MONASH University

DOCTORAL THESIS

Motion Estimation by Focus Optimisation: Optic Flow and Motion Segmentation with Event Cameras

Author:
Timo STOFFREGEN

Supervisor:
A/Prof. Lindsay KLEEMAN
Co-Supervisor:
Prof. Tom DRUMMOND

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy

in the

Australian Centre for Robotic Vision
Department of Electrical and Computer Systems Engineering

May 4, 2021

Declaration of Authorship

I, Timo STOFFREGEN, declare that this thesis titled, "Motion Estimation by Focus Optimisation: Optic Flow and Motion Segmentation with Event Cameras" and the work presented in it are my own. This thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Student signature:

Date: 4 December 2020

Copyright Notice

Notice 1

Notice 2

I certify that I have made all reasonable efforts to secure copyright permissions for third-party content included in this thesis and have not knowingly added copyright content to my work without the owner's permission.

Abstract

Event cameras are a recent revolution in machine visual perception. As opposed to conventional camera technology, event camera pixels asynchronously report small intensity changes with a high temporal precision on the order of µs. The benefits of this bio-inspired visual paradigm include high dynamic range ($\approx 120\,\mathrm{dB}$), low power usage ($\approx 5\,\mathrm{mW}$), low latency of the order of microseconds and a vastly decreased propensity for motion blur, due to a per-pixel sampling rate that adapts to the rate of change of brightness in the scene. The data produced by these sensors is however, novel to the computer vision community - best visualised in a three-dimensional spatiotemporal volume, events are more similar to point clouds than image frames. As a result, new algorithms and processing techniques are required to take advantage of the benefits of event cameras.

Since event cameras report on those parts of the scene which are in motion, they are a natural choice for tasks that involve producing a description of that motion, such as optic flow estimation, ego-motion estimation, or motion segmentation. This work develops a new Focus Optimisation (FO) framework for solving optic flow estimation, object tracking, and motion segmentation problems, while respecting the unique properties of event-based data. The framework relies on the data association that exists between events triggered by a common point feature moving across the image plane. If the trajectory of the point feature is known, these related events can be transported along the point trajectories to a reference time to form a motion-compensated image of the events. Since the focus of that image is maximised when these trajectories are correct, the motion estimates can be optimised with regard to the measured focus.

In the first chapter, I propose applying FO to a set of events using an optic flow motion model. Since the dominant motion is captured by the optimisation, it is possible to segment the events explained by that motion by selecting those regions of the image which are in focus. Continuing in a greedy manner, the events are segmented by optic flow motion. The resulting segmentation masks and motion models are used to initialise novel event-based trackers, whose motion estimates are updated on a per-event basis using a technique similar to particle filtering.

In the second chapter, I improve on this idea by applying a probabilistic approach to the motion segmentation. In this method, each event has a likelihood of belonging to a particular motion model, which is not limited to optic flow models. The parameters of the motion models are then optimised together, with respect to a global focus measurement. Then the likelihoods of the events are updated to account for the new motion estimates. By applying these steps iteratively, a solution for the motion segmentation and parametrisation of the events is converged on, similar to Expectation Maximisation (EM). This method has the advantage that it is not dependent on hyper-parameters and is able to accept any number and mixture of motion models.

Chapter three examines the choice of focus measure and proposes a simple classification for various measures. In this chapter, we mathematically and experimentally show that contrast measures that reward density have advantages in noise resilience, while those that reward sparsity have advantages in overcoming aperture uncertainty, a cousin of the aperture problem in conventional vision.

Chapter four shows how state-of-the-art results for optic flow can be achieved by training a Convolutional Neural Network (CNN) supervised by simulated events with ground truth optic flow. By using L1 distance as the loss, the network learns to predict fully dense optic flow, a first in event-based vision. I also propose a new measure for evaluating event optic flow which does not require ground truth data, yet is tailored to event-based data.

Finally I conclude the thesis by outlining opportunities for future research based on the previously described works.

Acknowledgements

My deepest thanks to my advisers Lindsay Kleeman and Tom Drummond for their invaluable help in performing the research and advice during the writing of this thesis.

I am also indebted to Guillermo Gallego and Davide Scaramuzza of the Robotics and Perception Group (RPG) for kindly hosting me for a semester at the University of Zurich/ETH Zurich. I benefited hugely from their enormous well of expertise in event based vision as well as feeling very welcome in their lab.

Particular thanks to my co-authors Cedric Scheerlinck, Henri Rebecq, and Guillermo Gallego without whose valuable insights and hard work my publications would have been much poorer. Also a great thanks to the professors Lindsay Kleeman, Robert Mahoney, Tom Drummond, Nick Barnes, and Davide Scaramuzza who lent their vast knowledge and experience to these projects.

A shout out to my panel, Bill Corcoran, James Saunderson, and Mehmet Yuce who guided me at intervals along the way.

Finally, many thanks to my thesis examiners for making the effort to review this work.

This research was supported by an Australian Government Research Training Program (RTP) Scholarship and the Australian Centre for Robotic Vision.

Contents

List of Figures

List of Tables

Glossary

Discretised Event Volume/Voxel Grid The Discretised Event Volume (DEV) or voxel grid (Figure 1.5b), is a popular event representation for deep learning applications, in which the events are placed into discretised temporal bins. For additional detail see Appendix A.1. 7, 9, 13, 117, 118

Event Image The event image I_0 is a common event representation, formed by adding the events in a time window at each pixel (Figure 1.5c). For additional detail see Appendix A.1. 7–9, 12, 13, 22, 36, 118, 130

Focus Optimisation Focus optimisation is an event processing framework in which the measured focus of events warped to an Image of Warped Events (IWE) is optimised w.r.t. the parameters of the warping operation (Figure 1.7). For additional detail see Appendix A.2. vii, xi, xvi, xvii, 3, 6, 11–16, 20, 22, 23, 33, 35, 36, 38, 44, 50, 59, 61–64, 66, 68, 70, 71, 81, 113–116, 118, 121, 129

Hough transform The Hough transform is a method detecting simple shapes (usually lines or circles) by per-pixel voting in a parameter space from which object candidates are obtained as local maxima. 12, 36

Image of Warped Events The Image of Warped Events (IWE), denoted I_ω, is the image that is obtained by transporting a set of events $\mathcal{E}$ to a common reference time t_{ref} through some warping operation $\mathcal{W}$. The warped events are then formed into an event image, using bilinear interpolation (Figure 1.7). For additional detail see A.1. xv, 12–14, 19, 20, 22–24, 26, 27, 38–41, 45, 48, 51–55, 61–65, 67, 99, 100, 102–104, 113–115, 118, 119, 122

K-means clustering K means is a clustering method in which n data are partitioned into K (determined *a priori*) clusters, where each data is assigned to the cluster with the nearest mean. xv, 12, 40, 58

L1 Distance The L1 distance or norm, is the distance between two vectors. For vectors $\vec{p}$ and $\vec{q}$ it is given as $\sum_{i=1}^{n} |p_i - q_i|$. vii, 90, 99

Lukas-Kanade A classic optic flow algorithm which uses assumptions of brightness and local flow constancy to solve a least squares formulation with optic flow as the solution. 7, 8, 15, 28

Optic Flow Optic flow is a vector field over the pixels in a scene that describes the motion of each pixel, caused by relative motion between the camera and the scene. 7

Surface of Active Events The Surface of Active Events (SAE) (also commonly known as time image) is a common event representation, in which the latest timestamp is recorded at each pixel location (Figure 1.5d). For additional detail see Appendix A.1. 5, 7–9, 11, 86, 130

Acronyms

AEE Average Endpoint Error. xvii, 99, 100

AER Address Event Representation. 4, 5

AGV Autonomous Ground Vehicle. 14

ANN Artificial Neural Network. 10, 11, 13

API Application Programming Interface. 4

APS Active Pixel Sensor. 3–5, 10, 12, 85–88, 125

BnB Branch and Bound. 13, 14, 115

CED Color Event Dataset. xvii, 84, 99

CNN Convolutional Neural Network. vii, xvii, 6, 9, 11, 16, 17, 83–86, 113, 114, 116

COCO Common Objects in COntext. 89

CPU Central Processing Unit. xvii, 33, 45, 46, 115

CT Contrast Threshold. xiii, xvii, 1, 2, 5, 17, 19, 84, 85, 87–89, 91, 104, 109, 110, 113, 114

DAVIS Dynamic and Active-pixel VIsion Sensor. xv, 4–6, 12, 17, 19, 27, 29, 42, 43, 51, 68, 72, 83, 85, 88–91, 123

DEV Discretised Event Volume. 117

DoF Degree of Freedom. 11, 13–15, 41–43, 49, 85, 115, 123

DS Direction Selective. 8

DVS Dynamic Vision Sensor. 4, 5, 29, 31–33, 48, 49, 66, 91

E2VID Events to Video. 86, 90, 91, 93, 94, 96, 97, 104, 109–111

EED Extreme Event Dataset. xv, 42, 43, 46

EFK Extended Kalman Filter. 85

EM Expectation Maximisation. vii, xii, 11–13, 15, 35, 40, 49, 52, 53, 114

ESIM Event Camera Simulator [85]. 87–89, 110

FO Focus Optimisation. xi, xiii–xv, 20, 22, 61, 62, 113, 114, 119, 121

FPGA Field Programmable Gate Array. 5, 45, 115

FWL Flow Warp Loss. 16, 85, 91, 93, 99, 110, 114

GAN Generative Adversarial Network. 13, 17

GCNN Graph Convolutional Neural Network. 11

GPU Graphics Processing Unit. xvii, 17, 45, 46, 115, 121, 125, 127, 128

GRU Gated Recurrent Unit. 84

HATS Histogram of Averaged Time Surfaces. 86

HDF5 Hierarchical Data Format v.5. 123, 125–127, 132

HDR High Dynamic Range. 15, 37, 42, 46, 59, 84

HQF High Quality Frames. xvi, xvii, 1, 83–85, 88, 91–93, 96, 102, 104, 105, 107, 110, 111, 114

HSV Hue-Saturation-Value colourspace. 102–104

ICP Iterative Closest Point. 12

IJRR Event Camera Dataset and Simulator. xvii, 84, 85, 87, 88, 90–93, 97, 103–105, 108–111

IWE Image of Warped Events. 12, 22, 61, 113, 118

jAER Java tools for Address-Event Representation neuromorphic processing. 27

LMS Layered Motion Segmentation. xv, xvii

LPIPS Learned Perceptual Image Patch Similarity. 90–93, 105, 109–111

LSTM Long Short-Term Memory. 84

MSE Mean Squared Error. 92, 93, 105, 110, 111

MVSEC Multi Vehicle Stereo Event Camera. xvii, 84, 86–88, 90–93, 98, 99, 104, 105, 109–111, 116

PCIe Peripheral Component Interconnect express. 115

PDF Probability Density Function. 29

PWM Pulse Width Modulation. 5

RANSAC RAndom SAmple Consensus. 9

ReLU Rectified Linear Unit. 83

RNN Recurrent Neural Network. 84

RoI Region of Interest. 121

ROS Robot Operating System. 91, 126

RPG Robotics and Perception Group, Zurich. 91, 125

SAE Surface of Active Events. 118, 119

SAR Synthetic Aperture Radar. 13, 16

SfM Structure from Motion. 11

SLAM Simultaneous Localisation and Mapping. 3, 6, 7

SNN Spiking Neural Network. 6, 116

SoC System on Chip. 115

SOFAS Simultaneous Optic Flow and Segmentation. xv, 19, 28, 29, 31–33, 43

SotA State of the Art. xvii, 3, 13, 16, 17, 27, 37, 43, 44, 83–86, 90, 93, 114, 115

SSIM Structural SIMilarity. 90, 92, 93, 105, 110, 111

UAV Unmanned Aerial Vehicle. 35

VGG Visual Geometry Group network. 90

VO Visual Odometry. 7

Notation

α Random variable. 89

B The number of bins when forming a DEV from events. 86, 89, 117

δ Kronecker delta function. 118

$\frac{dI_\omega}{d\omega}$ Derivative of image of warped events. 22

d_p Displacement (in pixels) used to calculate the lifetime of events, given a veolcity estimate. 26

Δt A slice/length/difference in time. 22, 26, 41, 42, 66, 89, 117

$\mathcal{E}$ A set of events. 20–22, 26, 27, 38, 40, 42, 49, 50, 52, 54, 99, 113, 117–119

e An Event. 5, 19, 21–23, 38–40, 50–54, 62, 66, 86, 89, 117–119

e' A warped event. 22, 39, 118

$\frac{\textbf{events}}{\textbf{pix}\cdot\textbf{s}}$ Events per pixel per second is a measure of how many events are generated by a given event camera and scene. xvi, 87–89, 109

$\mathcal{E}_\mathcal{W}$ A set of warped events. 22, 118

$\mathcal{G}$ Gaussian kernel. 26

γ Score of how well an event fits a tracker (tracker score). 21, 26

h Height. 74, 76–79

$\mathbf{h}$ Neighborhood of pixel. 21–23

I Image. 7, 20–23, 26, 27, 39, 40, 61–66, 68, 99, 118, 119, 121

I_0 Event image. 21–23, 27, 68, 99, 118

I_μ Mean-centered IWE ($I_\omega - \mu(I_\omega)$). 64

I_ω Image of warped events. 20, 22, 23, 26, 27, 39, 40, 61–66, 99, 118, 119, 121

I_t The Surface of Active Events (SAE). 119

L The log of the intensity (can be thought of as brightness). 5

ℓ A layer/cluster. 40, 54

$\lambda_\mathbf{c}$ Threshold to determine whether grid search optimisation has converged. 24

λ_c Shifting factor for r_{SoSA} reward function for FO. 65

$\lambda_{\mathbf{CT}}$ Event camera pixel contrast threshold. 2, 5, 86–88, 109

$\lambda_{\mathbf{CT}}^{-}$ Negative event camera contrast threshold. 87, 89

$\lambda_{\mathbf{CT}}^{+}$ Positive event camera contrast threshold. 87, 89

$\lambda_{\mathbf{err}}$ Threshold of expected events vs actual events, to determine whether a tracker is still valid. 27

$\lambda_{\mathbf{ev}}$ Threshold to determine if a pixel has enough events to count as a neighboring pixel when determining whether to perform focus optimisation. 22, 23, 33

λ_L The number of inputs given to the recurrent CNN at train time. 89, 90

$\lambda_{\mathbf{maxi}}$ Maximum iterations of grid search optimiser. 33

$\lambda_{\mathbf{pix}}$ Threshold to determine whether enough neighboring pixels have sufficient events for focus optimisation. 22, 23, 33

$\lambda_{\mathbf{sup}}$ Support threshold for r_{SoA}. 64, 65

m Mask. 22

μ Mean. 17, 39

n_a The actual number of events produced in an upcoming time interval. 27

N_e The number of events under consideration. 22, 33, 38–40, 42, 45, 51–55, 62, 65, 89, 115–119

$N_{\mathbf{it}}$ Number of iterations. 45, 54, 55

N_l The number of layers used in our motion segmentation algorithm (Section 3). xv, 36, 38–42, 45–47, 49–55, 59

$\mathcal{N}$ Normal distribution. 17, 87, 89, 90

N_p Number of pixels. 45, 54, 55, 63

n_p The number of events predicted to be generated in an upcoming time interval. 27

$\mathcal{O}$ Upper bound computational complexity (Bachmann-Landau notation). 54, 55

Ω The continuous image domain. 23, 27, 38, 39, 64, 65

ω Warp params. 20, 22, 23, 26, 27, 37–42, 50–54, 61–66, 99, 115, 118, 119, 121, 122

ω_v Angular velocity. 41, 42

$\mathbf{P}$ Matrix of event-cluster membership likelihoods. 37, 39, 40, 50, 52–54

P Probability function. 52–54, 90

$\boldsymbol{\pi}$ Cluster probabilities. 50, 52, 53

p Probability density function. 50–54

ϕ Angle of rotation around a point of the Surface of Active Events (SAE). 74–79

p_{kj} The probability of the k-th event belonging to the j-th cluster. 39, 40, 54

q_p State of nature: sequence is paused. 90

q_r State of nature: sequence is running. 90

r Focus measure. xi, 20, 22–24, 40, 62, 65, 115, 119

$\mathbb{R}$ Real numbers. 53, 118

$r_{\mathbf{MoA}}$ Maximum pixel value reward function for FO. 64, 66, 67, 70, 71

$r_{\mathbf{R1}}$ Combined reward for FO. 68, 78, 81, 122

$r_{\mathbf{R2}}$ Combined reward for FO. 68, 71, 78, 81, 122

r_{σ^2} Variance reward function for FO. 63, 64

$r_{\mathbf{SoA}}$ Sum of Accumulations reward for FO. xvi, 64–68, 70, 71, 73, 77, 79, 80

$r_{\mathbf{SoE}}$ Sum of Exponentials reward function for FO. 64, 66–68, 70, 71

$r_{\mathbf{SoS}}$ Sum of Squares reward function for FO. xvi, 64, 66–71, 73, 74, 76–81

$r_{\mathbf{SoSA}}$ Sum of Accumulations reward for FO with a suppression mechanism for large values. xvi, 65–68, 70, 71, 78

s Event polarity $s \in \{-1, +1\}$. 1, 5, 19, 22, 39, 62, 65, 89, 113, 117–119

σ Standard deviation. 17

σ^2 Variance. 39, 40, 70, 99

σ The size of the Guassian blur kernel. xiii, 78, 81

T The continuous time domain. 38, 42

$\mathcal{T}$ The set of current trackers. 21, 26, 27

t A tracker. 21, 26, 27

t_0 Timestamp of the first event e_0. 117

$\boldsymbol{\theta}$ Optimisation variables. 54

t_i^* Event timestamp normalised to the range $[0, B-1]$. 89, 117

t Event timestamp. 1, 2, 5, 19, 22, 39, 41, 49, 62, 86, 87, 89, 99, 113, 117–119, 126

t_N Timestamp of the Nth event e_N. 117

t_r Event camera pixel refractory period. 2, 87

$t_{\mathbf{ref}}$ Reference time. 12, 22, 23, 39, 41, 99, 118

ψ The state of nature of a random variable ([19, Ch.2]). 50–54

V A DEV of events discretised into B bins. 89, 117

V_e The volume containing the set of events $\mathcal{E}$. 38, 40, 50, 51, 53, 54

$\vec{v}$ Velocity vector. 7, 22–24, 26, 66, 68, 70, 99

v_x Optic flow x component. 7, 22, 41, 42, 66, 99

(v_x, v_y) Optic flow. 7, 20, 22, 41, 70, 99, 119

v_y Optic flow y component. 7, 22, 41, 42, 66, 99

$\mathcal{W}$ Warping function for the events. 22, 39, 41, 42, 51, 62, 66, 115, 118, 119

w Width. 74, 76–79

$\mathcal{W}_{\vec{v}}$ Optic flow warp function. 66

$\mathbf{x}$ Event position. 5, 19, 21–23, 26, 39, 40, 51, 62, 65, 66

x Event x position. 1, 5, 19, 22, 23, 41, 42, 62, 66, 86, 89, 99, 113, 117–119, 126

$\mathbf{x}'$ Warped event position. 26, 39, 40, 51, 53, 62, 65, 66

x_c Center of rotation x component. 41, 42

(x_c, y_c) Center of rotation. 41

y Event y position. 1, 5, 19, 22, 41, 42, 62, 66, 86, 89, 99, 113, 117–119, 126

y_c Center of rotation y component. 41, 42

Chapter 1

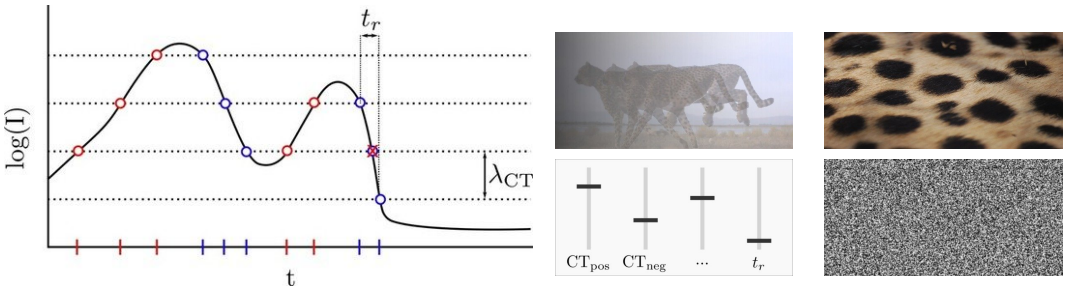
Introduction

1.1 Overview

Recently, event-based cameras have made a large impact on the computer vision community. Conventionally, visual information has been represented as a series of stills, typically collected at a fixed sampling rate (frame rate). In these sensors, light is projected through a lens system onto a grid of light-sensitive photoreceptors, which discretise the visual scene into pixels. The scene is then sampled periodically through the action of a mechanical or electronic shutter. Such systems have been the default visual sensor for robotic systems despite having several drawbacks over biological visual sensors. Since the sampling rate is fixed, the camera is not able to adapt to the rate of visual change in the scene. As a result, static scenes are oversampled, producing redundant data capture. Conversely, fast changing scenes are undersampled, causing motion blur or even leading the camera to miss important events entirely. Since pixels are exposed on a common shutter the total dynamic range of the camera is also limited, resulting in over- and underexposure of parts of the scene.

Event cameras solve several of these issues, by offering a biologically inspired, asynchronous visual sensing paradigm. In an event camera, each pixel keeps track of its current measured intensity. When the change in log intensity from the previous measurement exceeds the predefined CT, an event is triggered and transmitted off-camera (see Figure 1.1). Each event consists of a tuple of the event location $[x, y]$, polarity $s \in \{-1, +1\}$ (whether the pixel logged a positive or negative brightness change) and timestamp t (which is resolved to µs resolution in modern event cameras). Since each pixel is able to report events at any time, independently of other pixels, event cameras are able to take samples at a dynamic, scene-dependent rate, dependent only on the choice of CT and hold-off time (refractory period).

Since event data is four-dimensional (three-dimensional if the polarity information is discarded), it is natural to view it in a three-dimensional spatiotemporal plot, where each event is represented by its $[x, y], t$ position and coloured by its s polarity (see Figure 1.2). Viewed from this perspective, event data is much more similar to a point cloud than a series of frames. The generation of the events is directly dependent on four things: *i)* scene motion and lighting dynamics, *ii)* scene texture, *iii)* camera noise, and *iv)* camera parameters (see Figure 1.1b). Of these parameters, the most influential is the CT, where small variations can result in large changes in the event rate. Since event cameras only provide updates on changes in the scene they are an efficient means of encoding visual information. Conventional cameras undersample the scene when changes occur faster than the set framerate, resulting in missing occurrences or motion blur. Since event cameras sample the scene at an optimal rate with regard to per-pixel brightness changes, they often generate more data than conventional cameras, despite being a more efficient means of recording. Large event camera datasets regularly run into the hundreds of GB - the commonly cited IJRR event camera dataset, recorded with a comparatively low spatial resolution of 240×180 pixels and spanning just under 20 minutes is 13 GB (events and frames only). Of that, only 19 % is used up by the accompanying frames, taken with the same resolution at 23 fps. Other datasets are similar (see Table 1.1). Datasets such as the recent High Quality Frames (HQF) [105] and DDD20 [36], which are more compact than the frame-based representation contain comparatively slow moving scenes.



(a) (b)

Figure 1.1: (1.1a) Plot of the intensity recorded at an event camera pixel. Red and blue circles (shown as bars on the t axis) represent positive and negative events respectively, which are triggered when the measured intensity crosses the Contrast Threshold (CT). If the intensity crosses the CT (λ_{CT}) before the refractory period (t_r) has expired, no event is triggered. (1.1b) In real sensors, events are caused by motion and lighting variations, texture, camera parameters, and sensor noise (clockwise from top left).

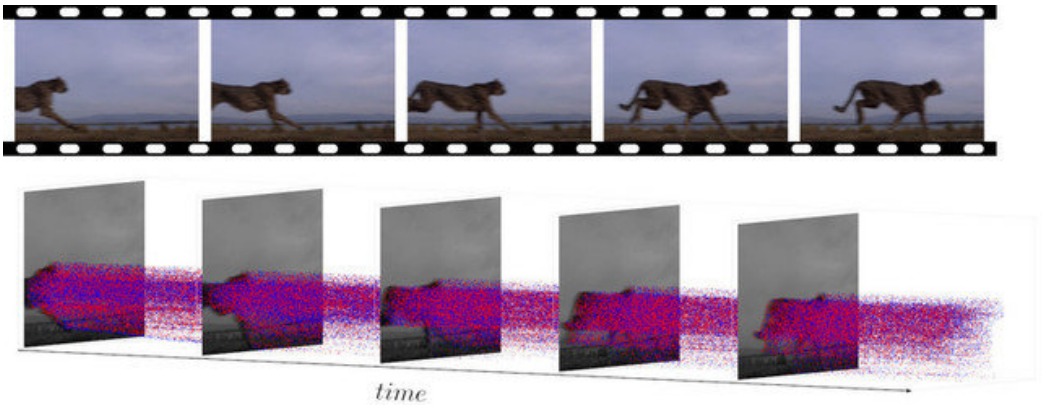


Figure 1.2: A sequence recorded with a standard camera (top) and with an event camera (bottom). The events, shown in a spatiotemporal view, are colored red for positive (increasing brightness) and blue for negative (decreasing brightness) events. Some key advantages of event cameras are immediately obvious; the event camera suffers minimal motion blur and gives a continuous representation of the visual data.

Conventional computer vision methods can find only limited application to asynchronous, spatiotemporal data; a new visual paradigm requires new algorithms. Currently the literature can be broken into two methodologies; batch processing and asynchronous processing of the events [26]. Since individual events do not carry much information, it only makes sense to consider them in aggregate. Batch processing does this directly, collecting a number of events in a temporal slice of the events and then processing them at once. Asynchronous processing accepts individual events as inputs and uses them to update the value of some state. In this case the state information contains the transformed information of the aggregated previous events. A core problem in both methodologies is to determine the relevance of historic events. In batch processing this boils down to questions about how wide the temporal slice should be, in asynchronous processing to how fast the effects of a given event should decay in the state memory. This is because data association (the fact that successive events are caused by motions of common point features in the scene), quickly makes old events irrelevant.

Table 1.1: Size in MB/s of various event camera datasets. The framerate (FR) and size of the accompanying frames is listed for comparison (frames are the same resolution as event sensor, grayscale, and uncompressed). Since various compression schemes could be applied to optimize, an assumption of 1 B per frame pixel and 7 B per event is made (based on [37]).

Dataset	Events [MB/s]	Frames [MB/s]	FR [fps]	Resolution
HQF [105]	1.74	2.03	22.6	240×180
IJRR [64]	8.88	2.08	23.1	240×180
MVSEC [118]	6.58	3.21	35.7	346×260
CED [96]	9.38	3.33	37.0	346×260
DDD20 [36]	1.81	2.58	20.5	346×260

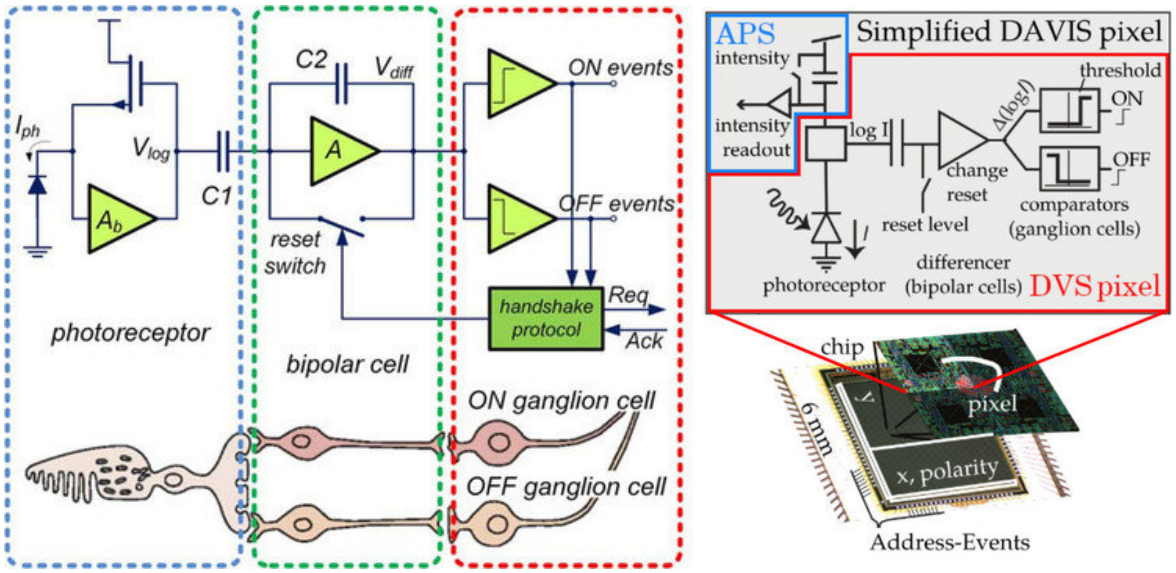
1.1.1 Aims of Thesis

This thesis aims to exploit the data association between events and leverage it to estimate optic flow, track objects and perform motion segmentation. Specifically, I aim to:

- Trace events along the trajectories they follow in the spatiotemporal volume as visual features move across the image plane to infer motion estimates (such as optic flow).
- Develop the theory of this tracing to produce optic flow estimates which are robust to noise and the aperture problem.
- Segment the seemingly tangled events generated by complex scenes into those generated by camera motion and individual moving objects.
- Track moving objects in an efficient per-event manner.
- Estimate State of the Art (SotA) pixel-dense optic flow from sparse events without the use of Active Pixel Sensor (APS) frames.
- Evaluate optic flow estimates without requiring ground-truth optic flow.

Event cameras lend themselves naturally to tasks which describe motion or objects in motion, such as optic flow estimation, tracking, or motion segmentation. Such descriptions for motions are important tools in computer vision and robotics as they are integral to tasks such as tracking, Simultaneous Localisation and Mapping (SLAM), visual odometry, or depth estimation. This work presents a focus optimisation technique for processing events which explicitly acknowledges the data association between events and makes use of the fine-grained temporal information of the events. The Focus Optimisation (FO) framework operates by transporting events to a reference time along spatiotemporal trajectories, created by the motions of brightness gradients across the image plane. The result is an image of the events, whose sharpness, or focus, is maximised when the estimated trajectories are equal to the actual trajectories. Through parametrisation of candidate motions, the focus can be optimised to reveal estimates of desired properties such as optic flow [104, 120], ego-motion [27, 75], depth [26], or more generic 3D motions [102].

My work explicitly acknowledges the data association of the events and leverages it to estimate optic flow, track objects, and perform motion segmentation. I develop a technique of optimising the focus of motion-compensated events and explore some of the resulting theory to improve results. In the final chapter, I estimate optic flow from a deep neural network, using simulated events with ground truth optic flow as a training signal. While it is usually challenging to predict pixel-dense optic flow from naturally sparse events, this network is able to produce dense optic flow estimates, a step up from current SotA.



(a) [83] (b) [24]

Figure 1.3: 1.3a The hardware of Dynamic and Active-pixel VIsion Sensor (DAVIS) pixels are directly bio-inspired by the structure of retinal cells. 1.3b A simplified circuit diagram of a DAVIS pixel, which contains both an Active Pixel Sensor (APS), which is able to record conventional frames and a Dynamic Vision Sensor (DVS), which reports events. The APS and DVS share a single photodiode and are thus perfectly registered to each other. For an in-depth comparison of event camera pixel design, see [83].

1.2 Event Camera Operating Principle

Motivation

Event cameras were first conceived of in the early 1990s, with pioneering work on the Silicon Retina and the Address Event Representation (AER) by Misha Mahowald and Carver Mead [54]. The motivation was to build a brain in silicon [24] and to mimic the spiking, asynchronous nature of biological vision. This first event camera was modelled after the three layer Kufler retina, summarised in Figure 1.3a. Biological retinas have many advantages over conventional cameras, such as high efficiency and low latency and it soon became apparent that these advantages could become available to artificial retinas. These bio-inspired origins are the reason why event cameras are also often referred to as neuromorphic cameras in the literature. At its core, an event camera relies on a smart pixel to generate events, which are able to keep track of their measured intensity, fire events when appropriate and reset, adjusting dynamically to changes in luminance in the scene. This dynamism gives rise to the other common nomenclature *Dynamic Vision Sensor*.

1.2.1 Dynamic and Active-pixel VIsion Sensor (DAVIS)

The first practical and robust event camera became commercially available in 2008, with the development of the Dynamic Vision Sensor (DVS) [45]. This seminal work resulted in the development of the DAVIS 240 and later the DAVIS 346 event cameras, with a 240×180 and 346×240 spatial sensor resolution respectively. A useful feature of these sensors was that they contained not only a DVS, but also an APS on the same chip. This allowed recording both events *and* conventional frames, perfectly registered and synchronised with each other. This feature in combination with easy to use Application Programming Interfaces (APIs) resulted in widespread adoption of the DAVIS cameras in the research community. As a result, they are the most commonly cited event camera hardware in

the literature, despite competition from large industry efforts led by manufacturers such as Prophesee or Samsung [100].

Figure 1.3 summarised the operating principle of the DAVIS sensor. The APS sensor of each pixel, while sharing a photodiode with the DVS sensor, operates on a global shutter independently of the DVS. Each DVS pixel contains a capacitor, which integrates changes in the brightness measured at the photodiode (via the proxy value of log intensity). The value of the capacitor is amplified and passed on to a pair of comparators, which trigger when the brightness passes either the positive or negative preset CTs. The CTs are a key parameter in the DVS as they determine the sensitivity to brightness changes of the sensor (see Figure 1.1a). If the brightness change is measured positive, the pixel reports a $(+)$ event, if negative a $(-)$ event. Subsequently the capacitor is drained, resetting the pixel. A programmable refractory period is then imposed on the pixel, preventing it from generating new events for a predetermined, short period of time. An event e, written as a tuple

$$e = \{\mathrm{x}, t, s\} \tag{1.1}$$

(where $\mathrm{x} = [x, y]$ is the position of the event on the image plane, t is the timestamp of the event, and s is the polarity of the event) is thus triggered whenever

$$\Delta L(\mathrm{x}, t) = s\lambda_{\mathrm{CT}} \tag{1.2}$$

where $L = \log$ intensity and λ_{CT} is the parametrisation of the CT.

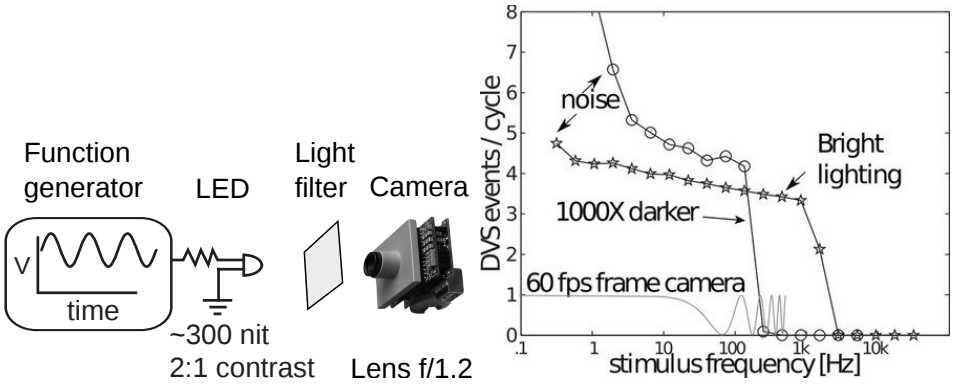
Events thus generated are sent to an arbiter for readout to a Field Programmable Gate Array (FPGA), where the events timestamped with µs precision, encoded (typically using the AER [8, 52]), and sent off chip. The arbitration circuitry operates on a 1 MHz clock, which allows for very low latency. If too many events are triggered, the bus can become saturated, leading to perturbation of the event timestamps.

Given this mode of operation, the number of events is clearly dependent on the scene and the CTs, summarised in Figure 1.1b. More texture, more lighting changes, and more motion will result in more events being generated. Although the DVS sensors are able to produce events very fast, they naturally have a limited bandwidth. Although the majority of motions are too slow to bring the DAVIS cameras to their limits, fast changing light sources can produce extraordinarily large numbers of events. Unfortunately, with Pulse Width Modulation (PWM) screen dimming or the 50 Hz power supply causing imperceptible flickering in lightbulbs, such sources of light are quite common. Nevertheless, the bandwidth is much higher than that of standard cameras. This is shown in Figure 1.4; a sinusoidally modulated light source is recorded with an event camera and plotted against the events generated per cycle of the light source. With a bright light source, the frequency response of the DVS is around 3 kHz, the equivalent of an exposure time of 300 µs in a conventional camera. In dim light, the frequency response drops to 300 Hz, but remains an order of magnitude higher than the 30 Hz Nyquist frequency of a 60 fps conventional camera [24].

1.2.2 Event Processing

All current methods of processing events require considering events in aggregate, since a single or few events do not contain much information. Two paradigms have emerged into which the entire literature can be neatly divided: per-event and batch processing of the events [26]. This division was used in the foundational survey of event-based vision in [24], which contains an exhaustive list of the literature up until early 2020.

In the per-event processing paradigm, a state representing the desired output is maintained and updated by incoming events. The state thus acts as a memory in which the information contained in the events can be aggregated. The Surface of Active Events (SAE) (Figure 1.5d) is a simple example of this; in a SAE, each pixel contains the timestamp of the most recent event at that location. The image represents the state, which is updated by each subsequent incoming event. The SAE thus



(a) (b)

Figure 1.4: 1.4a An LED is modulated through a sinusoidal function. The fluctuating brightness is recorded by a DAVIS event camera. 1.4b The number of events per cycle is shown for both bright and dark exposure and compared to the transfer function of a 60 fps camera. Courtesy of [24].

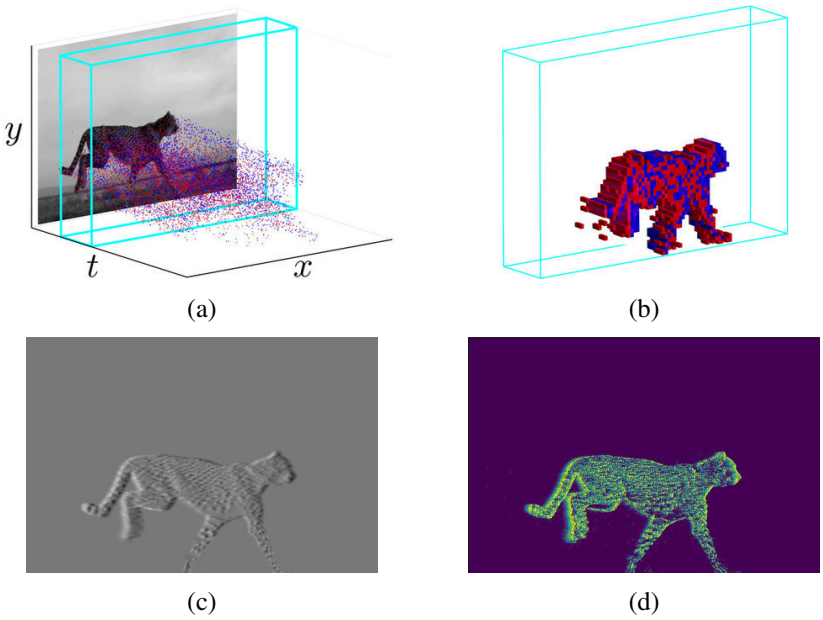
aggregates event information into a useful map of event activity. This paradigm has the advantage that the asynchronous nature of the events is preserved, so that these methods can theoretically achieve very low latency. Many of the works which fit into this category employ filtering methods to the events to estimate a state; filters are convenient as they are naturally asynchronous and are, by design, able to aggregate many different sources of information, such as events.

In batch processing, events are considered in groups to produce an output. This group is usually a slice of the events cut across the temporal dimension (such as the cyan selection in Figure 1.5). Unlike per-event processing, this introduces additional latency as events are collected for processing, although this can be eliminated if the window slides by one event. However doing so may be computationally expensive and often means recomputing the end results from scratch (an exception is optimisation based processing where the previous result can be used as a starting point for following estimates). Batch processing has the advantage that previous events can be accessed in a completely uncompressed way (one can think of the state of per-event processing as compressing the information of previous events) and can thus take full advantage of the natural associations that exist between successive events (such as in FO).

1.3 Literature

Processing events to perform useful tasks has been an active area of research due to the novelty of event data to the research community. In the past decade, event cameras have been applied to diverse tasks, such as feature detection and tracking, optic flow estimation, image reconstruction, video synthesis, image denoising, image superresolution, visual stabilisation and deblurring, stereo and monocular depth estimation, SLAM, ego-motion estimation, visual odometry, object and motion segmentation, object recognition, gesture recognition regression tasks, deep learning using Convolutional Neural Networks (CNNs), and Spiking Neural Networks (SNNs) and robot control.

This literature review focuses on the progress of optic flow estimation (Section 1.3.1), motion segmentation (Section 1.3.2), tracking (Section 1.3.3), and focus optimisation (Section 1.3.4) using events. For a comprehensive view of the event-based vision landscape, I refer to [26].



(a) (b)

(c) (d)

Figure 1.5: Several representations of the events taken from the cyan box in 1.5a. 1.5b shows the Discretised Event Volume (DEV), 1.5c shows the event image and 1.5d the Surface of Active Events (SAE).

1.3.1 Optic and Normal Flow

Optic Flow in Event Based Vision Context

Optic flow is the apparent motion of objects in a scene caused by relative motion of the camera and the scene. In conventional cameras, optic flow is estimated through the comparison of consecutive images and refers to a vector field over the pixels, where each vector represents the motion of that pixel between the frames. Since optic flow is caused by a combination of 3D scene information (depth) and camera motion, it is an essential tool in computer vision and robotics with leading roles in tasks such as Visual Odometry (VO), SLAM, object tracking, image registration, and robot control. Classically this is done with assumptions of brightness constancy (see Section 1.3.1), which together with spatial and temporal derivatives of the images ($\nabla I, \delta I/\delta t$) can be used to solve a formulation of these terms with optic flow as the solution.

Perhaps the most famous of these methods is Lukas-Kanade [53], which solves the least squares problem

$$\begin{bmatrix} I_x(q_1) & I_y(q_1) \\ I_x(q_2) & I_y(q_2) \\ \vdots & \vdots \\ I_x(q_n) & I_y(q_n) \end{bmatrix} \begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} -I_t(q_1) \\ -I_t(q_2) \\ \vdots \\ -I_t(q_n) \end{bmatrix} \tag{1.3}$$

for the optic flow $\vec{v} = (v_x, v_y)$, where each $I_x(q_i)$ and $I_t(q_i)$ is the spatial and temporal derivative respectively of the image at pixel q_i. The qs are the pixels in a window centered on the candidate pixel, which is the assumption of local constancy of the optic flow (which clearly is violated at object boundaries).

One issue with methods like Lukas-Kanade is that they suffer from the aperture problem. When viewing a line through a small aperture, such as a local patch, the apparent motion is perceived as normal to the line segment, since the component of the motion parallel to the segment is indiscernible (see Figure 1.6). Optic flow methods which do not correct for this effect produce what is known as *normal flow*, since the optic flow vectors are normal to line segments in the scene.

In event-based vision, optic flow refers to an instantaneous velocity vector field describing the velocity on the image plane of point features generating events, rather than the displacement of visual features between successive frames. Optic flow algorithms for event cameras come in three flavours, *sparse* (in which optic flow is estimated only at select parts of the scene), *event dense* (in which optic flow is estimated for each event) and *fully dense* (where optic flow is evaluated at each pixel, which do not necessarily have a recent event at their location).

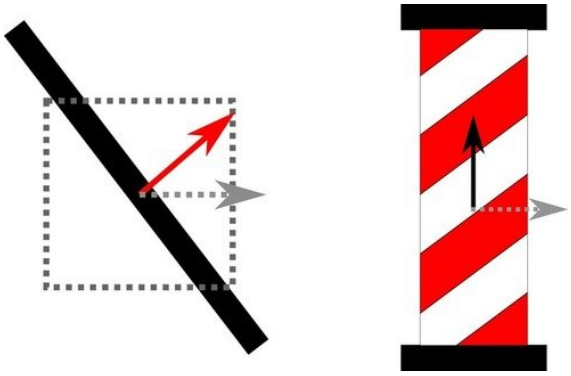


Figure 1.6: *Left* Illustration of the aperture problem. Although the line segment is moving right (grey dashed arrow), Lukas-Kanade will predict a flow normal to the segment, since it can only view the line through the local patch, or aperture (dashed box). This is also the basis of the barber pole illusion (*Right*); although the red line is moving left to right, it appears to move upward as we may only observe a small aperture.

Literature

An early normal flow algorithm was presented by [13], in which bioinspired *Direction Selective (DS)* filters were used. A DS is essentially a SAE where the timestamp of an incoming event is compared to the previous events in the 0°, 45°, 90°, etc. orientation, with the reasoning that moving edges fire neighbouring events almost instantaneously. Thus a small difference in timestamp between neighbours indicates an edge, which can be compared to prior edge detections in the SAE to compute the velocity via time-of-flight. This idea was ported to spiking neural networks by [73] in an implementation that can be thought of as several banks of DS filters. [10] also use direction selective Gabor filter banks to estimate optic flow and apply normalisation to ameliorate the aperture problem.

This method, however, only allows for discretised directions; [6] improved on this work by presenting a normal flow in which a histogram of previous events is maintained over a short time window and updated by subsequent events. The histogram is used to estimate spatial and temporal gradients, which serve as input to an overdetermined set of equations based on brightness constancy, which can be then solved for the optic flow vector by least squares. In this way, it is an adaptation of the famous Lukas-Kanade algorithm 1.3.

[91] improved on this further by providing symmetric gradients by employing central finite differences. Since the spatiotemporal derivatives of event images are used, this method is quite susceptible to noise. This is because the event images may take on small values locally due to the potentially small number of events triggered by edges crossing over the pixels and the natural sparsity of event data. This makes the derivative estimates quite unreliable, so, to mitigate this effect [91] employ a Savitsky-Golay filter to smooth the event image. Savitsky-Golay fits a low-order polynomial to adjacent points with linear least squares, an operation which is extremely efficient as it may be achieved with precomputed convolutional kernels.

The following works ([5, 61]) considered methods tailored more specifically to events. The core idea was, that since edges moving across the image form surfaces in the spatiotemporal volume (the SAE), fitting a plane to the neighbourhood of each event should reveal the local gradient of the surface. This gradient is then equivalent to the normal flow of the event at that location.

Since this planar fit is only an approximation of the surface at a given point, the assumption of constant velocity in the local neighbourhood is made to regularise the process. The planar fit is achieved through a least squares regression of the plane $ax + by + ct + d = 0$ to a set of at least three events (the event plus the local neighbours). Since this method is quite sensitive to noise and

other sources of outliers, [61] and [91] implement outlier detection (RAndom SAmple Consensus (RANSAC)), where a plane is computed and then the most distant events from the plane removed, iteratively until convergence. The velocity can then be given as:

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = -c \begin{bmatrix} \frac{1}{a} \\ \frac{1}{b} \end{bmatrix}$$

However, this formulation has the obvious defect that a, b must not be zero. In practice ([91]) these values may be very small however, in the case that the SAE is approximately aligned with the x or y axis, since this will cause the gradient of the resulting plane fit to vanish on the component along the edge's orientation. [5] deal with this by placing a threshold on the magnitude of the plane derivatives. [10, 91] use the fact that the gradient $g = (-\frac{a}{c}, -\frac{b}{c})$ encodes the direction of motion to compute the velocity. Since a and b are on a different scale (space) than c (time), this requires normalisation of the gradient vectors first, by dividing by the magnitude of the gradient, $|g| = \frac{\sqrt{a^2+b^2}}{c}$, followed by multiplication of the gradient vector length, resulting in the more robust expression

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \frac{1}{|g|^2} g = \frac{-c}{a^2+b^2} \begin{bmatrix} a \\ b \end{bmatrix}$$

These methods, however, still require the neighbourhood to be a good fit to the scene texture and dynamics. If the neighbourhood is too small, the plane fit will be arbitrary, if it is too large it will become inaccurate as the local curvature is not well approximated by the plane.

[3] estimate the optic flow jointly with image intensity. They combine several equations relating brightness constancy, the relationship of events to brightness (1.2), an optic flow constraint, and several smoothness constraints to come up with an optimisation problem which produces both optic flow and intensity estimates when solved. This method, however, does not produce results competitive with the state-of-the-art .

A highly efficient algorithm for sparse, full optic flow was presented by [51]. This method compresses the events into event images using time slices proportional to the previous detected optic flow. Then, block-matching is used to estimate the optic flow between event images, a technique from classical computer vision. It is assumed that the appearance of blocks varies only slightly between frames and thus a similarity metric can be used to find the motion of a block and thus the optic flow.

[104] applied the focus maximisation framework to optic flow estimation (see Section 2). They then used the flow estimates to segment the events fitting the motion model and refine the estimate. The resulting estimates were then updated asynchronously by incoming events using a particle filter approach. This approach has the advantage of being particularly robust to the aperture problem, since focus maximisation considers all events in aggregate, which also makes the optic flow event-dense. This idea was taken further in [102], where the segmentation and motion-compensation of the events is performed in one combined, iterative optimisation. This work showed results on optic flow, although other motion models were also used. Focus optimisation was also used in a patch-based manner to estimate sparse optic flow in [26], though again, optic flow was not a principal focus of this chapter.

[120, 121] bought modern CNNs to the problem of optic flow estimation from events with EV-FlowNet. In this work, events are aggregated and discretised to a Discretised Event Volume (DEV) representation which is passed through a UNet [89], a CNN which is fully convolutional and able to produce per-pixel outputs. The desired output, a $H \times W \times 2$ optic flow tensor, is supervised by warping the events along the optic flow trajectories as in focus optimisation. The focus of the resulting image is evaluated and applied as the loss. This work was improved on in [105], where simulated events with ground truth optic flow were shown to outperform this self-supervised method on real events. As a further benefit, this flow is actually fully dense as compared to the self-supervised method.

Table 1.2: Breakdown of extant event-based optic flow methods in chronological order (based on similar table in [24]). Methods are categorized by whether they provide normal (N) or full (F) optic flow , whether they are sparse (S), event-dense (ED) or fully-dense (FD) (see Section 1.3.1), and whether they are model based (Model) or employ Artificial Neural Networks (ANNs).

Reference	N/F?	S/ED/FD?	Model?
Delbruck et. al [13, 91]	Normal	ED	Model
Benosman et. al [6, 91]	Full	ED	Model
Orchard et. al [73]	Full	ED	ANN
Benosman et. al [5, 91]	Normal	ED	Model
Barranco et. al [4]	Normal	ED	Model
Brosch et. al [10]	Normal	ED	Model
Bardow et. al [3]	Full	FD	Model
Liu et. al [51]	Full	S	Model
Stoffregen et. al [102, 104]	Full	ED	Model
Gallego et. al [26]	Full	ED	Model
Haessig et. al [33]	Normal	ED	ANN
Zhu et. al [120, 121]	Full	ED	ANN
Almatrafi et. al [1]	Full	ED	Model
Paredes et. al [77]	Full	ED	ANN
Stoffregen et. al [105]	Full	FD	ANN
Pan et. al [75]	Full	FD	Model

Most recently, [75] estimate optic flow using both events and accompanying APS frames (which may be blurred). In this work, the authors formulate a photometric constancy constraint, which contains terms for the intensity image, deblurred intensity image, event image, contrast threshold, and optic flow. Through the addition of smoothness constraints on the optic flow and output image, the thus parametrised equations can be optimised using the primal-dual algorithm to reveal the deblurred intensity image and optic flow. This method outputs fully-dense optic flow, but is reliant on intensity images, making it unsuitable for many models of event camera.

1.3.2 Motion and Object Segmentation

As discussed in the introduction, event cameras are capable of producing large numbers of events, especially when the camera is moving. It can be difficult in this case to identify the events produced by objects of interest amongst the general event stream. In a vehicle, for example, one might be interested in the events produced by moving traffic and pedestrians and not those produced by the motion of the vehicle. In this case, motion segmentation of the events is required. An overview of the literature is in Table 1.3.

A method to detect and track circular objects (such as a ball) in the presence of clutter caused by camera ego-motion was presented in [29]. This work used an adapted directed Hough transform, where each incoming event contributes to values in Hough space on possible radii of a target circle. In order to reduce the complexity of the task and to improve the robustness, a directed Hough transform is used, using estimated normal flow (as in [5]). The assumption is that for a circular object such as a ball, most of the events are generated at the circumference. The idea was extended in [30] to a particle filter, which improved the robustness of the tracking. A downside of these methods is of course, that they require *a priori* knowledge of the target shape.

[108] detect corners in the event stream from an event camera mounted on a robot. In an *a priori* learning stage with a static scene, they learn to estimate the corner's motion as a function of the robot joint velocities. In the operation stage, corners are detected and clustered in the event stream; if there

Table 1.3: Breakdown of extant event-based motion segmentation methods in chronological order. Methods are categorized by whether they are model based (Model) or employ ANNs.

Reference	Model?
Glover et. al [29, 30]	Model
Mishra et. al [57]	Model
Vasco et. al [108]	Model
Stoffregen et. al [102, 104]	Model
Mitrokhin et. al [58]	Model
Mitrokhin et. al [59, 60]	ANN
Parameshwara et. al [76]	Model

exists a discrepancy between the motions of these corners and the expected motions, those events are segmented out. This technique, while able to detect more versatile objects, is dependent on a pre-learned setting and configuration and only works for scenes and objects with corners and is thus sparse.

[104] first proposes segmenting an arbitrary number of motions with arbitrary shapes in an event-dense manner. The method does this by collecting events up to a threshold, then applying focus-maximisation with a 2-Degree of Freedom (DoF) optic flow motion model. The events belonging to the dominant motion identified were then removed in a greedy manner and the process repeated. The thus segmented contours and motions were then used to initialise an asynchronous particle-filter based tracker. A downside of this approach is that the simple motion model requires breaking the scene down into many small moving objects if a motion such as rotation is applied.

[58] uses a similar scheme, whereby focus optimisation with a 4-DoF motion model is applied to a set of events to find the dominant motion, which is assumed to be the camera ego-motion. The average timestamp of the motion-compensated event image is the computed and discrepancies from the dominant motion average detected. These discrepancies are assumed to be independently moving objects and the resulting segmentation is applied to surrounding events by using flood-fill in the event image. This has the disadvantage that segmentation in densely textured environments or overlapping moving objects fails.

[102] improves on these results by providing a segmentation framework for arbitrary motion models and numbers of moving objects. This method uses focus optimisation on multiple motion models together with a probabilistic model for each event belonging to each model. The motion parameters and the event probabilities are then updated iteratively in a combined optimisation (w.r.t. the motion-compensated image focus), in an Expectation Maximisation (EM) approach.

[60] uses an Structure from Motion (SfM) CNN to estimate per-pixel pose, optic flow and motion segmentation masks. The work is analogous to frame based approaches, the largest difference being the input representation (a 3-channel image of the positive and negative event images and SAE stacked together). The loss is supervised by ground truth from a dataset of objects moving in 3D (recorded using a VICON motion capture system with 3D scanning), as well as a FO warping loss on the optic flow.

[59] uses a Graph Convolutional Neural Network (GCNN) architecture in which the nodes are the events. The edges connect nodes within a radius, which are parallel to the event surface. Then, given a point and its normal on the event surface (in the 3D spatiotemporal volume), only the edges orthogonal to the normal are retained, within a filtering threshold. This way most of the events with a data association are connected in the resulting graph, while still remaining compact enough for tractable computation. Training is supervised on a ground truth dataset and augmented by multiplying the timestamps by some factor. A disadvantage of this method is that it requires a ground truth dataset (in this case with only three different moving objects with one background), risking overfitting as

well as requiring rather large event slices to work well (potentially introducing a lot of latency and requiring heavy computational resources).

Finally [76] present a model based approach similar to [58], where a global motion-compensation is applied to the events, resulting in a sharp background and blurry object boundaries. They then apply motion tracking to the residual events and use K-means clustering to group the resulting tracklets. K is set to a large value and so the number of clusters is greater than the number of objects. The clusters are then merged using a contrast and distance function so that clusters that are far apart or do not match contrast during motion-compensation are not merged.

1.3.3 Tracking

Feature tracking is a fundamental tool in computer vision, so it is no surprise that it is a popular research area in event-based vision. Event cameras offer a compelling alternative to conventional cameras in this task, since they are able to track continuously without blind time between frames.

Early approaches to object tracking assumed a stationary camera, both to simplify the problem and to demonstrate the low latency and power requirements of event cameras. This allowed for cheap methods of detecting blob-like sources of events [15, 16, 18, 48, 49] or methods adapted to event cameras based on the Hough transform [12, 70].

Tracking more complex, yet still *a priori* user defined contours was shown in several following works [30, 40, 68, 69]. For example, [68] creates event images in which the value of each pixel is decayed by an exponential function with the time since the last event as the parameter. The classic Iterative Closest Point (ICP) algorithm is then used to match a pre-defined contour to these images and thus track the desired features.

Such methods simplify the tracking problem by incorporating *a priori* knowledge into the tracking algorithm, however are usually unsuited outside of their narrow use case. More generic tracking was proposed by [86, 90], by re-utilising proven trackers from conventional computer vision, by applying Harris corner detection [35] and conventional tracking [53] on patch-wise computed Images of Warped Events (IWEs).

[107] proposes a tracker in which conventional frames from the APS of the DAVIS are used to define model point sets, which are the registered to incoming events. Model point sets are generated by detecting conventional features in the APS frame and then converting them to point sets by using Canny edge maps [11] to detect the dominant source of events. So while the tracker is event-based, initialisation still requires conventional frames. [104] uses a similar technique, in which images of motion-compensated events are used as templates to which incoming events can be matched. Events update the estimate by being matched to additional candidate motions in a particle-filter based approach.

[117] proposes a purely event-based tracker in which event images are formed by integrating events for a period equal to three times their lifetimes and registering the resulting images using ICP. A variation of FO is used to estimate the optic flow, where the association between events is modeled probabilistically. The data association probabilities between events and features, the affine transform of the features, and the optic flow of the features is then estimated using EM.

1.3.4 Focus Optimisation (FO)

FO, also known as contrast maximisation, is a flexible framework for processing events in a batched manner. The core idea is that the point trajectories of visual features in the spatiotemporal volume can be parametrised according to a motion-model. Each visual feature will leave a trail of events as it moves, which are related through data association. If the events are transported back along the feature point trajectories to a common time t_{ref}, the result is a motion-compensated Image of Warped Events

(IWE). Intuitively, the more similar the estimated and actual trajectories are, the sharper the motion-compensated image will be. By optimising the focus of the image w.r.t. the motion parameters, the trajectories of visual features in the spatiotemporal volume can be estimated (Figure 1.7).

FO was first used in optical astronomy for correcting atmospherically degraded telescope images [65], interferometry [34], and correcting aberrations in Synthetic Aperture Radar (SAR) [79]. Focus optimisation has since found widespread use in both conventional [98], sonar [22], and radar imaging [97].

FO was first applied to event data by [27] to estimate angular camera velocity. The method worked by defining a 3-DoF motion model of the camera rotation. The variance is selected as the contrast measure of the motion-compensated images and optimised w.r.t. the motion parameters using nonlinear conjugate gradient descent [21].

[104] maximised the sum of squares of the pixel values (as [79]) w.r.t. an optic flow motion model to obtain the optic flow of moving objects. Optimisation of the metric was performed using a grid-search approach. Since the contrast function of multiple moving objects is not convex, the scene needed to be segmented into various individual motions.

As mentioned in Section 1.3.3, [117] used FO to track objects, by matching successive motion-compensated IWEs.

[26] produced the first description of FO as an event-based vision processing framework, showcasing the flexibility of the method by applying it to single object optic flow, depth estimation, camera rotation estimation, and planar homography estimation.

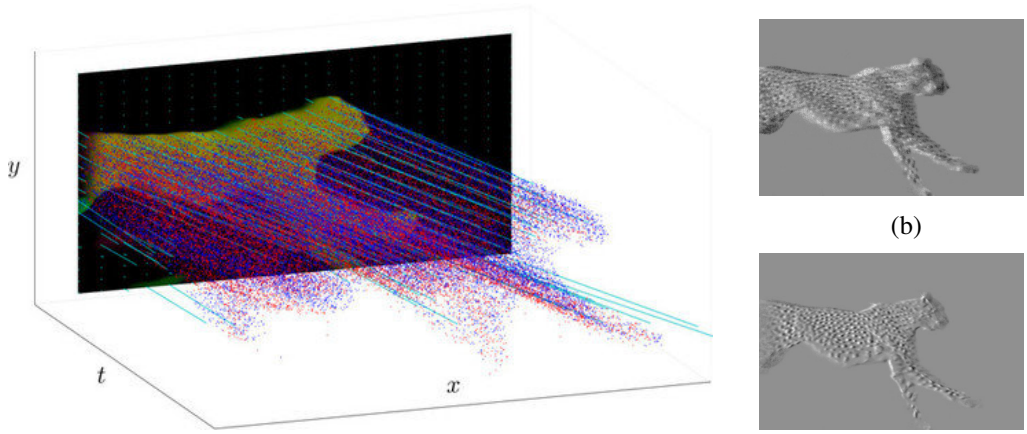
[58] uses FO for motion segmentation by optimising the total focus of a set of events w.r.t. to a 4-DoF motion model. A timestamp based focus measure is used to evaluate the focus, although this has been shown in [25] to be a less accurate measure than other standard measures. The algorithm then detects which events are not explained by this dominant scene motion and segments them using clustering. [76] uses this same concept, but builds on the clustering part of the method (see Section 1.3.2).

[102] also uses FO as a core component of motion segmentation, although the dominant motion and object motions are optimised together using an EM based approach. This method uses variance as a focus measure and arbitrary mixtures of parametrised motion models, making this approach quite general. A difference to other methods using FO is that each event is weighted by a cluster likelihood, which is used in the aggregation of the events into an event image.

[103] analyzed various focus measures for FO and described a natural classification for these based on their mode of operation. This work also demonstrated that some classes of focus measures are robust to noise and aperture uncertainty. [25] gives a thorough listing of functions that can be used as focus measures, with quantitative evaluation and applications on depth estimation, optic flow estimation, and unsupervised deep learning.

Previously, FO has been used with an explicitly parametrised model. However, the method also works as a supervisory signal in deep learning with much larger ANN models with millions of parameters. [120] used a UNet [89] to estimate optic flow, in which the supervision signal was provided by the focus of the warped image, measured by the contrast measure in [58]. This work was expanded to include depth, optic flow, and ego-motion estimation in [121]. [119] describes a Generative Adversarial Network (GAN) approach to generating event DEVs from conventional temporal image pairs. In this work one of the losses constraining the network is a FO warping loss on the optic flow predicted by a network with the DEV as input.

Most recently [50] presents a globally optimal variant of FO, which in contrast to previous methods, guarantees globally optimal convergence and does not require external initialisation. While the method *does* require a bound on the maximum angular rate $r_{\max}$ *a priori*, this parameter can be set within the reasonable limits of real scenes. The authors apply their method to 3-DoF camera angular velocity estimation, achieving SotA results. They achieve this by applying the Branch and Bound (BnB) algorithm [42] to the search space (in this case the variance w.r.t. to the angular velocity motion model). The lower bound can be trivially found as the current best (possibly sub-optimal) solution,



(a) (b) (c)

Figure 1.7: Optic flow map with events overlaid (1.7a). If the events are warped along the point trajectories (in cyan), a motion compensated IWE is formed (1.7c). In contrast, the event image (identical to applying $\mathcal{W} = 0$) in 1.7b).

the upper bound is derived by providing bounds on the sum of squares and mean of the pixel values separately, although the bound is dependent on the rotational motion model (i.e. it is a solution which applies only to the particular motion-model). This work is quite exciting as it implies that it may be possible to tractably find globally optimal solutions for FO methods. Currently the algorithm runs an order of magnitude slower than conventional local optimisation (which is itself quite slow), however the authors see ample opportunity for hardware acceleration.

Almost concurrently, [80] present a similar work, in which BnB is used to perform globally optimal FO. The authors of this work apply their algorithm to finding the motion parameters of a 2-DoF homographic warping under the assumption of fronto-parallel camera motions. The work expands on the ideas presented in this thesis, by providing bounds on all of the reward functions presented in Chapter 4. The idea is finally evaluated by performing visual odometry from a downward facing camera mounted on an Autonomous Ground Vehicle (AGV). The authors do not quote computational performance numbers, but imply that the choice of 2-DoF model is influenced by recognition that the complexity of the BnB optimisation strongly depends on the dimensionality of the search space. It seems likely that performance would be similar to that quoted in [50].

1.4 Contributions

In the following, we list our contributions in context of the literature. Key contributions are listed, with a brief discussion of the literature following.

Chapter 2

Chapter 2 presents a method for estimating optic flow from events using FO. The key contributions of this chapter are:

- A formulation for the recovery of optic flow from events via the optimisation of a focus measure of the motion-compensated events.
- An initialisation scheme that allows pairing events with their most likely motions to enable segmentation into motion clusters.
- A method of separating the constituent events for a given solution of the above optimisation problem (motion segmentation).

- A simple tracker for refining and maintaining the motion estimates based on particle filtering.

The main points of this work are a formulation for the recovery of optic flow from events through the optimisation of a focus measure of the image of the motion-compensated events. Since typical scenes contain many optic flow motions, we also provide an algorithm for motion segmentation of the events. These two elements of the method, motion-compensation and segmentation are paired in a greedy initialisation scheme, which accumulates events from the event stream and decides when segmentation is appropriate. The resulting segmentation masks and optic flow estimates are then used to generate instances of a novel, lightweight event-based tracker which is able to process events in an asynchronous, low latency manner.

In the literature: Our method addresses several shortcomings of preceding works. [13, 73, 91] show an interesting bioinspired approach, but only produce estimates in discretised directions. Lukas-Kanade based methods ([6, 91]) suffer from the aperture problem and moreover rely on estimates of the intensity image from the events in order to compute temporal and spatial gradients. Since this is usually done by direct integration of the events, the fine-grained temporal information in the events is lost. Local plane fitting methods [5, 91], while specifically tailored to events, also suffer from aperture problem and are quite susceptible to noise, relying on outlier detection to increase the robustness of the planar fits. Our work is tailored to events, making explicit use of the temporal information, while considering the events globally and avoiding the aperture problem [104].

Earlier in 2017, [27] proposed optimising the focus of motion-compensated event images w.r.t. motion parameters to estimate 3-DoF rotational velocity. The method, however, was limited to rotational motions of the camera on static scenes. Our work extends the idea to more generic scenes to estimate optic flow rather than camera rotation.

Chapter 3

Chapter 3 improves on the previously described method by using an EM approach to solving the motion-compensation and segmentation problem in one iterative optimisation. The key contributions are:

- A novel, iterative method for segmenting multiple objects based on their apparent motion on the image plane, producing a per-event classification into space-time clusters described by parametric motion models.
- The detection of independently moving objects without having to compute optical flow explicitly. Thus, we circumvent this difficult and error-prone step toward reaching the goal of motion-based segmentation.
- A thorough evaluation in challenging, real-world scenarios, such as high-speed and difficult illumination conditions, which are inaccessible to traditional cameras (due to severe motion blur and High Dynamic Range (HDR)), outperforming the state-of-the-art by as much as 10 %.
- A method for evaluating the performance of event-based motion segmentation methods in terms of *relative displacement*, with the recognition that given a larger time interval the segmentation problem becomes easier.

We achieve this by assigning a number of clusters, each with an associated motion-model. We then iteratively apply FO to multiple motion models simultaneously with a formulation for a global focus measure and then assign event-cluster likelihoods using a novel formulation. Although the number of motion models and the motions they represent need to be set *a priori*, overparametrisation does not break the method. Rather, some clusters will simply not have any events assigned to them at convergence. Our work provides highly accurate per-pixel motion segmentation, allows for usage of

arbitrary mixtures of motion models, and provides a means of quantitatively evaluating the accuracy of an event-based motion segmentation scheme [102]. This work was SotA on publication and still offers competitive advantages in that the events are segmented in an extremely fine-grained manner and that motion models can be mixed arbitrarily, to suit the use-case.

In the literature: A previous work by [58] which improved on [104], produced a motion segmentation paper in which FO was used to motion-compensate the events to identify the dominant motion as in [104]. On identification of the global motion, out of focus areas were identified by looking at anomalies in the average timestamp. Connected patches in the motion-compensated image were then grouped together using flood fill to produce a blob with a known motion, which was then tracked using a Kalman filter. An evaluation dataset was also included, with hand-labelled bounding boxes for moving objects. Our method improves on this work both quantitatively and in the output (as our method allows for the segmentation of occluded objects).

Chapter 4

Chapter 4 provides an analysis of the impact of chosen reward function for FO. Key contributions are:

- A description of two classes of reward function, magnitude and sparsity rewarding for FO.
- A definition of aperture uncertainty, an analogue to the aperture problem in conventional computer vision.
- An explanation why sparsity rewarding functions have a smoother objective function and why the two classes behave differently w.r.t. aperture uncertainty.
- Proof and experimental evidence of reward function properties.
- Proof that the sum of squares/variance metric does not necessarily have a single maximum.

We propose a simple classification of focus optimising rewards for events and show that one class of rewards has benefits in terms of noise tolerance while the other class gives benefits in terms of avoiding aperture uncertainty. We provide a definition of the aperture problem in FO and give mathematical proof together with experimental validation of our findings.

In the literature: FO as a framework for processing events was formally described in [26], however it still left the choice of contrast measuring function unexamined. In other applications of FO such as SAR and optical astronomy, this design choice was shown to have significant impact on performance. Our work was the first, concurrently with [25], to examine the choice of reward in FO.

Chapter 5

Chapter 5 applies deep learning to the problems of video reconstruction and optic flow from events. Key contributions are:

- A method for simulating training data that yields 20-40 % and up to 15 % improvement for event-based video reconstruction and optic flow CNNs.
- Dynamic train-time event noise augmentation.
- A novel High Quality Frames dataset.
- Extensive analysis and evaluation of our method.
- An optic flow evaluation metric *Flow Warp Loss (FWL)*, tailored to event data, that does not require ground truth flow.

- Open-source code, training data, and pretrained models.

CNNs have been applied to several tasks in event-based vision, including detection, segmentation, optic flow, and video reconstruction. An attractive source of training data is the use of simulators, since they are able to provide large quantities of cheap, accurately labelled training data. Acceptable results however, require that simulator output matches the target domain adequately. We use the events per second per pixel measurement as a proxy for the CT to compare different public datasets and match the settings in the simulator. These improved datasets allow us to train the same SotA datasets for 40 % and 15 % improvements in video reconstruction and optic flow estimation. Further, we are able to produce fully-dense optic flow in contrast to previous SotA networks which use focus losses on the warped events as a supervisory signal.

In the literature: Previous works using CNNs for events on simulated data usually sampled the CT values from $\mathcal{N}(\mu = 0.18, \sigma = 0.03)$, to mirror default settings in the DAVIS camera [88]. However, we show that these are probably not good assumptions for the most commonly cited datasets. [119] uses a GAN approach to generate event-voxel grids from frame pairs. The generator takes in image pairs and outputs voxel grids, while the discriminator attempts to discriminate the generated grid and one created from real data, in this case events from a DAVIS camera. The downside to this approach is that it requires a paired dataset for training and that it produces an event representation, rather than actual events, excluding alternative approaches such as asynchronous networks [56]. Recently, [14] produced a detailed noise model to greatly improve the realism of simulated events, and also used the events per pixel per second metric to compare simulation to real data.

Appendix B

As a final contribution, I present the event camera utility library, which provides python libraries for event-based algorithms. Key capabilities of the library are:

- Focus optimisation,
- Deep learning,
- Data format conversion,
- Event representation generation, and
- Visualisation

for event-based data. Most of the functionality uses `pytorch`, an open source library for Graphics Processing Unit (GPU) operations. Thus, the library makes it very easy to run algorithms on the GPU, all that needs to be done is to load the events onto the computation device in advance. Together with array broadcasting, which is used wherever possible, the library allows for fast processing of events, even within python's interpreter. For details, see Appendix B.

1.4.1 Outputs

Publications (first author)

Timo Stoffregen and Lindsay Kleeman. "Simultaneous Optical Flow and Segmentation (SOFAS) using Dynamic Vision Sensor". In: *Australasian Conf. Robot. Autom. (ACRA)*. 2017. URL: https://timostoff.github.io/18ACRA (*Best Paper Award*)

Timo Stoffregen and Lindsay Kleeman. "Event Cameras, Contrast Maximization and Reward Functions: an Analysis". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019, pp. 12300–12308.

URL: https://timostoff.github.io/19CVPR

Timo Stoffregen, Guillermo Gallego, Tom Drummond, Lindsay Kleeman, and Davide Scaramuzza. "Event-Based Motion Segmentation by Motion Compensation". In: *Int. Conf. Comput. Vis. (ICCV)*. 2019, pp. 7244–7253. URL: https://timostoff.github.io/19ICCV

Timo Stoffregen, Cedric Scheerlinck, Davide Scaramuzza, Tom Drummond, Nick Barnes, Lindsay Kleeman, and Robert Mahony. "Reducing The Sim-to-Real Gap for Event Cameras". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2020, pp. 534–549. URL: https://timostoff.github.io/20ecnn

Publications (second author)

Cedric Scheerlinck, Henri Rebecq, Timo Stoffregen, Nick Barnes, Robert Mahony, and Davide Scaramuzza. "CED: Color Event Camera Dataset". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2019. URL: https://timostoff.github.io/19CVPRW

Misc

Timo Stoffregen. "Event Camera Utility Library". In: 2020. URL: https://timostoff.github.io/projects/ecul

Chapter 2

Simultaneous Optic Flow and Segmentation

Based on [104] (Best Paper Award)

Event cameras are inherently suited to capturing dynamic rather than spatial information of a scene. For example, if the camera and lighting is static, no events are generated. This is because pixels report only relative brightness changes, rather than absolute brightness, by triggering events whenever the brightness at pixels changes above a predefined Contrast Threshold (CT). A natural association exists between events generated by a common visual feature moving through the scene. We present an algorithm to extract the dynamics from the events as optic flow by making use of this data association. We do this by transporting events along their spatiotemporal point trajectories to generate motion-compensated images. By measuring how in-focus these images are, we are able to determine how good our motion estimate is and thus optimise the focus w.r.t. the motion estimate to recover the optic flow parameters. Since our method treats objects globally, we are able to avoid the aperture problem common to other optic flow estimation methods. Since our optimisation only works for single moving objects, we also propose a segmentation scheme in order to identify events explained by a single optic flow trajectory. The result is a set of motion-compensated images (Images of Warped Events (IWEs)) with associated optic flow parameters. We use this output to efficiently track the objects using a particle filter approach, reducing the computational load of the algorithm. Since we simultaneously estimate optic flow and motion segmentation of the events, we dub our method Simultaneous Optic Flow and Segmentation (SOFAS).

2.1 Introduction

Event cameras are asynchronous, data-driven visual sensors which sample the scene at a rate proportional to the rate of change in brightness in the scene (brightness here refers to the log measured intensity at a given pixel). Event camera pixels achieve this by reporting only relative brightness changes, rather than absolute brightness, triggering events whenever the brightness at pixels changes above a predefined CT. Events are described by a tuple $e = \{\mathrm{x}, t, s\}$ where $\mathrm{x} = (x, y)$ describes the position of the event, t is the time at which the event was triggered and $s \in \{-1, +1\}$ denotes the sign of the brightness change (i.e. brighter or darker). Currently, popular implementations of event cameras such as the Dynamic and Active-pixel VIsion Sensor (DAVIS) series report timestamps with microsecond resolution. Although the accuracy and precision of these measurements are not necessarily at the microsecond level, the fine grained temporal data event cameras provide is unparalleled in computer vision. Because of these properties, the spatial information in the scene is not easily inferred from events, but needs to be estimated through some form of image reconstruction. The dynamics of the scene however are constantly reported by the events (or lack thereof). As a result, event cameras have been a very popular target of optic flow estimation innovations (see Table 1.2). The fine grained temporal information of the events allows for precise estimates of even high-speed flow, a domain which is challenging for conventional cameras and usually requires image deblurring.

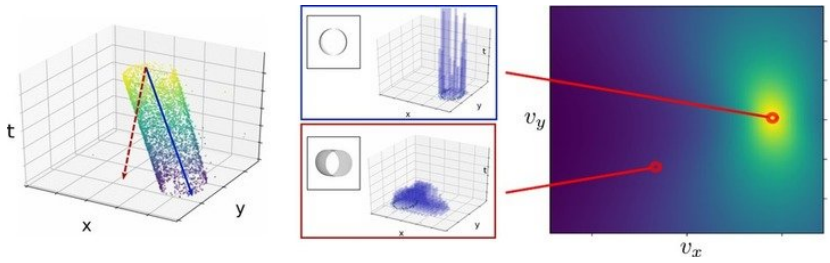


Figure 2.1: *Left:* The events produced by a circular object moving across the image plane, in the spatiotemporal volume. The blue arrow represents a good optic flow estimate, the red dashed arrow an incorrect estimate. *Middle:* Accumulations of events produced by the good and bad motion estimate, together with the IWE. *Bottom:* The reward landscape for the sum of squares of the IWE, with the locations of the good and bad estimates marked in red.

Estimating optic flow from events is also convenient, as events already represent edges in the scene, features for which flow estimation is less ambiguous [24].

Events generated by a common visual feature are naturally related. A feature moving through the scene will produce events along its trajectory, which, since they all originate from the same feature, are considered to be associated with one another. We call this relationship the data association between events. Our algorithm transports events along their spatiotemporal point trajectories, causing associated events to accumulate at common locations at a reference time. By aggregating the warped events, we can obtain a motion-compensated image of warped events (IWE) of the scene, whose focus depends on the quality of the estimated point trajectories. By measuring the focus of the IWE, we are able to determine how good our motion estimate is and thus optimise the focus w.r.t. the motion estimate to recover the optic flow parameters (Figure 2.1). Because we consider objects globally rather than only in local patches, we avoid the aperture problem common to optic flow estimation methods. This Focus Optimisation (FO) approach to processing events was demonstrated in [27] to estimate camera rotation. We extend on this idea by estimating optic flow, a very general format of motion description.

Our proposed Focus Optimisation (FO) scheme is only valid for single moving objects, since additional objects with different motions confuse the objective function landscape by introducing additional local extrema. Thus, we need to perform some form of motion segmentation on the events. We do this by motion-compensating patches of the scene with many events in them and then identifying events that are well explained by that motion. The result is a set of motion-compensated images (IWEs) with associated optic flow parameters. We use this output to efficiently track the objects using a particle filter approach, reducing the computational load of the algorithm.

2.1.1 Contributions

We propose making more direct use of the data association between events by optimising the focus $r(I_\omega)$ of the motion-compensated image I_ω obtained by warping a set of events $\mathcal{E}$ by the optic flow parameters (v_x, v_y) (for more detail, see Appendix A.2). Since the reward function r of this optimisation is multi-modal, with a local maximum per object motion in the scene, it is necessary to segment the events per mode and thus repeat the optimisation per object. This gives us a motion segmented model of the scene, where each object is represented by the object contours and the object motion.

This model has some predictive power, since, given constant illumination, the events are the product of spatial image gradients (edges made visible in the IWEs) and the motions of those edges (the optic flow). This allows us to track the recovered objects using a simple motion model together with a particle filter applied to the events and using the recovered edges as templates. While the initial optimisation step requires processing the events in batches, the tracker is very lightweight and accepts individual events to refine and update the optic flow and edge estimates. Thus, uniquely, our

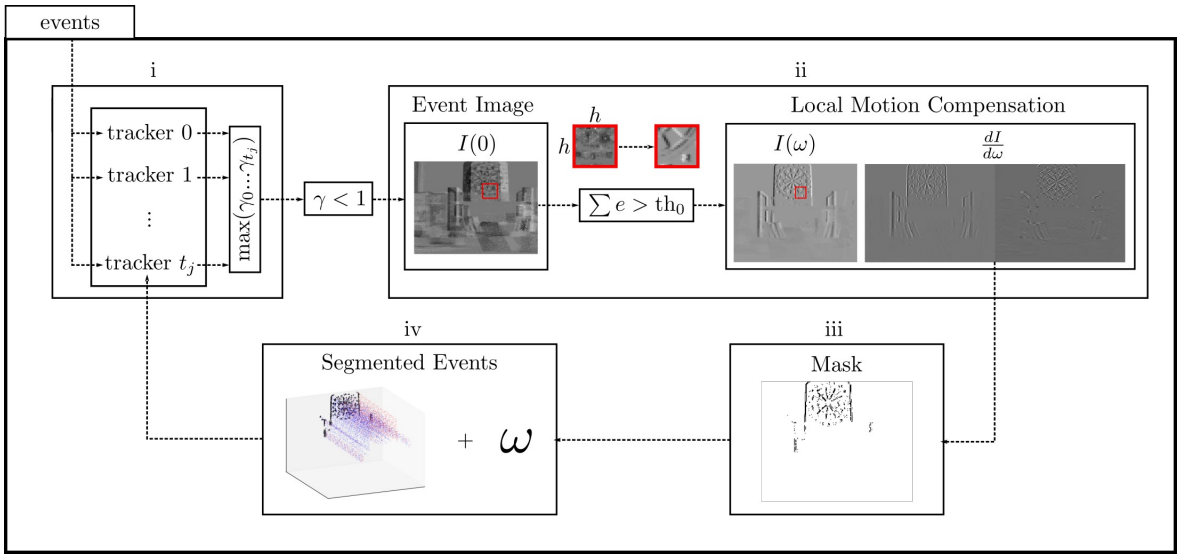


Figure 2.2: An overview of our proposed method. *(a)* An incoming event $e \in E$ is proposed to each tracker. Each tracker evaluates the event and produces a score γ which represents how well the event is predicted by the tracker. If the largest score is greater than 1, the event is attributed to the tracker. Otherwise, the event is forwarded to (b). *(b)* e is added to an event image I_0. If the sum of the pixels in the local neighborhood of e in I_0 is greater than threshold th_0, the events in that neighborhood are motion compensated using contrast maximization. The recovered motion parameters ω are applied to all events in I_0 to produce I_ω and $\frac{dI_\omega}{d\omega}$ is computed. *(c)* $\frac{dI_\omega}{d\omega}$ is used to produce a segmentation mask for the events. *(d)* The segmented events, mask and motion parameters are used to initialize a new tracker.

method partially allows for asynchronous processing of the events for the estimation of optic flow, segmentation, and edge recovery.

In summary, the key contributions are:

- A formulation for the recovery of optic flow from events via the optimisation of a focus measure of the motion-compensated events.
- An initialisation scheme that allows pairing events with their most likely motions to enable segmentation into motion clusters.
- A method of separating the constituent events for a given solution of the above optimisation problem.
- A simple tracker for refining and maintaining the motion estimates based on particle filtering.

We also offer an evaluation to contemporaneous methods based on a simple ground truth dataset as well as qualitative results on public datasets.

2.2 Method

Our method can be broken down into four main stages (see Figure 2.2):
(i) Incoming events $e_i \in \mathcal{E}$ are first compared to a set of possible extant trackers $\mathcal{T}$. Each tracker $t_j \in \mathcal{T}$ proposes a score, $\gamma_j(e_i)$, that represents how well the event e_i is predicted by t_j. The best tracker, $\gamma_{\text{max}} = \max(\gamma_0, \ldots \gamma_j)$, is selected to be updated by that event, if $\gamma_{\text{max}} \geq 1$.

(ii) If γ_{max} is less than 1, the event is added to an event image I_0 (Appendix A.2). The set of events accumulated in this event image is referred to as $\mathcal{E}_{I_0}$. The local neighborhood $\mathbf{h}$ around x_i in

I_0 is computed; if at least λ_{pix} pixels contain λ_{ev} events, the events inside the patch $\mathbf{h}$ are motion-compensated to recover the motion parameters $\omega_{\mathbf{h}}$ for the local patch, using FO. $\omega_{\mathbf{h}}$ is then used to produce an IWE I_ω for the entire set of events embedded in I_0, $\mathcal{E}_{I_0}$.

(iii) The local contrast of each event in the derivative of the IWE $\frac{dI_\omega}{d\omega}$ is then used to produce a segmentation mask for the events which can be attributed to the motion described by $\omega_{\mathbf{h}}$. This is possible, since the derivative shows clearly which parts of the image should expect a large change given a small change in parameters $\omega_{\mathbf{h}}$. Since $\omega_{\mathbf{h}}$ should be at a local optimum of the total objective function r, events which are attributed to $\omega_{\mathbf{h}}$ will have a large gradient magnitude $|\frac{dI_\omega}{d\omega}|$.

(iv) After the mask m is obtained, events $e_k \in \mathcal{E}_{I_0}$ are projected onto m using $\omega_{\mathbf{h}}$. All events projected to nonzero elements of the mask are collected into $\mathcal{E}_{t_j}$ and together with the motion parameters are used to initialise a new tracker.

SOFAS makes a few key assumptions: first, it is assumed that motions can be approximated by optic flow motion ((v_x, v_y) velocity on the image plane). This assumption only holds true over temporal slices Δt of the events which are small relative to the magnitude of the motion which those events describe. Second, it is assumed that events are caused by objects which are rigid over these small periods Δt. Interestingly, our method does not require brightness constancy assumptions, which are typically a cornerstone of optic flow estimation.

2.2.1 Focus Optimisation (FO)

The key component of our method is FO, which allows us to recover the motion parameters *on the image plane* from the set of events $\mathcal{E}$ generated by a moving object. The core idea is that events generated by a particular visual feature in motion share a natural association with one another. If these associated events are transported along their point trajectories in the spatiotemporal volume to a reference time t_{ref}, they will accumulate at common locations. Since these warped events lie on the same plane at t_{ref}, they can be formed into an image I_ω by summing the polarities of the events at each location (the Image of Warped Events (IWE)). Thus, to obtain a set of warped events $\mathcal{E}_\mathcal{W}$:

$$\mathcal{E}_\mathcal{W} = \mathcal{W}(e, t_{\text{ref}}; \omega) \; \forall e \in \mathcal{E} \tag{2.1}$$

where $\mathcal{W}$ is the warping function parametrised by ω that transports events to t_{ref}. If $\omega = 0$, this is akin to discarding the temporal dimension of the events and forms the *event image* (I_0). To obtain optic flow estimates, we apply

$$\mathcal{W}(\mathrm{x}_i, t_{\text{ref}}; \vec{v}) = \begin{bmatrix} x_i \\ y_i \end{bmatrix} - \Delta t \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{2.2}$$

Where $\mathrm{x}_i = (x_i, y_i)$ is the position of the i^{th} event, $\Delta t = t_i - t_{\text{ref}}$ is the time between the i^{th} event and t_{ref} and $\vec{v} = (v_x, v_y)$ is the optic flow velocity vector.

These warped events now lie on a common plane at t_{ref} and can be easily formed into an event image by summing the values of the events at each location. Since the warped pixel locations are not integer values, this summation is best performed using bilinear voting, to avoid aliasing effects. That is, given the set of N_e warped events $e^{'} \in \mathcal{E}_\mathcal{W}$, the value of each pixel $(u, v) \in I_\omega$ is given as

$$I_\omega(u, v) = \sum_{i=1}^{N_e} s_i^{'} \max(0, 1 - |x_i^{'} - u|) \max(0, 1 - |y_i^{'} - v|) \tag{2.3}$$

In the case that $\vec{v}$ transports the events along the point trajectories of the common visual features, the generation of the IWE can be interpreted as a motion-compensation of the events, resulting in a sharp, focused image of the original object's contours. If, however, the warping parameters are incorrect, the resulting IWE will be blurry. By squaring the IWE and summing the pixel values over the image

domain Ω

$$r = \sum_{\Omega} {I_\omega}^2 \tag{2.4}$$

we can measure the focus r, since the squaring operation will reward particularly large accumulations of events at given pixels (such accumulations occurring only in well focused images). We can thus recover the motion parameters of the moving object by solving the optimisation problem

$$\arg\max_{\vec{v}} r(\mathrm{x}, t_{\mathrm{ref}}; \vec{v}) \tag{2.5}$$

2.2.2 Initial Optimisation

The initialisation of our method needs to perform three tasks:

i Recognise when sufficient events have been collected to proceed with segmentation and flow estimation.

ii Identify the motion parameters of the dominant motion in the scene.

iii Segment the events which are caused by that motion.

The first task presents a chicken and egg problem, since recognising when sufficient events have been collected for successful optimisation requires *a priori* knowledge of object velocities. It is easy to see that in order to estimate the velocity of an object moving over the image plane, it is necessary for that object to move at least one pixel, since $\vec{v} = \frac{\Delta x}{\Delta t}$ and the minimal unit of x is 1 (since it measures in discrete units (pixels)). In practice, we have found that more than this is required for successful optic flow estimation. On the other hand, accumulating too many events introduces unwanted latency and risks violating assumptions of linearity with regards to the chosen motion model.

Our solution is to collect events into an event image I_0. When an event e is added to I_0 we inspect the neighbourhood $\mathbf{h}$ of $I_0(\mathrm{x})$ and see if at least λ_{pix} pixels contain λ_{ev} events in $\mathbf{h}$. If this is the case, FO is performed on the events which lie on the patch $\mathbf{h}$ (we use 15×15 pixels). We set λ_{pix} and λ_{ev} liberally, since the following optimisation is able to adapt to excessive events but will fail with insufficient events. We find $\lambda_{\mathrm{pix}} = 0.05|\mathbf{h}|^2$ (i.e. $0.05 \times 15 \times 15$) and $\lambda_{\mathrm{ev}} = 10$ to work well.

We then perform FO on the events inside the patch $\mathbf{h}$ to recover the motion parameters of the dominant motion in that patch. The reason we restrict ourselves to the patch during FO is that the reward function r only has exactly one maximum at the location of the correct motion parameter if there is also *exactly one* motion causing the events (see Section 2.2.2). Using a local patch of events improves the chances of capturing events from only one motion or at least disproportionately many events from one motion such that the patch contains a dominant motion with a clear optimum in r.

Optimisation

We solve the optimisation problem $\arg\max_\omega r(I_\omega)$ using a grid-search approach (see Figure 2.3). The focus function we use here is the sum of squares, that is

$$r = \sum_{\Omega} {I_\omega}^2 \tag{2.6}$$

where Ω is the image domain. We apply a blurring Gaussian kernel to the IWE prior to evaluation in order to enlarge the basin of convergence and to allow the dominant motion to be revealed as the solution to the optimisation (see Section 2.2.2). The blurring is then removed for subsequent fine-tuning.

In the grid search, the search space $\mathcal{D}$ is divided into $n_{\mathcal{D}}$ points at which r is evaluated. The new search range is then determined based on the location of the optimal sample - if it is at the edge of the

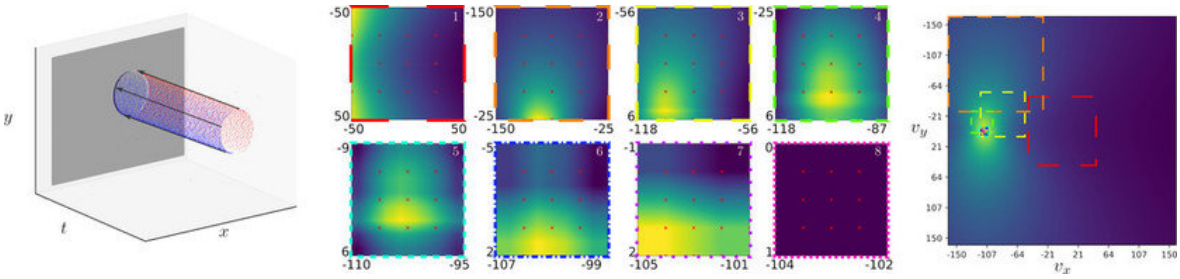


Figure 2.3: *Right:* The set of events to be optimized in the spatiotemporal volume. The arrows indicate the optic flow trajectory, with the motion-compensated image in the back. *Middle:* Successive grid searches of our optimization. Each image shows the objective function over the search space with respect to the optic flow (v_x, v_y). The red crosses indicate where samples were taken. *Left:* The objective function with the search grids overlaid. Although the optimum lies outside the range of the first search grid (red), it quickly converges on the correct solution.

search space, the new search space is enlarged, with the previous optimal sample just inside the new search space. Otherwise, the new search space is the region around the optimal sample. Convergence is achieved once the search space is smaller than a threshold λ_c (we use $\lambda_c = 1$) or if the maximum number of iterations is reached. The initial search range is $[-500, 500]$. We sample the grid using a logarithmic scale, that is, more samples are taken near the origin than at the outer reaches of the range.

Extrema of r

Given a single motion that fits the chosen motion model, it is reasonable to assume the sum of squares reward function will contain only one maximum, at the location of the actual object motion (although it is possible to construct counterexamples, see Section 4.2.1). Usually however, the reward function has only one maximum as in Figure 2.4d. Adding additional moving objects however increases the complexity of the reward landscape and although the correct optic flow solutions may be represented by peaks in that landscape, additional local maxima can be introduced which actually represent no valid solution. For example even in the simple case of two circles moving in opposite directions (Figure 2.4e), the reward landscape has clear peaks at the optic flow velocities of those circles, yet also contains a local maximum approximately around $\vec{v} = (0, -150)$ (easiest seen in digital copy). Yet more complicated is the case in which the slider sequence from [64] is overlaid with the circles from the previous example (Figure 2.4c), with multiple maxima across the reward landscape. This illustrates the difficulty of the problem and importance of the segmentation stage for accurate optic flow estimation of the scene and is further why we apply blurring to the reward during the initial optimisation steps, in order to mitigate the effect of false maxima and increase the basin of convergence for correct solutions (Figure 2.5).

Segmentation

Once a dominant motion is identified, the algorithm needs to segment the events which belong to that motion. We do this by perturbing the motion estimate in each component of $\vec{v}$ and recomputing the IWE for the perturbed motion parameters. We then inspect the change in local contrast for each event. If an event belongs to the motion model, its local contrast should decrease in the perturbed IWEs, if an event does not belong the local contrast should roughly stay the same or even improve. Events that are explained by the discovered motion parameters are thus identified.

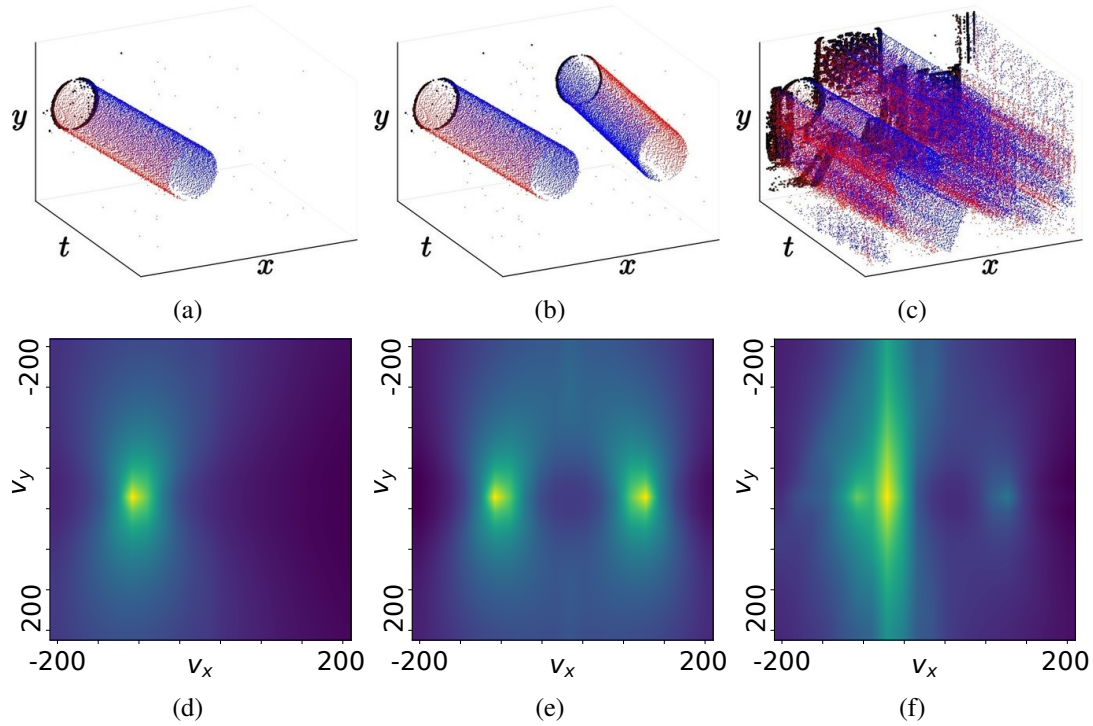


Figure 2.4: *Top row*: Events of various scenes (red/blue for positive/negative events, leading events in black for visibility). A single moving object (2.4a), two objects moving in opposite directions (2.4b) and a complex slider scene ([64], (2.4c)) blended with the two circles from (2.4b). *Bottom row*: The reward landscape of the sum-of-squares reward (2.6) for each scene in the top row. Note that (2.4d) is concave with a global optimum, but (2.4e) contains two peaks for each optic flow solution, with local maxima clearly visible in the region between them. (2.4f) has a yet more complicated reward function.

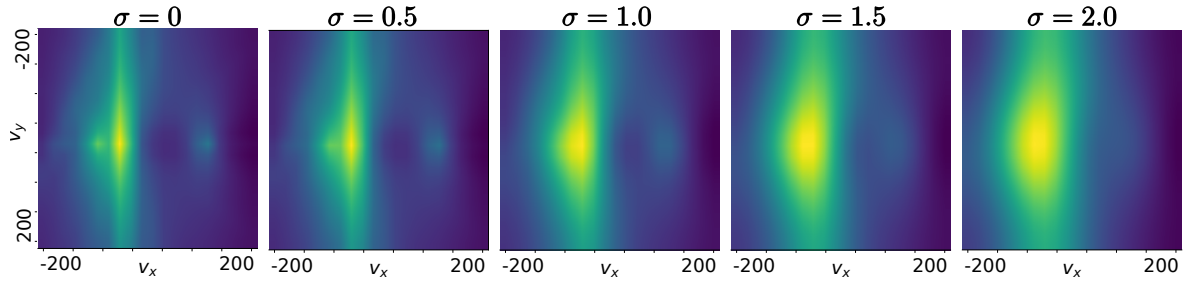
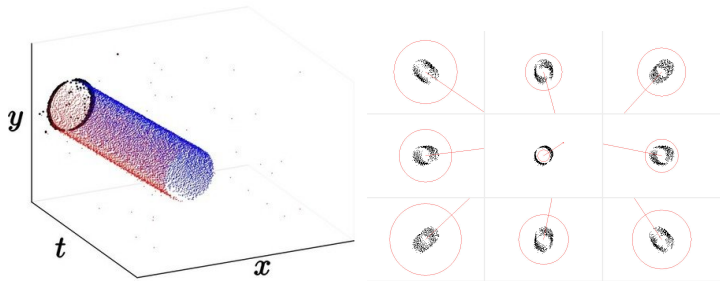


Figure 2.5: The reward function from Figure 2.4f is blurred with a Gaussian kernel with various standard deviations σ. This blurring enlarges the basin of convergence for dominant motions in the scene and reduces the impact of false maxima in the reward landscape. By applying regressive blurring per optimisation, the dominant motion of a scene can be found.



(a) (b)

Figure 2.6: 2.6b shows the tracker for the events in 2.6a. Center, the motion-compensated image representing the current best estimate. Surrounding this are several perturbed estimates, velocity shown by the size of the surrounding red circle and direction of the line. As events are added, they are compared to each template.

2.2.3 Tracking

At its core, each tracker $t_j \in \mathcal{T}$ consists of an IWE $I_{\omega j}$, a set of associated events $\mathcal{E}_j$ and the motion parameters ω_j. Once initialised, each tracker is able to receive events as an input and generate a score, γ_j, which is given simply as the local value at the warped pixel location x' of the blurred IWE:

$$\gamma = (I_\omega * \mathcal{G})(\mathrm{x}') \tag{2.7}$$

where $\mathcal{G}$ is a Gaussian kernel with $\mu = 0$, $\sigma = 1$ ($I(\mathrm{x})$ denotes the value of the pixel at location x in the image I). This score is akin to template matching, where the IWE represents the template to which the warped events are matched. After the event has been thus evaluated by all trackers, the event is assigned to the tracker $t_{\max}$ with the highest γ. The event then updates the constituent IWE, $I_{\omega\max}$, event buffer $\mathcal{E}_{\max}$ and motion parameters $\omega_{\max}$.

Solving the optimisation task described in Section 2.2.2 for each event would be prohibitively expensive and buffering events to perform the optimisation would introduce unwanted additional latency. Thus, a particle filter approach is proposed. In this scheme each tracker t_j holds k samples representing k prior estimates of ω_j. These estimates lie on a circle centered around ω_j with a radius r_ω (see Figure 2.6). The γ value of the event is computed for each sample. If the largest γ belongs to a sample, that sample's ω become the posterior estimate of ω_j and r_ω is expanded by a gain of g_ω to $g_\omega r_\omega$. Otherwise, the estimate of ω_j remains unchanged, but r_ω is shrunk by $\frac{1}{g_\omega} r_\omega$. In this way, the search space is adapted to give greater precision when the current estimate is good and to encompass a greater search space when the current estimate is poor. In practice, we found $g_\omega = 1.2$ to work well. Thus, $I_{\omega j}$ is gradually optimised w.r.t. ω_j by additional events.

Incoming events are added to the tracker's events, $\mathcal{E}_{\max}$, which is updated each time $\omega_{\max}$ is updated. Clearly, if too many events are retained in the tracker's buffer of events $\mathcal{E}_{\max}$, the assumption of linear motion is likely to be violated. Conversely, if insufficient events are retained, optimisation of ω_j will fail. To prevent this, a lifetime of events Δt is used to restrict the number of events retained. Events outside this lifetime are removed from $\mathcal{E}_{\max}$. The lifetime is set to the time it takes for the object moving over the image plane to move d_p pixels, so

$$\Delta t = \frac{d_p}{|\vec{v}|} \tag{2.8}$$

where $|\vec{v}|$ is the magnitude of the optic flow estimate. We found $d_p = 3$ to work well. This way trackers which represent slower motions have longer memory over which to store events than trackers which represent fast motions.

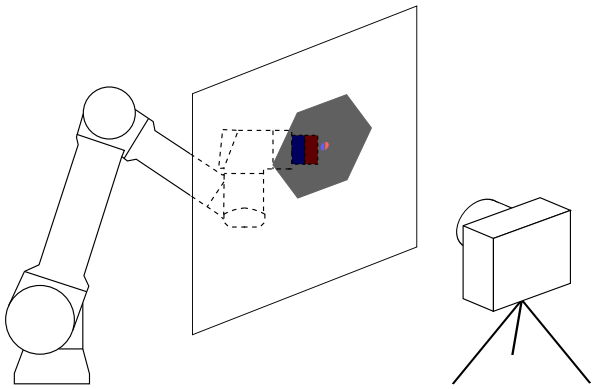


Figure 2.7: A UR5 robot arm with a magnet attached to the end effector moves behind a screen. A hexagon shape with a magnet attached follows the end effector. Since the velocity of object and field of view of event camera are known, the ground truth optic flow can be inferred.

Given the IWE $I_{\omega j}$ and the motion parameters ω_j, the expected number of future events over a time period Δt can be predicted as

$$n_p = |\vec{v}|\Delta t \int_\Omega \frac{I_{0j}}{\Delta t_{I_0}} \tag{2.9}$$

where Ω is the image domain, I_{0j} is the event image formed by the events in $\mathcal{E}_j$ and Δt_{I_0} is the time spanned by $\mathcal{E}_j$. If the number of actual events n_a being added to t_j is less than $\lambda_{\text{err}} n_p$ (the number of events predicted multiplied by a preset constant λ_{err}), the tracker is considered to have failed and is removed from $\mathcal{T}$ and the events in $\mathcal{E}_j$ are added to the initialisation image I_0 (Section 2.2.2).

2.3 Experiments

To generate ground truth optic flow for evaluation, we used a UR5 robot arm to move simple shapes across a screen. The robot was placed behind the screen and moved the objects by means of magnets attached to the end effector (see Figure 2.7. The relationship between distances on the object plane and distances on the calibrated image plane were easily measured, allowing for exact ground truth measurements. The dataset was recorded using the Java tools for Address-Event Representation neuromorphic processing (jAER) toolbox with a DAVIS-240C camera.

Several State of the Art (SotA) event-based optic flow algorithms are already implemented in the jAER toolbox [92], which motivated the implementation in this Java environment.

Since the algorithm performs several tasks, the tests are broken up into several sections. First the accuracy of the flow vectors estimated by algorithm is compared to that of Lucas-Kanade on our ground-truth dataset. We chose to benchmark our algorithm against Lucas-Kanade since it returned the lowest errors of flow vectors of any of the optic flow implementations in the jAER toolbox. Having established the competitiveness of our algorithm in comparison to traditional optical flow estimation algorithms, we then examine the performance of the algorithm at different velocities and with a rapidly accelerating dataset when compared to ground truth. Having tested the accuracy, we then show the rate at which the estimate converges and finally we show qualitative results in more complex scenes for which we were unable to generate ground truth data, but which show the ability of the algorithm to successfully segment the scene into structures with distinct flow velocities.

Ground truth optical flow data was generated by using a UR5/CB3 robot arm. Magnets were mounted on the end effector of the robot, so that the robot was able to move simple structures with magnets attached across a background panel without generating events itself. Motions were fronto-parallel to the camera and since the camera was calibrated, a direct correspondence between the robot

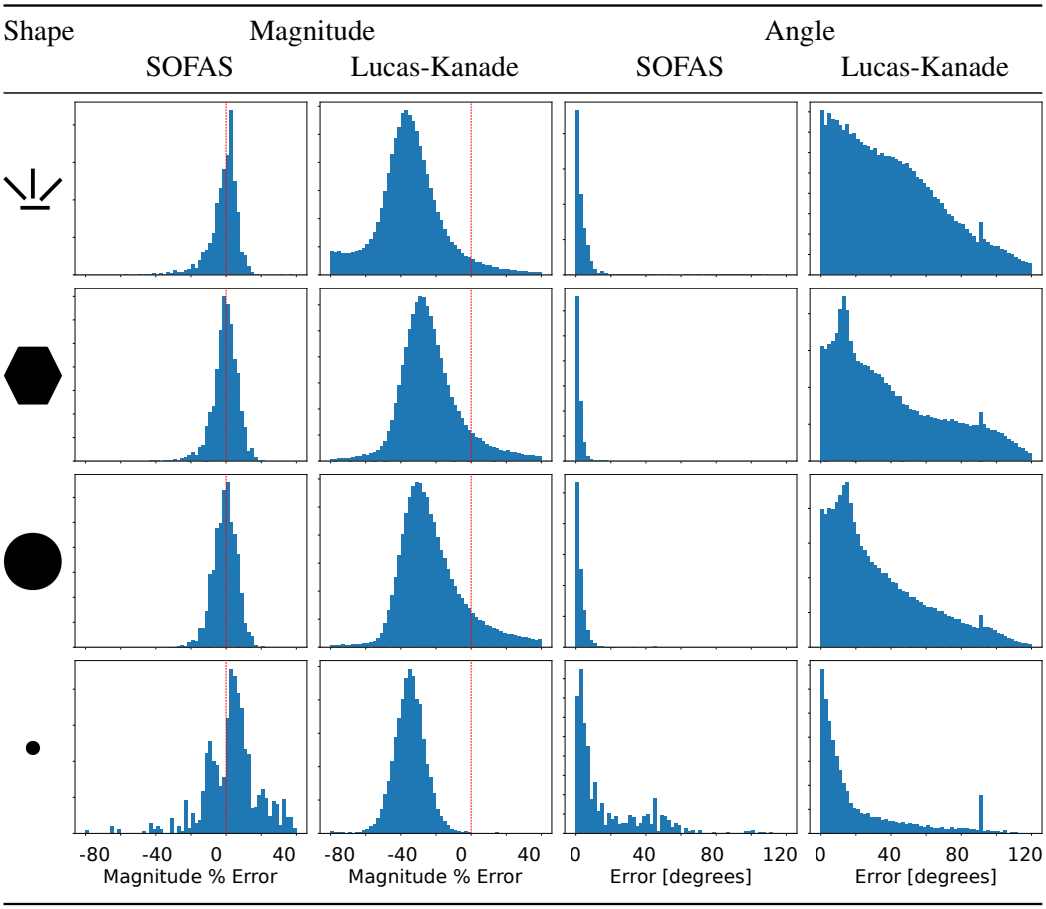


Figure 2.8: Comparison of Lukas-Kanade and SOFAS optic flow estimates. Plots show the % magnitude error and absolute angle error of optic flow estimates for both methods as normalised distributions for simplified comparison. The red dotted line indicates the 0 error point on the x axis.

end effector velocity in world coordinates and the end effector velocity on the camera plane could be determined.

2.3.1 Accuracy Comparison

Here we contrast the percentage error to ground truth of the flow vector magnitudes and the error of the flow vector directions. The data used four different shapes with a flow velocity of 58 pix/s (a collection of rectangles (width=120 pixels), a hexagon (width=65 pixels), a large circle (width=70 pixels) and a small circle (width=7 pixels)). The results (Figure 2.8) show that our algorithm not only had a lower standard deviation of errors, but also was able to avoid the aperture problem. Results from Lukas-Kanade under-estimated the flow magnitude by about 35 %. This is because Lukas-Kanade, being a local estimation algorithm (in this case using a 7x7 pixel window), tends to estimate the flow vectors normal to the image gradients rather than in the direction of optical flow. Therefore, our method was also far superior in estimating the direction. The exception is for the small circle, where the methods are roughly even, since most of the object is able to fit into the kernel window in Lucas Kanade and the aperture problem is therefore avoided.

2.3.2 Performance at Different Velocities

Figure 2.9 shows the effect of varying velocity on SOFAS. We test our algorithm over a range of velocities (5.8-289 pix/s) using the `large hexagon` dataset. The results show that the algorithm

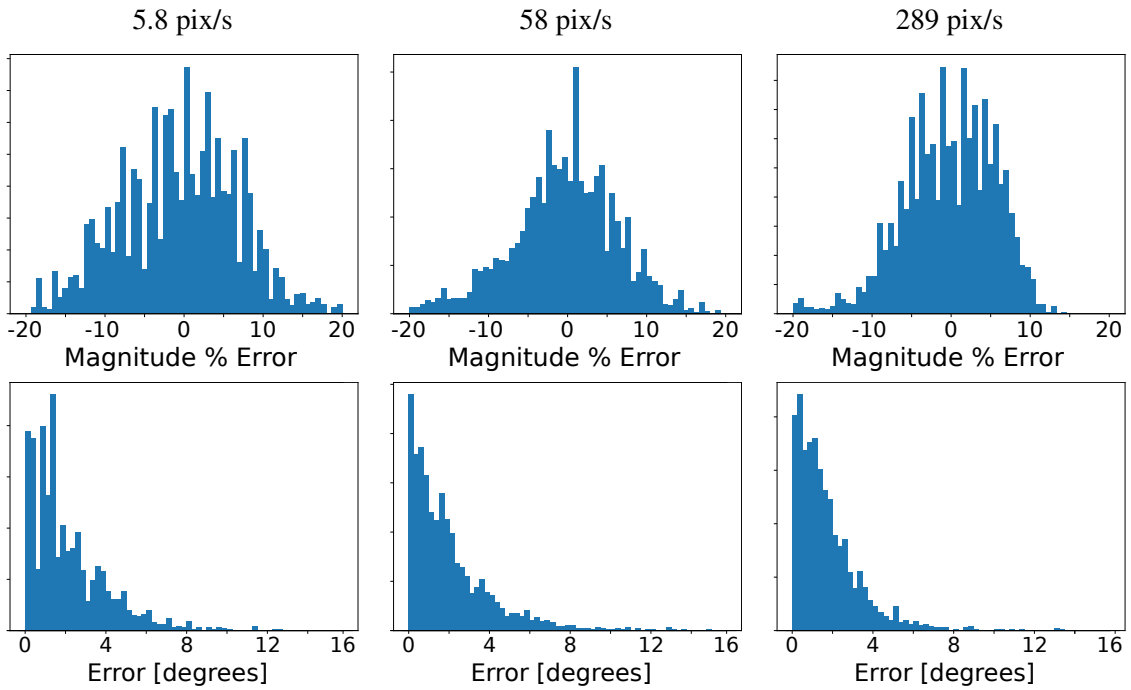


Figure 2.9: % error of flow vector magnitudes and flow vector angle error at a range of velocities computed using Simultaneous Optic Flow and Segmentation (SOFAS). Distributions normalised as Probability Density Functions (PDFs).

performs fairly consistently over the full range of velocities. A slightly larger spread in the magnitude error at 5.8 pix/s is likely because the DAVIS 240C camera used produces more events during faster motions than slower ones, over the same path. We are not certain what causes this non-ideal behaviour.

2.3.3 Performance with Acceleration

The ability of the algorithm to deal with acceleration (which is not explicitly modelled in the projections, which assume locally constant velocity) was tested by comparing the computed flow magnitude of a pendulum to the calculated velocity. Since the maximum velocity of a pendulum is given by $v_{\max} = \sqrt{2gL(1-cos(\theta_{\max}))}$ and the period by $T = 2\pi\sqrt{\frac{L}{g}}$ where $g = 9.82\,\mathrm{m/s^2}$, the pendulum length $L = 0.72\,\mathrm{m}$ and the maximum starting angle of the pendulum $\theta_{\max} = 23°$, we were able to fully characterise the velocity and frequency of the pendulum, assuming a lossless sinusoidal motion model. Since the size of the field of view of the camera was also known, we were able to get ground truth for the optic flow magnitude. The phase shift was estimated from the footage. As can be seen in Figure 2.10, the algorithm generates a relatively noisy estimate in cases where there are no constant velocities, but where the structures are under constant acceleration. Nevertheless, the algorithm is able to adapt to such circumstances to generate reasonable estimates.

2.3.4 Rotation

In this experiment, two bars were set to rotate in front of the Dynamic Vision Sensor (DVS). Predictably, SOFAS performs poorly in such circumstances, since the projection does not model rotation (Figure 2.11). Especially when the rotating bar fills the entire screen, the algorithm suffers somewhat from the aperture problem as indicated by the clearly incorrect flow predictions near the centre (blue and orange vectors, Figure 2.11b). Nevertheless, the algorithm does attempt to segment the bar

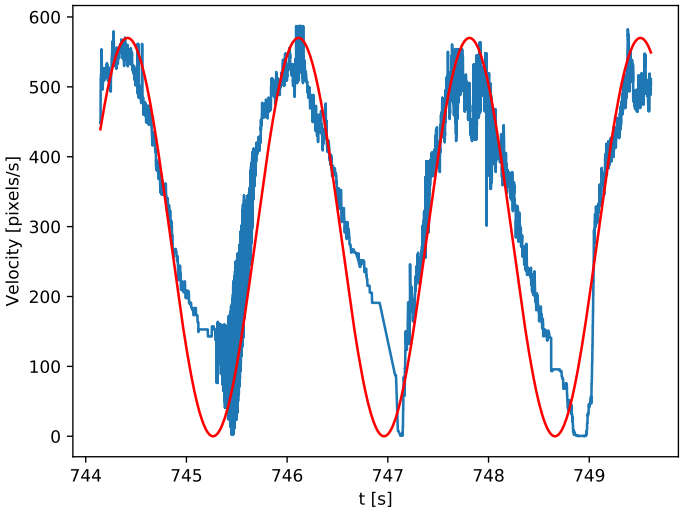
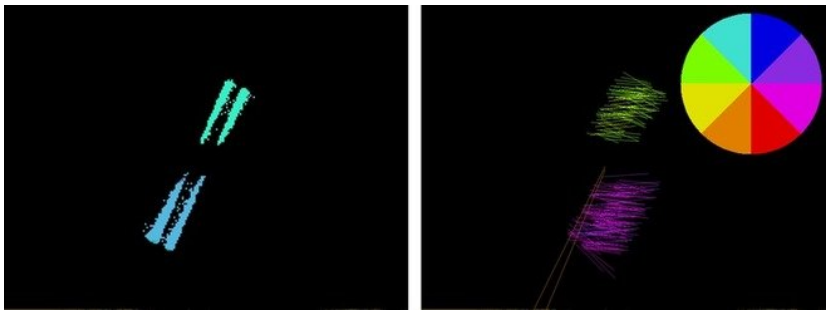


Figure 2.10: Estimated optic flow magnitude (blue) vs calculated ground truth (red) for the events generated by a pendulum (length $L = 0.72$m, starting angle $\theta_{max} = 23°$)



(a) Short bar

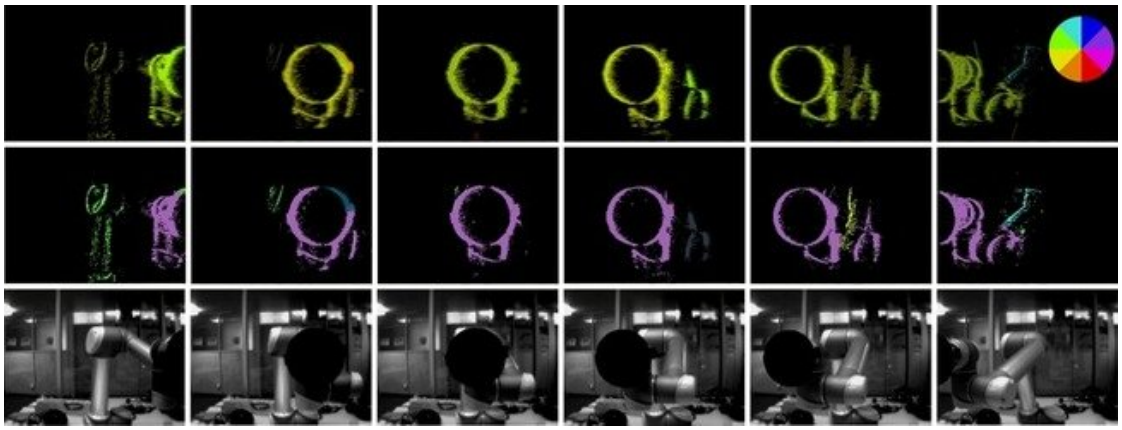
(b) Long bar (filling entire frame)

Figure 2.11: Two bars rotating counterclockwise at 60 rpm. On the left, the segmentation the algorithm performs, on the right the velocity vectors of the optic flow, coloured by direction.

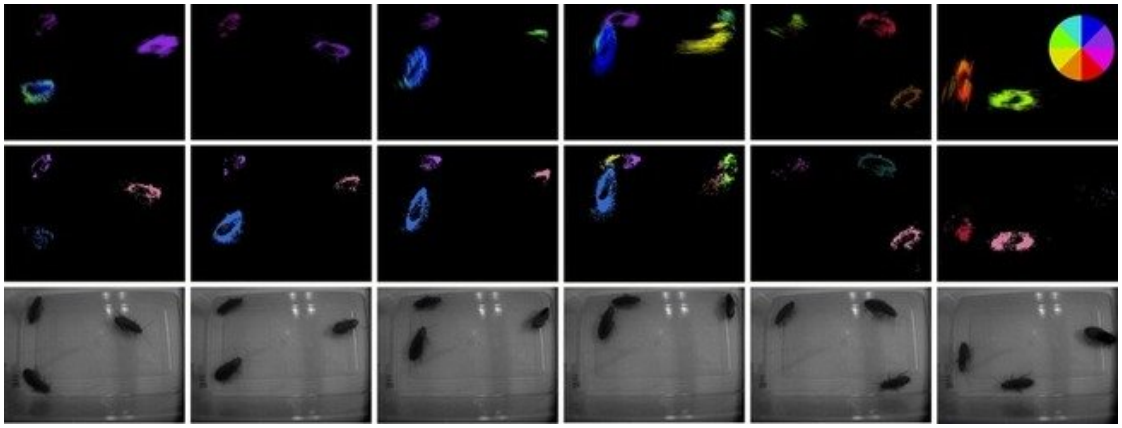
lengthwise and for the short bar at least generates reasonable optical flow. This is because over short enough timeframes it is able to assume the velocity as constant.

2.3.5 Segmentation

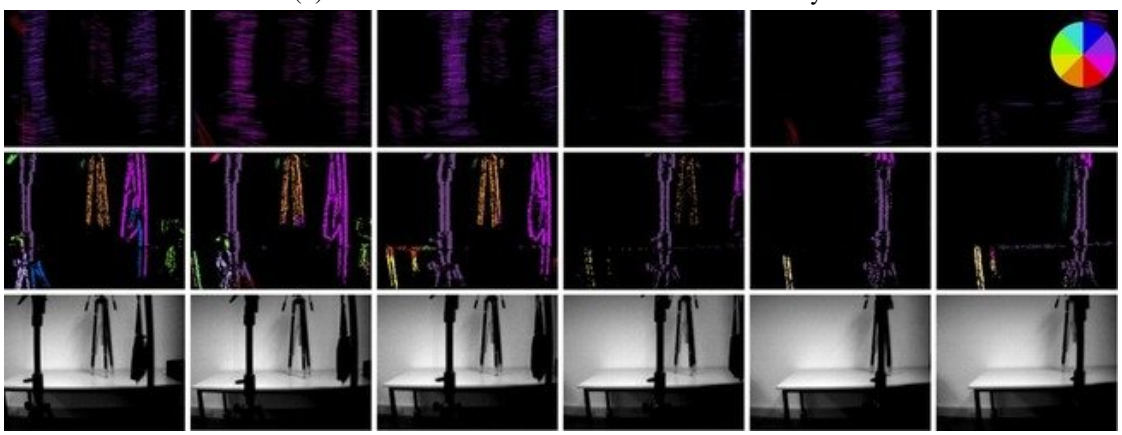
Here we aim to showcase the algorithms ability to segment the scene into structures with distinct velocities. To do this, we show the contours of the Track Planes generated on three datasets; one in which the UR5/CB3 robot arm moves the circle horizontally, one in which three cockroaches are moving randomly in a box and one in which the algorithm pans across a simple scene with four vertical bars at different distances to the DVS. As is clear (Figure 2.12), SOFAS is able to clearly segment objects by their flow velocities, and event when the same object is segmented into multiple sections (see frame two of Figure 2.12a), it is able to merge these together again. Because of the nature of the algorithm, there is some quantisation to the segmentation, though the extent of this is not easily quantifiable. In particular the result from Figure 2.12c is important, since the effective segmentation of structures at different distances makes this algorithm a potential candidate for visual odometry in the future. Indeed, it should be noted how well the segmentation in Figure 2.12c performs, separating the objects in the scene perfectly. When the tripod in the background is occluded, SOFAS is able to re-segment it as a structure immediately after reappearance. At the same time in this scene, the edges of the table are segmented into separate entities as they have distinct optic flow velocities.



(a) UR5 robot arm bearing a black cardboard circle moves the end effector from right to left in a horizontal, planar motion.



(b) Cockroaches in a box scuttle around aimlessly.



(c) The camera pans past objects at different distances from the DVS. The tripod in the background (orange segmentation) is occluded and immediately recognised as a separate structure upon reappearance (green segmentation).

Figure 2.12: Qualitative results of SOFAS. The top row shows the calculated flow vectors (see color wheel for direction), the middle row shows events coloured by segmentation, the bottom row shows grayscale stills of the scene.

2.4 Discussion

This chapter presents an algorithm for detecting the optical flow of events generated by a DVS in a novel way that, in contrast to previous works, directly makes use of the data association between events. We do this through the novel method of FO, whereby events are transported along their spatiotemporal trajectories to create motion-compensated images. We evaluate the motion-compensation result by means of a sum of squares measure. Optimising this measure with respect to the motion parameters, we are able to recover the optic flow of the events. Since the optimisation result is different for each moving object, we segment the events by means of a greedy algorithm which identifies events that can be explained by a candidate motion. We demonstrate that SOFAS is superior in accuracy when compared to a traditional optical flow algorithm.

The SOFAS algorithm does have some limitations however. *(i)* Our method requires motions to fit the optic flow motion model implemented. We do show in Section 2.3.3-2.3.4 that SOFAS is robust enough to provide reasonable optic flow estimates, however it would be better to provide motion models more suited to the target motion. For example, in the rotating bar experiment, SOFAS discretises the bar into several sections, showing the limits of the approximation. *(ii)* Our method is dependent on several hyperparameters, some of which are scene dependent. In particular, λ_{pix} and λ_{ev}, the thresholds to determine whether sufficient events have been buffered to perform FO, should ideally be lower for low contrast scenes (when fewer events are triggered) than for high contrast scenes. *(iii)* Finally, our implementation is quite slow. This may be partly due to the choice of Java as implementation language, but the complexity of the optimisation step is potentially quite large, requiring $4\lambda_{\text{maxi}}N_e$ warping operations in the worst case. Each optimisation sample can be taken independently of the other however and the events can be warped independently as well, so we see a large potential for parallelisation and therefore acceleration. Our implementation had an average throughput of 5600 events per second, running on a 3.4 GHz Central Processing Unit (CPU).

We extend the ideas presented here and address the limitations of SOFAS in Chapter 3.

2.5 Resources

Video and dataset: `https://timostoff.github.io/19ACRA`

Chapter 3

Motion Segmentation using Motion Compensation

Based on [102]

Previous methods for performing motion segmentation of an event stream [58, 104] are dependent on hyperparameters and spatial or morphological operations such as flood-fill. This has distinct disadvantages, since hyperparameters require tuning to the expected scene and morphological operations are prone to break for textured scenes or occlusions. The previously mentioned methods are also designed to work with one particular motion model in mind. We present a principled way of acquiring motion segmentation from an event stream which does not require hyperparameter tuning and is independent of the choice of motion model. We do this by jointly estimating the event-object associations (i.e. segmentation) and the motion parameters of the objects by Focus Optimisation (FO) in a manner reminiscent of Expectation Maximisation (EM). We also provide a novel way of experimentally evaluating the accuracy of motion segmentation algorithms for event cameras, recognising that the segmentation problem becomes easier the larger the time window under consideration is. We provide a thorough evaluation of our method on a public dataset, outperforming the state-of-the-art by as much as 10 % and show that our method is capable of 90 % accuracy at 4 pixels relative displacement of the moving objects.

3.1 Introduction

Motion segmentation is a particularly significant problem in event-based vision, since the sensor ultimately only detects parts of the scene which are in motion. In the case of a moving camera this can result in many events, which can be very difficult to differentiate from events caused not only by camera ego-motion, but also by independently moving objects. Cases in which such a differentiation might be vital are not difficult to think of; in a moving vehicle such as a car or Unmanned Aerial Vehicle (UAV), many events are generated due to ego motion, yet it may be desirable to differentiate moving objects such as pedestrians or other vehicles. In the case of a static camera, the problem is simplified but not trivial, since the asynchronous nature of the events require novel processing methods.

We consider the problem of segmenting a scene viewed by an event-based camera into independently moving objects. In the context of traditional cameras, this problem is known as *motion segmentation* [114], and it is an essential pre-processing step for several applications in computer vision, such as surveillance, tracking, and recognition [72]. Its solution consists of analysing two or more consecutive images from a video camera to infer the motion of objects and their occlusions. In spite of progress in this field, conventional cameras are not ideally suited to acquiring and analysing motion; since exposure time is globally applied to all pixels, they suffer from motion blur in fast moving scenes. Event-based cameras are a better choice since they sample at exactly the rate of scene dynamics, but conventional techniques cannot be applied to the event data.

3.1.1 Literature

Motion segmentation in the case of a static event-based camera is simple, because in this scenario events are solely due to moving objects (assuming there are no changes in illumination) [49, 69, 82]. More challenging is the case of a moving camera, since in this scenario events are triggered everywhere on the image plane, produced by both the moving objects as well as the apparent motion of the static scene induced by the camera's ego-motion. Hence, event-based motion segmentation consists of classifying each event into a different object, including the background. However, each event carries very little information and therefore it is challenging to perform the mentioned per-event classification. Event-based motion segmentation in the presence of event-clutter caused by camera ego-motion or many independently moving, overlapping objects has been addressed previously in [29, 58, 104, 108].

In [29], a method is presented for detection and tracking of a circle in the presence of event clutter. It is based on the Hough transform using optical flow information extracted from temporal windows of events. Segmentation of a moving object in clutter was also addressed in [108]. It considered more generic object types than [29] by using event corners as primitives, and it adopted a learning technique to separate events caused by camera motion from those due to the object. However, the method required the additional knowledge of the robot joints controlling the camera.

Segmentation has been recently addressed by [58, 104] using the idea of FO. For example, [104] first fitted a motion-compensation model to the dominant events, then removed these and fitted another motion model to the remaining events in a greedy manner. Similarly, [58] detected moving objects in clutter by fitting a motion-compensation model to the dominant events (i.e. the background) and detecting inconsistencies in the average timestamp with respect to that motion (i.e. the objects). The objects were then 'segmented' via morphological operations on the warped image, and were used for tracking. The method could handle occlusions during tracking, but not during detection.

Our method differs from [29] in that we demonstrate segmentation on objects with arbitrary shapes, and from [108] in that we do not require additional inputs (e.g. robot joints). Our work is most related to [58, 104]; however, it has the following novelties: (*i*) it actually performs *per-event* segmentation, rather than just providing bounding boxes for detected object regions, (*ii*) it allows for general parametric motion models (as those in [26]) to describe each cluster, (*iii*) it performs optimisation based on a single objective function (as opposed to two sequential optimisation criteria in [58]), (*iv*) it is able to handle occlusions between objects at any point in time. The proposed method is the first one that *jointly* estimates the apparent motion parameters of all objects in the scene and the event-object associations (i.e. segmentation), (*v*) it does not require setting of scene-dependent hyperparameters or the usage of morphological or 'conventional' vision operations on event images.

3.1.2 Method

We propose a method to tackle the event-based motion segmentation problem in its most general case, with a possibly moving camera. Inspired by classical layered models [109], our method classifies the events of a space-time window into separate clusters (i.e. 'layers'), where each cluster represents a coherent moving object or background (see Figure 3.1). Our method jointly estimates the motion parameters of the clusters (each of which represent a parametrised motion model) and the event-cluster associations (the likelihood of each event belonging to each cluster). Our method is flexible since it allows for mixtures of different parametric motion models (for example optic flow, rotations, zooming, or other higher degree combinations). While the number of clusters is fixed *a priori*, our method is not sensitive to the number of layers N_l, since overparametrisation simply causes redundant layers to be assigned no events and 'mode collapse'.

The method jointly estimates the motion parameters of the clusters and the event-cluster associations (i.e. likelihood of an event belonging to a cluster) in an iterative, alternating fashion, using

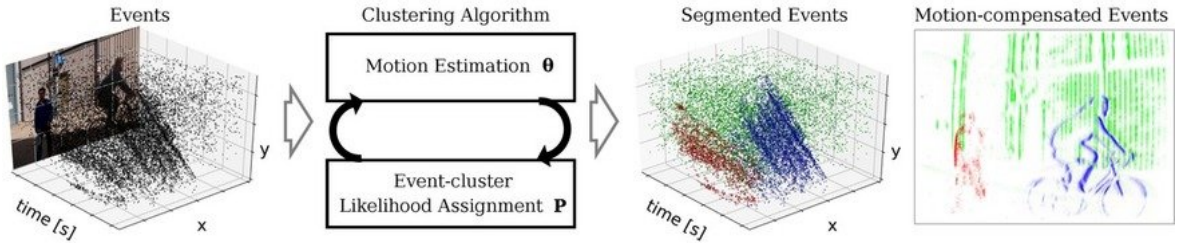


Figure 3.1: Our method segments a set of events produced by an event-based camera (Left, with color image of the scene for illustration) into the different moving objects causing them (Right: pedestrian, cyclist, and camera's ego-motion, in color). We propose an iterative clustering algorithm (Middle block) that jointly estimates the motion parameters ω and event-cluster membership probabilities $\mathbf{P}$ to best explain the scene, yielding motion-compensated event images on all clusters (Right).

an objective function based on motion-compensation [26, 27] (basically, the better the estimated unknowns, the sharper the motion-compensated event image of each cluster). Our method is flexible, allowing for different types of parametric motions of the objects and the scene (translation, rotation, zooming, etc.).

The method is thoroughly evaluated in challenging, real-world scenarios, such as high-speed and difficult illumination conditions, which are inaccessible to traditional cameras (due to severe motion blur and High Dynamic Range (HDR)). Our method quantitatively performs up to 10 % better than previous State of the Art (SotA) while our qualitative results showcase the capabilities of the sensor and the proposed method.

3.1.3 Contributions

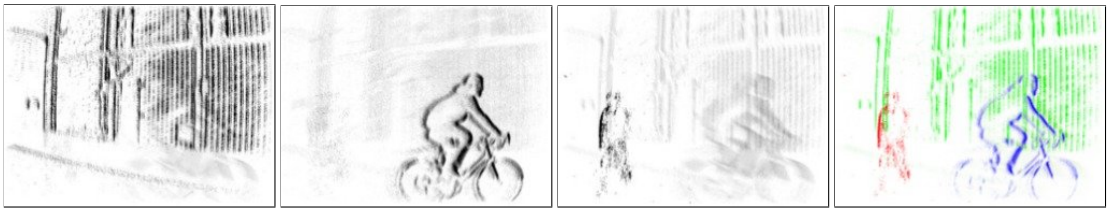
The key contributions of this chapter are:

- A novel, iterative method for segmenting multiple objects based on their apparent motion on the image plane, producing a per-event classification into space-time clusters described by parametric motion models.
- The detection of independently moving objects without having to compute optical flow explicitly. Thus, we circumvent this difficult and error-prone step toward reaching the goal of motion-based segmentation.
- A thorough evaluation in challenging, real-world scenarios, such as high-speed and difficult illumination conditions, which are inaccessible to traditional cameras (due to severe motion blur and HDR), outperforming the state-of-the-art by as much as 10 %.
- A method for evaluating the performance of event-based motion segmentation methods in terms of *relative displacement*, with the recognition that given a larger time interval the segmentation problem becomes easier.

As a by-product, our method produces sharp, motion-compensated images of warped events, which represent the appearance (i.e. shape or edge-map) of the segmented objects (or background) in the scene (Figure 3.1, Right).

3.1.4 Individual Contribution

This work was a collaboration with Guillermo Gallego, a post-doc at RPG Zurich. Guillermo helped develop the theoretical framework and was less involved with implementation and experimental work.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) All clusters

Figure 3.2: Our method splits the events into clusters (one per moving object), producing motion-compensated images of warped events Images of Warped Events (IWEs), as shown in (a)-(c) for the three objects in the scene of Figure 3.1. The likelihood of each event is represented by the darkness of the pixel. Since the likelihoods are nonzero, "ghosts" can be seen in the individual clusters. IWEs in (a)-(c) are merged into a single image (d), using a different color for each cluster. Segmented events in upcoming experiments are shown using this colored motion-compensated image representation.

3.2 Method

Our method is inspired by the combination of classical layered models [109] and event-based motion-compensation [26]. In the following we describe the motion segmentation problem addressed and present our proposed solution.

3.2.1 Problem Statement

Since each event carries little information and we do not assume prior knowledge of the scene, we process events in packets (or groups) to aggregate sufficient information for estimation. Specifically, given a set of events $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$ in a space-time volume of the image plane $V_e = \Omega \times T$, we address the problem of classifying them into N_l clusters (also called 'layers'), with each cluster representing a coherent motion, of parameters ω_j. We assume that the time slice of the events T is a sufficiently short interval so that the motion parameters of the clusters $\omega = \{\omega_j\}_{j=1}^{N_l}$ are constant.

The images on both sides of the algorithm block in Figure 3.1 illustrate the above-mentioned problem and its solution, respectively. Notice that since events have space-time coordinates, clusters are three-dimensional, contained in V_e. Also, because corresponding events caused by the same point of a moving edge describe point trajectories in V_e, optimal clusters should contain them and therefore have a 'tubular' shape (Figure 3.1, segmented events). Implicit in motion segmentation, if two objects share the same motion they belong to the same cluster, regardless of their location.

3.2.2 Summary of Proposed Solution

Leveraging the idea of FO [27, 104], we seek to separate the events $\mathcal{E}$ into clusters by maximising *event alignment*, i.e. maximising the sharpness of motion-compensated images (one per cluster) of warped events.

More specifically, the idea of FO is that, as an edge moves on the image plane, it triggers events on the pixels it traverses. The motion of the edge can be estimated by warping the events to a reference time and maximising their alignment, producing a sharp IWE. The sharpness can be measured and thus optimised w.r.t. the parameters of the chosen motion model. In the case of multiple objects with different motions, maximal event alignment cannot be achieved using a single warp, and so, several warps (i.e. motion models or 'clusters') are required, as well as identifying which events belong to which object (i.e. 'event-cluster associations'). The associations are discovered by means of a formulation which takes into account the ratio of local sharpness of the IWE for each event warped to each cluster. This is the essence of our approach, which is illustrated in Figure 3.1 and 3.2. Figure 3.1

shows the events produced by three objects in a scene with a moving camera: a pedestrian, a cyclist and a building facade, which produces events due to camera motion. Each object has a different motion relative to the camera and triggers events as it moves over the image plane. When events are warped to a reference time (e.g. $t_{\text{ref}} = 0$) according to a candidate motion model, they produce an IWE. If the candidate motion coincides with the true image-plane motion of the object causing the events, the warped events align, producing a sharp motion-compensated IWE, as shown in Figure 3.2 using three different motion models (one per object). Otherwise, they do not align, producing a blurred IWE. We use the sharpness of such IWEs (measured by the variance of the image) as the main cue to segment the events. Our method jointly identifies the events corresponding to each independently moving object as well as the object's motion parameters.

3.2.3 Mathematical Formulation

In contrast to previous methods [58, 104], we explicitly model event-cluster associations in the motion-compensation framework, i.e. $p_{kj} = \mathbf{P}(e_k \in \ell_j)$ is the probability of the k-th event belonging to the j-th cluster. Let $\mathbf{P} = (p_{kj})$ be an $N_e \times N_l$ matrix with all event-cluster associations. The entries of $\mathbf{P}$ must be non-negative, and each row must add up to one. Using these associations, we define the *weighted* IWE of the j-th cluster as

$$I_{\omega j}(\mathbf{x}) = \sum_{k=1}^{N_e} s p_{kj}\, \delta(\mathbf{x} - \mathbf{x}'_{kj}), \tag{3.1}$$

with $\mathbf{x}'_{kj} = \mathcal{W}(\mathbf{x}_k, t_k; \omega_j)$ the warped event location (via warping function $\mathcal{W}$), and δ the Dirac delta. Equation (3.1) states that events are warped,

$$e_k = (\mathbf{x}_k, t_k, s_k) \mapsto e'_k = (\mathbf{x}'_k, t_{\text{ref}}, s_k) = \mathcal{W}(e_k), \tag{3.2}$$

and the values $p_{kj} \geq 0$ (i.e. weights) are accumulated at the warped locations $\mathbf{x}'_k$. Event alignment within the j-th cluster is measured using image contrast [31], which is defined by a sharpness/dispersion metric, such as the variance [27]:

$$\sigma^2(I_{\omega j}) = \frac{1}{|\Omega|} \int_\Omega (I_{\omega j}(\mathbf{x}) - \mu_{I_{\omega j}})^2 d\mathbf{x}, \tag{3.3}$$

where $\mu_{I_{\omega j}}$ is the mean of the IWE over the image plane Ω.

We propose to find the associations $\mathbf{P}$ and cluster parameters ω that maximise the sum of contrasts of all clusters by solving the optimisation problem:

$$(\omega^*, \mathbf{P}^*) = \arg\max_{(\omega, \mathbf{P})} \sum_{j=1}^{N_l} \sigma^2(I_{\omega j}). \tag{3.4}$$

Since the optimisation of Equation 3.4 addressed does not admit a closed-form solution, we devise an iterative, alternating optimisation approach, which we describe in the next section.

The pseudo-code of our method is given in Algorithm 1. From the output of Algorithm 1, it is easy to compute motion-compensated images of events corresponding to each cluster, i.e. the weighted IWEs (3.1) shown in Figure 3.2. Each IWE shows the sharp, recovered edge patterns (i.e. and appearance model) of the objects causing the events.

Algorithm 1 Event-based Motion Segmentation

1: **Input**: events $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$ in a space-time volume V_e of the image plane, and number of clusters N_l.
2: **Output**: cluster parameters $\omega = \{\omega_j\}_{j=1}^{N_l}$ and event-cluster assignments $\mathbf{P} \equiv p_{kj} \doteq \mathbf{P}(e_k \in \ell_j)$.
3: **Procedure:**
4: Initialise the unknowns $(\omega, \mathbf{P})$.
5: **Iterate** until convergence:
6: • Compute the event-cluster assignments p_{kj} using (3.6).
7: • Update the motion parameters of all clusters (3.5).

3.2.4 Alternating Optimisation

Each iteration of Algorithm 1 has two steps (lines 6 and 7), as in a coordinate ascent algorithm. If the associations $\mathbf{P}$ are fixed, we may update the motion parameters

$$\omega \leftarrow \omega + \mu \nabla_\omega \Big(\sum_{j=1}^{N_l} \sigma^2(I_{\omega j}) \Big) \tag{3.5}$$

by taking a step ($\mu \geq 0$) in an ascent direction of the objective function (3.4) with respect to the motion parameters. Motion-compensation methods [26, 58] typically use gradient ascent or line search to solve for the motion parameters that maximise some objective function of the IWE. In our case, because the IWE (3.1) depends on both ω and $\mathbf{P}$ and we seek to jointly estimate them, we do not wastefully search for the best ω given the current estimate of $\mathbf{P}$. Instead, we update ω using (3.5), proceed to refine $\mathbf{P}$ (see (3.6)), and iterate.

Fixing the motion parameters ω, we may refine the associations $\mathbf{P}$ using a closed-form probability partitioning law:

$$p_{kj} = \frac{r_j(\mathrm{x}'_k(\omega_j))}{\sum_{i=1}^{N_l} r_i(\mathrm{x}'_k(\omega_i))}, \tag{3.6}$$

where $r_j(\mathrm{x}) \neq 0$ is the local contrast (i.e. sharpness) of the j-th cluster at pixel x, and it is given by the value of the weighted IWE, $r_j(\mathrm{x}) = I_{\omega j}(\mathrm{x})$. Thus, each event is softly assigned to each cluster based on how it contributes to the sharpness of all N_l IWEs. The alternating optimisation approach in Algorithm 1 resembles the EM algorithm, with the E-step given by (3.6) and the M-step given by (3.5).

3.2.5 Initialisation

The proposed alternating method converges locally in practice (i.e. there is no guarantee of convergence to a global solution), and it requires an appropriate initialisation of $\omega, \mathbf{P}$ to start the iteration.

Several initialisation schemes are possible, depending on the motion models. For example, if the warps of all clusters are of optical flow type, one could first extract optical flow from the events (e.g. using [5, 120]) and then cluster the optical flow vectors (e.g. using the K-means clustering algorithm). The resulting cluster centres in velocity space would provide an initialisation for the motion parameters of the clusters ω.

We follow a greedy approach similar to [104]. Essentially the events are motion compensated using each motion-model and the model that results in the best contrast is chosen to remove events in the gradient-based manner described in Section 2.2.2.

The difference is that the events, rather than being assigned an absolute segmentation value, are assigned likelihoods according to the ratio of the local contrast measures to the sum of contrast measures as in Equation 3.6. The process is repeated for the remaining clusters until all motion parameters ω and associations $\mathbf{P}$ have been filled.

This method works well in practice and results in estimates which are close enough to allow convergence of the main algorithm. There is no obvious reason our method should not work for arbitrary motion models and we found it to converge for all motion models we tried.

3.2.6 Discussion of the Segmentation Approach

The proposed approach is versatile, since it allows us to consider diverse parametric motion/warping models, such as linear motion (optic flow) [26, 104, 117], 3-Degree of Freedom (DoF) rotational motion [27], 4-DoF motion [58], 8-DoF homographic motion [26], etc. Moreover, each cluster may have a different motion model, $\{\mathcal{W}_j\}_{j=1}^{N_l}$, as opposed to having a single model for all events, and therefore, all clusters. For example, if the camera is rotating, the 'background' cluster may be modelled by a rotational warp, while clusters of 'foreground' objects may be described by a more appropriate motion model (such as linear, optical flow motion or a 3-DoF camera rotation model). This characteristic is unique to our method.

It is also worth noting, that the proposed method classifies events according to motion without having to explicitly compute optical flow, which is a widespread motion descriptor. Thus, our method is not simply optical flow clustering. Instead, our method encapsulates motion information in the warps of each cluster, thus by-passing the error-prone step of optical flow estimation in favour of achieving the desired goal of motion segmentation of the events.

The edge-like motion-compensated IWEs corresponding to each cluster are, upon convergence, a description of the intensity patterns (entangled with the motion) that caused the events. Thus our method recovers fine details of the appearance (e.g. shape) of the objects causing the events without having to estimate a (computationally expensive) 3D scene representation. In [58] fine details were only available for the dominant motion cluster.

Finally, the number of clusters N_l is a hyper-parameter that may be tuned by a meta-algorithm (in the experiments, we set N_l manually). This is a well-known topic in clustering [23]. While automatically determining the optimal N_l depending on the scene is outside the scope of this chapter, it should be noted that, as we show in Section 3.3.3, our method is not sensitive to excess clusters N_l.

3.2.7 Warp Functions

We use three warping functions. These are:
a) The optic flow warp:

$$\mathcal{W}_v(x, y, \Delta t; \omega) = \begin{bmatrix} x \\ y \end{bmatrix} - \Delta t \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{3.7}$$

where the warp parameters $\omega = (v_x, v_y)$ and Δt is the difference between the event timestamps and the reference time, $t - t_{\text{ref}}$. The Jacobian of this warp is:

$$\nabla \mathcal{W}_v = \begin{bmatrix} -\Delta t & 0 \\ 0 & -\Delta t \end{bmatrix} \tag{3.8}$$

This warp function represents 2D motions of objects on the image plane, not the per-pixel optic flow of the entire scene. Thus for motions such as rotations a different warp may be appropriate.
b) The pure rotation warp (rotation of angular velocity ω_v around a point at (x_c, y_c)):

$$\mathcal{W}_{\text{rot}}(x, y, \Delta t; \omega) = \begin{bmatrix} \cos(\theta_{\omega_v}) & -\sin(\theta_{\omega_v}) \\ \sin(\theta_{\omega_v}) & \cos(\theta_{\omega_v}) \end{bmatrix} \begin{bmatrix} x - x_c \\ y - y_c \end{bmatrix} + \begin{bmatrix} x_c \\ y_c \end{bmatrix} \tag{3.9}$$

where the angle travelled is $\theta_{\omega_v} = -\omega_v \Delta t$, the warp parameters $\omega = (\omega_v, x_c, y_c)$ and Δt is the difference between the event timestamps and the reference time, $t - t_{\text{ref}}$. The Jacobian for this warp

is:

$$\nabla \mathcal{W}_{\text{rot}} = \begin{bmatrix} 1-\cos(\theta_{\omega_v}) & \sin(\theta_{\omega_v}) & (y_c - y)\cos(\theta_{\omega_v}) + (x_c - x)\sin(\theta_{\omega_v}) \\ -\sin(\theta_{\omega_v}) & 1-\cos(\theta_{\omega_v}) & (x - x_c)\cos(\theta_{\omega_v}) + (y_c - y)\sin(\theta_{\omega_v}) \end{bmatrix} \tag{3.10}$$

c) The 4-DoF warp from [58], a 4-DoF transform, parametrised by ω which consists of a 2D translation (v_x, v_y), followed by a motion toward the image plane v_z and a rotation θ_z around the z axis of the camera:

$$\mathcal{W}_{xyz\theta}(x, y, \Delta t; \omega) = \begin{bmatrix} x \\ y \end{bmatrix} - \Delta t \left(\begin{bmatrix} v_x \\ v_y \end{bmatrix} + (v_z + 1) \begin{bmatrix} \cos(\theta_z) & -\sin(\theta_z) \\ \sin(\theta_z) & \cos(\theta_z) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} - \begin{bmatrix} x \\ y \end{bmatrix} \right) \tag{3.11}$$

[58]. Since the rotation is preceded by translations and a radial expansion of the events, it is not necessary to explicitly parameterise for the center of rotation which is accounted for by the shifts. The Jacobian is:

$$\nabla \mathcal{W}_{xyz\theta} = \begin{bmatrix} -\Delta t & 0 & \Delta t(y\sin(\theta_z) - x\cos(\theta_z)) \\ 0 & -\Delta t & \Delta t(-x\sin(\theta_z) - y\cos(\theta_z)) \end{bmatrix} \tag{3.12}$$

3.2.8 Sequence Processing

The above method segments the events $\mathcal{E}$ from a short time interval T. To process an entire stream of events, a sliding window approach is used, splitting the stream into overlapping packets of events $\{\mathcal{E}_n\}_{n=1}^{N_e}$. We process the n-th packet and then slide the window, thus selecting more recent events. The motions estimated by clustering $\mathcal{E}_n$ can be propagated in time to predict an initialisation for the clusters of the next packet, $\mathcal{E}_{n+1}$. We use a fixed number of events N_e per window, and slide by half of it, $N_e/2$.

3.3 Experiments

Overview

In this section we first provide a quantitative evaluation of our method on a publicly available, real-world dataset [58], showing that we significantly outperform two baseline methods [58, 104]. We provide further quantitative results on the accuracy of our method with regard to *relative motion differences* and we demonstrate the efficacy of our method on additional, challenging real-world data. Throughout the experiments, we demonstrate several features of our method, namely that (*i*) it allows arbitrary motion models for different clusters, (*ii*) it allows us to perform motion segmentation on difficult scenes (high speed and/or HDR), where conventional cameras would fail, (*iii*) it is robust to the number of clusters used (N_l), and (*iv*) that it is able to perform motion segmentation on non-rigid scenes. The sequences considered cover a broad range of motion speeds, from 12 pixel/s to several hundred pixel/s. We strongly recommend looking at the accompanying video [1].

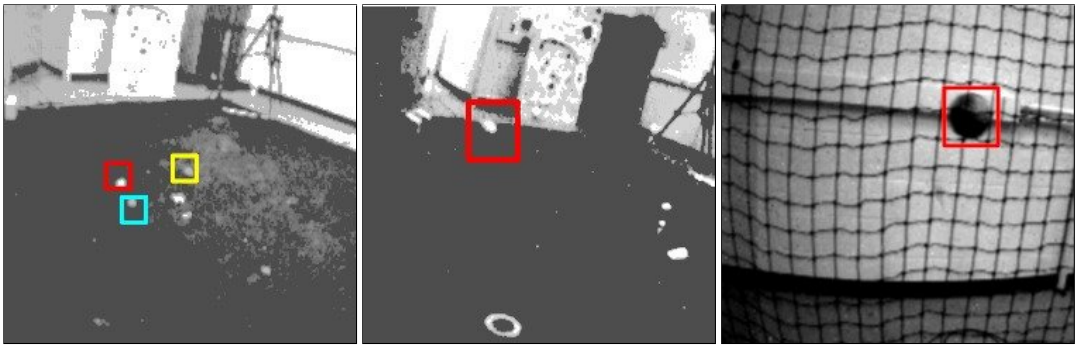
The following experiments were carried out with data from a Dynamic and Active-pixel VIsion Sensor (DAVIS) 240C camera [9], which provide both events and grayscale frames. The frames are not used in the experiments; they serve only an illustrative purpose.

3.3.1 Quantitative Evaluation

Results on Dataset from [58]

We ran our segmentation method on the Extreme Event Dataset (EED) from [58] and compared against the results from [58] and [104]. The sequences in the EED dataset showcase a range of scenes

[1] https://timostoff.github.io/19ICCV



(a) (b) (c)

Figure 3.3: Several scenes from the EED [58]: (a) `Multiple Objects`, (b) `Occluded Sequence` and (c) `What is Background?` Moving objects (drones, balls, etc.) are within hand-labeled bounding boxes. Images have been brightened for visualization.

Table 3.1: Comparison with SotA using the success rate proposed by [58] of detection of moving objects (in %). Note that there are some inconsistencies in the results posted by [58] (see Section 3.3.1). Success is recorded whenever the estimated bounding box overlaps at least 50 % with the hand-labelled one and it has more area within the hand-labelled bounding box than outside.

EED Sequence Name	**SOFAS** [104]	**Mitrokhin** [58]	**Ours (Alg. 1)**
`Fast moving drone`	88.89	92.78	**96.30**
`Multiple objects`	46.15	87.32	**96.77**
`Lighting variation`	0.00	**84.52**	80.51
`What is Background?`	22.08	89.21	**100.00**
`Occluded sequence`	80.00	90.83	**92.31**

(Figure 3.3 and Table 3.1), which are very challenging for conventional cameras. In particular, they comprise fast moving objects (around 600 pixel/s) in `Fast Moving Drone` and `Multiple Objects`, which are almost indiscernible on the frames provided by the DAVIS camera, as well as scenes with extreme lighting variations, such as `Lighting variation` (in which a drone is tracked despite a strobe light pointing at the camera), and object occlusions. Having object segmentation rather per-event segmentation in mind, the EED dataset provides timestamped bounding boxes around the moving objects in the scene and proposes to measure *object segmentation success* whenever the estimated bounding box overlaps at least 50 % with the hand-labelled one *and* it has more area within the hand-labelled bounding box than outside. To compare against [58], we perform motion segmentation on the events that occur around the timestamp of the bounding-box and count success if for a given cluster the above criterion is true for the segmented events. For a fair comparison, we used the same type of motion models (4-DoF warps) as in [58].

As a side-note, the reliability of the benchmark results in [58] is somewhat unclear, since the posted results are not always within the set of possible results. For example, for the sequence `Lighting variation`, the paper claims to have a score of 84.52 % and the accompanying dataset contains 76 bounding boxes. Since the score is calculated by $\frac{\text{number correct bboxes}}{\text{total bboxes}} = \text{score}$, it follows that for `Lighting variation`, $\frac{x}{76} = 0.8452$ and that x should be an integer value, which it is not ($x = 64.24$). The same goes for all of the remaining sequences. Unfortunately, when contacted the authors were not able to clear up this discrepancy.

Table 3.1 reports the results of the comparison of our method against [104] and [58]. Our method outperforms [104] in all sequences by a large margin (from 7.41 % to 84.52 %), and improves over [58] in all but one sequence, where it has comparable performance. In four out of five sequences

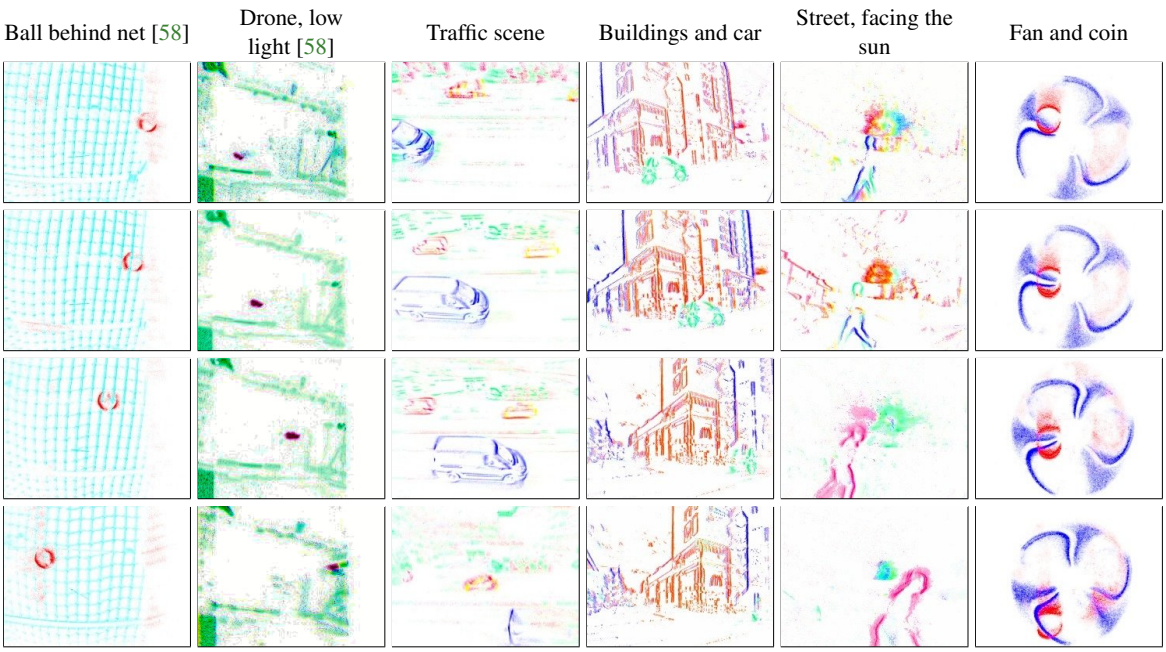


Figure 3.4: From top to bottom: snapshots (motion-compensated images, as in Figure 2) of events segmented into clusters on multiple sequences (one per column). Events coloured by cluster membership. Best viewed in accompanying video.

we achieve accuracy above 92 %, and in one of them, a perfect score, 100 %. Some results of the segmentation are displayed on the first columns of Figure 3.4. In the `What is Background?` scene (1st column of Figure 3.4), a ball is thrown from right to left behind a net while the camera is panning, following the ball. Our method clearly segments the scene into two clusters: the ball and the net, correctly handling occlusions. In the `Lighting variation` (2nd column of Figure 3.4), a quadrotor flies through a poorly lit room with strobe lightning in the background, and our method is able to segment the events due to the camera motion (green and blue) and due to the quadrotor (purple).

Accuracy vs Displacement

While the dataset from [58] provides a benchmark for comparison against the SotA, it does not allow us to assess the *per-event* accuracy of our method. Here we measure segmentation success directly as a percentage of correctly classified events, thus much more fine-grained than with bounding boxes. Since we wish to isolate our results from the noise which is usually present in real event data, we perform the quantitative analysis on event data from a SotA photorealistic simulator [85] (note that our other experiments are all on real sequences, some abnormally noisy, such as `Lighting variation`). Knowing which objects generated which events, allows us to finely resolve the accuracy of our method.

However, segmentation accuracy is closely coupled with the observation window over which events are collected. Intuitively, this makes sense; observing two objects with a relative velocity of 1 pixel/s for only 1 s means that the objects have moved only 0.1 pixels relative to each other, a difference that is difficult to measure. Observing these two objects for 10 s means a relative displacement of 10 pixels, which is easier to distinguish. Thus, the segmentation problem becomes easier to solve the larger the observation window is. Larger observation windows however add latency and can violate the assumptions of linear motion w.r.t. the motion model which FO based algorithms usually make. Therefore we feel it is important to show how a given method performs w.r.t. relative displacement.

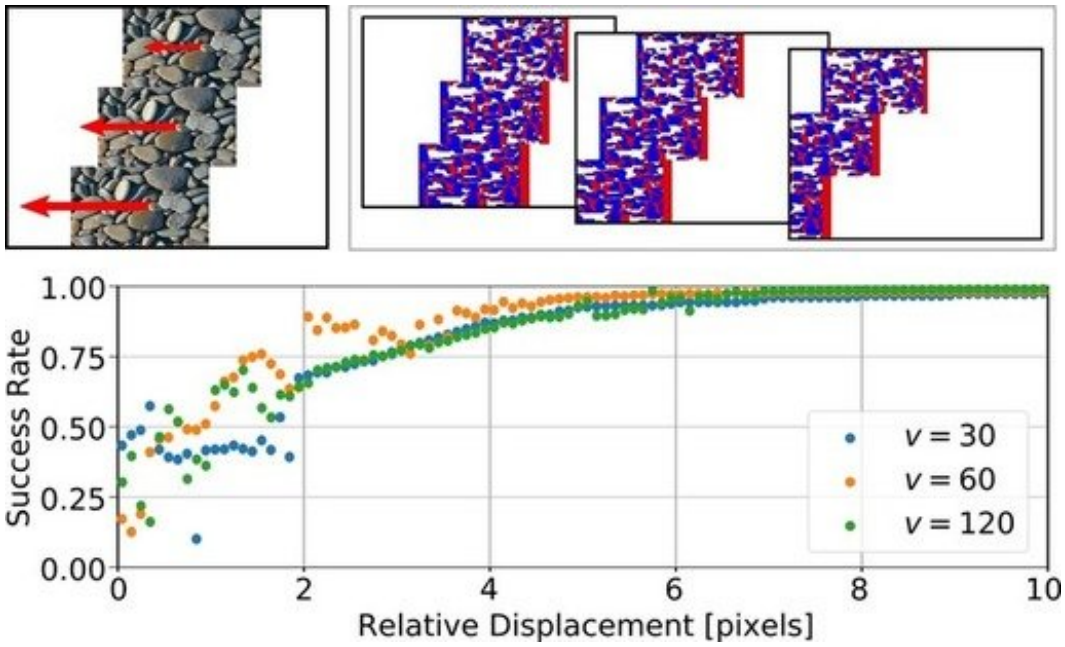


Figure 3.5: *Per-event Success Evaluation.* Segmentation accuracy vs relative object displacement on `pebbles` sequence. *Top:* A textured image (pebbles) is slid from right to left at various velocities. The individual images intersect so as to prevent morphological clustering. Top left: image, top right: successive event images (blue indicates negative, red positive events). *Bottom:* The success rate of our method w.r.t. the relative displacement of the pebble images at various relative velocities $v_{\text{rel}} = \{30, 60, 120\}\text{pixel/s}$). Our method requires around 4 pixels of relative displacement for 90 % segmentation accuracy.

Figure 3.5 evaluates the above effect on a sequence consisting of textured pebbles moving with different relative velocities. The plot of success rate vs relative displacement for various velocities shows that as the relative displacement increases, the proportion of correctly classified events, and therefore, the segmentation accuracy, increases. Our method requires that roughly 4 pixels of relative displacement have occurred in order to achieve 90 % accuracy. We show the result across a range of velocities to emphasise that our evaluation depends on relative displacement rather than absolute motion.

Computational Performance

The complexity of Algorithm 1 is linear in the number of clusters, events, pixels of the IWE and the number of optimisation iterations, in total, $\mathcal{O}((N_e + N_p)N_l N_{\text{it}})$. Our method generally converges in less than ten iterations of the algorithm, although this clearly depends on several attributes of the data being processed. Here, we provide a ballpark figure for the processing speed. We ran our method on a single core, 2.4 GHz Central Processing Unit (CPU) where we got a throughput of 240 000 events/s for optic flow type warps (Table 3.2). Almost 99 % of the time was spent in warping, which is in agreement with the performance analysis in [58]. As they point out, this operation can be done in parallel; they achieved a speedup of 1000× using a Graphics Processing Unit (GPU) implementation. While we were not able to replicate this result, we were able to achieve five times greater speed-ups on a Quadro K620 GPU (Table 3.2). The bottleneck during profiling was clearly memory transfer between the GPU memory and streaming multiprocessors. Using hardware that sits closer to memory, such as an Field Programmable Gate Array (FPGA), would potentially give much greater speedups. Using a GeForce 1080 GPU, we achieved a 10× speed-up factor, as reported in Table 3.2. The bottleneck is not in computation but rather in memory transfer to and from the GPU.

Throughput decreases as N_l increases, since all of the events need to be warped for every extra cluster in order to generate motion-compensated images. Further, extra clusters add to the dimensionality of the optimisation problem that is solved during the motion-compensation step.

Table 3.2: Throughput in kilo-events per second (optic flow motion) of Algorithm 1 running on a single CPU core vs GPU for varying N_ℓ (using the test sequence in Figure 3.7).

N_ℓ	**CPU** [kevents/s]	**GPU** [kevents/s]
2	239.86	3963.20
5	178.19	1434.66
10	80.93	645.02
20	32.43	331.50
50	12.62	113.78

Regardless of throughput, our method allows exploiting key features of event cameras, such as its very high temporal resolution and its HDR, as shown in experiments on the EED dataset (Table 3.1) and on some sequences in Figure 3.4 (Vehicle facing the sun, Fan and coin).

3.3.2 Further Real-World Sequences

We test our method on additional sequences in a variety of real-world scenes, as shown in Figure 3.4. The third column shows the results of segmenting a traffic scene, with the camera placed parallel to a street and tilting upwards while vehicles pass by in front of it. The algorithm segmented the scene into four clusters: three corresponding to the vehicles (blue, red, yellow) and another one corresponding to the background buildings (ego-motion, in green). Even the cars going in the same direction are separated (red and yellow), since they travel at different speeds.

The fourth column of Figure 3.4 shows the results of segmenting a scene with a car and some buildings while the camera is panning. We ran the algorithm to segment the scene into three clusters using optical flow warps. One cluster segments the car, and the other two clusters are assigned to the buildings. Note that the algorithm detects the differences in optical flow due to the perspective projection of the panning camera: it separates the higher speed in the periphery (blue) from the slower speed in the image center (red).

An HDR scene is shown on the fifth column of Figure 3.4. The camera is mounted on a moving vehicle facing the sun (central in field of view) while a pedestrian and a skateboarder cross in front of it. The camera's forward motion causes fewer events from the background than in previous (panning) examples. We run the segmentation algorithm with six clusters, allowing the method to adapt to the scene. Segmented events are coloured according to cluster membership. The algorithm correctly segments the pedestrian and the skateboarder, producing motion-compensated images of their silhouettes despite being non-rigidly moving objects. Note that none of the previous scenes have a constant depth, since the depth of the objects changes as the camera moves.

Finally, the last column of Figure 3.4 shows the versatility of our method to accommodate different motion models for each cluster. To this end, we recorded a coin dropping in front of the blades of a ventilator spinning at 1800 rpm. In this case the fan is represented by a rotational motion model and the coin by a linear velocity motion model. Our method converges to the expected, optimal solution, as can be seen in the motion-compensated images, and it can handle the occlusions on the blades caused by the coin.

Figure 3.6 shows that our method also works with a higher resolution (640×480 pixels) event-based camera [100].

3.3.3 Sensitivity to the Number of Clusters

The following experiment shows that our method is not sensitive to the number of clusters chosen N_l. We found that N_l is not a particularly important parameter; if it chosen to be too large, the unnecessary clusters end up not having any events allocated to them and thus mode collapse. This is a nice feature, since it means that in practice N_l can simply be chosen to be large and then not be worried about. We demonstrate this on the `slider_depth` sequence from [64]; where there are multiple objects at different depths (depth continuum), with the camera sliding past them. Because of parallax, this results in a continuum of image plane velocities and thus infinitely many clusters would in theory be needed to segment the scene with an optical flow motion-model. Thus, the sequence can only be resolved by adding many clusters which discretise the continuum of velocities.

Figure 3.7 demonstrates that our method is robust with regard to the number of clusters chosen (in Figure 3.7b–3.7d); too few clusters and the method will simply discretise the event cluster continuum, too many clusters and some clusters will mode collapse, i.e. no events will be assigned to them. By segmenting with enough clusters and preventing cluster collapse, our method can be used to detect depth variations; nevertheless, tailored methods for depth estimation [84] are more suitable for such a task. The experiment also shows that our method deals with object occlusions.

Similarly, Figure 3.7 shows that our method is not sensitive to the mixture of motion models either. Figure 3.7e shows the result with five clusters of optical flow type and five clusters of rotation type. As can be seen, our method essentially allocates no event likelihoods to these rotation models clusters, which clearly do not suit any of the events in this sequence. Figure 3.7f shows the result of using only rotation motion models, resulting in failure, as expected. As future work, a meta-algorithm could be used to select which motion models are most relevant depending on the scene.

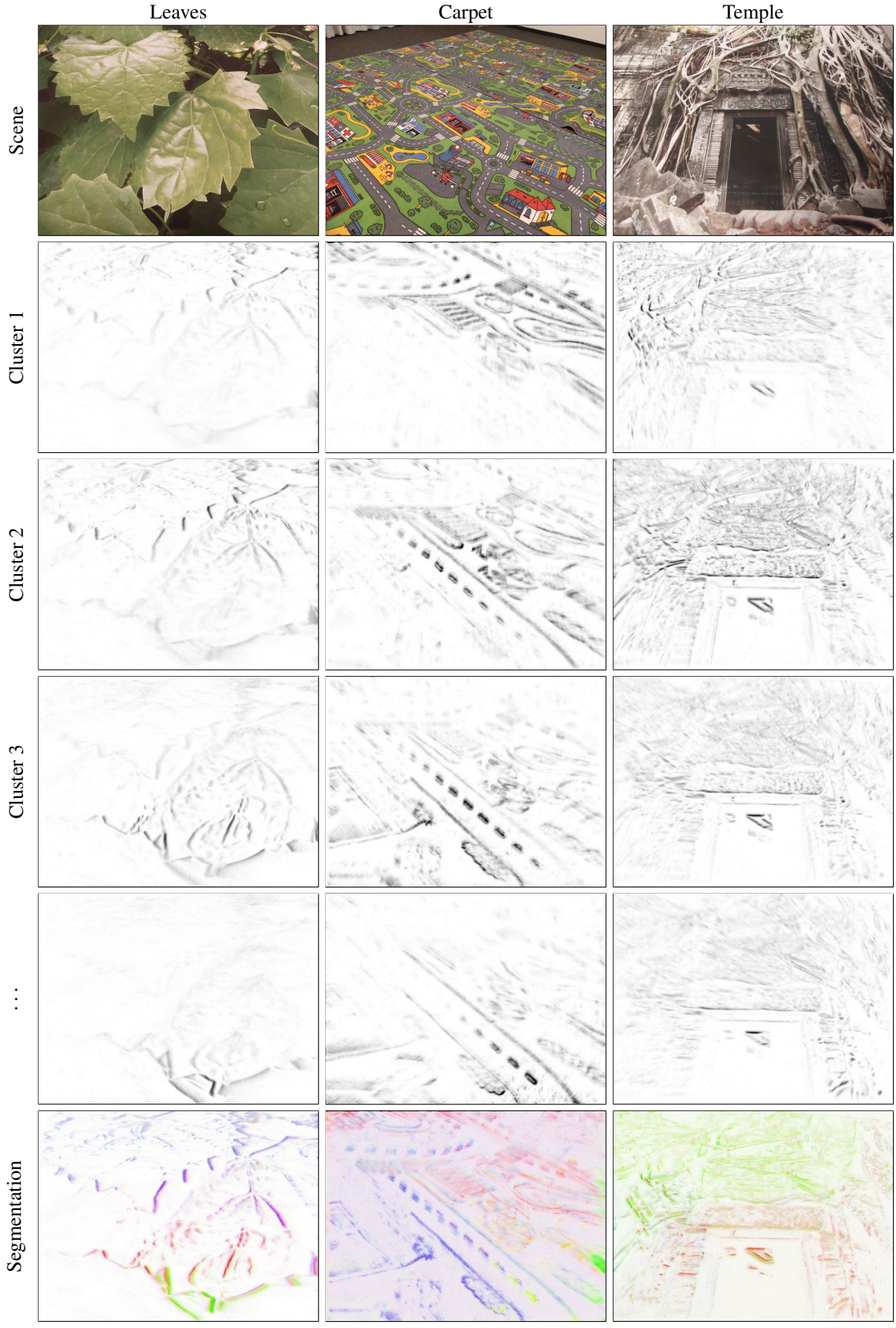


Figure 3.6: Scenes recorded with a Samsung Dynamic Vision Sensor (DVS) Gen3 event camera (640×480 pixels); algorithm run with ten clusters ($N_\ell = 10$). From top to bottom: scene recorded, four of the clusters (motion-compensated IWEs, with darkness indicating event likelihoods), and the accustomed coloured segmentation (as in Figure 3.2). These examples illustrate that our method can be used to segment the scene according to depth from the camera, although it is not its main purpose.

Continuous Depth Variation

In this experiment, we essentially show a scene similar to that in Figure 3.7. The difference is that the scene in Figure 3.8 shows a truly continuous depth variation. As can be seen in the results (using $N_l = 15$), our method discretises the segmentation, although it is noteworthy that each 'slice' of depth appears to fade toward foreground and background. This is because the method becomes less certain of the likelihood of events that sit between clusters, the darkness of a region reflecting the likelihood of a given event belonging to that cluster.

Due to the recent development of new, high resolution event-based cameras [100], we show the results of our method on the output of a Samsung DVS Gen3 sensor, with a spatial resolution of 640×480 pixels. In this experiment, we show the segmentation of several scenes (a textured carpet, some leaves and a temple poster) as the camera moves. Due to ego-motion induced parallax, there is a continuous gradient in the motion in the scene, i.e. the scenes present a continuum of depths. As can be seen in Figure 3.6, our method works the same on high-resolution data.

3.3.4 Non-rigid Moving Objects

In the following experiments, we show how our method deals with non-rigid objects. Our algorithm warps events $\mathcal{E}$ according to point-trajectories described by parametric motion models whose parameters are assumed constant over the (small) time Δt spanned by $\mathcal{E}$. Low-dimensional parametric warp models, such as the patch-based optic flow (2-DoF, linear trajectories), rotational motion in the plane (1-DoF) or in space (3-DoF) are simple and produce robust results by constraining the dimensionality of the solution space in the search for optimal point-trajectories. However, simple warp models (both in event-based vision or in traditional frame-to-frame vision) have limited expressiveness: they are good for representing rigidly moving objects, but do not have enough degrees of freedom to represent more complex motions, such as deformations (e.g. pedestrian, birds, etc.). One could consider using warps able to describe more complex motions, such as part-based warp models [78] or infinite-dimensional models [112]. But this would make the segmentation problem considerably harder, not only due to the increased dimensionality of the search space, but also because it could potentially contain multiple local minima. Even though we do not explore more complex models to segment such scenes, we are able to achieve acceptable results (although not as good as those on non-rigid objects).

Pedestrian

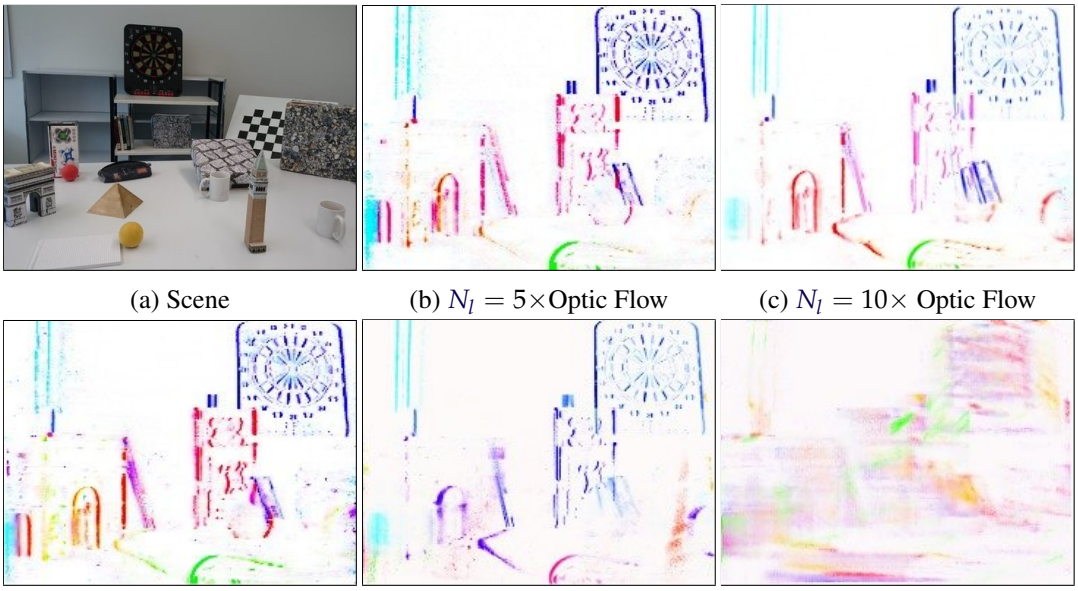
Figure 3.9 shows a pedestrian walking past the camera while it is panning. In spite of using simple warp models, our method does a good job at segmentation: the background (due to camera motion), the torso of the person and the swinging arms are segmented in separate clusters. This is so, because during the short time span of $\mathcal{E}$ (in the order of milliseconds), the objects move approximately rigidly.

Popping Balloon

In order to test the limits of this assumption, we recorded the popping of a balloon with the event camera (see Figure 3.10). While the segmentation struggles to give a clear result in the initial moments of puncturing (3.10b), it still manages to give reasonable results for the fast moving, contracting fragments of rubber flung away by the explosion (3.10c, 3.10d).

3.4 Additional Motion-Compensation Segmentation Methods

The similarity of our method to classical EM asks the question of how well similar clustering methods would perform. In this section, we describe how two classical clustering methods (mixture densities



(a) Scene (b) $N_l = 5\times$Optic Flow (c) $N_l = 10\times$ Optic Flow

(d) $N_l = 20\times$c Flow (e) $N_l = 5\times$ OF + $5\times$ Rotation (f) $N_l = 10\times$Rotation

Figure 3.7: *Experiment on* `slider_depth` *sequence* of [64]. Motion-compensated images in 3.7b to 3.7f show events coloured by cluster. Using Optic Flow (OF) warps (in 3.7b–3.7d), event clusters correspond to depth planes with respect to the camera. Using as few as five clusters, the events are discretised and approximately spread over the depth continuum. Using 5, 10 or 20 clusters in 3.7b, 3.7c, 3.7d gives very similar results, showing that our method is not sensitive to the value of N_l chosen. Adding clusters with motion models that do not suit the motion, as in 3.7e, where five clusters are pure rotational warps, does not disturb the output either as those motions mode collapse. For reasons why the OF warp fails in this situation, see Section 3.2.7.

and fuzzy k-means) can be modified to tackle the task of event-based motion segmentation, by leveraging the idea of FO (Section 3.4.1 and 3.4.3). Examples comparing the three per-event segmentation models developed (Algorithm 1 to 3) are given in Section 3.4.5, referred to as *proposed* (or *Layered*), *Mixture Densities* and *Fuzzy k-Means*, respectively.

3.4.1 Mixture Densities

The mixture models framework [7, 19] can be adapted to solve the segmentation problem addressed. The idea is to fit a mixture density to the events $\mathcal{E}$, with each mode representing a cluster of events with a coherent motion.

3.4.2 Problem Formulation

Specifically, following the notation in [19, Ch.10], we identify the elements of the problem: the data points are the events $\mathcal{E}$ without taking into account polarity; thus, feature space is the volume V_e, and, consequently, the clusters are comprised of events (i.e. they are not clusters of optic flow vectors in velocity space).

The mixture model states that events $e_k \in V_e$ are distributed according to a sum of several distributions (clusters), with mixing weights (cluster probabilities) $\boldsymbol{\pi} = \{\mathbf{P}(\psi_j)\}_{j=1}^{N_l}$:

$$p(e_k|\omega) = \sum_{j=1}^{N_l} p(e_k|\psi_j, \omega_j)\mathbf{P}(\psi_j), \tag{3.13}$$

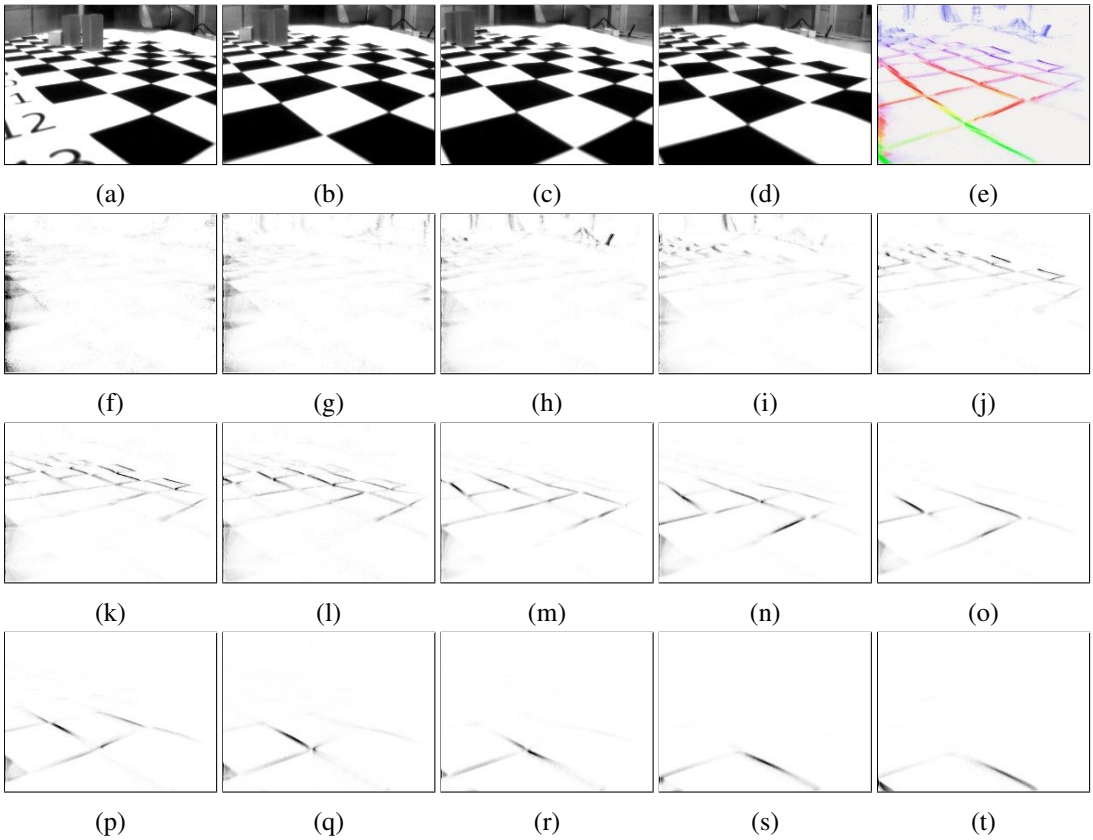


Figure 3.8: Sequence from a camera translating past a checkerboard (3.8a-3.8d). These grayscale frames, provided by the DAVIS [9] are not used by our method; they are just for visualisation purposes. Each image in 3.8f-3.8t shows the IWE of each cluster (15 clusters, optical flow motion models). 3.8e shows the segmented output (combined IWEs) in the accustomed coloured format.

where $\omega = \{\omega_j\}_{j=1}^{N_l}$ are the parameters of the distributions of each component of the mixture model and we assumed that the parameters of each cluster are independent of each other: $p(e_k|\psi_j,\omega) = p(e_k|\psi_j,\omega_j)$. The function $p(z|\omega)$ in (3.13), with $z \in V_e$, is a scalar field in V_e representing the density of events in V_e as a sum of several densities, each of them corresponding to a different cluster, and each cluster describing a coherent motion.

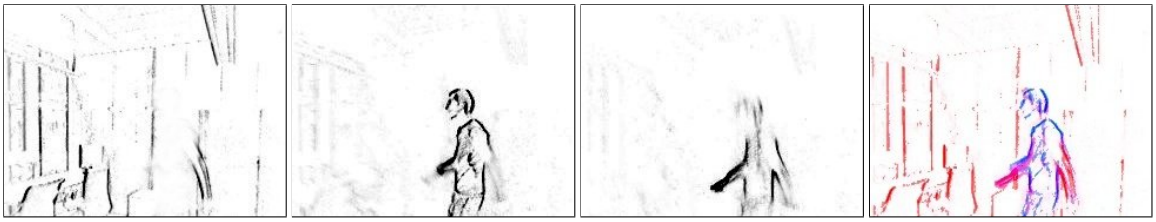
To measure how well the j-th cluster explains an event (3.13), we propose to use the unweighted IWE (Appendix A.1.3), to which the probability of z (given ψ_j,ω_j) is proportional:

$$p(z \,|\, \psi_j,\omega_j) \propto H_j\left(\mathbf{x}'(z;\omega_j)\right) \tag{3.14}$$

$$H_j\left(\mathbf{x}\right) = \sum_{m=1}^{N_e} \delta\left(\mathbf{x} - \mathbf{x}'_{mj}\right) \tag{3.15}$$

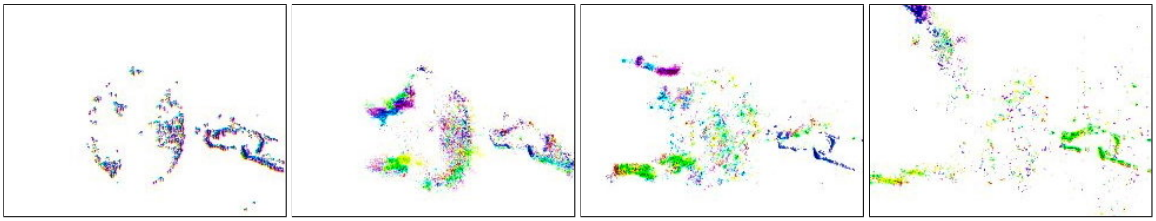
with warped event location $\mathbf{x}'_{mj} = \mathcal{W}(\mathbf{x}_m, t_m; \omega_j)$. The image point $\mathbf{x}'(z;\omega_j)$ corresponds to the warped location of point $z \in V_e$ using the motion parameters of the j-th cluster.

Notice that the choice (3.14) causes the distribution of each component in the mixture $p(z \,|\, \psi_j,\omega_j)$ to be constant along the point trajectories defined by the warping model of the cluster, which agrees with the 'tubular' shape mentioned in the problem statement (Section 3.2.1). The mixture model (3.13) may not be constant along point trajectories since it is a weighted sum of several distributions, each with its own point trajectories.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) All clusters (merged IWEs)

Figure 3.9: *Non-Rigid Scene.* A person walks across a room, arms swinging. The room 3.9a, the body 3.9b and the arms 3.9c are segmented out, with greater uncertainty to the event associations in areas of deformation (such as elbows), visible in the fact that events are associated to both clusters (events coloured by cluster in 3.9d).



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

Figure 3.10: *Non-Rigid Moving Objects.* From left to right: snapshots of segmentation of balloon popping. Run with $N_l = 4$ clusters, events coloured by cluster membership.

Algorithm 2 Event-based Motion Segmentation using Mixture Density Model

1: **Input**: events $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$ in a space-time volume V of the image plane, and number of clusters N_l.
2: **Output**: cluster parameters $\omega = \{\omega_j\}_{j=1}^{N_l}$ and mixing weights $\boldsymbol{\pi} \doteq \{P(\psi_j)\}_{j=1}^{N_l}$.
3: **Procedure:**
4: Initialise ω and $\boldsymbol{\pi}$.
5: **Iterate** until convergence:
6: • Update the mixing weights (3.17), using the current motion parameters ω and the mixing weights from the previous iteration in (3.18).
7: • Update motion parameters ω by ascending on (3.16).

Iterative Solver: EM Algorithm

With the above definitions, it is possible to apply the EM algorithm in [19, Ch.10] to compute the parameters of the mixture model, by maximising the (log-)likelihood of the mixture density:

$$(\omega^*, \boldsymbol{\pi}^*) = \arg\max_{(\omega,\boldsymbol{\pi})} \sum_{k=1}^{N_e} \log\, p(e_k|\omega) \tag{3.16}$$

subject to the choice of warping models.

In the E-step, the mixing weights $\boldsymbol{\pi}$ are updated using

$$\mathbf{P}(\psi_j) = \frac{1}{N_e} \sum_{k=1}^{N_e} p(\psi_j|e_k, \omega_j) \tag{3.17}$$

with membership probabilities given by the Bayes formula

$$p(\psi_j|e_k,\omega_j) = \frac{p(e_k|\psi_j,\omega_j)\mathbf{P}(\psi_j)}{\sum_{i=1}^{N_l} p(e_k|\psi_i,\omega_i)\mathbf{P}(\psi_i)}. \tag{3.18}$$

In the M-step, gradient ascent or conjugate gradient [71] of the log-likelihood (3.16) with respect to the warp parameters ω is used to update ω, in preparation for the next iteration.

From the mixing weights and the motion parameters, it is straightforward to compute the event-cluster assignment probabilities using (3.18). To initialise the iteration, we use the procedure described in Section 3.2.5.

Notice that, during the EM iterations, the above method not only estimates the cluster parameters ω and the mixing weights $\boldsymbol{\pi}$ but also the distributions $p(z|\psi_i,\omega_i)$ themselves, i.e. the 'shape' of the components of the mixture model. These distributions get sharper (more peaky or 'in focus') around the segmented objects as iterations proceed, and blurred around the non-segmented objects corresponding to that cluster. An example is given in Section 3.4.5.

3.4.3 Fuzzy k-Means

Event-based motion segmentation can also be achieved by designing an objective function similar to the one used in the fuzzy k-means algorithm [19, Ch.10].

Problem Formulation

This approach seeks to maximise

$$(\omega^*,\mathbf{P}^*) = \arg\max_{\omega,\mathbf{P}} \sum_{j=1}^{N_l}\sum_{k=1}^{N_e} p_{kj}^b d_{kj}, \tag{3.19}$$

where $b > 1$ (e.g. $b = 2$) adjusts the blending of the different clusters, and the goodness of fit between an event e_k and a cluster j in V_e is given in terms of event alignment (i.e. 'sharpness'):

$$d_{kj} = \log H_j(\mathbf{x'}_{kj}), \tag{3.20}$$

the value of the unweighted IWE (3.15) at the warped event location using the motion parameters of the cluster. We use the logarithm of the IWE, as in (3.16), to decrease the influence of large values of the IWE, since these are counted multiple times if the events are warped to the same pixel location. Notice that (3.19) differs from (3.1)-(3.4): the responsibilities p_{kj} appear multiplying the IWE (i.e. they are not included in a weighted IWE), and the sum is over the events (as opposed to over the pixels (3.3)).

Notice also that this proposal is different from clustering in optic flow space (Figure 3.15). As mentioned in Section 3.4.1, here the feature space is the space-time volume $V_e \in \mathbb{R}^3$ (i.e. event location), rather than the optic flow space ($\mathbb{R}^2$) (i.e. event velocity).

3.4.4 Iterative Solver: EM Algorithm

The EM algorithm may also be used to solve (3.19). In the E-step (fixed warp parameters ω) the responsibilities are updated using the closed-form partitioning formula

$$P_{kj} = d_{kj}^{\frac{1}{b-1}} \Big/ \sum_{i=1}^{N_l} d_{ki}^{\frac{1}{b-1}}. \tag{3.21}$$

Algorithm 3 Event-based Motion Segmentation using the Fuzzy k-Means Method

1: **Input**: events $\mathcal{E} = \{e_k\}_{k=1}^{N_e}$ in a space-time volume V_e of the image plane, and number of clusters N_l.
2: **Output**: cluster parameters $\omega = \{\omega_j\}_{j=1}^{N_l}$ and event-cluster assignments $P \equiv p_{kj} \doteq P(e_k \in \ell_j)$.
3: **Procedure:**
4: Initialisation (as in Section 3.2.5).
5: **Iterate** until convergence:
6: • Update the event-cluster assignments p_{kj} using (3.21).
7: • Update motion parameters ω by ascending on (3.19).

In the M-step (fixed responsibilities) the warp parameters of the clusters ω are updated using gradient ascent or conjugate gradient. The pseudo-code of the event-based fuzzy k-means segmentation method is given in Algorithm 3.

3.4.5 Comparison of Three Motion-Compensation Segmentation Methods

We compare our method with the two above-mentioned methods (Sections 3.4.1 and 3.4.3) that we also designed to leverage motion-compensation.

Figure 3.11 shows the comparison of the three methods on a toy example with three objects (a filled pentagon, a star and a circle) moving in different directions on the image plane. In the mixture density and fuzzy k-means methods, the motion-compensated IWEs do not include the event-cluster associations $\mathbf{P}$, and so, all objects appear in all IWEs, sharper in one IWE than in the others. In contrast, in the proposed method (Algorithm 1), the associations are included in the motion-compensated image of the cluster (weighted IWE), as per equation (3.1), and so, the objects are better split into the clusters (with minor ghosting effects, as illustrated in Figure 3.2), thus yielding the best results.

It is worth mentioning that the three per-event segmentation methods are novel: they have not been previously proposed in the literature.

3.5 Computational Complexity

Next, we analyse the complexity of the three segmentation methods considered (Algorithm 1 to 3), defined by objective functions (3.4), (3.16) and (3.19), respectively. The core of the segmentation methods is the computation of the images of warped events (IWEs (3.1) or (3.15); one per cluster), which has complexity $\mathcal{O}(N_e N_l)$.

3.5.1 Proposed (Layered) Model

The complexity of updating the event assignments using (3.6) is essentially that of computing the (weighted) IWEs of all clusters, i.e. $\mathcal{O}(N_e N_l)$. The complexity of computing the contrast (3.3) of a generic image is linear in the number of pixels, $\mathcal{O}(N_p)$, and so, the complexity of computing the contrast of one IWE is $\mathcal{O}(N_e + N_p)$. The computation of the contrast is negligible compared to the effort required by the warp. Computing the contrast of N_l clusters (corresponding to a set of candidate parameters) has complexity $\mathcal{O}\big((N_e + N_p)N_l\big)$. Since multiple iterations N_{it} may be required to find the optimal parameters, the total complexity of the iterative algorithm used is $\mathcal{O}\big((N_e + N_p)N_l N_{\text{it}}\big)$.

3.5.2 Mixture Density Model

The complexity of updating the mixture weights is that of computing the posterior probabilities $p(\psi_j | e_k, \boldsymbol{\theta}_j)$, which require computing the IWEs of all clusters, i.e. complexity $\mathcal{O}(N_e N_l)$. The

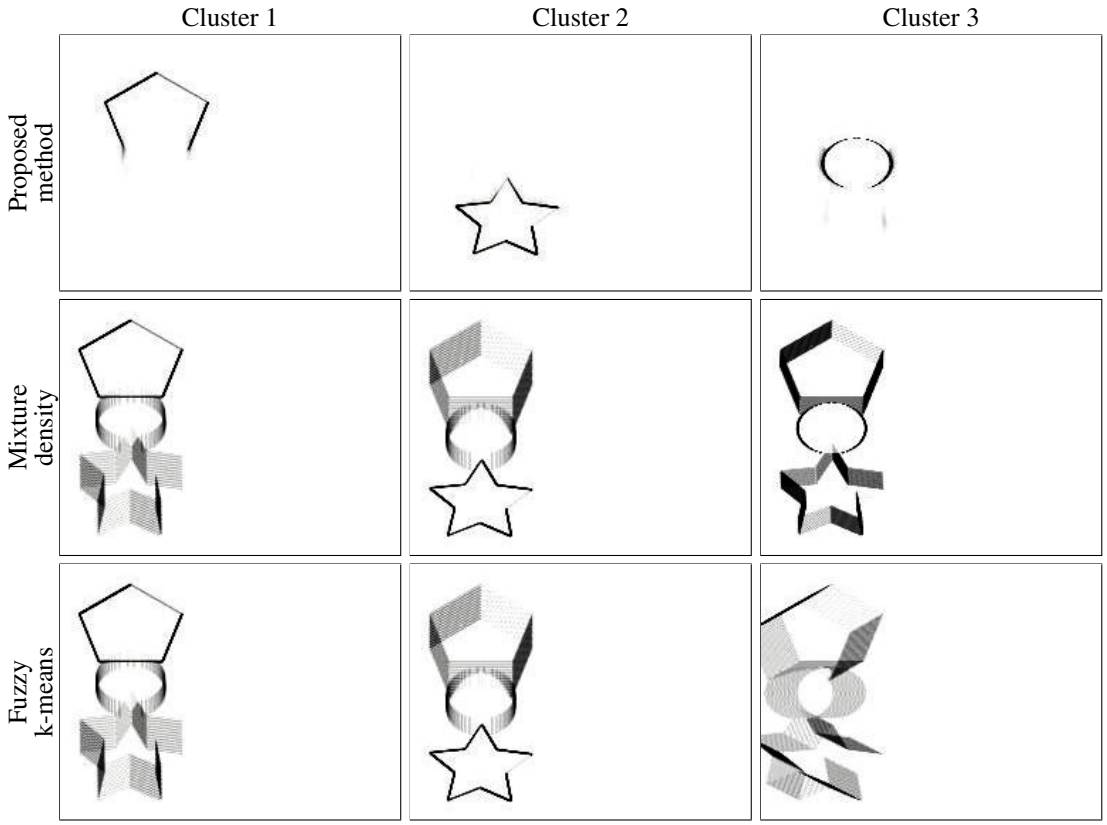


Figure 3.11: Comparison of three methods for event-based Motion Segmentation: Algorithms 1 to 3 (one per row).

complexity of updating the motion parameters is also that of computing the contrasts of the IWEs of all clusters, through multiple ascent iterations. In total, the complexity is $\mathcal{O}\big((N_e + N_p)N_l N_{\text{it}}\big)$.

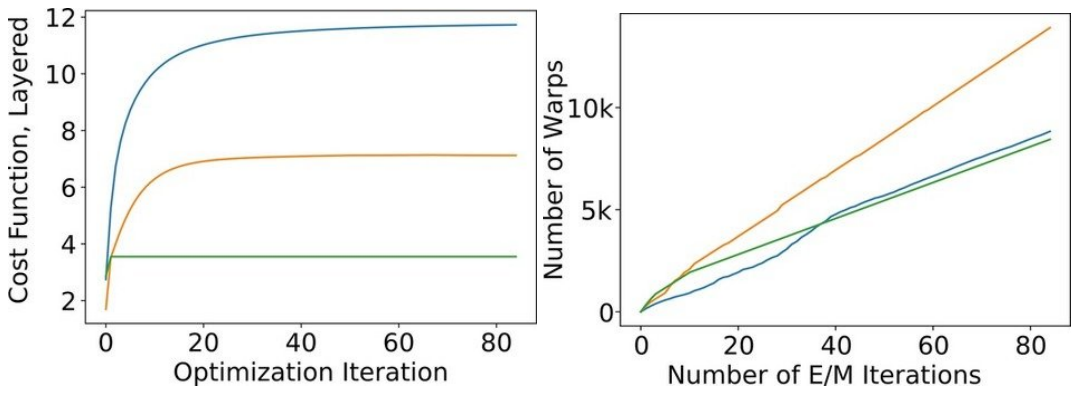
3.5.3 Fuzzy k-means Model

The complexity of computing the responsibilities (3.21) is that of computing N_l IWEs (values d_{kj}), i.e. $\mathcal{O}(N_e N_l)$. The complexity of updating the motion parameters is that of computing the objective function (3.19), $\mathcal{O}(N_e N_l)$, through multiple iterations. In total, the complexity is $\mathcal{O}(N_e N_l N_{\text{it}})$.

Figure 3.12 shows the convergence of the three above methods on real data from a traffic sequence that is segmented into four clusters (Figure 3.13 and third column of Figure 3.4): three cars and the background due to ego-motion. The top plot, Figure 3.12a, shows the evolution of the sum-of-contrasts objective function (3.4) vs the iterations. All methods flatten out after $\approx$ 20 iterations, and, as expected, the proposed method provides the highest score among all three methods (since it is designed to maximise this objective function). The Mixture model and Fuzzy k-means methods do not provide such a large score mostly due to the event-cluster associations, since they are not as confident to belonging to one cluster as in the proposed method. Figure 3.12b displays the number of warps (i.e. number of IWEs) that each method computes as the optimisation iterations proceed; as it can be shown, the relationship is approximately linear, with the proposed method performing the least warps for a considerable number of iterations, before flattening out (Figure 3.12a).

3.6 Comparison to k-means Optic Flow Clustering

Finally, the following experiments show the comparison of our method against k-means clustering of optic flow. We first illustrate the difference with a qualitative example and then quantitatively show



(a) (b)

Figure 3.12: *Comparison of three Methods.* We compare the convergence properties of three motion-compensated event-segmentation methods (Algorithms 1 to 3): proposed (layered) method (*blue*), Mixture Density Model (*orange*) and Fuzzy k-Means (*green*). Data used is from the Traffic Sequence (third column of Figure 3.4), the warped events at each iteration are visualised in Figure 3.13.

the ability of our method to resolve small differences in velocities compared to k-means. To this end, we use an event-based camera mounted on a motorised linear slider, which provides accurate ground truth position of the camera. Since the camera moves at constant speed in a 1-D trajectory, the differences in optical flow values observed when viewing a static scene are due to parallax from the different depths of the objects causing the events.

Numbers Sequence

In this experiment, we placed six printed numbers at different, known depths with respect to the linear slider. The event-based camera moved back and forth on the slider at approximately constant speed. Due to parallax, the objects at different depths appear to be moving at different speeds; faster the closer the object is to the camera. Thus we expect the scene to be segmented into six clusters, each corresponding to a different apparent velocity.

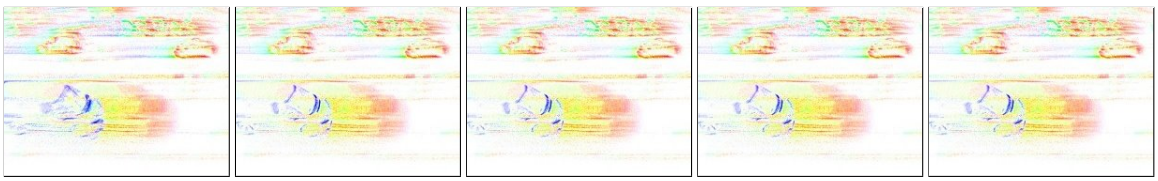
Figure 3.14 compares the results of k-means clustering optic flow and Algorithm 1. To compute optical flow we use conventional methods on reconstructed images at a high frame rate [94], with the optical flow method in [20] producing better results on such event-reconstructed images than state-of-the-art learning methods [106]. The results show that the velocities corresponding to the six numbers are too similar to be resolved correctly by the two-step approach (flow plus clustering), as evidenced by the bad segmentation of the scene (numbers 3, 4 and 5 are clustered together, whereas three clusters are used to represent the events of the fastest moving number–the zero, closest to the camera). In contrast, our method accurately clusters the events according to the motion of the objects causing them, in this case, according to velocities, since we used an optical flow warp (linear motion on the image plane). The higher accuracy of our method is easily seen in the sharpness of the motion-compensated images (cf. Figure 3.14d and Figure 3.14f).

Rocks at Different Speeds

We also tested our method on two real sequences with six objects of textured images of pebbles (similar to Figure 3.5), in which the relative velocities of the objects were $50\,\text{pixels/s}$ and $6\,\text{pixels/s}$, respectively. Figure 3.15 shows the results. If the objects are moving with sufficiently distinct velocities (Figure 3.15a), the clusters can be resolved by the two-step approach. However, once the objects move with similar velocities (Figure 3.15a), k-means clustering of optical flow is unable to

(a) Proposed, iter = 1 (b) iter = 5 (c) iter = 10 (d) iter = 20 (e) iter = 80

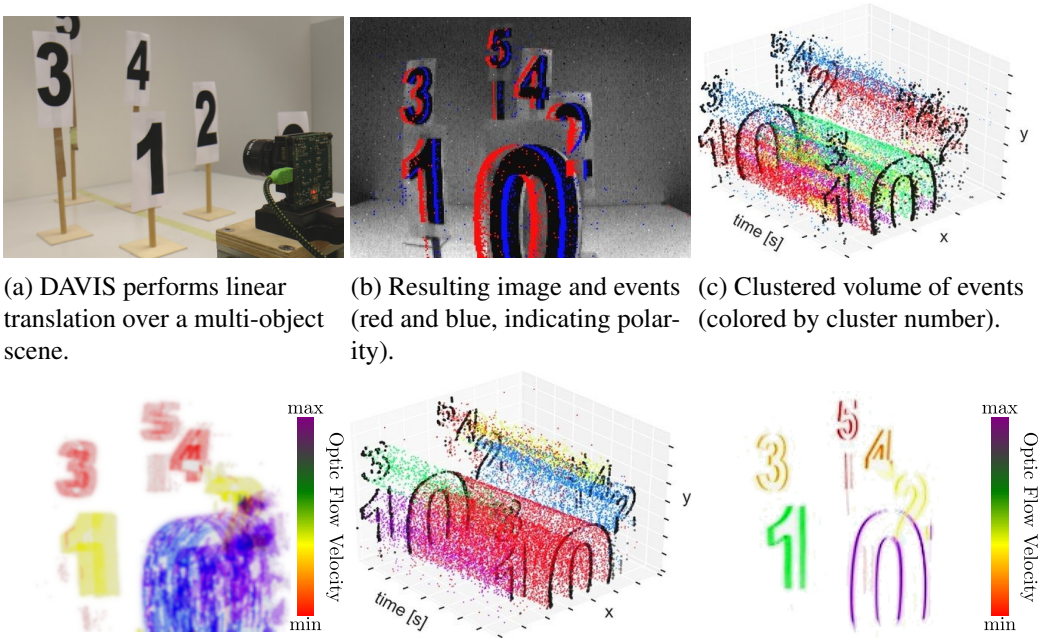
(f) Mixture Density, iter = 1 (g) iter = 5 (h) iter = 10 (i) iter = 20 (j) iter = 80



(k) Fuzzy k-Means, iter = 1 (l) iter = 5 (m) iter = 10 (n) iter = 20 (o) iter = 80

Figure 3.13: Images of the motion-corrected events for three segmentation methods. From left to right the images show the state after 1, 5, 10, 20 and 80 iterations respectively. *Top Row*: Algorithm 1, *Middle Row*: Algorithm 2, *Bottom Row*: Algorithm 3.

correctly resolve the different clusters. In contrast, our method works well in both cases; it is much more accurate: it can resolve differences of 6 pixel/s for objects moving at 50 pixel/s to 80 pixel/s, given the same slice of events.



(a) DAVIS performs linear translation over a multi-object scene.

(b) Resulting image and events (red and blue, indicating polarity).

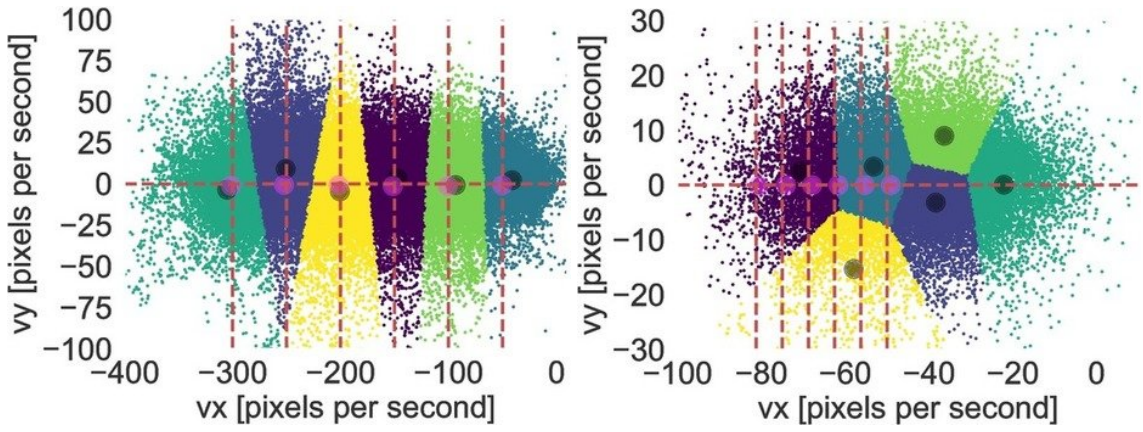
(c) Clustered volume of events (colored by cluster number).

(d) Motion-compensated image (colored by clustered optic flow).

(e) Clustered volume of events (colored by cluster number).

(f) Motion-compensated image (colored by recovered optic flow).

Figure 3.14: *Numbers Sequence*. Motion Segmentation by *K*-means clustering on estimated optic flow (center row) and by Algorithm 1 (bottom row).



(a) Minimum velocity between objects: 50 pixel/s

(b) Minimum velocity between objects: 6 pixel/s

Figure 3.15: *Rocks at Different Speeds*. Segmentation by k-means clustering of estimated optical flow ($k = 6$). The plots show the distribution of optical flow vectors and the six Voronoi diagrams resulting from k-means clustering on optic flow space. The crossings of red dashed lines indicate the ground truth optical flow velocity, the dark circles indicate the centroids of the k-means clusters. The pink circles indicate the cluster's optical flow estimated by our method (Algorithm 1).

3.7 Discussion

In this work we presented the first method for per-event segmentation of a scene into multiple objects based on their apparent motion on the image plane. We jointly segmented a given set of events and recovered the motion parameters of the different objects (clusters) causing them. Additionally, as a by-product, our method produced motion-compensated images with the sharp edge-like appearance of the objects in the scene, which may be used for further analysis (e.g. recognition). We showed that our method outperforms two recent methods on a publicly available dataset (with as much as 10 % improvement over the state-of-the-art [58]), and showed it can resolve small relative motion differences between clusters. Our method achieves this using a versatile cluster model and avoiding explicit estimation of optical flow for motion segmentation, which is error prone. All this allowed us to perform motion segmentation on challenging scenes, such as high speed and/or HDR, unlocking the outstanding properties of event-based cameras.

Our method is not reliant on scene-dependent hyperparameters and even the key parameters of the number of clusters N_l is not critical as overparametrisation leads to mode collapse of superfluous clusters. In the next chapter we explore alternative focus measures to the variance and their impact on FO.

3.7.1 Resources

Video and dataset: `https://timostoff.github.io/19ICCV`

Chapter 4

Comparison of Focus Measures

Based on [103]

The variance of the Image of Warped Events (IWE) is the most commonly used reward function in Focus Optimisation (FO), which has also been shown to have superior convergence properties [25] to other rewards, such as the sum of squares of the average timestamp image [58]. However, there are many different rewards which could be employed to estimate the focus of the IWE. In this work, we construct two classes of focus measuring functions: magnitude and sparsity rewarding. We define *aperture uncertainty*, an analogous concept to the aperture problem in conventional computer vision applied to focus optimisation. We show that sparsity rewarding functions are smoother and are less prone to aperture uncertainty. We also show why magnitude rewarding functions, such as the variance may have local or even global maxima at locations which do not represent the location of the desired solution. We back this theory up with experiments on various examples, from scenes constructed to display focus measure properties to a more complex office scene.

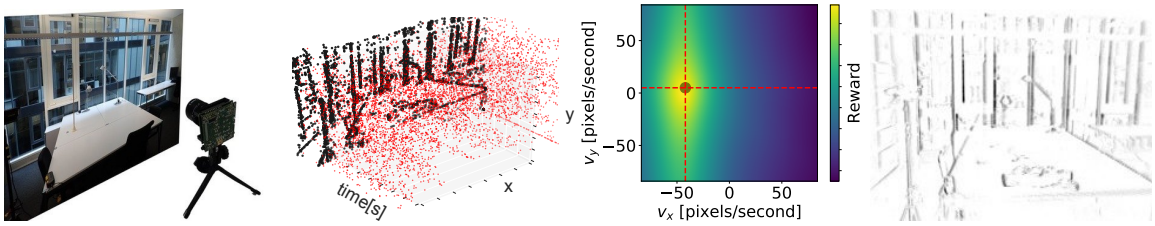
4.1 Introduction

Events carry little information individually and so are not meaningfully treated in isolation. So far, event-based algorithms have been in one of two categories; those which operate on *individual* events to update some previous state and those which operate on a *set* of events to perform a given task or estimate a particular quantity [26]. Those methods which operate on individual events typically require historic information, contained in the maintained state (e.g. reconstructed grayscale images) to make inferences. On the other hand, those which operate on a set of events require no external information. As noted in [26], the former category can be further broken down into those methods which *(a)* discard the temporal information carried by the events, for example by accumulating the events into frames over a temporal window and then performing computations on those frames (such as [51, 67, 90, 120]) and those which *(b)* utilise the temporal information of the events (such as [3, 26, 27, 58, 62, 63, 84, 104, 117, 120]). Since traditional computer vision algorithms are not usually designed with continuous time, asynchronous data such as events, these methods require novel techniques.

One such technique is that of Focus Optimisation (FO), whereby events are warped along point trajectories to the image plane. The trajectories can then be optimised with respect to the resulting Image of Warped Events (IWE) I_ω to recover the point trajectories that best fit the original set of events.

4.1.1 Focus Optimisation (FO)

As mentioned in Section 2.2.1, FO emerged recently as a promising technique for solving a number of problems in event-based vision. FO warps events along their motion trajectories to produce motion-compensated images called the IWE (Figure 4.1d), whose focus is optimal when the estimated trajectory matches the true spatiotemporal trajectory of the events on the image plane. The focus is



(a) Event camera moves around a scene.

(b) Events (red) generated by intensity gradients in the scene (black).

(c) Plot of reward vs optical flow estimate. Ground truth in red dotted lines, black circle represents estimated optical flow velocity.

(d) Motion-compensating the events reveals the original gradients.

Figure 4.1: Focus Optimisation: Events generated by scene and camera motion (4.1a) form a point cloud in a space-time volume (4.1b). If the events are motion-compensated by some trajectory, the contrast at that point can be evaluated by some reward r. Since the resulting reward has gradients with respect to trajectory parameters (4.1c), the original trajectory can be estimated, giving optic flow and motion-correction (4.1d) in one step.

usually measured via a proxy function such as the variance of the IWE. Since the variance also measures the contrast of the image, FO is also often referred to as Contrast Maximisation in the literature. More formally, given an event defined by its image position, time-stamp and sign of intensity change, $e_n = \{\mathrm{x}_n, t_n, s_n\}$, we define the warped location of the event with respect to the warp parameters ω as

$$\mathrm{x}'_n = \mathcal{W}(x_n, t_n; \omega), \tag{4.1}$$

Where $\mathcal{W}$ is the warping function. Thus the image of warped events from N_e events is

$$I_\omega = \sum_{n=1}^{N_e} b_n \delta(\mathrm{x} - \mathrm{x}'_n), \tag{4.2}$$

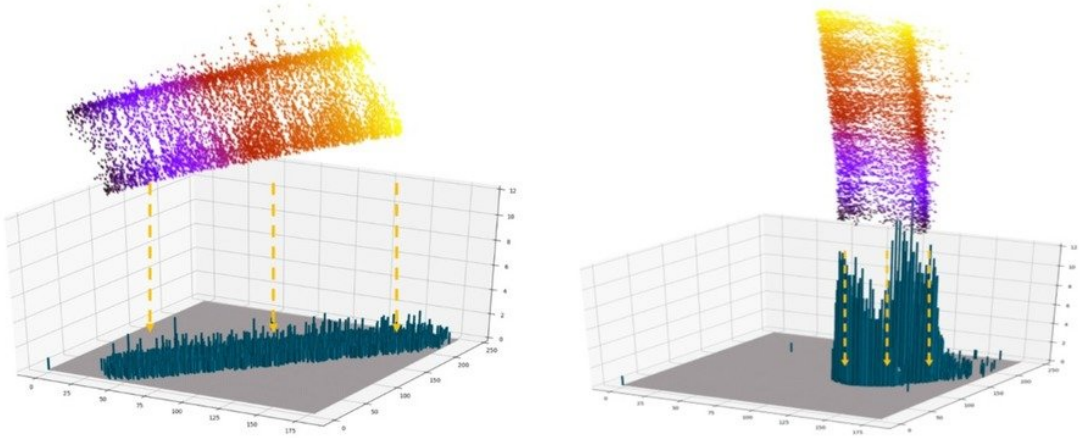
[26] where each pixel x sums the warped events x'_n that map to it (indicated by δ since they represent intensity spikes). $b_n = 1$ or $b_n = s_n$ depending on whether the event polarities should be considered or not. Since x' is usually not a discrete value, the events are distributed to the IWE via bi-linear interpolation, with each pixel (u, v) in the IWE given by:

$$I_\omega(u, v) = \sum_{n=1}^{N_e} b_n \max(0, 1 - |x'_n - u|) \max(0, 1 - |y'_n - v|) \tag{4.3}$$

The resulting IWE can then be evaluated using a reward function and optimised w.r.t. the warp parameters using an off-the-shelf optimiser. Thus, the steps of the FO method are:

- Collect a set of events generated by gradients moving across the image plane.
- Based on a motion model, generate image of warped events I_ω.
- Use a reward function to evaluate I_ω.
- Optimise the reward with respect to the motion parameters.

An advantage of this method is that the problem of event associations (which events were produced by the same feature) is solved implicitly. FO is a versatile method and has been recently used to estimate camera rotation on a static scene [27], estimate optical flow [104], track features in the event stream [117], estimate camera motion and depth [26], perform moving object detection [58] and provide a training signal for deep learning using events [120].



(a) (b)

Figure 4.2: Events from circle moving across the image plane projected along a good (4.2b) and a poor (4.2a) estimate of the actual trajectory. In 4.2a sum of accumulations squared (r_{SoS}) is $5,683$ while in 4.2b $r_{\text{SoS}} = 27,884$.

One aspect to FO which has not previously been explored is the choice of focus-measuring function. In this chapter we look at some additional classes of focus measure and their effect on FO.

4.1.2 Contributions

Our contributions are:

- A description of two classes of reward function, magnitude and sparsity rewarding for FO.
- A definition of aperture uncertainty, an analogue to the aperture problem in conventional computer vision.
- We show why sparsity rewarding functions have a smoother objective function and why the two classes behave differently w.r.t. aperture uncertainty.
- Proof and experimental evidence of reward function properties.

An unsolved problem with IWE is determining how many events should be processed at once. For the sake of efficiency the warping model used is typically a linearisation of some higher dimensional trajectory, thus it is important that the set of events does not span too great a time. However this duration is dependent on the dynamics of the scene. We propose a new solution to this problem which is fully compatible with the general IWE framework.

4.2 Reward Functions

In [26, 27] the total variance of $\mathcal{I}$,

$$r_{\sigma^2}(I_\omega) = \frac{1}{N_p}\sum_{i,j}(h_{i,j} - \mu(I_\omega)^2) \tag{4.4}$$

was used to evaluate the warp, (N_p is the number of pixels, $\mu(I_\omega)$ the mean of I_ω and $h_{i,j}$ is the value of pixel i,j in I_ω. In [104] the sum of squares of I_ω (4.5) was used. These two rewards are essentially equivalent as shown in [27]. The reason these two rewards work is because they disproportionately reward event accumulations of a high magnitude. (Figure 4.2). This occurs since, at the optimal trajectory, events are accumulated onto the small set of locations on the image plane at t_{ref} that was

occupied by the original gradients at t_{ref}. In other words any reward will work that rewards high accumulations more than the same total accumulations spread across more locations. At the same time, if most events are accumulated at fewer locations, it means that most locations at t_{ref} contain no events at all. Therefore we propose to explore the benefits of the following rewards for FO:

- **Sum of Squares** (r_{SoS}):

$$r_{\text{SoS}}(I_\omega) = \frac{1}{|\Omega|}\int_\Omega {I_\omega}^2. \tag{4.5}$$

The derivative of this reward with respect to the warp parameters is

$$\frac{\partial r_{\text{SoS}}}{\partial \omega} = \frac{1}{|\Omega|}\int_\Omega 2I_\omega \frac{\partial I_\omega}{\partial \omega} \tag{4.6}$$

Closely related to this reward is the variance:

$$r_{\sigma^2}(I_\omega) = \frac{1}{|\Omega|}\int_\Omega (I_\omega - \mu(I_\omega))^2 \tag{4.7}$$

for which the derivative is

$$\frac{\partial r_{\sigma^2}}{\partial \omega} = \frac{1}{|\Omega|}\int_\Omega 2I_\mu \frac{\partial I_\mu}{\partial \omega} \tag{4.8}$$

where I_μ is the mean-centred image $I_\mu = I_\omega - \mu(I_\omega)$ [27]. In practice the mean of the IWE is usually very small, so that $r_{\text{SoS}} \approx r_{\sigma^2}$. We will examine the r_{SoS} in this work but the results and conclusions are also valid for the variance.

- **Sum of Exponentials** (r_{SoE}):

$$r_{\text{SoE}}(I_\omega) = \frac{1}{|\Omega|}\int_\Omega \exp(I_\omega). \tag{4.9}$$

Exponentials reward higher numbers even more disproportionately than do polynomials (recall that $\lim_{n\to\infty} \frac{n^b}{a^n} = 0$). Therefore, this reward is more 'extreme' than r_{SoS} and r_{σ^2} in the sense that it rewards large values in the IWE yet more disproportionately. The derivative of this reward with respect to the warp parameters is

$$\frac{\partial r_{\text{SoE}}}{\partial \omega} = \frac{1}{|\Omega|}\int_\Omega \exp(I_\omega)\frac{\partial I_\omega}{\partial \omega} \tag{4.10}$$

- **Max of Accumulations** (r_{MoA}):

$$r_{\text{MoA}}(I_\omega) = \max_{u,v\in\Omega}(I_\omega(u,v)). \tag{4.11}$$

This reward simply returns the greatest accumulation. Since this reward does not measure other values in the IWE, it is yet more sensitive to large magnitudes in the IWE than r_{SoE}. Since the reward function is discontinuous, there is no analytic derivative. Thus, we use numeric gradients to optimise this function.

- **Sum of Accumulations** (r_{SoA}):

$$r_{\text{SoA}}(I_\omega) = -\frac{1}{|\Omega|}\int_\Omega I_\omega(u,v) > \lambda_{\text{sup}}. \tag{4.12}$$

This reward counts the number of locations $(u,v) \in \Omega$ on the IWE containing at least λ_{sup} events. The r_{SoA} is negated in order to frame the the optimisation as a maximisation, in keeping with the other reward functions. If $\lambda_{\text{sup}} = 1$ (the value we use), the λ_{sup} is just a measure of

the occupancy of the IWE and can be thought of as the support of the IWE. The trade-off with the threshold λ_{sup} is that for low values it is quite sensitive to noise, while for large values it may not trigger at all unless additional events are collected, which may introduce latency and motion-blur as well as violations of linear motion assumptions. The derivative is the piecewise function:

$$\frac{\partial r_{\text{SoA}}}{\partial \omega} = \frac{1}{|\Omega|}\int_{\Omega}\begin{cases}\frac{\partial I_\omega}{\partial \omega} & I_\omega(u,v) > \lambda_{\text{sup}} \\ 0 & \text{else}\end{cases} \tag{4.13}$$

- **Sum of Suppressed Accumulations** (r_{SoSA}):

$$r_{\text{SoSA}}(I_\omega) = \frac{1}{|\Omega|}\int_{\Omega}\exp(-\lambda_c I_\omega). \tag{4.14}$$

This reward gives locations with few accumulations in them a higher value than locations with many accumulations in them and saturates for large values of $I_\omega(u,v)$. This is to reduce the noise sensitivity of directly computing the support of the IWE as in the r_{SoA}. This reward is maximised at the optimal trajectory, since most events are accumulated at few locations and thus at most locations in the image $(u,v) \in \Omega$ will return a high value. The factor λ_c is an arbitrary shifting factor which decides the saturation point. In this work we used $\lambda_c = 10$. The derivative for this function is

$$\frac{\partial r_{\text{SoSA}}}{\partial \omega} = \frac{1}{|\Omega|}\int_{\Omega} -\lambda_c \exp(\lambda_c I_\omega)\frac{\partial I_\omega}{\partial \omega} \tag{4.15}$$

The problem we are trying to solve with all of these functions is

$$\arg\max_{\omega} r(I_\omega) \tag{4.16}$$

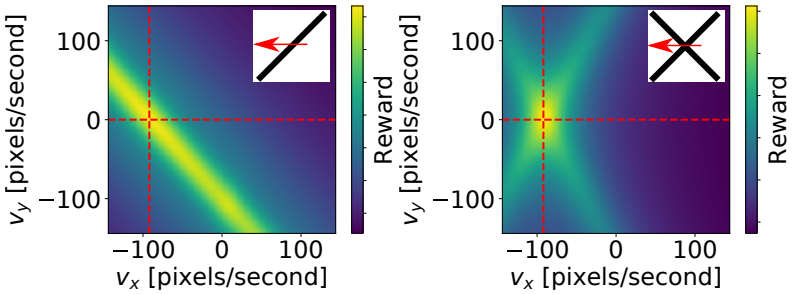
where r is the focus-measuring objective function used. Efficient optimisation algorithms make use of the derivative of the objective function in order to determine the search direction in the objective function. The generic solution to the gradient of a given objective function w.r.t. to the motion parameters is given by the chain rule:

$$\frac{\partial r(I_\omega)}{\partial \omega} = \frac{\partial r}{\partial I_\omega}\frac{\partial I_\omega}{\partial \omega} \tag{4.17}$$

The gradient of the IWE w.r.t. the warp parameters is given by

$$\frac{\partial I_\omega}{\partial \omega} = \sum_{i=1}^{N_e} s_i \nabla\delta(\mathrm{x}_i - \mathrm{x'}_i)\frac{\partial \mathrm{x'}_i}{\partial \omega} \tag{4.18}$$

[27], with x and x' being the position of the warped and unwarped event respectively, s the polarity and $\nabla\delta$ the derivative of the Dirac delta function, as per the usual notation in this thesis. Note that the polarity s is retained in the expression and that positive and negative events cancel each other. One can also ignore polarity information (in which case $s = 1$), however, we find optimisation to be more robust using polarity [25, 26].



(a) r_{SoS} of events generated by a line (motion vector in red).

(b) r_{SoS} of events generated by a cross (motion vector in red).

Figure 4.3: The reward function (using r_{SoS}) of the events generated by a straight line segment (4.3a) shows the effect of aperture problem on contrast maximisation techniques when compared to the reward function of a cross moving with the same optic flow velocity indicated by red dotted lines (4.3b)

$\frac{\partial \mathbf{x}'_i}{\partial \omega}$ is simply the Jacobian of motion model $\mathcal{W}$, with the value of each Jacobian added to an n channel image, where n is the size of the Jacobian. For example the optic flow warp function $\mathcal{W}_{\vec{v}}$:

$$\mathcal{W}_{\vec{v}}(\mathbf{x}, \Delta t; \omega) = \begin{bmatrix} x \\ y \end{bmatrix} - \Delta t \begin{bmatrix} v_x \\ v_y \end{bmatrix} \tag{4.19}$$

$$\nabla \mathcal{W}_{\vec{v}}(\mathbf{x}, \Delta t; \omega) = \begin{bmatrix} -\Delta t \\ -\Delta t \end{bmatrix} \tag{4.20}$$

In this case, each event e would contribute $-\Delta t$ to each location $\mathbf{x}$ in a 2-channel image, $\frac{\partial I_\omega}{\partial \omega}$. These values are distributed to pixels using bilinear voting as described previously.

Of these rewards, r_{SoS}, r_{SoE} and r_{MoA} favour trajectories that result in large accumulations (they are *magnitude-rewarding*) and r_{SoA} and r_{SoSA} favour those that result in many locations having few or no accumulations (they are *sparsity-rewarding*).

4.2.1 Aperture Problem

The aperture problem arises when optical flow is estimated using only a local region of a moving object. In this case it can happen that only a line feature of the object is visible and thus only the velocity component perpendicular to the local line feature can be estimated (Figure 1.6). FO techniques don't suffer from the aperture problem in the way that local optic flow estimators such as Lucas-Kanade [53] do, since they consider the scene globally.

However, long line segments introduce uncertainty to the optic flow estimates when using FO, which can be considered analogous to the aperture problem. A line segment moving over the Dynamic Vision Sensor (DVS) image plane will generate events which lie on a plane in the space-time volume [61]. Although warping the events along the trajectory of the line segment will generate a large value in the reward function, trajectories which vary slightly but still lie on the event plane will generate large values as well. This 'aperture uncertainty' can be seen in Figure 4.3a; the reward function for the straight line segment features a long ridge, along which the values are similar to each other.

It is actually worse than this - for scenes with very strong line segments there are likely to be two local maxima to either side of the true trajectory and none *at* the true trajectory. As a result, a good optimisation result would not guarantee a good motion estimate. This is because it is possible to achieve greater event accumulations when warping over the diagonal of the plane; since greater accumulations are rewarded in r_{SoS}, these trajectories will maximise the reward function (Figure 4.4).

Once the line segment gains features on other axes, this uncertainty is much reduced, the region around the ground truth forming a sharp spike, since changing the trajectory slightly in the direction

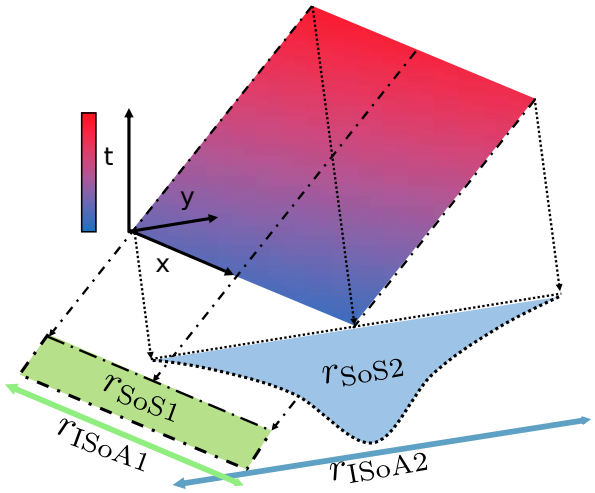


Figure 4.4: The plane in x, y, t represents a set of events generated by a line segment moving in the direction of the y axis. If the events are projected along the velocity vector (dashed arrows), they accumulate (green double arrow), giving the r_{ISoA1}. The integral of those accumulations (green area) squared is r_{SoS1}. If the events are instead (incorrectly) projected across the diagonal of the plane, the corresponding accumulations (blue arrow) give r_{ISoA2}, which being the inverse of the arrow length becomes smaller. The r_{SoS2} (blue area) however becomes larger, since it rewards the larger peak in the accumulations. Thus maximising the r_{SoA} would give a correct result, the r_{SoS} an incorrect one, showing how sparsity-rewarding rewards are less susceptible to the aperture problem.

of one of the image gradient's principal axes will cause events along the other axis to accumulate less (Figure 4.3b).

For sparsity-rewarding rewards the reward will experience much stronger relative change to slightly incorrect trajectories and thus suffer less from aperture problem. The reason for this is demonstrated in Figure 4.4; while warping the events parallel to the plane of events diagonally is not likely to influence the r_{SoS} strongly, it will cause the events on the resulting IWE to take up substantially more space and thus strongly affect the r_{SoA}. This effect is validated experimentally in datasets with dominant line segments (see Section 4.4.1, 4.4.3) and is visualised in Figure 4.5.

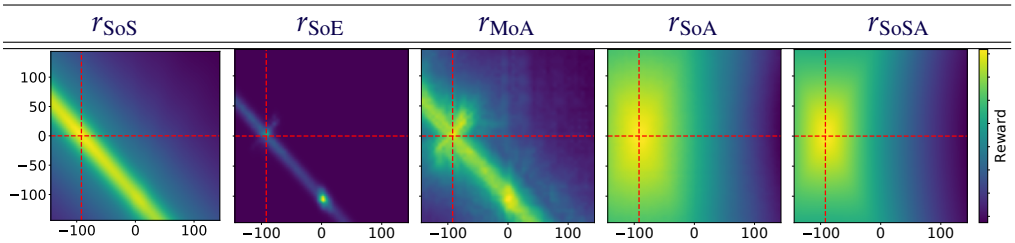


Figure 4.5: Various rewards sampled vs optical velocities v_x, v_y for a dataset with strong line features (see Figure 4.9). Ground truth is indicated by red dotted lines. Note that the magnitude-rewarding r_{SoS}, r_{SoE} and r_{MoA} are much more prone to having incorrect local maxima, for reasons illustrated in Figure 4.4 than the sparsity rewarding r_{SoA} and r_{SoSA}.

4.2.2 Noise Tolerance

As sparsity-rewarding methods essentially measure the number of locations containing events in them, they are susceptible to noise. For example, r_{SoA} becomes entirely meaningless in the worst case, where the event stream becomes so noisy that every location at t_{ref} contains at least one event.

As more noise is added, these rewards become more and more uniform, until they are almost entirely flat (Figure 4.8). Magnitude rewarding functions are far more robust in this respect, since noise events are generally distributed evenly across the image sensor (this is not the case for 'hot' pixels, which produce large numbers of events independently of stimulus or camera parameters, but these are easily removed in a calibration procedure and can be discounted). As a result, they do not cause large accumulations of their own and are easily overwhelmed by genuine events.

4.2.3 Data Sufficiency

FO falls into the category of algorithms that operate on groups of events [26]. This means that events need to be collected over some period of time before a meaningful estimate, such as optic flow can be made. In practice, waiting for a displacement of at least several pixels is necessary for a reliable estimate (Figure 4.7), which can take a long time for slow moving gradients. Estimating how long events should be collected before FO is applied is an important task, since most FO methods make an assumption of constant velocity over small periods of time in order to work. Further, the number of events generated by gradients in the image is dependent on the relative strength of the gradients, making simple event-counting methods somewhat scene dependent (as in Chapter 2). In our case, it is necessary to know how many events the gradient is producing per pixel moved, since the r_{SoSA} reward contains a shifting parameter, which needs to be tuned according to this value.

We can estimate how many events are produced per pixel moved directly from the r_{SoS} reward, by examining the image under projection along the zero velocity I_0. As a structure begins to move onto a new pixel, it generates events proportional to the intensity of its gradients. At $\vec{v} = 0$, these events on the same pixel will accumulate, causing the r_{SoS} reward to grow quadratically. As the structure moves on to the next pixel, quadratic growth will have to start anew, since the structure will be entering an empty set of pixels (Figure 4.6c). Thus the rate of change of the r_{SoS} along the $\vec{v} = (0,0)$ trajectory should flatten off periodically as the structure moves over the image plane. We can observe this in real data (Figure 4.6), allowing estimation of whether FO will be able to give an optic flow estimate given a set of events.

4.3 Combined Reward Functions

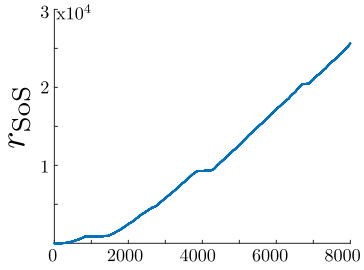
We have identified two classes of reward, sparsity- and magnitude-rewarding, that can be used to optimise the total contrast of an image of warped events and shown that the one class should be much better at dealing with aperture-uncertainty while the other should be more capable of tolerating noise. We wish now to use that knowledge to construct a new, hybrid reward, which is able to take the best from both classes of reward. Since r_{SoSA} gave better results than r_{SoA} (since it is not a binary measure it is slightly more noise-tolerant), we combined this with a variety of magnitude-rewarding rewards:

- $r_{\mathbf{R1}} = r_{\mathbf{SoS}} + r_{\mathbf{SoSA}}$ In this reward we use the r_{SoS} reward with the constraint during optimisation that successive improved estimates must not decrease the r_{SoSA}.
- $r_{\mathbf{R2}} = r_{\mathbf{SoS}} + r_{\mathbf{SoSA}} + r_{\mathbf{SoE}}$ Here we use the same reward as $r_{\mathbf{R1}}$, except that when we have finally found an estimate, we use it as a starting point for gradient ascent using the r_{SoE}.

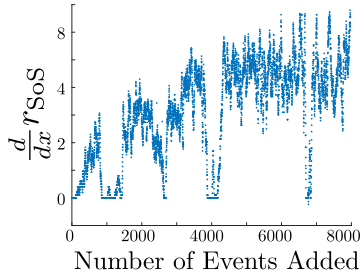
The r_{SoE} gives precise and noise-tolerant results, given that the starting point for the optimisation used is close to the maximum, which is why we incorporate it in the r_{R2} reward. Given a bad initial point r_{SoE} performs poorly since it is not sufficiently smooth for most optimisation methods.

4.4 Experimental Results

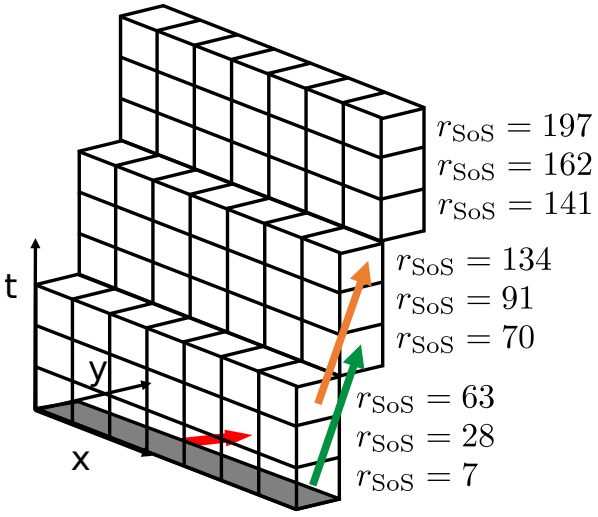
We tested our rewards on high quality optical flow ground truth data collected from a Dynamic and Active-pixel VIsion Sensor (DAVIS) 240C event camera [9], using a linear slider to pan over a variety



(a) r_{SoS} as events are added.



(b) Derivative of 4.6b.



(c) Events (white cubes) are accumulated on the image plane as an edge (grey) moves in the direction of the red arrow. The green arrow indicates the first crossing of a pixel boundary (where $\frac{d}{dx} r_{SoS}$), the orange arrow the second.

Figure 4.6: As an edge (grey) moves across the image plane of an idealised event camera, it generates some number of events proportional to the intensity of the edge. These events can be summed and squared to give the r_{SoS} and producing quadratic growth in the r_{SoS} as events are added. As the edge moves onto the next pixel, the r_{SoS} grows at a slower rate, so *the derivative of r_{SoS} with respect to added events* becomes close to zero. The minimum number of events required to compute the optic flow is shown by the green arrow. This effect can be seen in real data - in 4.6b can be seen the r_{SoS} as events are added from the sequence in Section 4.2. In 4.6b the temporal derivative is zero when the object crosses pixel boundaries.

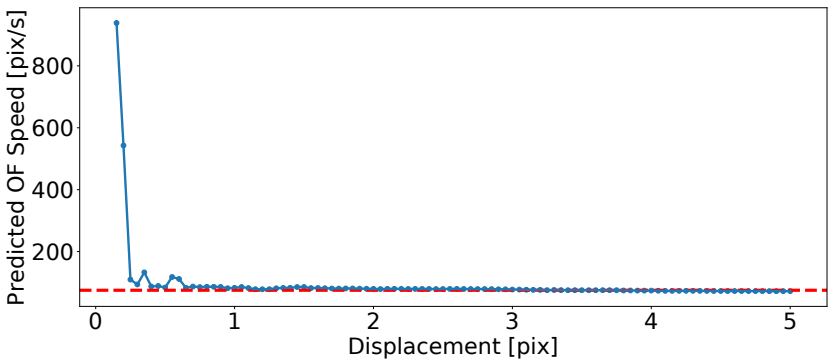


Figure 4.7: Predicted optic flow (OF) vs displacement using r_{SoS} reward function (*blue*). A displacement of around 2 pixels is necessary for optimisation to converge to the correct result (*red*).

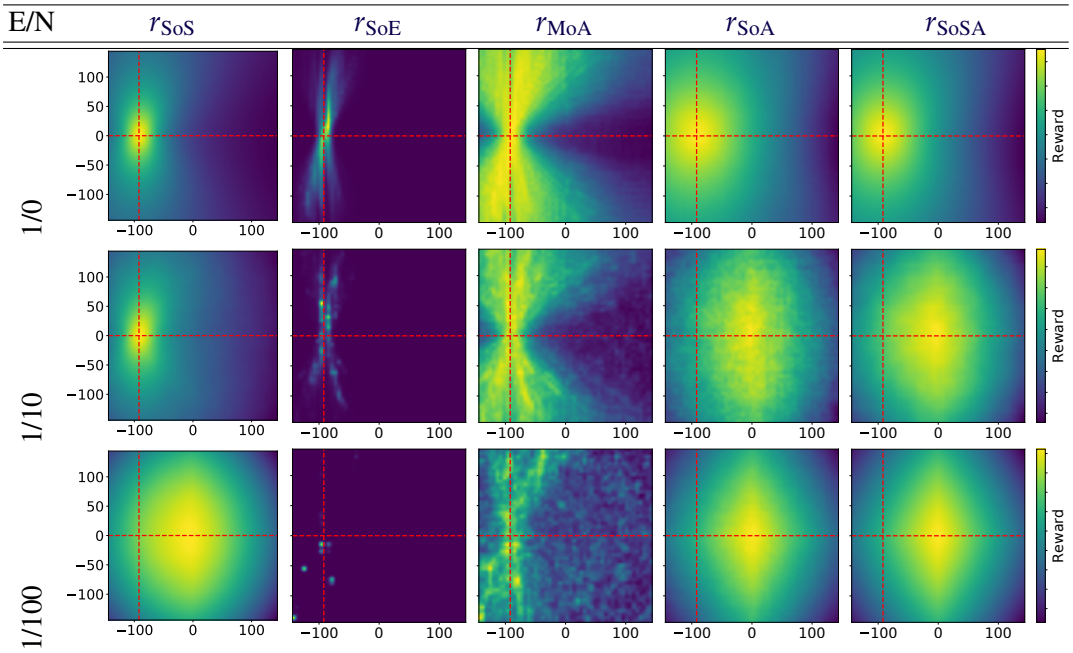


Figure 4.8: FO reward functions measured at varying optical velocities (v_x, v_y) under various Event-to-Noise (E/N) ratios. Ground truth indicated by red dotted lines. Note that the magnitude-rewarding rewards are much more robust at high event to noise ratios, still giving reasonable estimates at 1/100 E/N. The sparsity-rewarding rewards fail, becoming increasingly flat distributions with the center at $\vec{v} = 0, 0$.

of scenes. We tested on two simple sequences of a line segment (Table 4.1) and a circle (Table 4.2) moving across the image plane. The line segment sequence introduces a lot of aperture-uncertainty, while the circle has none. We also tested the rewards on a real office scene (Table 4.3) to show that the hypotheses tested in (Table 4.1) and (Table 4.2) have validity in scenes one might find 'in the wild'. We restricted ourselves to relatively simple scenes, since it made it easier to collect ground truth data and control the level of aperture uncertainty.

It is not our intent to prove the usefulness of the FO framework on real scenes here, since this has been done several times in other work [26, 27, 58, 104, 117, 120]. Instead the experiments are designed to highlight specific properties of the reward functions.

In the experiments we added random noise to the event stream to give us a signal/noise ratio of 1/10. In this way we were able to show the benefits of different rewards with respect to noise, which can be quite a significant component of current event camera output, as well as the benefits with respect to aperture-uncertainty.

In all of the experiments the same nonlinear conjugate-gradient optimisation was used (Polak-Ribiere variant), with numeric gradients derived through the same method. Optimisation was performed on consecutive batches of events, using a sliding window with a width of one pixel displacement with regard to the optic flow velocity, with 100 samples taken per sequence.

Optical flow with respect to events is a slightly different creature from the conventional definition; usually optic flow measures the displacement of pixels between consecutive frames, hence the usual way of evaluating optical flow is through looking at the average endpoint error. For events however, this definition makes less sense, since optical flow relates to the velocity of an event on the image plane, not the displacement. Instead we will look at the average absolute magnitude error $\mu(|\mathrm{ME}|)$, so the average error of the magnitudes of the optical flow vector estimates with respect to ground truth, and the average angular error $\mu(\mathrm{AE})$, so the average error of the estimated vector angles. This way we can also look at the standard deviation of the errors, $\sigma^2(|\mathrm{ME}|)$ and $\sigma^2(\mathrm{AE})$.

4.4.1 Line Segment Sequence

Table 4.1: Absolute magnitude error |ME| and angular error AE of flow estimate vectors for line segment sequence

Line Segment Sequence				
Event/Noise = 1/0 (No Noise)				
r	$\mu(\lvert ME \rvert)$	$\sigma^2(\lvert ME \rvert)$	$\mu(AE)$	$\sigma^2(AE)$
r_{SoS}	38.29	1.68	-0.311	0.007
r_{SoE}	137.51	14.02	-0.546	0.019
r_{MoA}	22.41	10.92	-0.280	0.038
r_{ISoA}	11.36	5.41	-0.204	0.006
r_{SoSA}	12.79	**1.02**	-0.234	0.006
r_{R1}	**10.31**	3.26	-0.122	0.003
r_{R2}	10.50	3.14	**-0.103**	**0.004**
Event/Noise = 1/10				
r	$\mu(\lvert ME \rvert)$	$\sigma^2(\lvert ME \rvert)$	$\mu(AE)$	$\sigma^2(AE)$
r_{SoS}	49.14	25.48	-0.611	0.077
r_{SoE}	47.64	30.86	-0.712	0.401
r_{MoA}	82.11	33.63	-0.482	0.051
r_{ISoA}	88.06	11.25	0.559	1.096
r_{SoSA}	67.05	3.60	-0.635	0.010
r_{R1}	55.33	12.56	-0.547	0.012
r_{R2}	**37.48**	**3.52**	**-0.440**	**0.007**

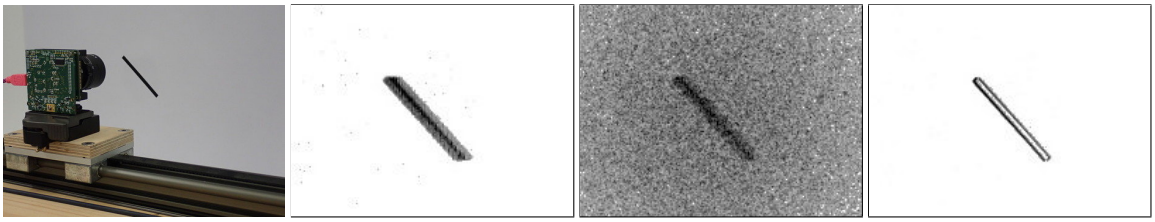
The line-segment sequence (Figure 4.9a) illustrates the behaviour of the different rewards when exposed to data with strong line features (this particular sequence consisting *only* of line features). As discussed in Section 4.2.1, one would expect sparsity-rewarding rewards to perform better on this sequence, at least in the case where the event to noise ratio is high. Indeed, of the conventional rewards, the sparsity rewarding ones r_{SoA} and r_{SoSA} score best under this extreme case of aperture uncertainty.

However, as hypothesised, once noise is added to the event stream, the sparsity-rewarding rewards perform much worse than the magnitude-rewarding ones. It is worth pointing out here the remarkable robustness of the FO method; even with an order of magnitude more noise polluting the events, the optic flow estimates are still quite reasonable and only slightly deteriorated. The hybrid methods, which are able to take advantage of the best properties of both types or reward, perform best under both normal conditions and with large amounts of noise.

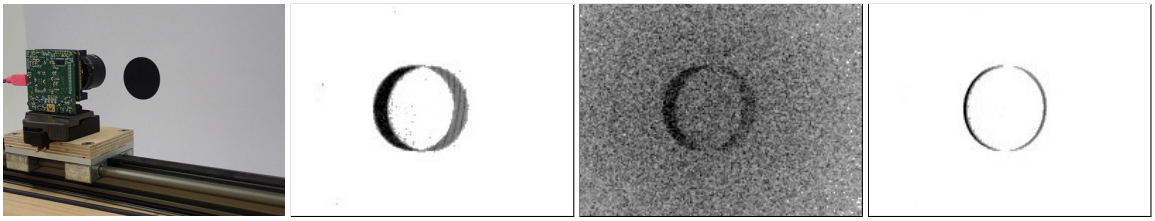
4.4.2 Circle Sequence

In the circle sequence the event camera slides past an image of a circle generating the events visualised in Figure 4.9e. This sequence illustrates a scene in which there is no aperture-uncertainty, since there are no dominant line features in the resulting events. In fact, the winner in such a scene is the commonly implemented r_{SoS}, which performs almost three times better than even our hybrid rewards, though these have got slightly better accuracy in the average angular error.

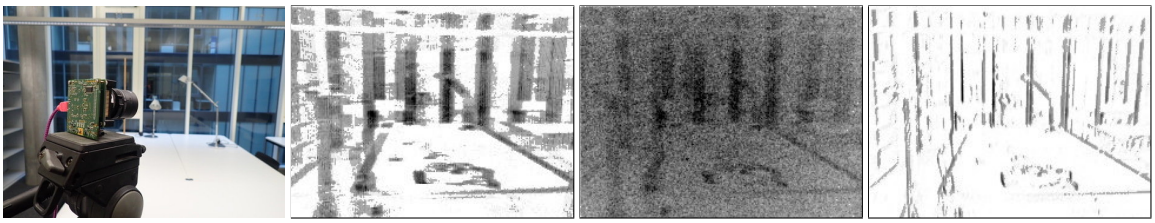
Once noise is added to the event stream however, the accuracy of r_{SoS} is two entire orders of magnitude worse, whereas the hybrid reward r_{R2} only becomes five times worse, clearly beating the other methods. Interestingly, the extreme magnitude-rewarding rewards r_{SoE} and r_{MoA} actually improve as the scene becomes nosier. This is because these reward functions have quite strong peaks and are thus prone to local convergence issues; the noise effectively blurs the reward function and thus makes gradient ascent easier. In the interest of a fair comparison, the same amount of smoothing was



(a) Setup (b) Timeslice of events (c) Noise added to the events (d) Motion-compensated



(e) Setup (f) Timeslice of events (g) Noise added to the events. (h) Motion-compensated



(i) Setup (j) Timeslice of events (k) Noise added to the events (l) Motion-compensated

Figure 4.9: *Top row*: DAVIS 240C camera mounted on a linear slider pans past a line segment primitive. *Middle row* DAVIS 240C camera mounted on a linear slider pans past a circle primitive. *Bottom row* DAVIS 240C camera pan over an office scene. Line segment sequence: in the experimental setup, the event camera is moved past a line segment using a linear slider. Slices of the resulting event stream are processed to extract optical flow estimates, both with noise added to the event stream and once using only the original events.

applied to all rewards during optimisation; for more details on the effect of smoothing on different reward types, see the supplementary materials.

Table 4.2: Absolute magnitude error $|\text{ME}|$ and angular error AE of flow estimate vectors for circle sequence

Circle Sequence				
Event/Noise = 1/0 (No Noise)				
r	$\mu(\|\text{ME}\|)$	$\sigma^2(\|\text{ME}\|)$	$\mu(\text{AE})$	$\sigma^2(\text{AE})$
r_{SoS}	**0.49**	**0.37**	-0.153	0.010
r_{SoE}	72.14	11.20	1.517	0.033
r_{MoA}	73.45	18.32	1.066	0.051
r_{ISoA}	2.37	1.91	-0.305	0.034
r_{SoSA}	1.25	0.78	-0.350	0.009
r_{R1}	1.58	1.27	-0.100	**0.009**
r_{R2}	1.69	1.42	**-0.036**	0.040
Event/Noise = 1/10				
r	$\mu(\|\text{ME}\|)$	$\sigma^2(\|\text{ME}\|)$	$\mu(\text{AE})$	$\sigma^2(\text{AE})$
r_{SoS}	43.41	2.52	-0.512	0.010
r_{SoE}	23.56	27.46	0.354	0.640
r_{MoA}	11.10	3.83	0.132	0.090
r_{ISoA}	89.33	7.76	0.898	1.053
r_{SoSA}	72.79	2.61	-0.737	0.009
r_{R1}	57.95	14.91	-0.531	0.014
r_{R2}	**5.29**	**1.57**	**0.113**	**0.053**

4.4.3 Office Sequence

The office sequence (Figure 4.9i) consists of panning across an office scene, to which the ground truth optic flow velocities were hand-annotated afterwards. The sequence serves to illustrate that the ideas tested in the previous experiments also apply to real scenarios. As is often the case in real world sequences, there are several strong line features visible in the event stream, due to the edges of windows, tables etc. As such, it is hardly surprising to see that our hybrid approach is able to out-compete the existing methods both with and without added noise.

4.5 Proofs and Additional Experiments

In this section we provide some geometric proofs and insights into the findings of the preceding sections as well as a few additional experiments regarding convergence of the optimisation. In Section 4.5.1 we prove the notion that the magnitude-rewarding function r_{SoS} will perform worse on line features than the sparsity-rewarding function r_{SoA}. In Section 4.5.2 we show the results of an experiment where we test weighted sums of rewards. In Section 4.5.3 we show the results of an experiment to test various blurring kernel strengths during optimisation.

Table 4.3: Absolute magnitude error |ME| and angular error AE of flow estimate vectors for office sequence

Office Sequence				
Event/Noise = 1/0 (No Noise)				
r	$\mu(\lvert ME \rvert)$	$\sigma^2(\lvert ME \rvert)$	$\mu(AE)$	$\sigma^2(AE)$
r_{SoS}	5.58	3.58	-0.116	0.029
r_{SoE}	17.89	18.43	0.178	0.879
r_{MoA}	19.79	19.68	0.369	0.890
r_{ISoA}	16.28	18.64	-0.064	0.365
r_{SoSA}	5.47	3.28	-0.170	0.041
r_{R1}	9.09	12.93	-0.009	**0.041**
r_{R2}	**4.95**	**3.23**	**-0.008**	0.112
Event/Noise = 1/10				
r	$\mu(\lvert ME \rvert)$	$\sigma^2(\lvert ME \rvert)$	$\mu(AE)$	$\sigma^2(AE)$
r_{SoS}	16.07	5.95	-0.455	**0.042**
r_{SoE}	15.83	14.98	-0.084	0.766
r_{MoA}	20.26	17.40	-0.139	0.692
r_{ISoA}	46.82	3.55	-0.679	1.121
r_{SoSA}	49.09	**0.40**	-0.993	0.684
r_{R1}	48.66	1.45	-1.119	0.312
r_{R2}	**15.08**	14.66	**-0.048**	0.262

4.5.1 Aperture-Uncertainty of Sparsity vs Magnitude Rewarding

In Section 4.2.1 we discuss why magnitude-rewarding rewards are likely to fail on line features (due to aperture uncertainty), together with a visual example (Figure 4.4) and plots of line features in real data (Figure 4.6). Here we will prove this notion geometrically. As a line feature moves across the image, it generates a plane of events. If one only considers event warp trajectories that run parallel to the plane, the possible trajectories become two-dimensional - we can thus model the event plane as a rectangle of height h and width w (see Figure 4.10).

Computing the sum of event accumulations along the actual trajectory of the line feature then just becomes

$$\int_a^b h dx, \tag{4.21}$$

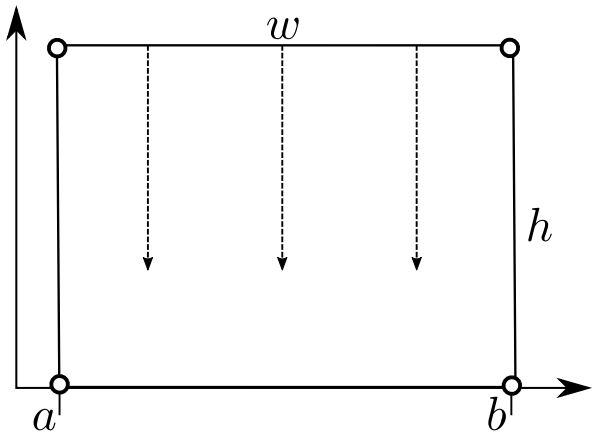
with the r_{SoS}

$$r_{SoS} = \int_a^b h^2 dx. \tag{4.22}$$

Likewise, incorrect projections parallel to the plane of events can now be modelled by rotations of the rectangle around point b by some angle ϕ (see Figure 4.11) - we rotate the events instead of rotating the warp vector.

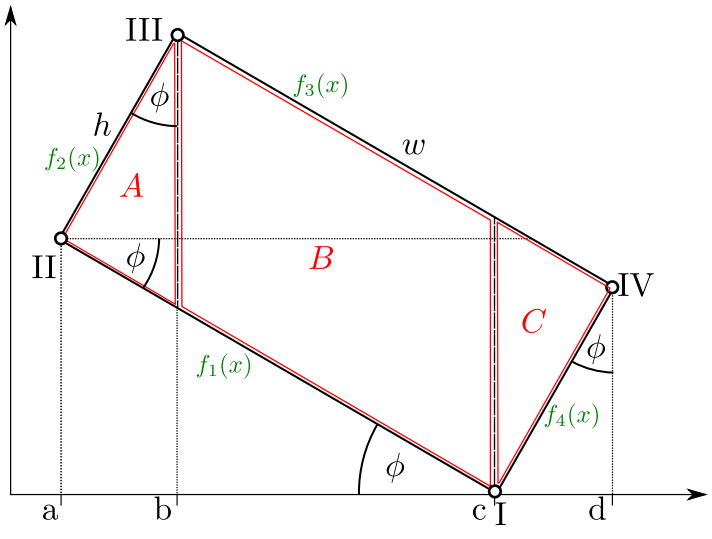
The accumulation of events can now be written as a piecewise function over the regions which are bounded by the lines $f_1(x)$, $f_2(x)$, $f_3(x)$ and $f_4(x)$. If I is set to equal $(0,0)$, then

$$\mathrm{I} = (0,0) \tag{4.23}$$
$$\mathrm{II} = (-w\cos(\phi), w\cos(\phi)) \tag{4.24}$$
$$\mathrm{III} = (-w\cos(\phi) + h\sin(\phi), w\cos(\phi) + h\cos(\phi)) \tag{4.25}$$
$$\mathrm{IV} = (h\sin(\phi), h\cos(\phi)). \tag{4.26}$$



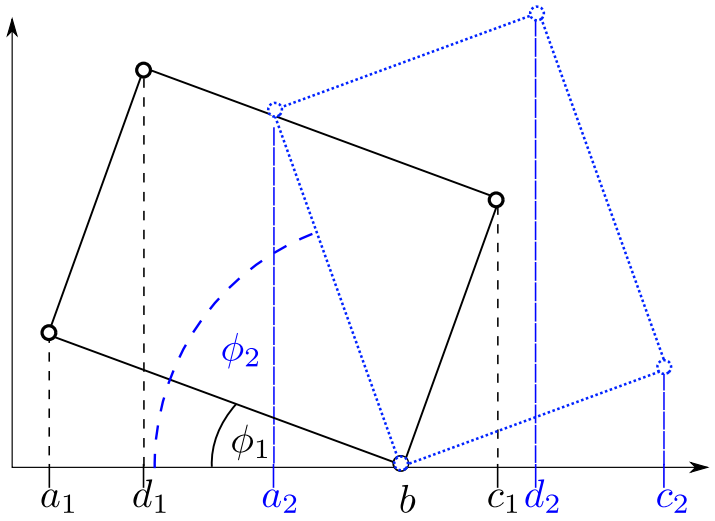
(a)

Figure 4.10: The local event plane modelled as a rectangle in 2D; the actual trajectory of the line feature is shown as the dashed arrows.



(a)

Figure 4.11: The event plane modelled as a rectangle; possible optic flow trajectories parallel to the event plane become rotations of the rectangle around point b by rotation ϕ.



(a)

Figure 4.12: Two situations exist for $0 \leq \phi \leq \pi/2$: that $d \leq b$ (black) and $b \leq d$ (blue).

Thus,

$$f_1(x) = -\tan(\phi)x \tag{4.27}$$
$$f_2(x) = w\csc(\phi) + x\cot(\phi) \tag{4.28}$$
$$f_3(x) = h\sec(\phi) - x\tan(\phi) \tag{4.29}$$
$$f_4(x) = \cot(\phi)x \tag{4.30}$$

The function for the sum of accumulations can now be written as the piecewise function

$$f_{\text{sum}}(x) = \begin{cases} \begin{cases} \csc(\phi)(w + x\sec(\phi)) & a \leq x < d \\ h\sec(\phi) & d \leq x < b \quad b > d \\ \sec(\phi)(h - x\csc(\phi)) & b \leq x < c \end{cases} \\ \begin{cases} \csc(\phi)(w + x\sec(\phi)) & a \leq x < b \\ w\csc(\phi) & b \leq x < d \quad b \leq d \\ \sec(\phi)(h - x\csc(\phi)) & d \leq x < c \end{cases} \end{cases} \tag{4.31}$$

The double piecewise function is necessary, since there are two distinct cases as ϕ increases (see Figure 4.12); that when $b > d$ and that when $b \leq d$.

The integral of $f_{\text{sum}}(x)$ is

$$\int_a^c f_{\text{sum}}(x) = \begin{cases} h^2\tan(\phi) + h(w - h\tan(\phi)) & b > d \\ w^2\cot(\phi) + w(h - w\cot(\phi)) & b \leq d, \end{cases}' \tag{4.32}$$

which after a bit of rearranging comes out as

$$\int_a^c f_{\text{sum}}(x) = \begin{cases} hw & b > d \\ hw & b \leq d, \end{cases} \tag{4.33}$$

(as one would expect, since the area of the original rectangle is invariant). If we now however look as the r_{SoS} of the rotated rectangle, things are a little more interesting. We simply integrate the square of

$f_{\text{sum}}(x)$:

$$f_{\text{sum}}(x)^2 = \begin{cases} \begin{cases} \csc(\phi)^2(w + x\sec(\phi))^2 & a \leq x < d \\ h^2\sec(\phi)^2 & d \leq x < b \quad b > d \\ \sec(\phi)^2(h - x\csc(\phi))^2 & b \leq x < c \end{cases} \\ \begin{cases} \csc(\phi)^2(w + x\sec(\phi))^2 & a \leq x < b \\ w^2\csc(\phi)^2 & b \leq x < d \quad b \leq d \\ \sec(\phi)^2(h - x\csc(\phi))^2 & d \leq x < c \end{cases} \end{cases} \tag{4.34}$$

and

$$\int_a^c f_{\text{sum}}(x)^2 = \begin{cases} \frac{2}{3}h^3\tan(\phi)\sec(\phi) + \sec(\phi)h^2w & b > d \\ \frac{2}{3}w^3\cot(\phi)\csc(\phi) + \csc(\phi)w^2h & b \leq d \end{cases} \tag{4.35}$$

$$= r_{\text{SoS}} \tag{4.36}$$

This integral is the familiar r_{SoS}. If we plot (4.35) for $0 \leq \phi \leq \frac{\pi}{2}$, $w = 10$ and a range of values for h (see Figure 4.13), we see that the actual maximum of the r_{SoS} is only at $\phi = 0$ (as it should be), when $h >> w$.

In the case of events, the 'height' of the event plane is only much greater than the width when a lot of time passes, since the time axis is the equivalent of h in this analogy. This shows rather neatly, not only why magnitude-rewarding rewards fail with line features, but that this uncertainty is dependent on the accumulation time.

We can show this formally too; looking at the equations

$$\frac{2}{3}h^3\tan(\phi)\sec(\phi) + \sec(\phi)h^2w \tag{4.37}$$

$$\frac{2}{3}w^3\cot(\phi)\csc(\phi) + \csc(\phi)w^2h \tag{4.38}$$

from (4.35), note that (4.37) is monotonically increasing for all values of $0 \leq \phi \leq \frac{\pi}{2}$ in the range $b > d$ and that (4.38) is monotonically decreasing for all values of $0 \leq \phi \leq \frac{\pi}{2}$ in the range $b \leq d$ (for fixed values of w and h, a valid assumption since the dimensions of the event plane don't change). This means that the maximum of the total function (4.35) is at the point where (4.37) = (4.38). Then

$$\frac{h^2}{w^2} = \cot^2(\phi) \tag{4.39}$$

and

$$\phi = \cot^{-1}\sqrt{\frac{h^2}{w^2}}. \tag{4.40}$$

Solving for $\phi = 0$,

$$0 = \cot^{-1}\sqrt{\frac{h^2}{w^2}} \tag{4.41}$$

for which no solution exists. However,

$$\lim_{n\to\infty}\cot^{-1}\sqrt{n} = 0. \tag{4.42}$$

Where $n = \frac{h}{w}$. Therefore the ratio of $\frac{h}{w}$ must be very large in order to get the correct trajectory of a line feature using magnitude-rewarding rewards.

The analogy for sparsity-rewarding rewards such as the r_{SoA} is of course the length of the range $[a, c]$, since this corresponds to the count of locations to which events have projected. The equation

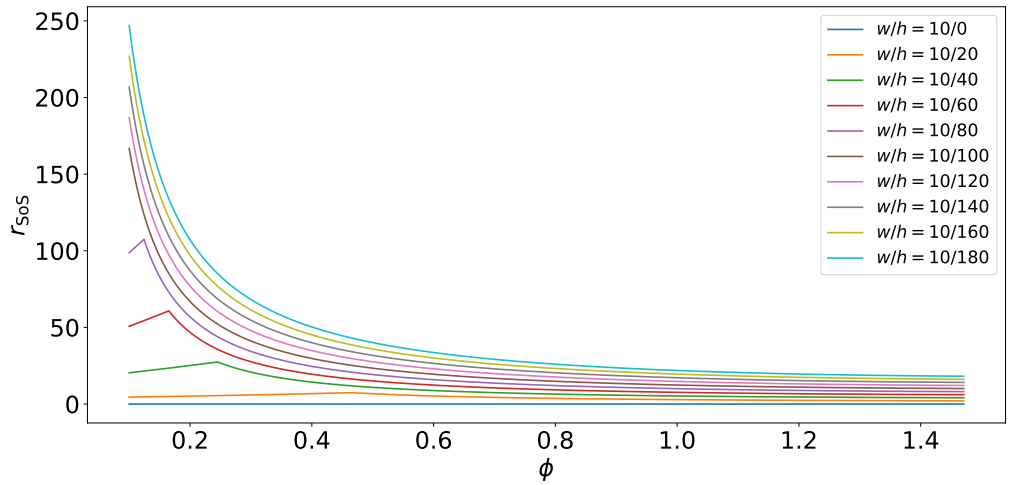


Figure 4.13: Plot of the r_{SoS} for various ratios of w/h, over $0 \leq \phi \leq \pi/2$.

describing this measure is

$$\frac{1}{a-b} = \frac{1}{h\sin(\phi) + w\cos(\phi)} \tag{4.43}$$

This is a convex function on the range $0 \leq \phi \leq \frac{\pi}{2}$, therefore the maximum must lie at either 0 or $\frac{\pi}{2}$ for the range $0 \leq \phi \leq \frac{\pi}{2}$. Recall that we require the maximum of this function to be at $\phi = 0$, since this represents no rotation and therefore the actual optic flow velocity. Being interested in those cases where the value of Equation 4.43 is greater at $\phi = 0$ than $\phi = \frac{\pi}{2}$:

$$\frac{1}{h\sin(0) + w\cos(0)} > \frac{1}{h\sin(\frac{\pi}{2}) + w\cos(\frac{\pi}{2})} \tag{4.44}$$

$$= \frac{1}{w} > \frac{1}{h} \tag{4.45}$$

$$= h > w \tag{4.46}$$

This means that a longer observation window (greater h) will produce a better optimisation result.

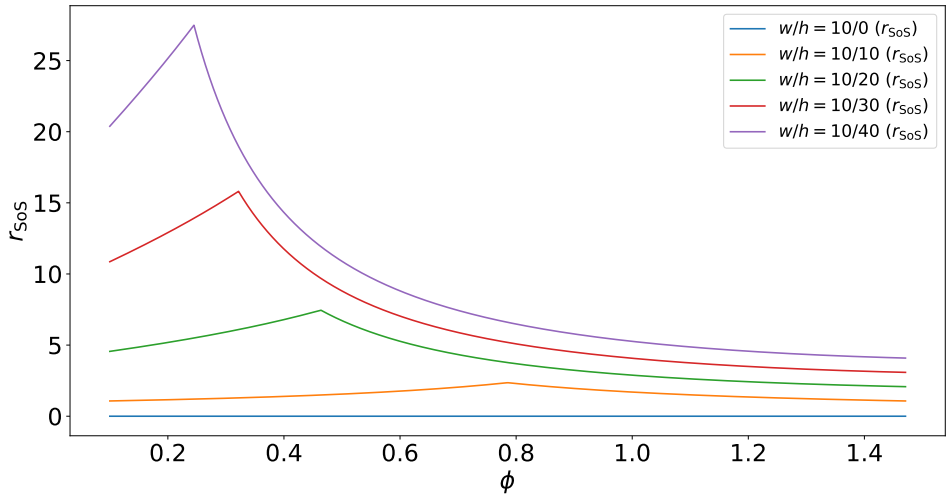
For a visualisation of this, see the plots in Figure 4.14. The result shows that optimising contrast with sparsity-rewarding functions can allow convergence to the true trajectory of pure line segments, whereas magnitude rewarding functions cannot.

4.5.2 Weighted Sums

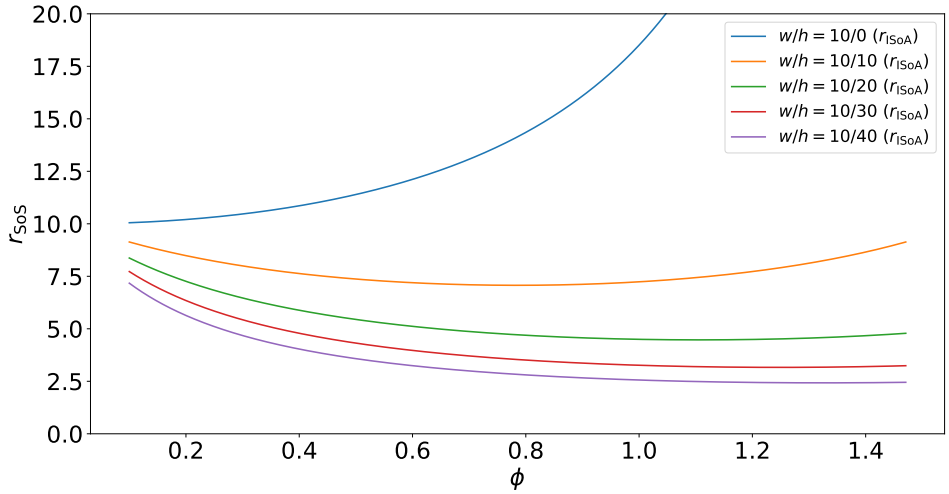
Since the errors presented in Section 4.4 had varying standard deviations, we thought it would be worth looking at how weighted combinations of magnitude- and sparsity-rewarding function perform. To this end we present the following plots in Figure 4.15. To generate these we estimated the optic flow of the office sequence (see Section 4.4.3) various times using different weighted combinations of the r_{SoS} and r_{SoSA}. The plots indicate that with no noise it is better to use only r_{SoSA} and otherwise to use a linear combination. If there is a lot of noise it is better to use r_{SoS}. This supports the conclusions drawn in the rest of the chapter. Note that the r_{R1} and r_{R2} reward are still better than these linear combinations.

4.5.3 Blurring σ

In practice, convergence in optimising the contrast is greatly aided by applying a blurring kernel to the image of warped events. In this experiment, we aim to discover which value of σ is best for which reward. To do this we estimated optical flow on the circle sequence (Section 4.4.2) using various blur

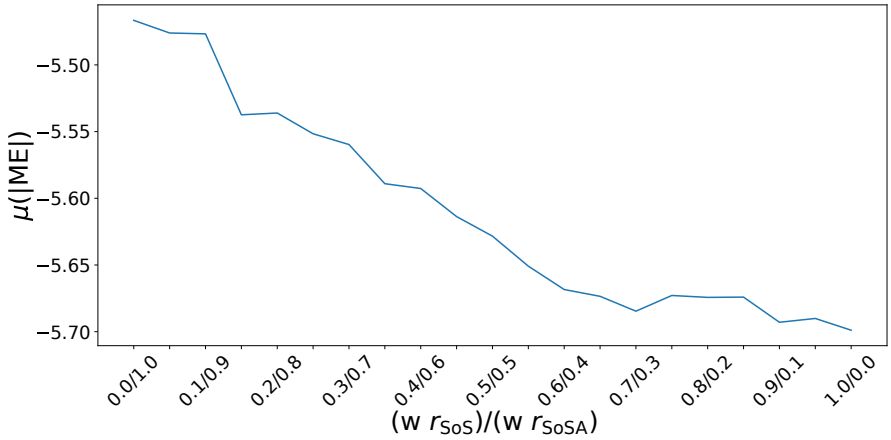


(a) Plot of the r_{SoS} for various ratios of w/h, over $0 \leq \phi \leq \pi/2$.

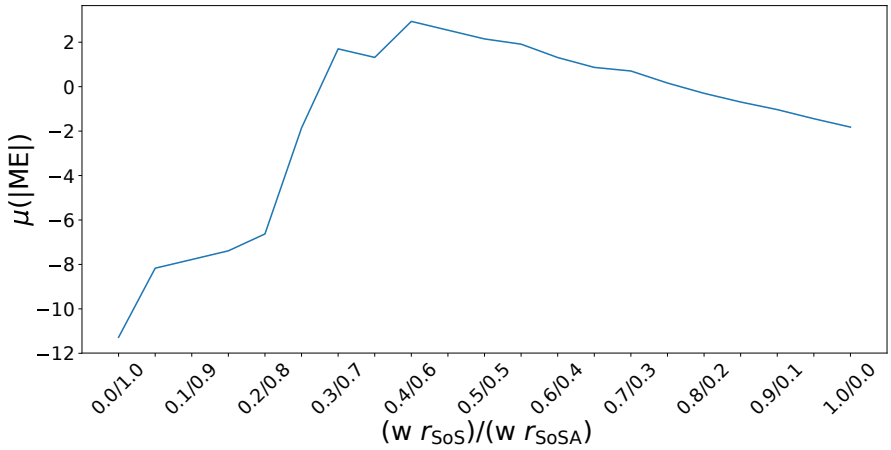


(b) Plot of the r_{SoA} for various ratios of w/h, over $0 \leq \phi \leq \pi/2$

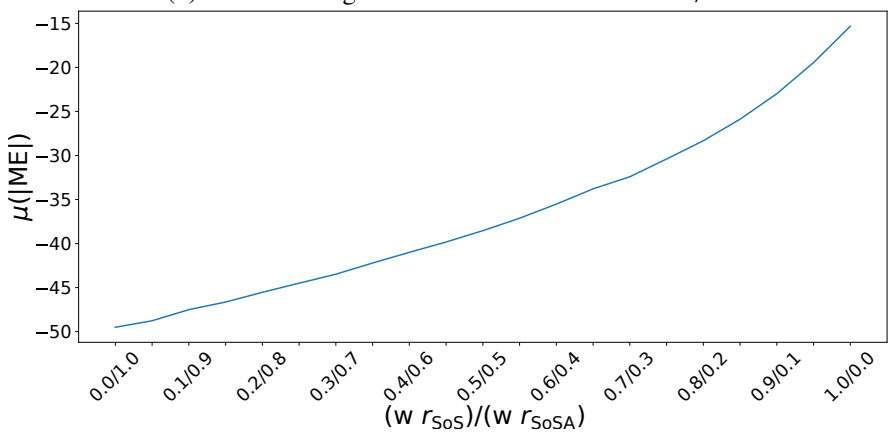
Figure 4.14: Comparison of theoretical convergence behaviour of r_{SoS} vs r_{SoA}.



(a) Error for weighted sum for event/noise ratio $1/0$ (no noise).



(b) Error for weighted sum for event/noise ratio $1/2$.



(c) Error for weighted sum for event/noise ratio $1/10$.

Figure 4.15: Plots for the error of the estimates vs the sum weights for weighted sums of r_{SoS} and r_{SoA} at different event to noise ratios.

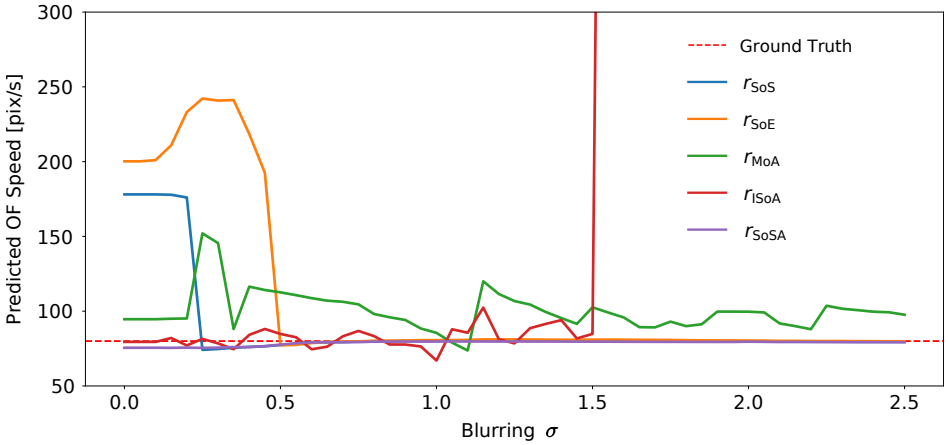


Figure 4.16: Plot for the velocity estimates of various rewards for various values of σ.

sigmas. As can be seen in Figure 4.16, the different rewards have quite different reactions to different degrees of blurring. However, a value of $\sigma = 1$ seems to give reasonable results across all of the rewards and is the value we used in other experiments.

4.6 Discussion

In this chapter we showed that the underlying assumption made about FO is that good trajectory estimates are those where events accumulate in fewer locations. From this observation we were able to devise two categories of reward function, sparsity and magnitude rewarding functions. Conventionally variance or sum of squares (r_{SoS}) has been used to maximise the contrast, but from our categorisation we created four other rewards.

Since the different rewards have different standard deviations with regard to their errors, it may make sense to use a weighted average of rewards or even sensor-fusion techniques such as the Extended Kalman Filter to further improve the results of combining rewards.

We touched upon the issue of how many events are needed to make good predictions and how this quantity can be estimated. We showed which kind of data is likely to cause errors due to aperture uncertainty, analogous to aperture problem in conventional image processing. We then showed that sparsity-rewarding rewards are much less susceptible to this uncertainty, but are also more prone to errors introduced by a noisy event stream. We tested various derived reward functions on real data and confirmed the hypothesised traits of sparsity and magnitude rewarding functions. In order to capitalise on the properties of both categories, we combined several of our reward functions to create the r_{R1} and r_{R2} rewards.

Experimentally, these perform better than any one of the individual rewards under different conditions of noise and object shape. Thus, we hope that this work will aid future event-based vision research, providing better reward functions and stimulating discussion about what these rewards fundamentally do.

4.6.1 Resources

Video, code and dataset: `https://timostoff.github.io/19ICCV`

Chapter 5

Dense Optic Flow using Deep Learning

Based on [105]

Event cameras are paradigm-shifting novel sensors that report asynchronous, per-pixel brightness changes called 'events' with unparalleled low latency. This makes them ideal for high speed, high dynamic range scenes where conventional cameras would fail. Recent work has demonstrated impressive results using Convolutional Neural Network (CNN)s for video reconstruction and optic flow with events. In this chapter we present strategies for improving training data for event-based CNNs that result in 20-40 % boost in performance of existing State of the Art (SotA) video reconstruction networks retrained with our method, and up to 15 % for optic flow networks (Figure 5.1). While many methods exist for computing event-dense optic flow (i.e. an optic flow vector exists for each event), our network is able to compute fully-dense optic flow (i.e. an optic flow vector exists for each pixel even at locations that do not contain an event). A challenge in evaluating event-based video reconstruction is lack of quality ground truth images in existing datasets. To address this, we present a new **High Quality Frames (HQF)** dataset, containing events and ground truth frames from a Dynamic and Active-pixel VIsion Sensor (DAVIS) 240C that are well-exposed and minimally motion-blurred. We evaluate our method on HQF as well as several existing major event camera datasets.

5.1 Introduction

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. Just as CNNs have set the new SotA in conventional computer vision, recent works have done the same for several tasks in event-based vision.

CNNs came to prominence in the ImageNet competition in 2012, where a deep CNN was applied to a data set of around one million images representing 1000 different classes. The CNN (AlexNet) achieved spectacular results, halving the error rates of the best competing approaches [43]. CNNs can be thought of as highly parameterised, trainable function approximators [32]. They consist of an input layer (in this case a voxel grid formed from events) and an output layer (in this case an image representing video reconstruction frame or optic flow). Connecting input and output are the hidden layers (the number and nature of which depend on the particular architecture), which take the output tensor of the previous layer and transform it to produce a new tensor. Layers are parameterised by weights and are differentiable, which enables efficient training, since the error between the prediction of the network and the actual value of a training example can be *back-propagated* through the network to adjust the weights. Stochastic gradient descent optimisers are used to nudge the network toward a local minimum. CNNs take their name from convolutional layers, which convolve the input with a kernel, whose weights are updated during training. The results of the convolution are passed through an activation function (such as Rectified Linear Unit (ReLU), a function which is $y = 0$ for $x < 0$ and $y = x$ for $x \geq 0$) to introduce non-linearity. These convolutions make CNNs well suited to finding spatial patterns in input tensors. Convolutional layers are usually chained in a CNN, with early layers detecting low level spatial features and later layers convolving with the resulting feature maps to detect

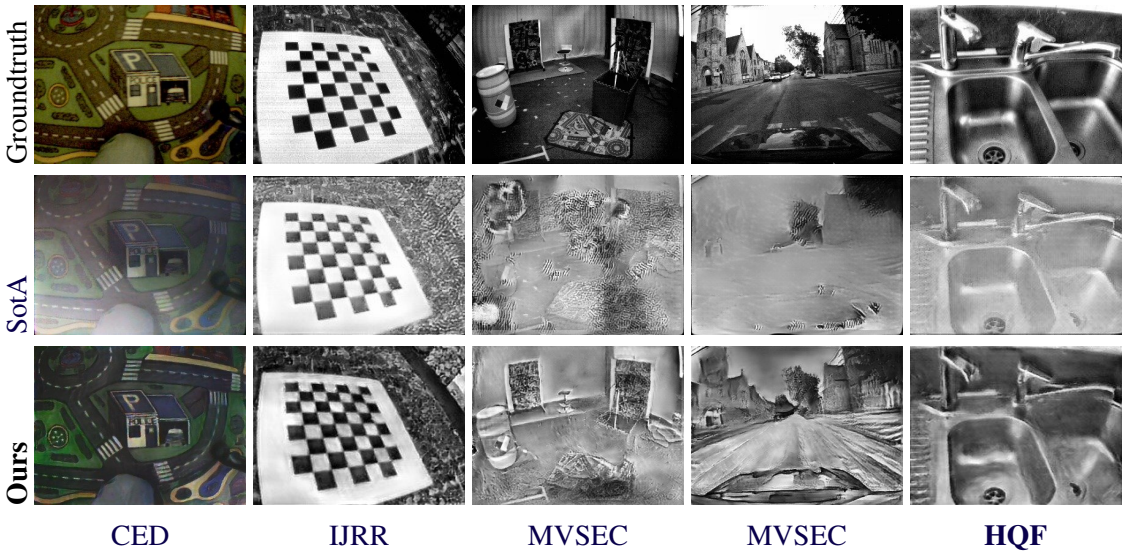


Figure 5.1: Top: ground truth reference image. Middle/bottom: state-of-the-art E2VID [88] vs our reconstructed images from events only. Challenging scenes from event camera datasets: CED [96], IJRR [64], MVSEC [118] and our **HQF** dataset.

high level features. One issue with CNNs is that they are not naturally well suited to dealing with sequential data which requires memory (e.g. speech recognition). A naive approach is to concatenate the network output with the next input to form a Recurrent Neural Network (RNN), however such networks tend to have poor long term memory performance. More sophisticated approaches such as Long Short-Term Memorys (LSTMs) or Gated Recurrent Units (GRUs) contain gates, which allow the network to decide which information to propagate through the next timestep. Gates can be thought of as sub-networks which control which information to forget, which new information to add to the network state and how to combine this information to form a new output. Sub-networks which exhibit recurrence are called *recurrent units*.

[120] propose an unsupervised network able to learn optic flow from real event data, while [87, 88] produced SotA video reconstructions from events. [87] also showed that supervised networks trained on synthetic events transferred well to real event data. Simulation shows promise since data acquisition and ground truth are easily obtainable, in contrast to using real data. However, mismatch between synthetic and real data degrades performance, so a key challenge in producing more accurate results from CNNs is simulating realistic data.

We generate training data that better matches real event camera data by analysing the statistics of existing datasets to inform our choice of simulation parameters. A major finding is that the contrast threshold (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, we observe that the apparent contrast threshold of real event cameras varies greatly, even within one dataset. Previous works such as event-based video reconstruction [88] 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 [120] may be retrained to match any real event camera - at the cost of new data collection and training. We 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. We also propose a simple noise model which yields up to 10 % improvement when added during training.

A challenge in evaluating image and video reconstruction from events is lack of quality ground truth images registered and time-synchronised to events, because most existing datasets focus on scenarios where event cameras excel (high speed, High Dynamic Range (HDR)) and conventional

cameras fail. To address this limitation, we introduce a new High Quality Frames (HQF) dataset that provides several sequences in well lit environments with minimal motion blur. These sequences are recorded with a DAVIS 240C 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. We quantitatively evaluate our method on two major event camera datasets: Event Camera Dataset and Simulator (IJRR) [64] and HQF [118], in addition to our HQF, demonstrating gains of 20-40 % for video reconstruction and up to 15 % for optic flow when we retrain existing SotA networks.

5.1.1 Contributions

We present a method to generate synthetic training data that improves generalisability to real event data, guided by statistical analysis of existing real datasets. We additionally propose a simple method for dynamic train-time noise augmentation that yields up to 10 % improvement for video reconstruction. Using our method, we retrain several network architectures from previously published works on video reconstruction [88, 95] and optic flow [120, 121] from events. We are able to show significant improvements that persist over architectures and tasks. Thus, we believe our findings will provide invaluable insight for others who wish to train models on synthetic events for a variety of tasks. We provide a new comprehensive High Quality Frames dataset targeting ground truth image frames for video reconstruction evaluation. Finally, we provide our data generation code, training set, training code, and our pretrained models, together with dozens of useful helper scripts for the analysis of event-based datasets to make this task easier for fellow researchers.

In summary, our major contributions are:

- A method for simulating training data that yields 20-40 % and up to 15 % improvement for event-based video reconstruction and optic flow CNNs.
- Dynamic train-time event noise augmentation.
- A novel High Quality Frames dataset.
- Extensive analysis and evaluation of our method.
- An optic flow evaluation metric *Flow Warp Loss (FWL)*, tailored to event data, that does not require ground truth flow.
- Open-source code, training data and pretrained models.

5.1.2 Individual Contribution

The contributions to the work made by myself and Cedric Scheerlinck are approximately equal. I contributed more to the training data generation, the analysis and CT estimation of previous datasets, and the optic flow network training and evaluation. Cedric did more on the image reconstruction training and evaluation, train-time augmentation, and ablation/FireNet studies.

5.2 Related Works

5.2.1 Video Reconstruction

Video and image reconstruction from events has been a popular topic in the event-based vision literature. Several approaches have been proposed in recent years; [38] used an Extended Kalman Filter (EFK) to reconstruct images from a rotating event camera, later extending this approach to full 6-Degree of Freedom (DoF) camera motions [39]. [3] used a sliding spatiotemporal window of events to simultaneously optimise both optic flow and intensity estimates using the primal-dual algorithm,

although this method remains sensitive to hyperparameters. [66] proposed direct integration with periodic manifold regularisation on the Surface of Active Events (SAE) [61] to reconstruct video from events. [93, 94] achieved computationally efficient, continuous-time video reconstruction via complementary and high-pass filtering. This approach can be combined with conventional frames, if available, to provide low frequency components of the image. However, if taken alone, this approach suffers from artefacts such as ghosting effects and bleeding edges.

Recently, CNNs have been brought to bear on the task of video reconstruction. [87, 88] presented Events to Video (E2VID), a recurrent network that converts events (discretised into a voxel grid) to video. A temporal consistency loss based on [41] was introduced to reduce flickering artefacts in the video, due to small differences in the reconstruction of subsequent frames. E2VID is current SotA. were able to reduce model complexity by 99 % with the *FireNet* architecture [95], with only minor trade-offs in reconstruction quality, enabling high frequency inference.

5.2.2 Optic Flow

Since event-based cameras are considered a good fit for applications involving motion [24], much work has been done on estimating optic flow with event cameras [1, 3, 5, 6, 10, 26, 51, 102, 104]. Recently, [120] proposed a CNN (EV-FlowNet) for estimating optic flow from events, together with the Multi Vehicle Stereo Event Camera (MVSEC) dataset [118] that contains ground truth optic flow estimated from depth and ego-motion sensors. The input to EV-FlowNet is a 4-channel image formed by recording event counts and the most recent timestamps for negative and positive events. The loss imposed on EV-FlowNet was an image-warping loss [113] that took photometric error between subsequent APS frames registered using the predicted flow. A similar approach was taken by [111], in a network that estimated depth and camera pose to calculate optic flow. In [121] the authors improved on prior work by replacing the image-warping loss with an event-warping loss that directly transports events to a reference time using the predicted flow. We use a similar method to evaluate optic flow performance of several networks (see Section 5.4.1). [121] also introduced a novel input representation based on event discretisation that places events into bins with temporal bilinear interpolation to produce a voxel grid. EV-FlowNet was trained on data from MVSEC [118] and [111] even trained, then validated on the same sequences; our results (Section 5.4.1) indicate that these networks suffer from overfitting.

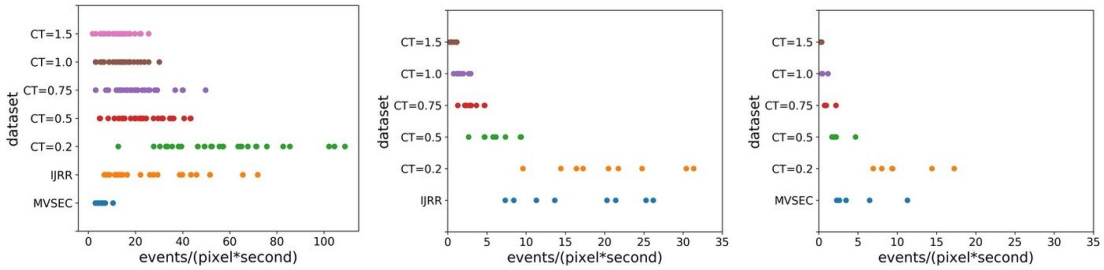
5.2.3 Input Representations

To use conventional CNNs, events must first be transformed into an amenable grid-based representation. While asynchronous spiking neural networks can process raw events and have been used for object recognition [44, 74, 81] and optic flow [5, 6], the lack of appropriate hardware or effective error backpropagation techniques renders them uncompetitive with SotA CNNs. Several grid-based input representations for CNNs have been proposed: simple event images [55, 120] (events are accumulated to form an image), SAE [120] (latest timestamp recorded at each pixel), Histogram of Averaged Time Surfaces (HATS) [99] and even learned input representations, where events are sampled into a grid using convolutional kernels [28]. [121] and [88] found best results using a voxel grid representation of events, where the temporal dimension is essentially discretised into B bins in a three-dimensional grid (Equation A.1).

5.3 Method

5.3.1 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 λ_{CT}. Event generation is also limited by



(a) IJRR/MVSEC vs ESIM (b) IJRR vs ESIM (c) MVSEC vs ESIM

Figure 5.2: Each dot represents a sequence from the given dataset (y-axis). 5.2a $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of IJRR and MVSEC vs. ESIM training datasets (CT 0.2-1.5) described in Section 5.3.2. 5.2b $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of IJRR vs. ESIM events simulated from IJRR *APS frames*. 5.2c $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of MVSEC vs. ESIM events simulated from MVSEC *APS frames*.

the refractory period t_r, the time after the firing of an event for which the pixel is blocked from firing another event (i.e. $t - t_{i-1} > t_r$). λ_{CT} is referred to as the Contrast Threshold (CT) and can be typically adjusted in modern event cameras. In reality, the CTs are usually different for the positive (λ^+_{CT}) and negative (λ^-_{CT}) thresholds, (thus λ_{CT} really refers to two separate thresholds $\lambda_{\text{CT}} = \{\lambda^-_{\text{CT}}, \lambda^+_{\text{CT}}\}$), nor are the values for λ_{CT} constant in time or homogeneous over the image plane. In simulation (e.g. using the Event Camera Simulator [85] (ESIM) [85]), CTs are typically sampled from the normal distribution $\mathcal{N}(\mu$=0.18, σ=0.03$)$ to model this variation [28, 87, 88]. The CT is an important simulator parameter since it determines the number and distribution of events generated from a given scene. We found that the refractory period made essentially no difference in simulation, unless it was raised to absurdly high values.

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 (see Section 5.3.1, we found that tuning the simulator CTs to match the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ 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), we generated a diverse variety of realistic scene dynamics. The result of this experiment (Figure 5.2a) indicates that a contrast threshold setting of between 0.2 and 0.5 would be more appropriate for sequences from the IJRR dataset [64]. The larger diversity of motions is also apparent in the large spread of the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ compared to MVSEC [118] whose sequences are tightly clustered.

As an alternative experiment to determine CTs of existing datasets, we measured the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of events simulated using the actual APS (ground truth) 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. 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 (Figure 5.3), so we were limited to using this approach on carefully curated snippets (Figure 5.4). The results of this experiment in Figure 5.2b and 5.2c indicate similar results of lower contrast thresholds for IJRR and higher for MVSEC, although accuracy is limited by the poor quality APS frames.

Additional means of estimating CT

As outlined in Section 5.3, we propose several methods for estimating the CTs for a given event-based dataset. In total, we tried three different methods:

- Creating a simulator scene with similar texture and range of motions as the real sequence and adjusting the CTs until the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ match.

(a) IJRR (b) IJRR (c) MVSEC (d) MVSEC (e) HQF

Figure 5.3: 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 our own High Quality Frames dataset.

(a) Poorly exposed from IJRR and MVSEC

(b) Well exposed from IJRR and MVSEC

Figure 5.4: Examples of frames from IJRR and MVSEC after local histogram equalisation, with poorly exposed sequences in 5.4a, and better exposed images in 5.4b.

- Simulating events from the APS frames (if they are available) and adjusting the CTs until the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ match the real sequence.
- Creating a calibration scene in the simulator, recording this scene on-screen and adjusting the CT of the event camera to match the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of the simulation.

We describe the first two approaches in the main chapter. These approaches indicate CTs of approximately 0.3 and 0.75 for IJRR and MVSEC respectively.

We also produced a calibration sequence in an attempt to match the simulator to our particular DAVIS 240C at default settings. For this, we moved a checkerboard across the image plane in ESIM, using various CTs. The scene was played on a high-refresh screen and recorded by our DAVIS. The resulting event-rate for each sequence, shown in Table 5.1, suggest a $\lambda_{\text{CT}} \approx 0.5$ to match the camera to the simulator. This is however in conflict with the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ (8.2) of the real sequences compared to the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of the best training data CT (19.6). In other words, there seems to be a mismatch for this method of calibration, perhaps stemming from a difference in recording events from real scenes to recording scenes from a screen as was done for the checkerboard calibration sequence.

Table 5.1: Comparison of the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ for simulated sequences at various CT settings with the $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ of a real calibration sequence. The sequence consists of a checkerboard in motion. The same sequence is also recorded by a real event camera (DAVIS 240C) using default bias settings. The result suggests that a CT value around 0.5 would be appropriate to match the simulator to the real camera.

CT	0.2	0.5	0.75	1.0	1.5	Real
$\frac{\text{events}}{\text{pix}\cdot\text{s}}$	26.5	19.6	16.2	10.8	7.2	8.2
LPIPS	0.289	0.285	0.289	0.311	0.316	-

5.3.2 Training Data

We used an event camera simulator, ESIM [85] to generate training sequences for our network. There are several modes of simulation available, of which we used 'Multi-Object-2D' that facilitates moving images in simple 2D motions, restricted to translations, rotations and dilations over a planar background. This generates sequences reminiscent of Flying Chairs [17], where objects move across the screen at varying velocities. In our generation scheme, we randomly selected images from Common Objects in COntext (COCO) dataset [47] and gave them random trajectories over the image plane. Our 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 encourages networks to generalise to arbitrary real world camera motions, since a wide range of scene dynamics are presented during training. Sequences were 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, the positive threshold $\lambda^{+}_{\text{CT}} = \lambda^{-}_{\text{CT}} \cdot \alpha, \alpha \in \mathcal{N}(\mu = 1.0, \sigma = 0.1)$.

The events thus generated were discretised into a voxel grid representation. In order to ensure synchronicity with the ground truth frames of our training set and later with the ground truth frames of our validation set, we always take all events between two frames to generate a voxel grid. Given N_e events $e_i = \{x_i, y_i, t_i, s_i\}_{i=0,\dots,N_e}$ spanning $\Delta t = t_{N_e} - t_0$ seconds, a voxel grid V with B bins can be formed through temporal linear interpolation via

$$V_{k\in[0,B-1]} = \sum_{i=1}^{N_e} s_i \max(0, 1 - |t_i^* - k|) \tag{5.1}$$

where t_i^* is the timestamp normalised to the range $[0, B-1]$ via $t_i^* = \frac{t_i - t_0}{\Delta t}(B-1)$ and the bins are evenly spaced over the range $[t_0, t_{N_e}]$. 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. Thus, it is important to train the network on a large range of event rates $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ and voxel densities. During inference, other strategies of voxel generation can be employed, as further discussed in Appendix A.1.1. We used $B = 5$ throughout the experiments in this chapter. In earlier experiments we found values of $B = 2, 5, 15$ produced no significant differences.

5.3.3 Sequence Length

To train recurrent networks, we sequentially passed λ_L inputs to the network and computed the loss for each output. Finally, the losses were summed and a backpropagation update was performed based on the gradient of the final loss with respect to the network weights. Since recurrent units in the network are initialised to zero, lower values of λ_L restrict the temporal support that the recurrent units see at train time. To investigate the impact of sequence length λ_L, we retrain our networks

using $\lambda_L = 40$ (as in E2VID [88]) and $\lambda_L = 120$. In the case of non-recurrent networks such as EV-FlowNet [120, 121], we ignore the sequence length parameter.

5.3.4 Loss

For our primary video reconstruction loss function we used 'learned perceptual image patch similarity' (Learned Perceptual Image Patch Similarity (LPIPS)) [116]. LPIPS is a fully differentiable similarity metric between two images that compares hidden layer activations of a pretrained network (e.g. Alex-Net or Visual Geometry Group network (VGG)), and was shown to better match human judgment of image similarity than photometric error or Structural SIMilarity (SSIM) [110]. Since our event tensors were synchronised to the ground truth image frames by design (the final event in the tensor matches the frame timestamp), we computed the LPIPS distance between our reconstruction and the corresponding ground truth frame. As recommended by the authors [116], we used the Alex-Net variant of LPIPS. We additionally imposed a temporal consistency loss [41] that measures photometric error between consecutive images after registration based on optic flow, subject to an occlusion mask. For optic flow, we used the L1 distance between our prediction and ground truth as the training loss.

5.3.5 Data Augmentation

During training, [88] 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 $P(q_p|q_r) = 0.05$, while the probability of maintaining the paused state when the sequence is already paused $P(q_p|q_p) = 0.9$ 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. We used pause augmentation to train all recurrent networks.

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, we applied a refractory period of 1 ms. At train time, for each sequence of λ_L input event tensors we optionally added zero-mean Gaussian noise ($\mathcal{N}(\mu$=0, $\sigma = 0.1)$) to the event tensor to simulate uncorrelated background noise, and randomly elected a few 'hot' pixels. The number of hot pixels was 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, σ=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, we retrained several models with and without noise (Table 5.19).

5.3.6 Architecture

To isolate the impact of our method from choice of network architecture, we retrained SotA video reconstruction network E2VID [88] and SotA optic flow network EV-FlowNet described in [120, 121]. Thus, differences in performance for each task are not due to architecture. Additionally, we aim to show that our method generalises to multiple architectures. While we believe architecture search may further improve results, it is outside the scope of this work.

5.3.7 High Quality Frames Dataset

To evaluate event camera image reconstruction methods, we compared reconstructed images to temporally synchronised, registered ground truth reference images. Event cameras such as the DAVIS [9] can capture image frames (in addition to events) that are timestamped and registered to the events, that may serve as ground truth. Previous event camera datasets such as IJRR [64] and MVSEC [118]

Table 5.2: Breakdown of the 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

contain limited high quality DAVIS frames, while many frames are motion-blurred and or under/over-exposed (Figure 5.3). As a result, [88] manually rejected poor quality frames, evaluating on a smaller subset of IJRR.

We present a new HQF aimed at providing ground truth DAVIS frames that are minimally motion-blurred and well exposed. In addition, our 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, we used two different DAVIS 240C sensors to capture data with different noise/CT characteristics. We used default bias settings loaded by the Dynamic Vision Sensor (DVS) Robot Operating System (ROS) driver from Robotics and Perception Group, Zurich (RPG) [1], and set exposure to either auto or fixed to maximise frame quality. Our HQF provides temporally synchronised, registered events and DAVIS frames (further details in Table 5.2).

5.4 Experiments

5.4.1 Evaluation

We evaluated our method by retraining two state-of-the-art event camera neural networks: E2VID [87, 88], and EV-FlowNet [120, 121]. Our method outperforms previous state-of-the-art in image reconstruction and optic flow on several publicly available event camera datasets including IJRR [64] IJRR and MVSEC [118], and our new HQF.

For video reconstruction on the datasets HQF, IJRR and MVSEC (Table 5.4) we obtained a 40 %, 20 % and 28 % improvement over E2VID [88] respectively, using LPIPS. For optic flow we obtained a 12.5 %, 10 % and 16 % improvement over EV-FlowNet [120] on flow warp loss (Flow Warp Loss (FWL), Equation 5.3). Notably, EV-FlowNet was trained on MVSEC data (`outdoor_day2` sequence), while ours was trained entirely on synthetic data, demonstrating the ability of our method to generalise to real event data.

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

Table 5.3: Start and end times for sequences in IJRR and MVSEC that we present validation statistics on. While both IJRR and MVSEC contain more sequences than the ones listed, those not included had very low quality accompanying frames (see Figure 5.3).

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			

Image

As in [88] we compared our reconstructed images to ground truth (DAVIS frames) on three metrics; mean squared error (Mean Squared Error (MSE)), structural similarity [110] (SSIM) and perceptual loss [115] (LPIPS) that uses distance in the latent space of a pretrained deep network to quantify image similarity.

Since many of these datasets show scenes that are challenging for conventional cameras, we carefully selected sections of those sequences where frames appeared to be of higher quality (less blurred, better exposure etc.). The exact cut times of the IJRR and MVSEC sequences can be found in the Table 5.3. However, we were also ultimately motivated to record our own dataset of high quality frames (HQF, Section 5.3.7) of which we evaluated the entire sequence.

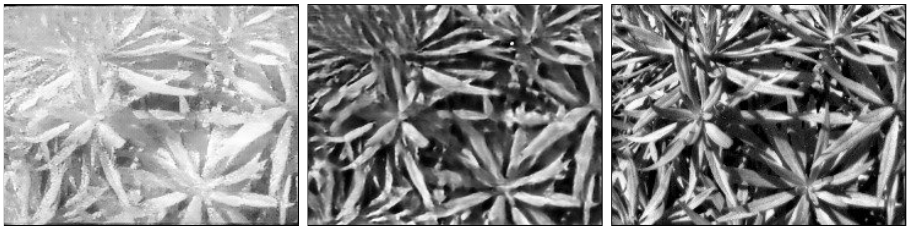
Table 5.4: Comparison of SotA methods of video reconstruction and optic flow to networks trained using our dataset on HQF, IJRR and MVSEC. Best in bold.

Sequence	MSE		SSIM		LPIPS		FWL	
	E2VID	Ours	E2VID	Ours	E2VID	Ours	EVFlow	Ours
				HQF				
bike_bay_hdr	0.16	**0.0299**	0.41	**0.5202**	0.51	**0.3038**	1.22	**1.2302**
boxes	0.11	**0.0345**	0.50	**0.5923**	0.38	**0.2575**	1.75	**1.8020**
desk_6k	0.15	**0.0300**	0.51	**0.5966**	0.39	**0.2213**	1.23	**1.3515**
desk_fast	0.12	**0.0354**	0.54	**0.6062**	0.40	**0.2504**	1.43	**1.4956**
desk_hand_only	0.12	**0.0480**	0.53	**0.5709**	0.63	**0.3864**	**0.9469**	0.85
desk_slow	0.16	**0.0410**	0.53	**0.6218**	0.47	**0.2480**	1.01	**1.0756**
engineering_posters	0.13	**0.0300**	0.42	**0.5710**	0.47	**0.2560**	1.50	**1.6479**
high_texture_plants	0.16	**0.0314**	0.37	**0.6476**	0.38	**0.1392**	0.13	**1.6809**
poster_pillar_1	0.14	**0.0318**	0.38	**0.4990**	0.54	**0.2672**	1.20	**1.2413**
poster_pillar_2	0.15	**0.0350**	0.40	**0.4745**	0.56	**0.2625**	**1.1621**	0.96
reflective_materials	0.13	**0.0334**	0.44	**0.5544**	0.44	**0.2802**	1.45	**1.5748**
slow_and_fast_desk	0.16	**0.0286**	0.48	**0.6237**	0.45	**0.2475**	0.93	**0.9893**
slow_hand	0.18	**0.0375**	0.41	**0.5652**	0.57	**0.3023**	**1.6353**	1.56
still_life	0.09	**0.0261**	0.51	**0.6263**	0.35	**0.2243**	1.93	**1.9815**
Mean	0.14	**0.0326**	0.46	**0.5791**	0.46	**0.2562**	1.20	**1.3540**
				IJRR				
boxes_6dof_cut	**0.0406**	0.04	0.63	**0.6392**	0.29	**0.2479**	1.42	**1.4571**
calibration_cut	0.07	**0.0315**	0.61	**0.6245**	0.22	**0.1805**	1.20	**1.3057**
dynamic_6dof_cut	0.17	**0.0525**	0.45	**0.5275**	0.38	**0.2673**	1.37	**1.3922**
office_zigzag_cut	0.07	**0.0369**	0.49	**0.5082**	0.31	**0.2599**	**1.1302**	1.11
poster_6dof_cut	0.07	**0.0307**	0.60	**0.6567**	0.26	**0.1947**	1.50	**1.5551**
shapes_6dof_cut	0.03	**0.0168**	**0.7982**	0.77	0.26	**0.2234**	1.15	**1.5699**
slider_depth_cut	0.08	**0.0308**	0.54	**0.6240**	0.35	**0.2434**	1.73	**2.1723**
Mean	0.07	**0.0344**	0.61	**0.6364**	0.28	**0.2240**	1.32	**1.4545**
				MVSEC				
indoor_flying1_data_cut	0.25	**0.0840**	0.19	**0.3635**	0.72	**0.4534**	1.02	**1.1380**
indoor_flying2_data_cut	0.23	**0.0931**	0.18	**0.3581**	0.71	**0.4530**	1.13	**1.3592**
indoor_flying3_data_cut	0.25	**0.0896**	0.18	**0.3650**	0.73	**0.4440**	1.06	**1.2291**
indoor_flying4_data_cut	0.21	**0.0784**	0.23	**0.3622**	0.72	**0.4475**	1.24	**1.4999**
outdoor_day1_data_cut	0.32	**0.1274**	0.31	**0.3395**	0.66	**0.5179**	1.15	**1.2719**
outdoor_day2_data_cut*	0.30	**0.0963**	0.29	**0.3394**	0.57	**0.4285**	**1.2149**	1.20
Mean	0.29	**0.1052**	0.27	**0.3461**	0.65	**0.4670**	1.12	**1.29962**

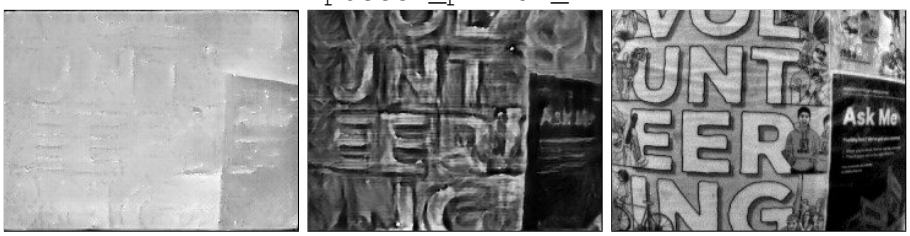
*Removed from mean tally for EV-FlowNet, as this sequence is part of the training set.


E2VID
Ours
Groundtruth
bike_bay_hdr
boxes
desk_6k
desk_fast
desk_hand_only
desk_slow
engineering_posters

high_texture_plants



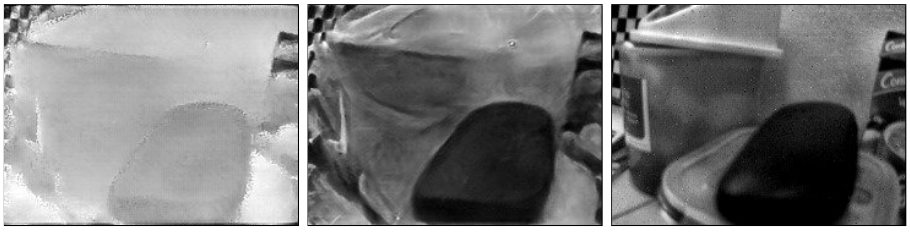
poster_pillar_1



poster_pillar_2

reflective_materials

slow_and_fast_desk



slow_hand

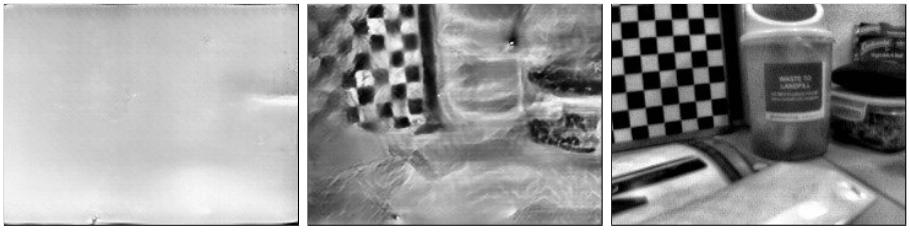
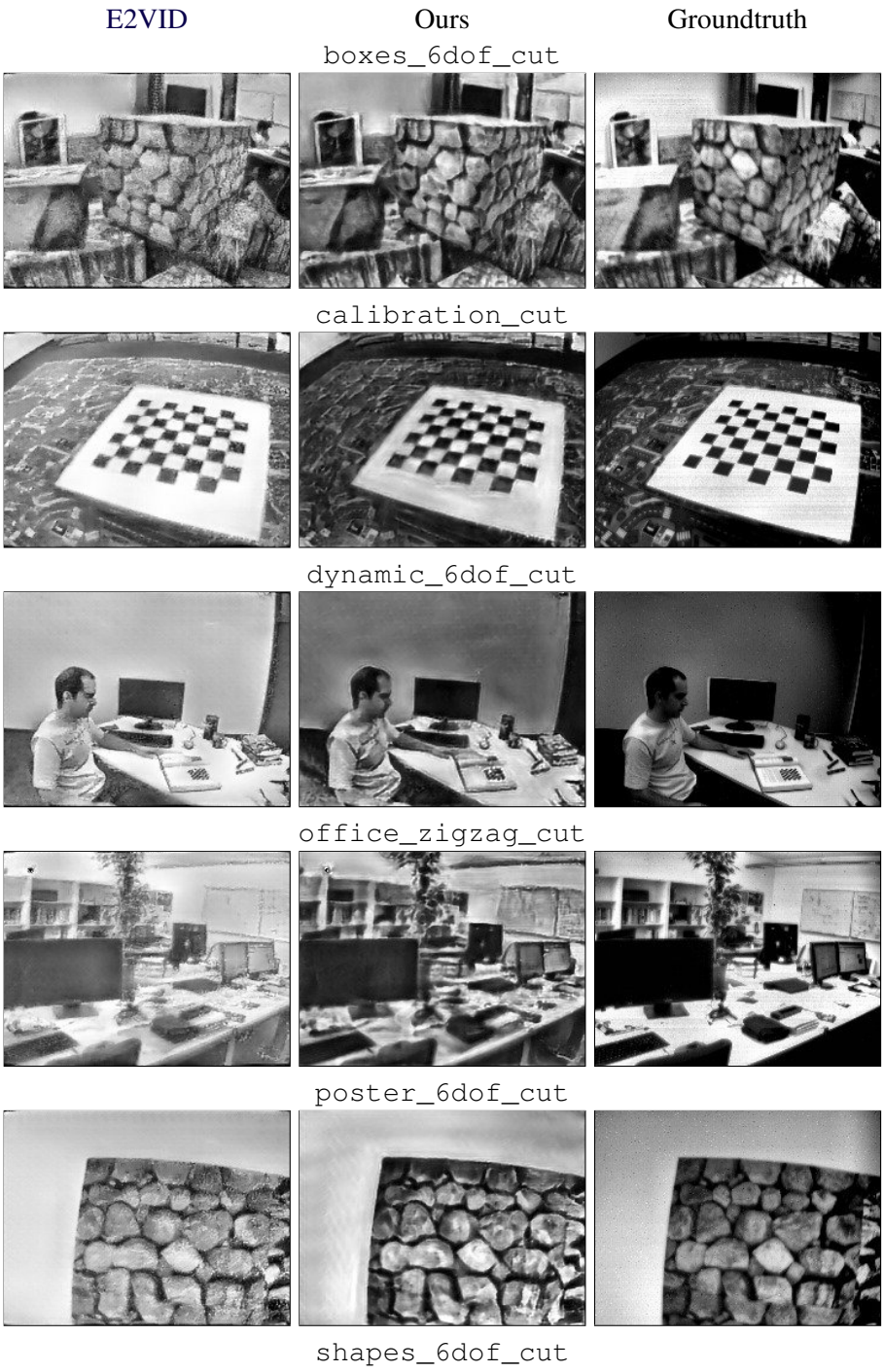


Table 5.5: Qualitative results for HQF. Random selection, not cherry picked.



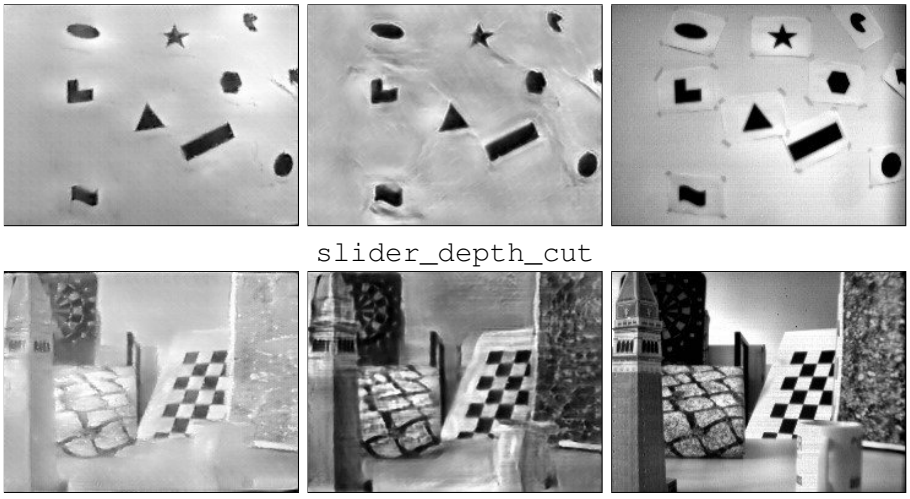


Table 5.6: Qualitative results for IJRR. Random selection, not cherry picked.

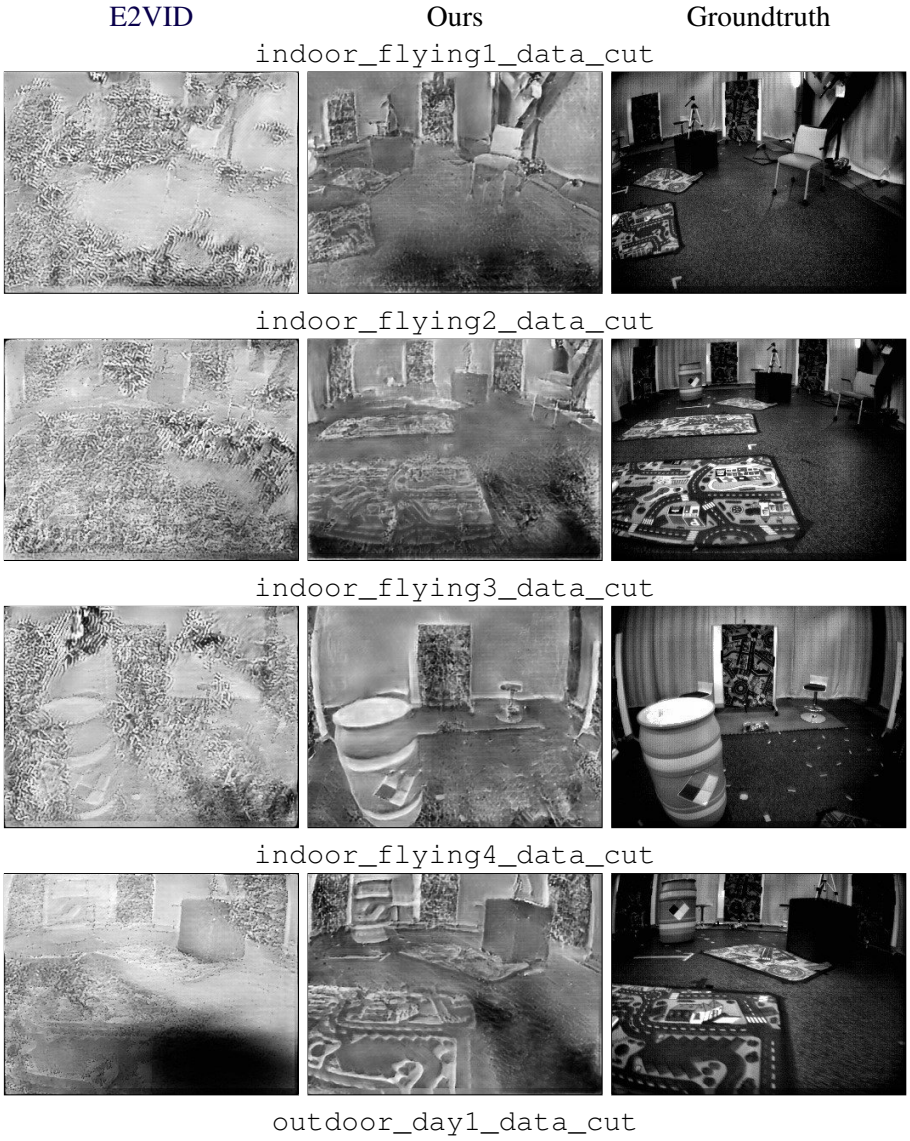


Table 5.7: Qualitative results for MVSEC. Random selection, not cherry picked.

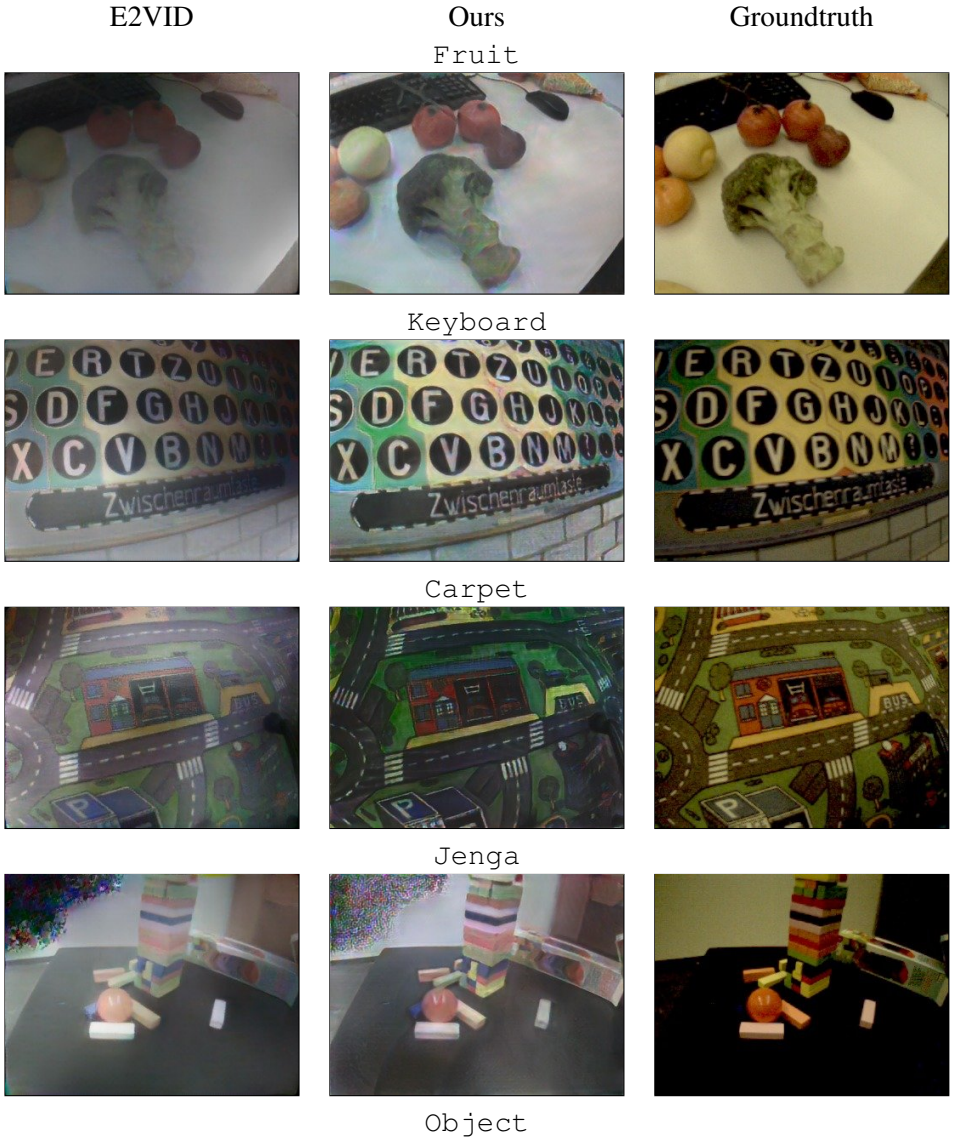


Table 5.8: Qualitative results for Color Event Dataset (CED) [96]. Random selection, not cherry picked. As a matter of interest, the Jenga sequence shows a region of the scene where there is only blank wall, so few events have been generated, resulting in the peculiar artefacts seen in the top left corner.

Flow

A warping loss (similar to [27]) was used as a proxy measure of accuracy as it doesn't require ground truth flow. Events $\mathcal{E} = (x_i, y_i, t_i, s_i)_{i=1,\dots,N}$ are warped by per-pixel optical flow $\vec{v} = (v_x, v_y)^\intercal$ to a reference time t_{ref} via

$$I_\omega = \begin{pmatrix} x_i' \\ y_i' \end{pmatrix} = \begin{pmatrix} x_i \\ y_i \end{pmatrix} + (t_{\text{ref}} - t_i) \begin{pmatrix} v_x \\ v_y \end{pmatrix}. \tag{5.2}$$

where $\omega = \vec{v}$. The resulting Image of Warped Events (IWE) I_ω becomes sharper if the flow is correct, as events are motion-compensated. Sharpness can be evaluated using the variance of the IWE $\sigma^2(I_\omega)$ [25, 103], where a higher value indicates a better flow estimate. Since image variance $\sigma^2(I)$ depends on scene structure and camera parameters, we normalise by the variance of the unwarped event image I_0 to obtain the Flow Warp Loss (FWL):

$$FWL := \frac{\sigma^2(I_\omega)}{\sigma^2(I_0)}. \tag{5.3}$$

$FWL < 1$ implies the flow is worse than a baseline of zero flow. FWL enables evaluation on datasets without ground truth optic flow. While we used ground truth from the simulator during training, we evaluated on real data using FWL (Table 5.4). We believe training on ground truth (L1 loss) rather than FWL encourages fully-dense flow predictions. This is because a network which only produces optic flow estimates where there are events present, will perform equally well as a network which produces flow estimates on every pixel, providing no incentive for the FWL trained network. L1 loss on the other hand will punish incorrect estimates on all pixels.

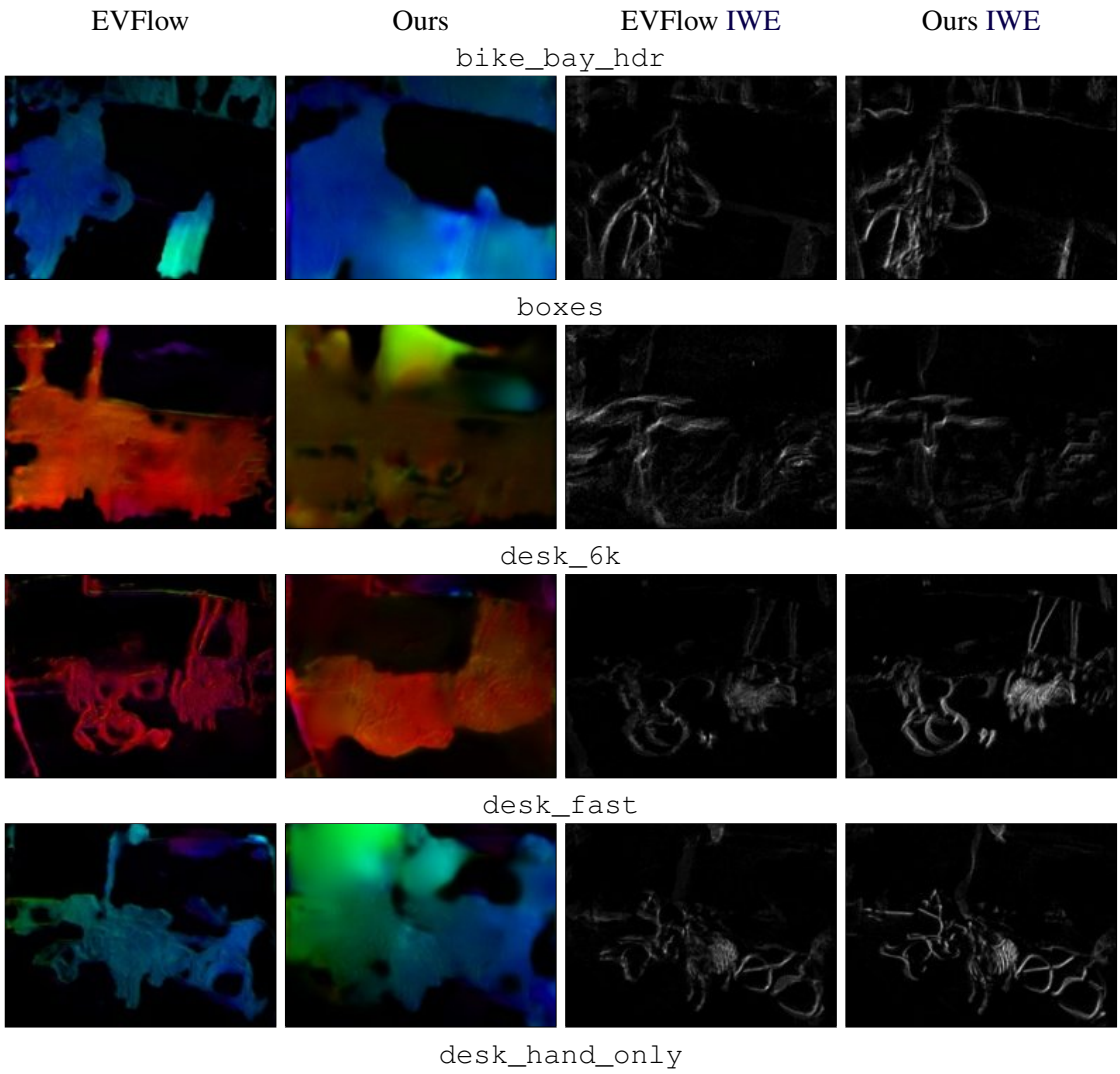
Table 5.9 shows Average Endpoint Error (AEE) of optic flow on MVSEC [118]. MVSEC provides optic flow estimates computed from lidar depth and ego motion sensors as 'ground truth', allowing us to evaluate AEE using code provided in [120]. However, lidar + ego motion derived ground truth is subject to sensor noise, thus, AEE may be an unreliable metric on MVSEC. For example, predicting zero flow achieves near state-of-the-art in some cases on MVSEC using AEE, though not with our proposed metric FWL (by construction, predicting zero flow yields $FWL = 1.0$).

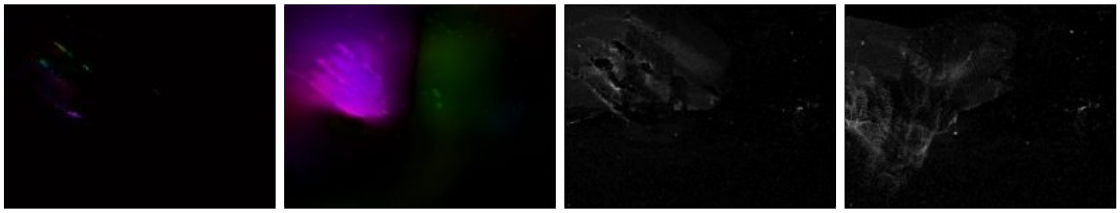
Interestingly, zero loss (doing nothing) is still the best overall at reducing outliers and is a strong contender for AEE (especially in the flying sequences), showing the importance of reporting the relative improvement as in our FWL (by construction, predicting zero flow would yield $FWL = 1.0$).

Table 5.9: Comparison of various methods to optic flow estimated from Lidar depth and ego-motion sensors [118]. The Average Endpoint Error (AEE) to the Lidar estimate 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). The time between frames is dt=1. Zeros is the baseline error resulting from always estimating zero flow.

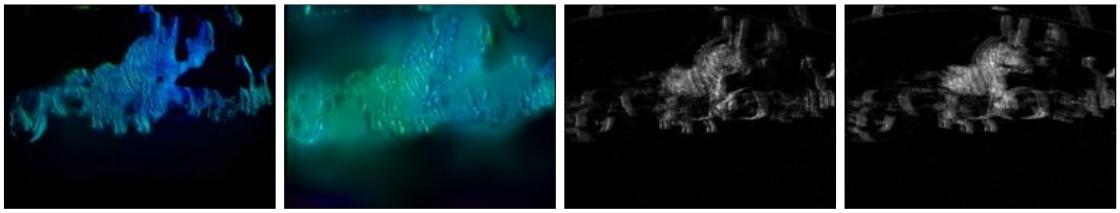
Dataset	outdoor_day1		outdoor_day2		indoor_flying1		indoor_flying2		indoor_flying3	
	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$	AEE	$\%_{\text{Outlier}}$
Zeros	4.31	0.39	1.07	**0.91**	1.10	1.00	1.74	**0.89**	1.50	**0.94**
EVFlow [120]	0.49	0.20	-	-	1.03	2.20	1.72	15.10	1.53	11.90
EVFlow+ [121]	**0.32**	**0.0**	-	-	0.58	**0.0**	1.02	4.00	0.87	3.00
Gehrig [28]	-	-	-	-	0.96	0.91	1.38	8.20	1.40	6.47
Ours	0.68	0.99	**0.82**	0.96	**0.56**	1.00	**0.66**	1.00	**0.59**	1.00
ECN* [111]	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.

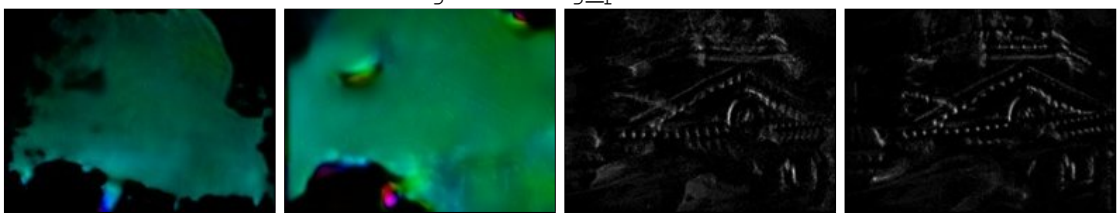




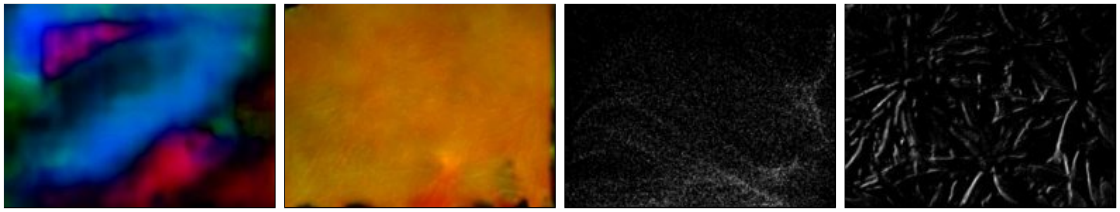
desk_slow



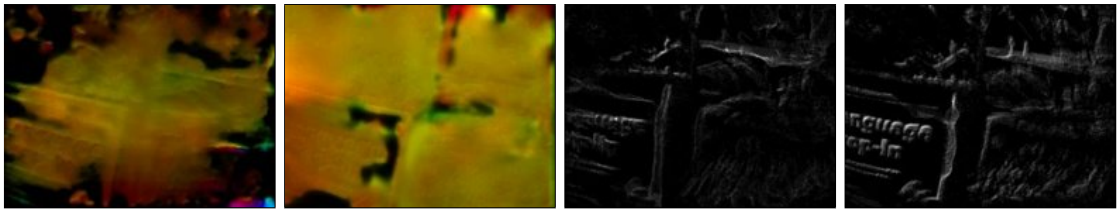
engineering_posters



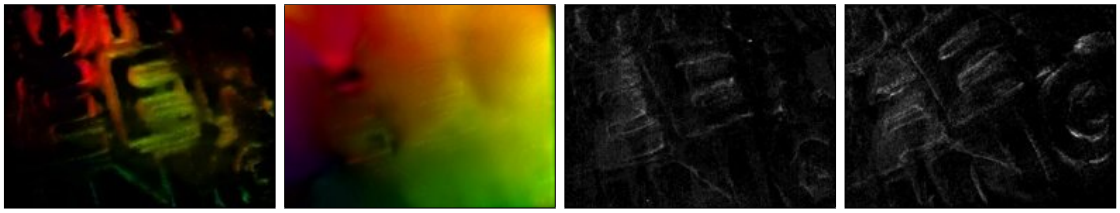
high_texture_plants



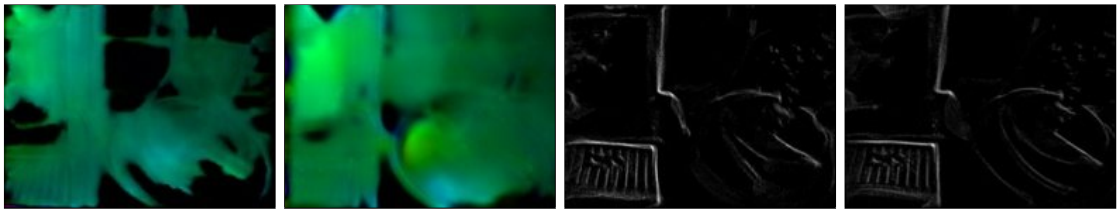
poster_pillar_1



poster_pillar_2



reflective_materials



slow_and_fast_desk

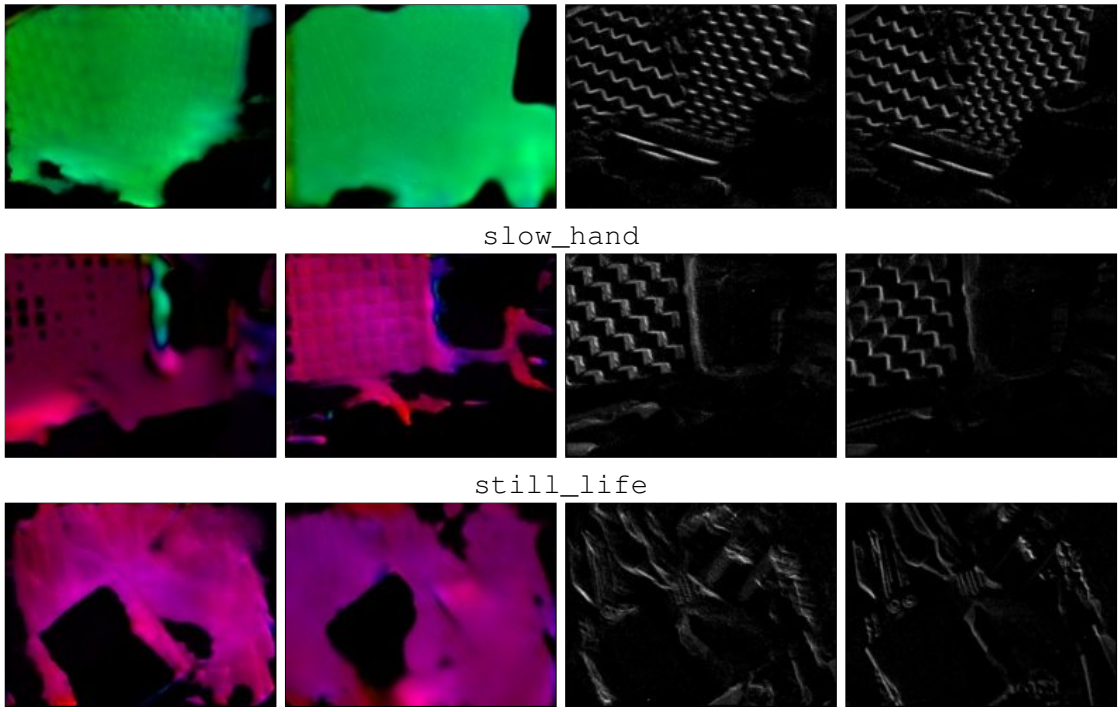
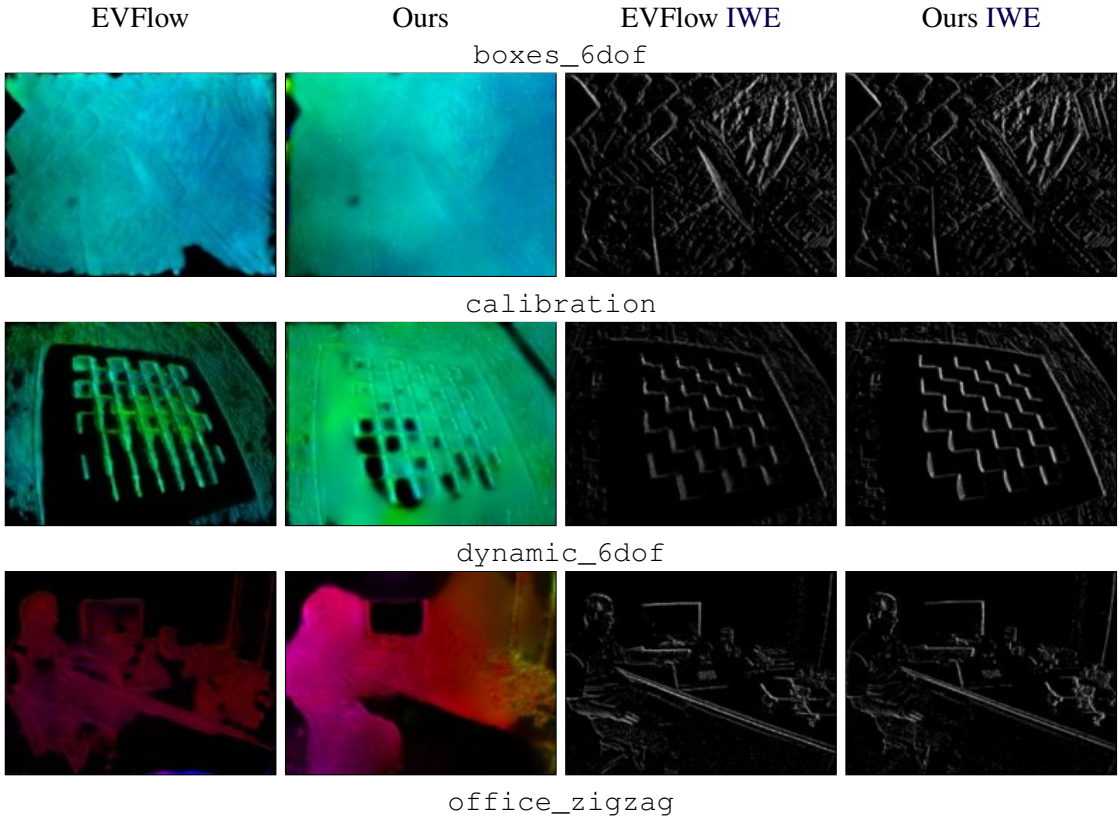


Table 5.10: Qualitative results for HQF. Left: optic flow vectors represented in HSV color space, right: image of warped events (IWE). Random selection, not cherry picked.



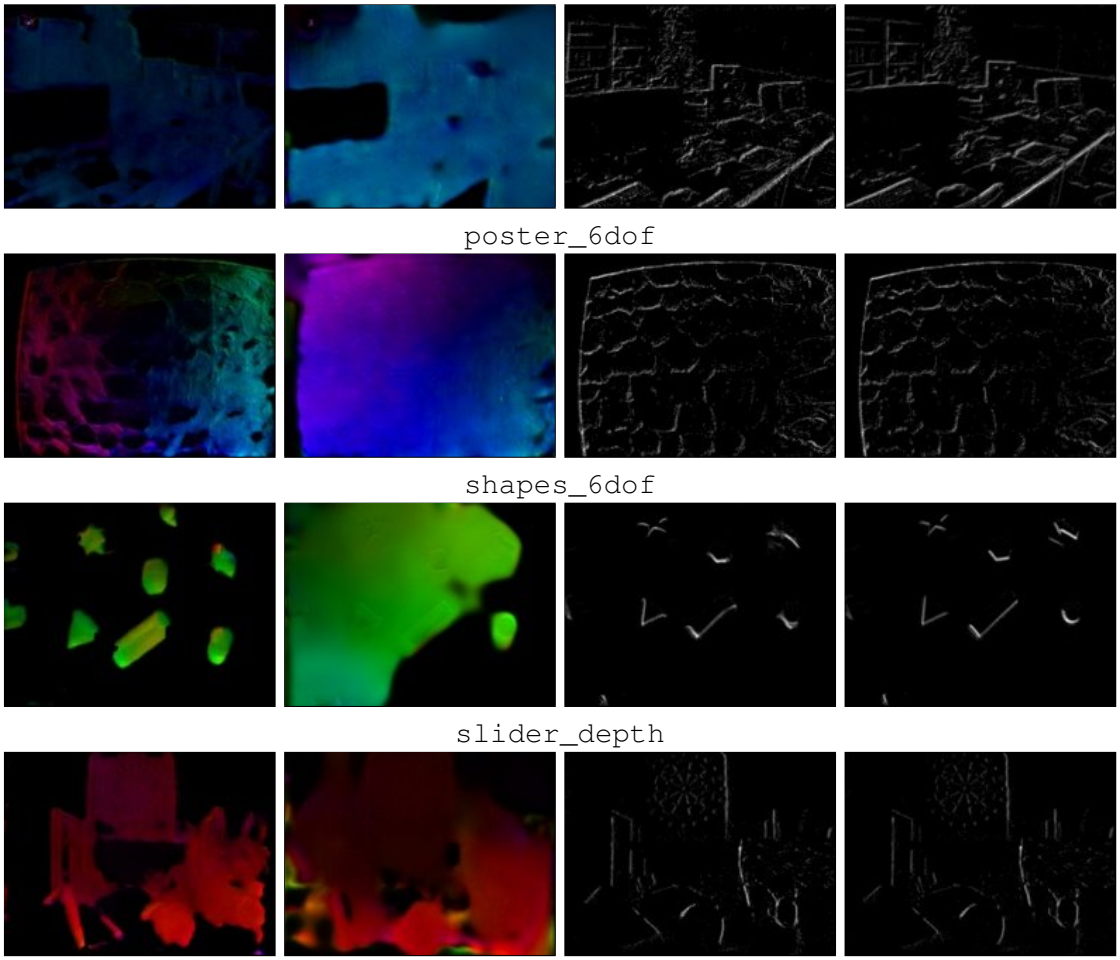
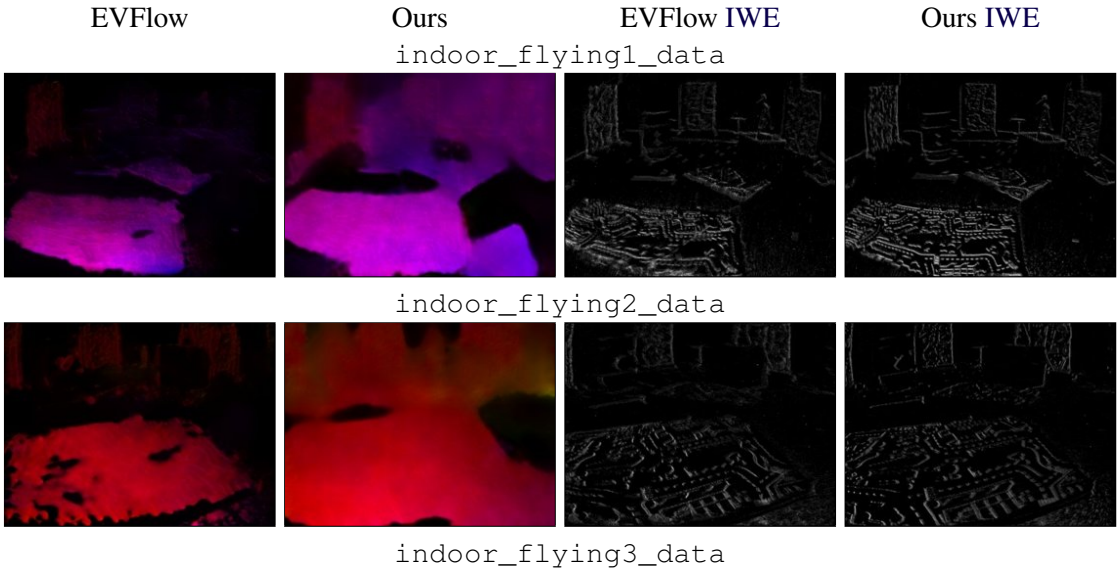


Table 5.11: Qualitative results for IJRR. Left: optic flow vectors represented in HSV color space, right: image of warped events (IWE). Random selection, not cherry picked.



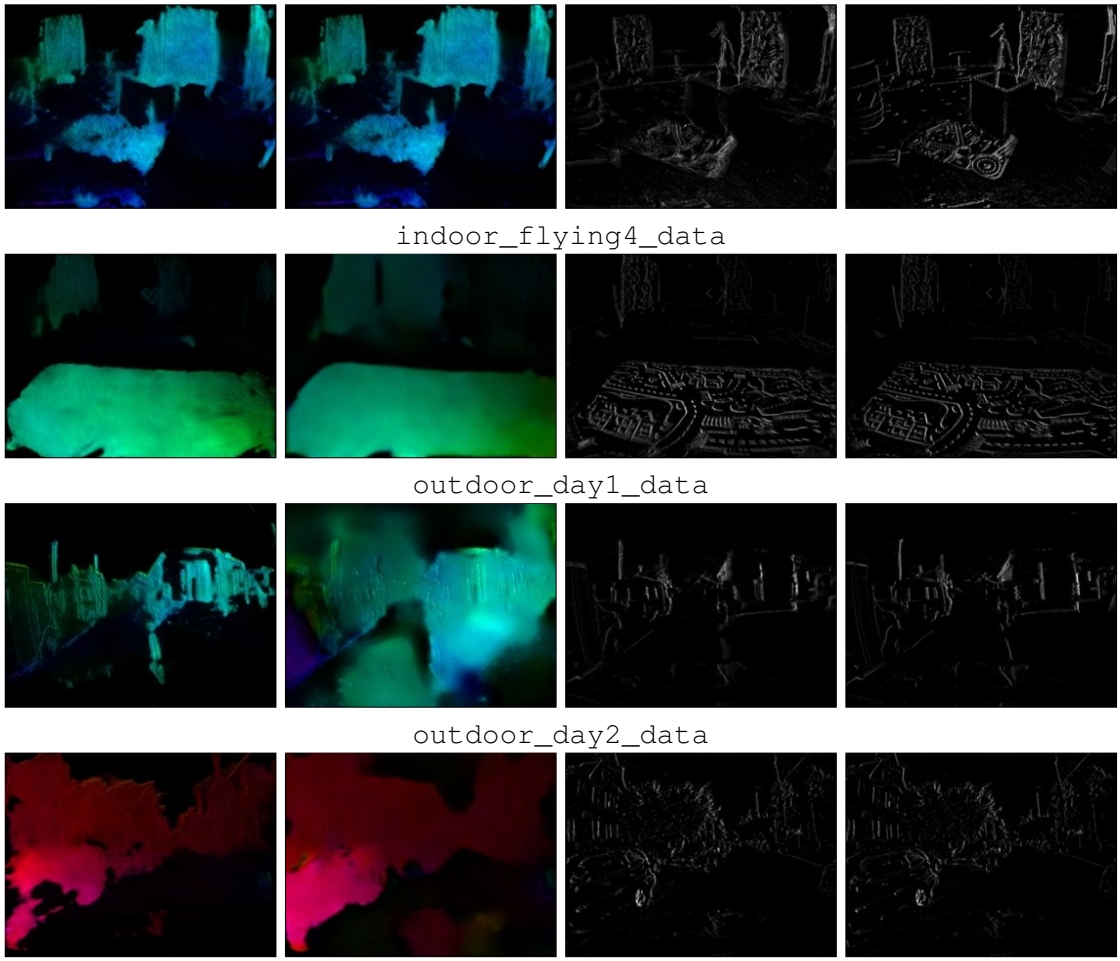


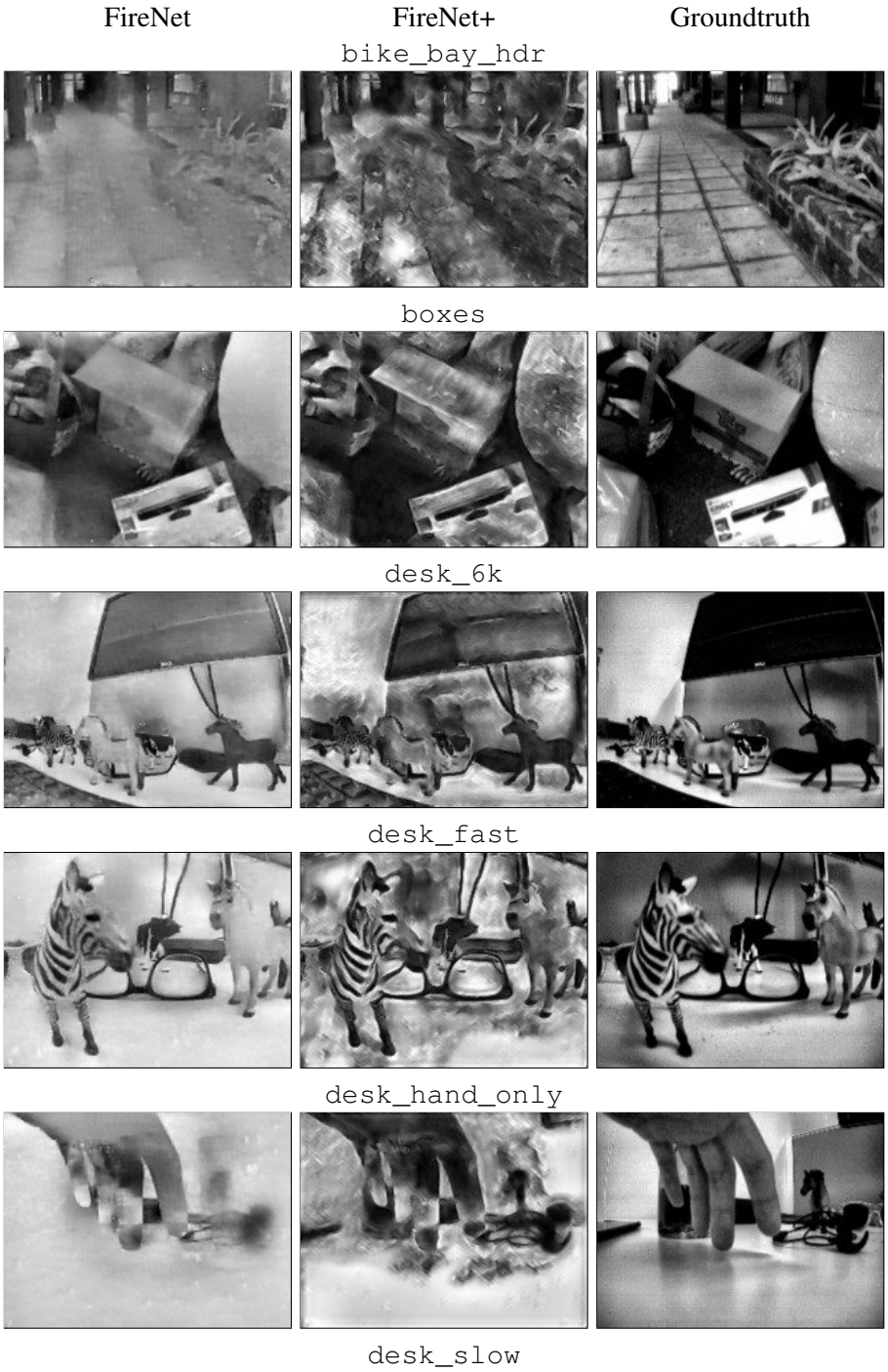
Table 5.12: Qualitative results for optic flow on MVSEC. Left: optic flow vectors represented in HSV color space, right: image of warped events (IWE). Random selection, not cherry picked.

5.4.2 FireNet

[95] propose a lightweight network architecture for fast image reconstruction with an event camera (FireNet) that has 99.6 % fewer parameters than E2VID [88] while achieving similar accuracy on IJRR [64]. We retrain FireNet using our method and evaluate the original (FireNet) vs retrained (FireNet+) on IJRR, MVSEC and HQF (Table 5.13). FireNet+ performs better on HQF and MVSEC though worse on IJRR. One possible explanation is that the limited capacity of a smaller network limits generalisability over a wider distribution of data, and the original FireNet overfits to data similar to IJRR, namely low CTs. If our hypothesis is correct, it presents an additional disadvantage to small networks for event cameras. Comprehensive evaluation (HQF + IJRR + MVSEC) reveals bigger performance gap between FireNet (Table 5.13) and E2VID (Table 5.4) architectures than shown in [95] (IJRR only). Qualitatively (Figure 5.14), FireNet+ looks noisier in textureless regions, while FireNet produces lower contrast images.

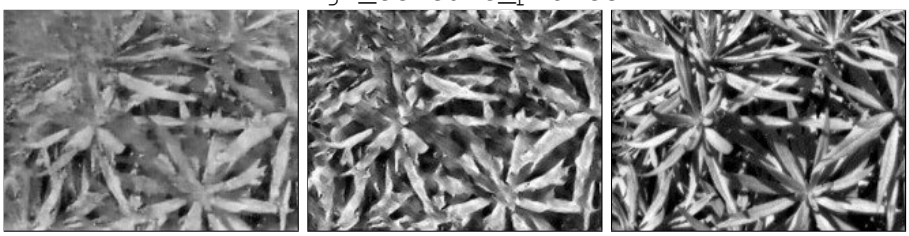
Table 5.13: Mean MSE, SSIM [110] and LPIPS [116] on our HQF dataset, IJRR [64] and MVSEC [118], for original FireNet vs. retrained with our method (FireNet+).

Model	HQF			IJRR			MVSEC		
	MSE	SSIM	LPIPS	MSE	SSIM	LPIPS	MSE	SSIM	LPIPS
FireNet	0.052	**0.5136**	0.387	**0.0554**	**0.6296**	**0.2566**	0.182	**0.3201**	0.594
FireNet+	**0.0485**	0.477	**0.3494**	0.058	0.503	0.327	**0.1568**	0.288	**0.5507**

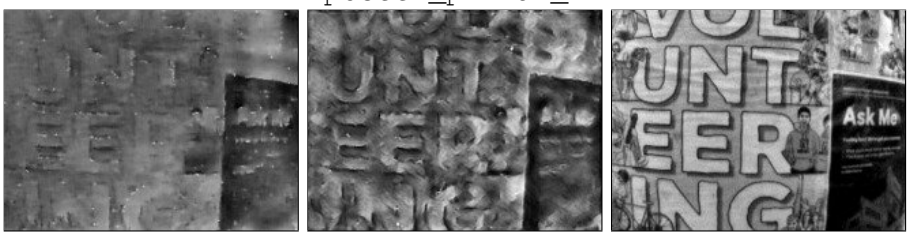


engineering_posters

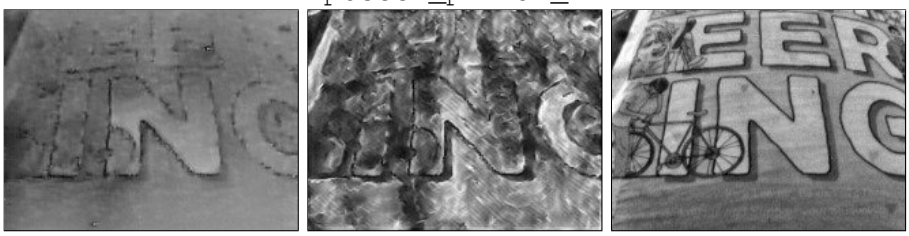
high_texture_plants



poster_pillar_1

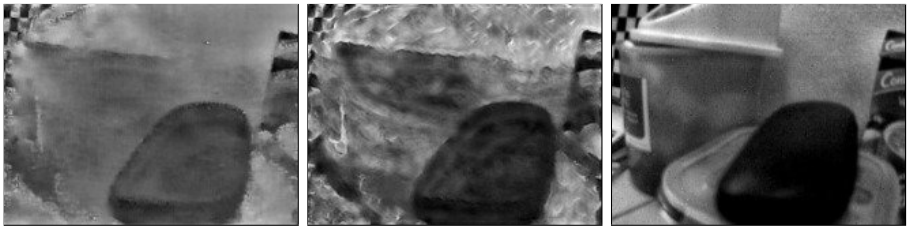


poster_pillar_2



reflective_materials

slow_and_fast_desk



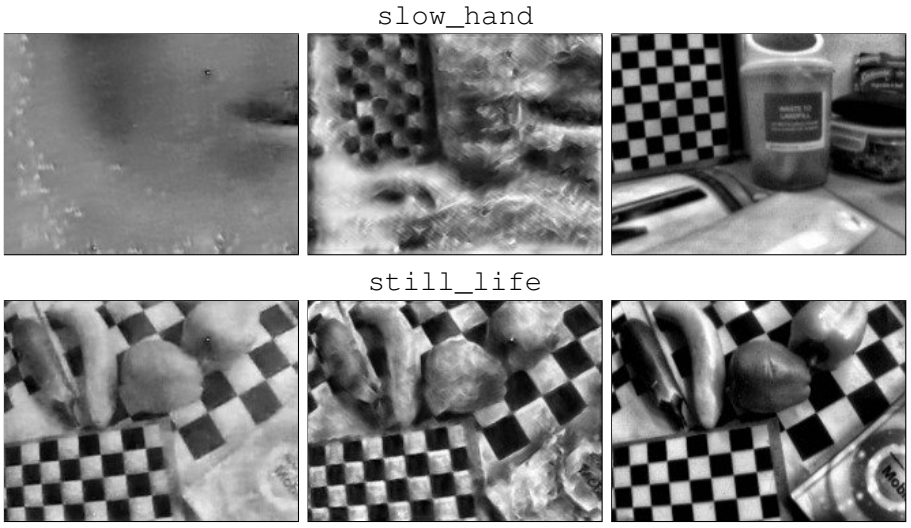
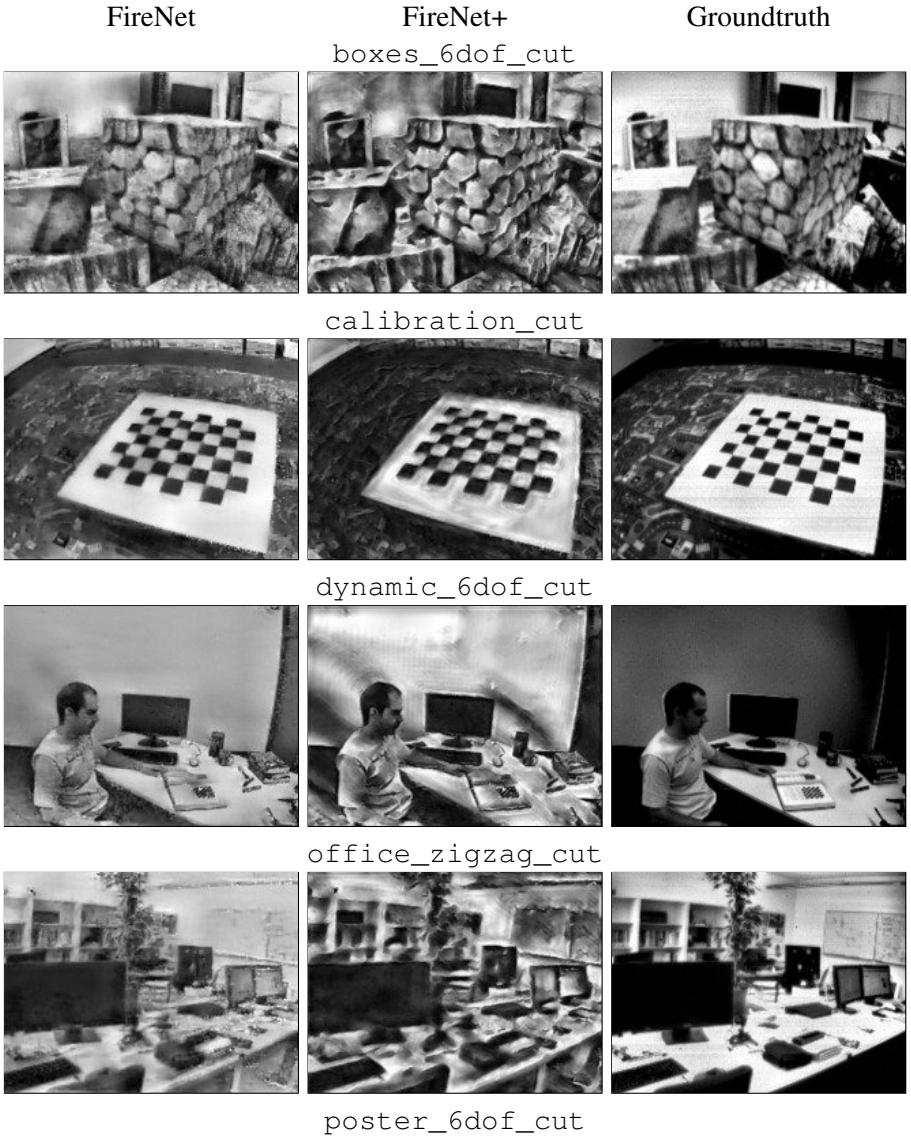


Table 5.14: Qualitative results for HQF. Random selection, not cherry picked.



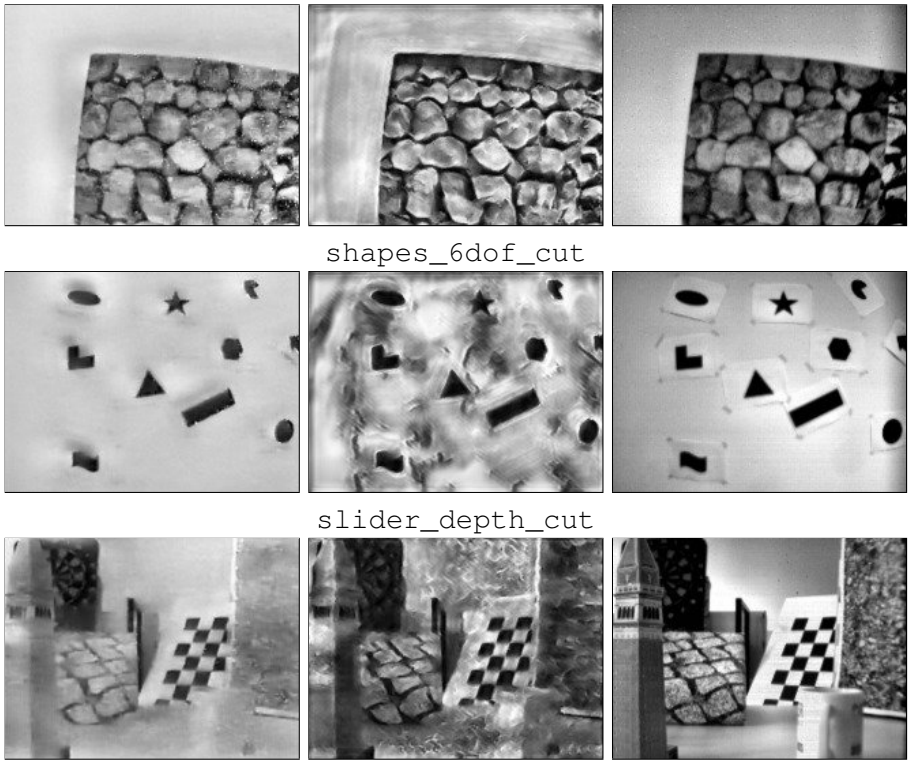


Table 5.15: Qualitative results for IJRR. Random selection, not cherry picked.

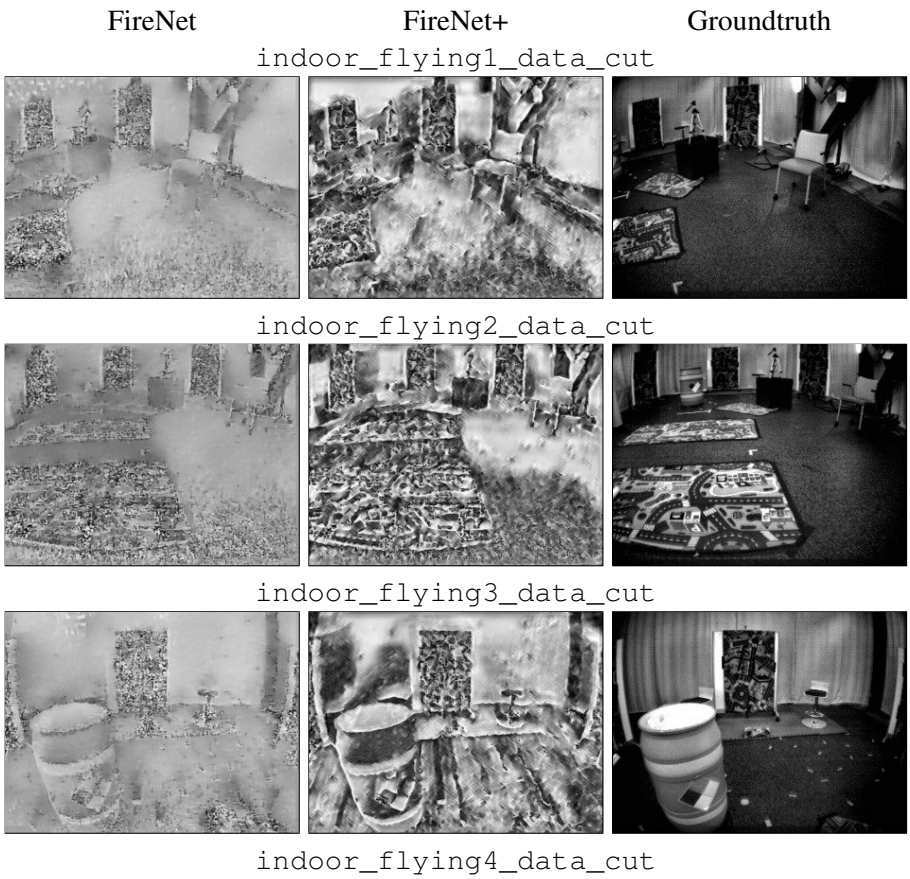


Table 5.16: Qualitative results for MVSEC. Random selection, not cherry picked.

5.4.3 Contrast Thresholds

We investigated the impact of the simulator CTs 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 5.17 shows that for reconstruction (evaluated on LPIPS), IJRR is best on a lower $\lambda_{CT} \approx 0.2$, while MVSEC is best on high $\lambda_{CT} \approx 1.0$. Best or runner up performance was achieved when a wide range of CTs was used, indicating that exposing a network to additional event statistics outside the inference domain is not harmful, and may be beneficial. We believe training with low CTs (thus higher $\frac{\text{events}}{\text{pix}\cdot\text{s}}$) reduces dynamic range in the output images (Table 5.18), perhaps because the network becomes accustomed to a high density of events during training but is presented with lower $\frac{\text{events}}{\text{pix}\cdot\text{s}}$ data at inference. When retraining the original E2VID network, dynamic range increases with CTs (Table 5.18).

5.4.4 Training Noise and Sequence Length

To determine the impact of sequence length and noise augmentation during training, we retrained E2VID architecture using sequence length 40 (L40) and 120 (L120), with and without noise augmentation (N) (see Table 5.19). 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 'reconstruct' noise events, resulting in artifacts (Figure 5.1) observed in E2VID [88] (trained without noise).

5.5 Discussion

The significant improvements gained by training models on our synthetic dataset exemplify the importance of reducing the sim-to-real gap for event cameras in both the event rate induced by varying

Table 5.17: Evaluation of image reconstruction and optic flow networks trained on simulated datasets with a variety of 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 datasets HQF (excluding `desk_hand_only` on FWL), IJRR [64], MVSEC [118]. We report mean squared error (MSE), structural similarity (SSIM) [110] and perceptual loss (LPIPS) [116] for reconstruction and FWL for optic flow. 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	1.93	**0.04**	**0.60**	**0.25**	*1.453*	0.10	**0.35**	0.55	1.15
0.50	**0.04**	*0.51*	0.36	1.90	0.04	0.57	0.27	1.42	*0.10*	0.31	0.52	1.19
0.75	0.05	**0.51**	**0.36**	1.90	0.05	0.56	0.28	1.44	0.11	0.29	0.53	*1.2238*
1.00	0.05	0.48	0.36	1.91	0.05	0.53	0.29	1.42	0.12	0.27	*0.51*	1.18
1.50	0.05	0.47	0.38	*1.9272*	0.06	0.52	0.30	1.44	0.09	0.30	0.52	1.14
All	*0.05*	0.50	**0.36**	**1.9617**	**0.04**	*0.59*	*0.27*	**1.459**	**0.08**	*0.34*	**0.51**	**1.2434**

Table 5.18: Dynamic range (DR) of reconstructed images from IJRR [64]: original E2VID [88] vs 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 [88]	Retrained					
CT	∼0.18	0.2	0.5	0.75	1.0	1.5	All
DR	77.3	89.2	103.7	105.9	104.8	100.0	103.3

the contrast thresholds and the dynamics of the simulation scenes. Our 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 %.

We 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 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, our model may be a better fit, since it targets 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, a challenge for prior self-supervised methods.

Similarly, our results speak for themselves on image reconstruction. While we outperform E2VID [88] on all datasets, the smallest gap is on IJRR, the dataset we found to have lower CTs. E2VID performs worst on MVSEC that contains higher CTs, consistent with our finding that performance is driven by similarity between training and evaluation event data.

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.

Table 5.19: Mean LPIPS [116] on our HQF dataset, IJRR [64] and MVSEC [118], 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	**0.5826**	0.296	0.042	**0.6497**	0.229	0.151	0.330	0.526
L40N	**0.0326**	0.579	**0.2562**	**0.0344**	0.636	**0.2240**	0.105	**0.3461**	**0.4670**
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	**0.0990**	0.344	0.498

5.5.1 Resources

Video, code and datasets: `https://timostoff.github.io/20ecnn`

Chapter 6

Conclusion

Event cameras are a novel, bioinspired mode of visual sensing and a paradigm shift from synchronous, conventional absolute intensity images to asynchronous update, low-latency, events. By reporting only *changes* in intensity, event camera pixels unlock several advantages of biological vision - low latency (on the order of µs), low power usage (on the order of mW), and high dynamic range ($\approx$ 120 dB). Events are reported as a tuple of (x, y) position, polarity of brightness change s and time t, which provides each event with a fine-grained timestamp. Due to their mode of operation, event camera pixels take samples at exactly the rate of change of brightness of the scene, limited only by the Contrast Threshold (CT) (the brightness change threshold required to trigger an event) and refractory period (the period after firing during which a pixel may not generate a new event).

Since event cameras respond only to the dynamics of the scene, they are a natural fit for algorithms which describe motion, such as optic flow estimators, object tracking, or motion segmentation. However, because events do not encode spatial patterns and relationships in the intuitive way that camera frames do, many of the methods that are standard in event-based vision cannot be directly applied to events. This thesis has helped to fill this knowledge gap by demonstrating how optic flow and motion segmentation can be estimated by using Focus Optimisation (FO), proposing new objective functions for the optimisation with benefits for convergence and noise tolerance and providing a means of adapting simulation parameters to real world datasets to provide Convolutional Neural Networks (CNNs) with high quality, well generalised, fully dense optic flow.

The Focus Optimisation (FO) approach to processing events has proved to be a versatile, powerful tool in event-based vision, with the ability to solve problems from camera rotation estimation, deep learning, optic flow, depth estimation, and motion segmentation. It does this by warping events along point trajectories which model the actual trajectories of event-generating point features in the spatiotemporal volume. This warping operation produces an Image of Warped Events (IWE), which becomes more focused as the parameters of the warp model align more closely to the actual trajectories of the events. The focus which can be measured by any number of sharpness measuring functions (such as the variance) can be used as an objective which is optimised w.r.t. the motion parameters of point trajectories.

Chapter 2 presents the first work to apply FO to the problem of optic flow estimation and one of the first works to use FO for event-based vision. Since the optimisation problem has multiple local maxima for more than one moving object, we need to also estimate motion segmentation over the events. We solve this via a greedy segmentation algorithm in which all of the events in the set $\mathcal{E}$ are first motion-compensated, then those events which belong to the discovered motion estimate removed. The process is then repeated for the remaining events until no events remain. The segmentation and motion estimates were then used to initialise asynchronous, event-based trackers to handle incoming events. Key contributions of this work were: (i) the description of a focus optimisation framework for optic flow estimation, (ii) a segmentation scheme for separating events into their respective motions, (iii) an asynchronous event-based motion tracker that makes use of motion-compensated Images of Warped Events (IWEs) and their associated motion parameters in a particle-filter approach, (iv) the fusion of per-event and batching event processing paradigms. We showed that this approach performed better than previous Lucas-Kanade [53] based methods [91]. Uniquely, our approach was

able to consider moving objects globally (insofar that they matched the set of motions modelled by our optic flow motion model), largely eliminating the aperture problem. However, our method was dependent on several scene-dependent hyper-parameters and was designed to work explicitly with optic flow motions models, restricting our method to simpler scenes.

Chapter 3 extended and improved on the ideas in Chapter 2 by applying a probabilistic approach to the motion segmentation. In this method, each event has a likelihood of belonging to a particular motion model, which is not limited to optic flow models. The parameters of the motion models are then optimised together, with respect to a global focus measurement. Subsequently, the likelihoods of the events are updated to account for the new motion estimates. By applying these steps iteratively, a solution for the motion segmentation and parametrisation of the events is converged on, similar to Expectation Maximisation (EM). This method has the advantage that it is not dependent on hyper-parameters and is able to accept any number and mixture of motion models. The key contributions are (i) an iterative method for segmenting multiple objects based on their apparent motion on the image plane, producing a per-event classification into space-time clusters described by parametric motion models, (ii) the detection of independently moving objects without having to compute optical flow explicitly, (iii) the ability to mix motion models at will to suit the expected scene, (iv) a method for evaluating the performance of event-based motion segmentation methods in terms of *relative displacement*, with the recognition that given a larger time interval the segmentation problem becomes easier. We evaluated our method on custom scenes as well as on a public motion segmentation dataset, showing that our method outperformed State of the Art (SotA). We also showed the sensitivity of our method via a novel relative-displacement experiment.

Integral to the work in Chapters 3 and 2 is the concept of FO. Chapter 4 takes a deeper dive into FO by examining the choice of focus measure and proposes a simple classification for various measures. We mathematically and experimentally showed that contrast measures that reward density in the IWE have advantages in noise tolerance, while those that reward sparsity have excelled in overcoming aperture uncertainty, a cousin of the aperture problem in conventional vision. Key contributions of this work are: (i) a classification of reward functions based on whether they operate by rewarding density or sparsity in the IWE, (ii) proof and experimental evidence that rewarding density provides noise tolerance and rewarding sparsity decreases aperture uncertainty, (iii) a method of estimating how much event data needs to be buffered to be able to accurately perform FO, (iv) the definition of aperture uncertainty, an analogue to the aperture problem. Our results show that by using these rewards in combination, we can achieve improvements in terms of noise tolerance and optimisation convergence.

Finally, Chapter 5 shows how performance and generalisability of event-based CNNs can be drastically improved for video reconstruction and optic flow, by reducing the sim-to-real gap between simulated training data and real events. Our key findings and contributions were: (i) that using a wide range of CTs during training data generation improves the performance of SotA models, (ii) that previous works may inadvertently overfit their models to evaluation datasets, (iii) producing a High Quality Frames (HQF) dataset to allow better evaluation of video reconstruction from events, (iv) constructing an optic flow CNN that produces SotA *fully dense* optic flow, (v) defining a novel evaluation metric Flow Warp Loss (FWL) for event-based optic flow which does not require ground truth, in recognition of the limitations of current ground truth optic flow datasets for event cameras. Since all of the improvement gains we report are on existing SotA architectures, I feel confident in attributing the large margins of our improvements (40 % and 15 % for reconstruction and optic flow respectively) to the improvements in our training data. We also discussed options for approximately matching the CTs of simulation to target data and provided experimental evidence that even if the exact target CTs are unknown, simply using a large range of simulation CTs provides desired improvements (i.e. there is no downside to using a wider range of CTs in training data than in target data).

My results have set the benchmark for optic flow estimation and motion segmentation from event

cameras, as well as expanding to the theory of FO. Still I see room for additional research and improved performance. With regards to FO as a framework, I see three major challenges which remain unsolved.

The first has to do with the application of focus optimisation to efficient hardware accelerators, such as an Field Programmable Gate Array (FPGA). Currently the algorithms described in this thesis and in the larger literature which employ FO do not run in real-time and in some cases are very slow. There is a lot of potential for speedups by implementing FO on hardware however, since many operations integral to the method can be performed in parallel. Some obvious opportunities for parallel processing can be found in:

> *Event Warping*: since each event is warped independently, they can all be warped simultaneously with potential speedups on the order of the number of events (N_e) warped.
>
> *IWE Generation*: When events are coalesced into an IWE, most events can be added simultaneously and independently. Events which occupy the same location cannot be added simultaneously, but use of parallel reduction techniques and atomic operations can still provide significant speedup.
>
> *Multiple Focus Evaluation*: Optimisation almost always requires sampling the focus measure (r) for various parametrisations (ω) of the warping function ($\mathcal{W}$). For optimisers such as Branch and Bound (BnB), grid-search or line search in conjugate-gradient methods, many different samples can be taken simultaneously.

While we have tried to take some steps in this direction by implementing parts of our algorithms on Graphics Processing Units (GPUs), this is not suitable for some applications, since the event-based power efficiency is lost as well as the low latency, due to the bottlenecks found in data-transfer rate between the event camera, Central Processing Unit (CPU), and then GPU (usually via Peripheral Component Interconnect express (PCIe)). Since events can be streamed to dedicated hardware such as FPGA or other heterogeneous System on Chips (SoCs) via a low-latency direct interface, they are likely to be a better fit for event-based vision, though substantially more difficult to implement.

A further challenge is finding methods which allow for global convergence of the objective function, potentially without constraints such as linear motion assumptions. Currently, FO methods require either a starting point that is reasonably near the optimum to be within the basin of convergence, or significant smoothing of the objective function. Our work also takes steps in this direction by observing that certain focus measuring functions produce a friendlier objective function and providing some patch-based methods for finding initial optimisation estimates as in Chapter 3. New ground is currently being broken in this field by two recent papers which apply the BnB algorithm to guarantee global convergence of the optimisation problem [50, 80]. The work by [80] tackles the problem of 3-Degree of Freedom (DoF) angular velocity estimation with an event camera, for which the authors achieve SotA results, while [50] apply the method to 2-DoF fronto-parallel motions. This approach is exciting, since it elegantly resolves many of the issues of FO using gradient-based optimisers, especially the issue of linear motion assumptions and the finding of incorrect solutions due to initialisation outside the basin of convergence. Future work will involve calculating the optimisation bounds for a wider variety of motion models and applying the problem to dedicated hardware (such as FPGA), since demonstrations of these methods have been somewhat slow (as with many FO works).

A further problem with FO which has not yet been addressed is the problem of identifying the best time window for optimisation and the effect that this time window might have. To our knowledge we are the only ones who acknowledge this aspect of the problem, by analysing our segmentation performance against relative displacement of moving objects (Figure 3.5). Conversely, there are many examples of works in the literature in which the time window is chosen to be very large, which can enormously simplify the problem being solved. The choice of time window is important, since it has an impact on the size of the basin of convergence, the violation of assumptions of linear motion,

and the sensitivity of segmentation. For example, an approach to training CNNs for the purpose of optic flow is to use a focus measure as a training loss [121]. However, such losses depend on the time slice of events to be the same during training, since larger time slices will naturally give larger losses, which subsequently have a larger impact on the network. This works well on datasets where the dynamics of the scene are homogeneous (such as in Multi Vehicle Stereo Event Camera (MVSEC)), but not for more heterogeneous datasets. I believe that finding losses that are invariant to N_e will be important to increase the robustness of FO.

Powerful results can be achieved by processing events in batches, however this methodology can introduce unwanted additional latency. Though using sliding windows of one event theoretically eliminates this issue, it is hard to picture an implementation where this will be computationally feasible. I believe that an exciting new step in FO will be to find intersections with per-event processing methods, for example as training signals for asynchronous Spiking Neural Networks (SNNs).

My research reveals that events, when considered together, are rich in information, embedded in their high temporal resolution and in the natural associations that exist between events emitted by a common stimulus. This information is sufficient to produce high quality optic flow, motion segmentation and video reconstructions. I believe that this work will help put event cameras at the forefront of mobile robotics, from self-driving cars to mobile phones. The recent explosion in the capabilities of computer vision, deep learning and robotics makes this an exciting time to be working at the forefront of a new frontier in machine vision. This thesis has helped lay the foundations of this burgeoning field, allowing the exploitation of the natural associations that exist between events.

Appendix A

Event Based Vision Concepts

A.1 Event Representations

A.1.1 Discretised Event Volume/Voxel Grid

The Discretised Event Volume (DEV) or voxel grid (Figure 1.5b), is a popular representation in deep learning applications. Many powerful deep learning architectures expect three-dimensional tensors as input. The DEV makes these architectures accessible to event data, although it does lose some of the temporal information in the events. By spreading events to the temporal bins by bilinear interpolation, less temporal information is lost than through simple binning (see Figure A.1). Given a set $\mathcal{E}$ of N_e events $e_i = \{x_i, y_i, s_i, t_i\}_{i=0,\dots,N_e-1}$ spanning $\Delta t = t_N - t_0$ seconds, a DEV V with B bins can be formed via

$$V_{k\in[0,B-1]}(\mathcal{E}) = \sum_{i=1}^{N_e} s_i \max(0, 1 - |t_i^* - k|) \tag{A.1}$$

where t_i^* is the timestamp normalised to the range $[0, B-1]$ via $t_i^* = \frac{t_i - t_0}{\Delta t}(B-1)$ and the bins are evenly spaced over the range $[t_0, t_N]$. Since negative and positive events cancel out in this formulation it is common to generate two separate voxel grids, one for each polarity.

Voxel Formation Methods

Two natural choices for generating Discretised Event Volume (DEV) grids from the event stream are *fixed rate* and *fixed events* (Figure A.2). In fixed rate, DEVs are formed from t second wide slices of the event stream (variable event count), endowing the resulting inference with a fixed frame rate. This has the downside that inference cannot adapt to changing scene dynamics, a disadvantage shared by conventional cameras. A special case of fixed rate is *between frames* where all events between two image frames are used to form a DEV grid.

In fixed events, one waits for N_e events before making a DEV grid no matter how long it takes (variable duration). Fixed events has the downside that if the camera receives few events, either

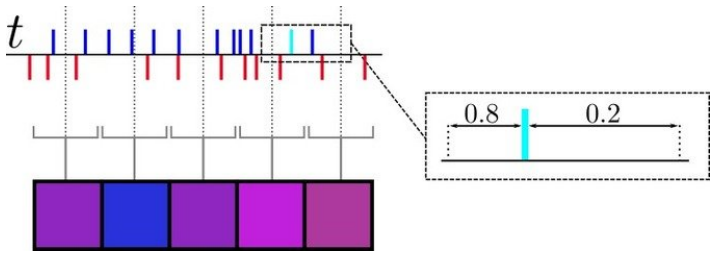


Figure A.1: A Discretised Event Volume (DEV) is formed by assigning the events (red and blue dashes, top) into temporal bins (red/blue boxes, bottom). Each event is assigned to the neighboring bins via bi-linear interpolation - for example, the cyan event is spread in the ratio 0.8:0.2 to the bin it is 20 % and 80 % distant to respectively.

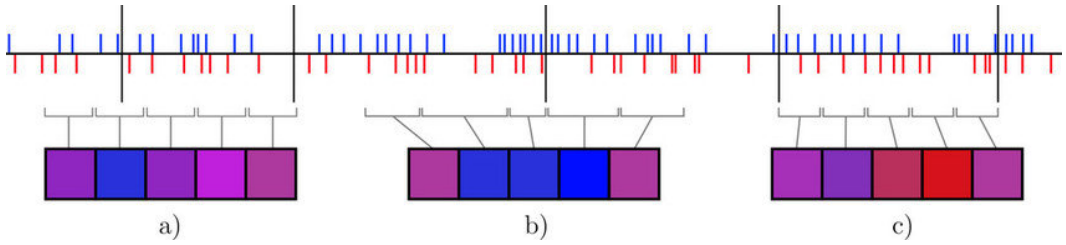


Figure A.2: Events (blue and red lines) on a timeline are discretised into voxels below (squares) according to: a) *fixed rate*, b) *fixed events*, c) *between frames* (frames denoted by black lines).

because the scene has little texture or the motion is slow, the inference rate can slow to a crawl. This method allows matching the value N_e to the average N_e of the training set during inference, potentially benefiting the network. As an example, the average events per DEV in the training set for chapter 5 is 0.0564 events *per voxel*.

A.1.2 Event Image

The event image I_0 is formed by a set of N_e events $e \in \mathcal{E}$ by

$$I_0(\mathcal{E}) = \sum_{i=1}^{N_e} s_i \delta(x - x_i, y - x_i) \tag{A.2}$$

where each event is given by the tuple $e_i = \{x_i, y_i, t_i, s_i\}$ and δ is the Kronecker delta (see Figure 1.5c). Positive and negative events cancel, though in some works separate images are formed for the positive and negative events respectively. The event image can be seen as a special case of the Image of Warped Events (IWE) I_ω with motion parameters $\omega = 0$.

A.1.3 Image of Warped Events

The Image of Warped Events (IWE), denoted I_ω, is the image that is obtained by transporting a set of events $\mathcal{E}$ to a common reference time t_{ref} through some warping operation $\mathcal{W}$. This warping operation is usually parameterised by a set of parameters ω. The warping operation $\mathcal{W}$ thus consists of a map from the events in $\mathbb{R}^3$ to a new location in $\mathbb{R}^3$. The set of warped events is:

$$\mathcal{E}_\mathcal{W} = \mathcal{W}(e, t_{\text{ref}}; \omega) \; \forall e \in \mathcal{E} \tag{A.3}$$

These warped events now lie on a common plane at t_{ref} and can be easily formed into an event image by summing the values of the events at each location. Since the warped pixel locations are not integer values, this summation is best performed using bilinear voting, to avoid aliasing effects. That is, given the set of N_e warped events $e' = \{x', y', t', s'\} \in \mathcal{E}_\mathcal{W}$, the value of each pixel $(u, v) \in I_\omega$ is given as

$$I_\omega(u, v) = \sum_{i=1}^{N_e} s_i' \max(0, 1 - |x_i' - u|) \max(0, 1 - |y_i' - v|) \tag{A.4}$$

In the case that $\mathcal{W}$ transports the events along the point trajectories of the common visual features, the generation of the IWE can be interpreted as a motion-compensation of the events (see Focus Optimisation (FO)).

A.1.4 Surface of Active Events

The Surface of Active Events (SAE) (also commonly known as time image) is a common event representation, in which the latest timestamp is recorded at each pixel location (see Figure 1.5d).

Formally, given a set of N_e events $e = \{x, y, s, t\} \in \mathcal{E}$, each pixel u, v of the SAE is given by:

$$I_t(u,v) = \max(t_0 f(e_0, u, v), t_1 f(e_1, u, v), \dots t_{N_e-1} f(e_{N_e-1}, u, v)) \tag{A.5}$$

where

$$f(e,u,v) = \min(0, |x-u|) \min(0, |y-v|) \tag{A.6}$$

Since the absolute timestamp value is somewhat arbitrary (the relative timing of the events being the important part), the SAE is often normalised. A popular normalisation scheme is sort normalisation [2], as this is independent of actual timestamp values. This is important, since if even a single pixel does not receive any new events over a long period, the normalisation range can become very large. Sort normalisation has been shown to be robust in several applications and can be computed and updated in an efficient and asynchronous manner [46].

A.2 Focus Optimisation

Focus Optimisation (FO) is an event processing framework which has been fruitful across a range of tasks [26]. FO makes direct use of the data association between events generated by a common visual feature, by warping these common events to one point on an Image of Warped Events (IWE) (see Figure 1.7). The warp function $\mathcal{W}$ should be parameterised by parameters ω and chosen to correlate with the desired property of the event stream to be ascertained. For example, if optic flow is the desired output, $\mathcal{W}$ should represent on optic flow motion model (v_x, v_y). A good estimate of ω should correspond to a focused I_ω, essentially motion-compensating the events. The focus can be measured and when optimised should reveal the correct (v_x, v_y). The problem is thus formulated:

$$\arg\max_{\omega} r(I_\omega) \tag{A.7}$$

where r is the function used to measure the focus of the IWE. The variance of the sample image is often used for r, which effectively measures the contrast of the image, which correlates with focus [25].

Appendix B

Event Utility Library

The event utility library (`https://github.com/TimoStoff/event_utils`) contains several libraries for event-based vision implemented in the Python programming language. These libraries contain utilities for Focus Optimisation (FO) (Section B.1), deep learning (Section B.2), augmentation (Section B.3), data format conversion (Section B.4), event representation generation (Section B.5) and visualisation (Section B.6) of event-based data. Most of the functionality uses `pytorch`, an open source library for Graphics Processing Unit (GPU) operations, making most of the library functionality easily accessible on GPU. Together with array broadcasting, which is used wherever possible, the library allows for fast processing of events, despite Python's interpreter, which is traditionally thought of as quite slow compared to compiled languages such as C++.

The following is a brief description of the functionality of the library. The purpose is to give a rough overview of the capabilities, without gong deep into the API and implementation details. As a result, not every little helper and private function is listed, only those that might be of immediate interest to event camera researchers. The library is documented using Doxygen conventions and auto-generated documentation can be found at the project website for those interested in a full documentation of the project.

B.1 Focus Optimisation (FO)

The FO library contains code that allows the user to perform FO on events. The important files of this library are:

B.1.1 events_cmax.py

This file contains code to perform FO. The most important functionality is provided by:

- `grid_search_optimisation`: Performs the grid search optimisation described in Chapter 2.
- `optimize`: Performs gradient based FO on the input events, given an objective function and motion model (Chapter 3).
- `grid_cmax`: Given a set of events, splits the image plane into Region of Interests (RoIs) of size `roi_size`. Performs FO on each RoI separately.
- `segmentation_mask_from_d_iwe`: Retrieve a segmentation mask for the events as described in Chapter 2 based on $\frac{dI_\omega}{d\omega}$.
- `draw_objective_function`: Draw the objective function for a given set of events, motion model and objective function. Produces plots as in Figure 2.4d.
- `main`: Demo showing various capabilities and code examples.

B.1.2 objectives.py

This file implements various objective functions described in this thesis as well as some other commonly cited works. Objective functions inherit from the parent class `objective_function`. The idea is to make it as easy as possible to add new, custom objective functions by providing a common API for the optimisation code. This class has several members that require initialisation:

- `name`: The name of the objective function (e.g. 'variance').
- `use_polarity`: Whether to use the polarity of the events in generating Images of Warped Events (IWEs).
- `has_derivative`: Whether this objective has an analytical derivative w.r.t. ω.
- `default_blur`: What σ should be default for blurring.
- `adaptive_lifespan`: An innovative feature to deal with linearisation errors. Many implementations of contrast maximisation use assumptions of linear motion wrt the chosen motion model. A given estimate of the motion parameters implies a lifespan of the events. If `adaptive_lifespan`: is True, the number of events used during warping is cut to that lifespan for each optimisation step, computed using `pixel_crossings`. e.g. If motion model is optic flow velocity and the estimate = 12 pixels/second and `pixel_crossings`= 3, then the lifespan will be $\frac{3}{12} = 0.25$s.
- `pixel_crossings`: Number of pixel crossings used to calculate lifespan.
- `minimum_events`: The minimal number of events that the lifespan can cut to.

The required function that inheriting classes need to implement are:

- `evaluate_function`: Evaluate the objective function for given parameters, events etc.
- `evaluate_gradient`: Evaluate the objective function and the gradient of the objective function w.r.t. motion parameters for given parameters, events etc.

The objective functions implemented in this file are:

- `variance_objective`: Variance objective (see [27]).
- `rms_objective`: Root Mean Squared objective.
- `sos_objective`: See Chapter 4, Equation 4.5.
- `soe_objective`: See Chapter 4, Equation 4.9.
- `moa_objective`: See Chapter 4, Equation 4.11.
- `soa_objective`: See Chapter 4, Equation 4.12.
- `sosa_objective`: See Chapter 4, Equation 4.14.
- `zhu_timestamp_objective`: Objective function defined in [121].
- `r1_objective`: Combined objective function r_{R1} in Chapter 4.
- `r2_objective`: Combined objective function r_{R2} in Chapter 4.

B.1.3 warps.py

This file implements warping functions described in this thesis as well as some other commonly cited works. Objective functions inherit from the parent class `warp_function`. The idea is to make it as easy as possible to add new, custom warping functions by providing a common API for the optimisation code. Initialisation requires setting member variables:

- `name`: Name of the warping function, e.g. `optic_flow`.
- `dims`: Degree of Freedom (DoF) of the warping function.

The only function that needs to be implemented by inheriting classes is `warp`, which takes events, a reference time and motion parameters as input. The function then returns a list of the warped event coordinates as well as the Jacobian of each event w.r.t. the motion parameters. Warp functions currently implemented are:

- `linvel_warp`: 2-DoF optic flow warp.
- `xyztheta_warp`: 4-DoF warping function from [58] ((x, y, z) velocity and angular velocity θ around the origin).
- `pure_rotation_warp`: 3-DoF pure rotation warp (x, y, θ where x, y are the center of rotation and θ is the angular velocity).

B.2 Deep Learning

The deep learning code can be found in the `data_loaders` library. It contains code for loading events and transforming them into voxel grids in an efficient manner as well as code for data augmentation. Actual networks and objective functions described in this thesis are not implemented in the library but at the project page for that paper.

`data_loaders` provides a highly versatile `pytorch` dataloader, which can be used across various storage formats for events (.txt, Hierarchical Data Format v.5 (HDF5), memmap etc.). As a result, it is very easy to implement new dataloader for a different storage format. The output of the dataloader was originally to provide voxel grids of the events, but can be used just as well to output batched events, due to a custom `pytorch` collation function. As a result, the dataloader is useful for any situation in which it is desirable to iterate over the events in a storage medium and is not only useful for deep learning. For instance, if one wants to iterate over the events that lie between all the frames of a Dynamic and Active-pixel VIsion Sensor (DAVIS) sequence, the following code is sufficient:

```
dloader = DynamicH5Dataset(path_to_events_file)
for item in dloader:
        print(item['events'].shape)
```

B.2.1 base_dataset.py

This file defines the base dataset class (`BaseVoxelDataset`), which defines all batching, augmentation, collation and housekeeping code. Inheriting classes (one per data format) need only to implement the abstract functions for providing events, frames and other data from storage. These abstract functions are:

- `get_frame(self, index)`: Given an index n, return the n^{th} frame.
- `get_flow(self, index)`: Given an index n, return the n^{th} optic flow frame.

- `get_events(self, idx0, idx1)`: Given a start and end index `idx0` and `idx1`, return all events between those indices.
- `load_data(self, data_path)`: Function which is called once during initialisation, which creates handles to files and sets several class attributes (number of frames, events etc.).
- `find_ts_index(self, timestamp)`: Given a timestamp, get the index of the nearest event.
- `ts(self, index)`: Given an event index, return the timestamp of that event.

The function `load_data` must set the following member variables:

- `self.sensor_resolution`: Event sensor resolution.
- `self.has_flow`: Whether or not the data has optic flow frames.
- `self.t0`: The start timestamp of the events.
- `self.tk`: The end timestamp of the events.
- `self.num_events`: The number of events in the dataset.
- `self.frame_ts`: The timestamps of the time-synchronised frames.
- `self.num_frames`: The number of frames in the dataset.

The constructor of the class takes following arguments:

- `data_path`: Path to the file containing the event/image data.
- `transforms`: Python dict containing the desired augmentations.
- `sensor_resolution`: The size of the image sensor.
- `num_bins`: The number of bins desired in the voxel grid.
- `voxel_method`: Which method should be used to form the voxels.
- `max_length`: If desired, the length of the dataset can be capped to `max_length` batches.
- `combined_voxel_channels`: If True, produces one voxel grid for all events, if False, produces separate voxel grids for positive and negative channels.
- `return_events`: If true, returns events in output dict.
- `return_voxelgrid`: If true, returns voxel grid in output dict.
- `return_frame`: If true, returns frames in output dict.
- `return_prev_frame`: If true, returns previous batch's frame to current frame in output dict.
- `return_flow`: If true, returns optic flow in output dict.
- `return_prev_flow`: If true, returns previous batch's optic flow to current optic flow in output dict.
- `return_format`: Which output format to use (options=`'numpy'` and `'torch'`).

The parameter `voxel_method` defines how the data is to be batched. For instance, one might wish to have data returned in windows `t` seconds wide, or to always get all data between successive Active Pixel Sensor (APS) frames. The method is given as a dict, as some methods have additional parametrisations. The current options are:
i) `k_events`: Data is returned every `k` events. The dict is given in the format:

```
method = {
    'method':'k_events',
    'k':  value_for_k,
    'sliding_window_w':value_for_sliding_window
}.
```

The parameter `sliding_window_w` defines by how many events each batch overlaps.
ii) `t_seconds`: Data is returned every `t` seconds. The dict is given in the format

```
method = {
    'method':'t_seconds',
    't':  value_for_t,
    'sliding_window_t':  value_for_sliding_window }.
```

The parameter `sliding_window_t` defines by how many seconds each batch overlaps.
iii) `between_frames`: All data between successive frames is returned. Requires time-synchronised frames to exist. The dict is given in the format
`method={'method':'between_frames'}.`
Generating the voxel grids can be done very efficiently and on the GPU (if the events have been loaded there) using the `pytorch` function `target.index_put_(index, value, accumulate = True)`. This function puts values from `value` into `target` using the indices specified in `indices` using highly optimised C++ code in the background. `accumulate` specifies if values in `value` which get put in the same location on `target` should sum (accumulate) or overwrite one another. In summary, `BaseVoxelDataset` allows for very fast, on-device data-loading and on-the-fly voxel grid generation.

B.2.2 hdf5_dataset.py and memmap_dataset.py

Currently the library contains implementations for loading events saved in the HDF5 format used at Monash University and Australia National University and events saved in the memmap format commonly used at Robotics and Perception Group, Zurich (RPG).

B.3 Augmentation

While the deep learning library (Appendix B.2) contains some code for tensor augmentation (such as adding Gaussian noise, rotations, flips, random crops etc.), the augmentation library allows for these operations to occur on the raw events. This functionality is contained within `event_augmentation.py`.

B.3.1 event_augmentation.py

The following augmentations are available:

- `add_random_events`: Generates N new random events, drawn from a uniform distribution over the size of the spatiotemporal volume.
- `remove_events`: Makes the event stream more sparse, by removing a random selection of N events from the original event stream.

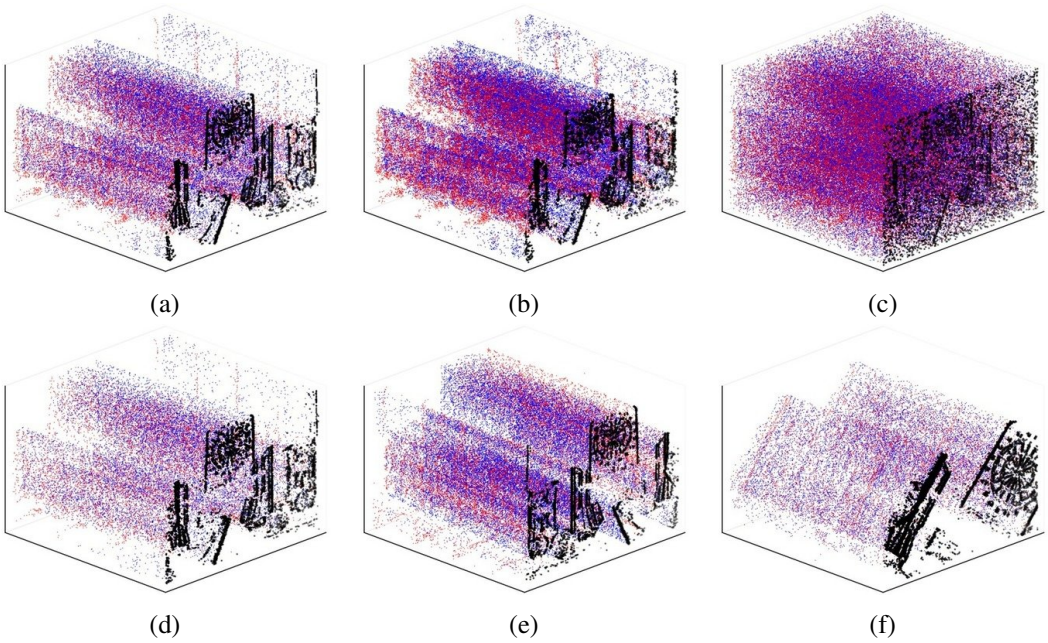


Figure B.1: Various augmentations of the events (events in red/blue with initial events in black for easier viewing). B.1a The original events from `slider_depth` [64]. B.1b The events are doubled using random correlated events. B.1c The events are doubled using random events. B.1d The events are halved using random selection. B.1e The events are flipped around the x axis. B.1f The events are rotated 45° around the center of the image (and cropped to fit sensor size).

- `add_correlated_events`: Makes the event stream more dense by adding N new events around the existing events. Each original event is fitted with a Gaussian bubble with standard deviation σ_{xy} in the x, y dimension and σ_t in the t dimension. New events are drawn from these distributions. Note that this also 'blurs' the event stream.
- `flip_events_x`: Flip events over x axis.
- `flip_events_y`: Flip events over y axis.
- `crop_events`: Spatially crop events either randomly, to a desired amount and either from the origin or as a center crop.
- `rotate_events`: Rotate events by angle θ around a center of rotation a, b. Events can then optionally be cropped in the case that they overflow the sensor resolution.

Figure B.1 shows some examples of possible augmentations. Since the augmentations are implemented using vectorisation, the heavy lifting is done in optimised C/C++ backends and is thus very fast.

B.4 Data Formats

The `data_formats` provides code for converting events in one file format to another. Even though many candidates have appeared over the years (rosbag, AEDAT, .txt, HDF5, pickle, cuneiform clay tablets, just to name a few), a universal storage option for event-based data has not yet crystallised. Some of these data formats are particularly useful within particular operating systems or programming languages. For example, rosbags are the natural choice for C++ programming with the Robot Operating System (ROS) environment. Since they also store data in an efficient binary format, they

have become a very common storage option. However, they are notoriously slow and impractical to process in Python, which has become the de-facto deep-learning language and is commonly used in research due to the rapid development cycle. More practical (and importantly, fast) options are the HDF5 and numpy memmap formats. HDF5 is a more compact and easily accessible format, since it allows for easy grouping and metadata allocation, however it's difficulty in setting up multi-threading access and subsequent buggy behaviour (even in read-only applications) means that memmap is more common for deep learning, where multi-threaded data-loaders can significantly speed up training.

B.4.1 event_packagers.py

The `data_formats` library provides a `packager` abstract base class, which defines what a `packager` needs to do. `packager` objects receive data (events, frames etc.) and write them to the desired file format (e.g. HDF5). Converting file formats is now much easier, since input files now need only to be parsed and the data sent to the `packager` with the appropriate function calls. The functions that need to implemented are:

- `package_events`: A function which given events, writes them to the file/buffer.
- `package_image`: A function which given images, writes them to the file/buffer.
- `package_flow`: A function which given optic flow frames, writes them to the file/buffer.
- `add_metadata`: Writes metadata to the file (number of events, number of negative/positive events, duration of sequence, start time, end time, number of images, number of optic flow frames).
- `set_data_available`: What data is available and needs to be written (i.e. events, frames, optic flow).

A `packager` for HDF5 and memmap is implemented.

B.4.2 h5_to_memmap.py and rosbag_to_h5.py

The library implements two converters, one for HDF5 to memmap and one for rosbag to HDF5. These can be easily called from the command line with various options that can be found in the documentation.

B.4.3 add_hdf5_attribute.py

`add_hdf5_attribute.py` allows the user to add or modify attributes to existing HDF5 files. Attributes are the manner in which metadata is saved in HDF5 files.

B.4.4 read_events.py

`read_events.py` contains functions for reading events from HDF5 and memmap. The functions are:

- `read_memmap_events`.
- `read_h5_events`.

B.5 Representations

This library contains code for generating representations from the events in a highly efficient, GPU ready manner.

B.5.1 voxel_grid.py

This file contains several means for forming and viewing voxel grids (Appendix A.1.1) from events. There are two versions of each function, representing a pure `numpy` and a `pytorch` implementation. The `pytorch` implementation is necessary for GPU processing, however it is not as commonly used as `numpy`, which is so frequently used as to barely be a dependency any more. Functions for `pytorch` are:

- `voxel_grids_fixed_n_torch`: Given a set of n events, return a voxel grid with B bins and with a fixed number of events.
- `voxel_grids_fixed_t_torch`: Given a set of events and a duration t, return a voxel grid with B bins and with a fixed temporal width t.
- `events_to_voxel_timesync_torch`: Given a set of events and two times t_0 and t_1, return a voxel grid with B bins from the events between t_0 and t_1.
- `events_to_voxel_torch`: Given a set of events, return a voxel grid with B bins from those events.
- `events_to_neg_pos_voxel_torch`: Given a set of events, return a voxel grid with B bins from those events. Positive and negative events are formed into two separate voxel grids.

Functions for `numpy` are:

- `events_to_voxel`: Given a set of events, return a voxel grid with B bins from those events.
- `events_to_neg_pos_voxel`: Given a set of events, return a voxel grid with B bins from those events. Positive and negative events are formed into two separate voxel grids.

Additionally:

- `get_voxel_grid_as_image`:Returns a voxel grid as a series of images, one for each bin for display.
- `plot_voxel_grid`: Given a voxel grid, display it as an image.

Voxel grids can be formed both using spatial and temporal interpolation between the bins.

B.5.2 image.py

`image.py` contains code for forming images from events in an efficient manner. The functions allow for forming images with both discrete and floating point events using bilinear interpolation. Images currently supported are event images (Appendix A.1.2) and timestamp images using either `numpy` or `pytorch`. Functions are:

- `events_to_image`: Form an image from events using `numpy`. Allows for bilinear interpolation while assigning events to pixels and padding of the image or clipping of events for events which fall outside of the range.
- `events_to_image_torch`: Form an image from events using `pytorch`. Allows for bilinear interpolation while assigning events to pixels and padding of the image or clipping of events for events which fall outside of the range.
- `image_to_event_weights`: Given an image and a set of event coordinates, get the pixel value of the image for each event using reverse bilinear interpolation.

- `events_to_image_drv`: Form an image from events and the derivative images from the event Jacobians (with options for padding the image or clipping out-of-range events). Of particular use for FO where analytic gradients motion models are known.
- `events_to_timestamp_image`: Method to generate the average timestamp images from Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. "Unsupervised Event-based Learning of Optical Flow, Depth, and Egomotion". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019, pp. 989–997 using `numpy`. Returns two images, one for negative and one for positive events.
- `events_to_timestamp_image_torch`: Method to generate the average timestamp images from Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. "Unsupervised Event-based Learning of Optical Flow, Depth, and Egomotion". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019, pp. 989–997 using `pytorch`. Returns two images, one for negative and one for positive events.

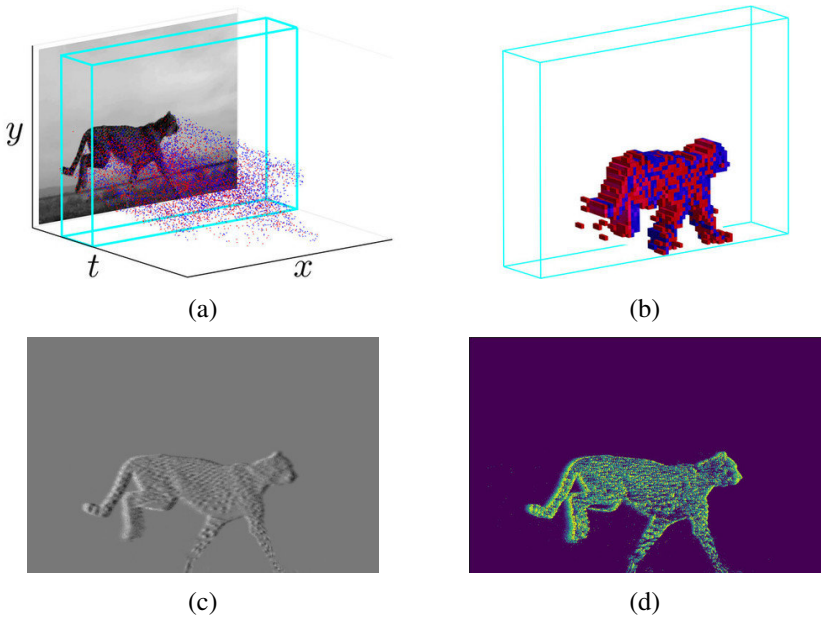
B.6 Visualisation

The `visualization` library contains methods for generating figures and movies from events. The majority of figures shown in the thesis were generated using this library. Two rendering backends are available, the commonly used `matplotlib` plotting library and `mayavi`, which is a VTK based graphics library. The API for both of these is essentially the same, the main difference being the dependency on `matplotlib` or `mayavi`. `matplotlib` is very easy to set up, but quite slow, `mayavi` is very fast but more difficult to set up and debug. I will describe the `matplotlib` version here, although all functionality exists in the `mayavi` version too (see the code documentation for details).

B.6.1 draw_event_stream.py

The core work is done in this file, which contains code for visualising events and voxel grids (see Figure B.2 for examples). The function for plotting events is `plot_events`. Input parameters for this function are:

- `xs`: x coords of events.
- `ys`: y coords of events.
- `ts`: t coords of events.
- `ps`: p coords of events.
- `save_path`: If set, will save the plot to here
- `num_compress`: Takes `num_compress` events from the beginning of the sequence and draws them in the plot at time $t = 0$ in black. This aids visibility (see Figure B.1).
- `compress_front`: If True, display the compressed events in black at the front of the spatiotemporal volume rather than the back
- `num_show`: Sets the number of events to plot. If set to -1 will plot all of the events (can be potentially expensive). Otherwise, skips events in order to achieve the desired number of events
- `event_size`: Sets the size of the plotted events.
- `elev`: Sets the elevation of the plot.



(a) (b)

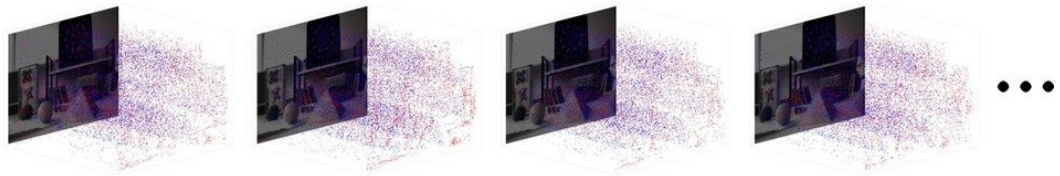
(c) (d)

Figure B.2: Several visualisations made using the *visualization* library. B.2a shows event visualisation, B.2b shows a voxel grid, B.2c shows the event image and B.2d the Surface of Active Events (SAE).

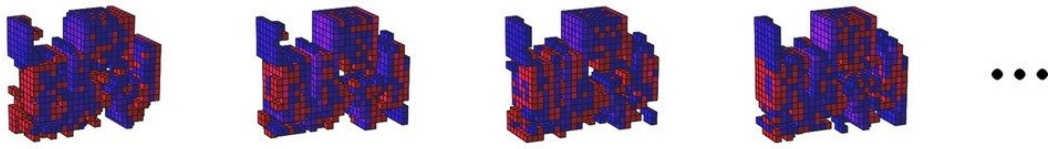
- `azim`: Sets the azimuth of the plot.
- `imgs`: A list of images to draw into the spatiotemporal volume.
- `img_ts`: A list of the position on the temporal axis where each image from `imgs` is to be placed.
- `show_events`: If False, will not plot the events (only images).
- `show_plot`: If True, display the plot in a `matplotlib` window as well as saving to disk.
- `crop`: A crop, if desired, of the events and images to be plotted.
- `marker`: Which marker should be used to display the events (default is '.', which results in points, but circles 'o' or crosses 'x' are among many other possible options).
- `stride`: Determines the pixel stride of the image rendering (1=full resolution, but can be quite resource intensive).
- `invert`: Inverts the colour scheme for black backgrounds.
- `img_size`: The size of the sensor resolution. Inferred if empty.
- `show_axes`: If True, draw axes onto the plot.

The analogous function for plotting voxel grids is:

- `xs`: x coords of events.
- `ys`:y coords of events.
- `ts`: t coords of events.
- `ps`: p coords of events.



(a)



(b)

Figure B.3: Slider sequence [64] visualised as video sequence. B.3a as event cloud with frames, B.3b as voxels.

- `bins`: The number of bins to have in the voxel grid.
- `frames`: A list of images to draw into the plot with the voxel grid.
- `frame_ts`: A list of the position on the temporal axis where each image from `frames` is to be placed.
- `sensor_size`: Event sensor resolution.
- `crop`: A crop, if desired, of the events and images to be plotted.
- `elev`: Sets the elevation of the plot.
- `azim`: Sets the azimuth of the plot.

To plot successive frames in order to generate video, the function `plot_events_sliding` can be used. Essentially, this function renders a sliding window of the events, for either the event or voxel visualisation modes. Similarly, `plot_between_frames` can be used to render all events between frames, with the option to skip every n^{th} event. To generate such plots from the command line, the library provides the scripts:

- `visualize_events.py`
- `visualize_voxel.py`
- `visualize_flow.py`

These provide a range of documented commandline arguments with sensble defaults from which plots of the events, voxel grids and events with optic flow overlaid can be generated. For example,
`python visualize_events.py /path/to/slider_depth.h5`
produces plots of the `slider_depth` [64] sequence (Figure B.3a). Invoking:
`python visualize_voxel.py /path/to/slider_depth.h5`
produces voxels of the `slider_depth` [64] sequence (Figure B.3b).

B.7 Util

This library contains some utility functions used in the rest of the library. Functions include:

- `infer_resolution`: Given events, guess the resolution by looking at the max and min values.
- `events_bounds_mask`: Get a mask of the events that are within given bounds.
- `clip_events_to_bounds`: Clip events to the given bounds.
- `cut_events_to_lifespan`: Given motion model parameters, compute the speed and thus the lifespan, given a desired number of pixel crossings.
- `get_events_from_mask`: Given an image mask, return the indices of all events at each location in the mask.
- `binary_search_h5_dset`: Binary search for a timestamp in an HDF5 event file, without loading the entire file into RAM.
- `binary_search_torch_tensor`: Binary search implemented for `pytorch` tensors (no native implementation exists).
- `remove_hot_pixels`: Given a set of events, removes the 'hot' pixel events. Accumulates all of the events into an event image and removes the `num_hot` highest value pixels.
- `optimal_crop_size`: Find the optimal crop size for a given `max_size` and `subsample_factor`. The optimal crop size is the smallest integer which is greater or equal than `max_size`, while being divisible by $2^{\texttt{max\_subsample\_factor}}$.
- `plot_image_grid`: Given a list of images, stitch them into a grid and display/save the grid.
- `flow2bgr_np`: Turn optic flow into an RGB image.

Bibliography

[1] Mohammed Mutlaq Almatrafi and Keigo Hirakawa. "DAViS Camera Optical Flow". In: *IEEE Trans. Comput. Imaging* (2019), pp. 1–11. DOI: 10.1109/tci.2019.2948787.

[2] Ignacio Alzugaray and Margarita Chli. "ACE: An efficient asynchronous corner tracker for event cameras". In: *3D Vision (3DV)*. 2018, pp. 653–661. DOI: 10.1109/3DV.2018.00080.

[3] Patrick Bardow, Andrew J. Davison, and Stefan Leutenegger. "Simultaneous Optical Flow and Intensity Estimation From an Event Camera". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2016, pp. 884–892. DOI: 10.1109/CVPR.2016.102.

[4] Francisco Barranco, Cornelia Fermuller, and Yiannis Aloimonos. "Contour Motion Estimation for Asynchronous Event-Driven Cameras". In: *Proc. IEEE* 102.10 (Oct. 2014), pp. 1537–1556. DOI: 10.1109/jproc.2014.2347207.

[5] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. "Event-Based Visual Flow". In: *IEEE Trans. Neural Netw. Learn. Syst.* 25.2 (2014), pp. 407–417. DOI: 10.1109/TNNLS.2013.2273537.

[6] Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. "Asynchronous frameless event-based optical flow". In: *Neural Netw.* 27 (2012), pp. 32–37. DOI: 10.1016/j.neunet.2011.11.001.

[7] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, Inc., 2006.

[8] Kwabena A. Boahen. "A Burst-Mode Word-Serial Address-Event Link-I: Transmitter Design". In: *IEEE Trans. Circuits Syst. I* 51.7 (July 2004), pp. 1269–1280. DOI: 10.1109/TCSI.2004.830703.

[9] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. "A 240x180 130dB 3us Latency Global Shutter Spatiotemporal Vision Sensor". In: *IEEE J. Solid-State Circuits* 49.10 (2014), pp. 2333–2341. DOI: 10.1109/JSSC.2014.2342715.

[10] Tobias Brosch, Stephan Tschechne, and Heiko Neumann. "On event-based optical flow detection". In: *Front. Neurosci.* 9 (Apr. 2015), p. 137. DOI: 10.3389/fnins.2015.00137.

[11] John Canny. "A Computational Approach to Edge Detection". In: *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-8.6 (Nov. 1986), pp. 679–698. DOI: 10.1109/TPAMI.1986.4767851.

[12] Jörg Conradt, Matthew Cook, Raphael Berner, Patrick Lichtsteiner, Rodney J. Douglas, and Tobi Delbruck. "A Pencil Balancing Robot using a Pair of AER Dynamic Vision Sensors". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2009, pp. 781–784. DOI: 10.1109/ISCAS.2009.5117867.

[13] Tobi Delbruck. "Frame-free dynamic digital vision". In: *Proc. Int. Symp. Secure-Life Electron.* 2008, pp. 21–26.

[14] Tobi Delbruck, Yuhuang Hu, and Zhe He. "V2E: From video frames to realistic DVS event camera streams". In: *arXiv e-prints* (July 2020). URL: http://arxiv.org/abs/2006.07722.

[15] Tobi Delbruck and Manuel Lang. "Robotic Goalie with 3ms Reaction Time at 4% CPU Load Using Event-Based Dynamic Vision Sensor". In: *Front. Neurosci.* 7 (2013), p. 223. DOI: 10.3389/fnins.2013.00223.

[16] Tobi Delbruck and Patrick Lichtsteiner. "Fast Sensory Motor Control Based on Event-Based Hybrid Neuromorphic-Procedural System". In: *IEEE Int. Symp. Circuits Syst. (ISCAS)*. 2007, pp. 845–848. DOI: 10.1109/ISCAS.2007.378038.

[17] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. "FlowNet: Learning Optical Flow with Convolutional Networks". In: *Int. Conf. Comput. Vis. (ICCV)*. 2015, pp. 2758–2766. DOI: 10.1109/ICCV.2015.316.

[18] David Drazen, Patrick Lichtsteiner, Philipp Häfliger, Tobi Delbrück, and Atle Jensen. "Toward real-time particle tracking using an event-based dynamic vision sensor". In: *Experiments in Fluids* 51.5 (2011), pp. 1465–1469. DOI: 10.1007/s00348-011-1207-y.

[19] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification*. Wiley, 2000. ISBN: 978-0471056690.

[20] Gunnar Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Scandinavian Conf. on Im. Analysis (SCIA)*. 2003, pp. 363–370.

[21] R. Fletcher and C. M. Reeves. "Function minimization by conjugate gradients". In: *The Computer Journal* 7.2 (Jan. 1964), pp. 149–154. ISSN: 0010-4620. DOI: 10.1093/comjnl/7.2.149.

[22] S. A. Fortune, M. P. Hayes, and P. T. Gough. "Contrast optimisation of coherent images". In: *IEEE OCEANS*. Vol. 5. 2003, pp. 2622–2628.

[23] C. Fraley. "How Many Clusters? Which Clustering Method? Answers Via Model-Based Cluster Analysis". In: *The Computer Journal* 41.8 (Aug. 1998), pp. 578–588. DOI: 10.1093/comjnl/41.8.578.

[24] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, and Davide Scaramuzza. "Event-based Vision: A Survey". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2020).

[25] Guillermo Gallego, Mathias Gehrig, and Davide Scaramuzza. "Focus Is All You Need: Loss Functions For Event-based Vision". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019, pp. 12280–12289.

[26] Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. "A Unifying Contrast Maximization Framework for Event Cameras, with Applications to Motion, Depth, and Optical Flow Estimation". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 3867–3876. DOI: 10.1109/CVPR.2018.00407.

[27] Guillermo Gallego and Davide Scaramuzza. "Accurate Angular Velocity Estimation with an Event Camera". In: *IEEE Robot. Autom. Lett.* 2.2 (2017), pp. 632–639. DOI: 10.1109/LRA.2016.2647639.

[28] Daniel Gehrig, Antonio Loquercio, Konstantinos G. Derpanis, and Davide Scaramuzza. "End-to-End Learning of Representations for Asynchronous Event-Based Data". In: *Int. Conf. Comput. Vis. (ICCV)*. 2019, pp. 5633–5643.

[29] Arren Glover and Chiara Bartolozzi. "Event-driven ball detection and gaze fixation in clutter". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2016, pp. 2203–2208. DOI: 10.1109/IROS.2016.7759345.

[30] Arren Glover and Chiara Bartolozzi. "Robust visual tracking with a freely-moving event camera". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2017, pp. 3769–3776. DOI: 10.1109/IROS.2017.8206226.

[31] Rafael C. Gonzalez and Richard Eugene Woods. *Digital Image Processing*. Pearson Education, 2009. ISBN: 978-0131687288.

[32] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[33] Germain Haessig, Andrew Cassidy, Rodrigo Alvarez-Icaza, Ryad Benosman, and Garrick Orchard. "Spiking Optical Flow for Event-based Sensors Using IBM's TrueNorth Neurosynaptic System". In: *IEEE Trans. Biomed. Circuits Syst.* 12.4 (Aug. 2018), pp. 860–870. DOI: 10.1109/TBCAS.2018.2834558.

[34] J. P. Hamaker, J. D. O'Sullivan, and J. E. Noordam. "Image sharpness, Fourier optics, and redundant-spacing interferometry". In: *J. Opt. Soc. Am.* 67.8 (Aug. 1977), pp. 1122–1123. DOI: 10.1364/JOSA.67.001122.

[35] Chris Harris and Mike Stephens. "A combined corner and edge detector". In: *Proc. Fourth Alvey Vision Conf.* Vol. 15. 1988, pp. 147–151. DOI: 10.5244/C.2.23.

[36] Yuhuang Hu, Jonathan Binas, Daniel Neil, Shih-Chii Liu, and Tobi Delbruck. "DDD20 End-to-End Event Camera Driving Dataset: Fusing Frames and Events with Deep Learning for Improved Steering Prediction". In: *IEEE Int. Conf. on Intelligent Transportation Systems* (Sept. 2020).

[37] Inivation. *DAVIS240 User Guide*. Accessed: 22/08/2020. 2020. URL: https://inivation.github.io/inivation-docs/Hardwareuserguides/User_guide_-_DAVIS240.html\#aer-format.

[38] Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew J. Davison. "Simultaneous Mosaicing and Tracking with an Event Camera". In: *British Mach. Vis. Conf. (BMVC)*. 2014, pp. 1–12. DOI: 10.5244/C.28.26.

[39] Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. "Real-Time 3D Reconstruction and 6-DoF Tracking with an Event Camera". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2016, pp. 349–364. DOI: \protect{10.1007/978-3-319-46466-4_21}.

[40] Xavier Lagorce, Cédric Meyer, Sio-Hoi Ieng, David Filliat, and Ryad Benosman. "Asynchronous Event-Based Multikernel Algorithm for High-Speed Visual Features Tracking". In: *IEEE Trans. Neural Netw. Learn. Syst.* 26.8 (Aug. 2015), pp. 1710–1720. DOI: 10.1109/TNNLS.2014.2352401.

[41] Wei-Sheng Lai, Jia-Bin Huang, Oliver Wang, Eli Shechtman, Ersin Yumer, and Ming-Hsuan Yang. "Learning Blind Video Temporal Consistency". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2018, pp. 170–185.

[42] A. H. Land and A. G. Doig. "An Automatic Method of Solving Discrete Programming Problems". In: *Econometrica* 28.3 (1960), pp. 497–520.

[43] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning". In: *Nature* 521 (2015), pp. 436–444.

[44] Jun Haeng Lee, Tobi Delbruck, and Michael Pfeiffer. "Training Deep Spiking Neural Networks Using Backpropagation". In: *Front. Neurosci.* 10 (2016), p. 508. DOI: 10.3389/fnins.2016.00508.

[45] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. "A 128×128 120 dB 15 μs latency asynchronous temporal contrast vision sensor". In: *IEEE J. Solid-State Circuits* 43.2 (2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[46] Shijie Lin, Fang Xu, Xuhong Wang, Wen Yang, and Lei Yu. "Efficient Spatial-Temporal Normalization of SAE Representation for Event Camera". In: *ral* 5.3 (2020), pp. 4265–4272.

[47] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. "Microsoft COCO: Common Objects in Context". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48.

[48] Martin Litzenberger, Ahmed Nabil Belbachir, N. Donath, G. Gritsch, H. Garn, B. Kohn, Christoph Posch, and Stephan Schraml. "Estimation of Vehicle Speed Based on Asynchronous Data from a Silicon Retina Optical Sensor". In: *IEEE Intell. Transp. Sys. Conf.* 2006, pp. 653–658. DOI: 10.1109/ITSC.2006.1706816.

[49] Martin Litzenberger, Christoph Posch, D. Bauer, Ahmed Nabil Belbachir, P. Schön, B. Kohn, and H. Garn. "Embedded Vision System for Real-Time Object Tracking using an Asynchronous Transient Vision Sensor". In: *Digital Signal Processing Workshop*. 2006, pp. 173–178. DOI: 10.1109/DSPWS.2006.265448.

[50] Daqi Liu, Alvaro Parra, and Tat-Jun Chin. "Globally Optimal Contrast Maximisation for Event-based Motion Estimation". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)* (2020), pp. 6349–6358.

[51] Min Liu and Tobi Delbruck. "Adaptive Time-Slice Block-Matching Optical Flow Algorithm for Dynamic Vision Sensors". In: *British Mach. Vis. Conf. (BMVC)*. 2018.

[52] Shih-Chii Liu, Tobi Delbruck, Giacomo Indiveri, Adrian Whatley, and Rodney Douglas. *Event-Based Neuromorphic Systems*. John Wiley & Sons, 2015.

[53] Bruce D. Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *Int. Joint Conf. Artificial Intell. (IJCAI)*. 1981, pp. 674–679.

[54] Misha Mahowald. "VLSI Analogs of Neuronal Visual Processing: A Synthesis of Form and Function". PhD thesis. Pasadena, California: California Institute of Technology, May 1992.

[55] Ana I. Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza. "Event-based Vision meets Deep Learning on Steering Prediction for Self-driving Cars". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 5419–5427. DOI: 10.1109/CVPR.2018.00568.

[56] Nico Messikommer, Daniel Gehrig, Antonio Loquercio, and Davide Scaramuzza. "Event-based Asynchronous Sparse Convolutional Networks". In: *Eur. Conf. Comput. Vis. (ECCV)* (2020), pp. 415–431.

[57] Abhishek Mishra, Rohan Ghosh, Jose C. Principe, Nitish V. Thakor, and Sunil L. Kukreja. "A Saccade Based Framework for Real-Time Motion Segmentation Using Event Based Vision Sensors". In: *Front. Neurosci.* 2017, pp. 83–93.

[58] Anton Mitrokhin, Cornelia Fermuller, Chethan Parameshwara, and Yiannis Aloimonos. "Event-based Moving Object Detection and Tracking". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2018, pp. 1–9.

[59] Anton Mitrokhin, Zhiyuan Hua, Cornelia Fermuller, and Yiannis Aloimonos. "Learning Visual Motion Segmentation using Event Surfaces". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2020, pp. 14414–14423.

[60] Anton Mitrokhin, Chengxi Ye, Cornelia Fermuller, Yiannis Aloimonos, and Tobi Delbruck. "EV-IMO: Motion Segmentation Dataset and Learning Pipeline for Event Cameras". In: *IEEE/RSJ Int. Conf. Intell. Robot. Syst. (IROS)*. 2019, pp. 6105–6112.

[61] Elias Mueggler, Christian Forster, Nathan Baumli, Guillermo Gallego, and Davide Scaramuzza. "Lifetime Estimation of Events from Dynamic Vision Sensors". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2015, pp. 4874–4881. DOI: 10.1109/ICRA.2015.7139876.

[62] Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. "Continuous-Time Visual-Inertial Odometry for Event Cameras". In: *IEEE Trans. Robot.* 34.6 (Dec. 2018), pp. 1425–1440. DOI: 10.1109/tro.2018.2858287.

[63] Elias Mueggler, Guillermo Gallego, and Davide Scaramuzza. "Continuous-Time Trajectory Estimation for Event-based Vision Sensors". In: *Robotics: Science and Systems (RSS)*. 2015. DOI: 10.15607/RSS.2015.XI.036.

[64] Elias Mueggler, Henri Rebecq, Guillermo Gallego, Tobi Delbruck, and Davide Scaramuzza. "The Event-Camera Dataset and Simulator: Event-based Data for Pose Estimation, Visual Odometry, and SLAM". In: *Int. J. Robot. Research* 36.2 (2017), pp. 142–149. DOI: 10.1177/0278364917691115.

[65] Richard A. Muller and Andrew Buffington. "Real-time correction of atmospherically degraded telescope images through image sharpening". In: *J. Opt. Soc. Am.* 64.9 (Sept. 1977), pp. 1200–1210. DOI: 10.1364/JOSA.64.001200.

[66] Gottfried Munda, Christian Reinbacher, and Thomas Pock. "Real-Time Intensity-Image Reconstruction for Event Cameras Using Manifold Regularisation". In: *Int. J. Comput. Vis.* 126.12 (July 2018), pp. 1381–1393. DOI: 10.1007/s11263-018-1106-2.

[67] Anh Nguyen, Thanh-Toan Do, Darwin G. Caldwell, and Nikos G. Tsagarakis. "Real-Time 6DOF Pose Relocalization for Event Cameras with Stacked Spatial LSTM Networks". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2019, pp. 1–8.

[68] Zhenjiang Ni, Aude Bolopion, Joel Agnus, Ryad Benosman, and Stéphane Régnier. "Asynchronous Event-Based Visual Shape Tracking for Stable Haptic Feedback in Microrobotics". In: *IEEE Trans. Robot.* 28.5 (2012), pp. 1081–1089. DOI: 10.1109/TRO.2012.2198930.

[69] Zhenjiang Ni, Sio-Hoï Ieng, Christoph Posch, Stéphane Régnier, and Ryad Benosman. "Visual Tracking Using Neuromorphic Asynchronous Event-Based Cameras". In: *Neural Computation* 27.4 (2015), pp. 925–953. DOI: 10.1162/NECO_a_00720.

[70] Zhenjiang Ni, Cécile Pacoret, Ryad Benosman, Sio-Hoï Ieng, and Stéphane Régnier. "Asynchronous event-based high speed vision for microparticle tracking". In: *J. Microscopy* 245.3 (2012), pp. 236–244. DOI: 10.1111/j.1365-2818.2011.03565.x.

[71] Jorge Nocedal and S. Wright. *Numerical Optimization*. Springer-Verlag New York, 2006.

[72] Björn Ommer, Theodor Mader, and Joachim M. Buhmann. "Seeing the Objects Behind the Dots: Recognition in Videos from a Moving Camera". In: *Int. J. Comput. Vis.* 83.1 (Feb. 2009), pp. 57–71. DOI: 10.1007/s11263-009-0211-7.

[73] Garrick Orchard, Ryad Benosman, Ralph Etienne-Cummings, and Nitish V. Thakor. "A spiking neural network architecture for visual motion estimation". In: *IEEE Biomed. Circuits Syst. Conf. (BioCAS)*. 2013, pp. 298–301. DOI: 10.1109/biocas.2013.6679698.

[74] Garrick Orchard, Cedric Meyer, Ralph Etienne-Cummings, Christoph Posch, Nitish Thakor, and Ryad Benosman. "HFirst: A Temporal Approach to Object Recognition". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 37.10 (2015), pp. 2028–2040. DOI: 10.1109/TPAMI.2015.2392947.

[75] Liyuan Pan, Miaomiao Liu, and Richard Hartley. "Single Image Optical Flow Estimation with an Event Camera". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)* (2020), pp. 1672–1681.

[76] Chethan M. Parameshwara, Nitin J. Sanket, Arjun Gupta, Cornelia Fermuller, and Yiannis Aloimonos. "MOMS with Events: Multi-Object Motion Segmentation With Monocular Event Cameras". In: *arXiv e-prints* (2020). URL: https://arxiv.org/abs/2006.06158.

[77] Federico Paredes-Valles, Kirk Y. W. Scheper, and Guido C. H. E. de Croon. "Unsupervised Learning of a Hierarchical Spiking Neural Network for Optical Flow Estimation: From Events to Global Motion Perception". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2019). DOI: 10.1109/TPAMI.2019.2903179.

[78] M. Pawan Kumar, P. H. S. Torr, and Andrew Zisserman. "Learning Layered Motion Segmentations of Video". In: *Int. J. Comput. Vis.* 76.3 (Mar. 2008), pp. 301–319. DOI: 10.1007/s11263-007-0064-x.

[79] R G Paxman and J. C Marron. "Aberration Correction Of Speckled Imagery With An Image-Sharpness Criterion". In: *Statistical Optics*. Ed. by G. Michael Morris. Vol. 0976. International Society for Optics and Photonics. SPIE, 1988, pp. 37 –47. DOI: 10.1117/12.948527.

[80] Xin Peng, Yifu Wang, Ling Gao, and Laurent Kneip. "Globally-Optimal Event Camera Motion Estimation". In: Aug. 2020, pp. 6349–6358. DOI: 10.1007/978-3-030-58574-7_4.

[81] José A. Perez-Carrasco, Bo Zhao, Carmen Serrano, Begoña Acha, Teresa Serrano-Gotarredona, Shouchun Chen, and Bernabé Linares-Barranco. "Mapping from Frame-Driven to Frame-Free Event-Driven Vision Systems by Low-Rate Rate Coding and Coincidence Processing–Application to Feedforward ConvNets". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 35.11 (Nov. 2013), pp. 2706–2719. DOI: 10.1109/tpami.2013.71.

[82] Ewa Piatkowska, Ahmed Nabil Belbachir, Stephan Schraml, and Margrit Gelautz. "Spatiotemporal multiple persons tracking using Dynamic Vision Sensor". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2012, pp. 35–40. DOI: 10.1109/CVPRW.2012.6238892.

[83] Christoph Posch, Teresa Serrano-Gotarredona, Bernabe Linares-Barranco, and Tobi Delbruck. "Retinomorphic Event-Based Vision Sensors: Bioinspired Cameras With Spiking Output". In: *Proc. IEEE* 102.10 (Oct. 2014), pp. 1470–1484. DOI: 10.1109/jproc.2014.2346153.

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

[85] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. "ESIM: an Open Event Camera Simulator". In: *Conf. on Robot. Learning (CoRL)*. 2018, pp. 969–982.

[86] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. "Real-time Visual-Inertial Odometry for Event Cameras using Keyframe-based Nonlinear Optimization". In: *British Mach. Vis. Conf. (BMVC)*. 2017.

[87] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. "Events-to-Video: Bringing Modern Computer Vision to Event Cameras". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019, pp. 3857–3866.

[88] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. "High Speed and High Dynamic Range Video with an Event Camera". In: *IEEE Trans. Pattern Anal. Mach. Intell.* (2020).

[89] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. 2015, pp. 234–241.

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

[91] Bodo Rueckauer and Tobi Delbruck. "Evaluation of Event-Based Algorithms for Optical Flow with Ground-Truth from Inertial Measurement Sensor". In: *Front. Neurosci.* 10.176 (2016). DOI: 10.3389/fnins.2016.00176.

[92] Bodo Rueckauer, Iulia-Alexandra Lungu, Yuhuang Hu, Michael Pfeiffer, and Shih-Chii Liu. "Conversion of Continuous-Valued Deep Networks to Efficient Event-Driven Networks for Image Classification". In: *Front. Neurosci.* 11 (2017), p. 682. DOI: 10.3389/fnins.2017.00682.

[93] Cedric Scheerlinck, Nick Barnes, and Robert Mahony. "Asynchronous Spatial Image Convolutions for Event Cameras". In: *IEEE Robot. Autom. Lett.* 4.2 (Apr. 2019), pp. 816–822. DOI: 10.1109/lra.2019.2893427.

[94] Cedric Scheerlinck, Nick Barnes, and Robert Mahony. "Continuous-time Intensity Estimation Using Event Cameras". In: *Asian Conf. Comput. Vis. (ACCV)*. Dec. 2018, pp. 308–324. DOI: 10.1007/978-3-030-20873-8_20.

[95] Cedric Scheerlinck, Henri Rebecq, Daniel Gehrig, Nick Barnes, Robert Mahony, and Davide Scaramuzza. "Fast Image Reconstruction with an Event Camera". In: *IEEE Winter Conf. Appl. Comput. Vis. (WACV)*. 2020, pp. 156–163.

[96] Cedric Scheerlinck, Henri Rebecq, Timo Stoffregen, Nick Barnes, Robert Mahony, and Davide Scaramuzza. "CED: Color Event Camera Dataset". In: *IEEE Conf. Comput. Vis. Pattern Recog. Workshops (CVPRW)*. 2019. URL: https://timostoff.github.io/19CVPRW.

[97] T. J. Schulz. "Optimal Sharpness Function for SAR Autofocus". In: *IEEE Signal Processing Letters* 14.1 (2007), pp. 27–30.

[98] Loren Shih. "Autofocus survey: a comparison of algorithms". In: *Digital Photography III*. Ed. by Russel A. Martin, Jeffrey M. DiCarlo, and Nitin Sampat. Vol. 6502. International Society for Optics and Photonics. SPIE, 2007, pp. 90 –100. DOI: 10.1117/12.705386.

[99] Amos Sironi, Manuele Brambilla, Nicolas Bourdis, Xavier Lagorce, and Ryad Benosman. "HATS: Histograms of Averaged Time Surfaces for Robust Event-Based Object Classification". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 1731–1740.

[100] Bongki Son, Yunjae Suh, Sungho Kim, Heejae Jung, Jun-Seok Kim, Changwoo Shin, Keunju Park, Kyoobin Lee, Jinman Park, Jooyeon Woo, Yohan Roh, Hyunku Lee, Yibing Wang, Ilia Ovsiannikov, and Hyunsurk Ryu. "A 640x480 dynamic vision sensor with a 9um pixel and 300Meps address-event representation". In: *IEEE Intl. Solid-State Circuits Conf. (ISSCC)*. 2017. DOI: 10.1109/ISSCC.2017.7870263.

[101] Timo Stoffregen. "Event Camera Utility Library". In: 2020. URL: https://timostoff.github.io/projects/ecul.

[102] Timo Stoffregen, Guillermo Gallego, Tom Drummond, Lindsay Kleeman, and Davide Scaramuzza. "Event-Based Motion Segmentation by Motion Compensation". In: *Int. Conf. Comput. Vis. (ICCV)*. 2019, pp. 7244–7253. URL: https://timostoff.github.io/19ICCV.

[103] Timo Stoffregen and Lindsay Kleeman. "Event Cameras, Contrast Maximization and Reward Functions: an Analysis". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019, pp. 12300–12308. URL: https://timostoff.github.io/19CVPR.

[104] Timo Stoffregen and Lindsay Kleeman. "Simultaneous Optical Flow and Segmentation (SOFAS) using Dynamic Vision Sensor". In: *Australasian Conf. Robot. Autom. (ACRA)*. 2017. URL: https://timostoff.github.io/18ACRA.

[105] Timo Stoffregen, Cedric Scheerlinck, Davide Scaramuzza, Tom Drummond, Nick Barnes, Lindsay Kleeman, and Robert Mahony. "Reducing The Sim-to-Real Gap for Event Cameras". In: *Eur. Conf. Comput. Vis. (ECCV)*. 2020, pp. 534–549. URL: https://timostoff.github.io/20ecnn.

[106] Deqing Sun, Xiaodong Yang, Ming-Yu Liu, and Jan Kautz. "PWC-Net: CNNs for Optical Flow Using Pyramid, Warping, and Cost Volume". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 8934–8943.

[107] David Tedaldi, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. "Feature Detection and Tracking with the Dynamic and Active-pixel Vision Sensor (DAVIS)". In: *Int. Conf. Event-Based Control, Comm. Signal Proc. (EBCCSP)*. 2016, pp. 1–6. DOI: 10.1109/EBCCSP.2016.7605086.

[108] Valentina Vasco, Arren Glover, Elias Mueggler, Davide Scaramuzza, Lorenzo Natale, and Chiara Bartolozzi. "Independent motion detection with event-driven cameras". In: *IEEE Int. Conf. Adv. Robot. (ICAR)*. 2017, pp. 530–536.

[109] John YA Wang and Edward H Adelson. "Layered representation for motion analysis". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 1993, pp. 361–366.

[110] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *IEEE Trans. Image Process.* 13.4 (Apr. 2004), pp. 600–612. DOI: 10.1109/tip.2003.819861.

[111] Chengxi Ye, Anton Mitrokhin, Chethan Parameshwara, Cornelia Fermüller, James A. Yorke, and Yiannis Aloimonos. "Unsupervised Learning of Dense Optical Flow and Depth from Sparse Event Data". In: *arXiv e-prints* (2019). URL: http://arxiv.org/abs/1809.08625.

[112] Anthony J. Yezzi and Stefano Soatto. "Deformotion: Deforming Motion, Shape Average and the Joint Registration and Approximation of Structures in Images". In: *Int. J. Comput. Vis.* 53.2 (July 2003), pp. 153–167. DOI: 10.1023/A:1023048024042.

[113] Jason Yu, Adam Harley, and Konstantinos Derpanis. "Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness". In: *Eur. Conf. Comput. Vis. Workshops (ECCVW)*. 2016, pp. 3–10.

[114] Luca Zappella, Xavier Lladó, and Joaquim Salvi. "Motion Segmentation: A Review". In: *Conf. Artificial Intell. Research and Development*. 2008, pp. 398–407.

[115] Linguang Zhang and Szymon Rusinkiewicz. "Learning to Detect Features in Texture Images". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 6325–6333.

[116] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2018, pp. 586–595.

[117] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. "Event-Based Feature Tracking with Probabilistic Data Association". In: *IEEE Int. Conf. Robot. Autom. (ICRA)*. 2017, pp. 4465–4470. DOI: 10.1109/ICRA.2017.7989517.

[118] Alex Zihao Zhu, Dinesh Thakur, Tolga Ozaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. "The Multivehicle Stereo Event Camera Dataset: An Event Camera Dataset for 3D Perception". In: *IEEE Robot. Autom. Lett.* 3.3 (July 2018), pp. 2032–2039. DOI: 10.1109/lra.2018.2800793.

[119] Alex Zihao Zhu, Ziyun Wang, Kaung Khant, and Kostas Daniilidis. "EventGAN: Leveraging Large Scale Image Datasets for Event Cameras". In: *arxiv* (2020). eprint: 1912.01584. URL: http://arxiv.org/abs/1912.01584.

[120] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. "EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras". In: *Robotics: Science and Systems (RSS)*. 2018. DOI: 10.15607/RSS.2018.XIV.062.

[121] Alex Zihao Zhu, Liangzhe Yuan, Kenneth Chaney, and Kostas Daniilidis. "Unsupervised Event-based Learning of Optical Flow, Depth, and Egomotion". In: *IEEE Conf. Comput. Vis. Pattern Recog. (CVPR)*. 2019, pp. 989–997.