10. We present qualitative reconstruction results on this dataset in Fig. F.18. All methods perform better on synthetic data than real data. This is expected because simulated events are free of noise. Nonetheless, the performance gap between our method and the state of the art is preserved, and even slightly increases (24% improvement in SSIM, 56% decrease in LPIPS). We note that perfect reconstruction, even on noise-free event streams is not possible, since image reconstruction from events is only possibly up to the quantization

Figure F.17 – Reconstruction from (i) a network trained only on real event data from the DAVIS240C sensor (left), and (ii) a network trained only on simulated event data (right). This sequence is from the MVSEC dataset, and was recorded with a mDAVIS346 sensor.

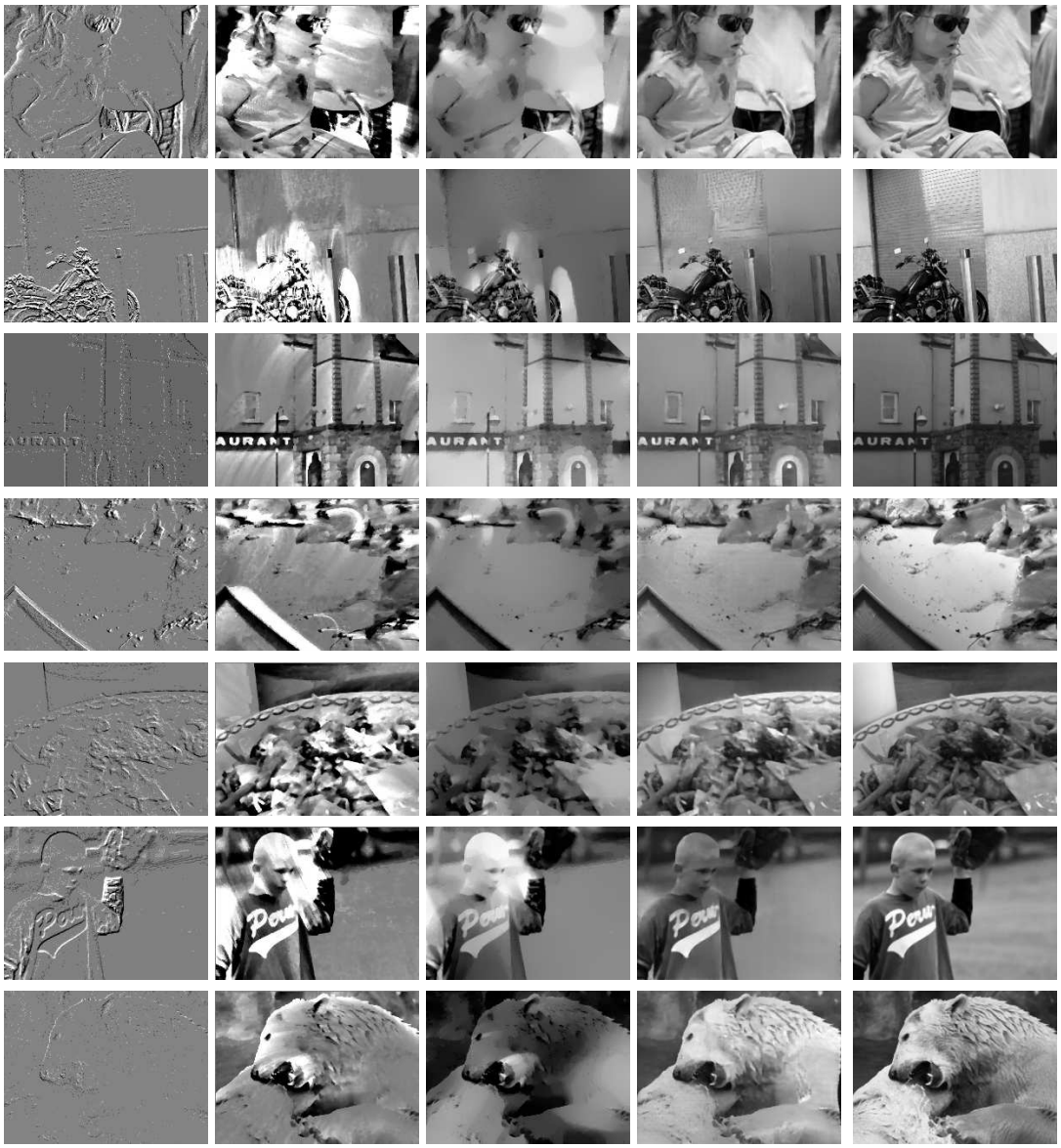
Dataset	MSE			SSIM			LPIPS		
	HF	MR	Ours	HF	MR	Ours	HF	MR	Ours
synthetic_0	0.08	0.06	**0.02**	0.54	0.61	**0.83**	0.49	0.47	**0.26**
synthetic_1	0.15	0.14	**0.06**	0.38	0.45	**0.60**	0.54	0.56	**0.37**
synthetic_2	0.07	0.08	**0.02**	0.60	0.68	**0.82**	0.42	0.42	**0.26**
synthetic_3	0.07	0.05	**0.04**	0.57	0.66	**0.75**	0.45	0.43	**0.33**
synthetic_4	0.08	0.06	**0.02**	0.62	0.67	**0.85**	0.41	0.42	**0.25**
synthetic_5	0.08	0.08	**0.02**	0.50	0.61	**0.74**	0.53	0.54	**0.36**
synthetic_6	0.07	0.04	**0.03**	0.56	0.65	**0.77**	0.44	0.48	**0.30**
Mean	0.09	0.07	**0.03**	0.54	0.62	**0.77**	0.47	0.47	**0.30**

Table F.10 – Comparison of image quality with respect to state of the art on synthetic event sequences.

limit imposed by the contrast threshold of the event camera.

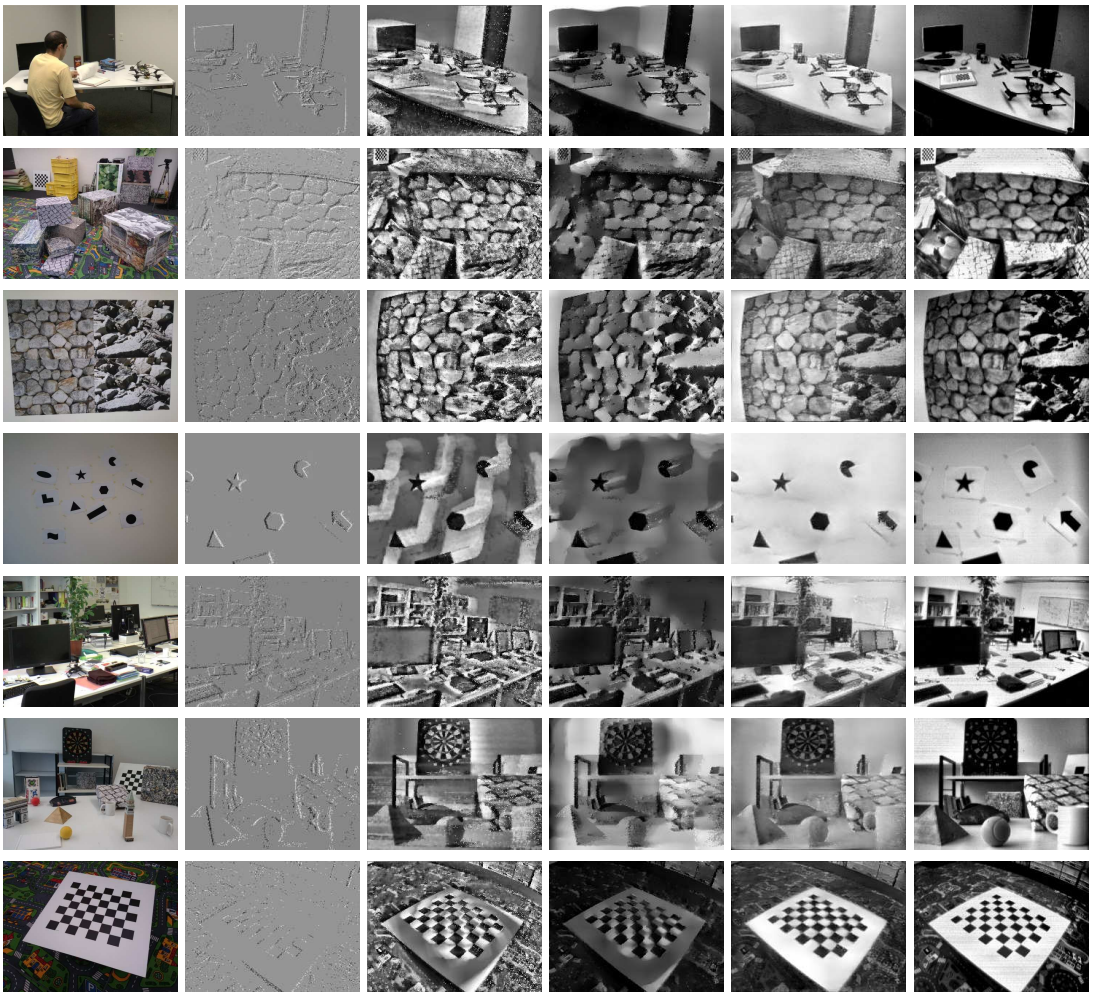
Additional Qualitative Results on Real Data

Fig. F.19 shows qualitative results on sequences from the Event Camera Dataset [124] (which we used for our quantitative evaluation). Fig F.20 shows qualitative results on the sequences introduced by Bardow *et al*. [10]. Figs. F.21, F.22 and F.23 present HDR reconstruction results on various publicly available datasets [212, 171, 172]. Further results are shown in the supplementary video which conveys these results in a better form than still images.



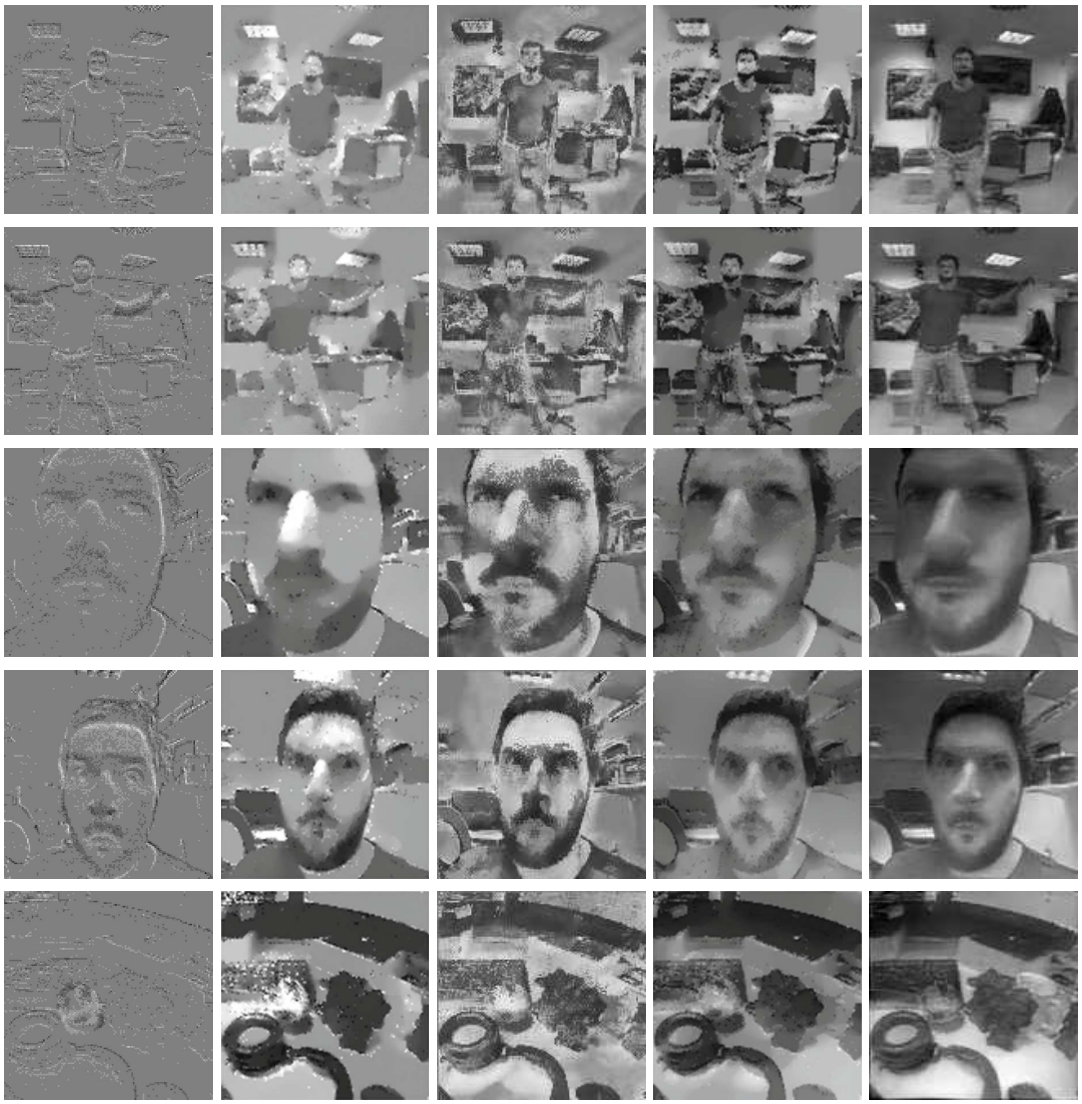
(a) Events (b) HF (c) MR (d) Ours (e) Ground truth

Figure F.18 – Qualitative comparison of our reconstruction method with HF [171] and MR [125] on synthetic sequences from the validation set. Note our method is able to reconstruct fine details such as the bear's fur (last row), which competing methods are not able to preserve.



(a) Scene (b) Events (c) HF (d) MR (e) Ours (f) Ground truth

Figure F.19 – Qualitative comparison of our reconstruction method with two recent competing approaches, MR [125] and HF [171], on sequences from [124], which contain ground truth frames from a DAVIS240C sensor. Our method successfully reconstructs fine details (textures in the second and third row) compared to other methods, while avoiding ghosting effects (particulary visible in the shapes sequences on the fourth row).

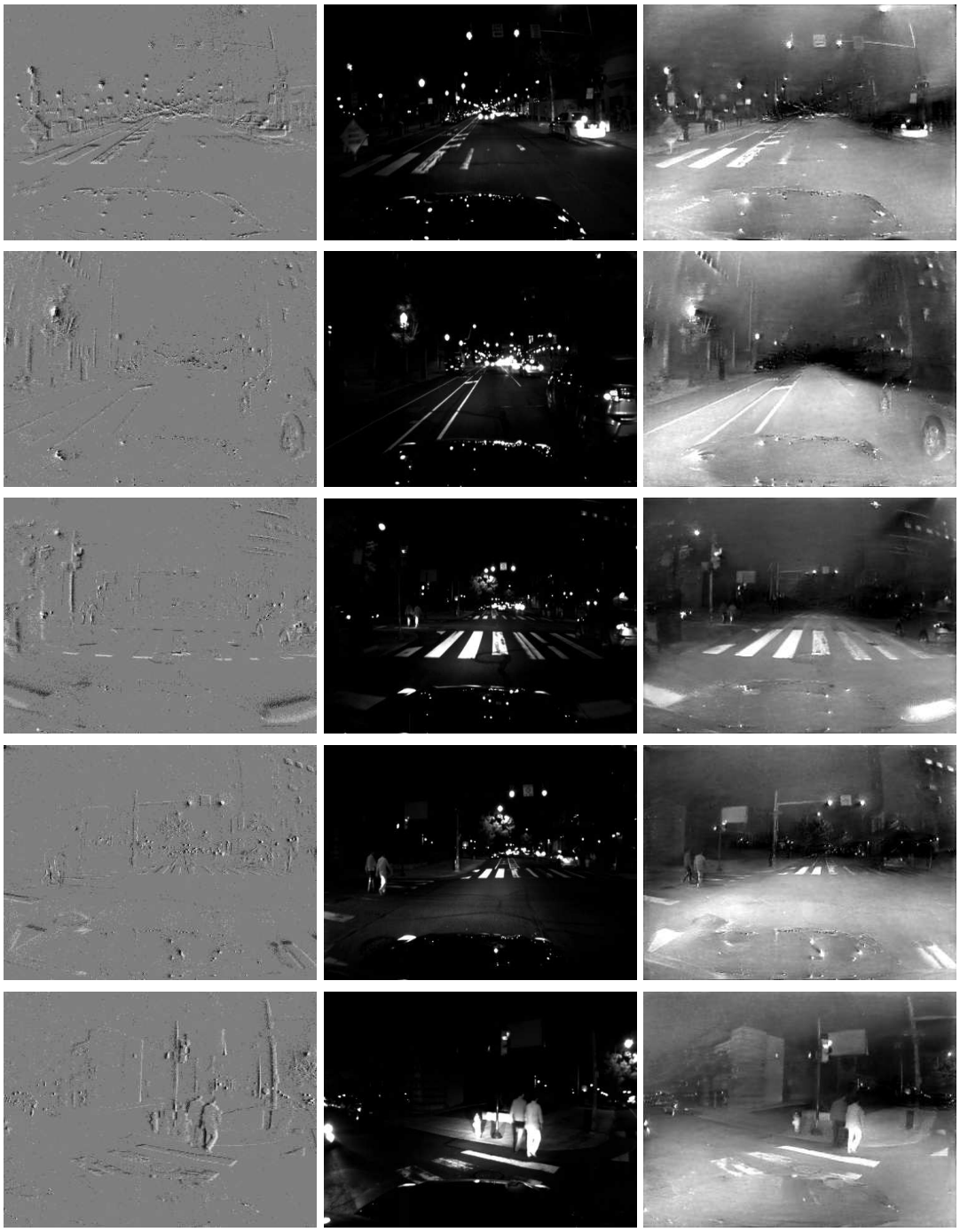


(a) Events (b) SOFIE [10] (c) HF [171] (d) MR [125] (e) Ours

Figure F.20 – Qualitative comparison of our reconstruction method with various competing approaches. We used the datasets from [10]. The dataset does not contain ground truth images, thus only a qualitative comparison is possible. For SOFIE and MR, we used images provided by the authors, for which the parameters were tuned for each dataset. For HF, we ran the code provided by the authors, manually tuned the parameters on these datasets to achieve the best visual quality, and additionally applied a bilateral filter to clean the high frequency noise present in the original reconstructions.

(a) Events (b) VI sensor frame (c) Our reconstruction

Figure F.21 – Example HDR reconstructions on the MVSEC automotive dataset [212]. The standard frames were recorded with a high-quality VI sensor with auto-exposure activated. Because the camera is facing directly the sun, the standard frames (b) are either under- or over-exposed since the limited dynamic range of the standard sensor cannot cope with the high dynamic range of the scene. By contrast, the events (a) capture the whole dynamic range of the scene, which our method successfully reconstructs to high dynamic range images (c), allow to discover details that were not visible in the standard frames.



(a) Events (b) VI sensor frame (c) Our reconstruction

Figure F.22 – Example HDR reconstructions on the MVSEC automotive dataset [212] at night. The standard frames were recorded with a high-quality VI sensor with auto-exposure activated. Because of low light during the night, the standard frames (b) are severely degraded. By contrast, the events (a) still can capture the whole dynamic range of the scene, which our method successfully recovers (c), allowing to discover details that were not visible in the standard frames.

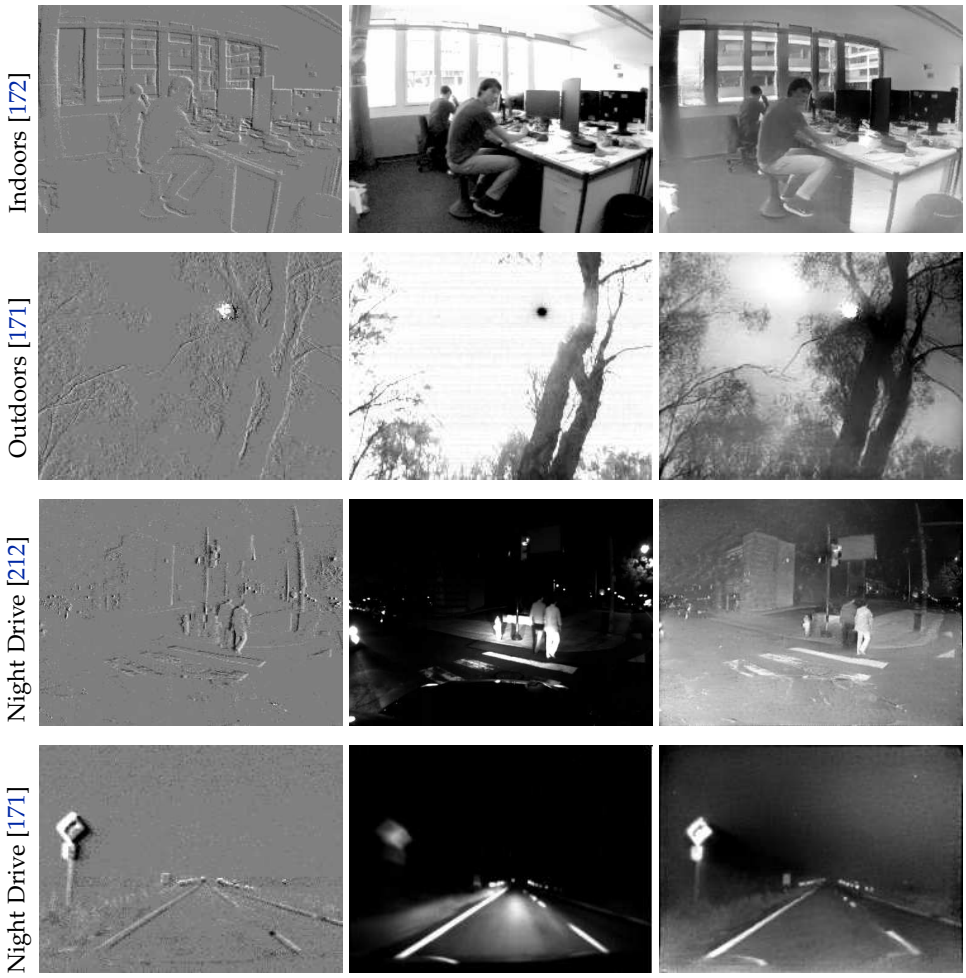


Figure F.23 – Video reconstruction under challenging lighting. First row: indoor sequence [172]. Second row: outdoor sequence from [171]. Third row: night driving sequence from [212]. Fourth row: night driving sequence from [171]. The frames from the conventional camera (b) suffer from under- or over-exposure, while the events (a) capture the whole dynamic range of the scene, which our method successfully recovers (c).

Appendix: Object Classification

Below we detail the exact modalities of our reconstruction method for each of the dataset which we used for our evaluation of object classification (Section 5.1 in the paper), as well as the specific architectures used and training modalities.

N-MNIST. To reconstruct images with our networks, we used an event window of $N = 1{,}000$ events. We passed every event sequence into our network, resulting in a video, from which we keep the final image as input for the classification network. To match the images from the original MNIST dataset, we additionally binarize the reconstructed image (whose values lie in $[0, 1]$) with a threshold of 0.5. The train and test images were normalized so that the mean value of each image is 0.1307 and the variance 0.3081. We used the official train and test split provided in the M-NNIST dataset. As there is no standard state of the art architecture for MNIST, we used a simple CNN architecture as our classification network, composed of the following blocks:

- 2D convolution (stride: 5, output channels: 32) + ReLU
- 2D convolution (stride: 5, output channels: 64) + ReLU
- 2D max pooling (size: 2) + Dropout
- Fully connected layer (output size: 128 neurons) + ReLU
- Fully connected layer (output size: 10 neurons)

We used the cross entropy loss, and trained the network for 15 epochs using the ADAM optimizer, with a learning rate of 0.001.

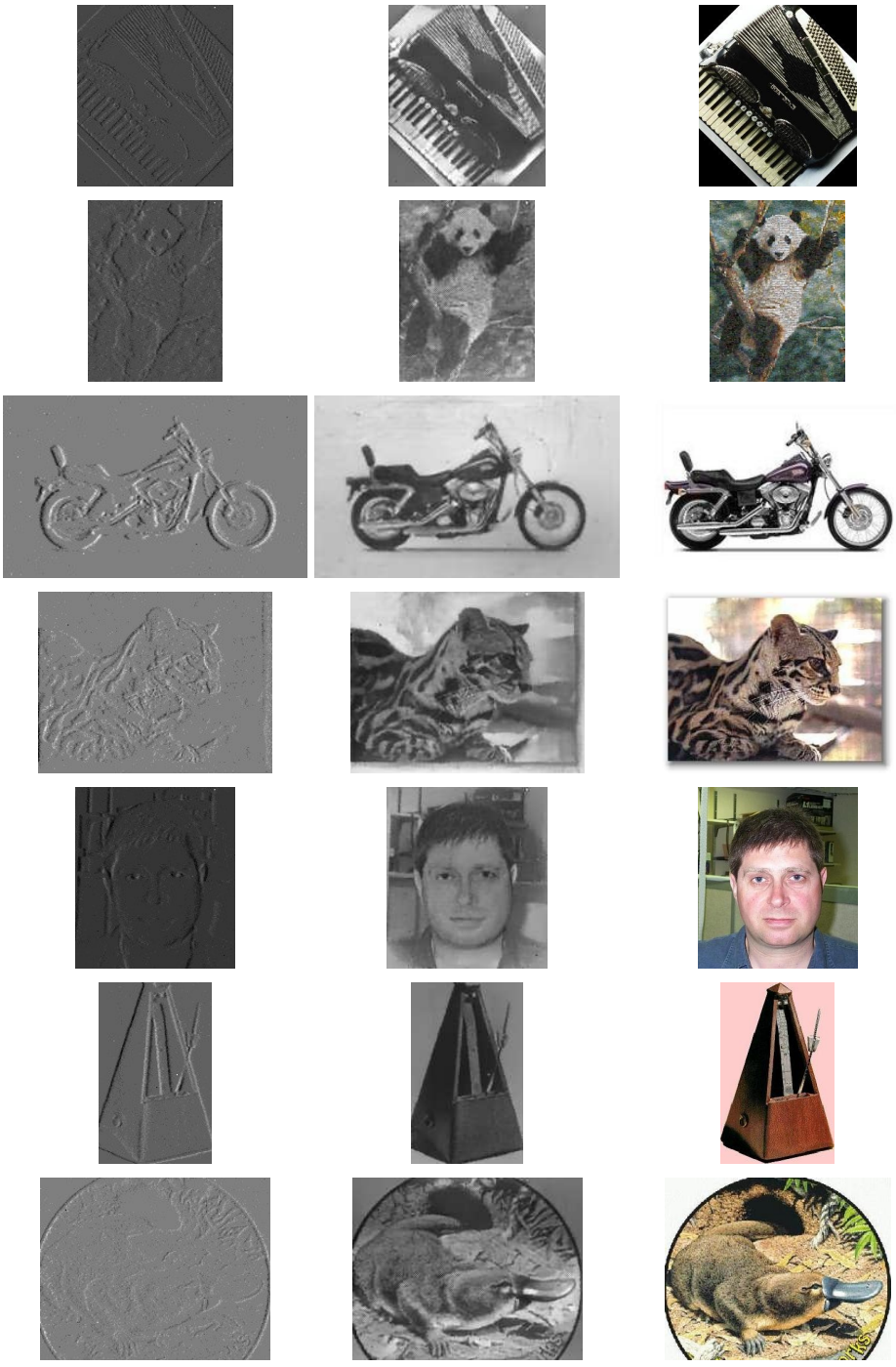
N-CARS. We used windows of events with a fixed temporal size of 20 ms, and used the last reconstructed image from the video as input to the classification network. We used the official train and test split provided by the N-CARS dataset. We used a ResNet18 [71] architecture (with an additional fully connected final layer with 2 output neurons), initialized with weights pretrained on ImageNet [167], and fine-tuned the network using the reconstructed images from the training set for 20 epochs, using SGD with a learning rate of 0.001 (decayed by factor of 0.1 every 7 epochs), and momentum of 0.1.

N-Caltech101. For image reconstruction, we used windows of $N = 10{,}000$, events and used the last reconstructed image as input to the classification network. Since there is no official train and test split for the N-Caltech101 dataset, we split the dataset randomly into two third training sequences (5,863 sequences) and one third testing sequences (2,396 sequences), following the methodology used by HATS [184]. The train and test images were converted to 3-channel grayscale images (*i.e.* the three channels

are the same), and normalized so that the mean value of each image is 0.485 and the variance 0.229. We also performed data augmentation at train time (random horizontal flips, and random crop of size 224). At test time, we resized all the images to 256×256 and cropped the image around the center with a size of 224. We used a ResNet18 architecture (with an additional fully-connected final layer with 101 output neurons), initialized with weights pretrained on ImageNet, and fine-tuned the network using the reconstructed images from the training set for 25 epochs using SGD with an initial learning rate of 0.001 (decayed by a factor of 0.1 every 7 epochs) and momentum of 0.1. Fig. F.24 shows additional reconstruction examples from the N-Caltech101 dataset.

Appendix: Visual-Inertial Odometry

Figs. F.25, F.26 and F.27 provide additional results on the visual-inertial odometry experiments presented in the main paper. Specifically, they provide, for each sequence used in our evaluation, the evolution of the mean translation and rotation error as a function of the travelled distance for our approach, UltimateSLAM (E+I), and UltimateSLAM (E+F+I).



(a) Events (b) Our Reconstruction (c) Original Image

Figure F.24 – (a) Previews of some event sequences from the N-Caltech101 dataset [132] which features event sequences converted from the Caltech101 dataset. (b) our reconstructions (from events only) preserve many of the details and statistics of the original images (c). Note that these datasets feature planar motion (since Caltech101 images were projected on white wall to record the events), which coincides with the type of motions present in the simulated data, which explains in part the outstanding visual quality of the reconstructions.

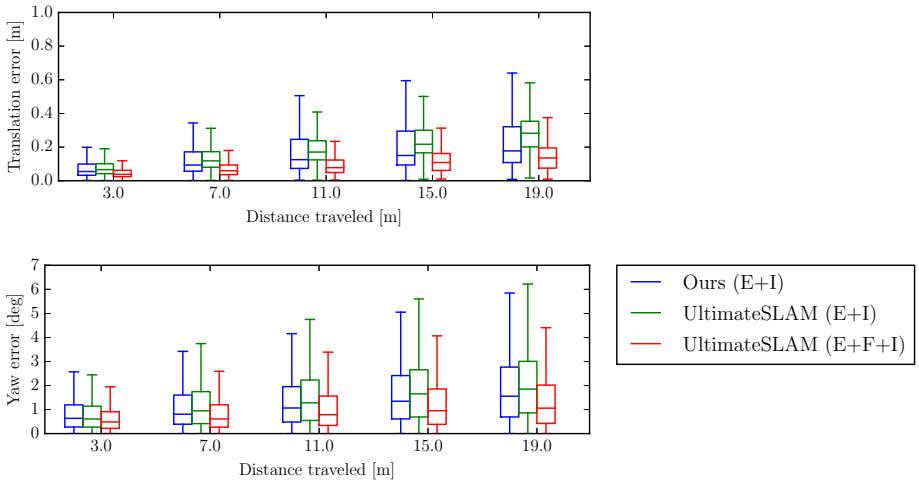


Figure F.25 – Evolution of the overall mean translation error (in meters) and mean rotation error (in degrees), averaged across all the datasets used in our evaluation.

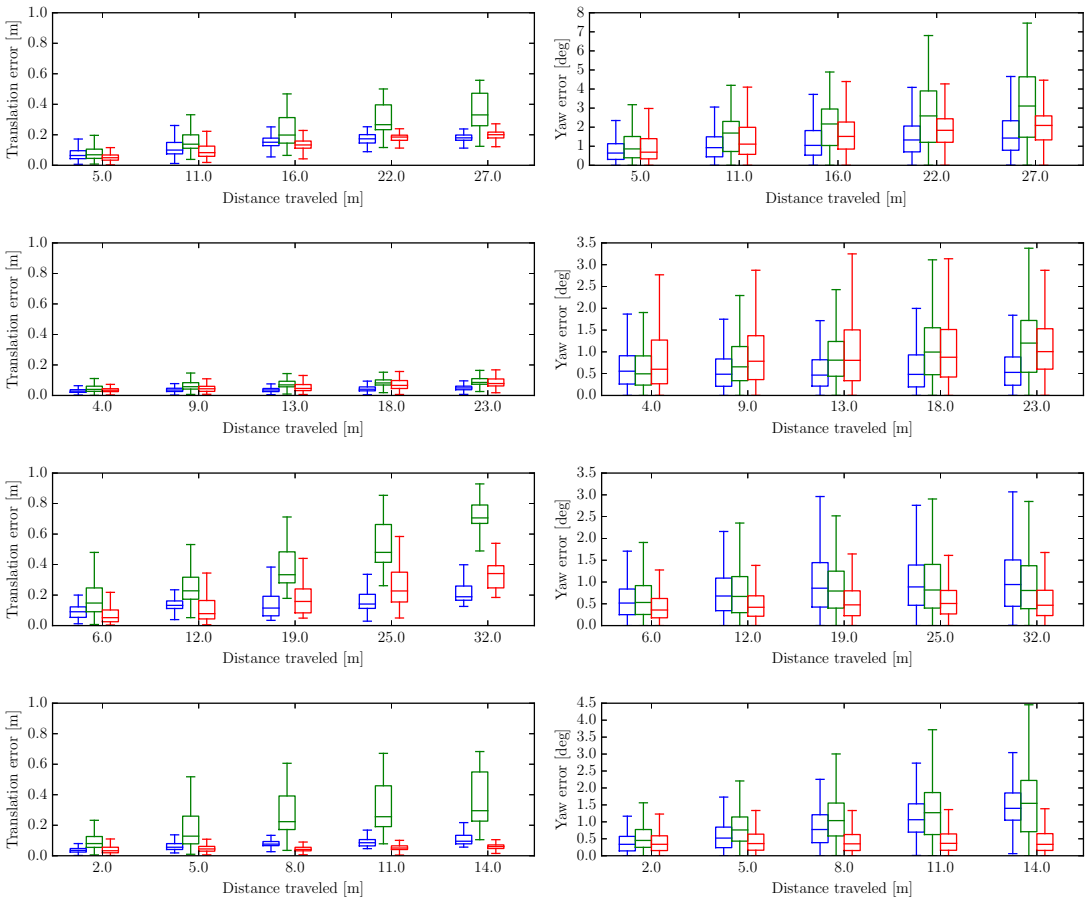


Figure F.26 – Evolution of the mean translation error (in meters) and mean rotation error (in degrees), as a function of the travelled distance. Sequences from top to bottom: 'shapes_translation', 'poster_translation', 'boxes_translation', 'dynamic_translation'. Legend identical to Fig. F.25.

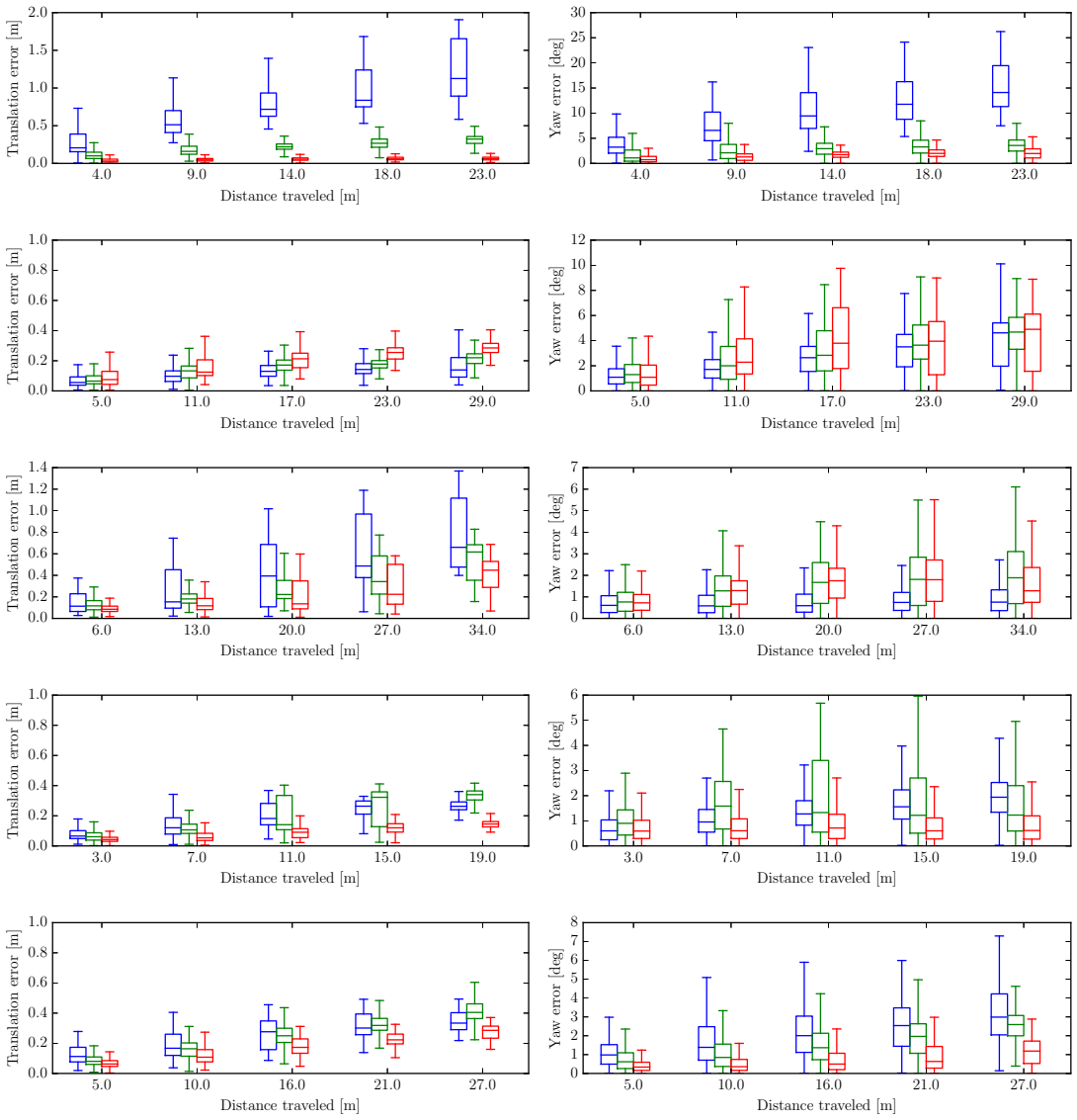


Figure F.27 – Evolution of the mean translation error (in meters) and mean rotation error (in degrees), as a function of the travelled distance. Sequences from top to bottom: 'shapes_6dof', 'poster_6dof', 'boxes_6dof', 'dynamic_6dof', 'hdr_boxes'. Legend identical to Fig. F.25.

Bibliography

[1] https://www.prophesee.ai/event-based-evk/. 2019.

[2] A. Agarwal, K. Mierle, et al. *Ceres Solver*. http://ceres-solver.org.

[3] M. Aharon, M. Elad, and A. M. Bruckstein. "K-SVD: An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation". In: *IEEE Trans. Signal Process.* 54.11 (2006), pp. 4311–4322. DOI: 10.1109/TSP.2006.881199.

[4] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I. Lungu, M. B. Milde, F. Corradi, A. Linares-Barranco, S. Liu, and T. Delbrück. "NullHop: A Flexible Convolutional Neural Network Accelerator Based on Sparse Representations of Feature Maps". In: *IEEE Trans. Neural Netw. Learn. Syst.* (2018). DOI: 10.1109/TNNLS.2018.2852335.

[5] F. Akopyan, J. Sawada, A. Cassidy, R. Alvarez-Icaza, J. Arthur, P. Merolla, N. Imam, Y. Nakamura, P. Datta, G.-J. Nam, B. Taba, M. Beakes, B. Brezzo, J. B. Kuang, R. Manohar, W. P. Risk, B. Jackson, and D. S. Modha. "TrueNorth: Design and Tool Flow of a 65 mW 1 Million Neuron Programmable Neurosynaptic Chip". In: *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.* 34.10 (2015), pp. 1537–1557. DOI: 10.1109/TCAD.2015.2474396.

[6] I. Alzugaray and M. Chli. "Asynchronous Corner Detection and Tracking for Event Cameras in Real Time". In: *IEEE Robot. Autom. Lett.* 3.4 (Oct. 2018), pp. 3177–3184. DOI: 10.1109/LRA.2018.2849882.

[7] A. Amir, B. Taba, D. Berg, T. Melano, J. McKinstry, C. D. Nolfo, T. Nayak, A. Andreopoulos, G. Garreau, M. Mendoza, J. Kusnitz, M. Debole, S. Esser, T. Delbruck, M. Flickner, and D. Modha. "A Low Power, Fully Event-Based Gesture Recognition System". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2017, pp. 7388–7397. DOI: 10.1109/CVPR.2017.781.

[8] S. Baker and I. Matthews. "Lucas-Kanade 20 Years On: A Unifying Framework". In: *Int. J. Comput. Vis.* 56.3 (2004), pp. 221–255.

[9] P. Bardow. "Estimating General Motion and Intensity from Event Cameras". PhD thesis. Imperial College London, Department of Computing, 2018.

[10] P. Bardow, A. J. Davison, and S. Leutenegger. "Simultaneous Optical Flow and Intensity Estimation From an Event Camera". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2016, pp. 884–892. DOI: 10.1109/CVPR.2016.102.

[11] F. Barranco, C. Fermuller, Y. Aloimonos, and T. Delbruck. "A Dataset for Visual Navigation with Neuromorphic Methods". In: *Front. Neurosci.* 10 (2016), p. 49. DOI: 10.3389/fnins.2016.00049.

[12] S. Barua, Y. Miyatani, and A. Veeraraghavan. "Direct face detection and video reconstruction from event cameras". In: *IEEE Winter Conf. Appl. Comput. Vis. (WACV)*. 2016, pp. 1–9. DOI: 10.1109/WACV.2016.7477561.

[13] R. Benosman, C. Clercq, X. Lagorce, S.-H. Ieng, and C. Bartolozzi. "Event-Based Visual Flow". In: *IEEE Trans. Neural Netw. Learn. Syst.* 25.2 (2014), pp. 407–417. DOI: 10.1109/TNNLS.2013.2273537.

[14] R. Benosman, S.-H. Ieng, C. Clercq, C. Bartolozzi, and M. Srinivasan. "Asynchronous frameless event-based optical flow". In: *Neural Netw.* 27 (2012), pp. 32–37. DOI: 10.1016/j.neunet.2011.11.001.

[15] J. Binas, D. Neil, S.-C. Liu, and T. Delbruck. "DDD17: End-To-End DAVIS Driving Dataset". In: *ICML Workshop on Machine Learning for Autonomous Vehicles*. 2017.

[16] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart. "Robust Visual Inertial Odometry Using a Direct EKF-Based Approach". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2015.

[17] D. Borer, T. Delbrück, and T. Rösgen. "Three-dimensional particle tracking velocimetry using dynamic vision sensors". In: *Experiments in Fluids* 58.165 (2017). DOI: 10.1007/s00348-017-2452-5.

[18] C. Brandli, R. Berner, M. Yang, S.-C. Liu, and T. Delbruck. "A 240x180 130dB 3us Latency Global Shutter Spatiotemporal Vision Sensor". In: *IEEE J. Solid-State Circuits* 49.10 (2014), pp. 2333–2341. DOI: 10.1109/JSSC.2014.2342715.

[19] C. Brandli, L. Muller, and T. Delbruck. "Real-time, high-speed video decompression using a frame- and event-based DAVIS sensor". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2014, pp. 686–689. DOI: 10.1109/ISCAS.2014.6865228.

[20] S. Bryner, G. Gallego, H. Rebecq, and D. Scaramuzza. "Event-based, Direct Camera Tracking from a Photometric 3D Map using Nonlinear Optimization". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2019.

[21] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. D. Reid, and J. J. Leonard. "Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age". In: *IEEE Trans. Robot.* 32.6 (2016), pp. 1309–1332.

[22] L. A. Camunas-Mesa, T. Serrano-Gotarredona, S. H. Ieng, R. B. Benosman, and B. Linares-Barranco. "On the use of Orientation Filters for 3D Reconstruction in Event-Driven Stereo Vision". In: *Front. Neurosci.* 8 (2014), p. 48. DOI: 10.3389/fnins.2014.00048.

[23] S. J. Carey, A. Lopich, D. R. Barr, B. Wang, and P. Dudek. "A 100,000 fps Vision Sensor with Embedded 535 GOPS/W 256x256 SIMD Processor Array". In: *VLSI Circuits Symp.* 2013.

[24] A. Censi and D. Scaramuzza. "Low-Latency Event-Based Visual Odometry". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 703–710. DOI: 10.1109/ICRA.2014.6906931.

[25] S. Chen. *Introduction of Celex Family Sensor and Event/Frame/Optical-flow Hybrid Processing*. http://rpg.ifi.uzh.ch/docs/CVPR19workshop/CVPRW19_CelePixel.pdf. 2019.

[26] T. Chin, S. Bagchi, A. P. Eriksson, and A. van Schaik. "Star Tracking using an Event Camera". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2019.

[27] X. Clady, S.-H. Ieng, and R. Benosman. "Asynchronous event-based corner detection and matching". In: *Neural Netw.* 66 (2015), pp. 91–106. DOI: 10.1016/j.neunet.2015.02.013.

[28] R. T. Collins. "A Space-Sweep Approach to True Multi-Image Matching". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 1996, pp. 358–363. DOI: 10.1109/CVPR.1996.517097.

[29] J. Conradt, M. Cook, R. Berner, P. Lichtsteiner, R. J. Douglas, and T. Delbruck. "A Pencil Balancing Robot using a Pair of AER Dynamic Vision Sensors". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2009, pp. 781–784. DOI: 10.1109/ISCAS.2009.5117867.

[30] M. Cook, L. Gugelmann, F. Jug, C. Krautz, and A. Steger. "Interacting maps for fast visual interpretation". In: *Int. Joint Conf. Neural Netw. (IJCNN)*. 2011, pp. 770–776. DOI: 10.1109/IJCNN.2011.6033299.

[31] P. Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer Tracts in Advanced Robotics. Springer, 2011.

[32] A. Crivellaro, P. Fua, and V. Lepetit. *Dense Methods for Image Alignment with an Application to 3D Tracking*. Tech. rep. 197866. EPFL, 2014.

[33] *DAVIS Specifications*. https://inivation.com/wp-content/uploads/2019/07/2019-07-09-DVS-Specifications.pdf. 2019.

[34] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse. "MonoSLAM: Real-time single camera SLAM". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 29.6 (June 2007), pp. 1052–1067.

[35] A. J. Davison. "FutureMapping: The Computational Structure of Spatial AI Systems". In: *arXiv e-prints* (Mar. 2018). URL: http://arxiv.org/abs/1803.11288.

[36] T. Delbruck. "Frame-free dynamic digital vision". In: *Proc. Int. Symp. Secure-Life Electron.* 2008, pp. 21–26.

[37] T. Delbruck. "Neuromorphic vision sensing and processing". In: *Eur. Solid-State Device Research Conf. (ESSDERC)*. 2016, pp. 7–14. DOI: 10.1109/ESSDERC.2016.7599576.

[38] T. Delbruck and M. Lang. "Robotic Goalie with 3ms Reaction Time at 4% CPU Load Using Event-Based Dynamic Vision Sensor". In: *Front. Neurosci.* 7 (2013), p. 223. DOI: 10.3389/fnins.2013.00223.

[39] T. Delbruck and P. Lichtsteiner. "Fast Sensory Motor Control Based on Event-Based Hybrid Neuromorphic-Procedural System". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2007, pp. 845–848. DOI: 10.1109/ISCAS.2007.378038.

[40] T. Delbruck, V. Villanueva, and L. Longinotti. "Integration of dynamic vision sensor with inertial measurement unit for electronically stabilized event-based vision". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2014, pp. 2636–2639. DOI: 10.1109/ISCAS.2014.6865714.

[41] J. Delmerico, T. Cieslewski, H. Rebecq, M. Faessler, and D. Scaramuzza. "Are We Ready for Autonomous Drone Racing? The UZH-FPV Drone Racing Dataset". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2019.

[42] J. Delmerico and D. Scaramuzza. "A Benchmark Comparison of Monocular Visual-Inertial Odometry Algorithms for Flying Robots". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2018.

[43] A. Dosovitskiy, P. Fischer, E. Ilg, P. Häusser, C. Hazırbaş, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox. "FlowNet: Learning Optical Flow with Convolutional Networks". In: *Int. Conf. Comput. Vis. (ICCV)*. 2015, pp. 2758–2766. DOI: 10.1109/ICCV.2015.316.

[44] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. "CARLA: An Open Urban Driving Simulator". In: *Conf. on Robotics Learning (CoRL)*. 2017.

[45] D. Drazen, P. Lichtsteiner, P. Häfliger, T. Delbrück, and A. Jensen. "Toward real-time particle tracking using an event-based dynamic vision sensor". In: *Experiments in Fluids* 51.5 (2011), pp. 1465–1469. DOI: 10.1007/s00348-011-1207-y.

[46] J. Engel, J. Schöps, and D. Cremers. "LSD-SLAM: Large-Scale Direct Monocular SLAM". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2014.

[47] J. Engel, V. Koltun, and D. Cremers. "Direct Sparse Odometry". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.3 (Mar. 2018), pp. 611–625. DOI: 10.1109/TPAMI.2017.2658577.

[48] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza. "Autonomous, Vision-based Flight and Live Dense 3D Mapping with a Quadrotor MAV". In: *J. Field Robot.* 33.4 (2016), pp. 431–450. DOI: 10.1002/rob.21581.

[49] D. Falanga, S. Kim, and D. Scaramuzza. "How Fast is Too Fast? The Role of Perception Latency in High-Speed Sense and Avoid". In: (2019), pp. 1884–1891. DOI: 10.1109/LRA.2019.2898117.

[50] L. Fei-Fei, R. Fergus, and P. Perona. "One-shot learning of object categories". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 28.4 (2006), pp. 594–611. DOI: 10.1109/TPAMI.2006.79.

[51] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. "IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation". In: *Robotics: Science and Systems (RSS)*. 2015. DOI: 10.15607/RSS.2015.XI.006.

[52] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. "On-Manifold Preintegration for Real-Time Visual-Inertial Odometry". In: *IEEE Trans. Robot.* 33.1 (2017), pp. 1–21. DOI: 10.1109/TRO.2016.2597321.

[53] C. Forster, M. Pizzoli, and D. Scaramuzza. "SVO: Fast Semi-Direct Monocular Visual Odometry". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 15–22. DOI: 10.1109/ICRA.2014.6906584.

[54] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza. "SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems". In: *IEEE Trans. Robot.* 33.2 (2017), pp. 249–265. DOI: 10.1109/TRO.2016.2623335.

[55] S. B. Furber, D. R. Lester, L. A. Plana, J. D. Garside, E. Painkras, S. Temple, and A. D. Brown. "Overview of the SpiNNaker System Architecture". In: *IEEE Trans. Comput.* 62.12 (2013), pp. 2454–2467. DOI: 10.1109/TC.2012.142.

[56] P. Furgale, J. Rehder, and R. Siegwart. "Unified Temporal and Spatial Calibration for Multi-Sensor Systems". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2013. DOI: 10.1109/IROS.2013.6696514.

[57] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza. "Event-based Vision: A Survey". In: *arXiv e-prints* abs/1904.08405 (2019). URL: http://arxiv.org/abs/1904.08405.

[58] G. Gallego, C. Forster, E. Mueggler, and D. Scaramuzza. "Event-based Camera Pose Tracking using a Generative Event Model". arXiv:1510.01972. 2015.

[59] G. Gallego, M. Gehrig, and D. Scaramuzza. "Focus Is All You Need: Loss Functions For Event-based Vision". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019.

[60] G. Gallego, J. E. A. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza. "Event-based, 6-DOF Camera Tracking for High-Speed Applications". arXiv:1607.03468. 2016.

[61] G. Gallego, J. E. A. Lund, E. Mueggler, H. Rebecq, T. Delbruck, and D. Scaramuzza. "Event-based, 6-DOF Camera Tracking from Photometric Depth Maps". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 40.10 (Oct. 2018), pp. 2402–2412. DOI: 10.1109/TPAMI.2017.2769655.

[62] G. Gallego, H. Rebecq, and D. Scaramuzza. "A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 3867–3876. DOI: 10.1109/CVPR.2018.00407.

[63] G. Gallego and D. Scaramuzza. "Accurate Angular Velocity Estimation with an Event Camera". In: *IEEE Robot. Autom. Lett.* 2.2 (2017), pp. 632–639. DOI: 10.1109/LRA.2016.2647639.

[64] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza. "Asynchronous, Photometric Feature Tracking using Events and Frames". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2018, pp. 766–781. DOI: 10.1007/978-3-030-01258-8_46.

[65] D. Gehrig, H. Rebecq, G. Gallego, and D. Scaramuzza. "EKLT: Asynchronous Photometric Feature Tracking using Events and Frames". In: *Int. J. Comput. Vis.* (2019). DOI: 10.1007/s11263-019-01209-w.

[66] A. Geiger, P. Lenz, and R. Urtasun. "Are we ready for Autonomous Driving? The KITTI Vision Benchmark Suite". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2012.

[67] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. "Generative adversarial nets". In: *Conf. Neural Inf. Process. Syst. (NIPS)*. 2014, pp. 2672–2680.

[68] B. Graham, M. Engelcke, and L. van der Maaten. "3D semantic segmentation with submanifold sparse convolutional networks". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 9224–9232. DOI: 10.1109/CVPR.2018.00961.

[69] C. Harris and M. Stephens. "A combined corner and edge detector". In: *Proc. Fourth Alvey Vision Conf.* Vol. 15. 1988, pp. 147–151. DOI: 10.5244/C.2.23.

[70] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. 2nd Edition. Cambridge University Press, 2003. DOI: 10.1017/CBO9780511811685.

[71] K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2016, pp. 770–778. DOI: 10.1109/cvpr.2016.90.

[72] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (Nov. 1997), pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735.

[73] B. J. P. Hordijk, K. Y. Scheper, and G. C. D. Croon. "Vertical landing for micro air vehicles using event-based optical flow". In: *J. Field Robot.* 35.1 (Jan. 2017), pp. 69–90.

[74] E. Ilg, N. Mayer, T. Saikia, M. Keuper, A. Dosovitskiy, and T. Brox. "FlowNet 2.0: Evolution of Optical Flow Estimation with Deep Networks". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2017, pp. 1647–1655. DOI: 10.1109/cvpr.2017.179.

[75] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proc. Int. Conf. Mach. Learning (ICML)*. 2015.

[76] J. Johnson, A. Alahi, and F. Li. "Perceptual Losses for Real-Time Style Transfer and Super-Resolution". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2016. DOI: 10.1007/978-3-319-46475-6_43.

[77] E. S. Jones and S. Soatto. "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach". In: *Int. J. Robot. Research* 30.4 (Apr. 2011).

[78] J. Kaiser, J. C. V. Tieck, C. Hubschneider, P. Wolf, M. Weber, M. Hoff, A. Friedrich, K. Wojtasik, A. Roennau, R. Kohlhaas, R. Dillmann, and J. M. Zöllner. "Towards a framework for end-to-end control of a simulated vehicle with spiking neural networks". In: *IEEE Int. Conf. on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*. 2016, pp. 127–134.

[79] H. Kim, A. Handa, R. Benosman, S.-H. Ieng, and A. J. Davison. "Simultaneous Mosaicing and Tracking with an Event Camera". In: *British Mach. Vis. Conf. (BMVC)*. 2014. DOI: 10.5244/C.28.26.

[80] H. Kim, S. Leutenegger, and A. J. Davison. "Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2016, pp. 349–364. DOI: \protect{10.1007/978-3-319-46466-4_21}.

[81] D. P. Kingma and J. L. Ba. "Adam: A Method for Stochastic Optimization". In: *Int. Conf. Learn. Representations (ICLR)* (2015).

[82] G. Klein and D. Murray. "Parallel tracking and mapping for small AR workspaces". In: *IEEE ACM Int. Sym. Mixed and Augmented Reality (ISMAR)*. Nara, Japan, Nov. 2007, pp. 225–234.

[83] N. Koenig and A. Howard. "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2004, pp. 2149–2154. DOI: 10.1109/IROS.2004.1389727.

[84] J. Kogler, M. Humenberger, and C. Sulzbachner. "Event-Based Stereo Matching Approaches for Frameless Address Event Stereo Data". In: *Int. Symp. Adv. Vis. Comput. (ISVC)*. 2011, pp. 674–685. DOI: 10.1007/978-3-642-24028-7_62.

[85] J. Kogler, C. Sulzbachner, M. Humenberger, and F. Eibensteiner. "Address-Event Based Stereo Vision with Bio-Inspired Silicon Retina Imagers". In: *Advances in Theory and Applications of Stereo Vision*. InTech, 2011, pp. 165–188. DOI: 10.5772/12941.

[86] A. Krizhevsky, I. Sutskever, and G. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Conf. Neural Inf. Process. Syst. (NIPS)*. 2012, pp. 1097–1105.

[87] B. Kueng, E. Mueggler, G. Gallego, and D. Scaramuzza. "Low-latency Visual Odometry using Event-based Feature Tracks". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2016, pp. 16–23. DOI: 10.1109/IROS.2016.7758089.

[88] X. Lagorce, G. Orchard, F. Gallupi, B. E. Shi, and R. Benosman. "HOTS: A Hierarchy Of event-based Time-Surfaces for pattern recognition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.7 (July 2017), pp. 1346–1359. DOI: 10.1109/TPAMI.2016.2574707.

[89] W. Lai, J. Huang, O. Wang, E. Shechtman, E. Yumer, and M. Yang. "Learning Blind Video Temporal Consistency". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2018.

[90] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. "Gradient-based learning applied to document recognition". In: *Proc. IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[91] J. H. Lee, T. Delbruck, M. Pfeiffer, P. K. Park, C.-W. Shin, H. Ryu, and B. C. Kang. "Real-Time Gesture Interface Based on Event-Driven Processing From Stereo Silicon Retinas". In: *IEEE Trans. Neural Netw. Learn. Syst.* 25.12 (2014), pp. 2250–2263.

[92] J. Lee, T. Delbruck, P. K. J. Park, M. Pfeiffer, C.-W. Shin, H. Ryu, and B. C. Kang. "Live demonstration: Gesture-Based remote control using stereo pair of dynamic vision sensors". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2012. DOI: 10.1109/ISCAS.2012.6272144.

[93] S. Leutenegger, P. Furgale, V. Rabaud, M. Chli, K. Konolige, and R. Siegwart. "Keyframe-Based Visual-Inertial SLAM Using Nonlinear Optimization". In: *Robotics: Science and Systems (RSS)*. 2013.

[94] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. "Keyframe-Based Visual-Inertial SLAM using Nonlinear Optimization". In: *Int. J. Robot. Research* (2015).

[95] W. Li, S. Saeedi, J. McCormac, R. Clark, D. Tzoumanikas, Q. Ye, Y. Huang, R. Tang, and S. Leutenegger. "InteriorNet: Mega-scale Multi-sensor Photo-realistic Indoor Scenes Dataset". In: *British Mach. Vis. Conf. (BMVC)*. 2018.

[96] Z. Li, T. Dekel, F. Cole, R. Tucker, N. Snavely, C. Liu, and W. T. Freeman. "Learning the Depths of Moving People by Watching Frozen People". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019.

[97] Z. Li and N. Snavely. "MegaDepth: Learning Single-View Depth Prediction From Internet Photos". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018.

[98] P. Lichtsteiner, C. Posch, and T. Delbruck. "A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor". In: *IEEE J. Solid-State Circuits* 43.2 (2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[99] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. "Microsoft COCO: Common Objects in Context". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2014.

[100] M. Litzenberger, A. N. Belbachir, N. Donath, G. Gritsch, H. Garn, B. Kohn, C. Posch, and S. Schraml. "Estimation of Vehicle Speed Based on Asynchronous Data from a Silicon Retina Optical Sensor". In: *IEEE Intell. Transp. Sys. Conf.* 2006, pp. 653–658. DOI: 10.1109/ITSC.2006.1706816.

[101] M. Litzenberger, C. Posch, D. Bauer, A. N. Belbachir, P. Schön, B. Kohn, and H. Garn. "Embedded Vision System for Real-Time Object Tracking using an Asynchronous Transient Vision Sensor". In: *Digital Signal Processing Workshop*. 2006, pp. 173–178. DOI: 10.1109/DSPWS.2006.265448.

[102] M. Liu and T. Delbruck. "Block-matching optical flow for dynamic vision sensors: Algorithm and FPGA implementation". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2017. DOI: 10.1109/ISCAS.2017.8050295.

[103] B. D. Lucas and T. Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Int. Joint Conf. Artificial Intell. (IJCAI)*. 1981, pp. 674–679.

[104] T. Lupton and S. Sukkarieh. "Visual-Inertial-Aided Navigation for High-Dynamic Motion in Built Environments Without Initial Conditions". In: *IEEE Trans. Robot.* 28.1 (Feb. 2012), pp. 61–76.

[105] Y. Ma, S. Soatto, J. Košecká, and S. Sastry. *An Invitation to 3-D Vision: From Images to Geometric Models*. Springer, 2004.

[106] M. Mahowald. "The Silicon Retina". In: *An Analog VLSI System for Stereoscopic Vision*. Boston, MA: Springer US, 1994, pp. 4–65. ISBN: 978-1-4615-2724-4. DOI: 10.1007/978-1-4615-2724-4_2.

[107] M. Mahowald. "VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function". PhD thesis. Pasadena, California: California Institute of Technology, May 1992.

[108] J. Manderscheid, A. Sironi, N. Bourdis, D. Migliore, and V. Lepetit. "Speed Invariant Time Surface for Learning to Detect Corner Points with Event-Based Cameras". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019.

[109] A. I. Maqueda, A. Loquercio, G. Gallego, N. García, and D. Scaramuzza. "Event-based Vision meets Deep Learning on Steering Prediction for Self-driving Cars". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 5419–5427. DOI: 10.1109/CVPR.2018.00568.

[110] A. Marcireau, S.-H. Ieng, C. Simon-Chane, and R. B. Benosman. "Event-Based Color Segmentation With a High Dynamic Range Sensor". In: *Front. Neurosci.* 12 (2018). DOI: 10.3389/fnins.2018.00135.

[111] N. Matsuda, O. Cossairt, and M. Gupta. "MC3D: Motion Contrast 3D Scanning". In: *IEEE Int. Conf. Comput. Photography (ICCP)*. 2015, pp. 1–10. DOI: 10.1109/ICCPHOT.2015.7168370.

[112] C. A. Mead and M. Mahowald. "A silicon model of early visual processing". In: *Neural Netw.* 1.1 (1989), pp. 91–97. DOI: 10.1016/0893-6080(88)90024-X.

[113] L. Meier, P. Tanskanen, L. Heng, G. H. Lee, F. Fraundorfer, and M. Pollefeys. "PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision". In: *Auton. Robots* 33.1–2 (2012), pp. 21–39.

[114] A. Mitrokhin, C. Fermuller, C. Parameshwara, and Y. Aloimonos. "Event-based Moving Object Detection and Tracking". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2018.

[115] D. P. Moeys, F. Corradi, C. Li, S. A. Bamford, L. Longinotti, F. F. Voigt, S. Berry, G. Taverni, F. Helmchen, and T. Delbruck. "A Sensitive Dynamic and Active Pixel Vision Sensor for Color or Neural Imaging Applications". In: *IEEE Trans. Biomed. Circuits Syst.* 12.1 (Feb. 2018), pp. 123–136. DOI: 10.1109/TBCAS.2017.2759783.

[116] D. P. Moeys, C. Li, J. N. P. Martel, S. Bamford, L. Longinotti, V. Motsnyi, D. S. S. Bello, and T. Delbruck. "Color Temporal Contrast Sensitivity in Dynamic Vision Sensors". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2017, pp. 1–4. DOI: 10.1109/ISCAS.2017.8050412.

[117] K. Mohta, M. Watterson, Y. Mulgaonkar, S. Liu, C. Qu, A. Makineni, K. Saulnier, K. Sun, A. Zhu, J. Delmerico, K. Karydis, N. Atanasov, G. Loianno, D. Scaramuzza, K. Daniilidis, C. J. Taylor, and V. Kumar. "Fast, autonomous flight in GPS-denied and cluttered environments". In: *J. Field Robot.* 35.1 (2018), pp. 101–120. DOI: 10.1002/rob.21774.

[118] S. Mostafavi I., L. Wang, Y.-S. Ho, and K.-J. Y. Yoon. "Event-based High Dynamic Range Image and Very High Frame Rate Video Generation using Conditional Generative Adversarial Networks". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019.

[119] A. I. Mourikis and S. I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. Apr. 2007, pp. 3565–3572.

[120] E. Mueggler, C. Bartolozzi, and D. Scaramuzza. "Fast Event-based Corner Detection". In: *British Mach. Vis. Conf. (BMVC)*. 2017.

[121] E. Mueggler, N. Baumli, F. Fontana, and D. Scaramuzza. "Towards Evasive Maneuvers with Quadrotors using Dynamic Vision Sensors". In: *Eur. Conf. Mobile Robots (ECMR)*. 2015, pp. 1–8. DOI: 10.1109/ECMR.2015.7324048.

[122] E. Mueggler, G. Gallego, H. Rebecq, and D. Scaramuzza. "Continuous-Time Visual-Inertial Odometry for Event Cameras". In: *IEEE Trans. Robot.* 34.6 (Dec. 2018), pp. 1425–1440. DOI: 10.1109/tro.2018.2858287.

[123] E. Mueggler, B. Huber, and D. Scaramuzza. "Event-based, 6-DOF Pose Tracking for High-Speed Maneuvers". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2014, pp. 2761–2768. DOI: 10.1109/IROS.2014.6942940.

[124] E. Mueggler, H. Rebecq, G. Gallego, T. Delbruck, and D. Scaramuzza. "The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM". In: *Int. J. Robot. Research* 36.2 (2017), pp. 142–149. DOI: 10.1177/0278364917691115.

[125] G. Munda, C. Reinbacher, and T. Pock. "Real-Time Intensity-Image Reconstruction for Event Cameras Using Manifold Regularisation". In: *Int. J. Comput. Vis.* 126.12 (July 2018), pp. 1381–1393. DOI: 10.1007/s11263-018-1106-2.

[126] R. Mur-Artal and J. D. Tardós. "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras". In: *IEEE Trans. Robot.* 33.5 (Oct. 2017), pp. 1255–1262. DOI: 10.1109/TRO.2017.2705103.

[127] F. B. Naeini, A. Alali, R. Al-Husari, A. Rigi, M. K. AlSharman, D. Makris, and Y. Zweiri. "A Novel Dynamic-Vision-Based Approach for Tactile Sensing Applications". In: *IEEE Trans. on Instr. and Meas.* (2019). DOI: 10.1109/TIM.2019.2919354.

[128] D. Neil, M. Pfeiffer, and S.-C. Liu. "Phased LSTM: Accelerating Recurrent Network Training for Long or Event-based Sequences". In: *Conf. Neural Inf. Process. Syst. (NIPS)*. 2016.

[129] R. A. Newcombe. "Dense Visual SLAM". PhD thesis. London, UK: Imperial College London, Dec. 2012.

[130] A. Nguyen, T. Do, D. G. Caldwell, and N. G. Tsagarakis. "Real-Time 6DOF Pose Relocalization for Event Cameras with Stacked Spatial LSTM Networks". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2019.

[131] Z. Ni, C. Pacoret, R. Benosman, S.-H. Ieng, and S. Régnier. "Asynchronous event-based high speed vision for microparticle tracking". In: *J. Microscopy* 245.3 (2012), pp. 236–244. DOI: 10.1111/j.1365-2818.2011.03565.x.

[132] G. Orchard, A. Jayawant, G. K. Cohen, and N. Thakor. "Converting Static Image Datasets to Spiking Neuromorphic Datasets Using Saccades". In: *Front. Neurosci.* 9 (2015), p. 437. DOI: 10.3389/fnins.2015.00437.

[133] G. Orchard, C. Meyer, R. Etienne-Cummings, C. Posch, N. Thakor, and R. Benosman. "HFirst: A Temporal Approach to Object Recognition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 37.10 (2015), pp. 2028–2040. DOI: 10.1109/TPAMI.2015.2392947.

[134] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. "Automatic differentiation in PyTorch". In: *NIPS Workshops*. 2017.

[135] A. Patron-Perez, S. Lovegrove, and G. Sibley. "A Spline-Based Trajectory Representation for Sensor Fusion and Rolling Shutter Cameras". In: *Int. J. Comput. Vis.* 113.3 (2015), pp. 208–219. DOI: 10.1007/s11263-015-0811-3.

[136] E. Piatkowska, A. N. Belbachir, and M. Gelautz. "Asynchronous Stereo Vision for Event-Driven Dynamic Stereo Sensor Using an Adaptive Cooperative Approach". In: *Int. Conf. Comput. Vis. Workshops (ICCVW)*. 2013, pp. 45–50. DOI: 10.1109/ICCVW.2013.13.

[137] E. Piatkowska, A. N. Belbachir, S. Schraml, and M. Gelautz. "Spatiotemporal multiple persons tracking using Dynamic Vision Sensor". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2012, pp. 35–40. DOI: 10.1109/CVPRW.2012.6238892.

[138] M. Pizzoli, C. Forster, and D. Scaramuzza. "REMODE: Probabilistic, Monocular Dense Reconstruction in Real Time". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 2609–2616. DOI: 10.1109/ICRA.2014.6907233.

[139] C. Posch, D. Matolin, and R. Wohlgenannt. "A QVGA 143 dB Dynamic Range Frame-Free PWM Image Sensor With Lossless Pixel-Level Video Compression and Time-Domain CDS". In: *IEEE J. Solid-State Circuits* 46.1 (Jan. 2011), pp. 259–275. DOI: 10.1109/JSSC.2010.2085952.

[140] C. Posch, D. Matolin, and R. Wohlgenannt. "A QVGA 143dB dynamic range asynchronous address-event PWM dynamic image sensor with lossless pixel-level video compression". In: *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*. 2010, pp. 400–401. DOI: 10.1109/ISSCC.2010.5433973.

[141] C. Poynton. *Chroma subsampling notation*. 2002. URL: http://vektor.theorem.ca/graphics/ycbcr/Chroma_subsampling_notation.pdf.

[142] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri. "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses". In: *Front. Neurosci.* 9 (2015), p. 141. DOI: 10.3389/fnins.2015.00141.

[143] T. Qin, P. Li, and S. Shen. "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator". In: *IEEE Trans. Robot.* 34.4 (2018), pp. 1004–1020. DOI: 10.1109/TRO.2018.2853729.

[144] W. Qiu, F. Zhong, Y. Zhang, S. Qiao, Z. Xiao, T. S. Kim, and Y. Wang. "UnrealCV: Virtual Worlds for Computer Vision". In: *Proc. ACM Int. Conf. Mult.* 2017, pp. 1221–1224. DOI: 10.1145/3123266.3129396.

[145] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. "ROS: an open-source Robot Operating System". In: *ICRA Workshop Open Source Softw.* Vol. 3. 2. 2009, p. 5.

[146] H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza. "EMVS: Event-based Multi-View Stereo—3D Reconstruction with an Event Camera in Real-Time". In: *Int. J. Comput. Vis.* 126.12 (Dec. 2018), pp. 1394–1414. DOI: 10.1007/s11263-017-1050-6.

[147] H. Rebecq, G. Gallego, and D. Scaramuzza. "EMVS: Event-based Multi-View Stereo". In: *British Mach. Vis. Conf. (BMVC)*. 2016. DOI: 10.5244/C.30.63.

[148] H. Rebecq, D. Gehrig, and D. Scaramuzza. "ESIM: an Open Event Camera Simulator". In: *Conf. on Robotics Learning (CoRL)*. 2018.

[149] H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization". In: *British Mach. Vis. Conf. (BMVC)*. 2017.

[150] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza. "EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-Time". In: *IEEE Robot. Autom. Lett.* 2.2 (2017), pp. 593–600. DOI: 10.1109/LRA.2016.2645143.

[151] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza. "EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-Time". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2018.

[152] H. Rebecq, T. Horstschäfer, G. Gallego, and D. Scaramuzza. "EVO: A Geometric Approach to Event-based 6-DOF Parallel Tracking and Mapping in Real-Time". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018.

[153] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. "Events-to-Video: Bringing Modern Computer Vision to Event Cameras". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019.

[154] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. "Events-To-Video: Real-Time Image Reconstruction With an Event Camera". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019.

[155] H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza. "High Speed and High Dynamic Range Video with an Event Camera". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2019). DOI: 10.1109/TPAMI.2019.2963386.

[156] H. Rebecq, A. Rosinol Vidal, T. Horstschäfer, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2018.

[157] H. Rebecq, A. Rosinol Vidal, T. Horstschäfer, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018.

[158] J. Redmon and A. Farhadi. "YOLOv3: An Incremental Improvement". In: *arXiv e-prints* abs/1804.02767 (2018). URL: http://arxiv.org/abs/1804.02767.

[159] *ReduceFlicker*. URL: http://avisynth.nl/index.php/ReduceFlicker.

[160] C. Reinbacher, G. Graber, and T. Pock. "Real-Time Intensity-Image Reconstruction for Event Cameras Using Manifold Regularisation". In: *British Mach. Vis. Conf. (BMVC)*. 2016. DOI: 10.5244/C.30.9.

[161] C. Reinbacher, G. Munda, and T. Pock. "Real-Time Panoramic Tracking for Event Cameras". In: *IEEE Int. Conf. Comput. Photography (ICCP)*. 2017, pp. 1–9. DOI: 10.1109/ICCPHOT.2017.7951488.

[162] P. Rogister, R. Benosman, S.-H. Ieng, P. Lichtsteiner, and T. Delbruck. "Asynchronous Event-Based Binocular Stereo Matching". In: *IEEE Trans. Neural Netw. Learn. Syst.* 23.2 (2012), pp. 347–353. DOI: 10.1109/TNNLS.2011.2180025.

[163] O. Ronneberger, P. Fischer, and T. Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2015.

[164] A. Rosinol Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza. "Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High Speed Scenarios". In: *IEEE Robot. Autom. Lett.* 3.2 (Apr. 2018), pp. 994–1001. DOI: 10.1109/LRA.2018.2793357.

[165] E. Rosten and T. Drummond. "Machine learning for high-speed corner detection". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2006, pp. 430–443. DOI: 10.1007/11744023_34.

[166] B. Rueckauer and T. Delbruck. "Evaluation of Event-Based Algorithms for Optical Flow with Ground-Truth from Inertial Measurement Sensor". In: *Front. Neurosci.* 10.176 (2016). DOI: 10.3389/fnins.2016.00176.

[167] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and F.-F. Li. "ImageNet Large Scale Visual Recognition Challenge". In: *Int. J. Comput. Vis.* 115.3 (Apr. 2015), pp. 211–252. DOI: 10.1007/s11263-015-0816-y.

[168] R. B. Rusu and S. Cousins. "3D is here: Point Cloud Library (PCL)". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2011.

[169] H. E. Ryu. *Industrial DVS Design; Key Features and Applications*. http://rpg.ifi.uzh.ch/docs/CVPR19workshop/CVPRW19_Eric_Ryu_Samsung.pdf. 2019.

[170] C. Scheerlinck, N. Barnes, and R. Mahony. "Asynchronous Spatial Image Convolutions for Event Cameras". In: *IEEE Robot. Autom. Lett.* 4.2 (Apr. 2019), pp. 816–822. DOI: 10.1109/lra.2019.2893427.

[171] C. Scheerlinck, N. Barnes, and R. Mahony. "Continuous-time Intensity Estimation Using Event Cameras". In: *Asian Conf. Comput. Vis. (ACCV)*. 2018.

[172] C. Scheerlinck, H. Rebecq, T. Stoffregen, N. Barnes, R. Mahony, and D. Scaramuzza. "CED: Color Event Camera Dataset". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2019.

[173] S. Schraml, A. N. Belbachir, and H. Bischof. "Event-driven stereo matching for real-time 3D panoramic vision". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2015, pp. 466–474. DOI: 10.1109/CVPR.2015.7298644.

[174] S. Schraml, A. N. Belbachir, N. Milosevic, and P. Schön. "Dynamic Stereo Vision System for Real-time Tracking". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2010, pp. 1409–1412. DOI: 10.1109/ISCAS.2010.5537289.

[175] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. "A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2006.

[176] R. Serrano-Gotarredona, M. Oster, P. Lichtsteiner, A. Linares-Barranco, R. Paz-Vicente, F. Gomez-Rodriguez, L. Camunas-Mesa, R. Berner, M. Rivas-Perez, T. Delbruck, S.-C. Liu, R. Douglas, P. Hafliger, G. Jimenez-Moreno, A. C. Ballcels, T. Serrano-Gotarredona, A. J. Acosta-Jimenez, and B. Linares-Barranco. "CAVIAR: A 45k Neuron, 5M Synapse, 12G Connects/s AER Hardware Sensory-Processing-Learning-Actuating System for High-Speed Visual Object Recognition and Tracking". In: *IEEE Trans. Neural Netw.* 20.9 (2009), pp. 1417–1438. DOI: 10.1109/TNN.2009.2023653.

[177] T. Serrano-Gotarredona and B. Linares-Barranco. "A 128x128 1.5% Contrast Sensitivity 0.9% FPN 3µs Latency 4mW Asynchronous Frame-Free Dynamic Vision Sensor Using Transimpedance Preamplifiers". In: *IEEE J. Solid-State Circuits* 48.3 (Mar. 2013), pp. 827–838. DOI: 10.1109/JSSC.2012.2230553.

[178] T. Serrano-Gotarredona and B. Linares-Barranco. "Poker-DVS and MNIST-DVS. Their History, How They Were Made, and Other Details". In: *Front. Neurosci.* 9 (Dec. 2015). DOI: 10.3389/fnins.2015.00481.

[179] S. Shah, D. Dey, C. Lovett, and A. Kapoor. "AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles". In: *Field and Service Robot.* 2017, pp. 621–635.

[180] P. A. Shedligeri, K. Shah, D. Kumar, and K. Mitra. "Photorealistic Image Reconstruction from Hybrid Intensity and Event based Sensor". In: *arXiv e-prints* (2018). URL: http://arxiv.org/abs/1805.06140.

[181] X. Shi, Z. Chen, H. Wang, D. Yeung, W. Wong, and W. Woo. "Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting". In: *Conf. Neural Inf. Process. Syst. (NIPS)*. 2015.

[182] S. B. Shrestha and G. Orchard. "SLAYER: Spike Layer Error Reassignment in Time". In: *Conf. Neural Inf. Process. Syst. (NIPS)*. Dec. 2018.

[183] K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: *Int. Conf. Learn. Representations (ICLR)*. 2015.

[184] A. Sironi, M. Brambilla, N. Bourdis, X. Lagorce, and R. Benosman. "HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 1731–1740.

[185] *Snapdragon Flight*. https://developer.qualcomm.com/hardware/snapdragon-flight.

[186] B. Son, Y. Suh, S. Kim, H. Jung, J.-S. Kim, C. Shin, K. Park, K. Lee, J. Park, J. Woo, Y. Roh, H. Lee, Y. Wang, I. Ovsiannikov, and H. Ryu. "A 640x480 dynamic vision sensor with a 9um pixel and 300Meps address-event representation". In: *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*. 2017. DOI: 10.1109/ISSCC.2017.7870263.

[187] H. Strasdat, J. Montiel, and A. Davison. "Real-time Monocular SLAM: Why Filter?" In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2010.

[188] R. Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. Springer, 2010. ISBN: 9781848829343.

[189] R. Szeliski and P. Golland. "Stereo Matching with Transparency and Matting". In: *Int. J. Comput. Vis.* 32.1 (1999), pp. 45–61. DOI: 10.1023/A:1008192912624.

[190] D. Tedaldi, G. Gallego, E. Mueggler, and D. Scaramuzza. "Feature Detection and Tracking with the Dynamic and Active-pixel Vision Sensor (DAVIS)". In: *Int. Conf. Event-Based Control, Comm. Signal Proc. (EBCCSP)*. 2016. DOI: 10.1109/EBCCSP.2016.7605086.

[191] A. Trémeau, S. Tominaga, and K. N. Plataniotis. "Color in Image and Video Processing: Most Recent Trends and Future Research Directions". In: *EURASIP Journal on Image and Video Processing* 2008 (2008), pp. 1–26. DOI: 10.1155/2008/581371.

[192] C. Troiani, A. Martinelli, C. Laugier, and D. Scaramuzza. "2-Point-based Outlier Rejection for Camera-IMU Systems with applications to Micro Aerial Vehicles". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014.

[193] V. Vasco, A. Glover, and C. Bartolozzi. "Fast event-based Harris corner detection exploiting the advantages of event-driven cameras". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2016. DOI: 10.1109/IROS.2016.7759610.

[194] G. Vogiatzis and C. Hernández. "Video-based, Real-Time Multi View Stereo". In: *Image Vis. Comput.* 29.7 (2011), pp. 434–441. DOI: 10.1016/j.imavis.2011.01.006.

[195] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE Trans. Image Process.* 13.4 (Apr. 2004), pp. 600–612. DOI: 10.1109/tip.2003.819861.

[196] D. Weikersdorfer, D. B. Adrian, D. Cremers, and J. Conradt. "Event-based 3D SLAM with a depth-augmented dynamic vision sensor". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2014, pp. 359–364. DOI: 10.1109/ICRA.2014.6906882.

[197] D. Weikersdorfer and J. Conradt. "Event-based Particle Filtering for Robot Self-Localization". In: *IEEE Int. Conf. Robot. Biomimetics (ROBIO)*. 2012, pp. 866–870. DOI: 10.1109/ROBIO.2012.6491077.

[198] D. Weikersdorfer, R. Hoffmann, and J. Conradt. "Simultaneous Localization and Mapping for event-based Vision Systems". In: *Int. Conf. Comput. Vis. Syst. (ICVS)*. 2013, pp. 133–142. DOI: 10.1007/978-3-642-39402-7_14.

[199] G. Wiesmann, S. Schraml, M. Litzenberger, A. N. Belbachir, M. Hofstatter, and C. Bartolozzi. "Event-driven embodied system for feature extraction and object recognition in robotic applications". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2012, pp. 76–82.

[200] G. Wolberg. *Digital Image Warping*. Wiley-IEEE Comp. Soc., 1990.

[201] S. Wozniak, A. Pantazi, and E. Eleftheriou. "Deep Networks Incorporating Spiking Neural Dynamics". In: *arXiv e-prints* (Dec. 2018). URL: http://arxiv.org/abs/1812.07040.

[202] Z. Xie, S. Chen, and G. Orchard. "Event-Based Stereo Depth Estimation Using Belief Propagation". In: *Front. Neurosci.* 11 (Oct. 2017). DOI: 10.3389/fnins.2017.00535.

[203] J. Xu, J. Zou, and Z. Gao. "Comment on Temperature and Parasitic Photocurrent Effects in Dynamic Vision Sensors". In: *IEEE Trans. Electron Devices* 65.7 (July 2018), pp. 3081–3082. DOI: 10.1109/TED.2018.2833106.

[204] C. Ye, A. Mitrokhin, C. Parameshwara, C. Fermüller, J. A. Yorke, and Y. Aloimonos. "Unsupervised Learning of Dense Optical Flow and Depth from Sparse Event Data". In: *arXiv e-prints* (2019). URL: http://arxiv.org/abs/1809.08625.

[205] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018.

[206] Z. Zhang. "A Flexible New Technique for Camera Calibration". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 22.11 (Nov. 2000), pp. 1330–1334. DOI: 10.1109/34.888718.

[207] Z. Zhang, H. Rebecq, C. Forster, and D. Scaramuzza. "Benefit of Large Field-of-View Cameras for Visual Odometry". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2016.

[208] Z. Zhang and D. Scaramuzza. "A Tutorial on Quantitative Trajectory Evaluation for Visual(-Inertial) Odometry". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2018.

[209] Y. Zhou, G. Gallego, H. Rebecq, L. Kneip, H. Li, and D. Scaramuzza. "Semi-Dense 3D Reconstruction with a Stereo Event Camera". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2018, pp. 242–258. DOI: 10.1007/978-3-030-01246-5_15.

[210] A. Z. Zhu, N. Atanasov, and K. Daniilidis. "Event-Based Feature Tracking with Probabilistic Data Association". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017, pp. 4465–4470. DOI: 10.1109/ICRA.2017.7989517.

[211] A. Z. Zhu, N. Atanasov, and K. Daniilidis. "Event-based Visual Inertial Odometry". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2017, pp. 5816–5824. DOI: 10.1109/CVPR.2017.616.

[212] A. Z. Zhu, D. Thakur, T. Ozaslan, B. Pfrommer, V. Kumar, and K. Daniilidis. "The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception". In: *IEEE Robot. Autom. Lett.* 3.3 (July 2018), pp. 2032–2039. DOI: 10.1109/lra.2018.2800793.

[213] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. "EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras". In: *Robotics: Science and Systems (RSS)*. 2018. DOI: 10.15607/RSS.2018.XIV.062.

[214] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. "Unsupervised Event-based Learning of Optical Flow, Depth, and Egomotion". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019.

[215] A. Z. Zhu, L. Yuan, K. Chaney, and K. Daniilidis. "Unsupervised Event-based Optical Flow using Motion Compensation". In: *Eur. Conf. Comput. Vis. Workshops (ECCVW)*. 2018.

Henri Rebecq

PhD in Computer Vision and Robotics

✉ *h.rebecq@gmail.com*
henri.rebecq.fr
3 January 1991

Academic

June 2015 - February 2020 **PhD Program in Computer Science**, *University of Zürich/ETH Zürich*.
Thesis: **Event Cameras: from SLAM to High Speed Video**. Advisor: Prof. Dr. Davide Scaramuzza.
A few selected publications:
- Learning High Speed and High Dynamic Range Video with an Event Camera.
  PDF, Video, Code
- Visual Odometry with Events, Frames and IMU. **Best Paper (Honorable Mention), RA-L'18**.
  Project page, PDF, Video 1, Video 2
- SLAM with an Event Camera.
  PDF, Video, US Patent
- 3D Reconstruction with an Event Camera. **Best Industry Paper, BMVC'16**.
  PDF, Video, Code

2013-2014 **M.Sc. MVA**, *École Normale Supérieure de Cachan*, Mathematics, Vision & Learning.
Received with highest distinction

2011-2014 **Télécom ParisTech**, *Paris*.
- Graduate school for applied mathematics and computer science engineering
- One of France's highly competitive engineering schools in the "Grandes Ecoles" system

2008-2011 **Classes Préparatoires**, *Lycée Aux Lazaristes*, Lyon, France.
Intensive preparatory course for competitive entrance into top French engineering schools

June 2008 **French scientific baccalaureate received with highest distinction**.
Equivalent to A level in Math, Physics and Chemistry

Professional Experience

August 2019 - November 2019 **Research Scientist Intern**, *Intel Labs*, Munich.
- Learning high speed and high dynamic range video reconstruction with an event camera (project page).
- My work lead to a paper at CVPR 2019, and a T-PAMI paper.

2017 - 2018 **Teaching Assistant**, *ETH Zurich*, Zurich, Vision Algorithms for Mobile Robotics.

April 2014 - June 2015 **Research Engineer**, *Orah*, Paris.
- Designed a full pipeline for self-calibration of a multiple wide-angle camera system based on video streams (C++/OpenCV). Highlighted as a key new feature of the software.

Awards

Georges Giralt PhD Award (Finalist), 2020.

Best Paper (Honorable Mention), *Robotics and Automation Letters (RA-L)*, 2018.
Awarded for my paper: *Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM*.

Qualcomm Innovation Fellowship, 2018, 40 000$.
Awarded for my proposal: *Learning Representations for Low-latency Perception with Frame and Event-based Cameras*.

Mischa Mahowald Prize for Neuromorphic Engineering, 2017, 3000$.
Awarded for *"pathbreaking applications of neuromorphic engineering to robot navigation"*.

Best Industry Paper, *British Machine Vision Conference (BMVC)*, 2016.
Awarded for my paper: *EMVS: Event-based Multi-view Stereo*.

People's Choice Prize & Technical Prize, *Final year project at Télécom ParisTech*, 2012.
Awarded for my project: *FLIP: an automated music page turner*.

Patents

H. Rebecq, G. Gallego, D. Scaramuzza, *Simultaneous Localization and Mapping with an Event Camera*, US 2019/0197715 A1, Issued on January 3, 2018. PDF.

H. Rebecq, T. Horstschaefer, D. Scaramuzza, *Visual-Inertial Odometry with an Event Camera*, EU 17189223.5 - 1906, Filed on November 6, 2017.

Publications

Journal Articles

H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "High speed and high dynamic range video with an event camera," *IEEE Trans. Pattern Anal. Machine Intell.*, 2019.

T. Rosinol Vidal*, **H. Rebecq***, T. Horstschaefer, and D. Scaramuzza, "Ultimate SLAM? combining events, images, and IMU for robust visual SLAM in HDR and high speed scenarios," *IEEE Robot. Autom. Lett.*, pp. 994–1001, 2018, **Best Paper award (Honorable Mention)**. (equal contribution).

H. Rebecq, G. Gallego, E. Mueggler, and D. Scaramuzza, "EMVS: Event-based multi-view stereo - 3D reconstruction with an event camera in real-time," *Int. J. Comput. Vis.*, pp. 1394–1414, 2018.

D. Gehrig, **H. Rebecq**, G. Gallego, and D. Scaramuzza, "EKLT: Asynchronous, photometric feature tracking using events and frames," *Int. J. Comput. Vis.*, 2019.

G. Gallego, J. E. A. Lund, E. Mueggler, **H. Rebecq**, T. Delbruck, and D. Scaramuzza, "Event-based, 6-DOF camera tracking from photometric depth maps," *IEEE Trans. Pattern Anal. Machine Intell.*, pp. 2402–2412, 2018.

H. Rebecq*, T. Horstschäfer*, G. Gallego, and D. Scaramuzza, "EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time," *IEEE Robot. Autom. Lett.*, pp. 593–600, 2017, (equal contribution).

E. Mueggler, **H. Rebecq**, G. Gallego, T. Delbruck, and D. Scaramuzza, "The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM," *Int. J. Robot. Research*, pp. 142–149, 2017.

E. Mueggler, G. Gallego, **H. Rebecq**, and D. Scaramuzza, "Continuous-time visual-inertial odometry for event cameras," *IEEE Trans. Robot.*, pp. 1425–1440, 2018.

Peer-Reviewed Conference papers

H. Rebecq, R. Ranftl, V. Koltun, and D. Scaramuzza, "Events-to-video: Bringing modern computer vision to event cameras," in *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2019.

C. Scheerlinck, **H. Rebecq**, T. Stoffregen, N. Barnes, R. Mahony, and D. Scaramuzza, "CED: color event camera dataset," in *IEEE Int. Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*, 2019.

J. Delmerico, T. Cieslewski, **H. Rebecq**, M. Faessler, and D. Scaramuzza, "Are we ready for autonomous drone racing? the UZH-FPV drone racing dataset," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

S. Bryner, G. Gallego, **H. Rebecq**, and D. Scaramuzza, "Event-based, direct camera tracking from a photometric 3D map using nonlinear optimization," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2019.

H. Rebecq, D. Gehrig, and D. Scaramuzza, "ESIM: an open event camera simulator," in *Conf. on Robotics Learning (CoRL)*, 2018.

D. Gehrig, **H. Rebecq**, G. Gallego, and D. Scaramuzza, "Asynchronous, photometric feature tracking using events and frames," in *Eur. Conf. Comput. Vis. (ECCV)*, 2018, *Oral presentation (acceptance rate: 2.4%)*.

Y. Zhou, G. Gallego, **H. Rebecq**, L. Kneip, H. Li, and D. Scaramuzza, "Semi-dense 3D reconstruction with a stereo event camera," in *Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 242–258.

G. Gallego, **H. Rebecq**, and D. Scaramuzza, "A unifying contrast maximization framework for event cameras, with applications to motion, depth, and optical flow estimation," in *IEEE Int. Conf. Comput. Vis. Pattern Recog. (CVPR)*, 2018, pp. 3867–3876.

H. Rebecq*, T. Horstschaefer*, and D. Scaramuzza, "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization," in *British Machine Vis. Conf. (BMVC)*, 2017, (equal contribution) *Oral presentation (acceptance rate: 5.6%)*.

H. Rebecq, G. Gallego, and D. Scaramuzza, "EMVS: Event-based multi-view stereo," in *British Machine Vis. Conf. (BMVC)*, 2016, **Best Industry Paper award.**

Z. Zhang, **H. Rebecq**, C. Forster, and D. Scaramuzza, "Benefit of large field-of-view cameras for visual odometry," in *IEEE Int. Conf. Robot. Autom. (ICRA)*, 2016.

Public Demos

Events-To-Video: Real-Time Image Reconstruction With an Event Camera, *CVPR*, 2019.

UltimateSLAM? Combining Events, Frames and IMU for Robust Visual SLAM, *ECCV & CVPR*, 2018.

EVO: Event-based 6-DOF Parallel Tracking and Mapping in Real-time, *ECCV & CVPR*, 2018.

Skills

Computer

Programming	C++, Python	Scientific	OpenCV, NumPy, CUDA
Machine Learning	PyTorch	Misc	HTML/CSS, LaTeX

Languages

French	**Native language**	
English	**Fluent**	*Cambridge CAE*
German	**Intermediate**	*Studied for 5 years in high school, 4 years casual speaking in Zurich*
Spanish	**Intermediate**	*Studied for 2 years, volunteer work in Mexico for 2 months*

Other Activities

Music	I have been playing the piano for 20 years (preferred genres: boogie-woogie and classical music).
Sport	Badminton.
Video	I am fond of videomontage & visual effects (using tools like Blender, Adobe Premiere, After Effects).