How to See with an Event Camera

Cedric Scheerlinck

A thesis submitted for the degree of
PhD
The Australian National University

January 2021

Except where otherwise indicated, this thesis is my own original work.

Cedric Scheerlinck
21 January 2021

To my family, for their love and support.

Acknowledgments

Many people helped make my PhD a reality. In particular, I would like to thank:

- Friends and colleagues at the Australian National University and the Australian Centre for Robotic Vision, for supporting me both in and out of the lab, and fostering a strong research community.
- Members of the Robotics and Perception Group, University of Zurich & ETH, led by Prof. Davide Scaramuzza, for welcoming me into the group for a 12 month visit and providing insightful discussion and collaboration.
- Folks from the Sensors Group, University of Zurich & ETH, led by Prof. Tobi Delbruck, and various spin-offs including Inivation and Insightness, that helped build and support the sensors.
- Fellow researchers Timo Stoffregen, Dr. Henri Rebecq, Liyuan Pan, Ziwei Wang, Daniel Gehrig and Prof. Guillermo Gallego for their ongoing collaboration and contributions to my publications and the chapters of this thesis.
- My supervisory panel members, Prof. Nick Barnes and Prof. Tom Drummond and especially my primary supervisor Prof. Robert Mahony, for their guidance, ideas and inspiration.
- Finally, my parents Jean-Pierre and Li Shuo, for their love and support, and for inspiring me to pursue a PhD.

This work was supported by an Australian Government Research Training Program Scholarship, the Australian Research Council through the "Australian Centre of Excellence for Robotic Vision" CE140100016 and the Swiss Government Excellence Scholarship program.

Abstract

Seeing enables us to recognise people and things, detect motion, perceive our 3D environment and more. Light stimulates our eyes, sending electrical impulses to the brain where we form an image and extract useful information. Computer vision aims to endow computers with the ability to interpret and understand visual information - an artificial analogue to human vision. Traditionally, images from a conventional camera are processed by algorithms designed to extract information. Event cameras are bio-inspired sensors that offer improvements over conventional cameras. They (i) are fast, (ii) can see dark and bright at the same time, (iii) have less motion-blur, (iv) use less energy and (v) transmit data efficiently. However, it is difficult for humans and computers alike to make sense of the raw output of event cameras, called *events*, because events look nothing like conventional images. This thesis presents novel techniques for extracting information from events via: (i) reconstructing images from events then processing the images using conventional computer vision and (ii) processing events directly to obtain desired information. To advance both fronts, a key goal is to develop a sophisticated understanding of event camera output including its noise properties. Chapters 3 and 4 present fast algorithms that process each event upon arrival to continuously reconstruct the latest image and extract information. Chapters 5 and 6 apply machine learning to event cameras, letting the computer learn from a large amount of data how to process event data to reconstruct video and estimate motion. I hope the algorithms presented in this thesis will take us one step closer to building intelligent systems that can *see* with event cameras.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Conventional video cameras see the world as a sequence of still images captured in rapid succession. An image is formed when a mechanical or electronic shutter opens to allow incoming light to hit the sensor array for an instance before closing again. Repeating this process, the visual scene is sampled - usually at a fixed rate - at discrete points in time. While ubiquitous, this method of visual sensing has several drawbacks: (i) the sampling rate is independent of scene dynamics, i.e., static scenes are repeatedly sampled at the same rate as high-speed scenes, (ii) motion-blur can occur while the shutter is open, (iii) uniform exposure across pixels can limit dynamic range, (iv) the camera is blind between consecutive images.

Event cameras are inspired by biological vision: they do not have a shutter and instead capture visual information continuously through time. Their pixels operate independently from one another and suppress redundant information by remaining silent unless a change in brightness is detected. Each pixel stores a reference level of brightness and continuously compares it with the current level. If the difference exceeds a preset 'contrast threshold', an event is triggered and sent off-chip, and the reference level is reset to the current level of brightness. The event is a packet of information containing the (x, y) address of the pixel, the polarity of the brightness change and a microsecond resolution timestamp. The final output of the camera is an asynchronous stream of events triggered by per-pixel changes in brightness.

An event stream contains spatiotemporal visual information, but looks nothing like a sequence of conventional images. One visualisation for an event stream is to plot a cloud of points (one per event) in 3D space-time with two spatial dimensions and a third temporal dimension (Fig. 2.3). The point cloud density (or sparsity) depends mainly on the scene, high-speed scenes are temporally dense and highly-textured scenes are spatially dense. Spatial resolution and responsiveness of the sensor also impact point cloud density. However, storing and using the

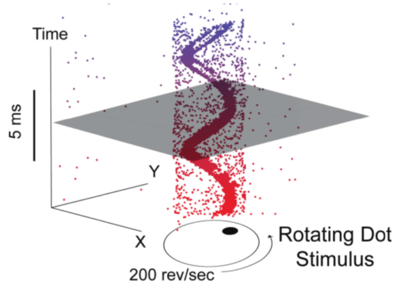


Figure 1.1: Spinning dot forms spiral of events in space-time. From [RPG publications, 2019].

entire event stream for downstream applications requires complete redesign of computer vision algorithms and can be computationally intractable - especially for time horizons larger than a few seconds.

Reconstructing images from events gives a compact representation of the latest available data compared to a full history of events and enables application of conventional computer vision to event cameras. Unlike raw events, images are naturally human-interpretable and provide insight into the information contained in events. Image reconstruction also allows us to assess event data, e.g., judging the amount of noise in an event stream by analysing the images. Solving the task of image reconstruction requires an understanding of how to deal with event camera noise, and overcoming these challenges provides insight that can be applied to other tasks such as event-based feature detection or optic flow.

Two distinct processing paradigms have arisen from the event camera research community, (i) asynchronous 'event-by-event' processing aimed at minimising latency and computational cost and (ii) synchronous 'batch' processing aimed at producing the highest quality output from a temporal window of events. Each have distinct advantages and challenges, and I will explore both perspectives in this thesis.

Asynchronous algorithms process each event upon arrival, matching the natural asynchronous output format of event cameras, leveraging low latency and sparsity of event data. Asynchronous processing is particularly promising where it can be realised on specialised hardware such as field programmable gate arrays (FPGAs), intelligent processing units (IPUs) or spiking neural network chips such as Intel's Loihi that are amenable to power efficient, low latency asynchronous operations. Designing and testing algorithms on conventional computers is an effective way to research asynchronous algorithms that may one day be implemented on specialised hardware. A challenge to designing asynchronous event-based algorithms is reconciling the discrete/continuous duality of events since events are discrete, though can occur in near continuous time. Another challenge is restricting computation to a local salient spatiotemporal domain to extract information locally and asynchronously rather than densely processing the entire image frame. Overcoming these challenges is key to unlocking the potential of event-based asynchronous sensors and algorithms.

Batch processing typically considers the last N events or Δt seconds of event data, truncating any prior events to avoid using 'out of date' information. Batching incurs latency in return for possibly (i) higher quality results and (ii) easier to design algorithms. At the time of writing, state-of-the-art image reconstruction, classification and optic flow use convolutional neural networks that process small batches of events at a time. Targeted CNN chips that are presently under development around the world potentially address the computational cost and latency associated with CNNs. However, even dedicated CNN chips will have limitations and there is critical need to understand how to design lightweight efficient networks. In particular, two key challenges are (i) reducing computational cost and (ii) effective training that generalises to the real world. The size, or number of parameters in a neural network architecture is closely related to the computational complexity of the network, and reducing the size is one way to improve computational efficiency. Training is another key challenge, with proposed solutions including self/un-supervised learning, generative adversarial networks, and supervised learning from simulated data. Using a simulator to generate training data is attractive because it provides unlimited groundtruth labelled data. Learning from simulated data requires understanding of the event generation and noise processes of real event cameras to accurately simulate training data that will help the network generalise to real data (sim-to-real transfer), an important condition to ensure reliable and safe performance of neural networks in the real world.

1.2 Thesis Outline

This thesis tackles four key challenges:

- Asynchronous image reconstruction
- Asynchronous convolution and feature detection
- Batch image reconstruction using convolutional neural networks
- Reducing the sim-to-real gap for event camera learning

Chapter 2 provides a general background and related works. Chapter 3 presents a framework for asynchronous, continuous-time image reconstruction. The key idea is to maintain a continuous-time image state, updated with each event upon arrival using a linear complementary filter. The complementary filter structure inherently combines (temporal) low and high frequency signals, and can fuse low frequency image frames from a video camera with high frequency events from an event camera. Chapter 4 applies the linear filtering ideas to asynchronous spatial image convolution. Instead of an intensity image state, a convolved image state (e.g., image gradient state) is reconstructed, allowing algorithms that require convolution, e.g., corner detection, to be implemented asynchronously for event cameras. Chapter 5 shifts the focus from low latency, asynchronous processing to batch processing with convolutional neural networks. The aim is to reduce the computational cost of neural networks for image reconstruction by finding a lightweight architecture with few parameters that runs fast. Chapter 6 addresses the sim-to-real gap for networks trained on simulated event data. Analysing major event camera datasets reveals insight into the nature of event data, ultimately improving sim-to-real transfer, and the lessons learnt can be applied to several tasks including image reconstruction and optic flow.

1.2.1 Collaborations

Chapters 3 and 4 are based on work done at the Australian National University (ANU) with Prof. Robert Mahony and Prof. Nick Barnes. Open source code associated with chapter 3 was developed while I was visiting the Robotics and Perceptions Group (RPG) led by Prof. Davide Scaramuzza at the University of Zurich. The color event camera mentioned in section 3.3.3 was developed by Inivation and the Sensors Group led by Prof. Tobi Delbruck, and kindly provided to us by Inivation. Chapter 5 is based on work done at the RPG in collaboration with Dr. Henri Rebecq and Daniel Gehrig. Throughout my 12 month stay with the RPG, Prof. Guillermo Gallego acted as a mentor, contributing to almost every project I was involved in. Chapter 6 is based on an equal first author collaboration with Timo Stoffregen across the ANU, Monash University and the University of Zurich.

1.3 Contributions

1.3.1 Journal papers

- C. Scheerlinck, N. Barnes, R. Mahony, "Asynchronous Spatial Image Convolutions for Event Cameras", IEEE Robotics and Automation Letters, 4(2), April 2019, pp. 816-822.
- L. Pan, R. Hartley, C. Scheerlinck, M. Liu, X. Yu, Y. Dao, "High Frame Rate Video Reconstruction based on an Event Camera", IEEE Transactions on Pattern Analysis and Machine Intelligence, November 2020.

1.3.2 Conference papers

- T. Stoffregen*, C. Scheerlinck*, D. Scaramuzza, T. Drummond, N. Barnes, L. Kleeman, R. Mahony, "Reducing the Sim-to-Real Gap for Event Cameras", European Conference on Computer Vision (ECCV), 2020.
- C. Scheerlinck, H. Rebecq, D. Gehrig, N. Barnes, R. Mahony, D. Scaramuzza, "Fast Image Reconstruction with an Event Camera", Winter Conference on Applications of Computer Vision (WACV), 2020.
- C. Scheerlinck*, H. Rebecq*, T. Stoffregen, N. Barnes, R. Mahony, D. Scaramuzza, "CED: Color Event Camera Dataset", Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019.
- L. Pan, C. Scheerlinck, X. Yu, R. Hartley, M. Liu, Y. Dao, "Bringing a Blurry Frame Alive at High Frame-Rate with an Event Camera", Conference on Computer Vision and Pattern Recognition (CVPR), 2019.
- C. Scheerlinck, N. Barnes, R. Mahony, "Asynchronous Spatial Image Convolutions for Event Cameras", IEEE International Conference on Robotics and Automation Letters (ICRA), 2019.
- C. Scheerlinck, N. Barnes, R. Mahony, "Continuous-time Intensity Estimation Using Event Cameras", Asian Conference on Computer Vision (ACCV), Perth, 2018, pp. 308-324.

*Equal contribution.

1.3.3 Miscellaneous Contributions

- Created the Event Camera Wikipedia page:
  https://en.wikipedia.org/wiki/Event_camera
- Z. Wang, Y. Ng, C. Scheerlinck, R. Mahony, "An Asynchronous Kalman Filter for Hybrid Event Cameras", arXiv 2020.
- High Quality Frames event camera dataset and sim-to-real learning code
  https://cedricscheerlinck.com/20ecnn
- High Speed and High Dynamic Range event camera dataset
  http://rpg.ifi.uzh.ch/E2VID.html
- CED: Color Event Camera Dataset:
  http://rpg.ifi.uzh.ch/CED.html
- DVS Image Reconstruction code and dataset:
  https://cedricscheerlinck.com/continuous-time-intensity-estimation

Chapter 2

Background and Related Work

This chapter aims to motivate event cameras and bring the reader up to speed on current event camera hardware. It then reviews existing algorithms that can be used to process the output of event cameras.

Section 2.1 explains the philosophy behind event cameras and the advantages they promise over conventional cameras.

Section 2.2 explains existing event camera technologies.

Sections 2.3 and 2.4 review existing event-based algorithms and techniques to process event data.

2.1 Motivation

The event camera was born in the field of neuromorphic engineering that aims to tackle an ambitious challenge: understanding how the brain works and building one on a chip [Gallego et al., 2020a]. Event cameras are inspired by biological retinas that have been optimised over millions of years by natural selection, and hope to

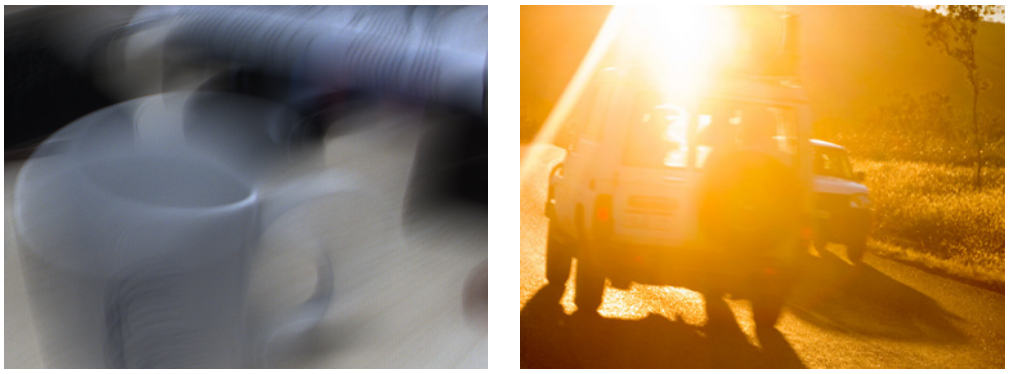

(a) Motion blur (b) Over-exposure

Figure 2.1: Limitations of conventional frame-based cameras. From [Bardow, 2018].

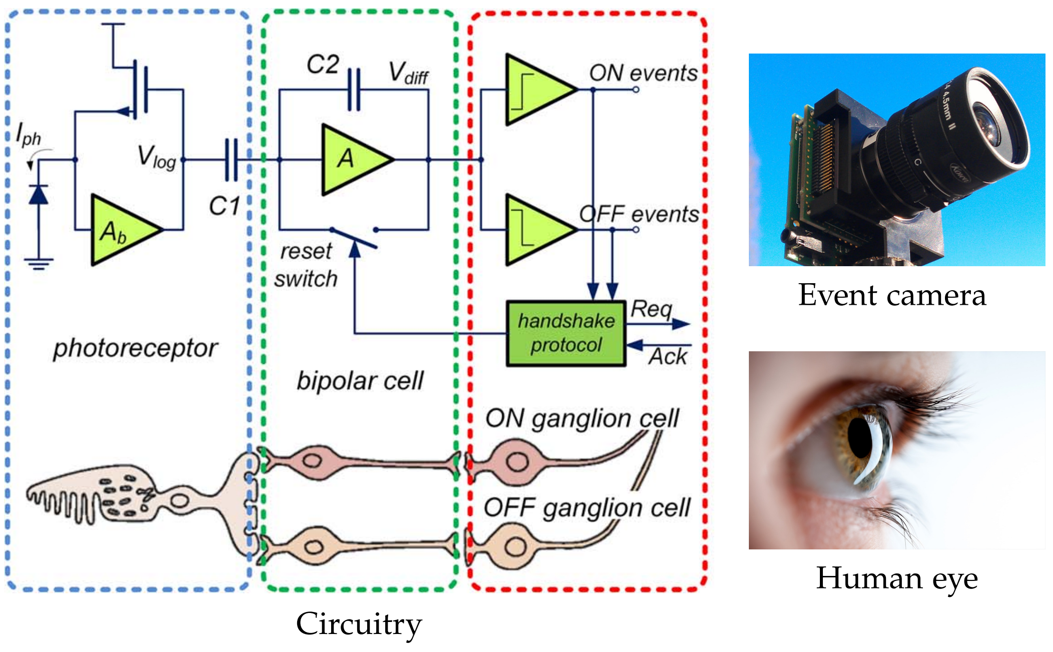


Figure 2.2: Circuitry of an event camera pixel (top) is directly inspired by biological retinas (bottom). Adapted from [Posch et al., 2014].

reap the benefits such as low power consumption, efficient information exchange and low-latency motion sensitivity. They are a radically different sensing paradigm from conventional frame-based cameras, potentially alleviating problems such as low temporal resolution, limited dynamic range (over/underexposure) and motion blur (Fig. 2.1). The core building block is a smart pixel that is activated by changes in light intensity and quiet otherwise, analogous to a transient ganglion cell in the retina of a biological eye (Fig. 2.2). Each pixel operates asynchronously and independently, with the net effect being a camera that sees dynamic (changing) scenes and is 'blind' to unchanging scenes, a *dynamic vision sensor*.

2.2 Event Cameras

Figure 2.3 summarises the working principle of the popular *Dynamic and Active-pixel Vision Sensor* (DAVIS) [Brandli et al., 2014a] event camera. Each pixel contains circuitry that allows (i) active pixel sensor (APS) intensity readout and (ii) dynamic vision sensor (DVS) event generation from the same photoreceptor. The APS is a standard global shutter camera that operates independently of the DVS. The DVS comprises of a capacitor that stores information about the *change* in log intensity, and an amplifier that leads to two comparators that check whether the change exceeds a preset *contrast threshold*. If the change is positive (i.e., the brightness increased), an ON event is generated, otherwise an OFF event. An event contains polarity (ON/OFF), pixel address (x, y) and a microsecond timestamp. When an event is generated, a reset switch drains the capacitor, resetting the change level to zero. A

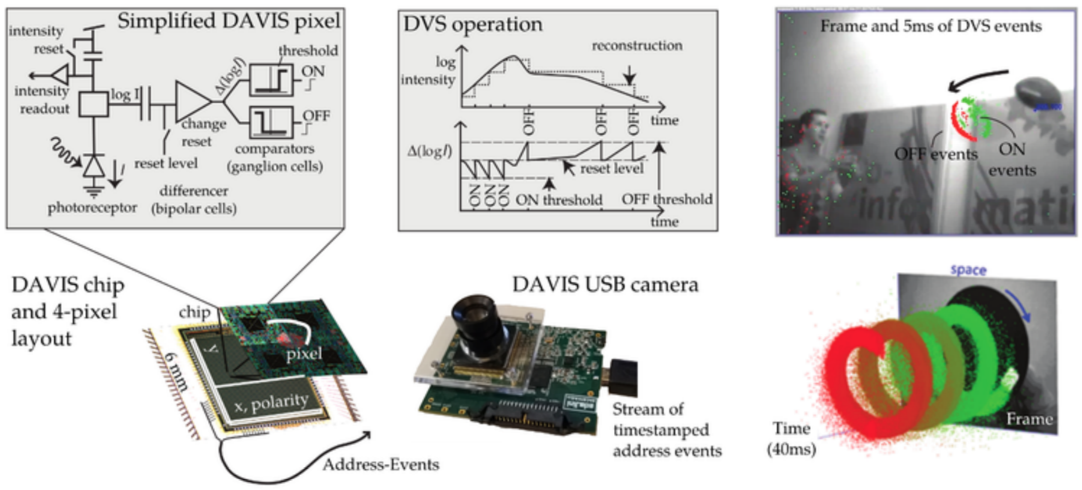


Figure 2.3: Top: simplified circuitry of a single DAVIS [Brandli et al., 2014a] pixel and event (ON/OFF) generation given a log intensity signal. Bottom: DAVIS chip and DAVIS USB camera. Right: events drawn on an intensity image (green ON, red OFF) and a 3D event point cloud. From [Gallego et al., 2020a].

refractory period is imposed on every pixel that silences it after every event for a short duration (e.g. 1ms) to reduce bandwidth. Pixels that generate an event send a request to an arbiter responsible for reading out events. The chip operates on a 1MHz clock, allowing low latency event generation and readout, especially compared to standard camera frame rates on the order of 100Hz.

Typical DVS response to a given log intensity signal at one pixel is shown in figure 2.3 (top right). Accumulating ON and OFF events (summing) yields a quantised reconstruction of the original signal. In contrast to conventional cameras, the quantisation is along the intensity dimension instead of the time dimension. Thus, the rate of events is scene dependent, adapting according to the rate and magnitude of change in brightness. Concretely, large changes trigger more events, and no change triggers no events. Scene-dependent output is one key difference between event and conventional fixed-rate cameras, the philosophy being that unchanging regions of the scene do not need to be sampled as often as dynamic regions, hence the name: *Dynamic Vision Sensor*.

Events encode visual information about dynamic parts of the scene, and figure 2.3 (bottom) illustrates this by plotting events in 3D space-time. For example, a spinning disc forms a spiral of events in space-time, and moving objects trigger events at their boundaries. Importantly, event cameras are blind to stationary (unchanging) parts of a scene, partly motivating the inclusion of a standard APS camera in the DAVIS to complement events. More strictly, event cameras are blind to unchanging scenes, including textureless moving objects, and can see static scenes if the sensor is stimulated, for example by a strobe light.

Event cameras have several known nonidealities and characteristic noise [Lichtsteiner et al., 2008; Brandli et al., 2014a; Delbruck et al., 2020] including: (i) mismatch of contrast threshold between pixels, (ii) bandwidth limitations (iii) hot pixels that fire many spurious events, (iv) refractory deadzone after an event firing, (v) random background noise events. In addition, cameras such as the DAVIS that share circuitry to produce image frames can generate noise events (in $\sim$0.25% of pixels under uniform, unchanging illumination) upon every frame acquisition due to undesirable parasitic coupling between APS (frame) and DVS (event) pixel circuitry [Brandli et al., 2014a]. The effective contrast threshold has been shown to vary spatially between pixels, and temporally for a given pixel [Lichtsteiner et al., 2008; Brandli et al., 2014b], rendering direct integration of the event signal ineffective without noise suppression and motivating online calibration to compensate temporal fluctuations in contrast threshold. The minimum contrast threshold for a typical event camera is $\sim$11% change in log-intensity [Brandli et al., 2014a], though sensitive cameras can achieve down to 3.5% for ON events and 1% for OFF events [Moeys et al., 2018]. Lower contrast thresholds improve fidelity and at the cost of elevated bandwidth since more events are emitted for a given intensity change, while higher contrast thresholds will fail to trigger events for small changes but save bandwidth. Event camera bandwidth is a monotonically increasing function of light intensity at about 3 kHz in bright conditions and 300 Hz at 1000$\times$ lower intensity [Gallego et al., 2020b]. High bandwidth sensors such as the Samsung DVS-Gen4 [Suh et al., 2020] and Prophesee Gen 4 CD [Finateu et al., 2020] have a maximum bandwidth of over 1 billion events per second. Hot pixels are characterised by spurious firing of many events in rapid succession, either continuously, in random bursts, or in overreaction to changing illumination, consuming valuable bandwidth. A refractory period at each pixel following an event firing can dampen hot pixels and relieve bus congestion, but introduces error in tracking fast brightness changes by suppressing events that 'should have' fired. A constant trickle of random background noise events are caused by leakage in the reset transistor in DVS pixel circuitry and are typically uncorrelated, thus amenable to filtering by discarding spatio-temporally isolated events [Delbruck, 2008].

2.3 Asynchronous Versus Batch Processing

Event camera algorithms today cover a diverse range of topics from image reconstruction to optic flow, feature detection, spiking neural networks, SLAM, stereo matching, recognition, control and more. A comprehensive review of all major topics is outside the scope of this thesis, however, readers may find more information in [Gallego et al., 2020a; Event-based Vision Resources, 2017]. Two distinct processing paradigms have arisen from the event camera research community:

- asynchronous 'event-by-event' processing,
- synchronous 'batch' processing.

Asynchronous algorithms process each event upon arrival, incorporating the additional information gained from that event into an information state, whether that be

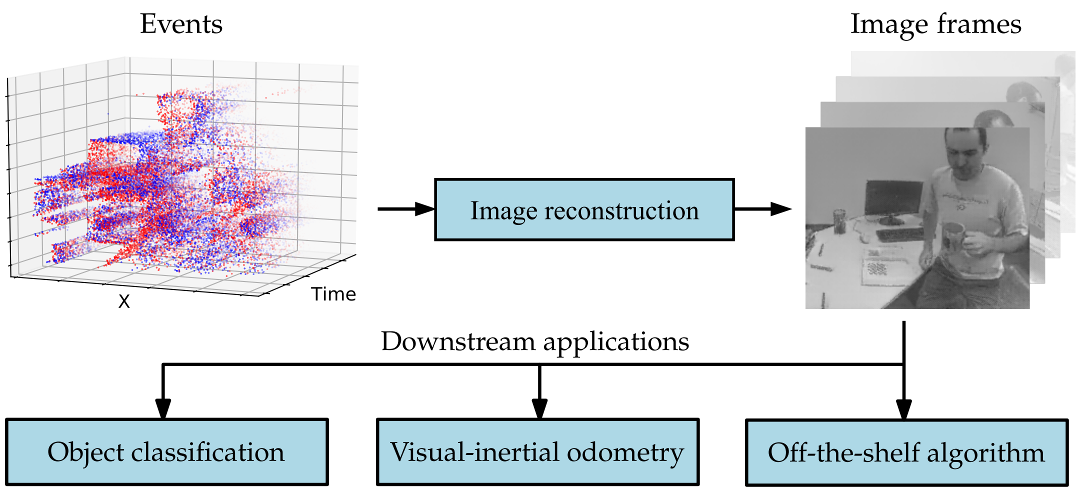


Figure 2.4: Image reconstruction is a bridge from events to conventional computer vision. From [Rebecq et al., 2019]

a reconstructed image state or a neuron membrane potential in a neural network. Since a single event alone contains little information, asynchronous algorithms rely on a state that captures relevant information from previous events, providing context for new events. Thus, to function properly, asynchronous algorithms require initialisation, e.g., from an image or by waiting for enough events to accumulate. Batch processing is a paradigm that waits for a temporal window of events to accumulate into a batch before processing the batch in one go, rather than processing each individual event upon arrival. A sliding temporal window may be used, though in practise most algorithms consider non-overlapping successive temporal windows to save computational resources and avoid processing events more than once. Batch algorithms may also maintain an internal state to extend the temporal context to outside of the current batch, i.e., to remember information from previous events and computation. In the following sections we will focus our attention on both asynchronous and batch processing techniques for image reconstruction, convolution and optic flow. Sections 2.3.1 and 2.3.2 review asynchronous and batch processing techniques for event cameras with a focus on video reconstruction. Section 2.4 reviews event-based optic flow methods and section 2.5 introduces related work on color event cameras.

2.3.1 Asynchronous Processing

Intensity image or video reconstruction is an important topic allowing human interpretable visualisation of events and a bridge to downstream processing with conventional computer vision tools (Fig. 2.4). Asynchronous image reconstruction algorithms are able to process each event as it arrives, though in practice typically output images at a much lower rate than the event rate, e.g., 100Hz vs. 1MHz. Brandli et al. [2014b] proposed a simple method to reconstruct images in the blind time be-

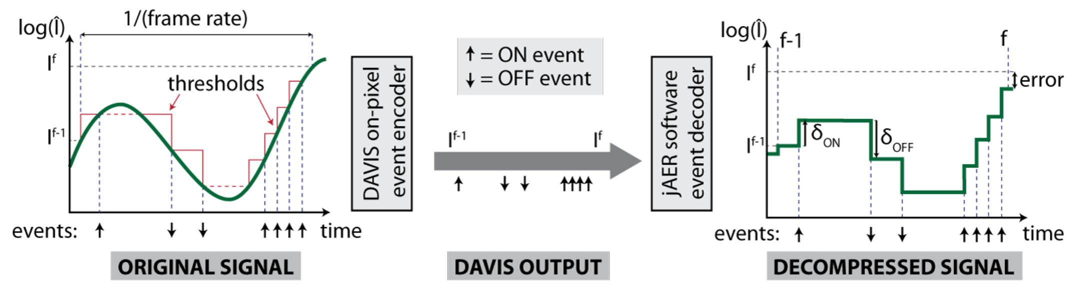


Figure 2.5: Relationship between original light intensity signal at a pixel (left, green), events (middle) and reconstructed quantised intensity image (right), based on estimating contrast thresholds (right; δ_{ON}, δ_{OFF} and left; red). From [Brandli et al., 2014b].

tween frames by adding events, weighted by their contrast threshold, to a DAVIS [Brandli et al., 2014a] image frame. The contrast threshold is estimated by matching the sum of events with the difference between two frames (Fig. 2.5), a concept further explored by Wang et al. [2019]. While a uniform global threshold is assumed between pixels, ON/OFF thresholds are decoupled and may differ. This method is computationally simple (as fast as $O(n)$ for n events) and easily runs in real-time on a CPU. Reinbacher et al. [2016] reconstruct images by integrating events with fixed global ON/OFF contrast threshold estimates with periodic regularisation to prevent noise build-up. They propose regularising on a manifold (Fig. 2.6) defined by the timestamp of the latest event at each pixel, a.k.a., Surface of Active Events [Benosman et al., 2014] or time surface. A wall clock time of 1.7ms per image (580fps) for 128×128 images was measured on a 3.4 GHz processor + NVIDIA Titan X GPU. The authors release open-source C++ code[1]. Scheerlinck et al. [2018] propose using a complementary filter to asynchronously reconstruct an intensity image state from events and can optionally incorporate information from image frames if available. The complementary filter performs temporal smoothing but no spatial smoothing, making it computationally efficient and the one of the fastest image reconstruction techniques, able to process up to 20M events per second on an i7 CPU.

Convolution is a fundamental image processing operation, used in feature detection (e.g., Harris corners [Harris and Stephens, 1988], SIFT [Lowe, 2004]), convolutional neural networks and more. Asynchronous convolution aims to reduce computational cost by exploiting sparsity of the input, only updating or convolving local neighbourhoods surrounding events instead of the whole image. An equivalent alternative interpretation is that asynchronous convolutions skip zero-entries in the full resolution input. Significant effort has gone into designing hardware-level implementation of asynchronous convolution, notably the “Convolution Address-Event-Representation (AER) Vision Architecture for Real-Time” (CAVIAR) project [Serrano-Gotarredona et al., 2009] and related work [Nageswaran et al., 2009; Perez-Carrasco et al., 2013; Linares-Barranco et al., 2019]. Recently, some works have ex-

[1] https://github.com/VLOGroup/dvs-reconstruction

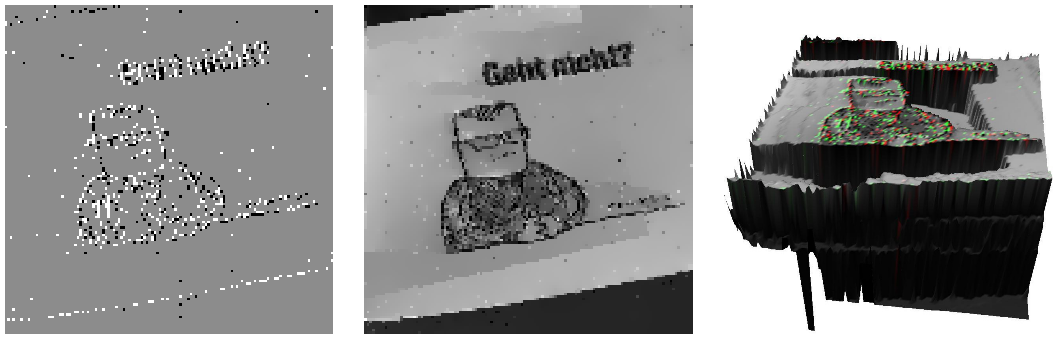
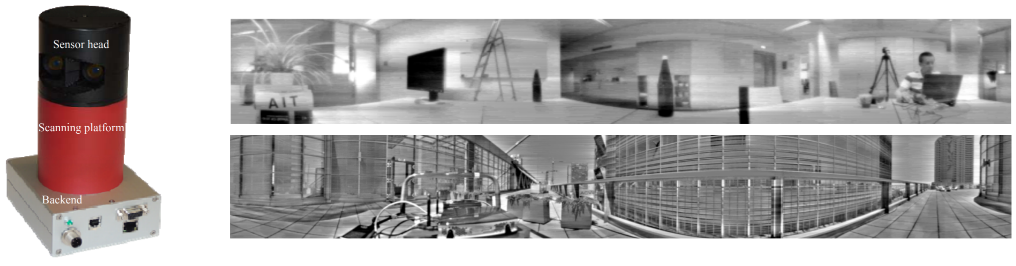

(a) Events (b) Reconstruction (c) Manifold

Figure 2.6: Image (b) reconstructed from events (a) based on regularisation on a time surface manifold (c). From [Reinbacher et al., 2016].

plored (software) implementation of asynchronous convolutions for event cameras. Scheerlinck et al. [2019a] show that asynchronous convolution can be combined with a complementary filter to reconstruct a 'convolved' image state e.g., image gradient. They take it a step further by asynchronously applying a Harris corner detector to a reconstructed image gradient state to obtain a continuous-time corner response state. Cannici et al. [2019]; Messikommer et al. [2020] apply asynchronous convolutions to neural networks, demonstrating reduced computational cost vs. conventional convolutions.

Spiking neural networks are a bio-inspired framework that simulate individual neurons as nodes that have a state (membrane potential) and can send and receive signals (spikes) asynchronously from connected nodes. In a leaky integrate and fire model (LIF) [Stein, 1965], the neuron state continuously decays (leaky), incoming spikes contribute to the state (integrate), and if the state value exceeds some threshold, the neuron sends out a spike and resets (fire). LIF is prevalent in event camera spiking neural network works for optical flow [Orchard et al., 2013; Haessig et al., 2018; Paredes-Valles et al., 2019], recognition [Orchard et al., 2015; Zhao et al., 2015; Negri et al., 2018; Xiao et al., 2019] and more [Perez-Carrasco et al., 2013; Osswald et al., 2017; Dikov et al., 2017; Haessig et al., 2018]. Parameters such as the decay rate, threshold and weights between connected nodes can be preset or learnt e.g., using spike timing dependent plasticity [Caporale and Dan, 2008]. The SpiNNaker project [Furber et al., 2014] has made considerable progress implementing spiking neural networks in hardware. Since event cameras are spike-based, spiking neural networks are a natural choice for processing event data with promises of low latency and low power consumption. However, they require specialised hardware to fully realise benefits such as low power consumption, are difficult to train and are typically less accurate than conventional neural networks.



(a) Rotating 1D event cameras

(b) Reconstructed panoramas

Figure 2.7: (a) A pair of 1D event cameras mounted on a scanning platform and (b) reconstructed grayscale panoramas. Adapted from [Belbachir et al., 2014].

2.3.2 Batch Processing

Batch image reconstruction aims to reconstruct images or video by considering a batch of events rather than each event independently. Cook et al. [2011] propose to set up a network of weakly interacting maps that converge to coherent estimates of pure camera rotation, optic flow, spatial gradient and image, driven by the intensity change measured by event cameras. Researchers have since proposed probabilistic filters [Kim et al., 2014, 2016] and geometric approaches [Rebecq et al., 2017] for event-based SLAM, capable of generating intensity image maps. An alternative approach is to rotate the event camera around a single axis to reconstruct a 360° panorama. Belbachir et al. [2014] built a rig with a pair of 1024-pixel 1D line event cameras, mounted on a 1-10Hz rotating platform (Fig. 2.7). They were able to demonstrate grayscale panorama reconstructions by integrating events together with high-pass filtering and contrast threshold estimation by matching the first and last intensity value after each 360° scan. Since stereo was available, they were also able to recover depth. Real-time performance at 10Hz for full 360° panoramic image and depth is possible on embedded compute (FPGA). Bardow et al. [2016] formulates a cost function based on brightness constancy [Horn and Schunck, 1981], smoothness and intensity change measured by event cameras. Optimising the cost over a batch of events yields intensity and optic flow estimates (Fig. 2.9). Shedligeri et al. [2018] temporally interpolate video by estimating depth and ego-motion to warp (static scene) image frames to intermediate locations. Pan et al. [2019] proposed the *event double integral* model, and a framework for reconstructing video from events based on (double) integrating from a starting image. A key component is the contrast threshold estimation algorithm that optimises for image edge sharpness and total variation.

The latest state-of-the-art image and video reconstruction methods at the time of writing are learning based, utilising methods such as dictionary learning, sparse coding, generative adversarial networks (GANs) and recurrent convolutional neural networks. Barua et al. [2016] use a learnt patch-based dictionary from simulated event data to reconstruct image gradient that can be upgraded to intensity via Poisson in-

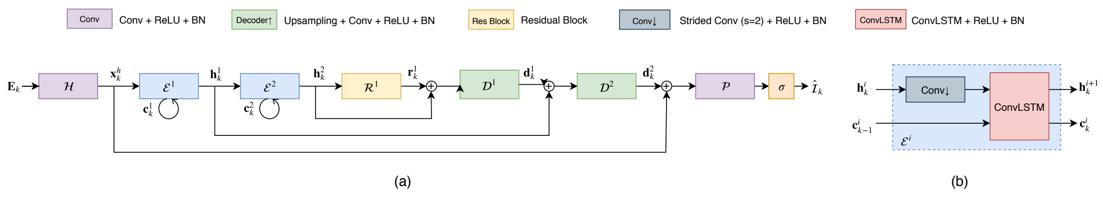


Figure 2.8: E2VID network architecture. $\mathbf{E}_k$: Event voxel (input), Conv: fully convolutional layer, BN: batch normalisation, σ: sigmoid activation, $\bigoplus$: channel-wise sum, $\hat{\mathcal{I}}_k$: output image. From [Rebecq et al., 2020a].

tegration [Agrawal et al., 2005, 2006]. Another approach is to simulate anatomically realistic retinal models [Watkins et al., 2018], trained with methods such as Locally Competitive Algorithm for sparse coding [Rozell et al., 2008]. GANs [Goodfellow et al., 2014] are a learning framework based on training a generator *G* and a discriminator *D* that estimates the probability that a sample came from training data rather than *G*. GANs can be applied to event cameras [Bardow, 2018; Mostafavi I. et al., 2019] by asking *G* to generate images given an event input and asking *D* to discriminate between real and generated images. One advantage of GANs is that ground truth correspondence between events and images is not required, only a dataset of real images and event data, since the (adversarial) loss from *D* can be used to train *G*.

Rebecq et al. [2019, 2020b] demonstrate superior results (Fig. 2.4) using a recurrent convolutional neural network called 'E2VID' to learn supervised end-to-end video reconstruction from simulated event data. The input to the network is a sequence of event voxels[2] or tensors: 3D space-time grids that contains events, with N temporal bins and width, height given by the image sensor. E2VID is based on recurrent UNet [Ronneberger et al., 2015; Alom et al., 2018], and is fully convolutional with LSTM units after every encoder, residual units in the bottleneck, skip sum connections between encoders and decoders, ReLU activations between all hidden layers and finally a sigmoid activation at the end (Fig. 2.8). E2VID is trained using two losses against groundtruth target images from the simulator, (i) learned perceptual image patch similarity (LPIPS) loss [Zhang et al., 2018] that minimises the perceptual distance between prediction and target, specifically the weighted L1 distance between hidden layers of a pretrained classification network, and (ii) temporal consistency loss [Lai et al., 2018] that minimises the photometric distance between consecutive images warped on top of each other using optic flow, subject to an occlusion mask. Key factors driving performance in [Rebecq et al., 2020b] are (i) LPIPS loss, (ii) clean simulated training data, (iii) temporal consistency loss and (iv) recurrent units in the network architecture. Scheerlinck et al. [2020] propose a lightweight network architecture that achieves similar accuracy to E2VID while using 99.4% fewer parameters, $10\times$ less floating point operations and running $3\times$ faster on the same GPU. Stoffregen et al. [2020] analyse several event camera datasets [Mueggler et al., 2017b; Zhu et al., 2018a] to generate more realistic, noise augmented, synthetic training data,

[2]first proposed by Zhu et al. [2019]. See also [Gehrig et al., 2019a].

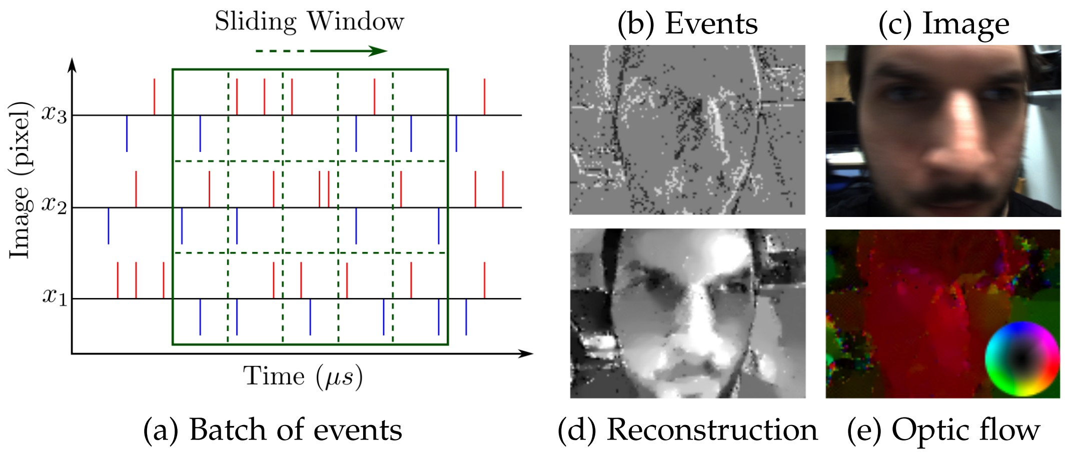


Figure 2.9: Joint optimisation of intensity (d) and optic flow (e) from a batch of events (a), along with raw events (b) and reference RGB image (c). Adapted from [Bardow, 2018].

showing up to 25% improvement when E2VID is retrained on realistic data.

2.4 Optic Flow

Optical flow is the apparent motion of light patterns, typically expressed as a 2D motion field on the image plane. Optic flow is classically defined as the pixel displacement between two image frames (pix), though event camera researchers typically estimate instantaneous flow (pix/s), sometimes called *visual flow*, because event cameras are inherently frame-free. In the sequel I will refer to instantaneous flow simply as optic flow. Event cameras appear well suited to observe optic flow since they respond to local brightness changes typically caused by motion. However, extracting optic flow from raw event data is (surprisingly) non-trivial, testified by the volume of research effort and literature produced. The key challenge appears to be the aperture problem, since events are inherently pixel-wise, thus cannot disambiguate direction when considered in isolation. Many works focus on how to group events appropriately both spatially and temporally into a representation amenable to extracting optic flow. In practical applications, optical flow is typically induced by relative motion between the scene and the camera, and obtaining optical flow can be helpful for: dynamic object tracking, pose estimation, visual odometry, SLAM, robot control, collision avoidance, VTOL landing. Optical flow provides dense correspondence between two images, useful for stereo matching, structure from motion, video interpolation. Additionally, researchers have found optical flow to improve action recognition and other tasks when used as (additional) input to neural networks [Zhou et al., 2019]. Further analysis of event-based optic flow can be found in [Rueckauer and Delbruck, 2016].

Recent hand-crafted optical flow algorithms may be loosely categorised into: (i) plane-fitting, (ii) gradient-based and (iii) block-matching. In plane-fitting and contrast maximisation, events are treated as a point-cloud in 3D space-time. The idea is to fit local planes to the points to capture linear motion of straight edges that sweep out planes in space-time [Aung et al., 2018; Akolkar et al., 2018]. The gradient of the plane with respect to the time-axis gives an estimate of the optical flow. A major drawback of this method is that it fails to capture (local) non-linear motion of non-straight edges: a scenario that occurs often reality. Contrast maximisation assumes global [Gallego et al., 2018] or local [Stoffregen and Kleeman, 2017; Stoffregen et al., 2019] affine motion parameters and warps every event in the point cloud to a single image plane according to the motion parameters. Optimising the motion parameters to maximise the contrast[3] of the warped event image yields an estimate for optical flow that is not constrained to straight edges as in plane-fitting, though still assumes (locally) linear motion. Gradient-based methods [Cook et al., 2011; Bardow et al., 2016; Bardow, 2018; Almatrafi and Hirakawa, 2019] (Fig. 2.9) aim to estimate both spatial intensity gradient and optical flow, typically assuming and invoking the brightness constancy equation [Horn and Schunck, 1981]. These methods can give dense optic flow and can re-use well established techniques such as smoothness regularisation [Lucas and Kanade, 1981; Horn and Schunck, 1981], however, error in gradient estimation may impact flow and vice versa. A promising direction for computationally efficient flow is block-matching [Liu and Delbruck, 2017, 2018], that compares a local patch to its neighbours to see 'where it went' using cheap operations e.g., sum of absolute differences. One key challenge is determining how many events to accumulate in a local patch before computing the flow to maximise throughput and the chance of finding a match while minimising redundant computations. While fast, block-matching techniques have limited accuracy and are spatially sparse.

Zhu et al. [2019] (Fig. 2.10) propose a learning based method that yields improved results over prior methods. In [Zhu et al., 2019], the authors propose two separate networks that share a similar architecture, but not the weights. One network predicts optical flow while the other predicts ego-motion and depth, that can be optionally converted to flow. Gehrig et al. [2019a] propose supervised training based on the ground-truth optical flow while others use a self/un-supervised warping loss with respect to registered DAVIS images [Zhu et al., 2018b], event voxels [Zhu et al., 2019] or slices (batch of events drawn onto an image plane) [Ye et al., 2019]. Ye et al. [2019] propose a novel architecture for a network that predicts depth and ego-motion, that can be converted to an optical flow estimate using a calibrated camera and rigid ego-motion model (in static environments). They choose to evaluate different scene types separately (e.g., driving vs. indoor flying), retraining the network for each scene type[4], and achieve high accuracy, though may be overfitting. A key challenge is evaluation, since ground-truth optic flow registered to real event data is difficult to obtain, and the researchers cannot rely on existing traditional datasets (e.g., [Baker et al., 2011; Menze and Geiger, 2015]) because they do not contain event

[3]Or similar quantity e.g., minimise variance. For a more comprehensive analysis of loss functions see [Gallego et al., 2019; Stoffregen and Kleeman, 2019].

[4]At the time of writing most works train on only driving data and evaluate on both driving and flying [Zhu et al., 2018b, 2019; Gehrig et al., 2019a].

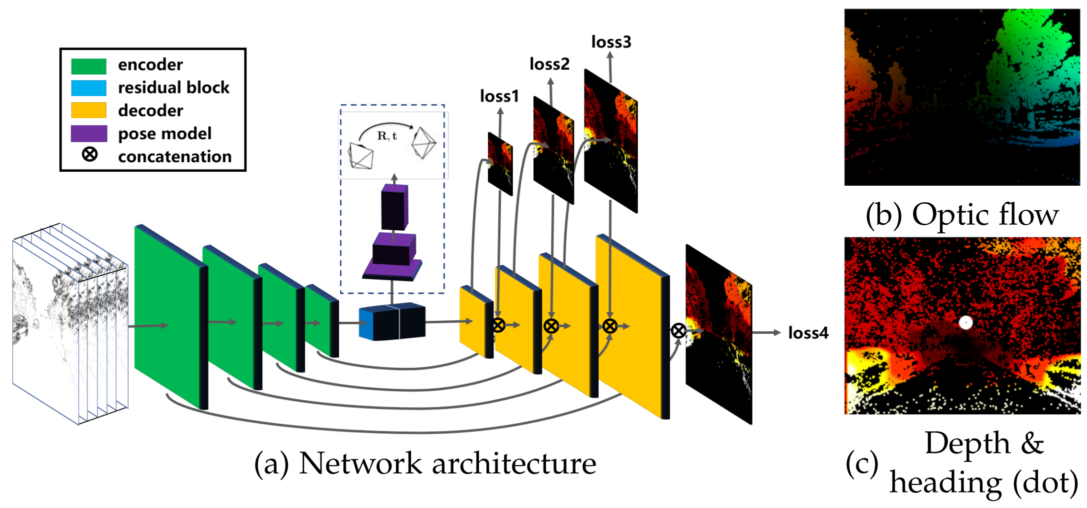
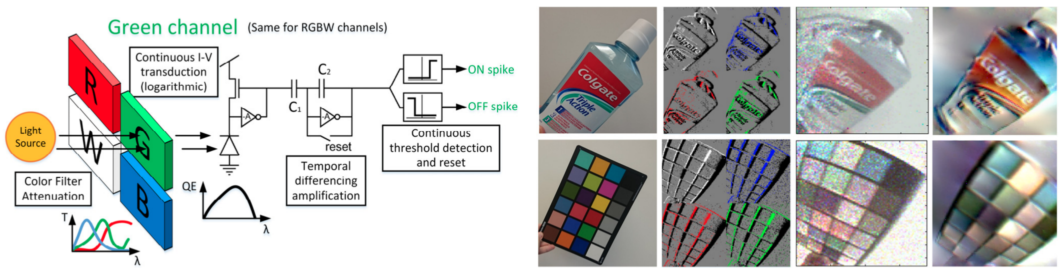


Figure 2.10: (a) Network architecture used for both (b) optic flow network (without pose model) and (c) depth and ego-motion network. At training, a loss at each decoder resolution is imposed. Adapted from [Zhu et al., 2019].

data. In response, new datasets have been proposed, however, ground-truth is limited to only the component of optical flow induced by ego-motion [Rueckauer and Delbruck, 2016; Zhu et al., 2018a], thus failing to test an algorithm's performance in dynamic scenes. This may lead to overfitting, for example, a network that predicts ego-motion and depth *instead* of optical flow may perform well on 'ego-motion optical flow datasets' [Rueckauer and Delbruck, 2016; Zhu et al., 2018a], while failing to produce accurate flow estimates of dynamic scenes, e.g., a traffic scenario with multiple vehicles. Stoffregen et al. [2020] show that EV-FlowNet [Zhu et al., 2018b], a network trained on one dataset [Zhu et al., 2018a], performs worse on different datasets such as [Mueggler et al., 2017b] and show that ensuring a variety of training data is key to improving generalisability.

2.5 Color

While the majority of event camera works to date use grayscale event cameras, a minority of researchers have explored color event cameras using Bayer patterned filters [Moeys et al., 2017, 2018; Li et al., 2015; Taverni et al., 2018] and even beam splitting rigs to separate colors [Marcireau et al., 2018]. Color event cameras may have improved performance on tasks such as classification and segmentation [Marcireau et al., 2018], medical (neural) imaging [Moeys et al., 2018], and can generate color images/video [Moeys et al., 2017] (Fig. 2.11). Researchers have also proposed improved sensor hardware design and characterisation [Li et al., 2015; Moeys et al., 2018; Taverni et al., 2018] targeted at color event cameras.


Green channel
(Same for RGBW channels)
Continuous I-V transduction (logarithmic)
C₂
C₁
reset
ON spike
OFF spike
Continuous threshold detection and reset
Temporal differencing amplification
Light Source
Color Filter Attenuation
R
G
W
B
QE
λ
T
λ

(a) Bayered event camera pixel (b) (c) (d) (e)

Figure 2.11: (a) RGBW Bayer pattern filter in front of a DVS pixel, (b) reference image (phone camera), (c) raw events (color-coded by filter), (d) naïve reconstruction based on integration, (e) improved reconstruction based on gradient estimation + Poisson integration. Adapted from [Moeys et al., 2017].

2.6 Summary

Chapter 2 motivated event cameras and explained their basic operating principles and output: a stream of asynchronous per-pixel brightness change *events*. Raw events have limited utility, so must be processed and useful information extracted before event event cameras can be used by intelligent systems to see. Asynchronous algorithms process each event upon arrival and can be used for computationally efficient image reconstruction and convolution. Batch processing incurs latency, though may achieve better results and easier algorithm design. Machine learning yields state-of-the-art performance for both image and optical flow estimation, however, may be computationally expensive, and limited interpretability makes it difficult to design robust, reliable models.

Chapter 3

Continuous-time Intensity Estimation

Website: https://cedricscheerlinck.com/continuous-time-intensity-estimation

Color Event Camera Dataset: http://rpg.ifi.uzh.ch/CED.html

Based on [Scheerlinck et al., 2018] and [Scheerlinck et al., 2019b]

Event cameras provide asynchronous, data-driven measurements of local temporal contrast over a large dynamic range with extremely high temporal resolution. Conventional cameras capture low-frequency reference intensity information. These two sensor modalities provide *complementary* information. I propose a computationally efficient, asynchronous filter that continuously fuses image frames and events into a single high-temporal-resolution, high-dynamic-range image state. In absence of conventional image frames, the filter can be run on events only. This image state can be queried locally or globally at any user-chosen time-instance(s) for computer vision tasks such motion estimation, object recognition or tracking, or to visualise data from an event camera in a human-interpretable way. I present experimental results on high-speed, high-dynamic-range sequences, as well as on synthetic datasets to evaluate the performance of the proposed algorithm. To aid future researchers I release open-source C++ code for the complementary filter and evaluation datasets.

3.1 Introduction

Due to high availability of contrast event cameras that output polarity (and not absolute brightness) with each event, such as the DAVIS, many researchers have tackled the challenge of estimating image intensity from contrast events. Image reconstruction algorithms that operate directly on the event stream typically perform spatiotemporal filtering [Reinbacher et al., 2016; Belbachir et al., 2014], or take a spatiotemporal window of events and convert them into a discrete image frame [Barua et al., 2016; Bardow et al., 2016; Rebecq et al., 2020b]. Windowing incurs a trade-off between length of time-window and latency. SLAM-like algorithms [Cook et al., 2011; Kim et al., 2014, 2016; Rebecq et al., 2017] maintain camera-pose and image gradient (or 3D) maps that can be upgraded to full intensity via Poisson integration [Agrawal

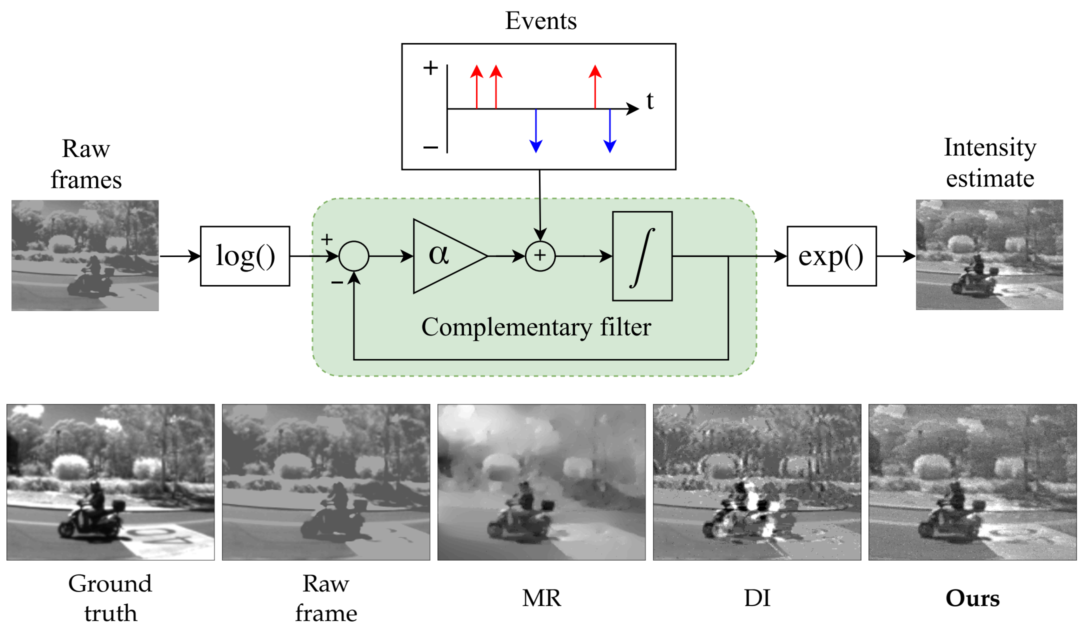


Figure 3.1: The complementary filter takes image frames and events, and produces a high-dynamic-range, high-temporal-resolution, continuous-time intensity estimate. Existing methods; manifold regularisation (MR) [Reinbacher et al., 2016] and direct integration [Brandli et al., 2014b] suffer smoothing and delay artifacts.

et al., 2005, 2006], however, so far these methods only work well for static scenes. Another image reconstruction algorithmic approach is to combine image frames directly with events [Brandli et al., 2014b]. Beginning with an image frame, events are integrated to produce inter-frame intensity estimates. The estimate is reset with every new frame to prevent growth of integration error.

In this chapter, I present a continuous-time formulation of event-based intensity estimation using complementary filtering to combine image frames with events (Fig. 3.1). I choose an asynchronous, event-driven update scheme for the complementary filter to efficiently incorporate the latest event information, eliminating windowing latency. The proposed approach does not depend on a motion-model, and works well in highly dynamic, complex environments. Rather than reset the intensity estimate with arrival of a new frame, the proposed formulation retains the high-dynamic-range information from events, maintaining an image state with greater temporal resolution and dynamic range than the image frames. The proposed method also works well on a pure event stream without requiring image frames. The result is a continuous-time estimate of intensity that can be queried locally or globally at any user-chosen time.

I demonstrate the proposed approach on datasets containing image frames and an event stream available from the DAVIS camera [Brandli et al., 2014a], and show that the complementary filter also works on a pure event stream without image frames.

The complementary filter does not perform spatial smoothing making amenable to Bayered color event data. I apply the proposed approach to the Color Event Camera Dataset [Scheerlinck et al., 2019b] to reconstruct a raw Bayered intensity image state able to be converted to a color image using a simple demosaicing algorithm [OpenCV, 2015]. If available, synthetic frames reconstructed from events via an alternative algorithm can also be used as input to the complementary filter. Thus, the proposed method can be used to augment any intensity reconstruction algorithm. Additionally, I show how an adaptive gain can be used to improve robustness against under/overexposed image frames.

In summary, the key contributions of the chapter are;

- a continuous-time formulation of event-based intensity estimation,
- a computationally simple, asynchronous, event-driven filter algorithm,
- a methodology for pixel-by-pixel adaptive gain tuning.

I also present a ground truth dataset for reconstruction of intensities from combined image frame and event streams, published with co-authors in [Scheerlinck et al., 2018]. Sequences of images taken on a high-speed camera form the ground truth. I retain full frames at 20Hz, and convert the inter-frame images to an event stream. I compare state-of-the-art approaches on this dataset.

The chapter is organised as follows: Section 3.2 summarises the mathematical representation and notation used, and characterises the full continuous-time solution of the proposed filter. Section 3.3 describes asynchronous implementation of the complementary filter, and introduces adaptive gains. Section 3.4 shows experimental results on the ground truth dataset and high-speed, high-dynamic-range sequences from the DAVIS. Section 3.5 concludes the chapter.

3.2 Approach

3.2.1 Mathematical Representation and Notation

Let $Y(\boldsymbol{p}, t)$ denote the intensity or irradiance of pixel $\boldsymbol{p}$ at time t of a camera. I will assume that the same irradiance is observed at the same pixel in both a classical and event camera, such as is the case with the DAVIS camera [Brandli et al., 2014a]. A classical image frame (for a global shutter camera) is an average of the received intensity over the exposure time

$$Y_j(\boldsymbol{p}) := \frac{1}{\epsilon}\int_{t_j-\epsilon}^{t_j} Y(\boldsymbol{p},\tau)\mathrm{d}\tau, \quad j \in 1, 2, 3 \dots, \tag{3.1}$$

where t_j is the time-stamp of the image capture and ϵ is the exposure time. In the sequel I will ignore the exposure time in the analysis and simply consider a classical image as representing image information available at time t_j. Although there will be image blur effects, especially for fast moving scenes in low light conditions (see the experimental results in §3.4), a full consideration of these effects is beyond the scope of the present chapter.

The approach taken in this chapter is to analyse image reconstruction for event cameras in the continuous-time domain. To this end, I define a continuous-time intensity signal $Y^F(\boldsymbol{p}, t)$ as the zero-order hold (ZOH) reconstruction of the irradiance from the classical image frames:

$$Y^F(\boldsymbol{p}, t) := Y_j(\boldsymbol{p}) = Y(\boldsymbol{p}, t_j), \quad t_j \le t < t_{j+1} \tag{3.2}$$

Since event cameras operate with log intensity I convert the image intensity into log-intensity:

$$L(\boldsymbol{p}, t) := \log(Y(\boldsymbol{p}, t)) \tag{3.3}$$
$$L_j(\boldsymbol{p}) := \log(Y_j(\boldsymbol{p})) \tag{3.4}$$
$$L^F(\boldsymbol{p}, t) := \log(Y^F(\boldsymbol{p}, t)). \tag{3.5}$$

Note that converting the zero-hold signal into the log domain is not the same as integrating the log intensity of the irradiance over the shutter time. I believe the difference will be insignificant in the scenarios considered and I do not consider this further in the present chapter.

Dynamic vision sensors (DVS), or event cameras, are biologically-inspired vision sensors that respond to changes in scene illumination. Each pixel is independently wired to continuously compare the current log intensity level to the last reset-level. When the difference in log intensity exceeds a predetermined threshold (contrast threshold), an event is transmitted and the pixel resets, storing the new illumination level. Each event contains the pixel coordinates, timestamp, and polarity ($\sigma = \pm 1$ for increasing or decreasing intensity). An event can be modelled in the continuous-time[1] signal class as a Dirac-delta function $\delta(t)$. I define an event stream $e_i(\boldsymbol{p}, t)$ at pixel $\boldsymbol{p}$ by

$$e_i(\boldsymbol{p}, t) := \sigma_i^{\boldsymbol{p}} c\, \delta(t - t_i^{\boldsymbol{p}}),\ i \in 1, 2, 3 \ldots, \tag{3.6}$$

where $\sigma_i^{\boldsymbol{p}}$ is the polarity and $t_i^{\boldsymbol{p}}$ is the time-stamp of the i^{th} event at pixel $\boldsymbol{p}$. The magnitude c is the *contrast threshold* (brightness change encoded by one event). Define an *event field* $E(\boldsymbol{p}, t)$ by

$$E(\boldsymbol{p}, t) := \sum_{i=1}^{\infty} e_i(\boldsymbol{p}, t) = \sum_{i=1}^{\infty} \sigma_i^{\boldsymbol{p}} c\, \delta(t - t_i^{\boldsymbol{p}}). \tag{3.7}$$

The event field is a function of all pixels $\boldsymbol{p}$ and ranges over all time, capturing the full output of the event camera.

A quantised log intensity signal $L^E(\boldsymbol{p}, t)$ can be reconstructed by integrating the

[1]Note that events are continuous-time signals even though they are not continuous functions of time; the time variable t on which they depend varies continuously.

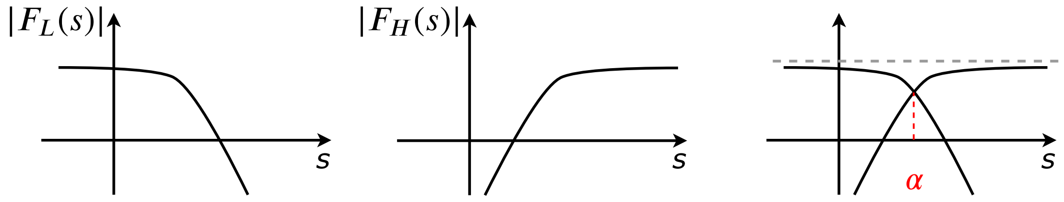


Figure 3.2: Bode (log) plots of: low pass filter $F_L(s)$ (left), high pass filter $F_H(s)$ (middle), all pass complementary filter $F_L(s) + F_H(s)$ (right). The complementary filter frequency response sums to unity at all frequencies (gray dashed line). The crossover frequency is denoted by α.

event field

$$L^E(\boldsymbol{p}, t) := \int_0^t E(\boldsymbol{p}, \tau) d\tau = \int_0^t \sum_{i=1}^{\infty} \sigma_i^{\boldsymbol{p}} c \, \delta(\tau - t_i^{\boldsymbol{p}}) d\tau. \tag{3.8}$$

The result is a series of log intensity steps (corresponding to events) at each pixel. In the absence of noise, the relationship between the log-intensity $L(\boldsymbol{p}, t)$ and the quantised signal $L^E(\boldsymbol{p}, t)$ is

$$L(\boldsymbol{p}, t) = L^E(\boldsymbol{p}, t) + L(\boldsymbol{p}, 0) + \mu(\boldsymbol{p}, t; c), \tag{3.9}$$

where $L(\boldsymbol{p}, 0)$ is the initial condition and $\mu(\boldsymbol{p}, t; c)$ is the quantisation error. Unlike $L^F(\boldsymbol{p}, t)$, the quantisation error associated with $L^E(\boldsymbol{p}, t)$ is bounded by the contrast threshold; $|\mu(\boldsymbol{p}, t; c)| < c$.

Events can be interpreted as the temporal derivative of $L^E(\boldsymbol{p}, t)$

$$E(\boldsymbol{p}, t) = \frac{\partial}{\partial t} L^E(\boldsymbol{p}, t). \tag{3.10}$$

3.2.2 Complementary Filter

I will use a complementary filter structure [Higgins, 1975; Mahony et al., 2008; Franklin et al., 1998] (Fig. 3.2) to fuse the event field $E(\boldsymbol{p}, t)$ with ZOH log-intensity frames $L^F(\boldsymbol{p}, t)$. Complementary filtering is ideal for fusing signals that have complementary frequency noise characteristics; for example, where one signal is dominated by high-frequency noise and the other by low-frequency disturbance. Events are a temporal derivative measurement (3.10) and do not contain reference intensity $L(\boldsymbol{p}, 0)$ information. Integrating events to obtain $L^E(\boldsymbol{p}, t)$ amplifies low-frequency disturbance (drift), resulting in poor low-frequency information. However, due to their high-temporal-resolution, events provide reliable *high-frequency* information. Classical image frames $L^F(\boldsymbol{p}, t)$ are derived from discrete, temporally-sparse measurements and have poor high-frequency fidelity. However, frames typically provide reliable *low-frequency* reference intensity information. The proposed complementary filter architecture combines a *high-pass* version of $L^E(\boldsymbol{p}, t)$ with a *low-pass* version of

$L^F(\boldsymbol{p}, t)$ to reconstruct an (approximate) all-pass version of $L(\boldsymbol{p}, t)$.

The proposed filter is written as a continuous-time ordinary differential equation (ODE)

$$\boxed{\frac{\partial}{\partial t}\hat{L}(\boldsymbol{p}, t) = E(\boldsymbol{p}, t) - \alpha\big(\hat{L}(\boldsymbol{p}, t) - L^F(\boldsymbol{p}, t)\big),} \tag{3.11}$$

where $\hat{L}(\boldsymbol{p}, t)$ is the continuous-time log-intensity state estimate and α is the complementary filter gain, or crossover frequency [Mahony et al., 2008] (Fig. 3.1).

The fact that the input signals in (3.11) are discontinuous poses some complexities in solving the filter equations, but does not invalidate the formulation. The filter can be understood as integration of the event field with an innovation term $-\alpha\big(\hat{L}(\boldsymbol{p}, t) - L^F(\boldsymbol{p}, t)\big)$, that acts to reduce the error between $\hat{L}(\boldsymbol{p}, t)$ and $L^F(\boldsymbol{p}, t)$.

The key property of the proposed filter (3.11) is that although it is posed as a continuous-time ODE, one can express the solution as a set of asynchronous-update equations. Each pixel acts independently, and in the sequel I will consider the action of the complementary filter on a single pixel $\boldsymbol{p}$. Recall the sequence $\{t_i^{\boldsymbol{p}}\}$ corresponding to the time-stamps of all events at $\boldsymbol{p}$. In addition, there is the sequence of classical image frame time-stamps $\{t_j\}$ that apply to all pixels equally. Consider a combined sequence of monotonically increasing *unique* time-stamps $\hat{t}_k^{\boldsymbol{p}}$ corresponding to event $\{t_i^{\boldsymbol{p}}\}$ or frame $\{t_j\}$ time-stamps.

Within a time-interval $t \in [\hat{t}_k^{\boldsymbol{p}}, \hat{t}_{k+1}^{\boldsymbol{p}})$ there are (by definition) no new events or frames, and the ODE (3.11) is a constant coefficient linear ordinary differential equation

$$\frac{\partial}{\partial t}\hat{L}(\boldsymbol{p}, t) = -\alpha\big(\hat{L}(\boldsymbol{p}, t) - L^F(\boldsymbol{p}, t)\big), \quad t \in [\hat{t}_k^{\boldsymbol{p}}, \hat{t}_{k+1}^{\boldsymbol{p}}). \tag{3.12}$$

The solution to this ODE is given by

$$\hat{L}(\boldsymbol{p}, t) = e^{-\alpha(t-\hat{t}_k^{\boldsymbol{p}})}\hat{L}(\boldsymbol{p}, \hat{t}_k^{\boldsymbol{p}}) + (1 - e^{-\alpha(t-\hat{t}_k^{\boldsymbol{p}})})L^F(\boldsymbol{p}, t), \qquad t \in [\hat{t}_k^{\boldsymbol{p}}, \hat{t}_{k+1}^{\boldsymbol{p}}). \tag{3.13}$$

It remains to paste together the piece-wise smooth solutions on the half-open intervals $[\hat{t}_k^{\boldsymbol{p}}, \hat{t}_{k+1}^{\boldsymbol{p}})$ by considering the boundary conditions. Let

$$(\hat{t}_{k+1}^{\boldsymbol{p}})^- := \lim_{t \to (\hat{t}_{k+1}^{\boldsymbol{p}})} t, \quad \text{for } t < \hat{t}_{k+1}^{\boldsymbol{p}} \tag{3.14}$$

$$(\hat{t}_{k+1}^{\boldsymbol{p}})^+ := \lim_{t \to (\hat{t}_{k+1}^{\boldsymbol{p}})} t, \quad \text{for } t > \hat{t}_{k+1}^{\boldsymbol{p}}, \tag{3.15}$$

denote the limits from below and above. There are two cases to consider:

New frame: When the index $\hat{t}_{k+1}^{\boldsymbol{p}}$ corresponds to a new image frame then the right hand side (RHS) of (3.11) has bounded variation. It follows that the solution is continuous at $\hat{t}_{k+1}^{\boldsymbol{p}}$ and

$$\hat{L}(\boldsymbol{p}, \hat{t}_{k+1}^{\boldsymbol{p}}) = \hat{L}(\boldsymbol{p}, (\hat{t}_{k+1}^{\boldsymbol{p}})^-). \tag{3.16}$$

Event: When the index $\hat{t}^p_{k+1}$ corresponds to an event then the solution of (3.11) is *not* continuous at $\hat{t}^p_{k+1}$ and the Dirac delta function of the event must be integrated. Integrating the RHS and LHS of (3.11) over an event

$$\int_{(\hat{t}^p_{k+1})^-}^{(\hat{t}^p_{k+1})^+} \frac{d}{d\tau}\hat{L}(\boldsymbol{p},\tau)d\tau = \int_{(\hat{t}^p_{k+1})^-}^{(\hat{t}^p_{k+1})^+} E(\boldsymbol{p},\tau) - \alpha\big(\hat{L}(\boldsymbol{p},\tau) - L^F(\boldsymbol{p},\tau)\big)d\tau \tag{3.17}$$

$$\hat{L}(\boldsymbol{p},(\hat{t}^p_{k+1})^+) - \hat{L}(\boldsymbol{p},(\hat{t}^p_{k+1})^-) = \sigma^p_{k+1}c, \tag{3.18}$$

yields a unit step scaled by the contrast threshold and sign of the event. Note the integral of the second term $\int_{(\hat{t}^p_{k+1})^-}^{(\hat{t}^p_{k+1})^+} \alpha\big(\hat{L}(\boldsymbol{p},\tau) - L^F(\boldsymbol{p},\tau)\big)d\tau$ is zero since the integrand is bounded. I use the solution

$$\hat{L}(\boldsymbol{p},\hat{t}^p_{k+1}) = \hat{L}(\boldsymbol{p},(\hat{t}^p_{k+1})^-) + \sigma^p_{k+1}c, \tag{3.19}$$

as initial condition for the next time-interval. Eqns. (3.13), (3.16) and (3.19) characterise the full solution to the filter equation (3.11). The filter can be run on *events only* without image frames by setting $L^F(\boldsymbol{p},t) = 0$ in (3.11), resulting in a high-pass filter with corner frequency α

$$\boxed{\frac{\partial}{\partial t}\hat{L}(\boldsymbol{p},t) = E(\boldsymbol{p},t) - \alpha\hat{L}(\boldsymbol{p},t).} \tag{3.20}$$

This method can efficiently generate a good quality image state estimate from pure events. Furthermore, it is possible to use alternative pure event-based methods to reconstruct a temporally-sparse image sequence from events and fuse this with raw events using the proposed complementary filter. Thus, the proposed filter can be considered a method to augment any event-based image reconstruction method to obtain a high temporal-resolution image state.

3.3 Method

3.3.1 Adaptive Gain Tuning

The complementary filter gain α is a parameter that controls the relative information contributed by image frames or events. Reducing the magnitude of α decreases the dependence on image frame data while increasing the dependence on events ($\alpha = 0 \to \hat{L}(\boldsymbol{p},t) = L^E(\boldsymbol{p},t)$). A key observation is that the gain can be time-varying at pixel-level ($\alpha = \alpha(\boldsymbol{p},t)$). One can therefore use $\alpha(\boldsymbol{p},t)$ to dynamically adjust the relative dependence on image frames or events, which can be useful when image frames are compromised, e.g. under/overexposed.

I propose to reduce the influence of under/overexposed image frame pixels by decreasing $\alpha(\boldsymbol{p},t)$ at those pixel locations. I use the heuristic that pixels reporting an intensity close to the minimum L_{min} or maximum L_{max} output of the camera may be compromised, and I decrease $\alpha(\boldsymbol{p},t)$ based on the reported log intensity. I choose two bounds L_1, L_2 close to L_{min} and L_{max}, then I set $\alpha(\boldsymbol{p}, t)$ to a constant (α_1) for all pixels within the range $[L_1, L_2]$, and linearly decrease $\alpha(\boldsymbol{p}, t)$ for pixels outside of this range:

$$\alpha(\boldsymbol{p}, t) = \begin{cases} \lambda\alpha_1 + (1-\lambda)\alpha_1 \frac{(L^F(\boldsymbol{p},t) - L_{\text{min}})}{(L_1 - L_{\text{min}})} & L_{\text{min}} \leq L^F(\boldsymbol{p}, t) < L_1 \\ \alpha_1 & L_1 \leq L^F(\boldsymbol{p}, t) \leq L_2 \\ \lambda\alpha_1 + (1-\lambda)\alpha_1 \frac{(L^F(\boldsymbol{p},t) - L_{\text{max}})}{(L_2 - L_{\text{max}})} & L_2 < L^F(\boldsymbol{p}, t) \leq L_{\text{max}} \end{cases} \tag{3.21}$$

where λ is a parameter determining the strength of the adaptive scheme (I set $\lambda = 0.1$). For α_1, typical suitable values are $\alpha_1 \in [0.1, 10]$ rad/s. For experiments I choose $\alpha_1 = 2\pi$ rad/s.

3.3.2 Asynchronous Update Scheme

Given temporally sparse image frames and events, and using the continuous-time solution to the complementary filter ODE (3.11) outlined in §3.2 one may compute the intensity state estimate $\hat{L}(\boldsymbol{p}, t)$ at any time. In practice it is sufficient to compute the image state $\hat{L}(\boldsymbol{p}, t)$ at the asynchronous time instances $t_k^{\boldsymbol{p}}$ (event or frame timestamps). I propose an asynchronous update scheme whereby new events cause state updates (3.19) *only* at the event pixel-location. New frames cause a global update (3.16) (note this is not a reset as in [Brandli et al., 2014b]) [2]. Algorithm 1 describes a per-pixel complementary filter implementation. At a given pixel $\boldsymbol{p}$, let $\hat{L}_\diamond$ denote the latest estimate of $\hat{L}(\boldsymbol{p}, t)$ stored in computer memory, and $\hat{t}_\diamond$ denote the time-stamp of the latest update at $\boldsymbol{p}$. Let $L_\diamond^F$ and $\alpha_\diamond$ denote the latest image frame and gain values at $\boldsymbol{p}$. To run the filter in events only mode (high-pass filter (3.20)), simply let $L_\diamond^F = 0$.

Algorithm 1 Per-pixel, Asynchronous Complementary Filter

```
 1: At each pixel:
 2: Initialise $\hat{L}_\diamond$, $\hat{t}_\diamond$, $L_\diamond^F$ to zero
 3: Initialise $\alpha_\diamond$ to $\alpha_1$
 4: for each new event or image frame do
 5:     $\Delta t \leftarrow t - \hat{t}_\diamond$
 6:     $\hat{L}_\diamond \leftarrow \exp(-\alpha_\diamond \cdot \Delta t) \cdot \hat{L}_\diamond + (1 - \exp(-\alpha_\diamond \cdot \Delta t)) \cdot L_\diamond^F$ based on (3.13)
 7:     if event then
 8:         $\hat{L}_\diamond \leftarrow \hat{L}_\diamond + \sigma c$ based on (3.19)
 9:     else if image frame then
10:         Replace $L_\diamond^F$ with new frame
11:         Update $\alpha_\diamond$ based on (3.21)
12:     $\hat{t}_\diamond \leftarrow t$
```

[2] The filter can also be updated (using (3.13)) at any user-chosen time instance (or rate). In my experiments I update the entire image state whenever I export the image for visualisation.

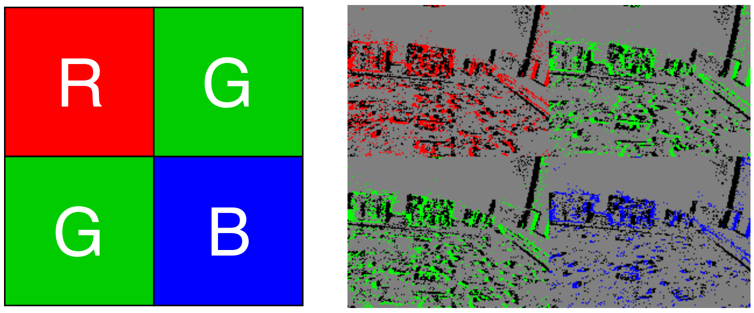


Figure 3.4: Left: 2×2 RGBG Bayer pattern in the Color-DAVIS346. Right: Events from the Color-DAVIS346 split into each color. Positive (ON) events are colored by the corresponding filter color, negative (OFF) events are black.

3.3.3 Color

Since equations (3.11) and (3.20) describe pixel-wise filters (i.e. no spatial smoothing), they are amenable to Bayered color event data, such as from [Scheerlinck et al., 2019b] (Fig. 3.3). Figure 3.4 describes the Bayer color filter pattern on the Color-DAVIS346 event camera. The 2×2 RGBG filter (Fig. 3.4) patterns the entire sensor, thus, generating events that are sensitive to either red, green or blue light based on pixel location.

Figure 3.3: "DAVIS346 Red Color" event camera used to record the *Color Event Camera Dataset* [Scheerlinck et al., 2019b].

Simply feeding Bayered color events into the (grayscale) complementary filters described in equations (3.11) or (3.20) allows reconstruction of raw Bayered images. Demosaicing [Kimmel, 1999] can be used to recover an RGB image at any point in time. This is only possible because the proposed approach does not involve spatial smoothing that would destroy the Bayer pattern.

A different algorithmic approach to color image reconstruction is to reconstruct each channel independently (at quarter resolution), then upsample the result back to the original resolution. While this technique is unnecessary for the proposed method, I apply it to contemporary image reconstruction methods for comparison.

3.4 Results

I compare the reconstruction performance of the complementary filter[3], both with frames (CF) and without frames in *events only* mode (HF), against three existing methods: manifold regularisation (MR)[4] [Reinbacher et al., 2016], direct integration (DI)

[3]Code for CF & HF: https://github.com/cedric-scheerlinck/dvs_image_reconstruction
[4]Code for MR: https://github.com/VLOGroup/dvs-reconstruction

[Brandli et al., 2014b], simultaneous optical flow and intensity estimation (SOFIE) [Bardow et al., 2016]. E2VID [Rebecq et al., 2020b] is a convolutional neural network that converts events to video, and I will provide comprehensive evaluation and comparison to E2VID in chapters 5 and 6. I use the DVS image reconstruction dataset [Scheerlinck et al., 2018] that consists of two ground truth sequences (Truck and Motorbike); and four sequences taken with the DAVIS240C [Brandli et al., 2014a] camera (Night drive, Sun, Bicycle, Night run). I evaluate the proposed method, MR and DI against the ground truth dataset [Scheerlinck et al., 2018] using quantitative image similarity metrics. Unfortunately, as code is not available for SOFIE I am unable to evaluate its performance on new datasets. Hence, I compare it with the proposed method on the jumping sequence made available by the authors. I also present qualitative results on the Color Event Camera Dataset (CED) [Scheerlinck et al., 2019b].

Ground truth is obtained using a high-speed, global-shutter, frame-based camera (mvBlueFOX USB 2) running at 168Hz. I acquire image sequences of dynamic scenes (Truck and Motorbike), and convert them into events following the methodology of Mueggler et al. [2017b]. To simulate event camera noise, a number of random noise events are generated (5% of total events), and distributed randomly throughout the event stream. To simulate low-dynamic-range, low-temporal-resolution input-frames, the upper and lower 25% of the maximum intensity range is truncated, and image frames are subsampled at 20Hz. In addition, a delay of 50ms is applied to the frame time-stamps to simulate the latency associated with capturing images using a frame-based camera.

The complementary filter gain $\alpha(\boldsymbol{p}, t)$ is set according to (3.21) and updated with every new image frame (Algorithm 1). I set $\alpha_1 = 2\pi$ rad/s for all sequences unless otherwise stated. The bounds $[L_1,\ L_2]$ in (3.21) are set to $[L_{\text{min}} + \kappa,\ L_{\text{max}} - \kappa]$, where $\kappa = 0.05(L_{\text{max}} - L_{\text{min}})$. The contrast threshold (c) is not easy to determine and in practice varies across pixels, and with illumination, event-rate and other factors [Brandli et al., 2014b]. Here I assume two constant contrast thresholds (ON and OFF) that are calibrated for each sequence using APS frames. I note that error arising from the variability of contrast thresholds appears as noise in the final estimate, and believe that more sophisticated contrast threshold models may benefit future works. For MR [Reinbacher et al., 2016], the number of events per output image (events/image) is a parameter that impacts the quality of the reconstructed image. For each sequence I choose events/image to give qualitatively best performance. I set events/image to 1500 unless otherwise stated. All other parameters are set to defaults provided by Reinbacher et al. [2016].

3.4.1 Qualitative Results

Night drive. Fig. 3.5 investigates performance in high-speed, low light conditions where the conventional camera image frame (Raw frame) is blurry and underexposed, and dark details are lost. Data is recorded through the front wind shield of a car, driving down an unlit highway at dead of night. The proposed method (CF) is able to recover motion-blurred objects (e.g. roadside poles), trees that are lost in

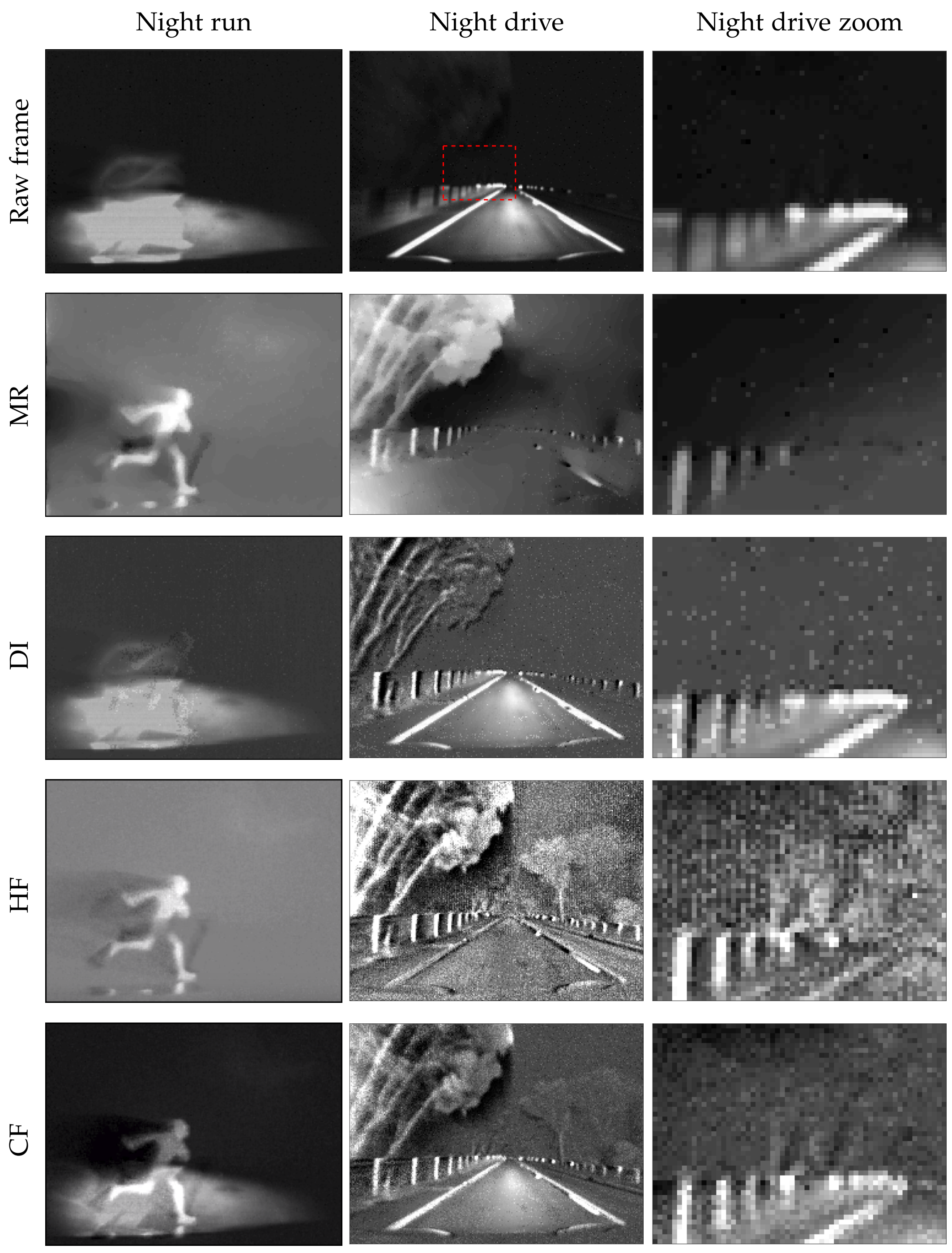


Figure 3.5: **Night drive:** Raw frame is motion-blurred and contains little information in dark areas, but captures road markings. MR is unable to recover some road markings. CF recovers sharp road markings, trees and roadside poles. **Night run:** Pedestrian is heavily motion-blurred and delayed in Raw frame. DI may be compromised by frame-resets. MR and CF recover sharp detail despite high-speed, low-light conditions.

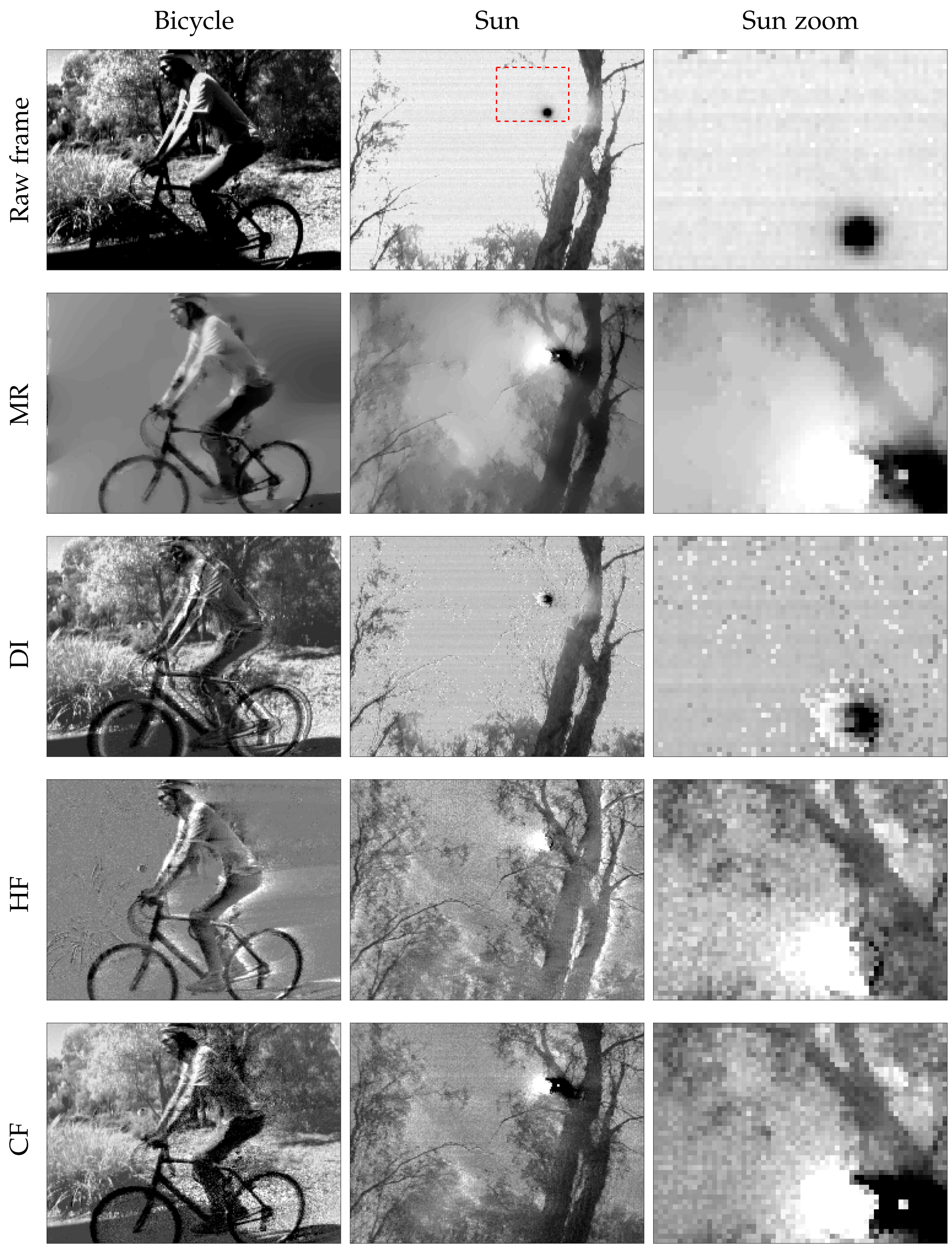


Figure 3.6: **Sun:** Raw frame is overexposed when pointing directly at the sun (black dot is a camera artifact caused by the sun). In MR, some features are smoothed out (see zoom). DI is washed out due to the latest frame reset. CF captures detailed leaves and twigs. **Bicycle:** Static background cannot be recovered from events alone in MR and HF. CF recovers both background and foreground.

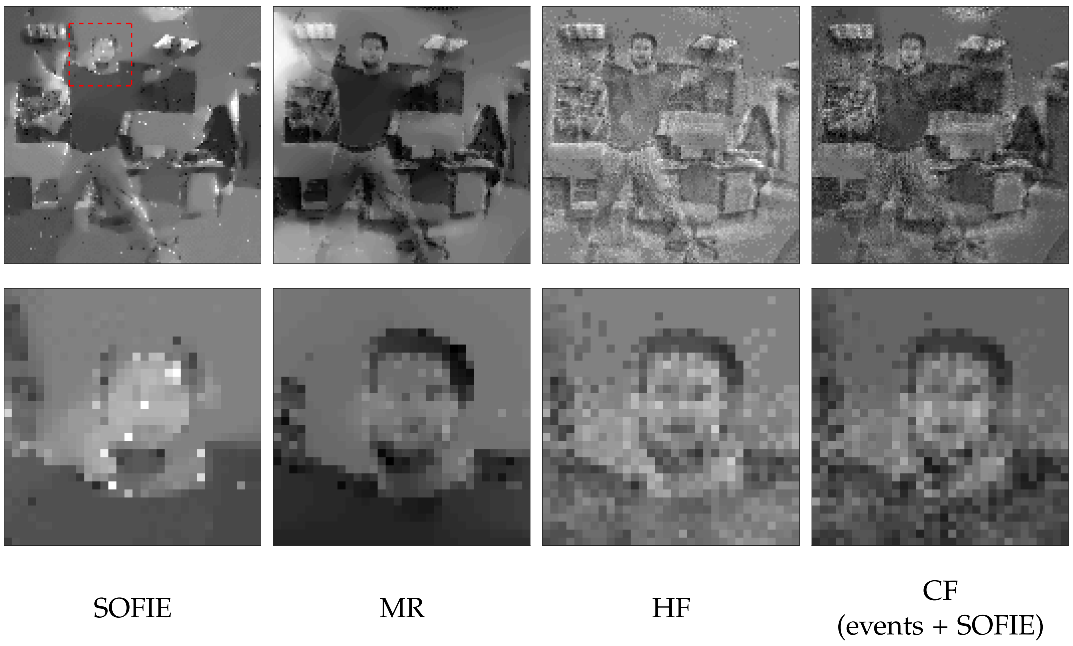


Figure 3.7: SOFIE [Bardow et al., 2016] and MR [Reinbacher et al., 2016] reconstruct images from a pure event stream. CF can reconstruct images from a pure event stream by either setting input-frames to zero HF, or by taking reconstructed images from other methods (e.g. SOFIE) as input-frames CF (events + SOFIE).

Raw frame, and road lines that are lost in MR. MR relies on spatial smoothing to reduce noise, hence faint features such as distant trees (Fig. 3.5; zoom) may be lost. DI loses features that require more time for events to accumulate (e.g. trees on the right), because the estimate is reset upon every new image frame. The APS (active pixel sensor in DAVIS) frame-rate was set to 7Hz.

Night run. Illustrates the benefit in challenging low-light pedestrian scenarios. Here a pedestrian runs across the headlights of a (stationary) car at dead of night. Raw frame is not only heavily motion-blurred, but also significantly delayed, since a large exposure duration is required for image acquisition in low-light conditions. DI is unreliable as an unfortunately timed new image frame could reset the image (Fig. 3.5). MR and CF manage to recover the pedestrian, and CF also recovers the background without compromising clarity of the pedestrian. The APS frame-rate was set to 4.5Hz.

Bicycle. Explores the scenario of static background, moving foreground. Raw frame is underexposed in shady areas because of large intra-scene dynamic range. When the event camera is stationary, almost no events are generated by the static background and it cannot be recovered by pure event-based reconstruction methods such as MR and HF. In contrast, CF recovers both stationary and non-stationary features, as well as high-dynamic-range detail. The APS frame-rate was set to 26Hz.

Sun. Investigates extreme dynamic range scenes where conventional cameras become overexposed. CF recovers features such as leaves and twigs, even when the camera is pointed directly at the sun. Raw frame is largely over-saturated, and the black dot (Fig. 3.6; Sun) is a camera artifact caused by extreme brightness, and marks the position of the sun. MR produces a clean looking image, though some features (small leaves/twigs) are smoothed out (Fig. 3.6; zoom). DI is washed out in regions where the frame is overexposed, due to the latest frame reset. Because the sun generates so many events, MR requires more events/image to recover fine features (with less events the image looks oversmoothed), so I increase events/image to 2500. The APS frame-rate was set to 26Hz.

SOFIE. The code for SOFIE [Bardow et al., 2016] was not available at the time of writing, however the authors kindly share their prerecorded dataset (using DVS128 [Lichtsteiner et al., 2008]) and results. I use their dataset to compare the proposed method to SOFIE (Fig. 3.7), and since no camera frames are available, I first demonstrate the proposed method by setting input-frames to zero (HF), then show that reconstructed image frames output from an alternative reconstruction algorithm such as SOFIE can be used as input-frames to the complementary filter (CF (events + SOFIE)) to generate intensity estimates.

Color Event Camera Dataset. For the proposed method HF I set $\alpha = 0.377$ and apply a 5×5, $\sigma = 25$ bilateral filter to the output. Figure 3.8 compares state-of-the-art image reconstruction methods on the Color Event Camera Dataset [Scheerlinck et al., 2019b]. For fairness I randomly select frames from each sequence. MR performs qualitatively poorer than HF (3.20) and E2VID [Rebecq et al., 2020a] on color event data. Figure 3.9 shows edge cases where various image reconstruction and imaging modalities fall short. For example, at initialisation, event-based image reconstruction requires a minimum number of events, especially MR and HF, before the image 'fills in', whereas frame-based cameras take a full image with one exposure. Fast motions increase the event rate, slowing down per-event algorithms, and can increase the number of noise events, producing artifacts in event reconstructed images (especially HF). Event-based algorithms typically suffer when the event rate slows down, and in this case manifests as the image fading. HF offers control (via the cutoff frequency α parameter) over the temporal dynamics of the reconstruction, whereas E2VID implicitly learns temporal dynamics (via recurrent units). E2VID fades more rapidly than HF with the parameters used, and unlike HF, there is no simple parameter that can be tuned at run-time to change the fade rate. Frame-based cameras have low intrascene dynamic range, whereas event reconstructed images have dynamic range in the order of event cameras, e.g., 120dB. In low light, frame-based cameras trade-off signal-to-noise with motion blur via the exposure duration. Low exposure reduces signal-to-noise and motion blur while high exposure increase both. Event reconstructed images are not limited by a uniform exposure, resulting in high signal-to-noise and low motion blur in low light.

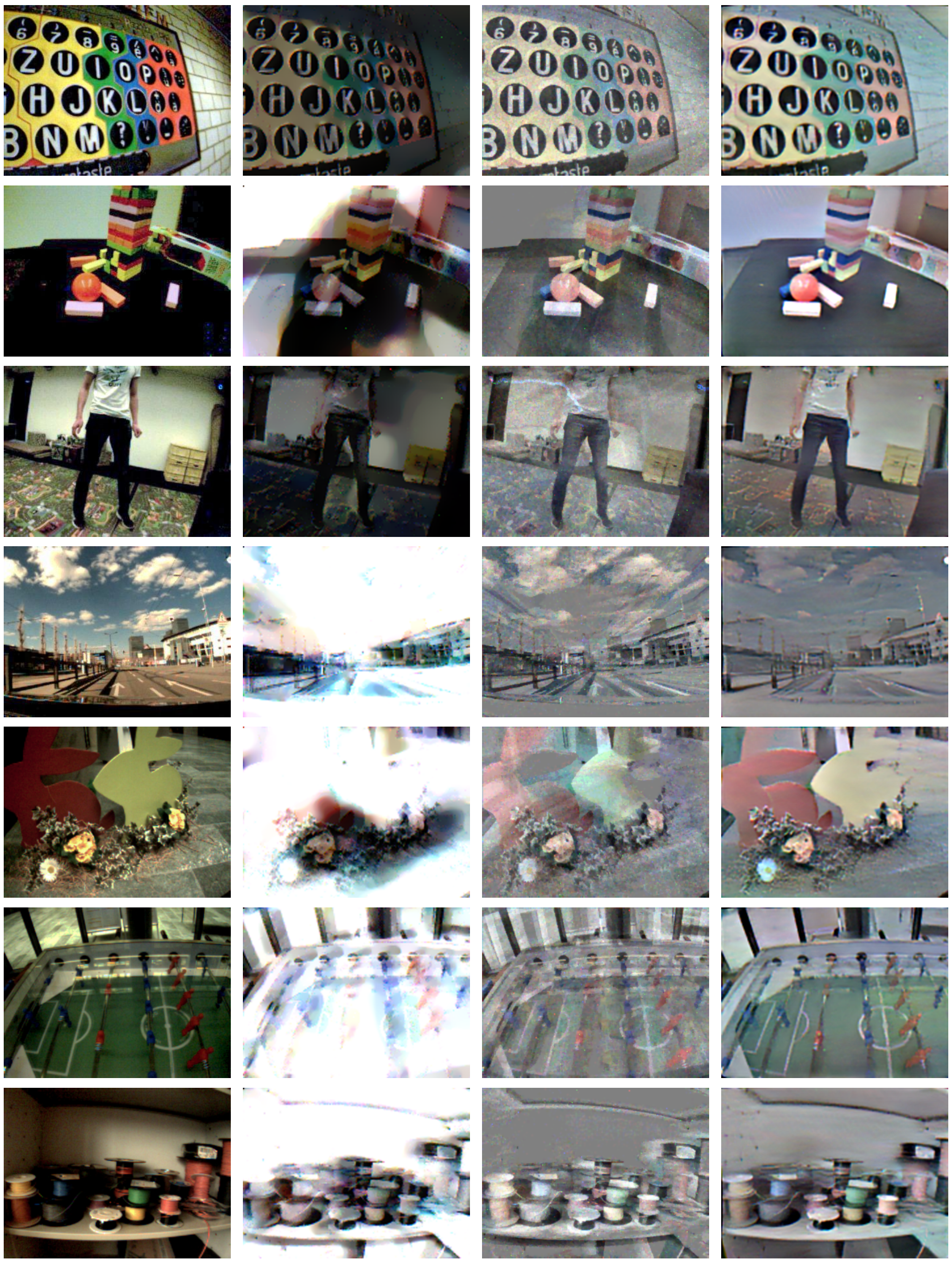


Figure 3.8: Qualitative comparison of different color video reconstruction methods on the Color Event Camera Dataset [Scheerlinck et al., 2019b] (images randomly selected). The DAVIS frame captured on the same sensor provides a reference image to compare against. Only events were used for each reconstruction method: MR [Reinbacher et al., 2016], HF (ours), E2VID [Rebecq et al., 2019]. HF achieves better quality than MR. While HF has inferior quality to E2VID, it is significantly faster to compute.

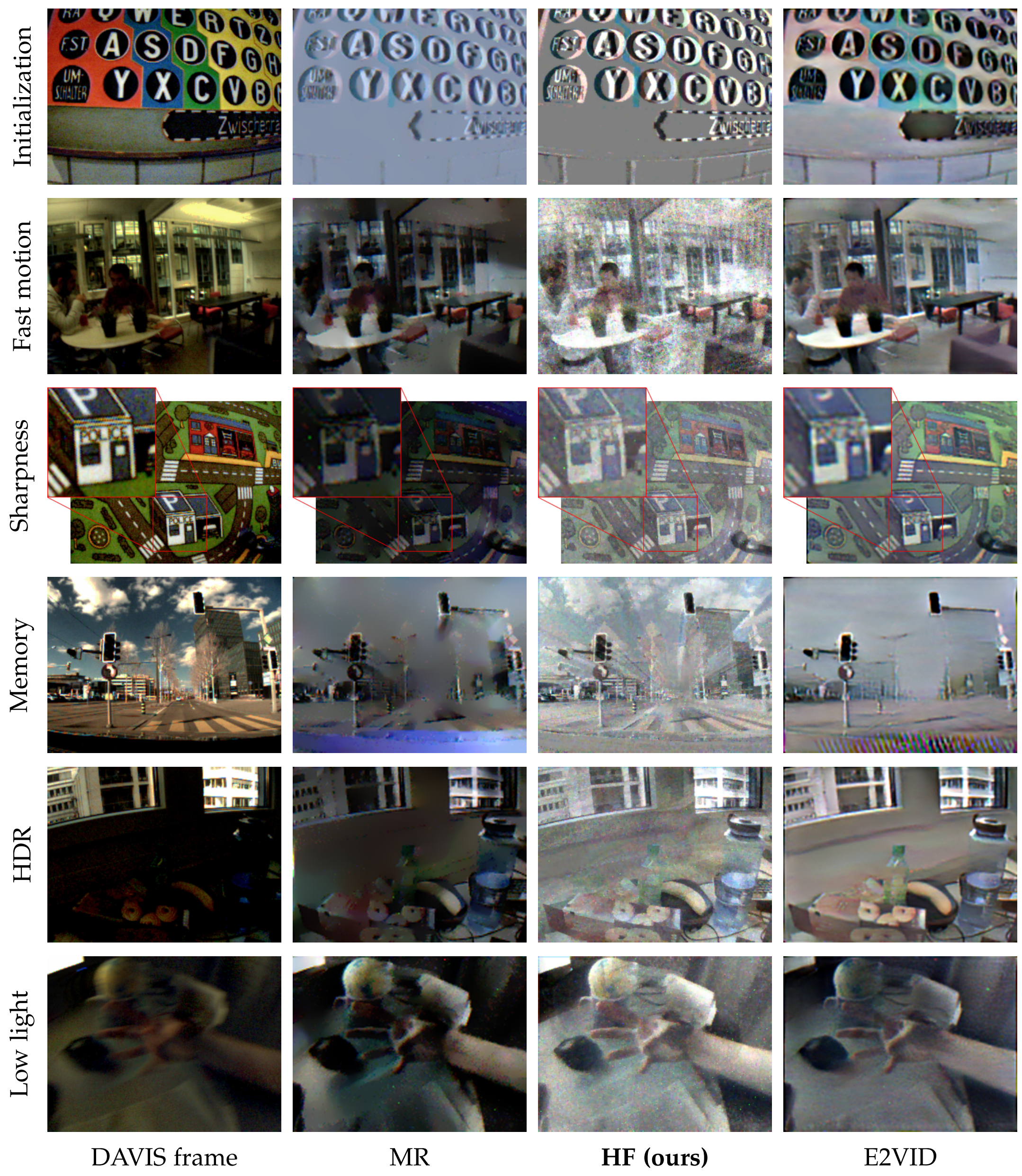


Figure 3.9: Edge cases for different reconstruction methods. First row: initialisation, all method but E2VID [Rebecq et al., 2019] fail. Second row: fast motion, HF and MR [Reinbacher et al., 2016] accumulate noise. Third row: zoom on carpet, HF preserves fine details better. Fourth row: Low apparent motion e.g., in the sky, HF preserves slow moving objects better. Fifth row: HDR scene, DAVIS cannot capture entire intensity range, reconstructions can. Sixth row: dark room (2 lux), DAVIS suffers motion blur, not the reconstructions.

Table 3.1: Overall performance of each reconstruction method on the ground truth dataset (Truck and Motorbike). Values are reported as mean ± standard deviation. The proposed method (CF) outperforms state-of-the-art on all metrics.

Truck sequence

Method	Photometric Error (%)	SSIM	FSIM
DI	12.25 ± 1.94	0.36 ± 0.07	0.93 ± 0.01
MR	16.81 ± 1.58	0.51 ± 0.03	0.96 ± 0.00
HF (ours)	15.67 ± 0.73	0.48 ± 0.03	0.95 ± 0.01
CF (**ours**)	$\mathbf{7.76 \pm 0.49}$	$\mathbf{0.57 \pm 0.03}$	$\mathbf{0.98 \pm 0.01}$

Motorbike sequence

Method	Photometric Error (%)	SSIM	FSIM
DI	11.78 ± 0.99	0.45 ± 0.05	0.94 ± 0.01
MR	14.53 ± 1.13	0.55 ± 0.05	0.94 ± 0.00
HF (ours)	15.14 ± 0.88	0.45 ± 0.03	0.94 ± 0.01
CF (**ours**)	$\mathbf{9.05 \pm 1.19}$	$\mathbf{0.58 \pm 0.04}$	$\mathbf{0.97 \pm 0.02}$

3.4.2 Quantitative Results

I evaluate the proposed method with (CF) and without (HF) 20Hz input-frames, and compare against DI [Brandli et al., 2014b] and MR [Reinbacher et al., 2016]. To assess similarity between ground truth and reconstructed images, each ground truth frame is matched with the corresponding reconstructed image with the closest time-stamp. Average absolute photometric error (%), structural similarity (SSIM) [Wang et al., 2004], and feature similarity (FSIM) [Zhang et al., 2011] are used to evaluate performance (Fig. 3.10 and Table 3.1). I initialise DI and CF using the first input-frame, and MR and HF to zero.

Fig. 3.10 plots the performance of each reconstruction method over time. The proposed method shows an initial improvement as useful information starts to accumulate, then maintains good performance over time as new events and frames are incorporated into the estimate. The oscillations apparent in DI arise from image frame resets. Table 3.1 summarises average performance for each sequence. The proposed method CF achieves the lowest photometric error, and highest SSIM and FSIM scores for all sequences. Fig. 3.11 shows the reconstructed image halfway between two input-frames of Truck ground truth sequence. Pure event-based methods (MR and HF) do not recover absolute intensity in some regions (truck body) due to sparsity of events. DI displays artifacts around edges, where many events are generated, because events are directly added to the latest input-frame. In CF, event and frame information is continuously combined, reducing edge artifacts (Fig. 3.11) and producing a more consistent estimate over time (Fig. 3.10).

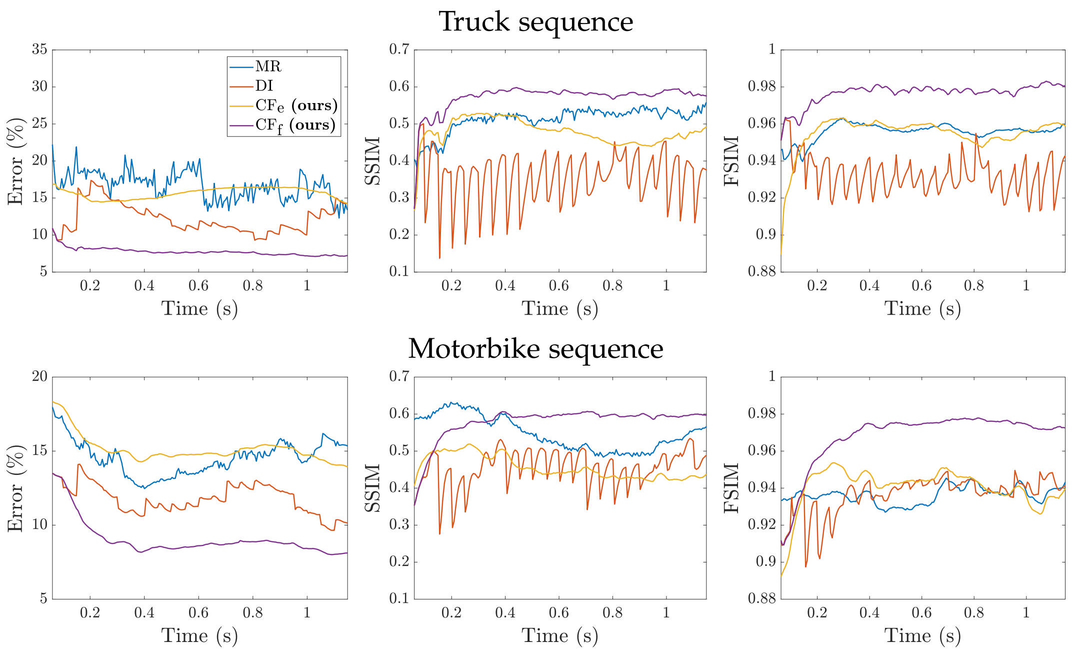


Figure 3.10: Full-reference quantitative evaluation of each reconstruction method on the ground truth datasets using photometric error (%), SSIM [Wang et al., 2004] and FSIM [Zhang et al., 2011].

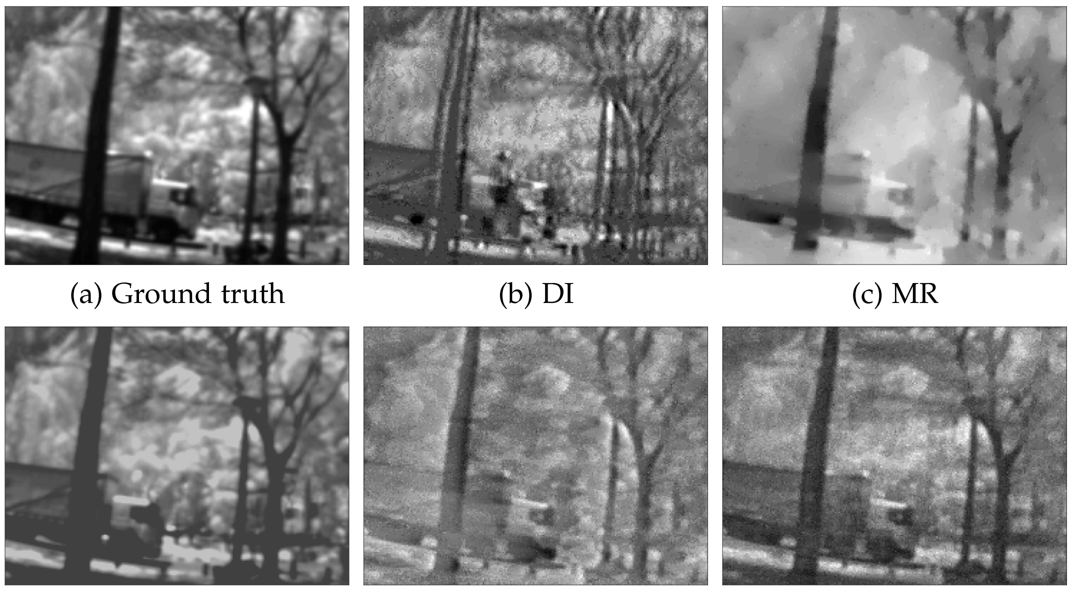


Figure 3.11: Reconstructed image for each method (DI [Brandli et al., 2014b], MR [Reinbacher et al., 2016], CF) on ground truth dataset (Truck) with raw input-frame (d) and ground truth (a) for comparison. DI (b) displays edge artifacts where new events are added directly to the latest input-frame. MR (c) produces smoothed images compared to CF (e), (f).

3.4.3 Computational performance

The complementary filter scales linearly with the number of events $O(n)$, thus I measure compute time per event. On an Intel i7 CPU on a desktop computer I measure a wall clock time of 50ns per event (20M events/s) and on an i5 laptop CPU, 0.14μs per event (7M events/s) using my open-source C++ code[5].

3.5 Conclusion

This chapter presented a continuous-time formulation for intensity estimation using an event-driven complementary filter. I compared complementary filtering with similar existing reconstruction methods on sequences recorded on a DAVIS camera, showing that the complementary filter outperforms competitors on a synthetic dataset [Scheerlinck et al., 2018]. The complementary filter naturally fuses image frames with events and can estimate intensity based on a pure event stream, by either setting the input-frame signal to zero, or by fusing events with the output of a different event-based image reconstruction method. Applications of event-based image reconstruction include event data visualisation, high speed, high dynamic range video capture or use in conjunction with other computer vision systems e.g., tracking [Gehrig et al., 2019b] or classification [Rebecq et al., 2020b].

[5]https://github.com/cedric-scheerlinck/dvs_image_reconstruction

Chapter 4

Asynchronous Spatial Image Convolutions

Website: https://cedricscheerlinck.com/event_convolutions

Based on [Scheerlinck et al., 2019a]

This chapter builds on linear filtering ideas introduced in Chapter 3, extending them from image reconstruction to asynchronous spatial image convolution and corner detection. Spatial convolution is arguably the most fundamental of 2D image processing operations. Conventional spatial image convolution can only be applied to a conventional image, that is, an array of pixel values (or similar image representation) that are associated with a single instant in time. Event cameras have serial, asynchronous output with no natural notion of an image frame, and each event arrives with a different timestamp. In this chapter, I propose a method to compute the convolution of a linear spatial kernel with the output of an event camera (Fig. 4.1). The approach operates on the event stream output of the camera directly without synthesising pseudo-image frames as is common in the literature. The key idea is the introduction of an internal state that directly encodes the convolved image information, which is updated asynchronously as each event arrives from the camera. The state can be read-off as-often-as and whenever required for use in higher level vision algorithms for real-time robotic systems. I demonstrate the application of the proposed method to corner detection, providing an implementation of an asynchronous Harris corner-response 'state' that can be used in real-time for feature detection and tracking on robotic systems.

4.1 Introduction

Spatial image convolutions are a core pre-processing step in almost all robotic vision algorithms. For example, Gaussian smoothing, gradient computation, computation of the Laplacian, etc, are convolutional operations that underlie fundamental vision algorithms such as: feature detection, optical flow computation, edge detection, etc. Classical image convolution requires a full image frame such as are generated by conventional synchronous cameras. Event cameras [Lichtsteiner et al., 2008; Brandli et al., 2014a] in contrast, provide asynchronous, data-driven measurements of

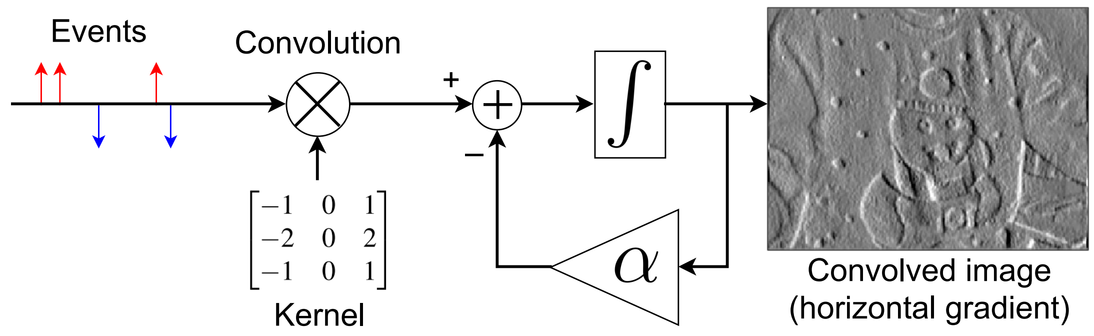


Figure 4.1: Asynchronous event convolution followed by a high-pass filter for convolved image reconstruction.

grayscale temporal contrast[2] at high temporal resolution and dynamic range. Event cameras have the potential to overcome many inherent limitations that conventional cameras exhibit for robotic applications: motion blur in high speed environments, under/overexposure in high dynamic range scenes, sparse temporal sampling (low frame rate), or high bandwidth and data requirements (high frame rate). With such advantages, event cameras are an ideal embedded visual sensor modality for robotic systems [Rebecq et al., 2017; Mueggler et al., 2015, 2014; Censi and Scaramuzza, 2014; Rosinol Vidal et al., 2018]. However, the lack of a conventional image frame means that any image processing algorithm that relies on convolution cannot be directly applied to the output of an event camera.

In this chapter, I propose a novel algorithm to compute the convolution of a linear kernel with the underlying radiometric scene information encoded by the output of an event camera. The key contribution of the chapter is the introduction of an internal 'state' that encodes the convolved image information. Each pixel of the internal state carries a timestamp of the last event that updated that pixel (analogous to the surface of active events [Benosman et al., 2014]), along with the latest state information, for example it could be values of: horizontal and vertical gradient, the Laplacian, or a Gaussian filtered intensity, etc.

The proposed algorithm uses continuous-time filter theory to compute a filtered or time-averaged version of the input event stream. Since spatial convolution is a linear process, it can be factored through the linear filter equations and applied directly to the event stream inputs. Thus, each event is spatially convolved with a linear kernel to generate a neighbouring collection of events, all with the same timestamp, which are then fed into pixel-by-pixel single-input-single-output continuous-time linear filters. The resulting filter equations can be solved explicitly, allowing asynchronous, discrete implementation of the continuous-time filter based on exact interpolation. Each asynchronous update of the internal state requires computation of one scalar exponential along with a small number of simple algebraic operations. The resulting algorithm does not require a motion-model for the camera and is truly

[2]I consider temporal contrast events (not gray-level events [Posch et al., 2011; Huang et al., 2017]).

asynchronous and highly efficient. The proposed method does not require reconstruction of pseudo-images, avoiding the latency and computational cost associated with synchronous reconstruction. The internal state can be separately read-off as-often-as and whenever required by a separate processing thread for use in higher level vision algorithms.

I demonstrate the approach using a variety of common kernels including Gaussian, Sobel and Laplacian kernels. To provide a more substantial example, I apply the method to the estimation of Harris corners. The approach taken is to augment the internal linear filter state with a (non-linear) Harris corner-response state. This 'state' is computed from the various gradients asynchronously as they are updated and provides a real-time measure of the Harris corner response of the underlying radiometric scene. The Harris corner state provides estimates of corners that I compare to a frame-based Harris detector, as well as state-of-the-art event-based corner detectors. I emphasise that in the proposed algorithm no grayscale image was required, or indeed is generated.

The remainder of the chapter is as follows: Section 4.2 outlines mathematical formulation and methodology. Section 4.3 presents experimental results and analysis. Section 4.4 concludes the chapter.

4.2 Method

The proposed method is formulated as a parallel collection of continuous-time filters that are solved asynchronously as discrete updates using exact interpolation. I compute the exact analytic solution to the associated ordinary differential equation of the filter in continuous time and evaluate at discrete time instances.

4.2.1 Mathematical Representation and Notation

Each pixel in the event camera responds independently and asynchronously to changes in brightness. When the change in log intensity relative to the previous reference level exceeds a preset threshold c,

$$|\log(I) - \log(I_{\text{ref}})| > c, \tag{4.1}$$

an event is triggered and the pixel reference I_{ref} resets to the new brightness level. For contrast event cameras [Lichtsteiner et al., 2008; Brandli et al., 2014a], each event contains the time-stamp (t; relative to a global clock), discrete pixel address $\boldsymbol{p} = (x, y)^T$, and polarity ($\sigma = \pm 1$ depending on the sign of the brightness change).

$$\text{event}_i = (t_i, \boldsymbol{p}_i, \sigma_i), \quad i \in 1, 2, 3... \tag{4.2}$$

The output of an event camera is a serial stream of asynchronous events.

Events can be modelled as Dirac-delta functions [Mueggler et al., 2017b]. Define an event $e_i(\boldsymbol{p}, t)$ as

$$e_i(\boldsymbol{p}, t) := \sigma_i \, c \, \delta(t - t_i) \, \delta_{\boldsymbol{p}_i}(\boldsymbol{p}), \tag{4.3}$$

where $\delta(t)$ is a Dirac-delta function and $\delta_{\boldsymbol{p}_i}(\boldsymbol{p})$ is a Kronecker delta function with indices associated with the pixel coordinates of $\boldsymbol{p}_i$ and $\boldsymbol{p}$. That is $\delta_{\boldsymbol{p}_i}(\boldsymbol{p}) = 1$ when $\boldsymbol{p} = \boldsymbol{p}_i$ and zero otherwise. In this chapter I use the common assumption that the contrast threshold c is constant [Reinbacher et al., 2016; Kim et al., 2016, 2014], although, in practice it does vary somewhat with intensity, event-rate and other factors [Brandli et al., 2014b]. The integral of events is

$$\int_0^t \sum_i e_i(\boldsymbol{p}, \tau) d\tau = L(\boldsymbol{p}, t) - L(\boldsymbol{p}, 0) + \int_0^t \eta(\boldsymbol{p}, \tau) d\tau, \tag{4.4}$$

where $L(\boldsymbol{p}, t)$ is the log intensity seen by the camera with initial condition $L(\boldsymbol{p}, 0)$, and $\eta(\boldsymbol{p}, t)$ represents quantisation and sensor noise. $L(\boldsymbol{p}, 0)$ is typically unknown and $\eta(\boldsymbol{p}, t)$ is unknown and poorly characterised. If left unchecked, integrated error arising from $\int_0^t \eta(\boldsymbol{p}, \tau) d\tau$ grows over time and quickly degrades the estimate of $L(\boldsymbol{p}, t)$ [Brandli et al., 2014b]. A method to deal with error arising from $L(\boldsymbol{p}, 0)$ and $\eta(\boldsymbol{p}, t)$ will be presented in section 4.2.3.

4.2.2 Event Convolutions

Let K denote a linear spatial kernel with finite support. Consider the convolution of K with $L(\boldsymbol{p}, t)$. Define

$$L^K(\boldsymbol{p}, t) := (K * L)(\boldsymbol{p}, t). \tag{4.5}$$

Using (4.3), (4.4) and omitting the noise term $\eta(\boldsymbol{p}, t)$ in the approximation

$$\begin{aligned} L^K(\boldsymbol{p}, t) &\approx (K * L)(\boldsymbol{p}, 0) + \int_0^t \sum_i (K * e_i)(\boldsymbol{p}, \tau) d\tau, \\ &\approx (K * L)(\boldsymbol{p}, 0) + \int_0^t \sum_i \sigma_i \, c \, \delta(t - t_i) \, (K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p}) d\tau, \\ &\approx (K * L)(\boldsymbol{p}, 0) + \int_0^t \sum_i e_i^K(\boldsymbol{p}, \tau) d\tau, \end{aligned} \tag{4.6}$$

where

$$e_i^K(\boldsymbol{p}, t) := \sigma_i \, c \, \delta(t - t_i) \, (K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p}). \tag{4.7}$$

Note that $(K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p})$ is a local spatial convolution of the finite support kernel K with a single non-zero image pixel (Fig. 4.2). The result of such a convolution is an image with pixel values of zero everywhere except for a patch centred on $\boldsymbol{p}_i$ (the same size as K) with values drawn from the coefficients of K. The convolved event $e_i^K(\boldsymbol{p}, t)$ can

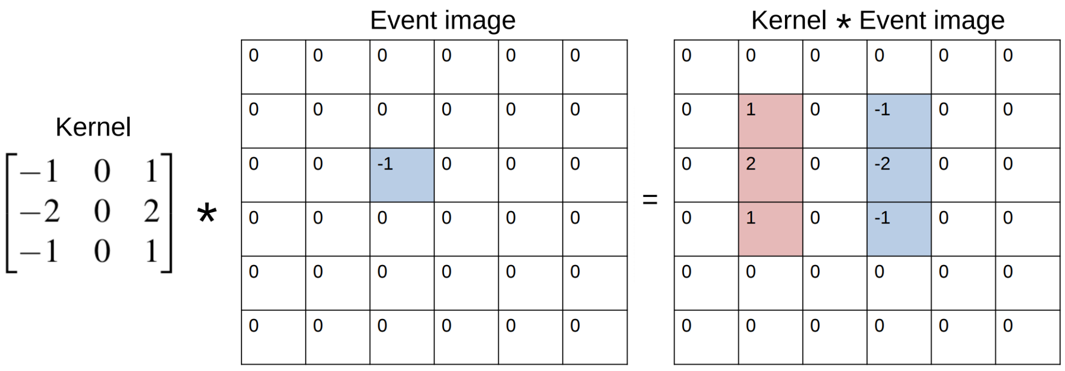

Kernel
$\begin{bmatrix}-1 & 0 & 1\\-2 & 0 & 2\\-1 & 0 & 1\end{bmatrix}$
Event image
Kernel ⋆ Event image

Figure 4.2: Convolution of a single event with horizontal Sobel kernel generates six convolved events.

be thought of as a finite (localised) collection of spatially separate events all occurring at the same time t_i.

4.2.3 Continuous-time Filter for Convolved Events

It is possible to compute the direct integral (4.6) using a similar approach to the direct integration schemes of Brandli et al. [2014b]; Reinbacher et al. [2016]. The drawback of this approach is integration of sensor noise, which results in drift, and undermines low temporal-frequency components of the estimate $L^K(\boldsymbol{p}, t)$ over time. Furthermore, I am often concerned with high temporal-frequency information (i.e. scene dynamics), especially in robotic systems scenarios where the scene around the robot is changing continually. This leads us to consider a simple high-pass filtered version of $L^K(\boldsymbol{p}, t)$.

Frequency domain: I design the high-pass filter in the frequency domain, and later implement it in the time domain via inverse Laplace transform. For $\alpha > 0$, a scalar constant, I define a high pass filter $F(s) := s/(s+\alpha)$ and apply it directly to the integrated event stream (4.6). Let $\mathcal{L}^K(\boldsymbol{p}, s)$ denote the Laplace transform of the signal $L^K(\boldsymbol{p}, t)$. Let $\hat{\mathcal{G}}(\boldsymbol{p}, s)$ denote the high-pass filtered version of $\mathcal{L}^K(\boldsymbol{p}, s)$. That is:

$$\begin{aligned}\hat{\mathcal{G}}(\boldsymbol{p}, s) &:= \frac{s}{s+\alpha}\mathcal{L}^K(\boldsymbol{p}, s),\\ &= \frac{s}{s+\alpha}\frac{1}{s}\sum_i \mathcal{E}_i^K(\boldsymbol{p}, s) + \frac{s}{s+\alpha}\frac{1}{s}(K * L)(\boldsymbol{p}, 0),\\ &= \frac{1}{s+\alpha}\sum_i \mathcal{E}_i^K(\boldsymbol{p}, s) + \frac{1}{s+\alpha}(K * L)(\boldsymbol{p}, 0), \qquad (4.8)\end{aligned}$$

where $\mathcal{E}_i^K(\boldsymbol{p}, s) = \sigma_i\, c\, \exp(-t_i s)(K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p})$ is the Laplace transform of $e_i^K(\boldsymbol{p}, t)$. The DC term associated with the unknown initial condition has an exponentially decreasing time-response $e^{-\alpha t}(K * L)(\boldsymbol{p}, 0)$ in the filter state and is quickly attenuated. The high-pass filter naturally attenuates low-frequency components of the noise signal $\eta(\boldsymbol{p}, t)$.

Time domain: Ignoring the $(K * L)(\boldsymbol{p}, 0)$ initial condition, the time domain signal $\hat{G}(\boldsymbol{p}, t)$ can be computed by taking the inverse Laplace of (4.8) and solving the resulting ordinary differential equation[8]

$$\frac{\partial}{\partial t}\hat{G}(\boldsymbol{p}, t) = -\alpha \hat{G}(\boldsymbol{p}, t) + \sum_i e_i^K(\boldsymbol{p}, t), \tag{4.9}$$

for each pixel $\boldsymbol{p}$. Here, $\hat{G}(\boldsymbol{p}, t)$ is a pixel-by-pixel internal state that provides an estimate of the high-pass component of $(K * L)(\boldsymbol{p}, t)$.

The continuous-time differential equation (4.9) is a constant coefficient linear differential equation except at time instances when an event occurs and can be solved explicitly. To exploit this property I store the timestamp of the latest event at each pixel $t^{\boldsymbol{p}}$ and use the explicit solution of (4.9) to asynchronously update the state when (and only when) a new event at that pixel occurs.

The constant-coefficient, first-order ODE for (4.9) assuming no events is

$$\frac{\partial}{\partial t}\hat{G}(\boldsymbol{p}, t) = -\alpha \hat{G}(\boldsymbol{p}, t). \tag{4.10}$$

Let t_i denote the timestamp of the current event and denote the limit to t_i from below by t_i^- and the limit to t_i from above by t_i^+. Integrate (4.10) from $t^{\boldsymbol{p}}$ (the timestamp of the previous event at $\boldsymbol{p}$) to t_i^-

$$\hat{G}(\boldsymbol{p}, t_i^-) = \exp(-\alpha(t_i - t^{\boldsymbol{p}}))\hat{G}(\boldsymbol{p}, t^{\boldsymbol{p}}). \tag{4.11}$$

Next integrate (4.9) over the convolved event, i.e. from t_i^- to t_i^+

$$\int_{t_i^-}^{t_i^+} \frac{\partial}{\partial t}\hat{G}(\boldsymbol{p}, t) dt = \int_{t_i^-}^{t_i^+} -\alpha \hat{G}(\boldsymbol{p}, t) + e_i^K(\boldsymbol{p}, t) dt.$$

The integral of the right-hand side is $\sigma_i\, c\, (K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p})$ since $\hat{G}(\boldsymbol{p}, t)$ is continuous and its infinitesimal integral is zero, and the Dirac delta integrates to unity. Thus, one has

$$\hat{G}(\boldsymbol{p}, t_i^+) = \hat{G}(\boldsymbol{p}, t_i^-) + \sigma_i\, c\, (K * \delta_{\boldsymbol{p}_i})(\boldsymbol{p}). \tag{4.12}$$

In addition, it is necessary to update the timestamp state

$$t^{\boldsymbol{p}} = t_i. \tag{4.13}$$

Equations (4.10), (4.12) and (4.13) together define an asynchronous distributed update that can be applied pixel-by-pixel to compute the filter state. The state could also be updated at any user-chosen time-instance (for example just before a read-out) with the time-instance stored in $t^{\boldsymbol{p}}$.

Multiple different filters can be run in parallel. For example, if gradient estima-

[8]Although I write this as a partial differential equation (the partial taken with respect to time) there is no coupling between pixel locations and the solution decouples into parallel pixel-by-pixel ODEs.

tion is required, then two filter states $(\hat{G}_x, \hat{G}_y)$ can be run in parallel for the x and y components using an appropriate directional kernels (Sobel, central difference, etc).

4.3 Results

The experiments were performed using a DAVIS240C [Brandli et al., 2014a] with default biases provided in the jAER software, and sequences from [Scheerlinck et al., 2018] and [Mueggler et al., 2017b]. The internal filter state of the system is asynchronous and for visualisation I display instantaneous snap shots taken at sample times. There is only a single parameter α in the filter. I set $\alpha = 2\pi\,\text{rad/s}$ for all sequences unless otherwise stated. The complexity of the proposed algorithm scales linearly with the number of (non-zero) elements in the kernel, and I find that a kernel size of 3×3 is usually sufficient. I fix the contrast threshold c constant.

4.3.1 Event Convolutions

Figure 4.3 displays a range of different filtered versions of an input sequence (sun and night drive are taken from [Scheerlinck et al., 2018]). The first row of Figure 4.3 shows the application of the identity kernel. This kernel returns a (temporal) high-pass filtered version of the original image. The sequences that follow, for a range of different kernels, are generated directly from events using the proposed algorithm and convincingly appear as one would expect if the kernel had been applied to the image reconstruction from the top row. The key advantage of the proposed approach is that is does not incur latency or additional computation associated with reconstruction. The sequences in Fig 4.3 are:

- Snowman *(left)*: The author wearing a knitted jumper with prominent snowman and snowflakes design (taken under normal office conditions).
- Sun *(centre)*: Looking directly at the sun through the trees. Exemplifies high dynamic range performance of the camera.
- Night drive *(right)*: Country road at night with no street lights or ambient lighting, only headlights. The car is travelling at 80km/h causing considerable motion in the scene. Exemplifies performance in high-speed, low-light conditions.

Despite noise in the event stream, the proposed approach reproduces a high-fidelity representation of the scene. It is particularly interesting to note the response for the two challenging sequences sun and night drive. In both cases the image is clear and full of detail, despite the high dynamic range of the scene.

The second row computes a (spatial) low pass Gaussian filter of the sequences. The low pass nature of the response is clear in the image. The authors note that if it was desired to compute an image pyramid then it is a straightforward generalisation of the filter equations to reduce the state dimension at a particular level of the image

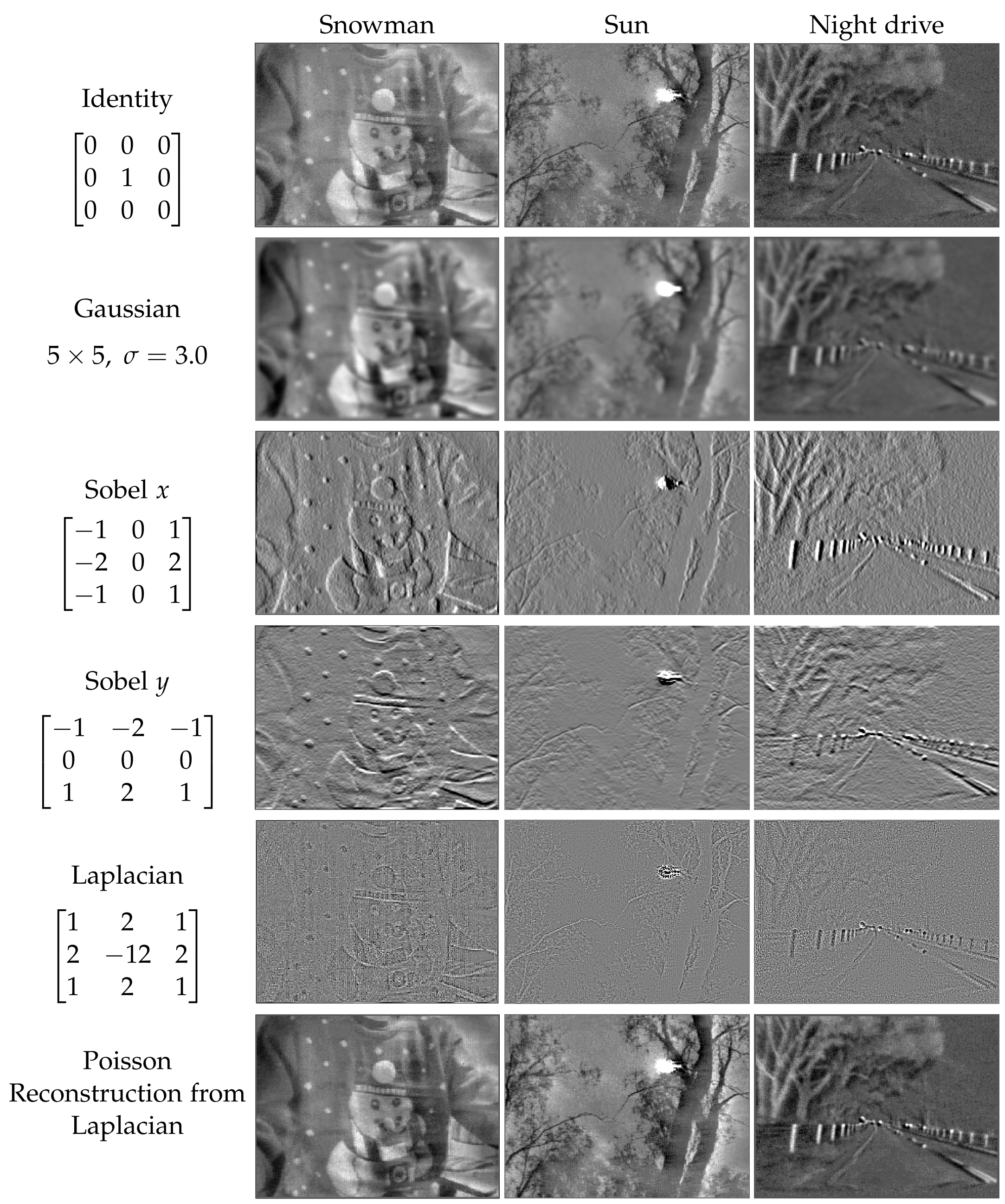

Snowman
Sun
Night drive
Identity
$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$
Gaussian
5×5, $\sigma = 3.0$
Sobel x
$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$
Sobel y
$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$
Laplacian
$\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$
Poisson Reconstruction from Laplacian

Figure 4.3: Different kernels K applied to events using high-pass filter (4.9). Sun demonstrates robustness in extreme dynamic range scenarios and night drive is captured in pitch black conditions, demonstrating excellent performance in low-light settings thanks to the event camera.

pyramid by linear combination of pixel values. The resulting filter would still be linear and the same filter equations would apply.

The third and fourth rows display the internal filter state for the Sobel kernels in both vertical and horizontal directions. The results show that the derivative filter state is operating effectively even in very low light and high dynamic range conditions.

Rows five and six display the Laplacian of the image (the sum of second derivatives of the image) and a Poisson reconstruction built from the Laplacian image. The Laplacian kernel computes an approximation of the divergence of the gradient vector field. It can be used for edge detection: zero crossings in the Laplacian response correspond to inflections in the gradient and denote edge pixels. It is also possible to reconstruct an original (log) intensity image from a Laplacian image using Poisson solvers [Agrawal et al., 2005, 2006]. In this case, I present the Poisson reconstruction of the Laplacian image (row six) primarily to verify the quality of the filter response.

It is important to recall that the internal state of the filter is computed directly from the event stream in all these cases. For example, if only the Laplacian is required then there is no need to compute a gray scale image or gradient image.

4.3.2 Continuous Harris Event Corners (CHEC)

To demonstrate a practical application of the proposed filter I consider applying the image processing framework to detection of Harris corners. I compute a continuous-time 2D state asynchronously that encodes the Harris-corner-response [Harris and Stephens, 1988] of pixels. Image gradients are computed using the filter architecture proposed in equations (4.10), (4.12) and (4.13) for Sobel kernels K_x and K_y. When a pixel gradient is updated then Harris corner response at that pixel location is also recomputed. A threshold is applied at the pixel level and then non-maximum suppression is applied locally to determine individual corners. I call corners detected using this algorithm *Continuous Harris Event Corners* (CHEC) since they are derived from a continuous-time Harris response image state. A key advantage of the proposed approach is that I am able to update the corner-response state asynchronously with each event, rather than having to synchronously update the entire state.

Let $\hat{G}(\boldsymbol{p},t) = \begin{bmatrix}\hat{G}_x(\boldsymbol{p},t) & \hat{G}_y(\boldsymbol{p},t)\end{bmatrix}^\top$ denote an internal gradient state (4.9). The Harris matrix is

$$M(\boldsymbol{p},t) := W * \hat{G}(\boldsymbol{p},t)\hat{G}(\boldsymbol{p},t)^T. \tag{4.14}$$

where W is a smoothing kernel, e.g. box or Gaussian. The Harris corner-response [Harris and Stephens, 1988] is

$$R(\boldsymbol{p},t) := \det(M(\boldsymbol{p},t)) - \gamma \operatorname{trace}(M(\boldsymbol{p},t))^2, \tag{4.15}$$

where γ is an empirically determined constant, in this case $\gamma = 0.04$.

Figure 4.4 shows the continuous-time Harris corner-response state (4.15) computed from the gradient state estimate (4.9), on real sequences from [Scheerlinck

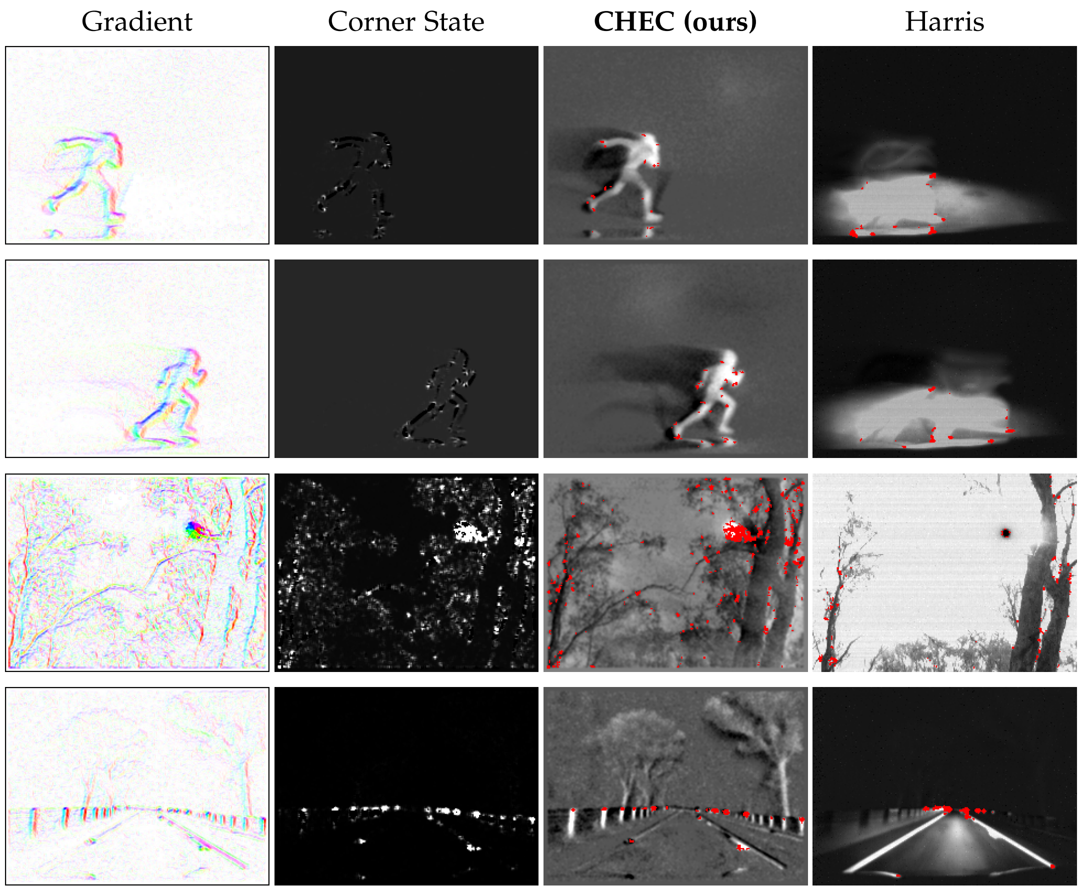


Figure 4.4: Harris corner-response: Continuous Harris Event Corners (CHEC) applied to events verses conventional Harris applied to raw camera frames. Gradient shows a snap shot of the internal gradient state, obtained by applying Sobel x and y kernels directly to events. Corner State shows a snap shot of the full Harris corner-response state (4.15) computed from only the gradient state. CHEC (ours) shows the corner-response thresholded at a suitable value and superimposed onto a log intensity image, obtained via Poisson reconstruction [Agrawal et al., 2005, 2006] of Gradient. Note: log intensity is displayed purely for visualisation and is not used to compute corners. Harris shows the Harris detector [Harris and Stephens, 1988] applied to raw image frames for comparison.

et al., 2018]. Column one shows the image gradient state. Column two shows the Harris corner-response state as a continuous-valued image state. Column three shows the binary output after applying a threshold to column two, overlaid on log intensity of the image (obtained via post processing of the stored gradient state and Poisson reconstruction) for visualisation purposes. I emphasise that the image intensity was not required and not computed. The final column shows the raw conventional frames and the binary output of the thresholded classical Harris response [Harris and Stephens, 1988].

Night_run (top row Fig 4.4) is captured in pitch black conditions as someone runs in front of headlights of a car. The conventional camera suffers extreme motion blur because of the high exposure-time required in low-light conditions. The proposed approach leverages the high-dynamic-range, high-temporal-resolution event camera, yielding crisp edges and corners. Sun (third row) displays artifacts in the corner state around the sun because of extreme brightness gradients caused by the high-dynamic range of the sun on a cloudless day, where the event camera is pushed to the limit. Nevertheless, I still get clear corners around the branches and leaves of the trees, whereas the conventional camera frame is largely over-saturated. Night drive (last row) demonstrates performance under challenging high-speed, low-light conditions. The proposed approach clearly detects corners on roadside poles and road-markings. The conventional camera frame is highly blurred and unable to detect corners in much of the image.

Figure 4.5 compares state-of-the-art event-based corner detectors [Vasco et al., 2016; Mueggler et al., 2017a; Alzugaray and Chli, 2018], as well as frame-based Harris detector (Harris) [Harris and Stephens, 1988], against the proposed method (CHEC), on real sequences from the event camera dataset [Mueggler et al., 2017b] (shapes_translation and dynamic_6dof). The corners identified are overlaid on the raw camera frame to improve visualisation of the results. eHarris was first developed by Vasco et al. [2016], and later improved by Mueggler et al. [2017a]. I use improved eHarris code implementation of Mueggler et al. [2017a]. I also compare against FAST event-based corner detector [Mueggler et al., 2017a] and ARC (asynchronous event-based corner detection) [Alzugaray and Chli, 2018]. For state-of-the-art I use default parameters provided in the open-source code. For CHEC, I increase the filter gain to $\alpha = 10\,\text{rad/s}$ to reduce low-temporal-frequency noise. To extract corners from the Harris response of both Harris and CHEC, I first threshold, then apply non-maximum suppression.

In simple low-texture environments (such as shapes) each method performs well finding similar points. In contrast, in high-texture environments (dynamic), state-of-the-art event-based detectors tend to find many spurious corners. The identification of too many corners is as much a problem for image processing pipelines as the failure of an algorithm to detect good corners. The CHEC detector demonstrates a very similar response to the classical Harris algorithm on large sections of the image. Points identified appear to be well correlated with visual corners in the image and correlate well with the corners identified by the classical Harris corner detector. I emphasise that the two algorithms use completely separate data, the CHEC algorithm

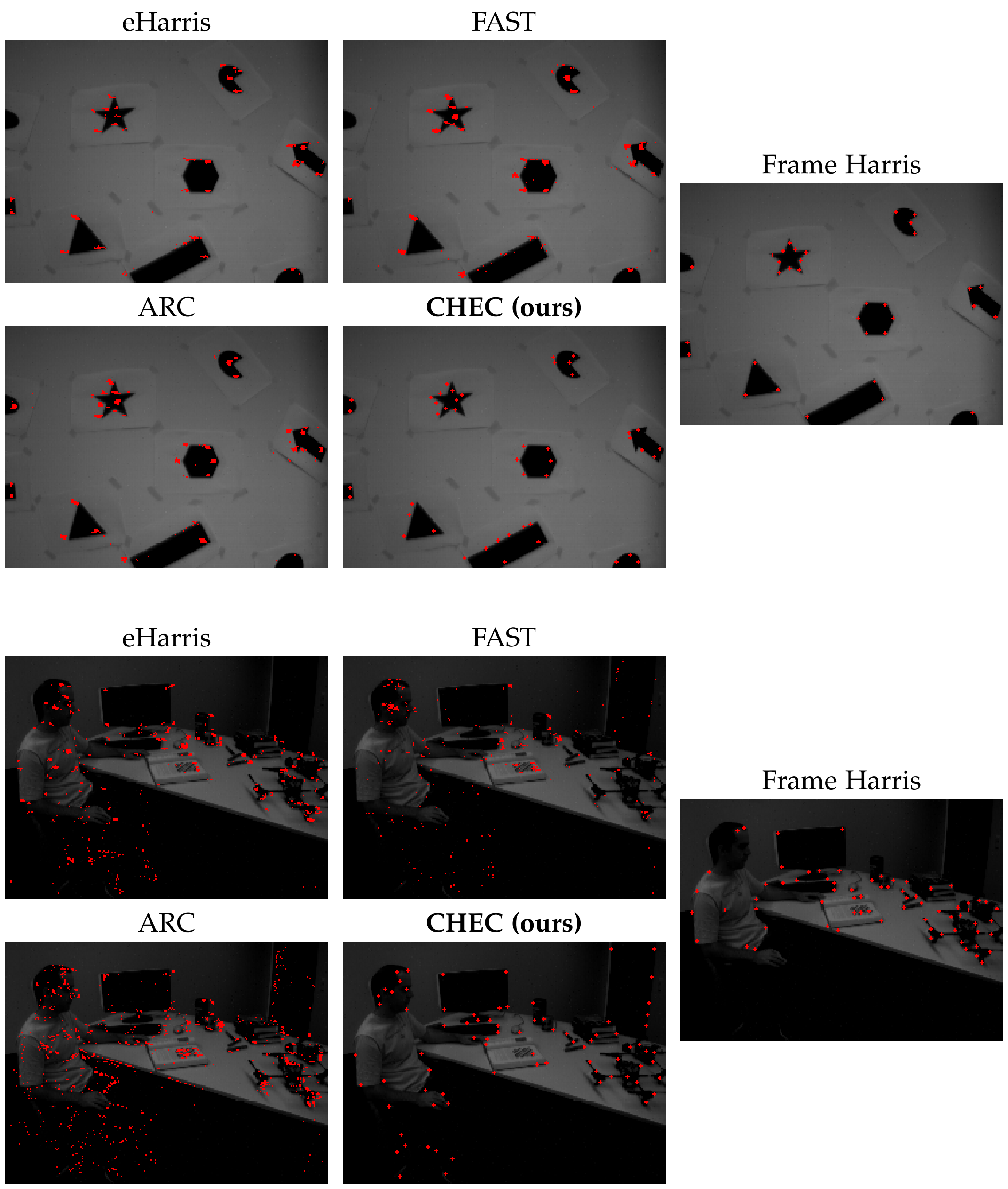


Figure 4.5: Top: shapes, bottom: dynamic. I plot the raw camera frame for visualisation, though it is not used in any event-based corner detection; eHarris [Mueggler et al., 2017a; Vasco et al., 2016], FAST [Mueggler et al., 2017a], ARC [Alzugaray and Chli, 2018] and CHEC (ours). Corners appear shifted due to the low temporal resolution of the raw frame. Since eHarris, FAST and ARC provide asynchronous corner events, I accumulate the last 30ms for visualisation. Harris computes corners from the raw frames, which are subject to low temporal resolution and limited dynamic range.

uses the event stream while the classical Harris algorithm is using the conventional frame output of the DAVIS camera.

On areas of high dynamic range, such as under the chair, on the first authors face, and in the top right of the image, the classical Harris algorithm is unable to extract sufficient contrast to generate an effective corner response while the CHEC algorithm functions effectively. Furthermore, the output of the CHEC algorithm will not suffer from image blur and can be computed asynchronously in real time, providing an ideal front end corner detector for real-world robotic systems.

4.4 Conclusion

This chapter introduced a method to compute asynchronous spatial convolutions for event cameras. Exploiting sparsity in asynchronous convolution offers potential reduction in the number of operations compared to conventional convolution [Serrano-Gotarredona et al., 2006; Camunas-Mesa et al., 2012]. However, not all operations are equal in cost; for example, memory access is relatively more expensive than other mathematical operations such as multiplication. Conventional hardware (such as CPU used in this chapter) cannot easily take advantage of memory reuse, e.g., weights and states that could be reused hundreds of times in overlapping operations [Serrano-Gotarredona et al., 2009]. Thus, specialised hardware such as neuromorphic and sparsity-aware hardware accelerators [Delbruck and Liu, 2019; Aimar et al., 2018] are required to realise the full benefits of asynchronous computation. Despite the potential challenges, the framework of asynchronous convolution is of significant interest in the context of neuromorphic hardware such as IBM TrueNorth [Merolla et al., 2014], Intel Loihi [Davies et al., 2018] and SpiNNaker Furber et al. [2014]. This chapter has provided a theoretical framework for asynchronous algorithms based on filter-theory and is extendable to future hardware implementation. A key feature is the continuous-time internal state that encodes convolved image information and allows asynchronous, event-driven, incremental updates. I extended the concept of an internal state to a Harris corner-response state, and demonstrated corner detection (CHEC) without requiring intensity. I believe there are many exciting possibilities in this direction, including alternative feature states, continuous-time optical flow state, and application of event-based convolutions to convolutional neural networks, similar to [Cannici et al., 2019; Messikommer et al., 2020].

Chapter 5

Machine Learning

Website: https://cedricscheerlinck.com/firenet

Based on [Scheerlinck et al., 2020]

This chapter takes a step back from asynchronous, per-event processing to explore ideas for applying machine learning and convolutional neural networks to event cameras, in pursuit of high quality video reconstruction. Event cameras are powerful new sensors able to capture high dynamic range with microsecond temporal resolution and no motion blur. Their strength is detecting brightness changes (called events) rather than capturing direct brightness images; however, algorithms can be used to convert events into usable image representations for applications such as classification. Previous works rely on hand-crafted spatial and temporal smoothing techniques to reconstruct images from events. State-of-the-art video reconstruction has recently been achieved using neural networks that are large (10M parameters) and computationally expensive, requiring 30ms for a forward-pass at 640 $\times$ 480 resolution on a modern GPU. I propose a novel neural network architecture for video reconstruction from events that is smaller (**38k** vs. 10M parameters) and faster (**10ms** vs. 30ms) than state-of-the-art with minimal impact to performance.

5.1 Introduction

The introduction of machine learning to event cameras has caused a proliferation of works, achieving state-of-the-art results in optical flow [Zhu et al., 2018b, 2019], 6-DOF pose relocalisation [Nguyen et al., 2019], steering prediction [Maqueda et al., 2018], classification [Gehrig et al., 2019a], segmentation [Alonso and Murillo, 2019], image reconstruction [Rebecq et al., 2019] and more. These methods typically convert raw events into time-surfaces, event images or voxel-grids to be passed to a convolutional neural network (CNN). Large CNN models can be memory and computationally intensive, consuming power and hampering the low latency of event cameras. This makes it harder to deploy large models on embedded platforms or IoT applications with power and memory constraints, where event cameras are ideal candidates due to their low power and bandwidth consumption. Reducing the model size can

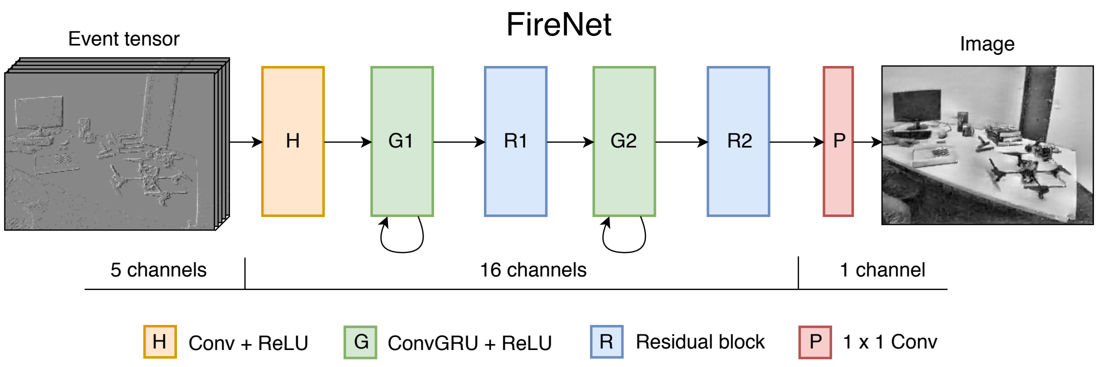


Figure 5.1: FireNet architecture. The input is an event tensor with 5 temporal bins. The network consists of convolutional layers (H, P), convolutional gated recurrent units (G1, G2) and residual blocks (R1, R2). Every layer uses ReLU activation except the final layer (P).

improve performance by reducing (i) memory footprint, (ii) FLOPs and power consumption and (iii) latency.

In this chapter, I introduce FireNet (Fig. 5.1): a novel neural network architecture that performs *fast image reconstruction from events*. FireNet is significantly smaller than state-of-the-art (E2VID) [Rebecq et al., 2019], requiring fewer parameters (**38k** vs 10M), less memory (**0.16Mb** vs 43Mb) and fewer FLOPs (**12.6G** vs 147.2G), and runs three times faster than E2VID [Rebecq et al., 2019] on a modern GPU. FireNet is a fully convolutional network that relies on recurrent connections to build a state over time, allowing a much smaller network that re-uses previous computed results, showing exciting potential for tiny recurrent networks that run fast.

5.2 Method

5.2.1 Input Representation

The input to FireNet is a $H \times W \times B$ event tensor (a.k.a., voxel grid) $V(x, y, t)$ proposed by Zhu et al. [2019], where H, W are the sensor height and width and B is the number of temporal bins. The event tensor is populated using trilinear voting (interpolation) where each event $(x_i, y_i, t_i, \sigma_i)$ contributes its polarity to its two closest temporal bins according to:

$$V(x, y, t_n) = \sum_i \sigma_i \max\left(0, 1 - |t_n - t_i^*|\right), \tag{5.1}$$

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

where n is the temporal bin index, σ_i is the polarity and t_i^* is the normalised timestamp of the i^{th} event. I use $B = 5$.

At runtime I pass consecutive, non-overlapping event tensors with a fixed number of N events. Thus, t_{min} and t_{max} may be different for each event tensor, depending on the timing of events. The input representation can be interpreted as adaptively rescaling the temporal dimension, i.e., the network never sees absolute timestamps, only relative event timings. This scheme can be relaxed e.g., having a variable number of events per event tensor. In addition, I normalise the non-zero entries in $V(x, y, t)$ to have zero mean and unit norm, mitigating the impact of unbalanced ON/OFF contrast thresholds, and making the network robust against different magnitudes of contrast threshold.

5.2.2 Architecture

FireNet is a convolutional recurrent neural network (Fig. 5.1). All layers use single-strided (no downsampling) 3×3 convolutions, inspired by Simonyan and Zisserman [2015], who demonstrate several layers of small convolutional filters outperforms prior-art configurations of larger filters. An exception is the final layer which is 1×1 to convert the penultimate feature map into a single-channel image. The head unit (H) consists of a 16-channel convolution ($y = w * x + b$) with ReLU activation [Nair and Hinton, 2010], a smaller version of the head unit described in [Rebecq et al., 2020b] (16 vs. 32 channels, 3×3 vs. 5×5 kernel). The convolutional gated recurrent units (G1, G2) consist of a 16-channel convolution with ReLU activation followed by a gated recurrent unit as described in [Ballas et al., 2016]. I chose GRUs instead of LSTMs because they have been shown to exhibit similar performance [Chung et al., 2014] while having less parameters (two gates instead of three). The residual blocks (R1, R2) use 16-channel convolutions with ReLU activation and skip connections as described in [He et al., 2016]. The final prediction layer (P) is a 1×1 single-channel convolution. The output is one image per input event tensor. Table 5.1 shows key differences between FireNet and E2VID [Rebecq et al., 2019]. A valid interpretation is that FireNet is a drastically reduced version of E2VID, without down/up-sampling (UNet), and some structural changes (e.g., no outer skip connections).

5.2.3 Training

To make a fair comparison to E2VID, the exact same training and validation data was used. The data was generated by the event simulator *ESIM* [Rebecq et al., 2018], and consists of 1,000 sequences of 2 seconds each (950 training, 50 validation). MS-COCO images [Lin et al., 2014b] were mapped to a 3D plane and random 6-DOF (simulated) camera motions were used to trigger events. To simulate contrast threshold mismatch and refractory, contrast threshold values (ON/OFF) for each sequence were drawn from a normal distribution with mean $\mu = 0.18$ and standard deviation $\sigma = 0.03$ and a refractory period of 1ms was applied after each event.

As in [Rebecq et al., 2020a], I used both (i) a *reconstruction loss* that measures the difference between the reconstruction and groundtruth image, and (ii) a *temporal loss* that penalises differences between consecutive reconstructed images. I used

Table 5.1: Network overview. Compared to E2VID [Rebecq et al., 2019, 2020a], FireNet has 280× fewer parameters, consuming only 0.37% of the memory.

	E2VID	Ours
No. parameters (k)	10700	38
Memory (Mb)	43	0.16
Downsampling	yes	no
Recurrent units	LSTM	GRU
Max. kernel size	5×5	3×3

Perceptual Similarity (LPIPS) [Zhang et al., 2018] to a groundtruth image as the reconstruction loss $\mathcal{L}_k^R = d(\hat{\mathcal{I}}_k, \mathcal{I}_k)$, where d is the LPIPS distance function, $\hat{\mathcal{I}}_k$ is the k^{th} reconstructed image and $\mathcal{I}_k$ is the groundtruth image. The groundtruth image was selected by matching its timestamp to the latest event in the input event tensor, thus, discouraging the network from predicting images in the past. I used the *temporal consistency loss* described in [Rebecq et al., 2020a] that aligns two successive reconstructed images based on the optical flow between them and measures a photometric error $\mathcal{L}_k^{TC} = c(\hat{\mathcal{I}}_{k-1}, \hat{\mathcal{I}}_k)$, where c is the temporal consistency function.

The final loss is a weighted sum of reconstruction and temporal losses over L consecutive images

$$\mathcal{L} = \sum_{k=0}^{L} \mathcal{L}_k^R + \lambda_T C \sum_{k=L_0}^{L} \mathcal{L}_k^{TC}, \tag{5.2}$$

where $L = 20$, $\lambda_{TC} = 2$ and $L_0 = 10$. I used the ADAM optimiser [Kingma and Ba, 2015] with default parameters, learning rate 1e-4, and trained for 1000 epochs.

5.3 Results

5.3.1 Overview

FireNet is 280× smaller than E2VID [Rebecq et al., 2019, 2020a] (Tab. 5.1) with only 38k parameters (0.36%) and consuming only 160kb of memory (0.37%). This yields a 3× speedup on GPU, 4× on CPU and 10× reduction in the number of FLOPs (Tab. 5.2). FireNet's accuracy on the event camera dataset [Mueggler et al., 2017b] is comparable to E2VID (Tab. 5.3). Qualitative comparison confirms that reconstructed images are of a similar quality to E2VID (Fig. 5.2), though in some challenging scenarios slightly worse (Fig. 5.7). I compared against the latest version of E2VID [Rebecq et al., 2020a] for all experiments.

Table 5.2: Computational cost. I report inference time on GPU and CPU, and the number of FLOPs for a single forward-pass at common sensor resolutions. 10× gap in FLOPs unmatched by 3× gap in GPU runtime due to efficient parallel computation. FLOPs/pix depends on aspect ratio and size that impacts efficiency of convolutional padding and down/up-sampling for E2VID.

Resolution	GPU (ms)		CPU (ms)		FLOPs (G)		FLOPs/pix ($\times10^{-5}$)	
	E2VID	Ours	E2VID	Ours	E2VID	Ours	E2VID	Ours
240 × 180	5.52	**1.89**	84.98	**22.86**	21.2	**1.8**	49.1	4.2
346 × 260	10.17	**3.22**	183.79	**40.96**	44.5	**3.7**	49.5	4.1
640 × 480	30.88	**10.15**	687.10	**264.39**	147.2	**12.6**	47.9	4.1
1280 × 720	93.34	**31.01**	2235.60	**1039.49**	441.7	**37.8**	47.9	4.1

5.3.2 Computational Performance

Table 5.2 compares the computational cost of FireNet against E2VID. I used an NVIDIA Titan Xp GPU and an Intel 3.20 GHz i7-6900K CPU for all experiments. To evaluate computational cost, I measured the compute time of a forward-pass through the network at various image sensor resolutions on both GPU and CPU. I selected resolutions of common event cameras such as the DAVIS240 [Brandli et al., 2014a], DAVIS346 [Taverni et al., 2018], Samsung, Prophesee and CeleX sensors. I also report the number of floating point operations per forward-pass (FLOPs) at each resolution, which is related to power consumption. E2VID and FireNet are agnostic to the number of events per forward-pass, that is, a forward-pass will take the same amount of time if the input event tensor contains zero or one million events. FireNet performs three times faster than E2VID on GPU, and up to four times faster on CPU, requiring less than one tenth the number of FLOPs. GPU runtime (Table 5.2) does not scale linearly with FLOPs due to efficient parallel computation, thus a 10× reduction in FLOPs versus E2VID yields only 3× improvement in runtime. FLOPs per pixel varies slightly with resolution due to convolutional padding that depends on the aspect ratio, pixel count, and down/up-sampling operations (in E2VID).

5.3.3 Accuracy

I evaluated the accuracy of FireNet against DAVIS240C [Brandli et al., 2014a] frames in the event camera dataset [Mueggler et al., 2017b] (Tab. 5.3, Fig. 5.2), and compared against several competitive methods: high-pass filter (HF) [Scheerlinck et al., 2018], manifold regularisation (MR) [Reinbacher et al., 2016] and E2VID [Rebecq et al., 2019, 2020a]. Since DAVIS frames were used as ground truth, I did not include methods that use DAVIS frames as input, considering only pure event reconstruction. In the evaluation dataset I discarded sections with poor frame quality, leaving seven sequences with 1,670 groundtruth frames. Given a pair of successive image frames

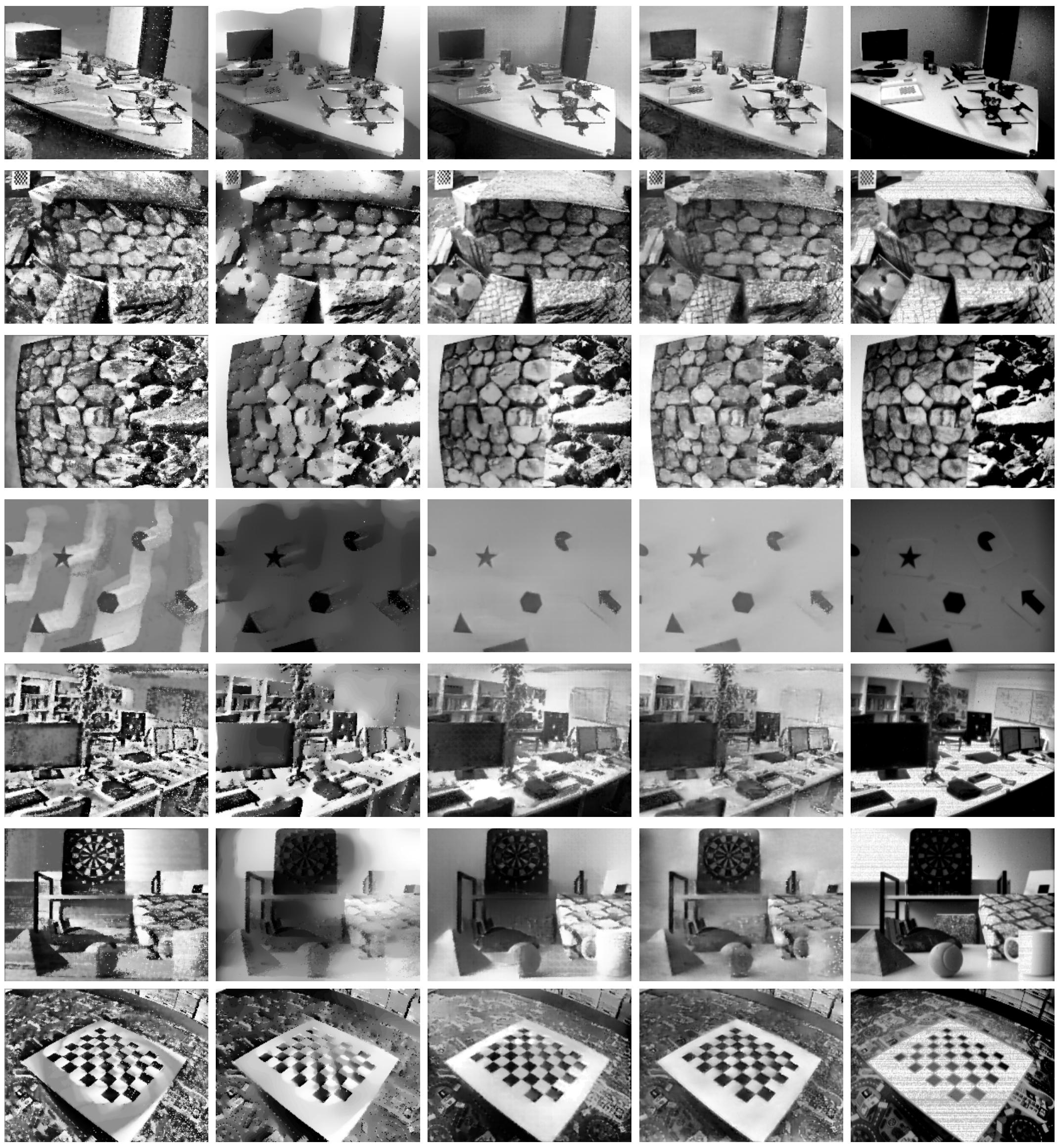


Figure 5.2: Qualitative comparison against state-of-the-art methods on the event camera dataset [Mueggler et al., 2017b]. FireNet performs comparably to E2VID [Rebecq et al., 2019, 2020a], and produces higher quality images than HF [Scheerlinck et al., 2018] and MR [Reinbacher et al., 2016].

Table 5.3: Comparison to state-of-the-art image reconstruction methods on the Event Camera Dataset [Mueggler et al., 2017b].

Dataset	MSE				SSIM				LPIPS			
	HF	MR	E2VID	Ours	HF	MR	E2VID	Ours	HF	MR	E2VID	Ours
dynamic_6dof	0.10	**0.05**	0.14	0.12	0.39	**0.52**	0.46	0.47	0.54	0.50	0.46	**0.44**
boxes_6dof	0.08	0.10	**0.04**	0.04	0.49	0.45	0.62	**0.64**	0.50	0.53	0.38	**0.37**
poster_6dof	0.07	0.05	0.06	**0.04**	0.49	0.54	0.62	**0.65**	0.45	0.52	0.35	**0.34**
shapes_6dof	0.09	0.19	0.04	**0.02**	0.50	0.51	**0.80**	0.79	0.61	0.64	0.47	**0.46**
office_zigzag	0.09	0.09	**0.03**	0.04	0.38	0.45	**0.54**	0.54	0.54	0.50	0.41	**0.40**
slider_depth	0.06	0.07	0.05	**0.05**	0.50	0.50	0.58	**0.59**	0.50	0.55	0.44	**0.41**
calibration	0.09	0.07	**0.02**	0.04	0.48	0.54	**0.70**	0.66	0.48	0.47	**0.36**	0.37
Mean	0.08	0.09	0.05	**0.05**	0.46	0.50	0.62	**0.62**	0.52	0.53	0.41	**0.40**

(∼20Hz for the event camera dataset [Mueggler et al., 2017b]), I took all events in between the frames and constructed an event tensor as specified in 5.2.1, creating a sequence of event tensors for each dataset sequence. I reconstructed a video for each sequence and compared against groundtruth frames. For all methods, I matched the timestamp of the latest event used in each reconstructed image to the nearest groundtruth frame with a tolerance of 1ms. For HF and MR, I used code provided by the authors and manually tuned the parameters to get best results possible. For HF I additionally applied a 5×5 bilateral filter with $\sigma = 25$ to smooth high-frequency noise, which improved results of HF in all metrics. To ensure the intensity values lay within a similar range, I applied local histogram normalisation to both the output and groundtruth frames. I compared reconstructed images against groundtruth frames using the metrics: mean squared error (MSE), structural similarity (SSIM) [Wang et al., 2004] and perceptual similarity (LPIPS) [Zhang et al., 2018].

Table 5.3 shows that FireNet performs favourably compared to hand-crafted methods HF and MR, achieving a 40% decrease in mean squared error, 20% increase in structural similarity and 20% improvement in perceptual similarity. FireNet quantitatively matched performance of E2VID for all metrics on the event camera dataset [Mueggler et al., 2017b], though with minor qualitative defects for some challenging scenarios (see Fig. 5.7).

5.3.4 Qualitative Evaluation

Figure 5.3 shows qualitative comparison to [Mostafavi I. et al., 2019] who used a generative adversarial network (GAN), trained on a mixture of real and synthetic data. While FireNet was trained exclusively on synthetic data, GAN was trained on data from the event camera dataset [Mueggler et al., 2017b] (rows 2, 3) (i.e., the network has seen these sequences at train time and may be overfitting), thus, I did not include quantitative comparison out of fairness. Images reconstructed with GAN appear less sharp, and contain artifacts in textureless regions of the scene (where

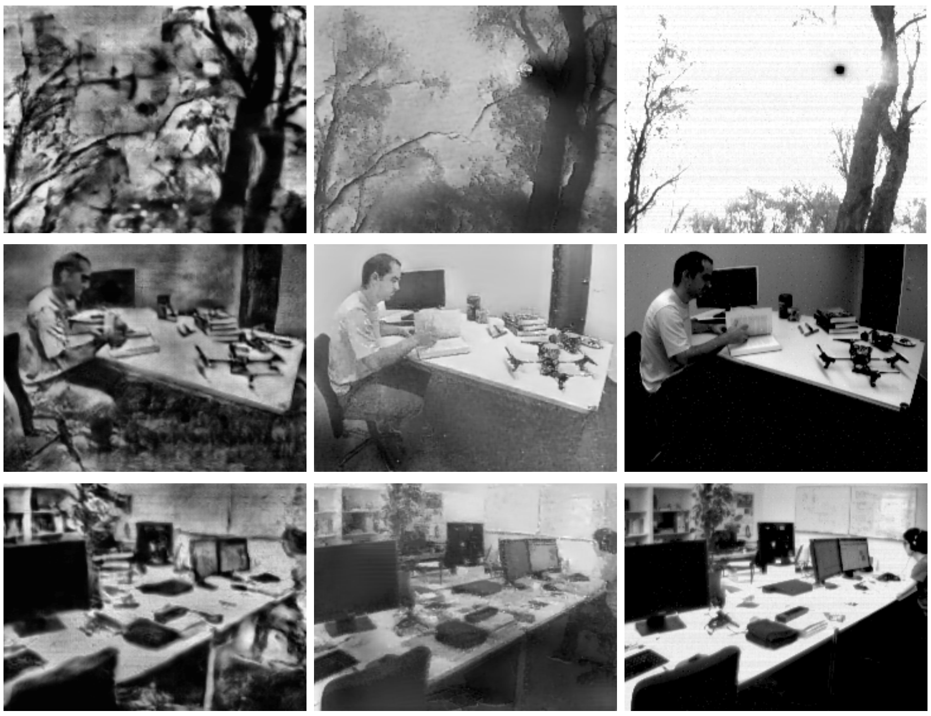


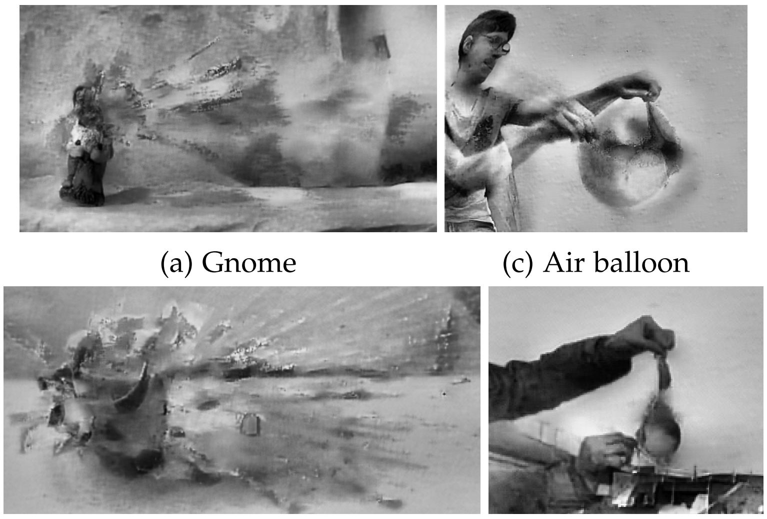
Figure 5.3: Left: GAN [Mostafavi I. et al., 2019] appears less sharp and exhibits artifacts in textureless regions (note: rows 2, 3 were used in training). Middle: FireNet looks cleaner but still suffers from very noisy events e.g., Sun (top). Right: the DAVIS frame has lower dynamic range than events, evident in the reconstructions of both methods.

there are no events). In challenging sequences, such as Sun (from [Scheerlinck et al., 2018]), there are many noise events that translate into artifacts for both GAN and FireNet, however, FireNet appears less impacted. Note that both methods appear to contain more information than the oversaturated DAVIS frame.

Figure 5.4 shows results on the *High Speed and HDR Dataset*[1] [Rebecq et al., 2020a], demonstrating that FireNet can generalise to a different sensor (Samsung DVS Gen3 [Son et al., 2017]). Each sequence was reconstructed using 50k events per input tensor (higher due to the higher sensor resolution: 640×480 vs. 240×180). Local histogram equalisation was applied to improve the visual contrast of the images.

Figure 5.5 shows results on the Color Event Camera Dataset (CED)[2] [Scheerlinck et al., 2019b]. Each color channel was reconstructed independently at quarter resolution then upsampled following [Rebecq et al., 2020b]. Qualitatively, colors have been distorted and images are low contrast, indicating that FireNet has not generalised well to color - possibly due to limited network size/capacity or lack of color training examples.

[1]Available at: http://rpg.ifi.uzh.ch/E2VID.html
[2]http://rpg.ifi.uzh.ch/CED.html



(a) Gnome (c) Air balloon

(b) Mug (d) Water balloon

Figure 5.4: High speed phenomenon from the *High Speed and HDR Dataset* [Rebecq et al., 2020a]. (a) and (b) are moments after a bullet impact from a gun. (c) and (d) are moments after an air and water balloon are popped. The water (d) initially retains the original shape of the balloon as it falls.

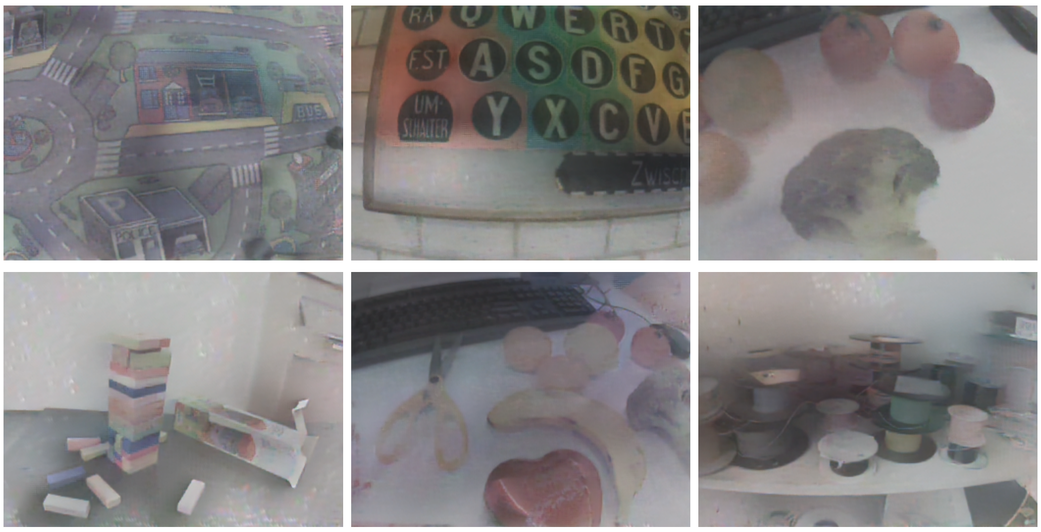


Figure 5.5: FireNet reconstructed images using color events from the Color Event Camera Dataset (CED) [Scheerlinck et al., 2019b].

5.3.5 Recurrent Connection Ablation

Recurrent connections in a network give it the ability to build a hidden state $h(t)$ that can be maintained and improved over time. Given temporal sequences of training data, the network learns some form of temporal integration, re-using previous computed results. Figure 5.6 shows that E2VID can reconstruct images from batches of 10k events per input tensor without recurrent connections. I believe this is because its large size and receptive field allows it to spatially propagate information from events (edges). However, smaller models such as FireNet cannot reliably reconstruct images without recurrent connections because of a limited receptive field (maximum 15×15 for FireNet). Because FireNet is fully convolutional, pixels in the prediction layer can only see events within their receptive field. Thus, I can conclude that recurrent connections are the primary driver enabling a smaller network.

5.3.6 Limitations

In challenging scenarios such as very fast motions, and initialisation, FireNet exhibits defects such as smearing, or incomplete reconstruction in places with no events. Figure 5.7 shows selected challenging scenes where FireNet artifacts are apparent, while E2VID [Rebecq et al., 2020a] typically does a better job. To highlight smearing artifacts I used a fixed time-window of 50ms per input event tensor for both methods. Using a smaller time-window or fixed number of events per input tensor may decrease smearing for fast motions.

5.4 Conclusion

This chapter has presented FireNet, a fast, lightweight CNN that reconstructs images directly from events. FireNet performs almost as well as state-of-the-art (E2VID [Rebecq et al., 2019, 2020a]) at a fraction of the computational cost, yielding a $3\times$ speedup, $10\times$ reduction in FLOPs with $280\times$ fewer parameters. I showed that recurrent connections are a key component enabling smaller networks because it allows them to build and improve a hidden state over time, re-using previous computed results. I believe FireNet shows exciting potential for fast, lightweight recurrent networks for event processing, and that the reconstructed images reveal an exciting depth of information that can be unlocked from events with a surprisingly small network.

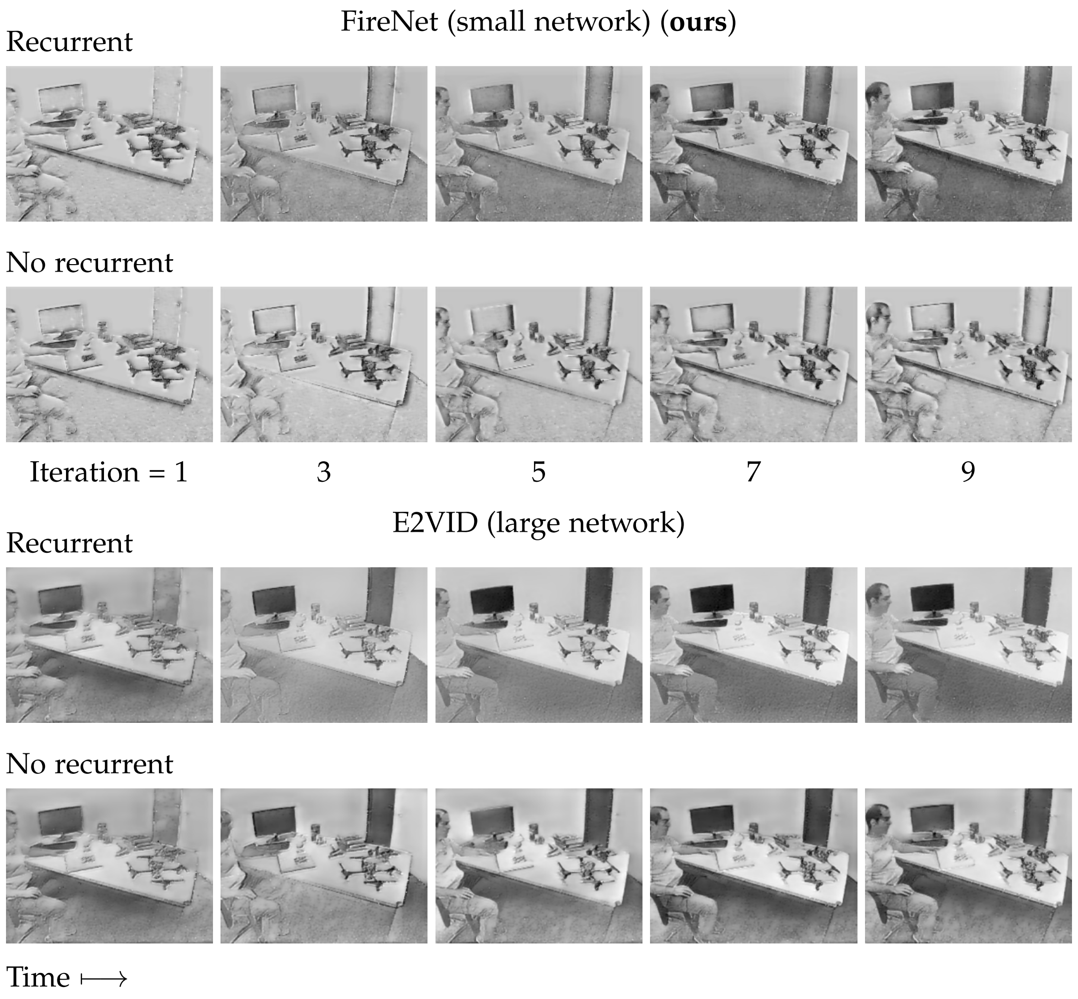


Figure 5.6: Recurrent connection ablation study. Image is initialised at zero. Top: small network (ours) relies on recurrent connection to build hidden state over time. When recurrent connection is disabled (second row), the network fails, indicating that recurrent connections are a key component. Bottom: While recurrent connections help stabilise video (third row), large networks (E2VID [Rebecq et al., 2019, 2020a]) are still able to reconstruct images without recurrent connection (fourth row).

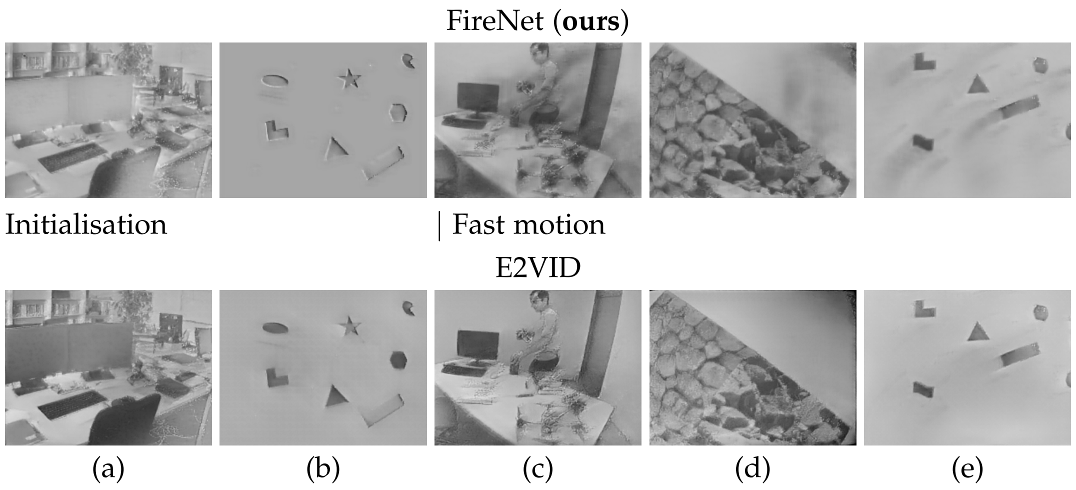


Figure 5.7: (a) and (b) show the network output at initialisation. FireNet (ours) does not "fill in" the image as quickly as E2VID [Rebecq et al., 2020a, 2019]. (c)-(e) are captured when the camera is undergoing fast motion. FireNet exhibits more smearing artifacts than E2VID.

Chapter 6

Reducing the Sim-to-Real Gap for Event Cameras

Website: https://cedricscheerlinck.com/20ecnn

Based on [Stoffregen et al., 2020][1]

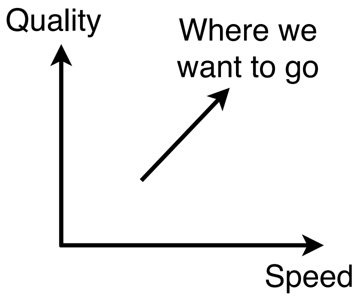


Figure 6.1: Neural networks commonly trade-off quality for speed. Ideally we want to increase both.

In the previous chapter I explored speeding up inference (and training) time by cutting down the size of the network, possibly at the cost of image reconstruction quality. In this chapter I aim to explore along the quality axis (Fig. 6.1) and identify failure modes of existing state-of-the-art and how to fix them. Recent work has demonstrated impressive results using Convolutional Neural Networks (CNNs) for video reconstruction and optic flow with events. Using a simulator to generate training data is appealing because groundtruth labels are easy to obtain and large volumes of data can be obtained. However, generalisation to real data depends on the similarity between simulated (training) and real (test) data, thus reducing the sim-to-real gap is essential. 'Sim-to-real' in robotic vision typically reflects training machine learning algorithms using data acquired from simulated environments. In this chapter, sim-to-real refers to training a network using synthetic events generated from real images. I present strategies for reducing the sim-to-real gap for event cameras that result in 20-40 % boost in performance of existing state-of-the-art video reconstruction networks retrained with the proposed method, and up to 15 % for optic flow networks. A challenge in evaluating event based video reconstruction is lack of quality groundtruth images in existing datasets. To address this, I present a **High Quality Frames (HQF)** dataset[2], containing events and groundtruth frames from a DAVIS240C that are well-exposed and minimally motion-blurred. I evaluate the proposed method on HQF in addition to several existing major event camera

[1]Equal first authorship.

[2]published with co-authors in [Stoffregen et al., 2020].

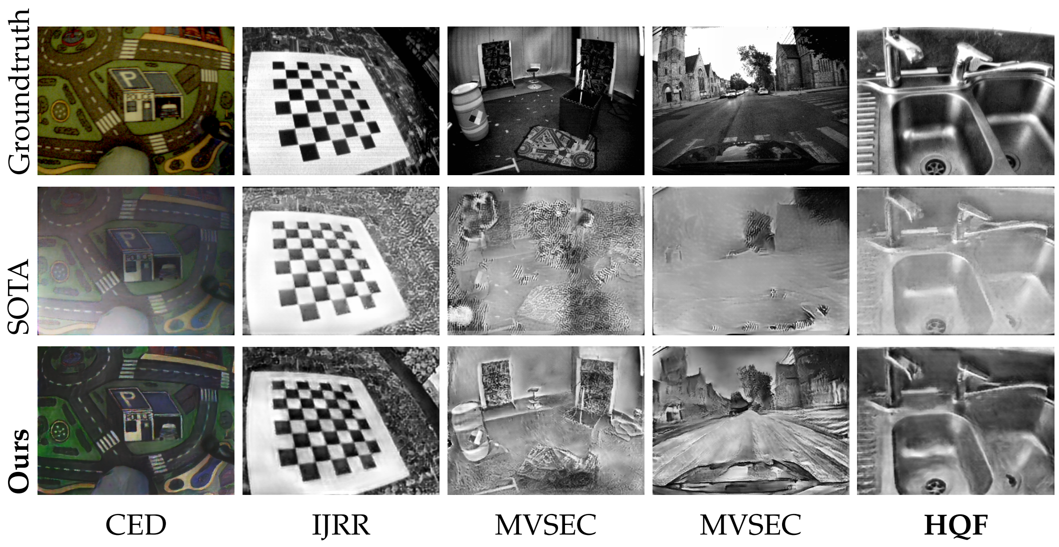


Figure 6.2: Top: groundtruth reference image. Middle/bottom: state-of-the-art (SOTA) E2VID [Rebecq et al., 2020a] vs. our reconstructed images from events only. Challenging scenes from event camera datasets: CED [Scheerlinck et al., 2019b], IJRR [Mueggler et al., 2017b], MVSEC [Zhu et al., 2018a] and **HQF** [Stoffregen et al., 2020].

datasets and show that training on data that better matches the distribution of real data is key to improving quality of results 6.2.
This chapter was jointly authored with Timo Stoffregen.

6.1 Introduction

Event-based cameras such as the Dynamic Vision Sensor (DVS) [Lichtsteiner et al., 2008] are novel, bio-inspired visual sensors. Presenting a paradigm-shift in visual data acquisition, pixels in an event camera operate by asynchronously and independently reporting intensity changes in the form of events, represented as a tuple of x, y location, timestamp t and polarity of the intensity change σ. By moving away from fixed frame-rate sampling of conventional cameras, event cameras deliver several key advantages in terms of low power usage (in the region of 5 mW), high dynamic range (140 dB), low latency and timestamps with resolution on the order of µs.

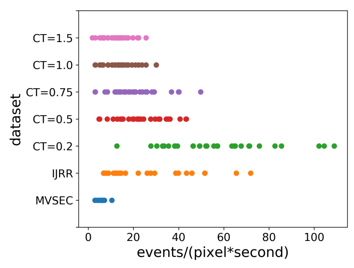
With the recent preponderance of deep learning techniques in computer vision, the question of how to apply this technology to event data has been the subject of several recent works. Zhu et al. [2018b] proposed an unsupervised network able to learn optic flow from real event data, while Rebecq et al. [2019] showed that supervised networks trained on synthetic events transferred well to real event data. Simulation shows promise since data acquisition and groundtruth are easily obtainable, in contrast to using real data. However, mismatch between synthetic and real data degrades performance, so a key challenge is simulating realistic data that covers a large range of scenarios.

I generated training data that better matches real event camera data by analysing the statistics of existing datasets to inform my choice of simulation parameters. A major finding is that the contrast threshold (CT) - the minimum change in brightness required to trigger an event - is a key simulation parameter that impacts performance of supervised CNNs. Further, I observed that the apparent contrast threshold of real event cameras varies greatly, even within one dataset. Previous works such as event based video reconstruction [Rebecq et al., 2020b] choose contrast thresholds that work well for some datasets, but fail on others. Unsupervised networks trained on real data such as event based optic flow [Zhu et al., 2018b] may be retrained to match any real event camera - at the cost of new data collection and training. I show that using CT values for synthetic training data that are correctly matched to CTs of real datasets is a key driver in improving performance of retrained event based video reconstruction and optic flow networks across multiple datasets. In addition, I propose a simple noise model that can be used to dynamically augment event data at train time, yielding up to 10 % improvement.

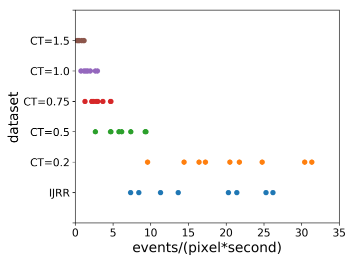
A challenge in evaluating image and video reconstruction from events is lack of quality groundtruth images registered and time-synchronised to events, because most existing datasets focus on scenarios where event cameras excel (high speed, HDR) and conventional cameras fail. To address this limitation, I used the High Quality Frames (HQF) dataset [Stoffregen et al., 2020] that provides several sequences in well lit environments with minimal motion blur. HQF sequences were recorded with a DAVIS240C event camera that provides perfectly aligned frames from an integrated Active Pixel Sensor (APS). HQF also contains a diverse range of motions and scene types, including slow motion and pauses that are challenging for event based video reconstruction. I quantitatively evaluate the proposed method on two major event camera datasets: IJRR [Mueggler et al., 2017b] and MVSEC [Zhu et al., 2018a], in addition to HQF, demonstrating gains of 20-40 % for video reconstruction and up to 15 % for optic flow when I retrain existing SOTA networks.

This chapter presents a method to generate synthetic training data that improves generalisability to real event data, guided by statistical analysis of existing datasets. I additionally propose a simple method for dynamic train-time noise augmentation that yields up to 10 % improvement for video reconstruction. Using the proposed method, I retrain several network architectures from previously published works on video reconstruction [Rebecq et al., 2020b; Scheerlinck et al., 2020] and optic flow [Zhu et al., 2018b, 2019] from events. I am able to show significant improvements that persist over architectures and tasks. Thus, I believe my findings will provide invaluable insight for others who wish to train models on synthetic events for a variety of tasks.

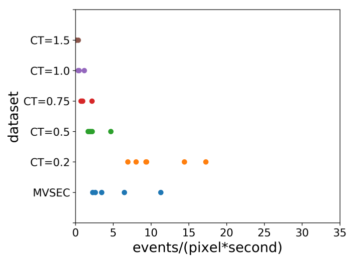
The remainder of the chapter is as follows. Section 6.2 outlines the proposed method for generating training data, training and evaluation, and describes the HQF dataset. Section 6.3 presents experimental results on video reconstruction and optic flow. Section 6.4 discusses the major findings and concludes the chapter.



(a) IJRR/MVSEC vs. ESIM



(b) IJRR vs ESIM



(c) MVSEC vs ESIM

Figure 6.3: Each dot represents a sequence from the given dataset (y-axis). (a) $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of IJRR and MVSEC vs. ESIM training datasets (CT 0.2-1.5) described in Section 6.2.2. (b) $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of IJRR vs. ESIM events simulated from IJRR *APS frames*. (c) $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of MVSEC vs. ESIM events simulated from MVSEC *APS frames*.

6.2 Method

6.2.1 Event Camera Contrast Threshold

In an ideal event camera, a pixel at (x, y) triggers an event e_i at time t_i when the brightness since the last event e_{i-1} at that pixel changes by a threshold c, given $t - t_{i-1} > r$, the refractory period of that pixel. c is referred to as the contrast threshold (CT) and can be typically adjusted in modern event cameras. In reality, the values for c are not constant in time nor homogeneous over the image plane nor is the positive threshold c_p necessarily equal to the negative threshold c_n. In simulation (e.g., using ESIM [Rebecq et al., 2018]), CTs are typically sampled from $\mathcal{N}(\mu{=}0.18, \sigma{=}0.03)$ to model this variation [Rebecq et al., 2019, 2020b; Gehrig et al., 2019a]. The CT is an important simulator parameter since it determines the number and distribution of events generated from a given scene.

While the real CTs of previously published datasets are unknown, one method to estimate CTs is via the proxy measurement of average events per pixel per second ($\frac{\text{events}}{\text{pix}\cdot\text{s}}$). Intuitively, higher CTs tend to reduce the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ for a given scene. While other methods of CT estimation exist [Wang et al., 2019], I found that tuning the simulator CTs to match $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of real data worked well. Since this measure is affected by scene dynamics (i.e., faster motions increase $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ independently of CT), I generated a diverse variety of realistic scene dynamics. The result of this experiment (Fig. 6.3a) indicates that a contrast threshold setting of between 0.2 and 0.5 would be more appropriate for IJRR sequences. The larger diversity of motions is also apparent in the large spread of the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ over the sequences, compared to MVSEC whose sequences are tightly clustered.

As an alternative experiment to determine CTs of existing datasets, I measured the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of events simulated using the actual APS (groundtruth) frames of IJRR and MVSEC sequences. Given high quality images with minimal motion blur and little displacement, events can be simulated through image interpolation and subtraction.

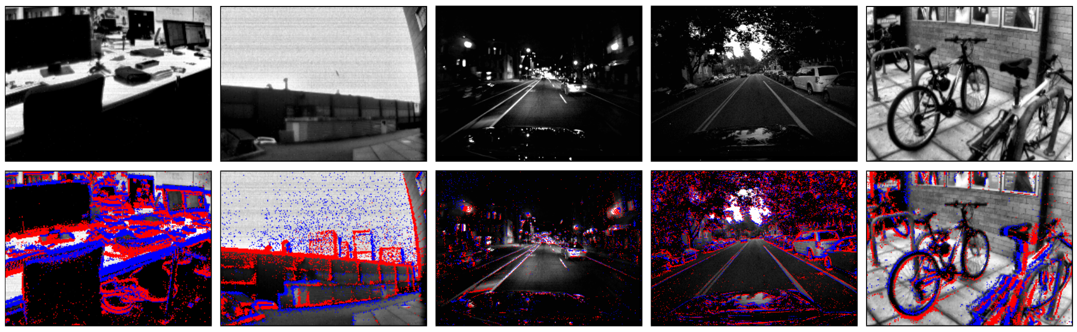


Figure 6.4: Note that in many sequences from the commonly used IJRR and MVSEC datasets, the accompanying APS frames are of low quality. The top row shows the APS frames, the bottom row overlays the events. As can be seen, many features are not visible in the APS frames, making quantitative evaluation difficult. This motivates the High Quality Frames dataset (HQF).

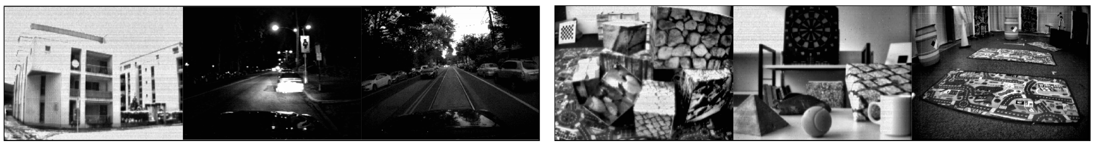

(a) Poorly exposed from IJRR and MVSEC
(b) Well exposed from IJRR and MVSEC

Figure 6.5: Examples of frames from IJRR and MVSEC after local histogram equalisation, with poorly exposed sequences in 6.5a, and better exposed images in 6.5b.

Given an ideal image sequence, the simulator settings should be tunable to get the exact same $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ from simulation as from the real sensor. Unfortunately APS frames are not usually of a very high quality (Fig. 6.4), so I was limited to using this approach on carefully curated snippets (Fig. 6.5). The results of this experiment in Fig. 6.3b and 6.3c indicate similar results of lower contrast thresholds for IJRR and higher for MVSEC, although accuracy is limited by the poor quality APS frames.

6.2.2 Training Data

I use an event camera simulator, ESIM [Rebecq et al., 2018] to generate training sequences for the network. There are several modes of simulation available, of which I use "Multi-Object-2D", which facilitates moving images in simple 2D motions, restricted to translations, rotations and dilations over a planar background. This generates sequences reminiscent of Flying Chairs [Dosovitskiy et al., 2015], where objects move across the screen at varying velocities. In the data generation scheme, I randomly select images from COCO [Lin et al., 2014a], which receive random trajectories over the image plane. The dataset contains 280 sequences, 10 s in length. Sequences alternate between four archetypal scenes; slow motion with 0-5 foreground objects, medium speed motion with 5-10 foreground objects, fast speed with 5-20 foreground objects and finally, full variety of motions with 10-30 foreground objects. This variety allows models trained on the dataset to generalise well to arbitrary real world camera motions, since the network has seen a wide range of scene dynamics. Sequences are generated with contrast thresholds (CTs) between 0.1 and 1.5 in ascending order. Since real event cameras do not usually have perfectly balanced positive and negative thresholds, I set the positive threshold $c_p = c_n \cdot x, x \in \mathcal{N}(\mu{=}1.0, \sigma{=}0.1)$.

The events thus generated are discretised into a voxel grid representation. In order to ensure synchronicity with the ground truth frames of the training set and later with the ground truth frames of the validation set, I always take all events between two frames to generate a voxel grid. Given N events $e_i = \{x_i, y_i, t_i, p_i\}_{i=0,...,N}$ spanning $\Delta t = t_N - t_0$ seconds, a voxel grid $V(x, y, t_n)$ with B bins can be formed through temporal bilinear interpolation via

$$V(x, y, t_n) = \sum_{i=0}^{N} \sigma_i \max(0, 1 - |t_n - t_i^*|), \tag{6.1}$$

$$t_i^* = \frac{(t_i - t_{\text{min}})}{(t_{\text{max}} - t_{\text{min}})}(B - 1)$$

where t_i^* is the timestamp normalised to the range $[0, B - 1]$ and the bins are evenly spaced over the range $[t_0, t_N]$. This method of forming voxels has some limitations; it is easy to see that the density of the voxels can vary greatly, depending on the camera motion and frame rate of the camera. This makes it important to show the network a dataset with varying numbers of events generated, so that it sees a large variety of voxel densities. During inference, other strategies of voxel generation can be employed e.g., using a fixed number of events per voxel grid. I use $B = 5$ throughout the experiments in the chapter. In earlier experiments I found values of $B = 2, 5, 15$ to produce no significant differences.

6.2.3 Sequence Length

To train recurrent networks, I sequentially passed L inputs to the network and computed the loss for each output. Finally, the losses are summed and a backpropagation update is performed based on the gradient of the final loss with respect to the network weights. Since the recurrent units in the network are initialised to zero, lower values of L restrict the temporal support that the recurrent units see at train time. To investigate the impact of sequence length L, I retrained the networks using $L = 40$ (as in E2VID [Rebecq et al., 2020b]) and $L = 120$. In the case of non-recurrent networks such as EV-FlowNet [Zhu et al., 2018b, 2019], I ignored the sequence length parameter.

6.2.4 Loss

For the primary video reconstruction loss function I used "learned perceptual image patch similarity" (LPIPS) [Zhang et al., 2018] (table 6.1). LPIPS is a fully differentiable

Table 6.1: Losses used to train image reconstruction and optic flow networks.

Loss	**Equation**	**Description**
LPIPS	$\sum_l \frac{1}{H_l W_l} \sum_{h,w} \lvert\lvert w_l \bigodot (\hat{y}^l_{hw} - y^l_{hw}) \rvert\rvert^2_2$	l: layer, H: height, W: width, w_l: learned weights, $\hat{y}^l$: channel-wise unit normalised activations from a pretrained network e.g., Alex-Net, VGG etc., $\hat{y}$ vs. y: prediction vs. groundtruth [Zhang et al., 2018].
Temporal consistency loss	$\exp(-\alpha \lvert\lvert \mathcal{I}_k - \mathcal{W}^k_{k-1}(\mathcal{I}_{k-1}) \rvert\rvert^2_2) \times \lvert\lvert \hat{\mathcal{I}}_k - \mathcal{W}^k_{k-1}(\hat{\mathcal{I}}_{k-1}) \rvert\rvert_1 \div (\lvert\mathcal{I}\rvert + \lvert\hat{\mathcal{I}}\rvert + \epsilon)$	α: Scalar weight (hand-tuned), $\mathcal{I}_k$: image, $\mathcal{W}^k_{k-1}$ warping function from timestep $k-1$ to k, $\hat{\mathcal{I}}$ vs. $\mathcal{I}$: prediction vs. groundtruth, ϵ: Avoid zero-division [Lai et al., 2018].
Flow L1 loss	$\lvert\lvert \hat{d} - d \rvert\rvert_1$	$\hat{d}$ vs. d: prediction vs. groundtruth displacement.

similarity metric between two images that compares hidden layer activations of a pretrained network (e.g., Alex-Net or VGG), and is shown to better match human judgement of image similarity than photometric error and SSIM [Wang et al., 2004]. Since the event tensors are synchronised to the groundtruth image frames by design (the final event in the tensor matches the frame timestamp), I compute the LPIPS distance between the reconstruction and the corresponding groundtruth frame. As recommended by the authors Zhang et al. [2018], I use the Alex-Net variant of LPIPS. I additionally impose a temporal consistency loss [Lai et al., 2018] that measures photometric error between consecutive images after registration based on optic flow, subject to an occlusion mask. For optic flow, I use the L1 distance between the prediction and groundtruth as the training loss.

6.2.5 Data Augmentation

During training, Rebecq et al. [2020b] occasionally set the input events to zero and performed a forward-pass step within a sequence, using the previous ground truth image frame to compute the loss. The probability of initiating a pause when the sequence is running $P(p|r) = 0.05$, while the probability of maintaining the paused state when the sequence is already paused $P(p|p) = 0.9$ to encourage occasional long pauses. This encourages the recurrent units of the network to learn to 'preserve' the output image in absence of new events.

Event cameras provide a noisy measurement of brightness change, subject to background noise, refractory period after an event and hot pixels that fire many spurious events. To simulate real event data, I applied a refractory period of 1ms. At train time, for each sequence of L input event tensors I optionally add zero-mean

Table 6.2: Breakdown of sequences included in HQF. To provide some inter-device variability, the dataset is taken with two separate DAVIS 240C cameras, 1 and 2.

Sequence	Length [s]	Cam.	Frames [k]	Events [M]	Description
bike_bay_hdr	99.0	1	2.4	19.8	Camera moves from dim to bright
boxes	24.2	1	0.5	10.1	Indoor light, translations
desk	65.8	2	1.5	13.5	Natural light, various motions
desk_fast	32.0	2	0.7	12.6	Natural light, fast motions
desk_hand_only	20.6	2	0.5	0.8	Indoor light, static camera
desk_slow	63.3	2	1.4	1.9	Natural light, slow motions
engineering_posters	60.7	1	1.3	15.4	Indoor light, text and images
high_texture_plants	43.2	1	1.1	14.6	Outdoors, high textures
poster_pillar_1	41.8	1	1.0	7.1	Outdoors, text and images
poster_pillar_2	25.4	1	0.6	2.5	Outdoors, text and images, long pause
reflective_materials	28.9	1	0.6	7.8	Natural light, reflective objects
slow_and_fast_desk	75.6	1	1.7	15.0	Natural light, diverse motion
slow_hand	38.9	1	0.9	7.6	Indoor, slow motion, static camera
still_life	68.1	1	1.2	42.7	Indoors, Indoor light, 6DOF motions

Gaussian noise ($\mathcal{N}(\mu{=}0,\ \sigma{=}0.1)$) to the event tensor to simulate uncorrelated background noise, and randomly elect a few 'hot' pixels. The number of hot pixels is drawn from a uniform distribution from 0 to 0.0001, multiplied by the total number of pixels. Hot pixels have a random value ($\mathcal{N}(\mu{=}0,\ \sigma{=}0.1)$) added to every temporal bin in each event tensor within a sequence. To determine whether augmenting the training data with noise benefits performance on real data, I retrained multiple models with and without noise (Tab. 6.17).

6.2.6 Architecture

To isolate the impact of the proposed method from choice of network architecture, I retrained state-of-the-art (SOTA) video reconstruction network E2VID [Rebecq et al., 2020b] and the SOTA optic flow network described in [Zhu et al., 2018b, 2019]. Thus, differences in performance for each task are not due to architecture. Additionally, I aim to show that the proposed method generalises to multiple architectures. While I believe architecture search may further improve results, it is outside the scope of this chapter.

6.2.7 High Quality Frames Dataset

To evaluate event camera image reconstruction methods, I compared reconstructed images to temporally synchronised, registered groundtruth reference images. Event cameras such as the DAVIS [Brandli et al., 2014a] can capture image frames (in addition to events) that are timestamped and registered to the events, that may serve as groundtruth. Previous event camera datasets such as IJRR [Mueggler et al., 2017b] and MVSEC [Zhu et al., 2018a] (table 6.3) contain limited high quality DAVIS frames, while many frames are motion-blurred and or under/overexposed (Fig. 6.4). As a

Table 6.3: Overview of event camera datasets.

Name	**Sensor**	**Year**	**Description**
Event Camera Dataset and Simulator (IJRR)	DAVIS240C	2017	Indoor office scenes, handheld event camera, groundtruth image frames (DAVIS) [Mueggler et al., 2017b].
Multi-Vehicle Stereo Event Camera Dataset (MVSEC)	DAVIS346B	2018	Driving scenes (day & night), Indoor flying (hexacopter), groundtruth frames (VI-Sensor, DAVIS) [Zhu et al., 2018a].
Color Event Camera Dataset (CED)	DAVIS346-Color	2019	Indoor office scenes, driving, color events and groundtruth frames (RGBG Bayer DAVIS) [Scheerlinck et al., 2019b].
High Speed and HDR Dataset (HSD)	Samsung DVS Gen3	2019	Bullet impacts, balloon popping and driving scenes. No groundtruth frames [Rebecq et al., 2020b].
High Quality Frames Dataset (HQF)	DAVIS240C	2020	Large interscene motion, low event rate. Groundtruth frames (DAVIS) [Stoffregen et al., 2020].

result, Rebecq et al. [2020b] manually rejected poor quality frames, evaluating on a smaller subset of IJRR.

I captured a High Quality Frames dataset (HQF) aimed at providing groundtruth DAVIS frames that are minimally motion-blurred and well exposed, published with co-authors at [Stoffregen et al., 2020] (Table 6.2). HQF covers a wider range of motions and scene types than the evaluation dataset used for E2VID, including: static/dynamic camera motion vs. dynamic camera only, very slow to fast vs. medium to fast and indoor/outdoor vs. indoor only. To record HQF, I used two different DAVIS240C sensors to capture data with different noise/CT characteristics. I used default bias settings loaded by the RPG DVS ROS driver[3], and set exposure to either auto or fixed to maximise frame quality. HQF provides temporally synchronised, registered events and DAVIS frames.

[3]https://github.com/uzh-rpg/rpg_dvs_ros

Table 6.4: Start and end times for sequences in IJRR and MVSEC that I present validation statistics on. While both IJRR and MVSEC contain more sequences than the ones listed, those excluded had low quality accompanying frames (see Figure 6.4).

IJRR			MVSEC		
Sequence	Start [s]	End [s]	Sequence	Start [s]	End [s]
boxes_6dof	5.0	20.0	indoor_flying1	10.0	70.0
calibration	5.0	20.0	indoor_flying2	10.0	70.0
dynamic_6dof	5.0	20.0	indoor_flying3	10.0	70.0
office_zigzag	5.0	12.0	indoor_flying4	10.0	19.8
poster_6dof	5.0	20.0	outdoor_day1	0.0	60.0
shapes_6dof	5.0	20.0	outdoor_day2	100.0	160.0
slider_depth	1.0	2.5			

6.3 Experiments

6.3.1 Evaluation

I evaluated the proposed method by retraining two state-of-the-art event camera neural networks: E2VID [Rebecq et al., 2019, 2020b], and EV-FlowNet [Zhu et al., 2018b, 2019]. I show that the proposed method outperforms previous state-of-the-art in image reconstruction and optic flow on several publicly available event camera datasets including IJRR [Mueggler et al., 2017b] and MVSEC [Zhu et al., 2018a], and the High Quality Frames dataset [Stoffregen et al., 2020] (table 6.5).

For video reconstruction on the datasets HQF, IJRR and MVSEC (Tab. 6.5) I obtain a 40 %, 20 % and 28 % improvement over E2VID [Rebecq et al., 2020b] respectively, using LPIPS. For optic flow I obtained a 12.5 %, 10 % and 16 % improvement over EV-FlowNet [Zhu et al., 2018b] on flow warp loss (FWL, eq. 6.3). Notably, EV-FlowNet was trained on MVSEC data (`outdoor_day2` sequence), while ours was trained entirely on synthetic data, demonstrating the ability of the proposed method to generalise to real event data.

6.3.1.1 Image

As in [Rebecq et al., 2020b] I compared reconstructed images to groundtruth (DAVIS frames) on three metrics; mean squared error (MSE), structural similarity [Wang et al., 2004] (SSIM) and perceptual loss [Zhang and Rusinkiewicz, 2018] (LPIPS) which uses distance in the latent space of a pretrained deep network to quantify image similarity (Table 6.5). While retraining the original E2VID architecture [Rebecq et al., 2020b] (E2VID vs. Ours; Table 6.5) demonstrated significant improvement, I found mixed results when retraining FireNet [Scheerlinck et al., 2020] with the proposed method. I believe this is because E2VID has a larger capacity (due to larger number of parameters), enabling it to better capture and learn from a wider distri-

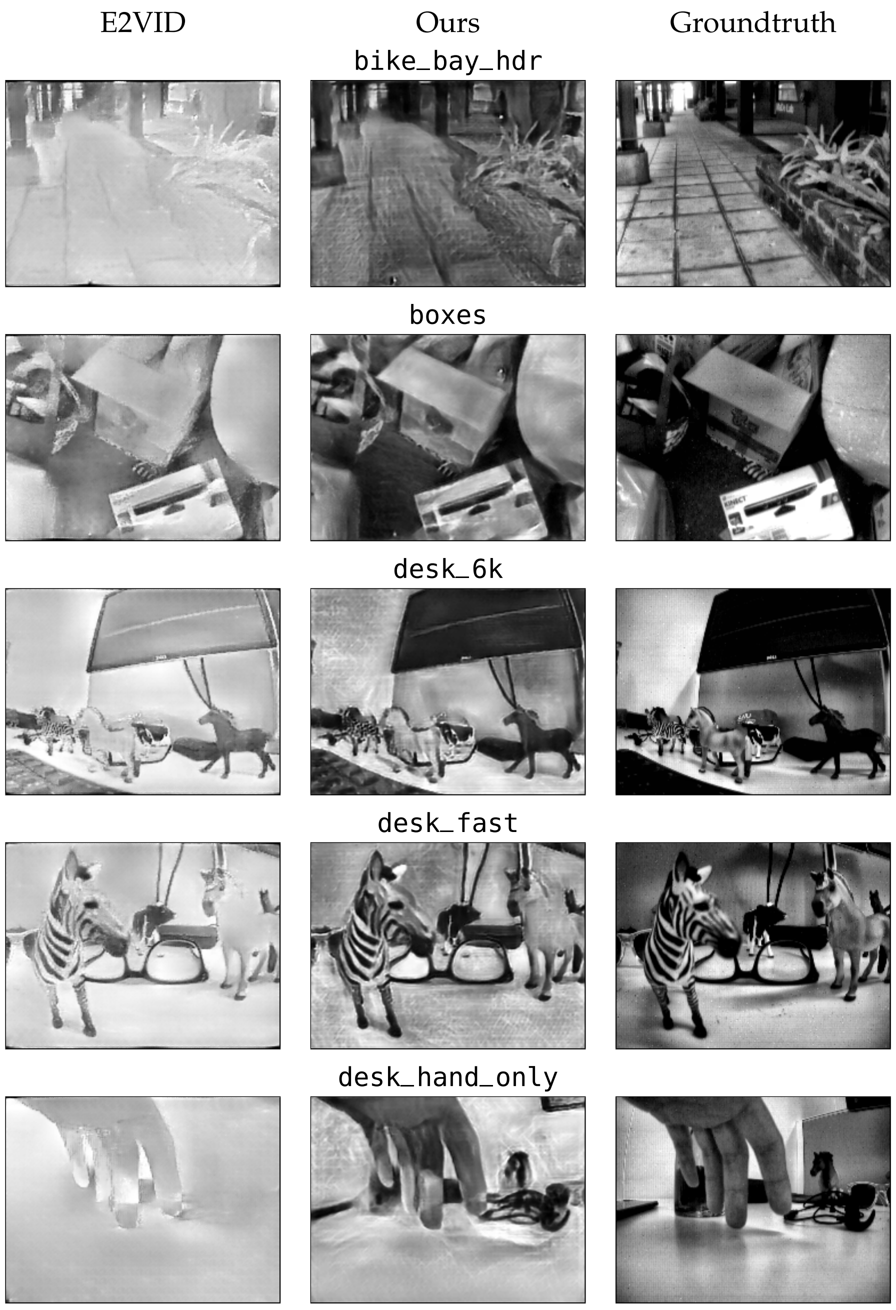
Table 6.5: Comparison of state-of-the-art methods of video reconstruction and optic flow to networks trained using our dataset on HQF, IJRR and MVSEC. Best in bold.

Sequence	MSE		SSIM		LPIPS		FWL	
	E2VID	Ours	E2VID	Ours	E2VID	Ours	EVFlow	Ours
					HQF			
bike_bay_hdr	0.16	**0.03**	0.41	**0.52**	0.51	**0.30**	1.22	**1.23**
boxes	0.11	**0.03**	0.50	**0.59**	0.38	**0.26**	1.75	**1.80**
desk_6k	0.15	**0.03**	0.51	**0.60**	0.39	**0.22**	1.23	**1.35**
desk_fast	0.12	**0.04**	0.54	**0.61**	0.40	**0.25**	1.43	**1.50**
desk_hand_only	0.12	**0.05**	0.53	**0.57**	0.63	**0.39**	**0.95**	0.85
desk_slow	0.16	**0.04**	0.53	**0.62**	0.47	**0.25**	1.01	**1.08**
engineering_posters	0.13	**0.03**	0.42	**0.57**	0.47	**0.26**	1.50	**1.65**
high_texture_plants	0.16	**0.03**	0.37	**0.65**	0.38	**0.14**	0.13	**1.68**
poster_pillar_1	0.14	**0.03**	0.38	**0.50**	0.54	**0.27**	1.20	**1.24**
poster_pillar_2	0.15	**0.04**	0.40	**0.47**	0.56	**0.26**	**1.16**	0.96
reflective_materials	0.13	**0.03**	0.44	**0.55**	0.44	**0.28**	1.45	**1.57**
slow_and_fast_desk	0.16	**0.03**	0.48	**0.62**	0.45	**0.25**	0.93	**0.99**
slow_hand	0.18	**0.04**	0.41	**0.57**	0.57	**0.30**	**1.64**	1.56
still_life	0.09	**0.03**	0.51	**0.63**	0.35	**0.22**	1.93	**1.98**
Mean	0.14	**0.03**	0.46	**0.58**	0.46	**0.26**	1.20	**1.35**
					IJRR			
boxes_6dof_cut	**0.04**	0.04	0.63	**0.64**	0.29	**0.25**	1.42	**1.46**
calibration_cut	0.07	**0.03**	0.61	**0.62**	0.22	**0.18**	1.20	**1.31**
dynamic_6dof_cut	0.17	**0.05**	0.45	**0.53**	0.38	**0.27**	1.37	**1.39**
office_zigzag_cut	0.07	**0.04**	0.49	**0.51**	0.31	**0.26**	**1.13**	1.11
poster_6dof_cut	0.07	**0.03**	0.60	**0.66**	0.26	**0.19**	1.50	**1.56**
shapes_6dof_cut	0.03	**0.02**	**0.80**	0.77	0.26	**0.22**	1.15	**1.57**
slider_depth_cut	0.08	**0.03**	0.54	**0.62**	0.35	**0.24**	1.73	**2.17**
Mean	0.07	**0.03**	0.61	**0.64**	0.28	**0.22**	1.32	**1.45**
					MVSEC			
indoor_flying1_data_cut	0.25	**0.08**	0.19	**0.36**	0.72	**0.45**	1.02	**1.14**
indoor_flying2_data_cut	0.23	**0.09**	0.18	**0.36**	0.71	**0.45**	1.13	**1.36**
indoor_flying3_data_cut	0.25	**0.09**	0.18	**0.37**	0.73	**0.44**	1.06	**1.23**
indoor_flying4_data_cut	0.21	**0.08**	0.23	**0.36**	0.72	**0.45**	1.24	**1.50**
outdoor_day1_data_cut	0.32	**0.13**	0.31	**0.34**	0.66	**0.52**	1.15	**1.27**
outdoor_day2_data_cut*	0.30	**0.10**	0.29	**0.34**	0.57	**0.43**	**1.21**	1.20
Mean	0.29	**0.11**	0.27	**0.35**	0.65	**0.47**	1.12	**1.30**

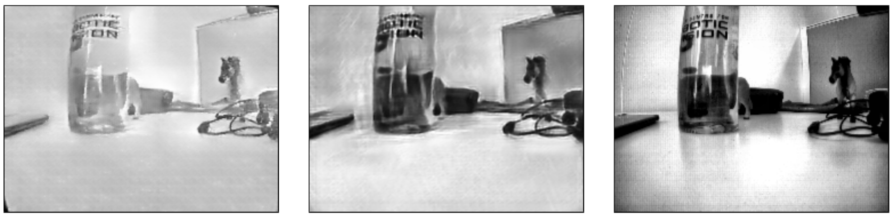
*Removed from mean tally for EV-FlowNet, as this sequence is part of the training set.

bution of training data, whereas FireNet is too small to benefit, perhaps resulting in training instability.

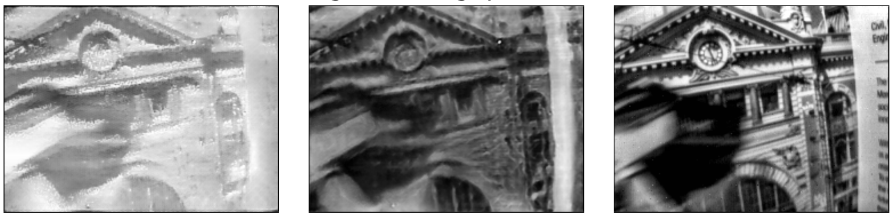
Since many existing datasets show scenes that are challenging for conventional cameras, I carefully selected sections where the image frames appeared to be higher quality (less blurred, better exposure etc., Table 6.4). However, I was also ultimately motivated to record my own dataset of high quality frames (HQF; Section 6.2.7) of which I evaluated the entire sequence. Figures 6.6, 6.7, 6.8, 6.9 show randomly sampled qualitative impressions of reconstructed images from HQF, IJRR, MVSEC and CED.



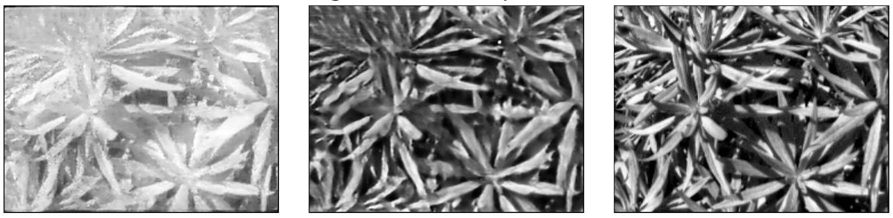
desk_slow



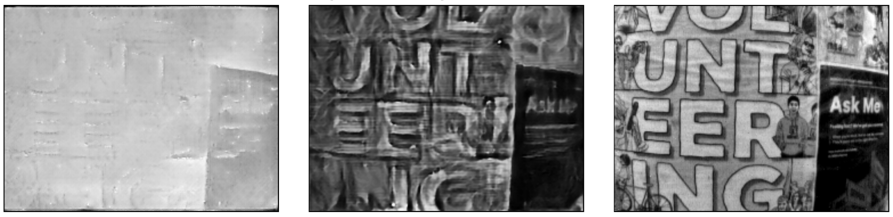
engineering_posters



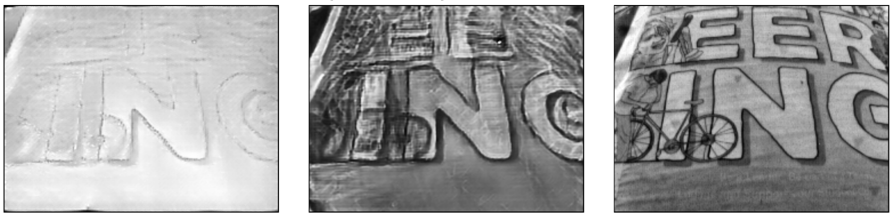
high_texture_plants



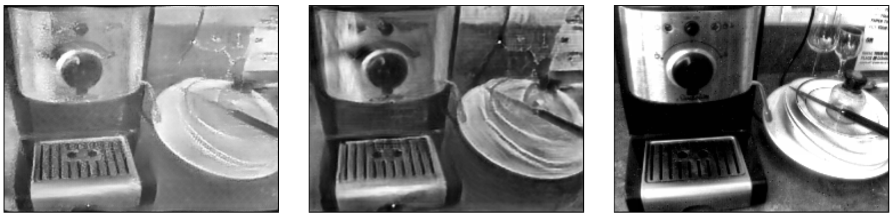
poster_pillar_1



poster_pillar_2



reflective_materials



slow_and_fast_desk

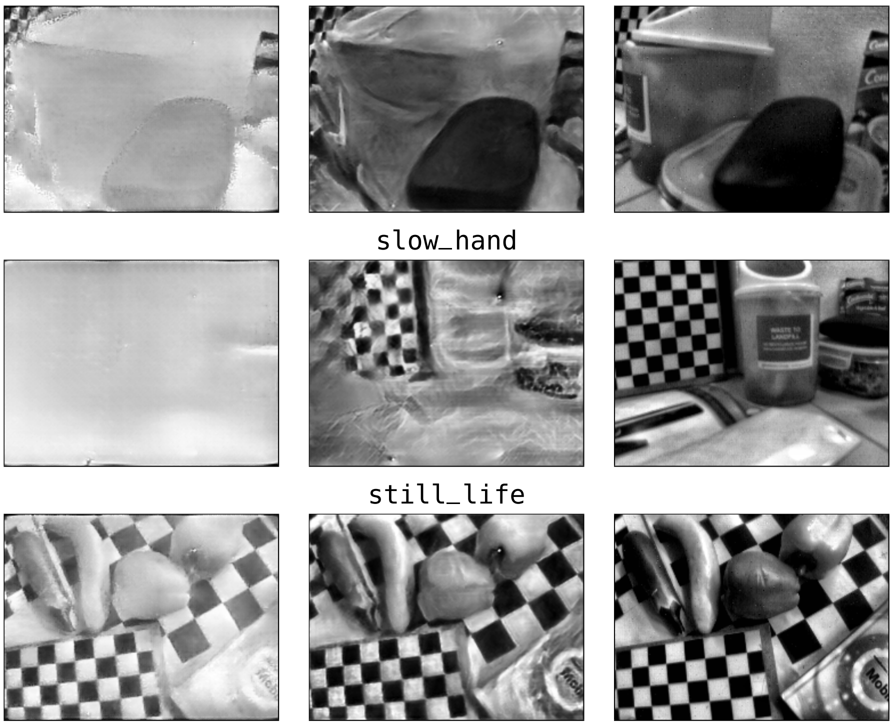
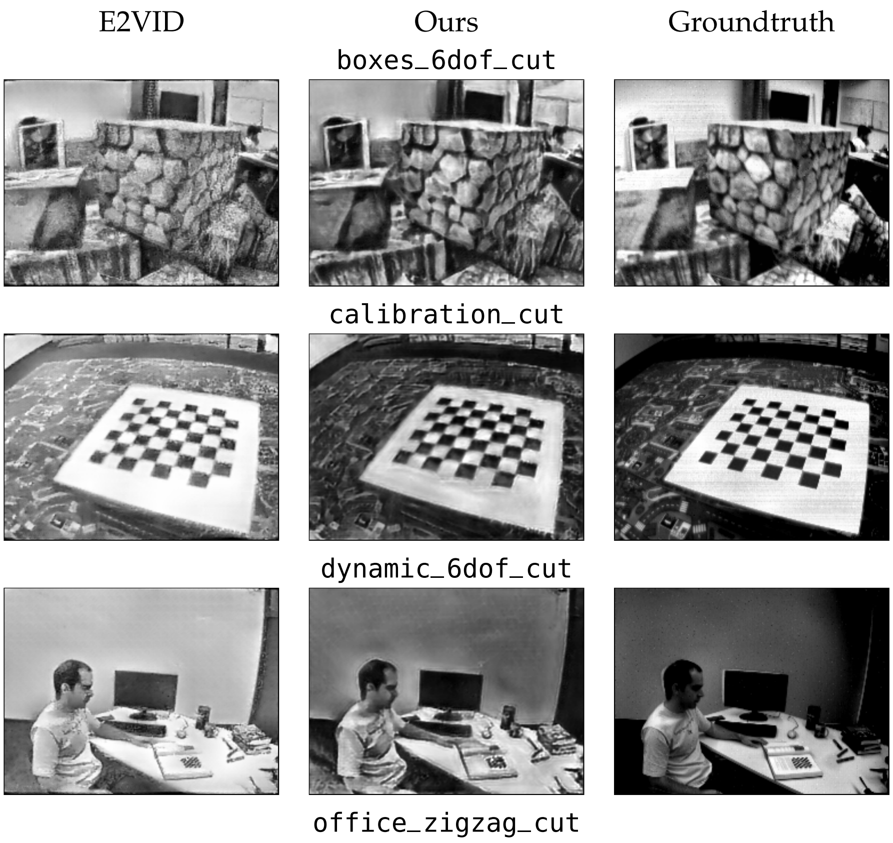
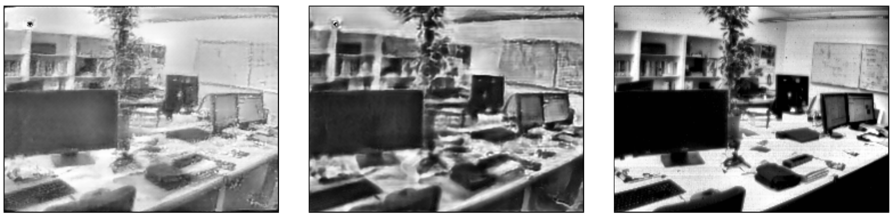
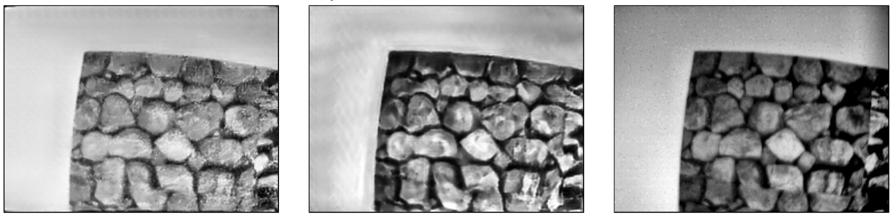


Table 6.6: Qualitative results for HQFD. Random selection, not cherry picked.

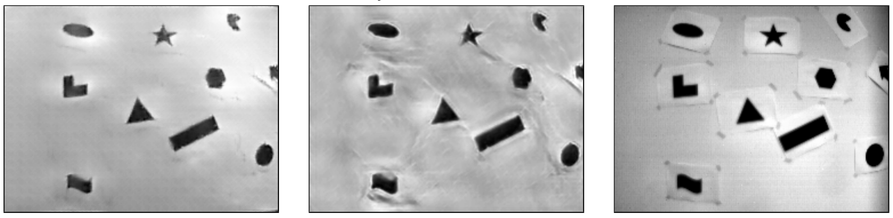




`poster_6dof_cut`



`shapes_6dof_cut`



`slider_depth_cut`

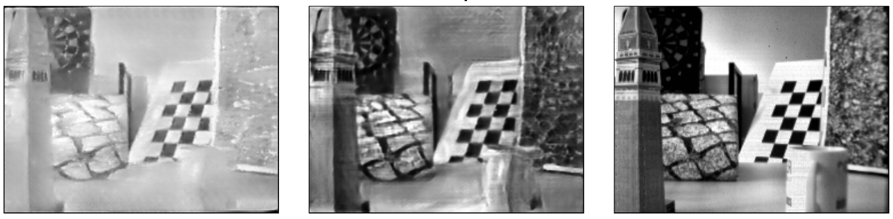
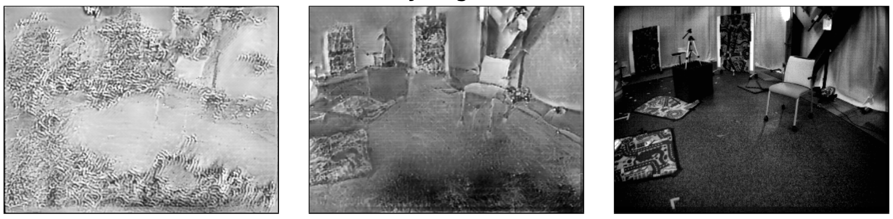


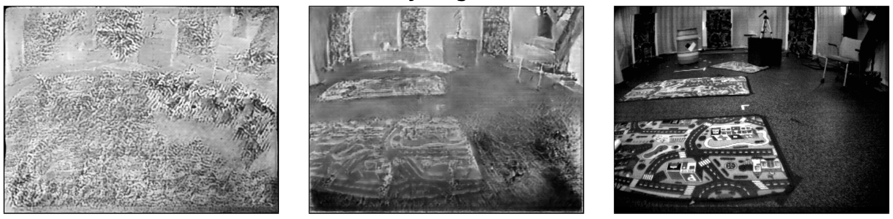
Table 6.7: Qualitative results for IJRR. Random selection, not cherry picked.

E2VID	Ours	Groundtruth

`indoor_flying1_data_cut`



`indoor_flying2_data_cut`



`indoor_flying3_data_cut`

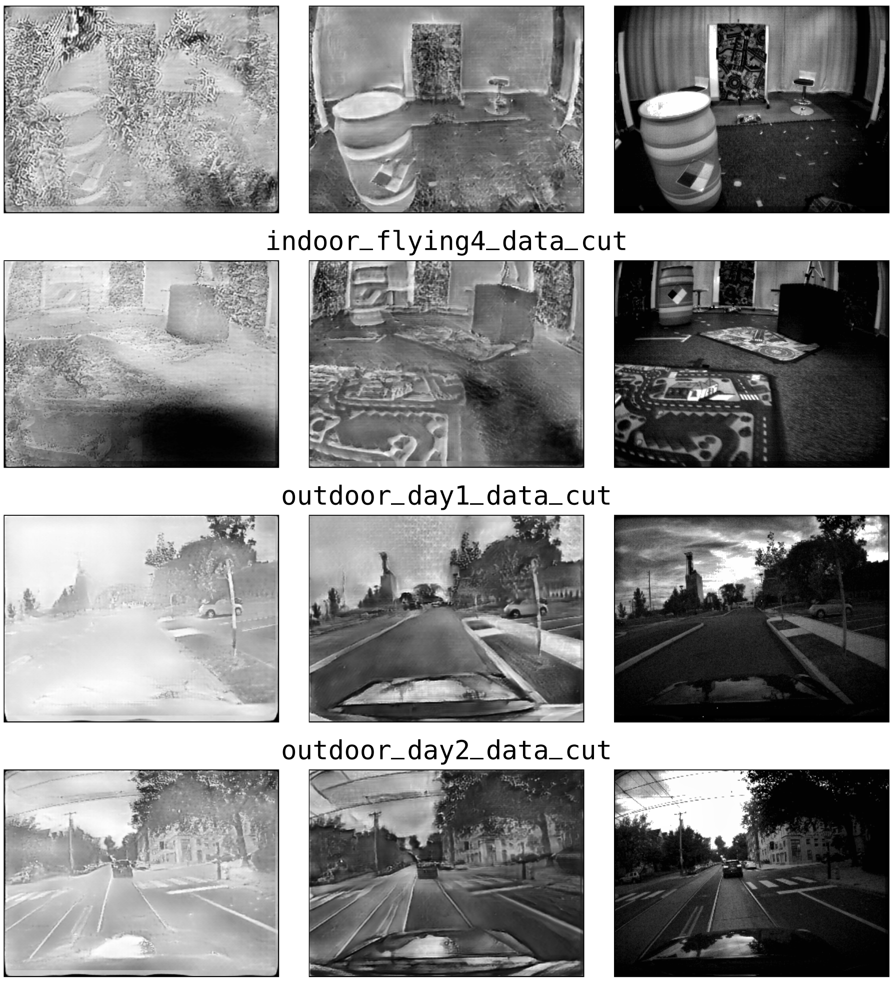
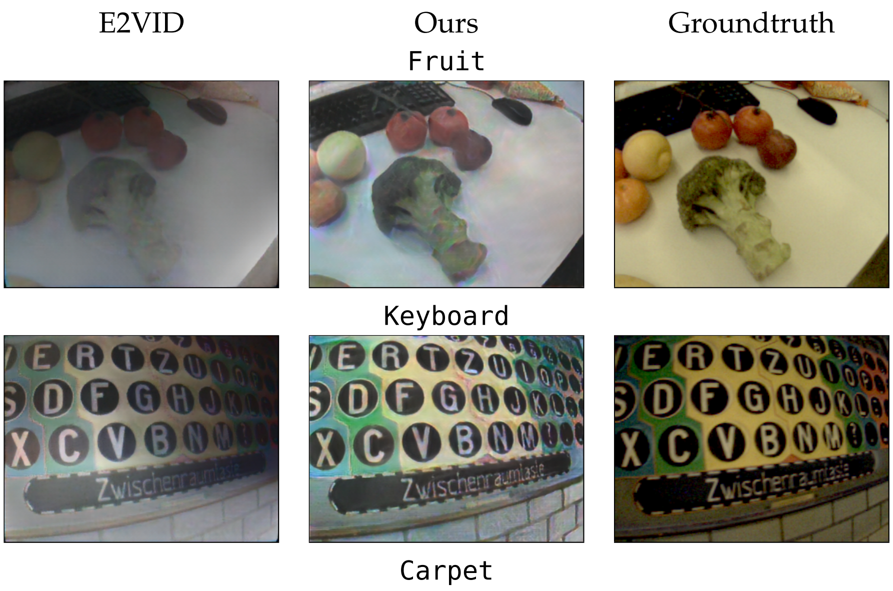


Table 6.8: Qualitative results for MVSEC. Random selection, not cherry picked.



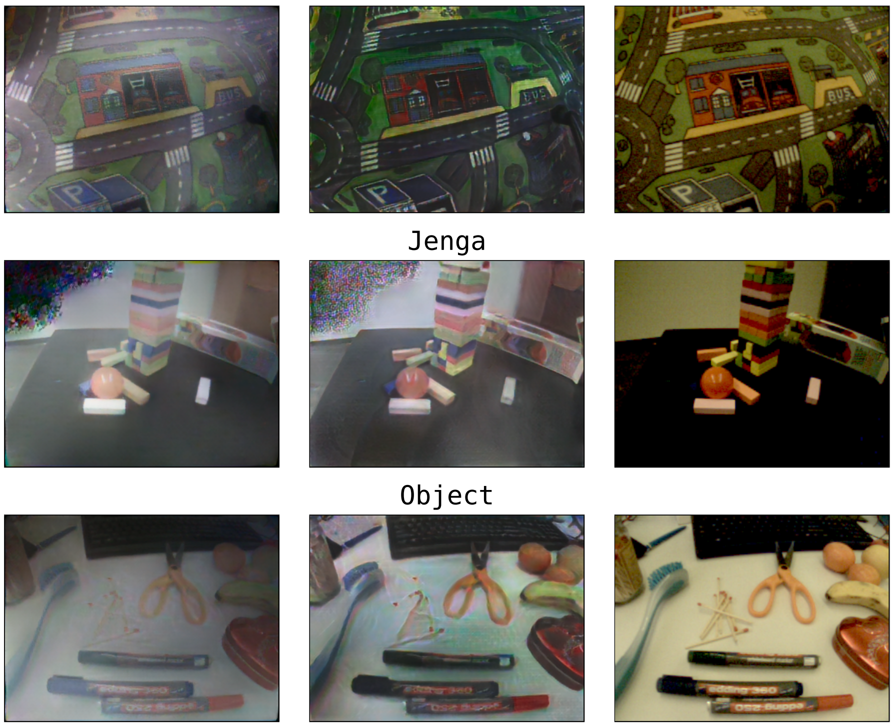


Table 6.9: Qualitative results for CED [Scheerlinck et al., 2019b]. Random selection, not cherry picked. As a matter of interest, the Jenga sequence shows a region of the scene where there is only blank wall, so few events have been generated, resulting in the peculiar artifacts seen in the top left corner.

Table 6.10: Comparison of various methods to optic flow estimated from Lidar depth and ego-motion sensors [Zhu et al., 2018a]. The average-endpoint-error to the Lidar estimate (AEE) and the percentage of pixels with AEE above 3 and greater than 5 % of the magnitude of the flow vector ($\%_{\text{Outlier}}$) are presented for each method (lower is better, best in bold). Zeros is the baseline error resulting from always estimating zero flow.

Dataset	outdoor_day1		outdoor_day2		indoor_flying1		indoor_flying2		indoor_flying3	
	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$
Zeros	4.31	0.39	1.07	**0.91**	1.10	**1.00**	1.74	**0.89**	1.50	**0.94**
EVFlow [Zhu et al., 2018b]	**0.49**	**0.20**	-	-	1.03	2.20	1.72	15.10	1.53	11.90
Ours	0.68	0.99	**0.82**	0.96	**0.56**	**1.00**	**0.66**	1.00	**0.59**	1.00
ECN* [Ye et al., 2019]	0.35	0.04	-	-	0.21	0.01	-	-	-	-

*ECN is trained on 80 % of the sequence and evaluated on the remaining 20 %. This prevents direct comparison, however we include their result for completeness sake.

6.3.1.2 Flow

In the absence of ground truth optic flow with which to compare, I use a warping loss [Gallego and Scaramuzza, 2017] as a proxy measure of prediction accuracy (referred to henceforth as flow warping loss, FWL). Essentially, events $E = (x_i, y_i, t_i, s_i)_{i=1,\dots,N}$ are warped by per-pixel optical flow $\phi = (u(x,y), v(x,y))^T$ to a reference time t' via

$$I(E, \phi) = \begin{pmatrix} x'_i \\ y'_i \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + (t' - t_i) \begin{pmatrix} u(x_i, y_i) \\ v(x_i, y_i) \end{pmatrix}. \tag{6.2}$$

The resulting image I becomes sharper if the flow is correct, as events are motion compensated. Sharpness can be evaluated using the variance of the image $\sigma^2(I)$ [Gallego et al., 2019; Stoffregen and Kleeman, 2019], where a higher value indicates a better flow estimate. Since image variance $\sigma^2(I)$ depends on scene structure and camera parameters, I normalize by the variance of the unwarped event image $I(E, 0)$ to obtain the *Flow Warp Loss (FWL)*:

$$\text{FWL} := \frac{\sigma^2(I(E,\phi))}{\sigma^2(I(E,0))}. \tag{6.3}$$

FWL < 1 implies the flow is worse than a baseline of zero flow. FWL enables evaluation on datasets without ground truth optic flow. While ground truth from the simulator was used during training, I evaluated on real data using FWL (Table 6.5). I believe training on ground truth (L1 loss) rather than FWL encourages dense flow predictions.

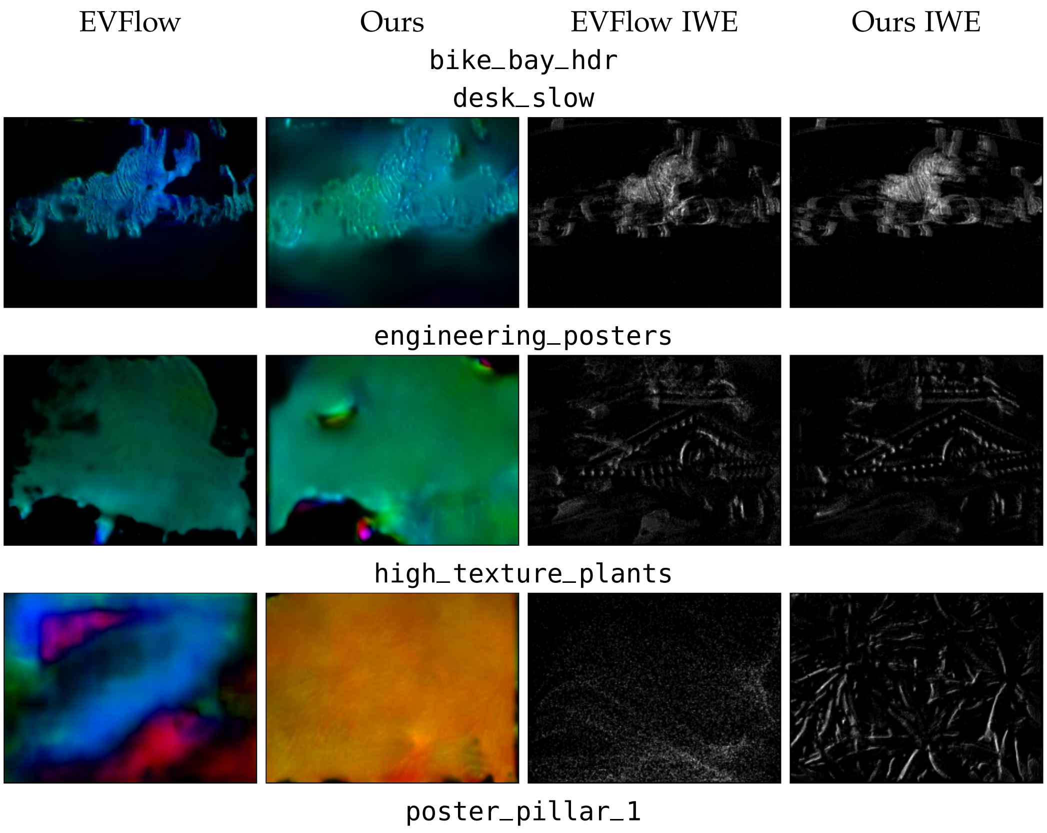
Table 6.11 shows average endpoint error (AEE) of optic flow on MVSEC [Zhu et al., 2018a]. MVSEC provides optic flow estimates computed from lidar depth and ego motion sensors as 'ground truth', allowing evaluation on average endpoint error (AEE) using code provided in [Zhu et al., 2018b]. However, lidar + ego motion derived ground truth is subject to sensor noise, thus, AEE may be an unreliable

Table 6.11: Comparison of various methods to optic flow estimated from Lidar depth and ego-motion sensors [Zhu et al., 2018a]. The average-endpoint-error to the Lidar estimate (AEE) and the percentage of pixels with AEE above 3 and greater than 5 % of the magnitude of the flow vector ($\%_{\text{Outlier}}$) are presented for each method (lower is better, best in bold). Zeros is the baseline error resulting from always estimating zero flow.

Dataset	outdoor_day1		outdoor_day2		indoor_flying1		indoor_flying2		indoor_flying3	
	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$
Zeros	4.31	0.39	1.07	**0.91**	1.10	**1.00**	1.74	**0.89**	1.50	**0.94**
EVFlow [Zhu et al., 2018b]	**0.49**	**0.20**	-	-	1.03	2.20	1.72	15.10	1.53	11.90
Ours	0.68	0.99	**0.82**	0.96	**0.56**	**1.00**	**0.66**	1.00	**0.59**	1.00
ECN* [Ye et al., 2019]	0.35	0.04	-	-	0.21	0.01	-	-	-	-

*ECN is trained on 80 % of the sequence and evaluated on the remaining 20 %. This prevents direct comparison, however we include their result for completeness sake.

metric on MVSEC. For example, predicting zero flow achieves near state-of-the-art in some cases on MVSEC using AEE, though not with the proposed metric FWL (by construction, predicting zero flow yields FWL = 1.0). Nevertheless I provide results on this metric for the sake of completeness. Figures 6.12, 6.13, 6.14 show randomly sampled qualitative impressions of optic flow and the image of warped events (IWE, eq. 6.2) from HQF, IJRR and MVSEC.



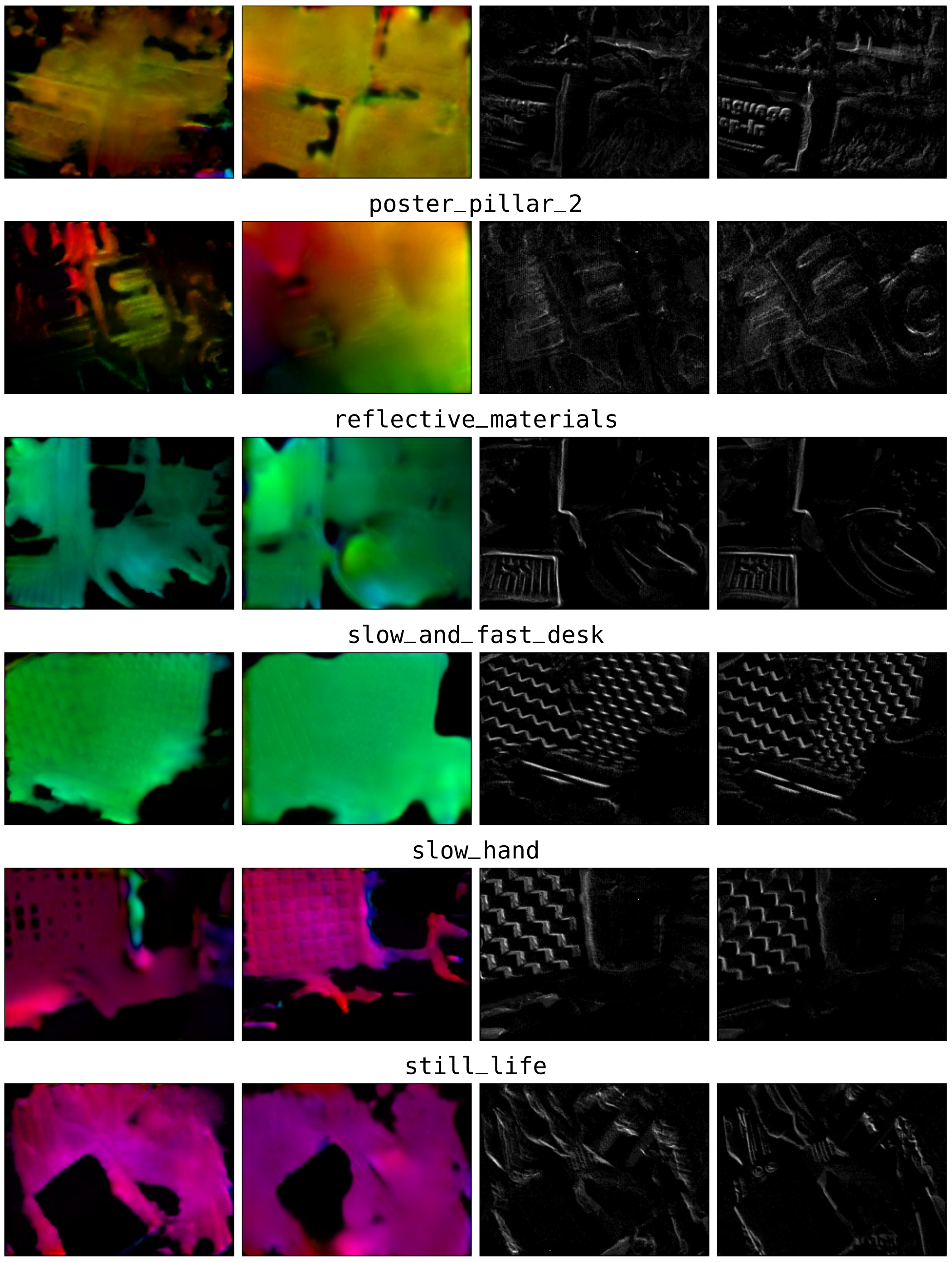


Table 6.12: Qualitative results for HQFD. Left: optic flow vectors represented in HSV color space, right: image of warped events (IWE). Random selection, not cherry picked.

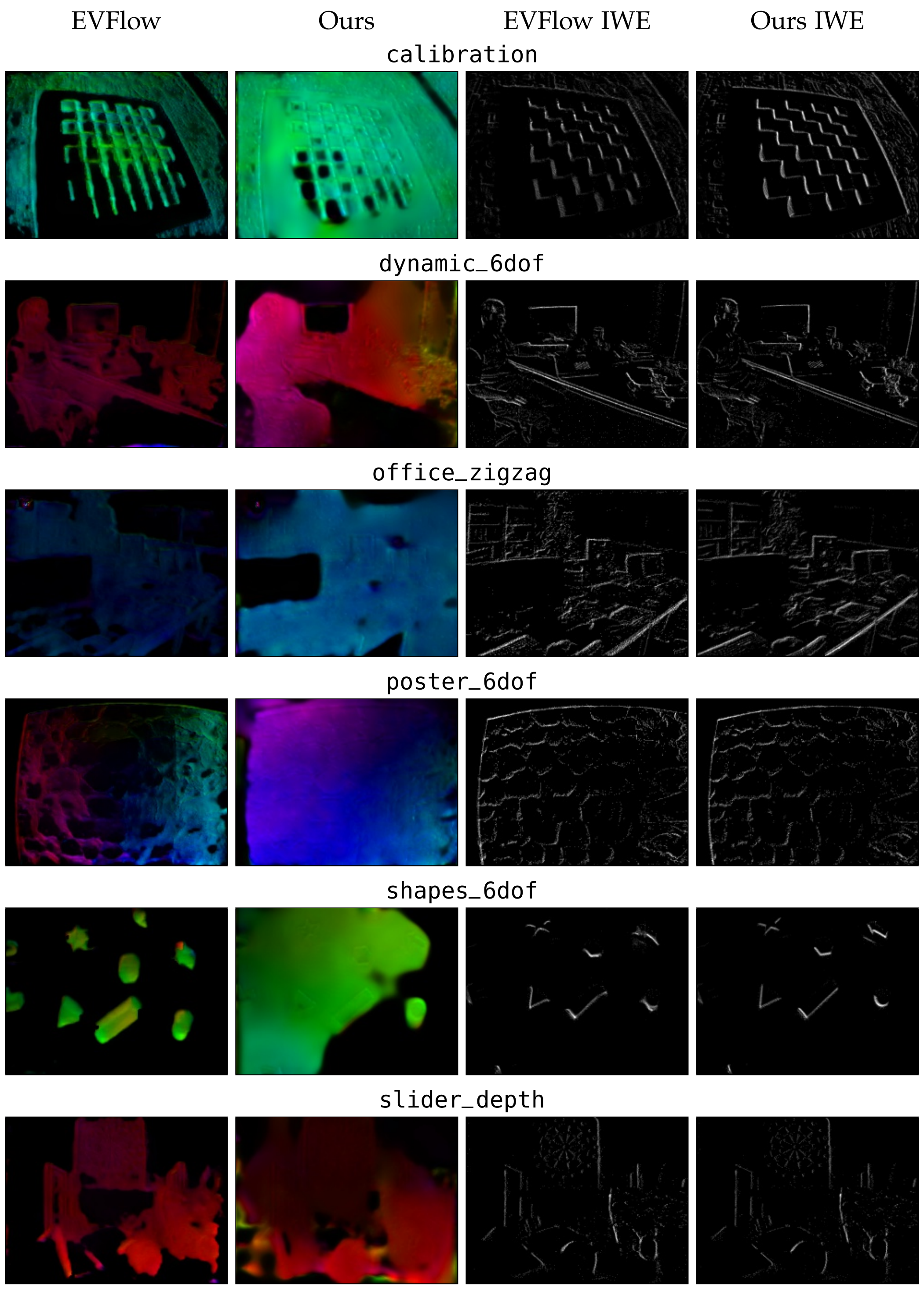


Table 6.13: Qualitative results for IJRR. Left: optic flow vectors represented in HSV color space, right: image of warped events (IWE). Random selection, not cherry picked.

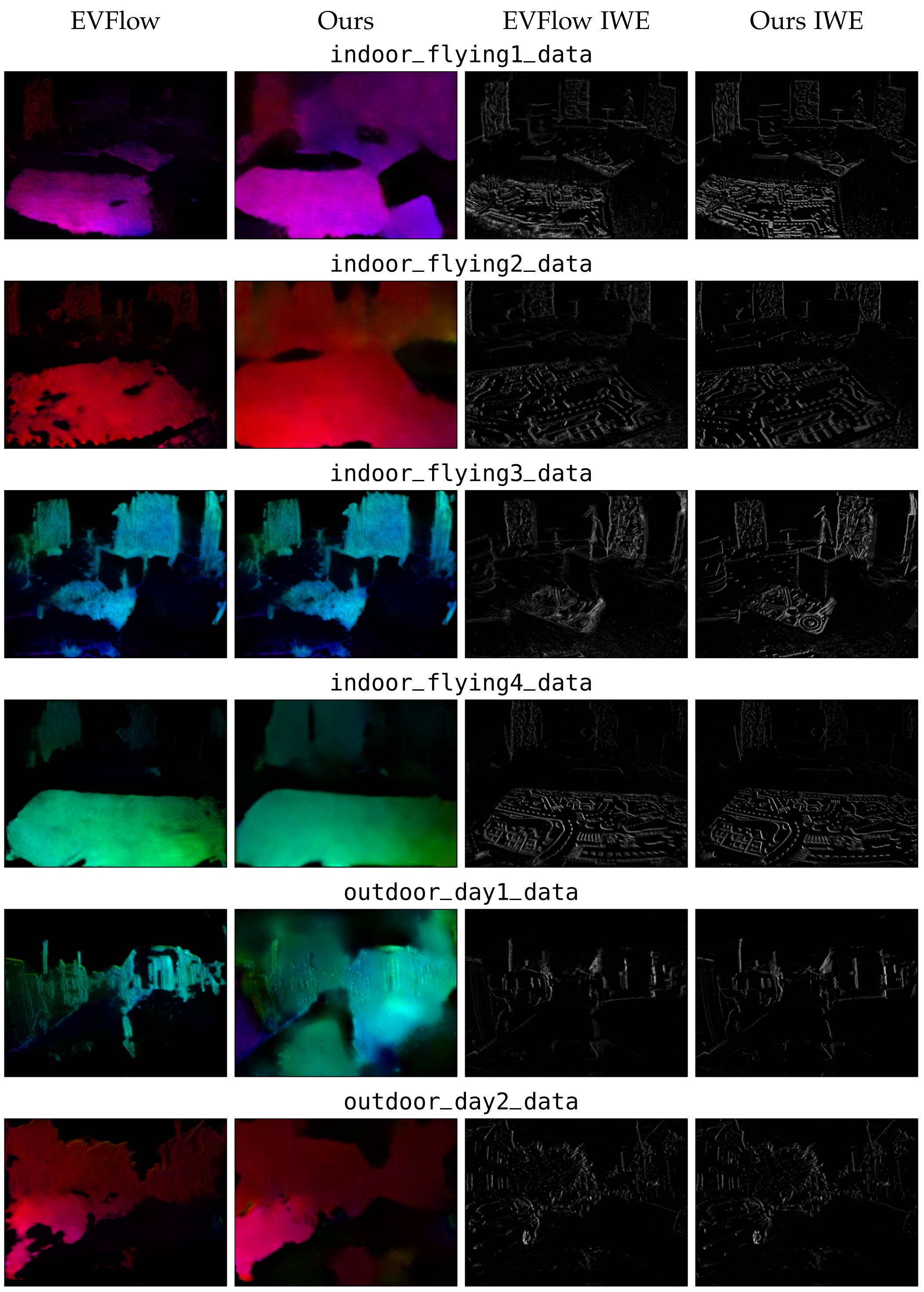


Table 6.14: Qualitative results for optic flow on MVSEC. Left: optic flow vectors represented in HSV color space, right: image of warped events (IWE). Random selection, not cherry picked.

Table 6.15: Evaluation of image reconstruction and optic flow networks trained on simulated datasets with a variety of contrast thresholds (CTs) from 0.2 to 1.5. 'All' is a dataset containing the full range of CTs from 0.2 to 1.5. All networks are trained for 200 epochs and evaluated on real datasets HQF (Section 6.2.7), IJRR [Mueggler et al., 2017b], MVSEC [Zhu et al., 2018a]. For image reconstruction networks we report mean squared error (MSE), structural similarity (SSIM) [Wang et al., 2004] and perceptual loss (LPIPS) [Zhang et al., 2018]. Optic flow networks are evaluated using flow warp loss (*FWL*) described in Section 6.3.1.2. Key: **best** | *second best*.

Contrast threshold	HQF				IJRR				MVSEC			
	MSE	SSIM	LPIPS	*FWL*	MSE	SSIM	LPIPS	*FWL*	MSE	SSIM	LPIPS	*FWL*
0.20	0.05	0.50	0.38	0.89	**0.04**	**0.60**	**0.25**	4.12	0.10	**0.35**	0.55	0.14
0.50	**0.04**	*0.51*	0.36	0.89	0.04	0.57	0.27	4.02	*0.10*	0.31	0.52	0.14
0.75	0.05	**0.51**	**0.36**	0.89	0.05	0.56	0.28	4.18	0.11	0.29	0.53	0.14
1.00	0.05	0.48	0.36	0.88	0.05	0.53	0.29	3.93	0.12	0.27	*0.51*	0.14
1.50	0.05	0.47	0.38	0.86	0.06	0.52	0.30	3.70	0.09	0.30	0.52	0.14
All	*0.05*	0.50	**0.36**	**0.92**	**0.04**	*0.59*	*0.27*	**4.53**	**0.08**	*0.34*	**0.51**	**0.14**

Table 6.16: Dynamic range of reconstructed images from IJRR [Mueggler et al., 2017b]: original E2VID [Rebecq et al., 2020b] versus E2VID retrained on simulated datasets covering a range of contrast thresholds CTs. We report the mean dynamic range of the 10th-90th percentile of pixel values.

	Original E2VID	Retrained					
Contrast threshold	∼0.18	0.2	0.5	0.75	1.0	1.5	All
Dynamic range	77.3	89.2	103.7	105.9	104.8	100.0	103.3

6.3.2 Contrast Thresholds

I investigated the impact of simulator CT parameter (see Sec. 6.2.1) by retraining several networks on simulated datasets with CTs ranging from 0.2 to 1.5. Each dataset contained the same sequences, differing only in CT. Table 6.15 shows that for reconstruction (evaluated on LPIPS), IJRR is best on a lower CT ≈ 0.2, while MVSEC is best on high CT ≈ 1.0. Best or runner up performance was achieved when a wide range of CTs is used, indicating that exposing a network to additional event statistics outside the inference domain is not harmful, and may be beneficial. I also believe that a symptom of training with low CTs (thus higher $\frac{\text{events}}{\text{pix}\cdot\text{s}}$) is a loss of dynamic range in the output images. This occurs because the network, trained using many events, receives only few at inference, thus narrowing the range of output values. When retraining the original E2VID network, dynamic range increases with CTs (Table 6.16).

Table 6.17: Mean LPIPS [Zhang et al., 2018] on our HQF dataset, IJRR [Mueggler et al., 2017b] and MVSEC [Zhu et al., 2018a], for various training hyperparameter configurations. E2VID architecture retrained from scratch in all experiments. Key: L40/L120=sequence length 40/120, N=noise augmentation during training.

Model	HQF			IJRR			MVSEC		
	MSE	SSIM	LPIPS	MSE	SSIM	LPIPS	MSE	SSIM	LPIPS
L40	0.044	**0.583**	0.296	0.042	**0.650**	0.229	0.151	0.330	0.526
L40N	**0.033**	0.579	**0.256**	**0.034**	0.636	**0.224**	0.105	**0.346**	**0.467**
L120	0.040	0.544	0.279	0.038	0.619	0.237	0.132	0.311	0.478
L120N	0.036	0.547	0.290	0.040	0.608	0.241	**0.099**	0.344	0.498

6.3.3 Training Noise and Sequence Length

To determine the impact of sequence length and noise augmentation during training, I retrained E2VID architecture using sequence length 40 (L40) and 120 (L120), with and without noise augmentation (N) (see Table 6.17). Increasing sequence length from 40 to 120 didn't impact results significantly. Noise augmentation during training improved performance of L40 models by $\sim$ 5-10 %, while giving mixed results on different datasets for L120 models. Qualitatively, adding more noise encourages networks to smooth outputs, while less noise may encourage the network to 're-construct' noise events, resulting in artifacts (Figure 6.2) observed in E2VID [Rebecq et al., 2020b] (trained without noise).

6.4 Discussion

The significant improvements gained by training models on the proposed synthetic dataset exemplify why it is important to try and minimise the gap between real and simulated events in both the event rate induced by varying the contrast thresholds and the dynamics of the simulation scenes. The results are quite clear on this, with consistent improvements across tasks (reconstruction and optic flow) and architectures (recurrent networks like E2VID, and U-Net based flow estimators) of up to 40 %. I believe this highlights the importance for researchers to pay attention to the properties of the events they are training on; are the settings of the camera or simulator such that they are generating more or less events? Are the scenes they are recording representative of the wide range of scenes that are likely to be encountered during inference?

In particular, it seems that previous works have inadvertently overfit their models to the events found in the chosen target dataset. EV-FlowNet performs better on sequences whose dynamics are similar to the slow, steady scenes in MVSEC used for training, examples being `poster_pillar_2` or `desk_slow` from HQF that feature long pauses and slow motions, where EV-FlowNet is on par or better than ours. For researchers looking to use an off-the-shelf pretrained network, my model may be a better fit, since it applies for a greater variety of sensors and scenes. A further advantage of our model that is not reflected in the FWL metric, is that training in simulation allows our model to predict *dense* flow (Figure 6.12), a challenge for prior self-supervised methods.

Similarly, the results speak for themselves on image reconstruction. While I outperform E2VID [Rebecq et al., 2020b] on all datasets, the smallest gap is on IJRR, the dataset I found to have lower CTs. E2VID performs worst on MVSEC that contains higher CTs, consistent with the finding that performance is driven by similarity between training and evaluation event data.

I believe the proposed framework can be easily extended to synthetic events generated from fully synthetic 3D scenes (e.g., urban environments from Ai.Reverie). There may be benefits to including 3D scenes since this would better match reality, though even 2D training data used in this chapter generalised well to real test sequences with no obvious '2D artifacts'.

In conclusion, future networks trained with synthetic data from ESIM or other simulators should take care to ensure the statistics of their synthetic data match the final use-case, using large ranges of CT values and appropriate noise and pause augmentation in order to ensure generalised models.

Chapter 7

Conclusion

Inspired by biology, event cameras embody a paradigm shift from conventional 'frame-based' video to asynchronous event-driven vision. Responding only to per-pixel changes in brightness enables lower latency, bandwidth and power requirements, and higher dynamic range (HDR) than frame-based cameras. Similar to biological eyes, event cameras output a series of spikes (called events) that must be processed to form an image of the scene. This thesis has presented algorithms for processing raw events to obtain images, features and optic flow, drawing on filter theory and machine learning. Chapters 3, 4 presented fast asynchronous algorithms for event-based video reconstruction and convolution, while chapters 5, 6 explored machine learning approaches that improve quality at the expense of computation.

I wanted to do this research because event cameras inspire a fresh perspective on traditional image processing, and I saw an opportunity to combine knowledge from the seemingly disparate field of systems theory with computer vision. The continuous-time linear filtering approach and Dirac delta model introduced in chapters 3 and 4 ended up being a natural fit for asynchronous event data, reconciling discrete events that live in continuous time. The approach was implemented asynchronously, leveraging the low latency and sparsity of event cameras to enable efficient computation, even on conventional CPU hardware - further gains are expected on specialised hardware (e.g., FPGA). During the course of my PhD I was lucky enough to witness an explosion of machine learning works applied to event cameras, in particular convolutional neural networks (CNNs). A common criticism is that CNNs degrade event cameras by effectively converting their output from asynchronous low latency to frame-based high latency in the pre-processing event batching step. CNNs were also believed to be computationally expensive and slow compared to the lightning speed of event cameras. Thus, I saw an opportunity to research lightweight efficient network architectures that ran considerably faster than prevailing networks (chapter 5). Another problem was that networks evaluated on one dataset typically had widely varying results on other datasets, i.e., generalisability issues were hidden by the evaluation pipeline and narrow datasets. Thus, I was motivated to seek strategies for improving evaluation and reducing the sim-to-real gap to improve generalisability as discussed in chapter 6. While I have separated my research into two distinct categories of asynchronous algorithms vs. synchronous machine learning approaches, I believe there is great potential for further intersection of machine learning with asynchronous processing, whether that be spiking neural networks, asynchronous convolutional networks or something else. Ultimately, asynchronous processing is necessary to eliminate batching latency and take full advantage of event cameras' sparsity and speed.

Chapter 3 presented a simple filtering approach using raw events or combining (low quality) video with events to reconstruct high temporal resolution, HDR video. Key contributions were: (i) formulation of a continuous-time internal image state (instead of discrete sequence of frames), (ii) continuous-time event model and filter design (complementary, high-pass), (iii) asynchronous update scheme based on exact interpolation and (iv) open-source C++ code[1]. The approach is computationally efficient and $O(n)$ scaling with the number of events, able to process up to 20M events/s on an i7 CPU making it the fastest event-based video reconstruction algorithm to date. However, the algorithm suffers from smearing artifacts in some scenarios. Nevertheless, it has proven useful as a real-time visualisation tool for the event camera community.

Chapter 4 extends the continuous-time image state idea from chapter 3 to spatial convolutions. Rather than computing the convolution of an image, chapter 4 proposes to incrementally and asynchronously update a convolved image state using a high-pass filter presented in chapter 3. The convolved image state may be used for downstream applications e.g., corner detection. The approach is far more efficient than convolving the full image with every event, however, in practise, one would not typically require per-event temporal resolution. If lower temporal resolution is acceptable, images may be first reconstructed then periodically convolved at low frequency. Thus, asynchronous convolutions are suited to high temporal resolution applications (e.g., high-speed scenarios).

Chapter 5 revisits the task of event-based video reconstruction from a machine learning perspective, aiming to find a lightweight network architecture containing only necessary components. It fixes some of the smearing/ghosting artifacts in chapter 3 at the cost of greater computational complexity and cost. The proposed *Fast Image Reconstruction from Events Network (FireNet)* runs $3\times$ faster and is a fraction of the size (<1%) of contemporary state-of-the-art networks while achieving similar reconstruction quality. Ablation studies reveal that recurrent units (such as convolutional gated recurrent units) are key for event-based image reconstruction, especially as the size and receptive field of the network decreases, as with FireNet. FireNet provides a point on the quality-speed trade-off curve and is an ideal open-source[2] network for when quality is desirable at moderate computational cost.

Chapter 6 takes a deeper dive into how to improve performance and generalisability of event-based convolutional neural networks for video reconstruction and optic flow, aiming to probe existing state-of-the-art networks and fix their shortcomings. Key insights were: (i) Synthetic training data is more effective when large range of contrast thresholds are used, (ii) Random noise and temporal pause augmentation at train time may improve performance and (iii) Evaluation on limited data

[1]https://github.com/cedric-scheerlinck/dvs_image_reconstruction
[2]code and weights: https://cedricscheerlinck.com/firenet

may hide overfitting as in previous works, motivating a new High Quality Frames Dataset[3]. Since existing (state-of-the-art) network architectures were retrained using new training data, gains in performance may be attributed to the training data rather than choice of architecture hyperparameters. This chapter demonstrates the importance of tuning simulator settings (for training data) to cover the distribution of real data ultimately used to evaluate networks. In addition, previous works were thoroughly analysed revealing that they may have overfit to the dataset they were evaluated on to some extent, highlighting the need for evaluation on datasets with diverse motion types and camera settings.

My results on event-based video reconstruction reveal that the raw event data stream contains rich information about the scene - at least enough to reconstruct photorealistic high speed, HDR video. While my results set a benchmark on the quality of information that can be extracted from event cameras, I believe we are still far from the upper limit. Videos and images reconstructed from events may stand in for conventional image frames for classification, segmentation, navigation etc., turning event cameras into powerful front-end devices for existing systems e.g., self-driving cars. Additionally, I believe the lessons learnt generalise beyond video reconstruction to other event-based vision challenges such as feature detection, optic flow and segmentation. In particular, (i) internal states as discussed in chapters 3, 4 and reappear in the form of recurrent units in chapters 5, 6 are crucial for aggregating temporal information and allowing asynchronous update operations, and (ii) network architecture (chapter 5) and training parameter choices (chapter 6) are key to improving computational efficiency and generalisability to real world data. This thesis has pushed the frontier on techniques to extract information from events, along both the speed and quality axes, allowing humans and intelligent systems alike to *see* with event cameras. Let's see where this bio-inspired vision paradigm will take us.

[3]https://cedricscheerlinck.com/20ecnn

Bibliography

AGRAWAL, A.; CHELLAPPA, R.; AND RASKAR, R., 2005. An algebraic approach to surface reconstruction from gradient fields. In *Int. Conf. Comput. Vis. (ICCV)*, 174–181. doi:10.1109/iccv.2005.31. (cited on pages xviii, 15, 21, 49, and 50)

AGRAWAL, A.; RASKAR, R.; AND CHELLAPPA, R., 2006. What is the range of surface reconstructions from a gradient field? In *Eur. Conf. Comput. Vis. (ECCV)*, 578–591. (cited on pages xviii, 15, 22, 49, and 50)

AIMAR, A.; MOSTAFA, H.; CALABRESE, E.; RIOS-NAVARRO, A.; TAPIADOR-MORALES, R.; LUNGU, I.; MILDE, M. B.; CORRADI, F.; LINARES-BARRANCO, A.; LIU, S.; AND DELBRÜCK, T., 2018. NullHop: A flexible convolutional neural network accelerator based on sparse representations of feature maps. *IEEE Trans. Neural Netw. Learn. Syst.*, (2018). doi:10.1109/TNNLS.2018.2852335. (cited on page 53)

AKOLKAR, H.; IENG, S.-H.; AND BENOSMAN, R., 2018. See before you see: Real-time high speed motion prediction using fast aperture-robust event-driven visual flow. *arXiv e-prints*, (2018). https://arxiv.org/abs/1811.11135. (cited on page 17)

ALMATRAFI, M. M. AND HIRAKAWA, K., 2019. DAViS camera optical flow. *IEEE Trans. Comput. Imaging*, (2019), 1–11. doi:10.1109/tci.2019.2948787. (cited on page 17)

ALOM, M. Z.; HASAN, M.; YAKOPCIC, C.; TAHA, T. M.; ; AND ASARI, V. K., 2018. Recurrent residual convolutional neural network based on U-Net (R2U-Net) for medical image segmentation. *arXiv e-prints*, (May 2018). https://arxiv.org/abs/1802.06955. (cited on page 15)

ALONSO, I. AND MURILLO, A. C., 2019. EV-SegNet: Semantic segmentation for event-based cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. (cited on page 55)

ALZUGARAY, I. AND CHLI, M., 2018. Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robot. Autom. Lett.*, 3, 4 (Oct. 2018), 3177–3184. doi:10.1109/LRA.2018.2849882. (cited on pages xviii, 51, and 52)

AUNG, M. T.; TEO, R.; AND ORCHARD, G., 2018. Event-based plane-fitting optical flow for dynamic vision sensors in FPGA. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*. doi:10.1109/iscas.2018.8351588. (cited on page 17)

BAKER, S.; LEWIS, D. S. J.; ROTH, S.; BLACK, M. J.; AND SZELISKI, R., 2011. A database and evaluation methodology for optical flow. *Int. J. Comput. Vis.*, 92, 1 (2011), 1–31. (cited on page 17)

Ballas, N.; Yao, L.; Pal, C.; and Courville, A., 2016. Delving deeper into convolutional networks for learning video representations. In *Int. Conf. Learn. Representations (ICLR)*. https://arxiv.org/abs/1511.06432. (cited on page 57)

Bardow, P., 2018. *Estimating General Motion and Intensity from Event Cameras*. Ph.D. thesis, Imperial College London, Department of Computing. (cited on pages xv, 7, 15, 16, and 17)

Bardow, P.; Davison, A. J.; and Leutenegger, S., 2016. Simultaneous optical flow and intensity estimation from an event camera. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 884–892. doi:10.1109/CVPR.2016.102. (cited on pages xvi, 14, 17, 21, 30, 33, and 34)

Barua, S.; Miyatani, Y.; and Veeraraghavan, A., 2016. Direct face detection and video reconstruction from event cameras. In *IEEE Winter Conf. Appl. Comput. Vis. (WACV)*. doi:10.1109/WACV.2016.7477561. (cited on pages 14 and 21)

Belbachir, A. N.; Schraml, S.; Mayerhofer, M.; and Hofstaetter, M., 2014. A novel HDR depth camera for real-time 3D 360-degree panoramic vision. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. (cited on pages xv, 14, and 21)

Benosman, R.; Clercq, C.; Lagorce, X.; Ieng, S.-H.; and Bartolozzi, C., 2014. Event-based visual flow. *IEEE Trans. Neural Netw. Learn. Syst.*, 25, 2 (2014), 407–417. doi:10.1109/TNNLS.2013.2273537. (cited on pages 12 and 42)

Brandli, C.; Berner, R.; Yang, M.; Liu, S.-C.; and Delbruck, T., 2014a. A 240x180 130dB 3us latency global shutter spatiotemporal vision sensor. *IEEE J. Solid-State Circuits*, 49, 10 (2014), 2333–2341. doi:10.1109/JSSC.2014.2342715. (cited on pages xv, 8, 9, 10, 12, 22, 23, 30, 41, 43, 47, 59, and 74)

Brandli, C.; Muller, L.; and Delbruck, T., 2014b. Real-time, high-speed video decompression using a frame- and event-based DAVIS sensor. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*, 686–689. doi:10.1109/ISCAS.2014.6865228. (cited on pages xv, xvi, xvii, 10, 11, 12, 22, 28, 30, 37, 38, 44, and 45)

Camunas-Mesa, L.; Zamarreno-Ramos, C.; Linares-Barranco, A.; Acosta-Jimenez, A. J.; Serrano-Gotarredona, T.; and Linares-Barranco, B., 2012. An event-driven multi-kernel convolution processor module for event-driven vision sensors. *IEEE Journal of Solid-State Circuits*, 47, 2 (Feb. 2012), 504–517. doi:10.1109/jssc.2011.2167409. (cited on page 53)

Cannici, M.; Ciccone, M.; Romanoni, A.; and Matteucci, M., 2019. Asynchronous convolutional networks for object detection in neuromorphic cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. (cited on pages 13 and 53)

Caporale, N. and Dan, Y., 2008. Spike timing-dependent plasticity: A hebbian learning rule. *Annual Review of Neuroscience*, 31, 1 (Jul. 2008), 25–46. doi:10.1146/annurev.neuro.31.060407.125639. (cited on page 13)

Censi, A. and Scaramuzza, D., 2014. Low-latency event-based visual odometry. In *IEEE Int. Conf. Robot. Autom. (ICRA)*, 703–710. doi:10.1109/ICRA.2014.6906931. (cited on page 42)

Chung, J.; Gulcehre, C.; Cho, K.; and Bengio, Y., 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NeurIPS Workshop on Deep Learning*. arxivi://arxiv.org/abs/1412.3555. (cited on page 57)

Cook, M.; Gugelmann, L.; Jug, F.; Krautz, C.; and Steger, A., 2011. Interacting maps for fast visual interpretation. In *Int. Joint Conf. Neural Netw. (IJCNN)*, 770–776. doi:10.1109/IJCNN.2011.6033299. (cited on pages 14, 17, and 21)

Davies, M.; Srinivasa, N.; Lin, T.-H.; Chinya, G.; Cao, Y.; Choday, S. H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al., 2018. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38, 1 (2018), 82–99. (cited on page 53)

Delbruck, T., 2008. Frame-free dynamic digital vision. In *Proc. Int. Symp. Secure-Life Electron.*, 21–26. (cited on page 10)

Delbruck, T.; Hu, Y.; and He, Z., 2020. V2E: From video frames to realistic DVS event camera streams. *arXiv e-prints*, (Jun. 2020). http://arxiv.org/abs/2006.07722. (cited on page 9)

Delbruck, T. and Liu, S.-C., 2019. Data-driven neuromorphic dram-based cnn and rnn accelerators. In *2019 53rd Asilomar Conference on Signals, Systems, and Computers*, 500–506. IEEE. (cited on page 53)

Dikov, G.; Firouzi, M.; Röhrbein, F.; Conradt, J.; and Richter, C., 2017. Spiking cooperative stereo-matching at 2ms latency with neuromorphic hardware. In *Conf. Biomimetic and Biohybrid Systems*, 119–137. (cited on page 13)

Dosovitskiy, A.; Fischer, P.; Ilg, E.; Häusser, P.; Hazirbaş, C.; Golkov, V.; van der Smagt, P.; Cremers, D.; and Brox, T., 2015. FlowNet: Learning optical flow with convolutional networks. In *Int. Conf. Comput. Vis. (ICCV)*, 2758–2766. doi:10.1109/ICCV.2015.316. (cited on page 71)

Event-based Vision Resources, 2017. https://github.com/uzh-rpg/event-based_vision_resources. (cited on page 10)

Finateu, T.; Niwa, A.; Matolin, D.; Tsuchimoto, K.; Mascheroni, A.; Reynaud, E.; Mostafalu, P.; Brady, F.; Chotard, L.; LeGoff, F.; Takahashi, H.; Wakabayashi, H.; Oike, Y.; and Posch, C., 2020. 5.10 a 1280×720 back-illuminated stacked temporal contrast event-based vision sensor with 4.86¯m pixels, 1.066geps readout, programmable event-rate controller and compressive data-formatting pipeline. In *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*. doi:10.1109/isscc19947.2020.9063149. (cited on page 10)

Franklin, G. F.; Powell, J. D.; Workman, M. L.; et al., 1998. *Digital control of dynamic systems*, vol. 3. Addison-wesley Menlo Park, CA. (cited on page 25)

Furber, S. B.; Galluppi, F.; Temple, S.; and Plana, L. A., 2014. The SpiNNaker project. *Proc. IEEE*, 102, 5 (May 2014), 652–665. doi:10.1109/jproc.2014.2304638. (cited on pages 13 and 53)

Gallego, G.; Delbruck, T.; Orchard, G. M.; Bartolozzi, C.; Taba, B.; Censi, A.; Leutenegger, S.; Davison, A.; Conradt, J.; Daniilidis, K.; and Scaramuzza, D., 2020a. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2020). doi:10.1109/tpami.2020.3008413. (cited on pages xv, 7, 9, and 10)

Gallego, G.; Delbruck, T.; Orchard, G. M.; Bartolozzi, C.; Taba, B.; Censi, A.; Leutenegger, S.; Davison, A.; Conradt, J.; Daniilidis, K.; and Scaramuzza, D., 2020b. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2020). doi:10.1109/tpami.2020.3008413. (cited on page 10)

Gallego, G.; Gehrig, M.; and Scaramuzza, D., 2019. Focus is all you need: Loss functions for event-based vision. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. (cited on pages 17 and 84)

Gallego, G.; Rebecq, H.; and Scaramuzza, D., 2018. 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)*, 3867–3876. doi:10.1109/CVPR.2018.00407. (cited on page 17)

Gallego, G. and Scaramuzza, D., 2017. Accurate angular velocity estimation with an event camera. *IEEE Robot. Autom. Lett.*, 2, 2 (2017), 632–639. doi:10.1109/LRA.2016.2647639. (cited on page 84)

Gehrig, D.; Loquercio, A.; Derpanis, K. G.; and Scaramuzza, D., 2019a. End-to-end learning of representations for asynchronous event-based data. In *Int. Conf. Comput. Vis. (ICCV)*. (cited on pages 15, 17, 55, and 70)

Gehrig, D.; Rebecq, H.; Gallego, G.; and Scaramuzza, D., 2019b. EKLT: Asynchronous photometric feature tracking using events and frames. *Int. J. Comput. Vis.*, (2019). doi:10.1007/s11263-019-01209-w. (cited on page 39)

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y., 2014. Generative adversarial nets. In *Conf. Neural Inf. Process. Syst. (NeurIPS)*, 2672–2680. (cited on page 15)

Haessig, G.; Cassidy, A.; Alvarez-Icaza, R.; Benosman, R.; and Orchard, G., 2018. Spiking optical flow for event-based sensors using IBM's truenorth neurosynaptic system. *IEEE Trans. Biomed. Circuits Syst.*, 12, 4 (Aug. 2018), 860–870. doi:10.1109/TBCAS.2018.2834558. (cited on page 13)

Harris, C. and Stephens, M., 1988. A combined corner and edge detector. In *Proc. Fourth Alvey Vision Conf.*, vol. 15, 147–151. doi:10.5244/C.2.23. (cited on pages xviii, 12, 49, 50, and 51)

He, K.; Zhang, X.; Ren, S.; and Sun, J., 2016. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 770–778. doi:10.1109/cvpr.2016.90. (cited on page 57)

Higgins, W., 1975. A comparison of complementary and kalman filtering. *IEEE Trans. Aerospace and Electron. Syst.*, AES-11, 3 (May 1975), 321–325. doi:10.1109/taes.1975.308081. (cited on page 25)

Horn, B. K. and Schunck, B. G., 1981. Determining optical flow. *J. Artificial Intell.*, 17, 1 (1981), 185 – 203. doi:10.1016/0004-3702(81)90024-2. (cited on pages 14 and 17)

Huang, J.; Guo, M.; and Chen, S., 2017. A dynamic vision sensor with direct logarithmic output and full-frame picture-on-demand. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*. doi:10.1109/ISCAS.2017.8050546. (cited on page 42)

Kim, H.; Handa, A.; Benosman, R.; Ieng, S.-H.; and Davison, A. J., 2014. Simultaneous mosaicing and tracking with an event camera. In *British Mach. Vis. Conf. (BMVC)*. doi:10.5244/C.28.26. (cited on pages 14, 21, and 44)

Kim, H.; Leutenegger, S.; and Davison, A. J., 2016. Real-time 3D reconstruction and 6-DoF tracking with an event camera. In *Eur. Conf. Comput. Vis. (ECCV)*, 349–364. doi:\protect{10.1007/978-3-319-46466-4_21}. (cited on pages 14, 21, and 44)

Kimmel, R., 1999. Demosaicing: Image reconstruction from color CCD samples. *IEEE Trans. Image Process.*, (1999), 1221–1228. doi:10.1109/83.784434. (cited on page 29)

Kingma, D. P. and Ba, J. L., 2015. Adam: A method for stochastic optimization. *Int. Conf. Learn. Representations (ICLR)*, (2015). (cited on page 58)

Lai, W.; Huang, J.; Wang, O.; Shechtman, E.; Yumer, E.; and Yang, M., 2018. Learning blind video temporal consistency. In *Eur. Conf. Comput. Vis. (ECCV)*. (cited on pages 15 and 73)

Li, C.; Brandli, C.; Berner, R.; Liu, H.; Yang, M.; Liu, S.-C.; and Delbruck, T., 2015. Design of an RGBW color VGA rolling and global shutter dynamic and active-pixel vision sensor. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*. doi:10.1109/ISCAS.2015.7168734. (cited on page 18)

Lichtsteiner, P.; Posch, C.; and Delbruck, T., 2008. A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circuits*, 43, 2 (2008), 566–576. doi:10.1109/JSSC.2007.914337. (cited on pages 9, 10, 34, 41, 43, and 68)

Lin, T.; Maire, M.; Belongie, S. J.; Bourdev, L. D.; Girshick, R. B.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L., 2014a. Microsoft COCO: Common objects in context. In *Eur. Conf. Comput. Vis. (ECCV)*. (cited on page 71)

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L., 2014b. Microsoft COCO: Common objects in context. In *Eur. Conf. Comput. Vis. (ECCV)*, 740–755. doi:10.1007/978-3-319-10602-1_48. (cited on page 57)

Linares-Barranco, A.; Perez-Pena, F.; Moeys, D. P.; Gomez-Rodriguez, F.; Jimenez-Moreno, G.; Liu, S.-C.; and Delbruck, T., 2019. Low latency event-based filtering and feature extraction for dynamic vision sensors in real-time FPGA applications. *IEEE Access*, 7 (2019), 134926–134942. doi:10.1109/access.2019.2941282. (cited on page 12)

Liu, M. and Delbruck, T., 2017. Block-matching optical flow for dynamic vision sensors: Algorithm and FPGA implementation. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*. doi:10.1109/ISCAS.2017.8050295. (cited on page 17)

Liu, M. and Delbruck, T., 2018. Adaptive time-slice block-matching optical flow algorithm for dynamic vision sensors. In *British Mach. Vis. Conf. (BMVC)*. (cited on page 17)

Lowe, D. G., 2004. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.*, 60, 2 (Nov. 2004), 91–110. doi:10.1023/B:VISI.0000029664.99615.94. (cited on page 12)

Lucas, B. D. and Kanade, T., 1981. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. Artificial Intell. (IJCAI)*, 674–679. (cited on page 17)

Mahony, R.; Hamel, T.; and Pflimlin, J.-M., 2008. Nonlinear complementary filters on the special orthogonal group. *IEEE Trans. Autom. Control*, 53, 5 (Jun. 2008), 1203–1218. doi:10.1109/tac.2008.923738. (cited on pages 25 and 26)

Maqueda, A. I.; Loquercio, A.; Gallego, G.; García, N.; and Scaramuzza, D., 2018. Event-based vision meets deep learning on steering prediction for self-driving cars. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*, 5419–5427. doi:10.1109/CVPR.2018.00568. (cited on page 55)

Marcireau, A.; Ieng, S.-H.; Simon-Chane, C.; and Benosman, R. B., 2018. Event-based color segmentation with a high dynamic range sensor. *Front. Neurosci.*, 12 (2018). doi:10.3389/fnins.2018.00135. (cited on page 18)

Menze, M. and Geiger, A., 2015. Object scene flow for autonomous vehicles. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. doi:10.1109/cvpr.2015.7298925. (cited on page 17)

Merolla, P. A.; Arthur, J. V.; Alvarez-Icaza, R.; Cassidy, A. S.; Sawada, J.; Akopyan, F.; Jackson, B. L.; Imam, N.; Guo, C.; Nakamura, Y.; et al., 2014. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345, 6197 (2014), 668–673. (cited on page 53)

Messikommer, N.; Gehrig, D.; Loquercio, A.; and Scaramuzza, D., 2020. Event-based asynchronous sparse convolutional networks. *arXiv e-prints*, (Mar. 2020). (cited on pages 13 and 53)

Moeys, D. P.; Corradi, F.; Li, C.; Bamford, S. A.; Longinotti, L.; Voigt, F. F.; Berry, S.; Taverni, G.; Helmchen, F.; and Delbruck, T., 2018. A sensitive dynamic and active pixel vision sensor for color or neural imaging applications. *IEEE Trans. Biomed. Circuits Syst.*, 12, 1 (Feb. 2018), 123–136. doi:10.1109/TBCAS.2017.2759783. (cited on pages 10 and 18)

Moeys, D. P.; Li, C.; Martel, J. N. P.; Bamford, S.; Longinotti, L.; Motsnyi, V.; Bello, D. S. S.; and Delbruck, T., 2017. Color temporal contrast sensitivity in dynamic vision sensors. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*, 1–4. doi:10.1109/ISCAS.2017.8050412. (cited on pages xvi, 18, and 19)

Mostafavi I., S.; Wang, L.; Ho, Y.-S.; and Yoon, K.-J., 2019. 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)*. (cited on pages xviii, 15, 61, and 62)

Mueggler, E.; Bartolozzi, C.; and Scaramuzza, D., 2017a. Fast event-based corner detection. In *British Mach. Vis. Conf. (BMVC)*. (cited on pages xviii, 51, and 52)

Mueggler, E.; Gallego, G.; and Scaramuzza, D., 2015. Continuous-time trajectory estimation for event-based vision sensors. In *Robotics: Science and Systems (RSS)*. doi:10.15607/RSS.2015.XI.036. (cited on page 42)

Mueggler, E.; Huber, B.; and Scaramuzza, D., 2014. Event-based, 6-DOF pose tracking for high-speed maneuvers. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*, 2761–2768. doi:10.1109/IROS.2014.6942940. (cited on page 42)

Mueggler, E.; Rebecq, H.; Gallego, G.; Delbruck, T.; and Scaramuzza, D., 2017b. The event-camera dataset and simulator: Event-based data for pose estimation, visual odometry, and SLAM. *Int. J. Robot. Research*, 36, 2 (2017), 142–149. doi:10.1177/0278364917691115. (cited on pages xviii, xix, xxi, xxii, 15, 18, 30, 43, 47, 51, 58, 59, 60, 61, 68, 69, 74, 75, 76, 89, and 90)

Nageswaran, J.; Dutt, N.; Wang, Y.; and Delbrueck, T., 2009. Computing spike-based convolutions on GPUs. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*. doi:10.1109/iscas.2009.5118157. (cited on page 12)

Nair, V. and Hinton, G. E., 2010. Rectified linear units improve restricted boltzmann machines. In *Proc. Int. Conf. Mach. Learning (ICML)*, 807–814. https://icml.cc/Conferences/2010/papers/432.pdf. (cited on page 57)

Negri, P.; Soto, M.; Linares-Barranco, B.; and Serrano-Gotarredona, T., 2018. Scene context classification with event-driven spiking deep neural networks. In *IEEE Int. Conf. Electron. Circuits Syst. (ICECS)*. doi:10.1109/icecs.2018.8617982. (cited on page 13)

Nguyen, A.; Do, T.; Caldwell, D. G.; and Tsagarakis, N. G., 2019. Real-time 6DOF pose relocalization for event cameras with stacked spatial LSTM networks. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. (cited on page 55)

OpenCV, 2015. Opencv color conversions. https://docs.opencv.org/3.1.0/de/d25/imgproc_color_conversions.html. Accessed: 2020-05-17. (cited on page 23)

Orchard, G.; Benosman, R.; Etienne-Cummings, R.; and Thakor, N. V., 2013. A spiking neural network architecture for visual motion estimation. In *IEEE Biomed. Circuits Syst. Conf. (BioCAS)*, 298–301. doi:10.1109/biocas.2013.6679698. (cited on page 13)

Orchard, G.; Meyer, C.; Etienne-Cummings, R.; Posch, C.; Thakor, N.; and Benosman, R., 2015. HFirst: A temporal approach to object recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 37, 10 (2015), 2028–2040. doi:10.1109/TPAMI.2015.2392947. (cited on page 13)

Osswald, M.; Ieng, S.-H.; Benosman, R.; and Indiveri, G., 2017. A spiking neural network model of 3D perception for event-based neuromorphic stereo vision systems. *Scientific Reports*, 7, 1 (Jan. 2017). doi:10.1038/srep40703. (cited on page 13)

Pan, L.; Scheerlinck, C.; Yu, X.; Hartley, R.; Liu, M.; and Dai, Y., 2019. Bringing a blurry frame alive at high frame-rate with an event camera. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. (cited on page 14)

Paredes-Valles, F.; Scheper, K. Y. W.; and de Croon, G. C. H. E., 2019. Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2019). doi:10.1109/TPAMI.2019.2903179. (cited on page 13)

Perez-Carrasco, J. A.; Zhao, B.; Serrano, C.; Acha, B.; Serrano-Gotarredona, T.; Chen, S.; and Linares-Barranco, B., 2013. Mapping from frame-driven to frame-free event-driven vision systems by low-rate rate coding and coincidence processing–application to feedforward ConvNets. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35, 11 (Nov. 2013), 2706–2719. doi:10.1109/tpami.2013.71. (cited on pages 12 and 13)

Posch, C.; Matolin, D.; and Wohlgenannt, R., 2011. A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE J. Solid-State Circuits*, 46, 1 (Jan. 2011), 259–275. doi:10.1109/JSSC.2010.2085952. (cited on page 42)

Posch, C.; Serrano-Gotarredona, T.; Linares-Barranco, B.; and Delbruck, T., 2014. Retinomorphic event-based vision sensors: Bioinspired cameras with spiking output. *Proc. IEEE*, 102, 10 (Oct. 2014), 1470–1484. doi:10.1109/jproc.2014.2346153. (cited on pages xv and 8)

Rebecq, H.; Gehrig, D.; and Scaramuzza, D., 2018. ESIM: an open event camera simulator. In *Conf. on Robot. Learning (CoRL)*. (cited on pages 57, 70, and 71)

Rebecq, H.; Horstschäfer, T.; Gallego, G.; and Scaramuzza, D., 2017. EVO: A geometric approach to event-based 6-DOF parallel tracking and mapping in real-time. *IEEE Robot. Autom. Lett.*, 2, 2 (2017), 593–600. doi:10.1109/LRA.2016.2645143. (cited on pages 14, 21, and 42)

Rebecq, H.; Ranftl, R.; Koltun, V.; and Scaramuzza, D., 2019. Events-to-video: Bringing modern computer vision to event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. (cited on pages xv, xvii, xviii, xix, xxi, 11, 15, 35, 36, 55, 56, 57, 58, 59, 60, 64, 65, 66, 68, 70, and 76)

Rebecq, H.; Ranftl, R.; Koltun, V.; and Scaramuzza, D., 2020a. High speed and high dynamic range video with an event camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2020). (cited on pages xv, xviii, xix, xxi, 15, 34, 57, 58, 59, 60, 62, 63, 64, 65, 66, and 68)

Rebecq, H.; Ranftl, R.; Koltun, V.; and Scaramuzza, D., 2020b. High speed and high dynamic range video with an event camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, (2020). (cited on pages xxii, 15, 21, 30, 39, 57, 62, 69, 70, 72, 73, 74, 75, 76, 89, 90, and 91)

Reinbacher, C.; Graber, G.; and Pock, T., 2016. Real-time intensity-image reconstruction for event cameras using manifold regularisation. In *British Mach. Vis. Conf. (BMVC)*. doi:10.5244/C.30.9. (cited on pages xv, xvi, xvii, xviii, 12, 13, 21, 22, 29, 30, 33, 35, 36, 37, 38, 44, 45, 59, and 60)

Ronneberger, O.; Fischer, P.; and Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. (cited on page 15)

Rosinol Vidal, A.; Rebecq, H.; Horstschaefer, T.; and Scaramuzza, D., 2018. Ultimate SLAM? combining events, images, and IMU for robust visual SLAM in HDR and high speed scenarios. *IEEE Robot. Autom. Lett.*, 3, 2 (Apr. 2018), 994–1001. doi:10.1109/LRA.2018.2793357. (cited on page 42)

Rozell, C. J.; Johnson, D. H.; Baraniuk, R. G.; and Olshausen, B. A., 2008. Sparse coding via thresholding and local competition in neural circuits. *Neural Computation*, 20, 10 (Oct. 2008), 2526–2563. doi:10.1162/neco.2008.03-07-486. (cited on page 15)

RPG publications, 2019. http://rpg.ifi.uzh.ch/publications.html. (cited on pages xv and 1)

Rueckauer, B. and Delbruck, T., 2016. Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. *Front. Neurosci.*, 10, 176 (2016). doi:10.3389/fnins.2016.00176. (cited on pages 16 and 18)

Scheerlinck, C.; Barnes, N.; and Mahony, R., 2018. Continuous-time intensity estimation using event cameras. In *Asian Conf. Comput. Vis. (ACCV)*. (cited on pages xviii, 12, 21, 23, 30, 39, 47, 49, 59, 60, and 62)

Scheerlinck, C.; Barnes, N.; and Mahony, R., 2019a. Asynchronous spatial image convolutions for event cameras. *IEEE Robot. Autom. Lett.*, 4, 2 (Apr. 2019), 816–822. doi:10.1109/lra.2019.2893427. (cited on pages 13 and 41)

Scheerlinck, C.; Rebecq, H.; Gehrig, D.; Barnes, N.; Mahony, R.; and Scaramuzza, D., 2020. Fast image reconstruction with an event camera. In *IEEE Winter Conf. Appl. Comput. Vis. (WACV)*. (cited on pages 15, 55, 69, and 76)

Scheerlinck, C.; Rebecq, H.; Stoffregen, T.; Barnes, N.; Mahony, R.; and Scaramuzza, D., 2019b. CED: color event camera dataset. In *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. (cited on pages xvi, xvii, xviii, xix, xxi, 21, 23, 29, 30, 34, 35, 62, 63, 68, 75, and 83)

Serrano-Gotarredona, R.; Oster, M.; Lichtsteiner, P.; Linares-Barranco, A.; Paz-Vicente, R.; Gomez-Rodriguez, F.; Camunas-Mesa, L.; Berner, R.; Rivas-Perez, M.; Delbruck, T.; Liu, S.-C.; Douglas, R.; Hafliger, P.; Jimenez-Moreno, G.; Ballcels, A. C.; Serrano-Gotarredona, T.; Acosta-Jimenez, A. J.; and Linares-Barranco, B., 2009. CAVIAR: A 45k neuron, 5M synapse, 12G connect-s/s AER hardware sensory-processing-learning-actuating system for high-speed visual object recognition and tracking. *IEEE Trans. Neural Netw.*, 20, 9 (2009), 1417–1438. doi:10.1109/TNN.2009.2023653. (cited on pages 12 and 53)

Serrano-Gotarredona, R.; Serrano-Gotarredona, T.; Acosta-Jimenez, A.; and Linares-Barranco, B., 2006. A neuromorphic cortical-layer microchip for spike-based event processing vision systems. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 53, 12 (Dec. 2006), 2548–2566. doi:10.1109/tcsi.2006.883843. (cited on page 53)

Shedligeri, P. A.; Shah, K.; Kumar, D.; and Mitra, K., 2018. Photorealistic image reconstruction from hybrid intensity and event based sensor. *arXiv e-prints*, (2018). http://arxiv.org/abs/1805.06140. (cited on page 14)

Simonyan, K. and Zisserman, A., 2015. Very deep convolutional networks for large-scale image recognition. In *Int. Conf. Learn. Representations (ICLR)*. (cited on page 57)

Son, B.; Suh, Y.; Kim, S.; Jung, H.; Kim, J.-S.; Shin, C.; Park, K.; Lee, K.; Park, J.; Woo, J.; Roh, Y.; Lee, H.; Wang, Y.; Ovsiannikov, I.; and Ryu, H., 2017. A 640x480 dynamic vision sensor with a 9um pixel and 300Meps address-event representation. In *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*. doi:10.1109/ISSCC.2017.7870263. (cited on page 62)

Stein, R. B., 1965. A theoretical analysis of neuronal variability. *Biophysical Journal*, 5, 2 (Mar. 1965), 173–194. doi:10.1016/s0006-3495(65)86709-1. (cited on page 13)

Stoffregen, T.; Gallego, G.; Drummond, T.; Kleeman, L.; and Scaramuzza, D., 2019. Event-based motion segmentation by motion compensation. In *Int. Conf. Comput. Vis. (ICCV)*. (cited on page 17)

Stoffregen, T. and Kleeman, L., 2017. Simultaneous optical flow and segmentation (SOFAS) using Dynamic Vision Sensor. In *Australasian Conf. Robot. Autom. (ACRA)*. (cited on page 17)

Stoffregen, T. and Kleeman, L., 2019. Event cameras, contrast maximization and reward functions: an analysis. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. (cited on pages 17 and 84)

Stoffregen, T.; Scheerlinck, C.; Scaramuzza, D.; Drummond, T.; Barnes, N.; Kleeman, L.; and Mahony, R., 2020. Reducing the Sim-to-Real gap for event cameras. In *Eur. Conf. Comput. Vis. (ECCV)*. https://www.cedricscheerlinck.com/20ecnn. (cited on pages xix, 15, 18, 67, 68, 69, 75, and 76)

Suh, Y.; Choi, S.; Ito, M.; Kim, J.; Lee, Y.; Seo, J.; Jung, H.; Yeo, D.-H.; Namgung, S.; Bong, J.; Yoo, S.; Shin, S.-H.; Kwon, D.; Kang, P.; Kim, S.; Na, H.; Hwang, K.; Shin, C.; Kim, J.-S.; Park, P. K. J.; Kim, J.; Ryu, H.; and Park, Y., 2020. A 1280×960 dynamic vision sensor with a 4.95-μm pixel pitch and motion artifact minimization. In *IEEE Int. Symp. Circuits Syst. (ISCAS)*. doi:10.1109/iscas45731.2020.9180436. (cited on page 10)

Taverni, G.; Moeys, D. P.; Li, C.; Cavaco, C.; Motsnyi, V.; Bello, D. S. S.; and Delbruck, T., 2018. Front and back illuminated Dynamic and Active Pixel Vision Sensors comparison. *IEEE Trans. Circuits Syst. II*, 65, 5 (2018), 677–681. doi:10.1109/TCSII.2018.2824899. (cited on pages 18 and 59)

Vasco, V.; Glover, A.; and Bartolozzi, C., 2016. Fast event-based Harris corner detection exploiting the advantages of event-driven cameras. In *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. doi:10.1109/IROS.2016.7759610. (cited on pages xviii, 51, and 52)

Wang, Z.; Bovik, A. C.; Sheikh, H. R.; and Simoncelli, E. P., 2004. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.*, 13, 4 (Apr. 2004), 600–612. doi:10.1109/tip.2003.819861. (cited on pages xvii, xxii, 37, 38, 61, 73, 76, and 89)

Wang, Z.; Ng, Y.; van Goor, P.; and Mahony, R., 2019. Event camera calibration of per-pixel biased contrast threshold. In *Australasian Conf. Robot. Autom. (ACRA)*. (cited on pages 12 and 70)

Watkins, Y.; Thresher, A.; Mascarenas, D.; and Kenyon, G. T., 2018. Sparse coding enables the reconstruction of high-fidelity images and video from retinal spike trains. In *Proceedings of the International Conference on Neuromorphic Systems*. doi:10.1145/3229884.3229892. (cited on page 15)

Xiao, R.; Tang, H.; Ma, Y.; Yan, R.; and Orchard, G., 2019. An event-driven categorization model for AER image sensors using multispike encoding and learning. *IEEE Trans. Neural Netw. Learn. Syst.*, (2019), 1–9. doi:10.1109/tnnls.2019.2945630. (cited on page 13)

Ye, C.; Mitrokhin, A.; Parameshwara, C.; Fermüller, C.; Yorke, J. A.; and Aloimonos, Y., 2019. Unsupervised learning of dense optical flow and depth from sparse event data. *arXiv e-prints*, (2019). http://arxiv.org/abs/1809.08625. (cited on pages 17, 84, and 85)

Zhang, L. and Rusinkiewicz, S., 2018. Learning to detect features in texture images. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. (cited on page 76)

Zhang, L.; Zhang, L.; Mou, X.; and Zhang, D., 2011. FSIM: A feature similarity index for image quality assessment. *IEEE Trans. Image Process.*, 20, 8 (Aug. 2011), 2378–2386. doi:10.1109/tip.2011.2109730. (cited on pages xvii, 37, and 38)

Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O., 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. (cited on pages xxii, 15, 58, 61, 72, 73, 89, and 90)

Zhao, B.; Ding, R.; Chen, S.; Linares-Barranco, B.; and Tang, H., 2015. Feedforward categorization on AER motion events using cortex-like features in a spiking neural network. *IEEE Trans. Neural Netw. Learn. Syst.*, 26, 9 (Sep. 2015), 1963–1978. doi:10.1109/tnnls.2014.2362542. (cited on page 13)

Zhou, B.; Krähenbühl, P.; and Koltun, V., 2019. Does computer vision matter for action? *Science Robotics*, 4, 30 (May 2019), 1–12. doi:10.1126/scirobotics.aaw6661. (cited on page 16)

Zhu, A. Z.; Thakur, D.; Ozaslan, T.; Pfrommer, B.; Kumar, V.; and Daniilidis, K., 2018a. The multivehicle stereo event camera dataset: An event camera dataset for 3D perception. *IEEE Robot. Autom. Lett.*, 3, 3 (Jul. 2018), 2032–2039. doi:10.1109/lra.2018.2800793. (cited on pages xix, xxii, 15, 18, 68, 69, 74, 75, 76, 84, 85, 89, and 90)

Zhu, A. Z.; Yuan, L.; Chaney, K.; and Daniilidis, K., 2018b. EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. In *Robotics: Science and Systems (RSS)*. doi:10.15607/RSS.2018.XIV.062. (cited on pages 17, 18, 55, 68, 69, 72, 74, 76, 84, and 85)

Zhu, A. Z.; Yuan, L.; Chaney, K.; and Daniilidis, K., 2019. Unsupervised event-based learning of optical flow, depth, and egomotion. In *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. (cited on pages xv, 15, 17, 18, 55, 56, 69, 72, 74, and 76)