Self-Supervised Neuromorphic Perception for Autonomous Flying Robots

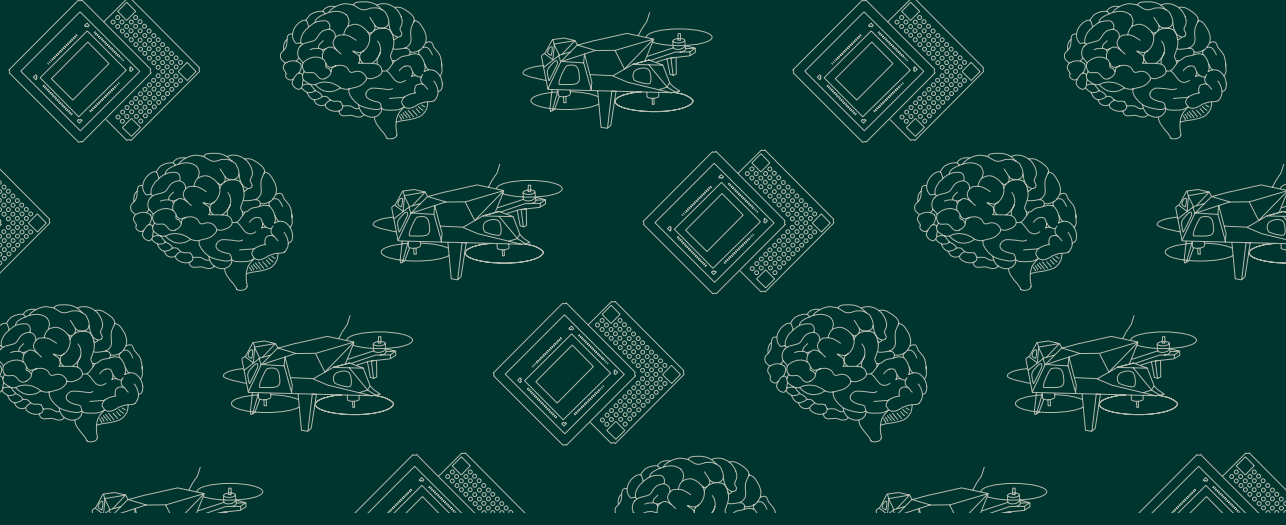
Paredes-Vallés, Federico

DOI
10.4233/uuid:f5f62ff1-28d8-4c5f-93f5-3f9d90d06d28

Publication date
2023

Document Version
Final published version

Citation (APA)
Paredes-Vallés, F. (2023). *Self-Supervised Neuromorphic Perception for Autonomous Flying Robots*. [Dissertation (TU Delft), Delft University of Technology]. https://doi.org/10.4233/uuid:f5f62ff1-28d8-4c5f-93f5-3f9d90d06d28



Self-Supervised Neuromorphic Perception for Autonomous Flying Robots

Federico Paredes-Vallés

Self-Supervised Neuromorphic Perception for Autonomous Flying Robots

Self-Supervised Neuromorphic Perception for Autonomous Flying Robots

Dissertation

for the purpose of obtaining the degree of doctor
at Delft University of Technology,
by the authority of the Rector Magnificus, prof. dr. ir. T. H. J. J. van der Hagen,
chair of the Board for Doctorates,
to be defended publicly on Friday 17 November 2023 at 10:00 o'clock

by

Federico Paredes-Vallés

Master of Science in Aerospace Engineering,
Delft University of Technology, The Netherlands,
born in Murcia, Spain.

This dissertation has been approved by the promotors.

Composition of the doctoral committee:

Rector Magnificus,	chairman
prof. dr. G. C. H. E. de Croon,	Delft University of Technology, promotor
dr. ir. C. De Wagter,	Delft University of Technology, copromotor
Independent members:	
prof. dr. S. M. Bohté,	Centrum Wiskunde & Informatica, The Netherlands
prof. dr. ir. P. Campoy,	Universidad Politécnica de Madrid, Spain
prof. dr. ir. G. Gallego,	Technische Universität Berlin, Germany
dr. Y. Sandamirskaya,	Intel Labs and Zurich University of Applied Sciences, Switzerland
prof. dr. ir. M. Wisse,	Delft University of Technology
prof. dr. ir. M. Mulder,	Delft University of Technology, reserve member

Keywords: Artificial neural networks; Autonomous drone racing; Deep learning; Event-based cameras; Flying robots; Neuromorphic computing; Optical flow; Self-supervised learning; Spiking neural networks; Unsupervised learning

Printed by: Ipskamp Printing, www.ipskampprinting.nl

Cover by: F. Paredes-Vallés and C. Soriano Segura

ISBN 978-94-6366-755-5

An electronic version of this dissertation is available at http://repository.tudelft.nl/.

Contents

Summary ix

Acronyms xi

1 Introduction 1
- 1.1 Sensing: Event-based cameras . . . 2
- 1.2 Algorithms: Spiking neural networks . . . 3
- 1.3 Processing: Neuromorphic processors . . . 4
- 1.4 Neuromorphic computing in robotics . . . 4
- 1.5 Problem statement and research questions . . . 5
- 1.6 Scope and limitations . . . 9
- 1.7 Outline . . . 10

2 Frame-Based, Autonomous Drone Racing: Winning AIRR 2019 11
- 2.1 Introduction . . . 13
- 2.2 Method . . . 18
  - 2.2.1 Drone specifications . . . 18
  - 2.2.2 Perception . . . 18
  - 2.2.3 State estimation . . . 21
  - 2.2.4 Flight planning . . . 22
  - 2.2.5 Control . . . 23
- 2.3 Results and analysis . . . 25
  - 2.3.1 Perception . . . 26
  - 2.3.2 State estimation . . . 28
  - 2.3.3 Control and path planning . . . 30
  - 2.3.4 Robustness . . . 31
  - 2.3.5 Competition outcome . . . 33
- 2.4 Conclusion . . . 33
- 2.5 Discussion . . . 35
- 2.6 Future directions . . . 36

3 Optical-Flow-Aided, Self-Supervised Frame Reconstruction From Events 39
- 3.1 Introduction . . . 41
- 3.2 Related work . . . 42
- 3.3 Method . . . 43
  - 3.3.1 Overview . . . 43
  - 3.3.2 Input event representation . . . 44
  - 3.3.3 Learning optical flow via contrast maximization . . . 44
  - 3.3.4 Learning reconstruction via photometric constancy . . . 45
  - 3.3.5 Network architectures . . . 47

3.4 Experiments . . . 48
3.4.1 Optical flow evaluation . . . 49
3.4.2 Reconstruction evaluation . . . 51
3.4.3 Impact of event deblurring . . . 53
3.5 Conclusion . . . 54

4 Unsupervised Learning of Event-Based Optical Flow with SNNs 55
4.1 Introduction . . . 57
4.2 Related work . . . 58
4.3 Method . . . 62
4.3.1 Optical flow visual observables . . . 62
4.3.2 Adaptive spiking neuron model . . . 64
4.3.3 Stable STDP learning rule . . . 65
4.3.4 Network architecture for motion perception . . . 67
4.4 Experiments . . . 70
4.4.1 Synthetic data experiment . . . 71
4.4.2 Real data experiments . . . 73
4.4.3 STDP evaluation . . . 76
4.4.4 Additional experiments . . . 77
4.5 Conclusion . . . 81

5 Self-Supervised Learning of Event-Based Optical Flow with SNNs 83
5.1 Introduction . . . 85
5.2 Related work . . . 86
5.3 Method . . . 87
5.3.1 Input event representation . . . 87
5.3.2 Learning optical flow via contrast maximization . . . 88
5.3.3 Spiking neuron models . . . 89
5.3.4 Network architectures . . . 90
5.4 Experiments . . . 92
5.4.1 Evaluation of the ANN and SNN architectures . . . 92
5.4.2 Impact of adaptive mechanisms for spiking neurons . . . 94
5.4.3 Further lessons on training deep SNNs . . . 95
5.4.4 Additional experiments . . . 96
5.5 Conclusion . . . 104

6 Robustifying the SSL of Low-Latency, Event-Based Optical Flow 105
6.1 Introduction . . . 107
6.2 Related work . . . 108
6.3 Method . . . 109
6.3.1 Input format and contrast maximization . . . 110
6.3.2 Iterative event warping . . . 111
6.3.3 Deblurring at multiple timescales . . . 113
6.3.4 Network architecture . . . 114

6.4 Experiments . . . 114
6.4.1 Evaluation procedure . . . 115
6.4.2 Optical flow evaluation . . . 115
6.4.3 Additional experiments . . . 119
6.5 Limitations . . . 123
6.6 Conclusion . . . 124

7 Fully Neuromorphic Vision and Control for Autonomous Drone Flight 125
7.1 Introduction . . . 127
7.2 Method . . . 133
7.2.1 Simulating the on-chip spiking neuron model . . . 133
7.2.2 Four-point parametrization to estimate homography . . . 133
7.2.3 Self-supervised learning of event-based optical flow . . . 134
7.2.4 From a vision-based state estimate to control . . . 136
7.2.5 Training control in simulation . . . 136
7.2.6 Hardware setup . . . 138
7.3 Experiments . . . 139
7.3.1 Robust vision-based state estimation . . . 139
7.3.2 Control through visual observables: From sim to real . . . 143
7.3.3 Other examples of versatility and robustness . . . 144
7.3.4 Benchmarking inference speed and energy consumption . . . 147
7.4 Conclusion . . . 148

8 Conclusion 151
8.1 Answers to research questions . . . 151
8.2 Discussion . . . 154
8.3 Outlook . . . 156

A How Do Neural Networks Estimate Optical Flow? 159
A.1 Introduction . . . 161
A.2 Related work . . . 162
A.2.1 Dense optical flow estimation with CNNs . . . 162
A.2.2 Receptive field mapping . . . 164
A.2.3 Aperture problem . . . 165
A.3 Model details . . . 165
A.4 Gabor fitting for translation . . . 166
A.4.1 Methodology . . . 166
A.4.2 Results . . . 169
A.4.3 Temporal bandwidth . . . 173
A.5 Network response to dilation & rotation . . . 174
A.5.1 Limitations of the spectral response profile fitting . . . 174
A.5.2 Methodology . . . 177
A.5.3 Results . . . 178
A.6 Solving the aperture problem . . . 179
A.7 Additional experiments . . . 180
A.7.1 Original vs. our FlowNetS . . . 180
A.7.2 Generalizability to natural images . . . 180

A.8 Discussion . . . 181
A.9 Conclusion . . . 183

B Real-Time, Frame-Based, Dense Optical Flow on a Nano Quadcopter 185
B.1 Introduction . . . 187
B.2 Related work . . . 188
B.2.1 Autonomous navigation of nano quadcopters . . . 188
B.2.2 Real-time dense inference with CNNs . . . 189
B.3 Method . . . 189
B.3.1 Motion boundary detail guidance . . . 189
B.3.2 Strided STDC module redesign . . . 190
B.3.3 Reduced input/output dimensionality . . . 190
B.4 Experiments . . . 191
B.4.1 Implementation details . . . 191
B.4.2 Performance and latency on public benchmarks . . . 192
B.4.3 Additional experiments . . . 192
B.4.4 Obstacle avoidance application . . . 193
B.5 Conclusion . . . 196

References 197

Acknowledgments 231

Curriculum Vitæ 235

List of Publications 237

Summary

In the ever-evolving landscape of robotics, the quest for advanced synthetic machines that seamlessly integrate with human lives and society becomes increasingly paramount. At the heart of this pursuit lies the intrinsic need for these machines to perceive, understand, and navigate their surroundings autonomously. Among the senses, vision emerges as a cornerstone of human perception, providing a wealth of information about the world we inhabit. Thus, it comes as no surprise that equipping robots with vision-based perception capabilities, or computer vision, has captivated researchers for decades. Recent breakthroughs, fueled by the advent of deep learning, have propelled computer vision to new heights. However, challenges persist in leveraging the power of deep learning, as its hunger for computational resources poses hurdles in the realm of robotics, particularly for small flying robots with their inherent limitations of payload and power consumption.

This dissertation embarks on a journey that begins at the intersection of two groundbreaking technologies with the potential to revolutionize computer vision and enhance its accessibility to small robots: event-based cameras and neuromorphic processors. These two technologies draw inspiration from the information processing mechanisms employed by biological brains. Event-based cameras output sparse events encoding pixel-level brightness changes at microsecond resolution, while neuromorphic processors leverage the power of spiking neural networks to realize a sparse and asynchronous processing pipeline.

Throughout this dissertation, comprehensive investigations have been conducted, presenting innovative solutions and advancements in the fields of computer vision and robotics. The thesis begins by presenting the winning solution of the 2019 AIRR autonomous drone racing competition, which showcases a monocular vision-based navigation approach specifically designed to address the limitations of conventional sensing and processing methods. Moreover, it explores the bridging of the gap between event-based and frame-based domains, enabling the application of conventional computer vision algorithms on event-camera data. Building upon these achievements, the thesis introduces a pioneering spiking architecture that enables the estimation of event-based optical flow, with emergent selectivity to local and global motion through unsupervised learning. Additionally, the thesis presents a framework that addresses the practicality and deployability of the models by training spiking neural networks to estimate low-latency, event-based optical flow with self-supervised learning. Finally, this dissertation culminates with a demonstration of the integration of neuromorphic computing in autonomous flight. By utilizing an event-based camera and neuromorphic processor in the control loop of a small flying robot for optical-flow-based navigation, this research showcases the practical implementation of neuromorphic systems in real-world scenarios. Overall, our studies demonstrate the benefits of incorporating neuromorphic technology into the vision-based state estimation pipeline of autonomous flying robots, paving the way for the development of more power-efficient and faster neuromorphic vision systems.

Acronyms

AC	Accumulate operation
AER	Address event representation
AI	Artificial intelligence
AIRR	Artificial Intelligence Robotic Racing
ALIF	Adaptive leaky integrate-and-fire
ANN	Artificial neural network
APS	Active pixel sensor
BL	Bottom-left corner
BPTT	Backpropagation through time
BR	Bottom-right corner
CL	Cluster
CNN	Convolutional neural network
CPU	Central processing unit
CUBA-LIF	Current-based leaky-integrate-and-fire
DNN	Deep neural network
DRL	Drone Racing League
EPE	Endpoint error
EW	Event warping
FC	Fabric controller
FLOP	Floating point operation
FOE	Focus of expansion
FOV	Field of view
FPS	Frames per second
FW	Frame warping
FWL	Flow warp loss
GPU	Graphics processing unit
GT	Ground truth
HDR	High dynamic range
I/O	Input/Output
IoU	Intersection over union
IWE	Image of warped events
LIF	Leaky integrate-and-fire
LPIPS	Learned perceptual image patch similarity
LTD	Long-term depression
LTP	Long-term potentiation
MAC	Multiply and accumulate operation
MAE	Mean absolute error
MAV	Micro air vehicle

MHE	Moving horizon estimator
MSE	Mean squared error
PD	Proportional-derivative controller
PI	Proportional-integral controller
PID	Proportional-integral-derivative controller
PLIF	Presynaptic leaky integrate-and-fire
PnP	Perspective-n-point
RANSAC	Random sample consensus
RQ	Research question
RSAT	Ratio of the squared average timestamps
SLAM	Simultaneous localization and mapping
SNN	Spiking neural network
SoC	System-on-chip
SSIM	Structural similarity
SSL	Self-supervised learning
STDP	Spike-time-dependent plasticity
SWaP	Size, weight, and power
TL	Top-left corner
TR	Top-right corner
VIO	Visual inertial odometry
WTA	Winner-take-all

1

Introduction

The overarching objective of robotics research is to design and create advanced synthetic machines that will ultimately enhance human lives and societal well-being. To achieve this objective, these systems will need to perform tasks autonomously or collaboratively with humans, while effectively interacting with and navigating their environments. Therefore, their ability to perceive the world and make sense of it is of paramount importance. In particular, vision is arguably the most important sense for humans [1], as it provides the richest source of information about the environment. For this reason, it is no wonder that providing robots with vision-based perception capabilities (further referred to as *computer vision*) has been the focus of intense research in robotics for decades [2].

Fueled by the convergence of computational intelligence and engineering prowess, computer vision has made tremendous progress in recent years. In particular, the advent of deep learning has led to a paradigm shift, with deep neural networks now outperforming handcrafted algorithms on many tasks [3, 4]. However, training these networks usually requires large amounts of labeled data, which is often difficult and/or costly to obtain in a robotics context. Moreover, their deployment on robots that are safe to interact with remains a challenge, as they are typically computationally expensive and thus require powerful hardware and large amounts of energy. This is especially problematic for flying robots, which are inherently constrained by their payload and power consumption [5].

In pursuit of mission-capable, autonomous, small flying robots, many roboticists have turned to nature for inspiration. Although the examples are numerous, flying insects stand out as perhaps the most successful case of agile and robust flying machines [6–8]. They are able to navigate complex environments at high speeds, while avoiding obstacles and reacting to unexpected events. In addition, they are able to perform these tasks with a brain that is orders of magnitude smaller than that of a human [9]. Therefore, the quest for inspiration from the animal kingdom not only frequently manifests itself in the design of the "body" of some these robots [10, 11], but also, since more recently, in the design of their on-board sensors and processors [12–14]. The field of neuromorphic engineering has emerged as a promising alternative to conventional computing for edge devices, as it aims to develop an approach to sensing and computing that, inspired by the function

and structure of biological brains, is able to perform complex tasks with low latency and minimal energy consumption [15–17].

The combination of neuromorphic vision sensors, further referred to as *event* or *event-based* cameras interchangeably [18], and neuromorphic processors [14, 19] running trainable, *spiking neural networks* (SNNs) [20] holds the promise of highly efficient and high-bandwidth vision-based perception and processing. The challenge, however, lies in the fact that the working principle of these systems is fundamentally different from those of conventional vision sensors and processors (see Fig. 1.1), and therefore, another paradigm shift is needed to exploit their full potential. In this dissertation, we address this challenge by proposing novel learning-based solutions for neuromorphic, vision-based perception and processing. In addition, we demonstrate their effectiveness in the context of the autonomous flight of small flying robots, as they form a prime example of resource-constrained platforms with urgent need for low-energy, low-latency sensing and processing. The proposed solutions are formulated using either unsupervised or self-supervised learning (SSL) not only to remove the need for labeled data, but also to ensure that the learned solutions perform well on the robots — avoiding the "reality gap" that arises when learning on synthetic data.

The following sections use Fig. 1.1 as a common thread to provide an overview of previous research on the different elements of the neuromorphic computing field that are relevant to this dissertation (namely neuromorphic vision sensors, algorithms, and processors), and on how they compare to their conventional counterparts. Thereafter, the research objective and questions of this thesis are formulated, and its outline is presented.

1.1 Sensing: Event-based cameras

Contrary to standard cameras, which make use of an external clock to sample light and provide images at a constant frequency (see Fig. 1.1, top left); event-based cameras are bio-inspired vision sensors that feature a pixel array that asynchronously reacts to pixel-level brightness changes [18]. This working principle results in a sparse output stream consisting of digital, timestamped *events*; with every event representing a (log–)brightness change of a predefined magnitude, and encoding the x–y location and polarity of the change. As illustrated in Fig. 1.1 (bottom left), the data-rate of the output event stream depends on the dynamics of the visual scene. The larger the brightness change, the more events per second are generated (and vice versa). More importantly for this dissertation is that, under constant illumination, events are triggered by the apparent motion (i.e., optical flow [21]) of contrast in the image space [18].

Because of the sparse and asynchronous operating principle, these cameras are characterized by several advantages with respect to their conventional frame-based counterparts: a very high temporal resolution (in the order of microseconds), a sub-millisecond latency, and (potentially) a low power consumption (in the order of milliwatts) [18]. Despite the paradigm shift, the potential that these advantages entail for many fields has quickly triggered the generation of an extensive body of literature [22] on, not only how to use events to solve a wide range of computer vision tasks [23–28], but also on how to best process the events to retain the advantages over frame-based algorithms [29–32]. Nevertheless, regarding the latter, the current mainstream solution is to buffer events over substantially long time windows and create grid-like representations [33] that are compatible with artificial

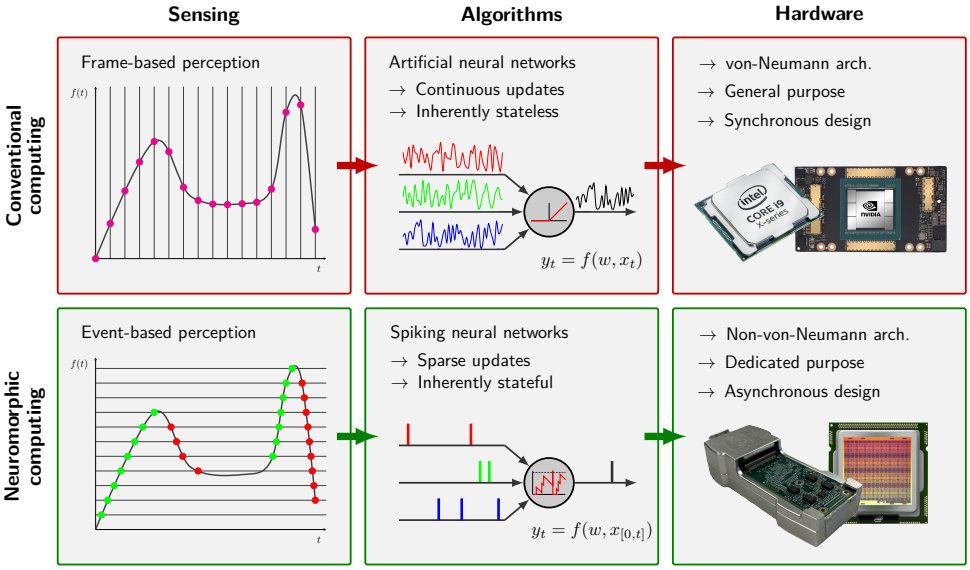


Figure 1.1: Elements from the field of neuromorphic computing relevant to this dissertation (*bottom*), and how they compare to their conventional counterparts (*top*). In this thesis, we cover both approaches to computing in the context of the vision-based autonomous flight of small flying robots (see Chapters 2 and 7).

neural networks (ANNs). These representations allow ANNs to extract the information encoded in the collective of events, but usually come at the cost of a high latency despite the high levels of accuracy reported [34]. In addition, to run under real-time constraints, ANNs usually require GPU-based hardware accelerators that, being characterized by a power consumption usually in the order of tens of watts, limit their deployability on edge platforms such as small flying robots.

1.2 Algorithms: Spiking neural networks

Instead of processing event data with ANNs (see Fig. 1.1, top center), the neuromorphic approach to retaining the advantages of event cameras is to process the sparse and asynchronous events as they come, i.e., one-by-one in a per-event processing fashion, with SNNs. As in Fig. 1.1 (bottom center), these architectures are bio-inspired computational models comprised of spiking neurons: processing units that produce binary activations (i.e., *spikes*) whenever their internal state (usually referred to as *membrane potential* or *voltage*) surpasses a predefined threshold. This spike-based operating principle leads to a sparse and asynchronous computing which not only is a perfect match for event cameras due to their common nature, but also has the potential of being low-latency and low-power when deployed on dedicated processors [14, 19].

The limited availability of neuromorphic hardware is one factor that has hindered the widespread adoption of SNNs in fields like computer vision and robotics. However, another significant challenge comes from the fact that learning algorithms designed for ANNs do not transfer well to the spiking domain. This has driven extensive research

mainly in two directions: (i) finding ANN-SNN conversion strategies that lead to high efficiency gains without drops in accuracy [35, 36], and (ii) direct training of SNN with traditional backpropagation (through time) using a few adjustments to deal with the non-differentiability of the spiking activation function [20, 37, 38]. This lack of consensus on how to design training pipelines for SNNs, which in turn underscores the complexity of the problem, has limited the application of these architectures to often less complicated, discrete problems [23, 24, 39, 40]. In addition, note that, prior to this dissertation, learning in the SNN domain was dominated by spike-timing-dependent plasticity (STDP) [41]: a form of Hebbian (i.e., unsupervised) learning [42] that adapts the strength of a connection between two neurons based on their correlated activity, and was only successfully applied in the computer vision domain to image classification tasks [43–47].

1.3 Processing: Neuromorphic processors

In order to unlock the potential of SNNs as low-power and low-latency computing solutions for event-based computer vision, dedicated hardware accelerators that are able to exploit the sparse and asynchronous nature of these networks are needed. In this regard, the field of neuromorphic engineering has made tremendous progress in recent years with the development of several neuromorphic processors, such as the TrueNorth chip [48], the BrainScaleS wafer-scale system [49], the DynapCNN [50], the Loihi processors [14, 51], and others [48, 52]. Despite their differences (e.g., number of neurons and synapses), these processors share a few characteristics, some of which are highlighted in Fig. 1.1 (bottom right). Mainly, they differ from general-purpose, von-Neumann architectures (e.g., CPUs, GPUs) in that, instead of having separate memory and processing units, the memory is in close proximity to the processing. These two elements are arranged in several (small) neurocores [19], which in turn are interconnected via the address-event representation protocol [53] following a network-on-chip communication scheme [54, 55]. With these architectures, the different layers of an SNN can be distributed among the available neurocores. This parallelization of the storage and computation, together with the sparse, binary activations of the spiking neurons, allows for a low-power and low-latency operation [19]. Note that, in spite of this potential, most of the processors remain as research prototypes and are not commercially available, thus limiting their adoption.

1.4 Neuromorphic computing in robotics

As aforementioned, the field of neuromorphic computing has the potential to revolutionize the way robots perceive and interact with their environments. The combination of event-based vision sensors and neuromorphic processors has the potential to enable robots to perform complex tasks with low latency and minimal energy consumption. However, despite the extensive (yet young) body of literature on neuromorphic computing, its application to robotics has been limited to a few examples, and even fewer in the context of autonomous flight. In this section, we provide a brief overview of these examples.

Many of the applications that have been explored over recent years in the event-based camera domain can be used, in one way or another, to provide robots with some of the visual information required to autonomously navigate an environment. This includes information about the robot's ego-motion, the motion of other objects in the scene, and

the structure of the environment; and usually comes from algorithms performing optical flow estimation [25, 33, 56, 57], feature detection and tracking [58–60], 3D reconstruction [61–63], visual (inertial) odometry [64–66], or simultaneous localization and mapping [67], among others. Early works incorporating an event camera in the control loop of robotic platforms already demonstrated the advantages of these sensors through low-latency state updates and efficient data processing [68, 69]. Despite their algorithmic simplicity, these findings motivated researchers in aerial robotics to adopt event-based cameras in detriment of their conventional, frame-based counterparts. The work of Pijnacker-Hordijk *et al.* [70], which was later extended by Scheper *et al.* [71], was the first in showing these sensors in the control loop of a flying robot, with the task being optical-flow-based landing [72–75]. Thereafter, Vidal *et al.* [67] and Sun *et al.* [76] demonstrated the event camera in the context of visual inertial navigation, also for flying robots. In both cases, the authors reported that event-based cameras, mainly because of their high temporal resolution and data sparsity, allowed for pushing the limits of their robotic platforms to flying speeds that were out of reach for frame-based cameras (because of their low update rates and/or motion blur). This was later confirmed by research on event-based avoidance of static and dynamic obstacles [77–81]. Note that the algorithms powering these examples were deployed on conventional computing hardware (e.g., CPUs, GPUs) in all cases.

Robotic examples featuring event-based vision algorithms running on neuromorphic processors are rare, despite their aforementioned potential, and so far have been limited in complexity. Galluppi *et al.* presented in [82] a robotic setup in which an event camera was connected to a SpiNNaker processor [52] to allow a ground robot to differentiate between two lights flashing a different frequencies. Years later, in [83], Milde *et al.* designed and deployed an SNN for event-based obstacle avoidance and target acquisition on a ground robot equipped with a ROLLS processor [84]. The most recent example, and the closest to an aerial robotic context, is the work of Vitale *et al.* in [85]. Inspired by [86], the authors deployed an SNN on a Loihi processor [14] on board a bench-fixed dual-rotor to align the roll angle of this platform with a black-and-white disk located in front of an event camera. The SNN was in charge of not only finding the line to align with, but also of implementing the controller for generating the motor commands. Note that, in all these examples, the SNNs were handcrafted and no learning/evolutionary/optimization algorithm of any kind was used in their design.

Lastly, despite not having a camera in the loop, it is still relevant to highlight the neuromorphic research that has recently taken off on low-level control for aerial robots. The works from Stroobants *et al.* [87–89] demonstrate the joint potential of (trainable) SNNs and neuromorphic processors to achieve highly efficient, IMU-based state-estimation and control in resource-constrained, aerial platforms.

1.5 Problem statement and research questions

In this thesis, we focus on the neuromorphic computing elements that have been discussed in previous sections, examining their application in the context of flying robots. The aim is to achieve vision-based navigation with an event-based camera and a neuromorphic processor, running an SNN for perception, on board and in the control loop of a robot. This is yet to be accomplished in the literature, mostly due to the inherent complexity of solving real-world, large-scale problems with (trainable) spiking networks that can, in turn,

be deployed on neuromorphic processors. Consequently, the core question driving the research conducted in this dissertation can be formulated as follows:

Driving Research Question

How can neuromorphic perception and processing be incorporated into the vision-based, state-estimation pipeline of an autonomous flying robot?

This question can be approached from various perspectives since, regardless of the computing paradigm, there is no unique way of providing robots with vision-based navigation capabilities for autonomous flight. Among the available options, this thesis focuses on how this can be achieved using 2D motion information in the image plane, i.e., optical flow [21]. This choice is motivated by two primary reasons. Firstly, optical flow is core to many computer vision algorithms and, in the event-camera domain, it represents a relatively young but active area of research. However, existing literature often emphasizes maximizing accuracy while overlooking algorithm latency [33, 34]. Therefore, we believe that any contributions made towards our central research question will also be of relevance to the wider computer vision community. Secondly, there is extensive literature that, after drawing (once again) inspiration from flying insects, has demonstrated the potential of optical flow in tasks such as autonomous landing [72–75], attitude control [90] or obstacle avoidance [91]. Considering these factors, the specific problem statement for this dissertation is formulated as follows:

Problem Statement

How can optical-flow-based autonomous navigation be realized with an event-based camera and a neuromorphic processor in the control loop of a flying robot?

Our attention will now shift to the requirements needed to address this question to the best of our ability. In order to achieve this, the problem statement has been subdivided into five research questions, which were addressed in a sequential manner as the research unfolded.

Frame-based perception: Understanding the challenge

Before making the jump to neuromorphic technology and optical flow, it is of importance to understand the limits and complexities of vision-based autonomous flight with frame-based cameras and conventional computing. One particular scenario that pushes this technology to its limits is autonomous (indoor) drone racing [92], where flying robots must autonomously navigate a predefined track as quickly as possible using only on-board resources. In such conditions, the maximum speed achievable by the drones is typically determined by the robustness and computational efficiency of the control and perception algorithms, rather than by the physical constraints of the platform itself [93]. The first research question of this dissertation was then formulated in this context as follows:

RQ1: Research Question 1

How can fast, autonomous flight through gates be achieved with frame-based perception and conventional processing in a GPS-denied environment?

This question led to the development of a robust yet efficient vision-based navigation solution (frame-based, monocular) for very fast autonomous flying robots. In fact, this solution proved to be successful, as evidenced by our victory in the 2019 Artificial Intelligence Robotic Racing (AIRR) Circuit, where it outperformed other state-of-the-art methods, both monocular and stereo [94], when deployed on the same robot (i.e., same sensors and processors). However, this competition also provided first-hand experience on some of the inherent limitations of conventional computing for robotic applications. Specifically, motion blur, the limited (and fixed) update rate of the camera, and the need for large amounts of labeled data for our machine learning algorithm were the primary bottlenecks of our approach.

Bridging event-based and frame-based computer vision

As discussed in Section 1.1, event cameras react to changes in brightness by generating events with a very high temporal resolution [18]. This makes these sensors particularly robust to motion blur issues [95], but downstream applications, such as the perception algorithm developed for AIRR, remain limited due to the sparse and asynchronous nature of event data. This need to bridge the gap between the event-based and frame-based computer vision domains leads to the second research question:

RQ2: Research Question 2

How can we leverage the knowledge of the inner working of event cameras to learn event-based frame reconstruction in a self-supervised fashion?

The outcome of this research was the development of the first solution to the problem of frame reconstruction from events that, instead of relying on labeled data, leverages the relation between the events, optical flow, and the brightness signal [18] to train ANNs in a self-supervised manner. Event-based optical flow was also obtained through SSL using the (at the time) state-of-the-art method from literature [33, 96, 97].

Unsupervised learning for motion-selective SNNs

Despite the potential of the previous solution to reconstruct blur-less images from the events at a much higher rate than frame-based cameras (and the research direction that this unlocks), the downstream algorithms would still be applied on conventional images. This entails losing their potential of being low-latency and low-power since ANNs would be required to perform this events-to-frame conversion as a first stage. For this reason, the third research question directed our attention towards SNNs:

RQ3: Research Question 3

How can a spiking neural network learn to develop event-based motion selectivity in an unsupervised fashion?

This third research question led to the development of the first SNN architecture in which selectivity to both local and global motion emerged in an unsupervised manner using event-camera data. The learning rule employed was a novel formulation of STDP, a correlation-based, local plasticity rule inspired by biological processes that, up until this point, had primarily been successful in classification tasks [43–47]. Through this research, we were able to confirm that event cameras and SNNs are an ideal combination, as the latter can extract meaningful patterns from the input events while maintaining high sparsity and low latency levels. However, the absence of supervision posed challenges in controlling the learned features and comparing their performance with other approaches.

SSL of low-latency, event-based optical flow with SNNs

With the aim of improving the robustness of learning event-based optical flow estimation, the fourth research question of this dissertation was formulated as follows:

RQ4: Research Question 4

How can low-latency, event-based optical flow be learned in a self-supervised fashion with spiking neural networks?

The outcome of this research was a novel self-supervised pipeline that, by leveraging the knowledge on event-based optical flow and SNNs from previous research questions, allows for the training of models that can be scaled up to high inference frequencies. Contrary to the event-camera literature, which, as aforementioned, is mostly focused on maximizing accuracy, this pipeline targets minimal latency without compromising performance. Backpropagation through time was used to promote SNNs to exploit their internal dynamics to extract motion information from the sparse input events, which are processed nearly as they a triggered.

Fully neuromorphic pipeline for vision-based flight

Lastly, what was remaining to provide an answer to the driving research question of this dissertation was to demonstrate the effectiveness of the proposed SSL pipeline in the context of the autonomous flight of a robot. This was done by formulating the fifth and final research question as follows:

RQ5: Research Question 5

How can a spiking neural network be trained in a self-supervised fashion to perform event-based optical flow estimation while running on a neuromorphic processor in the control loop of an autonomous flying robot?

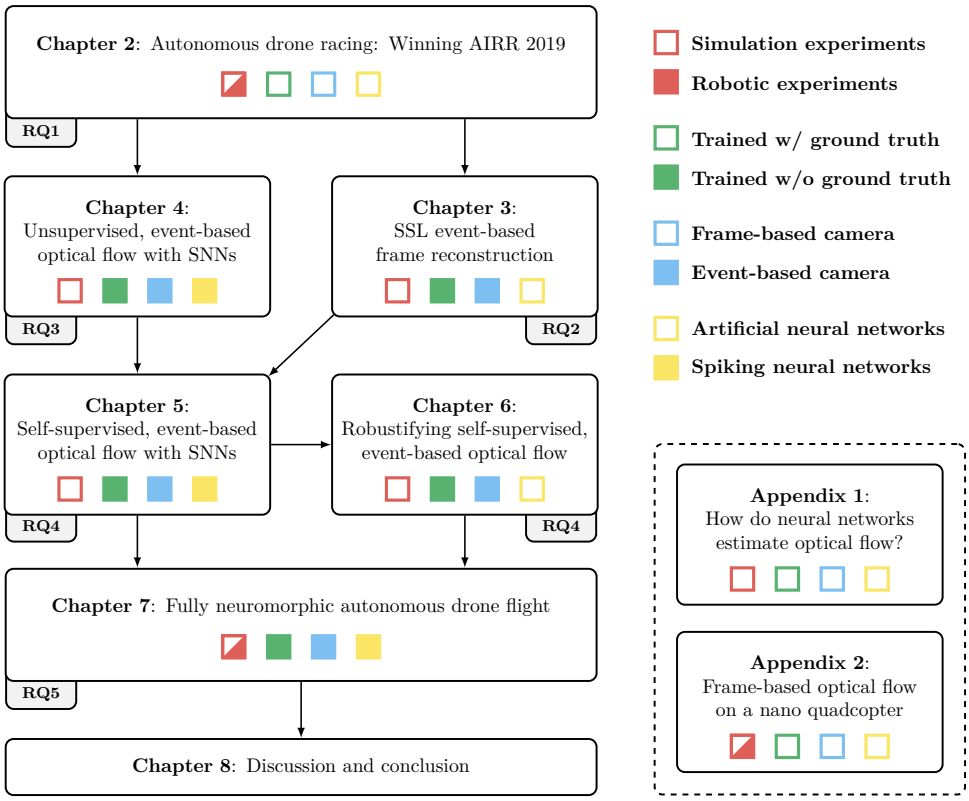


Figure 1.2: Outline of the rest of this dissertation.

The outcome of this research was the pioneering demonstration of the potential of neuromorphic sensing and processing in achieving vision-based autonomous flight. Specifically, we showed that the proposed SSL pipeline can be used to effectively train SNNs that, when deployed on Intel's Loihi neuromorphic processor [14], can perform optical-flow-based state estimation with low latency and power consumption, while maintaining high levels of accuracy. This demonstration was conducted in a real-world scenario, where a quadrotor successfully accomplished various tasks, including hovering, landing, and sideways maneuvering, in an autonomous manner.

1.6 Scope and limitations

To provide an answer, out of the many possible, to the driving research question of this dissertation, several limitations were imposed on the scope of the conducted research. Some of these limitations were already alluded to in previous sections, but they are summarized here for clarity. Firstly, the research primarily centers around event-based optical flow and its application for estimating the ego-motion of the flying platform. Other uses of optical flow, such as obstacle avoidance, are outside the scope of this thesis. Secondly, the emphasis

is on training neural network architectures, including both ANNs and SNNs, without relying on labeled or synthetic data. Therefore, only unsupervised and self-supervised training frameworks are considered, despite the state-of-the-art in most computer vision domains being dominated by pure supervised learning approaches. Lastly, although training without labels unlocks the possibility of refining the models in an online fashion during deployment, what is in the scope of this dissertation is offline (i.e., batch) training. This means training the models on general-purpose processors (e.g., CPUs, GPUs) before deployment. By setting these limitations, the research aims to provide a specific answer to the driving research question while acknowledging and addressing the challenges inherent in the chosen scope.

1.7 Outline

As depicted in Fig. 1.2, the subsequent chapters of this dissertation present comprehensive responses to the five research questions derived from the problem statement, ultimately leading to an answer to the driving research question. In Chapter 2, we address **RQ1** by describing the winning solution to the 2019 AIRR autonomous drone racing competition: a monocular vision-based navigation approach designed around the limitations of conventional sensing (i.e., frame-based) and processing (i.e., general-purpose, synchronous) [98, 99]. Moving forward, Chapter 3 already takes a leap into event-based cameras and addresses **RQ2** by focusing on the problem of frame-based reconstruction from the events. This chapter aims to bridge the gap between the event-based and frame-based domains [100], facilitating the application of conventional computer vision algorithms for downstream tasks. Thereafter, the focus is already on estimating event-based optical flow with SNNs, and Chapter 4 proposes the first spiking architecture in which selectivity to both local and global motion emerges in an unsupervised fashion from the input events [101], thus addressing **RQ3**. To enhance the deployability of the models, Chapter 5 describes the first self-supervised framework for training SNNs to estimate low-latency, event-based optical flow [102]. This pipeline was later extended in Chapter 6 [103] to address some of the limitations inherent to learning event-based optical flow in a self-supervised fashion. The combined contributions of these two chapters effectively tackle **RQ4**. Finally, neuromorphic computing was demonstrated in the context of autonomous flight in Chapter 7, where we showed an event-based camera and a neuromorphic processor in the control loop of a flying robot performing optical-flow-based navigation [104]. The neuromorphic processor was used to effectively run the SNN with low latency and power consumption, while maintaining competitive levels of accuracy. This research addresses **RQ5**.

To conclude, Chapter 8 provides a summary of the main contributions made throughout the research. It discusses the implications of these contributions and explores potential future research directions in the field of neuromorphic computing for robotics. Lastly, Chapters A and B present two additional research directions that were explored in parallel to the main research questions. Specifically, Chapter A presents a neuropsychology-inspired investigation on how deep neural networks estimate optical flow from frame-based data [105]. This study provides valuable insights into the limitations and robustness of these architectures, offering recommendations for future work. Chapter B, on the other hand, focuses on demonstrating frame-based optical flow running on ultra-low power computing hardware in the context of vision-based obstacle avoidance for nano flying robots [106].

2

Frame-Based, Autonomous Drone Racing: Winning AIRR 2019

Autonomous drone racing presents an extreme challenge in the field of robotics, requiring flying robots to navigate complex tracks at high speeds. However, achieving similar levels of adaptability and speed as human pilots requires addressing fundamental problems in computer vision, artificial intelligence, and robotics while operating under severe resource constraints. In this chapter, we present the winning solution of the 2019 Artificial Intelligence Robotic Racing Circuit, an autonomous drone race competition in which all participating teams used the same robot, which was equipped with multiple frame-based cameras and a GPU accelerator. Our proposed approach combines an efficient yet robust gate-detection pipeline leveraging artificial neural networks and model-based active vision, along with a reliable state estimation and risk-aware control algorithms. We thoroughly analyze the performance of each component using both competition and simulation data, aiming to address the core challenge of designing a vision-based navigation pipeline that enables flying robots equipped with conventional cameras and processors to achieve robust and highly agile flight.

The contents of this chapter have been published in:

C. De Wagter†, **F. Paredes-Vallés**†, N. Sheth†, G. C. H. E. de Croon, *The sensing state-estimation and control behind the winning entry to the 2019 Artificial Intelligence Robotic Racing competition*, Field Robotics, 2022.

C. De Wagter†, **F. Paredes-Vallés**†, N. Sheth†, G. C. H. E. de Croon, *Learning fast in autonomous drone racing*, Nature Machine Intelligence (NMI), 2021.

† Equal contribution.

Contribution: The research leading to this chapter's work was the result of a collaborative effort with multiple researchers, all from the Micro Air Vehicle Laboratory (Delft University of Technology). We all equally contributed to the conception of the approach and to the analysis and interpretation of the results. Specifically, I designed the perception module for the relative localization of the flying robot with respect to the gates in the drone racing environment. In addition, I made major technical contributions to all the other parts of the pipeline and their integration into the robot.

2.1 Introduction

Artificial intelligence (AI) has seen tremendous progress over the last decade, especially due to the advent of deep neural networks [3, 4]. The major milestones in the history of AI have always been associated with competitions against human experts. These competitions clearly show the increasing complexity of the tasks in which AI can extend beyond human performance. In 1997, IBM's Deep Blue showed the power of search methods combined with expert systems [107] by beating the world champion in the game of chess, Garry Kasparov. Chess is a fully observable, turn-based game, with $\approx 10^{123}$ possible game states. After chess, the AI community started to aim for the game of Go, which has a much larger branching factor that also results in a much higher number of $\approx 10^{360}$ possible game states, rendering most search methods ineffective. In 2017, the Master version of Google Deepmind's AlphaGo beat Ke Jie, the top-ranked Go player at the time. AlphaGo used an elegant combination of Monte Carlo tree search and deep neural networks for evaluating board positions [108]. In 2019, Google Deepmind's AlphaStar reached a GrandMaster status in the real-time strategy game StarCraft II [109]. This game represents yet a higher complexity, as it involves real-time instead of turn-based play, partial observability, and a large and varied action space. Finally, even the online multiplayer game Dota 2 was tackled with reinforcement learning, forming another step in complexity [110].

Robotics will form a new frontier in AI research since the associated problems are even more complex [111]. Typical robotics problems are high-dimensional, continuous, and only partially observable. Moreover, and most importantly, robots have to operate in the real world, of which many relevant aspects remain hard to model or simulate. Sample-intensive learning methods may apply to simplified robot models in simulation, allowing for faster than real-time learning, but transferring them to an actual robotic system typically leads to a reality gap [112–114] that substantially reduces performance. One part of the reality gap is the difference in sensory input like visual appearance and sensor noise. The other part of this reality gap is the specifics of a robot itself, concerning both its "body" (energy source, structure, sensors, actuators) and "brain" (processing power, memory). For example, there may be unmodeled aerodynamic effects or different timings in the perception-action cycle of the actual robotic hardware.

One extreme challenge at the moment for AI in robotics is formed by autonomous drone racing. Similar to human drone races, the goal for the drones is to finish a pre-determined racing track in as short a time as possible. The drones have to race by using only their on-board resources, which are heavily restricted in terms of size, weight, and power (SWaP) [5]. To be successful, the drones will have to fly through complex tracks at very high speeds (human racers reach speeds of up to 190 km/h). This means that they need a fast perception-action cycle on lightweight hardware, which additionally should be robust, as the margin for error is small.

The research on autonomous drone racing finds its roots in seminal work on agile and aggressive flight [115–117]. The focus of many of these early studies was mostly on high-performance control, outsourcing sensing and state estimation to external motion tracking systems and associated central computers. Later studies focused on also getting the sensing and state estimation on board, allowing the drones to perform quick maneuvers through gaps [118, 119]. A real drone race additionally requires the drone to detect racing gates in more complex spaces, with multiple gates and potential distractors in view, while

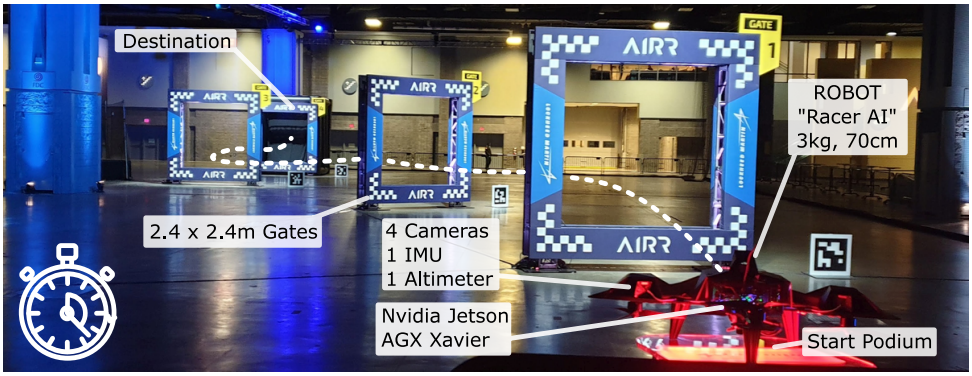


Figure 2.1: Setup of the Washington race (i.e., second race of the AIRR season).

not only passing one gate, but flying a full trajectory in sequence, dealing with unforeseen deviations on the way. The research on drone racing received a boost by the first-ever autonomous drone race competition, organized in conjunction with the IROS robotics conference in 2016, in Daejeon, South Korea [120]. This competition let the participants free in their choice of platform and only required that all sensing and processing took place on board. The first competition showed the difficulty of the problem, with the winner reaching 10 gates at an average speed of 0.6 m/s. This is in stark contrast to the impressive racing performance reached a year later by Morrell *et al.* [121], whose drone only lost by a few seconds from an expert human pilot on their track. In-competition flight speeds remained inferior to out-of-competition flight speeds also over the ensuing years, with IROS drone race speeds of the winner reaching 2.5 m/s in 2018 [122]. The reason for this mostly lies in the real-world aspects of the competitions. They take place in environments previously unknown by the teams, with no opportunity for benign, solution-specific changes, and little time for adapting the developed solution to the environment in situ. Moreover, competitions often pose a more challenging environment, with gates located slightly differently than on the pre-communicated maps or even moving during the race, unforeseen lighting effects optimized for spectators rather than for drones, and large crowds of moving people around the flight arena.

In this chapter, the winning approach to the 2019 AI Robotic Racing (AIRR) competition is presented. This competition which is also refered to as *AlphaPilot*, was organized by Lockheed Martin and the Drone Racing League (DRL) in 2019 and had a grand prize for the best AI of $1M. The AIRR competition strives to support the development of AI for racing drones that will be able to surpass human drone racing pilots. It is completely different from the previous autonomous drone racing competitions in many aspects. For example, the competition did not take place on a single day at a conference but had two phases: a qualifier phase and a competition phase. In the qualifier phase, 424 teams registered worldwide and had to qualify by performing a computer vision task, racing in simulation, and describing their proposed approach and team composition. The competition phase, with only 9 teams participating, was also unique as it was organized as a complete season with three seasonal races and a championship race. The races themselves were organized

by DRL as e-sports events, also aiming for the amusement of the audience, adding specific requirements on the teams and robots. Moreover, the organization provided all teams with the same type of racing drone, developed by the organizers. These drones were equipped with four high-resolution, wide field-of-view cameras, and an NVIDIA AGX Xavier board to run the embedded AI (see Fig. 2.1). Hence, the robotic hardware was the same for all teams, making the competition only about the difference between the AI software. Moreover, the teams had very little direct access to the racing drone hardware, making it very hard to get acquainted with the hardware, perform calibrations, and identify potential reality gaps. The amount of flight testing was low and happened in different conditions than the races in terms of light, room size, and even air density. Finally, during the races, the AI code was uploaded to refurbished drones that had never flown this particular code before without the possibility to improve in between the runs. Consequently, the AI developed for the competition had to be very data efficient and robust.

The goal was to mainly develop AI solutions on the provided DRL simulator, which figured a substantial reality gap in terms of drone dynamics and sampling characteristics of especially the camera. The simulator, which contained unknown drone dynamics, did have a hardware-in-the-loop setup for the processing, i.e., it communicated with the same NVIDIA Xavier board and allowed teams to accurately test the computational effort of the developed algorithms.

Human-inspired, gate-based approach

In our approach to developing an AI for the AIRR competition, we used the characteristics and restrictions of the competition as a point of departure (see Figs. 2.1 and 2.2). First and foremost, we desired to fly as fast as possible, ideally close to the physical limits of the drone. This implied that we did not use perception methods that would restrict the drone's maximum speed. Importantly, it meant the exclusion of state-of-the-art methods for feature-based visual inertial odometry (VIO), e.g. [123], since the blurry images that occur at higher speeds lead to more difficulties in finding and tracking features. The reliance on this type of VIO was one of the main reasons that the runner-up team limited their velocity to 8 m/s [94]. Moreover, current accurate VIO methods have the disadvantage that they are computationally intensive. For similar reasons, we did not employ feature-based simultaneous localization and mapping (SLAM) methods, e.g. [124], as used by the winning team in the IROS 2017 competition [92]. Additionally, SLAM methods have difficulties handling changes in the map, like the foreseen gate displacements.

Instead, we drew inspiration from human pilots who focus greatly on the gates, while combining their observations with knowledge of the drone's responses to control inputs and an approximate map of the track (see Fig. 2.2). Hence, we developed an accurate, robust, and computationally efficient monocular gate detection method. We aimed to process images at the fastest speeds the drone could achieve. Whereas previous competitions contained gates of a uniform, unique color [92, 120], the AIRR competition featured more complex gates, precluding hand-designed detection methods as in Li *et al.* [125, 126]. Relative localization can also not be done with standard rectangle-based, single-shot detectors [127] since the bounding boxes generated by such methods by themselves do not allow for an accurate determination of the drone's relative pose. Furthermore, we did not choose a deep neural network that immediately maps images to relative pose, as in [122, 128]. Such networks

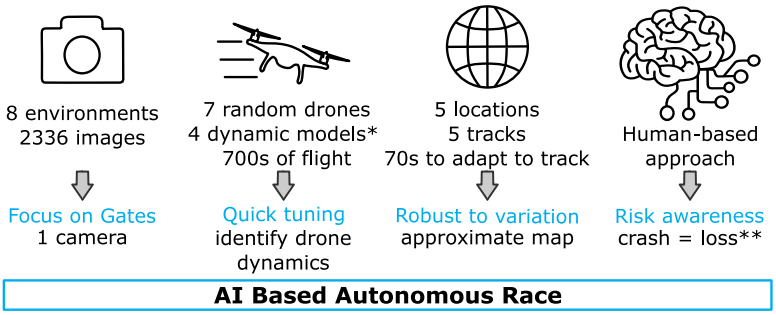


Figure 2.2: The human-inspired approach to drone racing using single-camera position estimation, dynamics prediction, a rough model of the track, and risk-aware control. (*) Two simulators, 5000 ft-dynamics, sea-level-dynamics. (**) Loss of log files, test opportunities, or races.

experience difficulties when multiple gates are in sight and are more difficult to analyze and fix.

We developed a novel gate segmentation deep neural network (DNN) called "GateNet" to create a fast vision pipeline that is minimally sensitive to the various known perturbations at high velocities. This includes the increasing levels of motion blur, rolling shutter deformation, and the possible absence of texture in large parts of the scene due to the lack of features in parts of the man-made environment. The DNN also was used to overcome over-exposure as teams could control neither the exposure settings of the camera nor the light conditions. The vision pipeline also had to deal with the presence of moving unknown entities, the absence of precise frame timing information and unknown shutter-times.

Subsequently finding gate corners was done with very efficient active perception [125]. Especially on flying robots where every gram matters, active vision is highly relevant [129–131] and we show it can be part of successful engineering designs. Pose estimates finally can then be computed using perspective-n-point (PnP) combined with the racecourse map.

Further inspired by human first-person-view pilots' ability to predict drone motion, we enhanced the drone's state estimates with model-based predictions from a dynamic model fitted on flight data. Merging the visual measurements with the predictions is then done with a random sample consensus (RANSAC) based moving horizon estimator (MHE) from previous work [126] but extended to better estimate the drone's yaw angle during the race.

Concerning control, we designed a strategy that would permit high speeds but would allow for flying very early on and would have short intuitive tuning cycles, given the little available flight time. As a result, promising methods such as deep reinforcement learning [132] or imitation learning [133] were ruled out. The short timeline and little flight time would not allow for a thorough investigation of the reality gap between the drone and the simulator with methods like abstraction [114, 134]. Even online adaptation [135] would yield limited benefits in a race where every drone only flew once, measurements could be very noisy, and the time to the first gate is a mere 2 seconds. Finally, the long down-times, loss of log files (stored in RAM) and failed competition runs in case of a crash, made risk management a crucial part of the control development.

While perception-aware trajectory generation [136] can optimize speed and perception, it does not take into account that collision risks depend on the relative position to the

gate. The control is therefore designed from a gate-centered perspective in which risks and constraints, but also position uncertainty, vary depending on the distance to the gate. The controller makes use of classical control theory but was gradually augmented to fly increasingly close to the platform limits. This allowed us to start flying early in the process and gather crucial log files to steadily investigate drone limitations while minimizing risks.

On top of the initial scheme, we implemented an open-loop, full-throttle take-off called "boost" to overcome sensor boot-time delays. We adopted a pitch-for-altitude controller to maintain altitude when thrust saturated, and an offline optimized gate-approach-line strategy. Finally, we developed a human-inspired, risk-aware strategy that speeds the robot up substantially when far from obstacles or aligned with the next gate, but that slows down when uncertain or misaligned. Whereas both in computer vision and robotics a lot of research effort is invested in increasing the accuracy of methods, we put computational efficiency at the core of our approach. The reason for this is that control performance is not only determined by accuracy but also by the control delay, two factors which are most often on a trade-off with each other. Moreover, not saturating processing power allowed us to have additional threads logging all data (images, states, measurements, etc.). This data was extremely important to estimate the drone's model and fusion parameters, and for retraining and improving the perception pipeline.

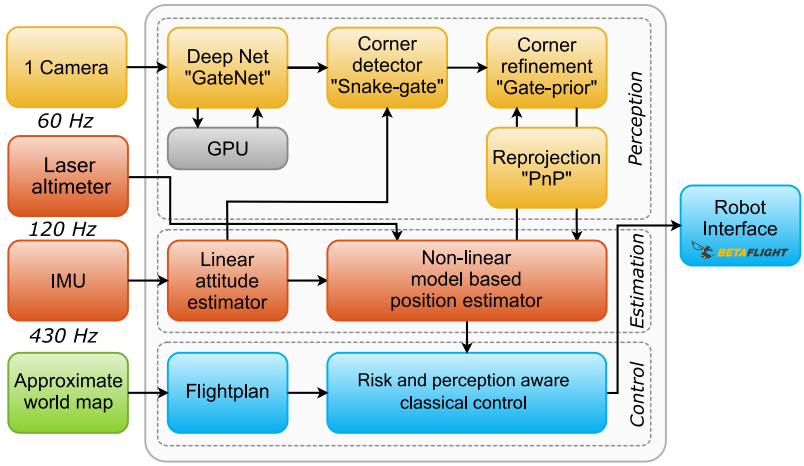


Figure 2.3: System schematic of the approach.

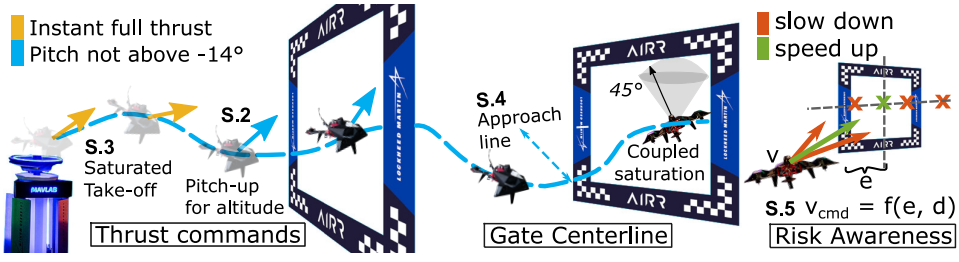


Figure 2.4: Illustration of the various controller modifications, further denoted by S.2–S.5.

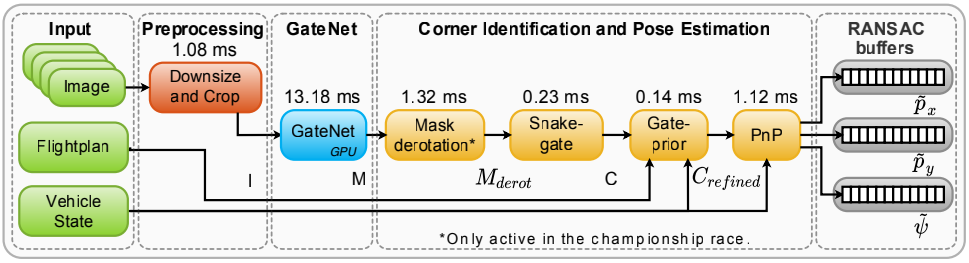


Figure 2.5: Overview of the perception pipeline and execution time of each of the submodules on the Jetson AGX Xavier. Both RANSAC and the remaining vision pipeline were running on separate CPU threads. The DNN was executed on the GPU.

The full scheme is shown in Fig. 2.3, and in the next section, we give an in-depth explanation of the implementation used in the competition and our approach.

2.2 Method

2.2.1 Drone specifications

All participating teams in the 2019 AIRR competition operated the same race drone type called "Racer AI" (see Fig. 2.1). This plus-configuration quadcopter was approximately 70 cm in diameter, weighed around 3 kg, and had a thrust-to-weight ratio of about 1.4. It was equipped with two sets of forward-facing stereo camera pairs which looked sideways with an angle of ±30° and up with an angle of 15° (see Fig. 2.8). The cameras were the global-shutter, color Sony IMX 264 sensor, which provided 1200×720 resolution images at a rate of 60 Hz. The wide field of view lens had a focal length of ≈590 pixels. Besides cameras, the Racer AI had a Bosch BMI088 IMU, with a measurement range of ±24 g and ±34.5 rad/s (with a resolution of 7e-4 g and 1e-3 rad/s) provided at an update rate of 430 Hz; and a downward-facing laser rangefinder Garmin LIDAR-Lite v3 with a measurement range of up to 40 m (resolution of 0.01 m) and update rate of 120 Hz. As the embedded computer, the Racer AI was equipped with an NVIDIA Jetson AGX Xavier, containing a GPU with 512 CUDA cores and an 8-core ARM CPU. It ran Linux with the PREEMPT RT kernel patch. Lastly, the Racer AI had a BetaFlight low-level autopilot controlling the angular velocities of the drone and accepting commands at a rate of 50 Hz.

2.2.2 Perception

The perception modules were executed sequentially in a dedicated thread, while a separate thread did the logging of images. We only used one out of the four cameras, as this setup matches the challenge that human pilots have to face. Although monocular vision is more challenging in terms of depth perception, it entails less computational load and calibration requirements. Moreover, it allows for lighter and hereby faster drones to be created in the future. Since none of the cameras faced forward, we had the drone fly in the direction of the optical axis of the selected camera. The original 1200×720 images provided by the camera were first centrally cropped to 720×720 to remove parts of the own robot that were in sight, and then downsized to 360×360 using bilinear interpolation. No radial lens

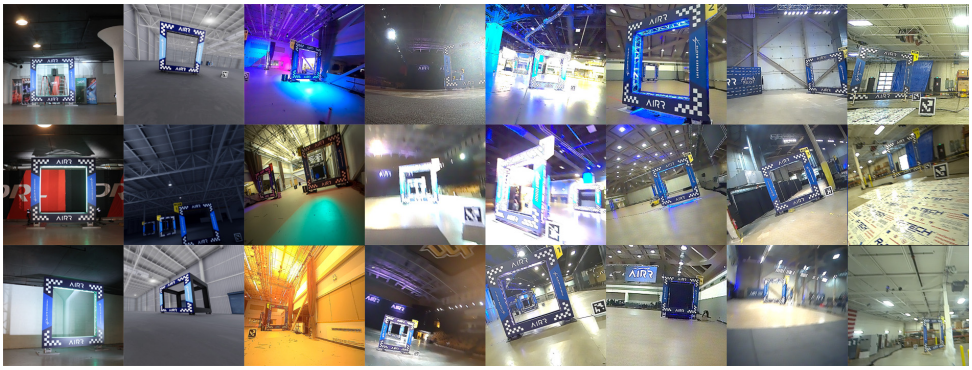


Figure 2.6: Overview of training images. From left to right per column: qualification data, simulator, workshop, Orlando, Washington, Baltimore, Austin and the testing area in Littleton.

undistortion was performed on the images, but instead, the lens parameters were used in the pose estimation. The correction of only a few corners through reverse lens parameters saves a lot of CPU load when compared to correcting all pixels.

GateNet: Gate detection by semantic segmentation

In the first stage of our perception pipeline, GateNet was used to transform each resized input image $\mathcal{I}$ into a binary mask $\mathcal{M}$ that segments all visible gates regardless of their distance to the camera. GateNet is a fully convolutional deep neural network architecture that consists of a 4-level U-Net [137] with [64, 128, 256, 256] convolutional filters of size 3×3 and (element-wise sum) skip connections. All layers use ReLU activation functions except the final prediction layer, which uses a sigmoid to keep $\mathcal{M}$ in the range [0, 1]. GateNet was trained in a supervised manner through a weighted combination of the binary cross-entropy and soft-Dice loss functions on a dataset eventually consisting of 2336 images recorded in 8 distinct environments (see Fig. 2.6). The ground-truth mask for each sample in the training dataset was manually annotated. We augmented the training data through random affine transformations, variations in the HSV color space, and artificial motion blur. The blur consists of the convolution of a squared averaging filter of random size between 5 and 15 pixels, and random orientation. For the deployment on NVIDIA's Jetson AGX Xavier, we ported the network to TensorRT 5.0.2.6 with a batch size of 1 and full precision FP32 mode. The network contains 1723.7k trainable parameters. The execution time, measured on the CPU, to send an image, have it processed on the GPU, and retrieve the result was 13.18 ms (≈75.9 Hz) (see Fig. 2.5).

We deployed a different GateNet version in each competition race, with the only differences being the size of the training dataset and the data augmentation mechanisms. Networks were always trained from scratch when changing the augmentation mechanisms. Before new races, we quickly fine-tuned the models to deployment environments with training data from the test sessions through incremental training after adding hand-labeled training data from typically roughly 50 manually selected difficult images.

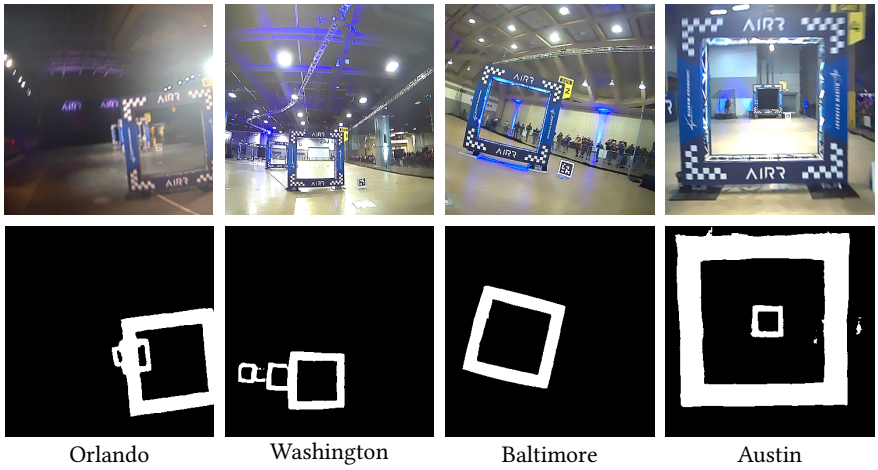


Figure 2.7: Qualitative results of the generated segmentation masks by the final GateNet model from on-board images of each of the events.

Snake-gate: Active vision for corner identification

To then retrieve the inner and outer corners of only the next gate, even when multiple gates were in sight, we employed a variation of the lightweight, active-vision algorithm [131] known as snake-gate and first presented in [125]. This two-stage, iterative sampling method reports the desired gate corners in a fixed order. The first stage starts at the intersection of the vertical and horizontal histograms of $\mathcal{M}$. The histograms represent the number of white pixels per column or row and the maxima in the histogram point to the pixel with a high probability of belonging to the closest (largest) gate. From that point, it starts sampling white pixels in a fixed direction in the image space (i.e., top-left, top-right, bottom-right, or bottom-left) until the corresponding outer corner is found. Thereafter, the sampling direction changes until all corners have been identified. A pixel is considered to be a corner if the sampling method cannot progress in the specified direction. Once the outer contour was identified, we used the centroid of this set of corners as the starting point to identify the inner corners of the gate by sampling black pixels instead. To overcome incorrect corner association at bank angles greater than $|45|$ deg, the mask $\mathcal{M}$ was first de-rotated using the drone's estimated roll angle around the optical axis.

Snake-gate requires (i) the mask of a gate to be continuous, and (ii) no gate overlap in the image space. The first requirement was normally met thanks to our robust and accurate GateNet model (see Fig. 2.7). However, most of the AIRR tracks had gates placed in front of each other, violating the second requirement. To cope with this, we developed gate-prior.

Gate-prior: Sanity check on the identified corner locations

Snake-gate does not provide any form of confidence metric regarding the identified corners. Therefore we developed a sanity check. The expected 3D location of the next gate based on the internal flight plan was projected into the image space, and is called "gate-prior". We then compared the sides and angles of both inner and outer contours of this projection to those of the identified gate and only accepted the validity of a corner if the error of the

associated sides and angle was below 25%. Rejected corners of a contour with at least two valid corners were corrected using the shape of the gate-prior (see Fig. 2.11). This actively reduced the number of outliers and it improved the robustness to challenging scenarios that could lead to discontinuous masks (e.g., HDR scenes, fast motion, partial gate in the image) and gate overlap (see Fig. 2.11). If no valid corners were found for two full seconds, a recovery mechanism would override "gate-prior" and accept any gate corners given by snake-gate.

Localization via perspective-n-point

The size, approximate location, and orientation of the AIRR gates were known in all races. The estimation of the drone's position and orientation was found by solving the PnP problem, using the identified corner locations in the image space and their corresponding 3D locations (maximum eight corners). As in [125, 126], instead of relying on pure vision-only PnP, we combined it with the on-board attitude estimate of the drone to retrieve the camera's 3D location, as this was shown to be more robust in drone racing conditions. We solved the PnP problem with an iterative method based on Levenberg-Marquardt optimization, which minimizes the reprojection error and requires at least three point-correspondences.

2.2.3 State estimation

Attitude estimation was performed using a complementary filter fed with gyroscope and accelerometer data. Position and velocity estimates were propagated using a drag and thrust model in the "flat-body" frame ${}^{fb}R_{\mathrm{W}}$ shown in Fig. 2.8, which is a local tangent plane rotated by the yaw ψ of the drone. The predicted drag specific forces in the flat-body frame (a_x^{fb}, a_y^{fb}) were modeled as:

$$\begin{bmatrix} a_x^{fb} \\ a_y^{fb} \end{bmatrix} = \begin{bmatrix} \hat{d}_x & 0 \\ 0 & \hat{d}_y \end{bmatrix} \underbrace{\begin{bmatrix} c_\psi & s_\psi \\ -s_\psi & c_\psi \end{bmatrix}}_{{}^{fb}R_{\mathrm{W}}} \begin{bmatrix} v_x^{\mathrm{W}} \\ v_y^{\mathrm{W}} \end{bmatrix} \tag{2.1}$$

where c_ψ and s_ψ present the cosine and sine of the yaw angle, (v_x^{W}, v_y^{W}) the velocities in the world frame and the linear drag parameters $\hat{d}_x$ and $\hat{d}_y$ were found by fitting the integrated path to best match the known gate locations using flight logs. To reduce the drift of drag-model predictions in the world frame a^W, an additional first-order linear filter called "alpha" fused the drag specific force model with accelerometer measurements (subscript m). The resulting prediction model is:

$$\mathbf{a}^{\mathrm{W}} = \alpha \cdot {}^{\mathrm{W}}R_{fb} \begin{bmatrix} a_x^{fb} \\ a_y^{fb} \\ 0 \end{bmatrix} + (1-\alpha) {}^{\mathrm{W}}R_B \begin{bmatrix} a_x^B \\ a_y^B \\ 0 \end{bmatrix}_m + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} + {}^{\mathrm{W}}R_B \begin{bmatrix} 0 \\ 0 \\ a_z^B \end{bmatrix}_m \tag{2.2}$$

where α determines the ratio between predictions based on drag model or accelerometers, ${}^{\mathrm{W}}R_B$ is the rotation from body-to-world, g is the local gravitational acceleration and $\mathbf{a}_{x,y,z}^B$ are the accelerometer measurements. A value of $\alpha = 85\%$ was found to yield the best predictions. The predicted velocity and position in the world frame were obtained through integration: $\mathbf{v}^W = \int \mathbf{a}^W$ and $\mathbf{p}^W = \int \mathbf{v}^W$.

For the altitude, a Kalman filter merged the low-pass filtered (6 Hz cut-off) vertical accelerations and the low-pass (50 Hz cutoff) filtered attitude-corrected downward-facing laser range measurements.

Horizontal position corrections were performed by merging the PnP estimates in world coordinates from the perception pipeline with the predicted path (see Fig. 2.9). As vision estimates occasionally still contained large errors, an MHE based on RANSAC was used. It is directly adopted from [126], but besides position and velocity corrections, yaw corrections were also made to account for initial heading alignment errors and yaw integration drift. The corrections were done by running the MHE filter independently on each axis (p_x, p_y, ψ). Separate buffers with a maximum of 180 samples hold information about PnP estimates and delay-compensated inertial estimates.

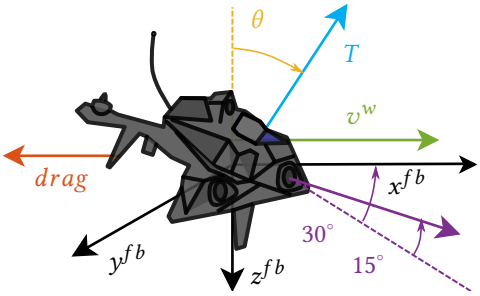


Figure 2.8: Axis definition of the drag-based model with the camera viewing angle and its 30° degree offset in the $x-y$ plane, and 15° up.

Samples older than 2 seconds were removed. The delay was fixed to 20 measurements (at 0.04 second intervals) on the drone and 110 when run as hardware-in-the-loop simulation. Using RANSAC with 200 iterations with 80% of the samples, the filter fitted the predicted world path and heading with the PnP measurements. The result was a position correction $\hat{e}_{p_{x,y}}$ and a velocity correction $\hat{e}_{v_{x,y}}$ on top of the predicted estimates to obtain the current state at each time step. The heading correction $\hat{e}_{\psi}$ on the other hand, was only applied once upon passing each gate. The least-squares fit for RANSAC was written as $\hat{x} = (A^T A + \partial I)^{-1} A^T y$ where the prior δ ensured a preference for small corrections in velocity estimates, and $\hat{x}$, A and y were defined to map the position and velocity errors in function of time Δt over the buffer with samples $n = 1$ to N (given only for p_x):

$$\underbrace{\begin{bmatrix} \Delta p_x|_{n=1} \\ \Delta p_x|_{n=2} \\ \vdots \\ \Delta p_x|_{n=N} \end{bmatrix}}_{y} = \underbrace{\begin{bmatrix} 1 & \Delta t|_{n=1} \\ 1 & \Delta t|_{n=2} \\ \vdots & \vdots \\ 1 & \Delta t|_{n=N} \end{bmatrix}^{\dagger}}_{A} \cdot \underbrace{\begin{bmatrix} \hat{e}_{\mathrm{p}} \\ \hat{e}_{\mathrm{v}} \end{bmatrix}}_{x} \tag{2.3}$$

This estimator ran in a separate thread and was executed each time there were enough samples in the buffers. When a gate was crossed, the prediction was reset to the value of the state and the MHE buffers were cleared. A minimum number of 27 PnP estimates (18 in simulation due to lower frame rates) were then required before the solution was allowed to jump to the new estimate.

2.2.4 Flight planning

Path planning was done by tracking position waypoints from a list of approximate gate locations. We used the locations provided by the organizers during the practice runs and a

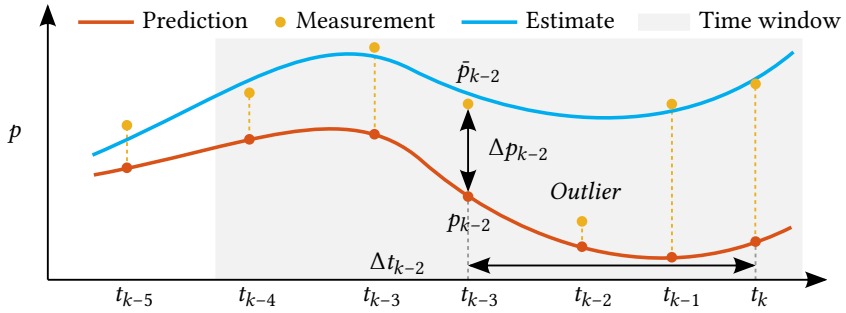


Figure 2.9: Moving horizon estimator. The predictions x_k at various time steps k are scaled ($\hat{e}_{v_{x,y}}$) and offset ($\hat{e}_{p_{x,y}}$) to best match the visual measurements $\bar{x}_k$ within the 2 second time window. RANSAC is used to remove outliers by sampling N random points from the buffer.

manually updated flight plan during the races to better correspond to the perceived gate locations, which corresponds to true locations only in case of perfect calibrations. The altitude setpoint was kept constant at 1.75 meters since all gate heights were identical and fixed. To better align with gates, the current commanded position $p_{x,y,z(cmd)}$ was not the gate waypoint itself, but a temporary waypoint placed 6 meters perpendicularly in front of the gate along its so-called "centerline." When the robot got closer than 7 meters to this target, the point remained at 7 meters from the drone and moved towards the gate along the centerline until reaching the gate center.

2.2.5 Control

The heading ψ was commanded to align the active camera with the center of the next gate. The selected active camera was either the right-center camera for tracks with right turns or the left-center camera for tracks with only left turns. This maximized the time gates were in-view and minimized the open-loop odometry phases. When arriving close to a gate, heading commands could get unnecessarily aggressive and reduce the quality of the model-based predictions. The yaw rate was therefore limited to 180 deg/s and the robot even stopped aligning the camera with the current gate when getting closer than 2.2 meters from the gate. This corresponds to the point where the gate would not be completely visible anymore.

The proportional position controller mapped the horizontal position errors in the flat-body frame $\Delta p^{fb}_{x,y}$ to commanded horizontal velocities $v^{fb}_{x,y}$. An additional proportional term was mixed in the lateral axis to have to robot align with the gate by computing the perpendicular distance towards the gate centerline $\Delta p^{\text{gate}}_{y}$. The total lateral control became:

$$v^{fb}_{y} = (1 - \alpha_{\text{center}}) \cdot k_{p1} \cdot \Delta p^{fb}_{y} + \alpha_{\text{center}} \cdot k_{p2} \cdot \Delta p^{\text{gate}}_{y} \tag{2.4}$$

where $k_{p1} = 0.45$ and $k_{p2} = 0.45$ were gains, and the mixing parameter α_{center} would determine if the robot flew directly to the waypoint along the shortest path or followed the gate centerline to improve perception and improve approach angles at the cost of increased distance to fly. Since the distance between gates was small in the last races, no obstacles were present along the gate centerlines, and some gates were placed at shallow angles, in the end, a value of $\alpha_{\text{center}} = 60\%$ was used.

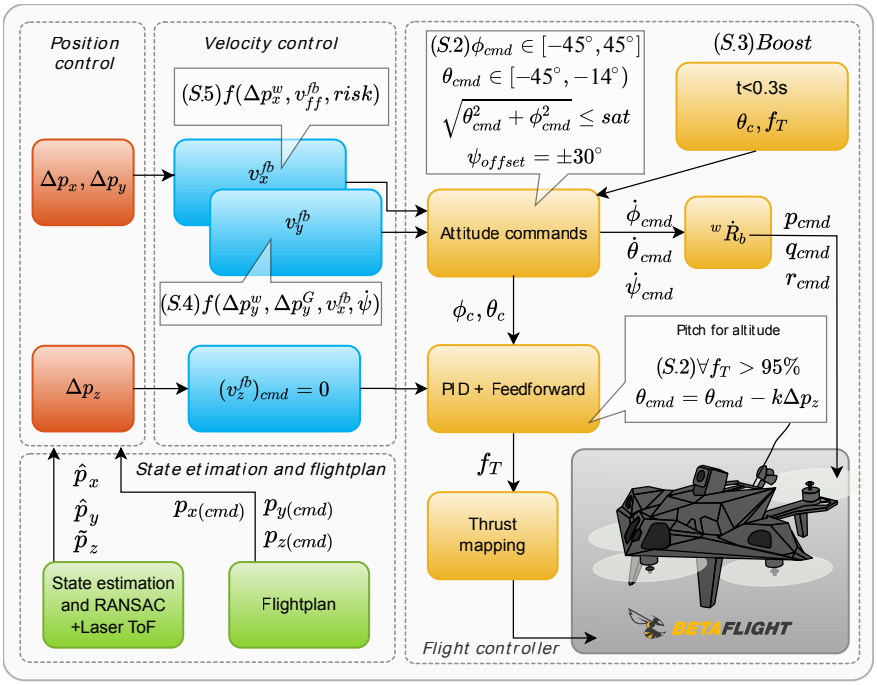


Figure 2.10: Schematic representation of the cascaded PID control pipeline with several enhancements for fast and risk-aware flight S.2–S.5.

The forward velocity was a function of the distance to the gate Δp_x^{fb} and the current motion vector. Far from the gate (>10 meters), the winning entry used a commanded velocity of 7.5 m/s. Then the speed was reduced to an alignment speed of 5.5 m/s. Once the state estimation predicted that the robot was sufficiently well aligned to pass through the gate within 80 centimeters of the center, it was allowed to speed up as much as possible. If the robot got so close to the gate that the gate was not in-sight anymore, to minimize the open-loop time spent in the gate, it would always accelerate if it had not reached at least the gate-crossing speed of 7.5 m/s.

A velocity control loop converted the velocity commands to desired pitch and roll angles using a feedforward gain of 0.009 rad/m/s and a velocity error feedback gain of 0.4 rad/m/s. The commanded pitch angle was constrained between -45° and -14° pitch down, hence preventing pitching up. This served in keeping a good forward speed and helped perception as the fixed 15° upward-looking angle of the camera meant that the bottom of the gate could fall outside the field of view when pitching up. Moreover, slower speeds and fast deceleration into the own propeller downwash also led to a larger drift of our drag-based odometry approach. The total bank angle was saturated at 45 degrees by maintaining the ratio between pitch and roll and is referred to as coupled saturation. Finally, a rate limiter of 320 deg/sec was applied to reduce the effects of attitude changes on the available throttle.

Thrust commands were generated using traditional PID with a feedforward hover-thrust of 67% at sea-level and 73% in Littleton, which scaled with the inverse cosine of the total bank angle of the drone. Saturation was applied to the altitude error Δp_z^w (+/- 2 meter) and T_{cmd} (15 – 100%). An integrator windup protection was added to the PID loop by not integrating when the T_{cmd} saturation was active.

When the throttle would saturate in full throttle, a "pitch-for-altitude" controller was activated. As the throttle saturation could occur both in forward flight as well as in turns, instead of implementing a traditional "pitch-up to climb" controller, a max-bank reduction was used on top. In fast forward flight, the pitch-for-altitude controller would command pure pitch-up while during saturating turns the maximum roll angle of 45 degrees of roll was also reduced by the same amount.

Attitude control was achieved by computing feedforward rate commands for the BetaFlight low-level autopilot that was running a rate controller tuned by DRL. This was augmented with a bounded feedback controller on the error between the commanded and the current attitude. The errors in attitude are given as e_ϕ, e_θ, e_ψ while the feedback and feedforward gains are $k_p = 0.12$ and $k_{ff} = 1/dt$. The change in desired pitch and roll angles in the given discrete time step are noted $\Delta\theta_{\text{cmd}}$ and $\Delta\phi_{\text{cmd}}$ with time step dt. The rate commands in roll, pitch, and yaw then become:

$$\begin{bmatrix} p_{\text{cmd}} \\ -q_{\text{cmd}} \\ r_{\text{cmd}} \end{bmatrix} = \begin{bmatrix} 1 & 0 & -s_\theta \\ 0 & c_\phi & c_\theta s_\phi \\ 0 & -s_\phi & c_\theta c_\phi \end{bmatrix} \begin{bmatrix} k_{ff}\Delta\phi_{\text{cmd}} + k_p e_\phi \\ f_{ff}\Delta\theta_{\text{cmd}} + k_p e_\theta \\ k_p e_\psi \end{bmatrix} \tag{2.5}$$

These were scaled and sent together with the commanded thrust to the BetaFlight low-level controller at 50 Hz.

The gains were tuned based on a total of 60 short remote outsourced flight tests, lasting 5 to 15 seconds, after which logs would be returned. The test flights were performed at a separate, roughly 60% smaller track with a different density altitude than at the competition locations. This altered the drone dynamics, made flights very short, limited the types of maneuvers, and made it hard to reach full speed. The parameters were then manually fine-tuned during the 1-hour test slot the day before the races.

2.3 Results and analysis

In this section, we show the impact of the various elements of our approach on its performance, for perception, state estimation, and control. These experiments are conducted with a combination of real data collected with the drone, and synthetic data from the hardware-in-the-loop simulation platform provided by the competition organizers. Regarding perception, we assess the accuracy and robustness of the GateNet model qualitatively and quantitatively. Additionally, we provide an overall view of the computational expenses of the perception pipeline. Subsequently, we compare the performance of the developed state estimation scheme with that used in previous competitions. Concerning our control strategy, we conduct a detailed investigation of the various control improvements we introduced. Then, we determine the robustness of our approach to inaccuracies in the internal drone race map. Lastly, we discuss the results of the competition, for the qualification stage, the seasonal races, and the final winning championship race.

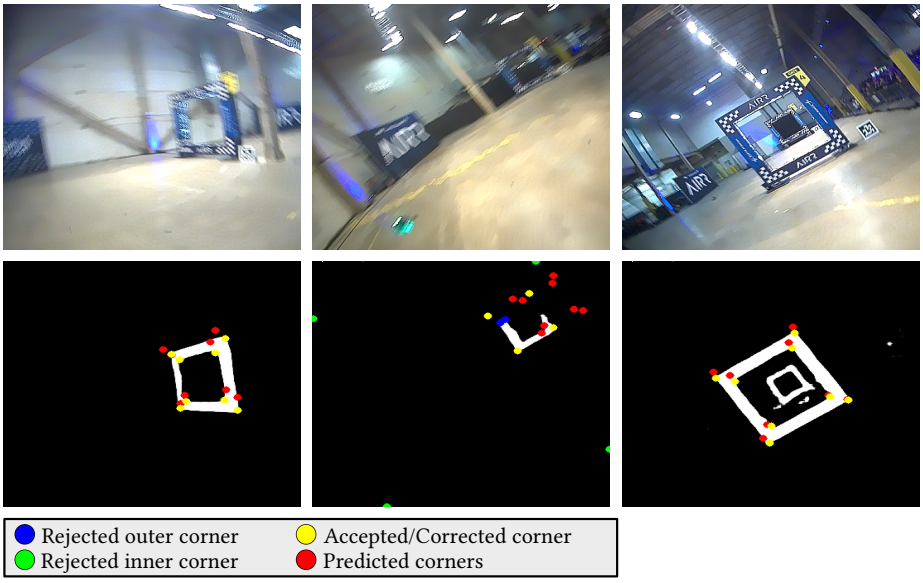


Figure 2.11: Robustness of our GateNet-based perception pipeline and corner refinement to motion blur (*left*), distant gates leading to incomplete segmentation masks (*middle*), and gate overlap, and aggressive bank angles (*right*).

2.3.1 Perception

To quantitatively analyze the GateNet model, we collected and manually annotated a dataset consisting of 165 images logged during our 12-second winning run in the championship race. This dataset is characterized by (i) motion blur on the images due to the high-speed profile achieved in this run, (ii) strong illumination changes, and (iii) a challenging environment with banners containing visual features similar to those of the gates along the course of the track. The reason for only using logged images from this race in this evaluation is that it is the only data that was not used in any training dataset. The GateNet model deployed for the championship race achieves an average intersection-over-union (IoU) of 87%. After the third race, we added an artificial motion blur data augmentation mechanism to the training pipeline which notably improved the resilience towards blur. Note that this artificial blur was not applied to the ground truth masks to still promote sharp segmentations (see Fig. 2.11, left).

The example images and segmentations in Fig. 2.11 show the performance of GateNet in challenging scenarios like motion blur, distant gates, and adverse lighting conditions. They also show the robustness of our corner association algorithm. Our computationally efficient gate corner detector called "snake-gate" identifies the inner and outer corners of the front panel of the gates by actively sampling a small percentage of the pixels of the segmentation result. Then, our state-prediction-based sanity check and refinement of identified corner locations called "gate-prior," compares the sides and angles of the inner and outer contours of both the detected and expected gate in the image plane to neglect distractor gates when multiple gates are in sight (like overhead projected gates from the

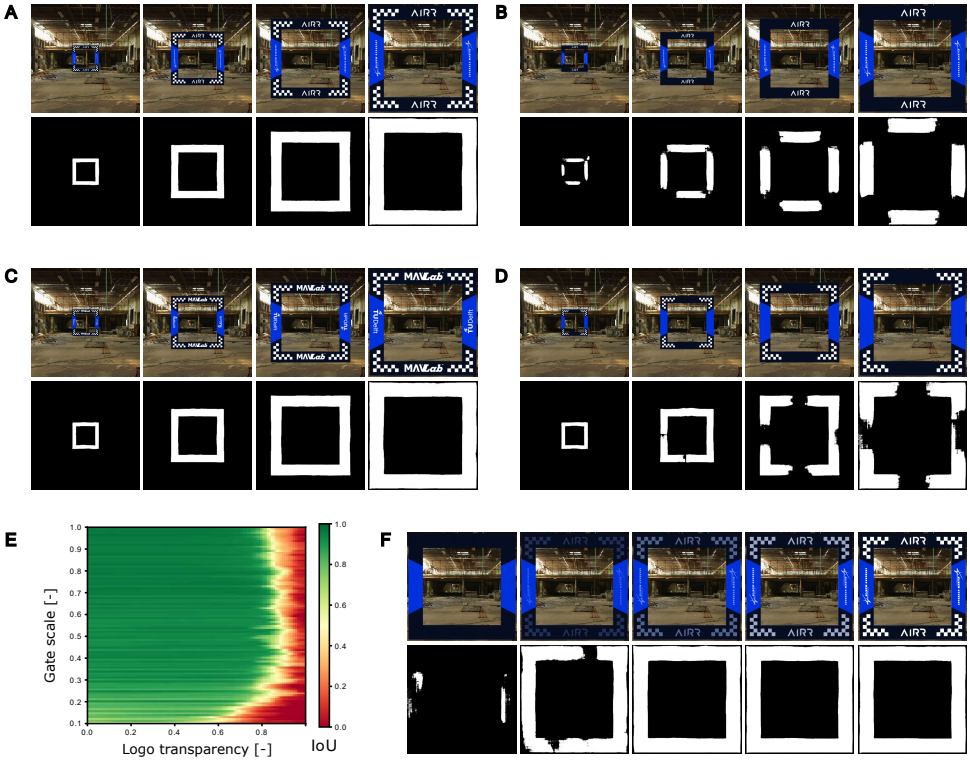


Figure 2.12: Qualitative and quantitative results of the impact of the gate's features on the segmentation accuracy. Different manipulation techniques were employed: (*A*) variation of the scale of the gate in the image space, (*B*) removal of checkerboards, (*C*) substitution of default logos with our own, (*D*) removal of logos, and (*F*) variation of feature transparency. (*E*) summarizes the importance of scale and feature transparency. It presents the IoU as a function of the scale of the gate and the feature transparency for hundreds of data points. This shows that good detections are possible if the logo has a maximum of 80% transparency (20% contrast) for gate sizes down to 30%, but requires more contrast to detect the smaller gates. In all cases, the maximum scale of the gate was set so that the four outer corners of the gate coincide with the extremes of the image space, of size (360×360).

live video stream). It also corrects the location of the estimated corners in case snake-gate didn't identify a corner properly. The resulting corrected corners are finally fed to the PnP-based pose estimation. This allows the drone to localize itself with respect to the next gate even in the case of a discontinuous GateNet mask (see Fig. 2.11, center) or gate overlap in the image space (see Fig. 2.11, right).

Computing the forward pass of GateNet on the GPU models requires on average 13.18 ms and thus can be performed faster than the camera update rate (i.e., 60 Hz). The estimated gate mask from the cropped and downsized input image is de-rotated using the estimated camera roll angle around the optical axis to prevent incorrect corner association and requires an average of 1.32 ms computing time (see Fig. 2.5). The active-vision-based snake-gate method requires accessing the intensity information of only 1.64% of the pixels of a 360×360 mask, which translates to a workload of 0.23 ms per image. We used the full horizontal and vertical histograms of the masks for snake-gate initialization even

though computationally more efficient alternatives exist [125, 126]. Gate-prior takes on average 0.14 ms to correct the identified corners, and lastly, solving the PnP to localize the drone with respect to the gate requires 1.12 ms. Combined with the 1.08 ms import and pre-processing, the perception pipeline takes 17.07 ms while the thread runs at an average of 54 Hz, as it occasionally needs to wait a few milliseconds for delayed images. Note that the full image derotation was only added just before the championship. It was not implemented in the active-vision-based corner detector for risk mitigation purposes, although this could allow the vision pipeline to execute under 16 ms.

Due to the importance of GateNet in the perception pipeline, we also analyzed the impact that each of the features of the AIRR gates has on the segmentation accuracy. We experimented with synthetic data in which we varied the appearance of the gates' features (i.e., checkerboards and text) and assessed the quality of the segmentation both qualitatively and quantitatively. Multiple conclusions can be derived from the results in Fig. 2.12. These results confirm the importance of the contrast changes introduced by the logos and checkerboards, and that removing the logos or checkerboard patterns leads to local gaps in the segmentation. The test with the different logos confirms that GateNet has not learned the specific shapes of the AIRR logos but exploits more generic contrast in this region. The transparency test shows the importance of the presence of contrast. GateNet is quite robust to low contrasts (Fig. 2.12F), but there is a dependency on the scale. Fig. 2.12E shows the IoU for different gate scales and transparencies. At most scales, GateNet's performance only breaks down at ≈80% transparency, whereas at the smallest scale (0.1) it breaks down at ≈60%. Note that, in all cases, the maximum scale of the gate was set so that the four outer corners of the gate coincide with the extremes of the image space, of size (360×360).

2.3.2 State estimation

The vision-based position estimates are fused with model-inertial-based odometry to smoothen the measurements and overcome periods in which no gates are detected. This odometry is primarily based on a linear drag model of the quadrotor in the "flat-body" frame (see Fig. 2.8). The values of the linear drag were fitted with the scarce data from the real flights. Under low flight speed and constant altitude assumptions, this easy-to-identify model was shown to be a reasonable approximation [126]. To improve the predictions during more aggressive maneuvers, instead of fitting a more complex model for which insufficient data was available, we chose to fuse accelerometer data in the odometry (see Eq. 2.2). The difference in performance was compared between the drag-only model called "flat-body", the combined model-inertial "alpha" method (named after its α parameter to set the relative importance of the drag-model versus accelerometer odometry), and traditional body-frame accelerometer-only odometry.

Since no position ground truth is available for the competition flights, the comparison is made with drone observations and track knowledge instead. This can in theory be subject to scaling and offset errors, but as long as the robot perception matches its predictions, they can successfully be merged, just like walking animals merge step-based odometry with visual observations without the need for a calibrated meter representation.

Position measurements close to the gate are very precise thanks to the very good geometry of the PnP triangulation. In other words, small changes in position appear as

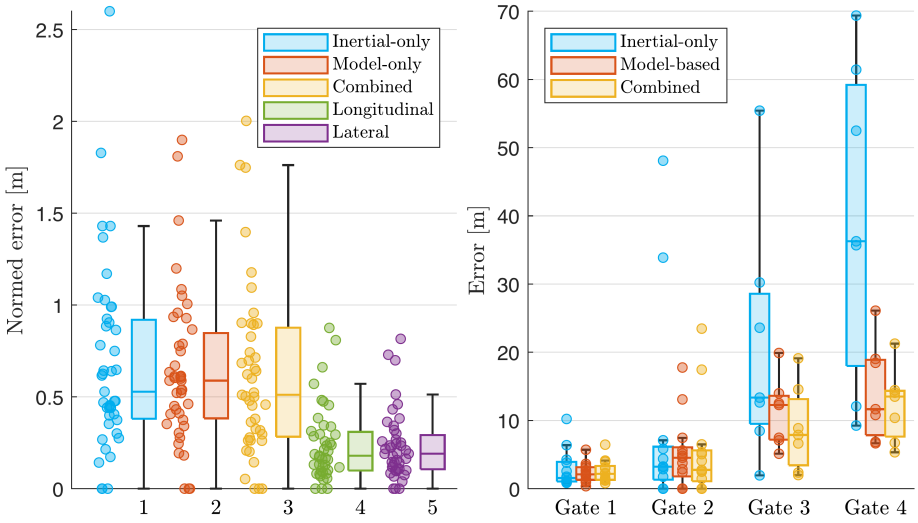


Figure 2.13: Odometry results based on the real-world data from all competition and training runs. *Left*: average of errors in odometry of the three dynamic models (inertial-only, model-only "flat-body" and combined "alpha", and the lateral and longitudinal components of "alpha") after 1.8 s of prediction, where the gate size is indicated as a dashed line. *Right*: statistics of the total accumulated odometry errors from the starting podium to each gate for the 13 full tracks flown competition tracks.

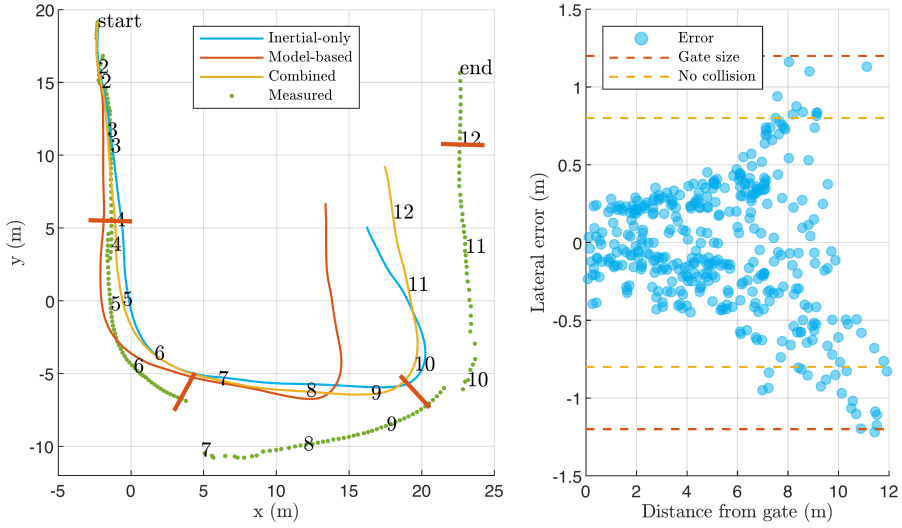


Figure 2.14: Odometry-based position estimate top view for a typical track based on the real-world data from the Baltimore track. Note that the third gate had a different orientation than expected from the flight plan, which causes the vision measurements to appear rotated. The flight times in seconds is indicated, with the total track lasting about 12 seconds.

Setting number	S.1	S.2	S.3	S.4	S.5
Pitch for altitude & coupled saturation	×	✓	✓	✓	✓
Boosted take-off	×	×	✓	✓	✓
Gate centerline	×	×	×	✓	✓
Risk-awareness	×	×	×	×	✓
Robustness	5 / 10	8 / 10	6 / 10	9 / 10	10 / 10
Completion time (s)	13.25±0.12	13.95±0.26	13.71±0.25	13.77± 0.32	12.08±0.27
Avg. speed (m/s)	4.57±0.52	4.55±0.25	4.62±0.23	4.73±0.22	5.38±0.18
Max. speed (m/s)	7.82±0.51	7.51±0.40	7.82±0.44	8.84±1.03	9.69±0.74

Table 2.1: Contribution of the various controller modifications in simulation on the Austin track.

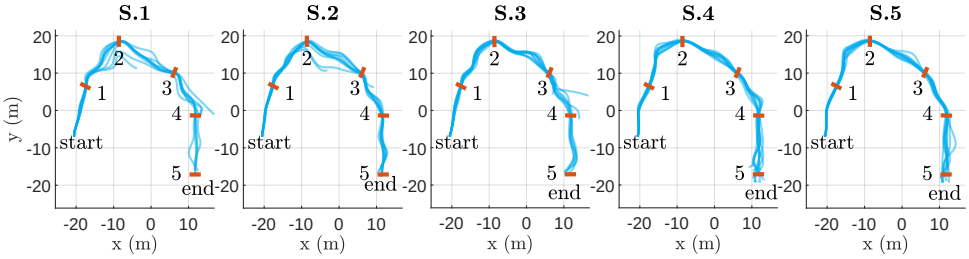


Figure 2.15: Top view of the simulation test results of the different controllers on the Austin track.

large changes in pixel position of gate corners. Moreover, passing the gate is a crucial part of the race and relies on odometry only for the last few meters. We therefore first compared the odometry methods on 1.8-second stretches just before a gate (see Fig. 2.13 for the statistics and Fig. 2.14 for example stretches). Subsequently, we integrated the odometry methods from start to end on 13 full tracks and compared it with the end-gate in the relatively accurate gate map (<1 m displacements). The results can be seen in Fig. 2.13 and a specific track in Fig. 2.14. They show that the model-inertial "alpha" method obtains the best results, which is why we used it in the final championship race. In general, the model-inertial-based odometry can obtain very good results given the scarce resources it requires (50% within <15 m endpoint error without calibration for a 12 s prediction horizon). Nevertheless, it only seems well-suited for tracks that have sufficient gates to perform position corrections.

2.3.3 Control and path planning

To qualitatively validate the contribution to speed and reliability of the different control additions, a simulation study was performed. The initial classical control setup, marked as S.1 and shown in Table 2.1 and Fig. 2.15, only finished the track half the time when configured to fly at competitive speeds. Four modifications were made to increase its speed and reliability.

The first modification to the classical control scheme combines the maximum roll and pitch angles into a single maximum bank called "coupled saturation" since separate maximum pitch and roll angles could yield 42% higher bank angles when saturating together.

Moreover, instead of putting a low and safe maximum bank angle of about 35° to never saturate thrust, we increased the limit past this point to 45°. This would occasionally lead to insufficient thrust, which was addressed by introducing a pitch-for-altitude control loop. As shown in Table 2.1 S.2, the pitch-for-altitude control loop leads to higher robustness, now finishing 8 / 10 runs with only a minor loss in speed.

An open-loop take-off with 100% thrust and a saturating nose-down attitude called "boost" (S.3) was added to initiate the take-off before the slow laser-range altitude sensor had finished booting. Likewise, a saturating pitch down was applied just before the final gate to get an even quicker finish in case the drone sensed it was well aligned. This reduces robustness (6 out of 10 runs finishing the track) but leads to slightly quicker flight times in simulation and much quicker finish times in real races by skipping the up to 1.5 s laser-range startup delay (not present in simulation).

Instead of moving along the shortest path towards the gate, an optimal approach line called "gate-centerline" was defined to prevent sharp approach angles to gates. Too sharp angles not only significantly decrease the safety margin of passing through gates but also affect the position dilution of precision of PnP corners, in turn reducing the quality of state estimates. This addition (S.4) increased the robustness to a success rate of 9 out of 10 through safer gate crossing angles and increased quality of perception.

The final addition to the pipeline is an adaptive, risk-based longitudinal velocity controller (Fig. 2.4, S.5). At large distances from the gate, the drone is allowed to accelerate to a higher speed until it arrives at the optimal gate viewing distance, where it has to make sure the camera sees the gate. Once the drone is sufficiently confident that it is aligned properly with the gate, it can accelerate again. On the other hand, when a gate is not at the expected location or takes longer to identify, or if the control fails to align quickly, the drone slows down. This combination of risk and perception awareness was simple to implement, very light, intuitive to tune, and resulted in robust fast behavior. Table 2.1 S.5 shows that including risk-aware accelerations led to 10 successes out of 10 runs, while substantially increasing the average speed from 4.7 m/s to 5.4 m/s in simulation.

2.3.4 Robustness

Robustness was required to deal with possible camera calibration issues, the random initial starting podiums, the uncertainty about which drone was used for which race, and the inability to measure the track precisely (initially gate locations were even planned to change between runs).

Variations in the track

To evaluate the robustness of our approach to changes in the racing environment, we performed a set of simulation experiments in which we perturb our drone's internal representation of the individual gates and starting podium. Both position and orientation are altered. The drone is thereby forced to react to unanticipated gate locations. This is evaluated in the DRL simulator, which has fixed gate locations, by adding uniform errors to the flight plan.

Fig. 2.16 shows that the state estimates quickly jump to the correct relative location with respect to the internally gate locations (prior waypoints). These red gates represent the (deliberately biased) expected gate locations in the robot's internal map while the

Cases	Original	Gate position error		Gate yaw error	
Flightplan error	0m, 0°	3 m	5 m	20°	40°
Completion time (s)	13.01±0.19	13.02±0.31	12.79±0.81	13.03±0.25	13.60±0.48
Runs completed	10/10	10/10	6/10	9/10	6/10

Table 2.2: Robustness to errors in internal track representation in simulation. Overview of the perturbed flight plans and the effect on lap completion times.

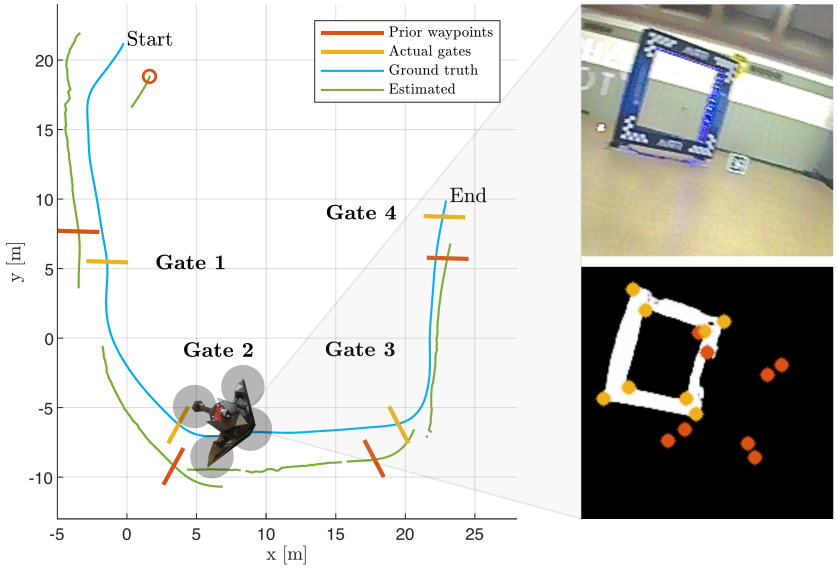


Figure 2.16: Robustness to errors in internal track representation in simulation. The on-board state estimation based on the internal model (red gates) jumps to the correct relative solution (green track) after gathering sufficient evidence.

yellow gates mark the actual locations. The blue line is the simulator's ground-truth of the drone trajectory, while the green line represents the drone's internal state estimates. The advantage of this approach in a race with only a single lap is that the drone does not need to distinguish between its state error and internal map error.

Fig. 2.17 show that our pipeline can finish the course even with 3 m perturbations in the course map or about 25% of the 12 m inter-gate distance, albeit with lateral swings due to the control initially aligning with a wrong gate location and then needing to correct. Fig. 2.17 also shows simulation with flight plan perturbations of 5 m, 20° and 40°. The table in Table 2.2 summarizes the robustness of the approach for various perturbations. The success rate only starts to drop substantially (to 6/10 runs) when gates are displaced 5 m on a track where the inter-gate distance is about 12 m or about 41% of the inter-gate-distance.

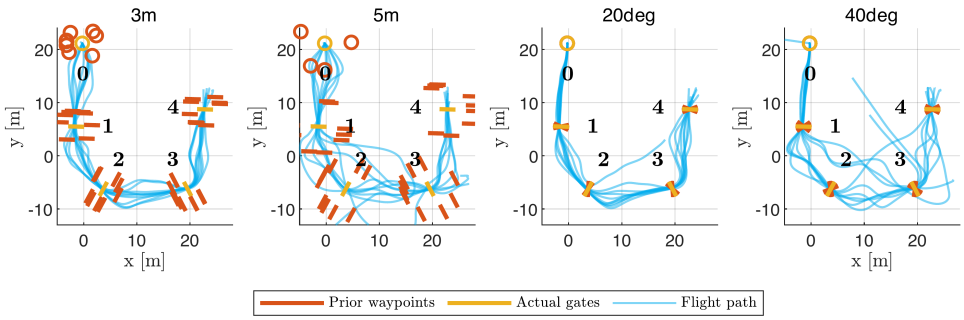


Figure 2.17: Robustness analysis in simulation on the Baltimore track with uniform flight plan gate position errors of 3 m and 5 m, and uniform gate orientation errors of 20° and 40°. Perturbations in trajectories can be seen when the drone aligns with expected position of gates, but quickly has to correct after observing the real positions.

2.3.5 Competition outcome

Fig. 2.18 represents our competition results for the three seasonal races and the championship race and that of the best opponent. Each race consisted of several "heats" which could use a different version of the code. This allowed teams to ensure completion of the track with a steady speed in the initial heats and setting best finish times in the later heats. Fig. 2.19 shows the trajectories flown by our best run during all races. Since there was no ground-truth position measurement system, we show the on-board position estimates.

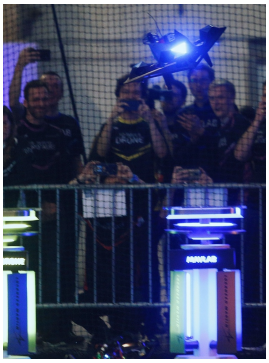
The first race of the season in Orlando was won by team KAIST from South Korea, who were able to fly through two gates of the track. Our drone was not able to pass any gate primarily due to unanticipated enormous differences in illumination between testing and competition. The second race took place in Washington DC and our team was the first to fully finish any track. Since then, our racing speed increased over the seasonal races. The championship race held at Austin was a tight competition as multiple teams were finishing the track during their training and qualification runs. The best performance of the finalists is shown in Fig. 2.18. Our finish time of 12 s with an average velocity of 6.75 m/s and maximum velocity of 9.19 m/s was the prize-winning run (see Fig. 2.20).

After winning the AI vs. AI challenge, our drone was staged against the DRL champion GAB707 in a Human vs. AI challenge. In this race where we were not allowed to change any parameter, our first deployment led to a crash into the first gate, due to a change in starting position of more than 50% of the distance to the gate. Both other heats against the human also started from unanticipated podiums but were within the robustness of the system and finished with the same 12 s lap time.

2.4 Conclusion

We presented our approach to the AIRR competition, which led to winning two out of three seasonal races, the championship race, and the title of "AIRR World Champion 2019". Our approach was human-inspired in the sense that the developed AI focuses on the drone racing gates, which serve as waypoints for the race trajectory, relies significantly on model-based odometry, and accelerates as much as possible when the situation is safe.

1	MAVLab	Finish	12s
2	UZH RPG	Finish	15s
3	USRG @ KAIST	Gate 2	14s
4	Warsaw MIMotaurs	Gate 1	7s
5	KEF Robotics	Gate 1	8s
6	Formula Drone	DNF	
7	Team Puffin	DNF	
8	XQuad	DNF	
9	ICARUS	DNF	



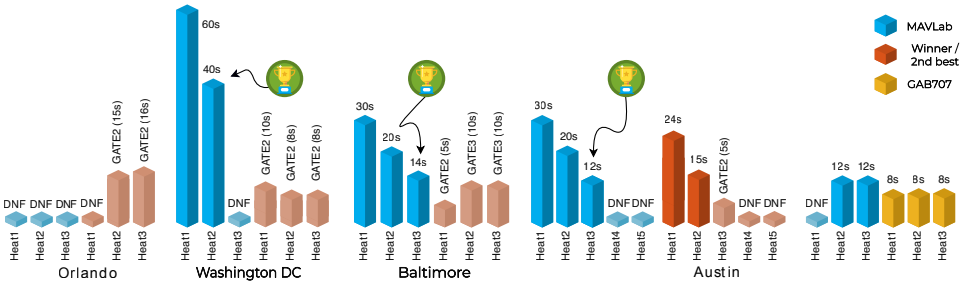


Figure 2.18: Overview of our performance in the 2019 AIRR competition. *Top left*: Leader-board of the championship race, indicating the time it took for each team to reach their farthest waypoint on the track (DNF = "did not finish"). *Top right*: our MAVLab autonomous drone taking off at the championship race, before finishing the 74 m course in 12 s. Picture credit: DRL. *Bottom*: Completion times at the different tracks.

The approach successfully dealt with the scarcity of data and was highly computationally efficient, allowing for a very fast perception and action cycle. By having a deep neural network vision front end, our approach proved to be particularly resilient to frequent changes in the environment. The only occasion that the changes proved to be too much was during the first season race where all training was done in ambient light conditions while the race took place in showbiz illumination conditions that over-exposed the gates.

The constraints of the event drove the current implementation to make several simplifications. First of all the vertical, lateral, and longitudinal dynamics are decoupled in the controller and the estimator. The current model also makes extensive use of the constant altitude properties of the competition. Finally, the navigation solution is tailored towards detecting specific gates. But the light monocular approach opens the road to implementation on board much lighter and hereby faster robots. Finally, merging sparse visual observations with a dynamic model capable of predicting the drone motion for longer distances has shown great results and allowed record in-competition velocities.

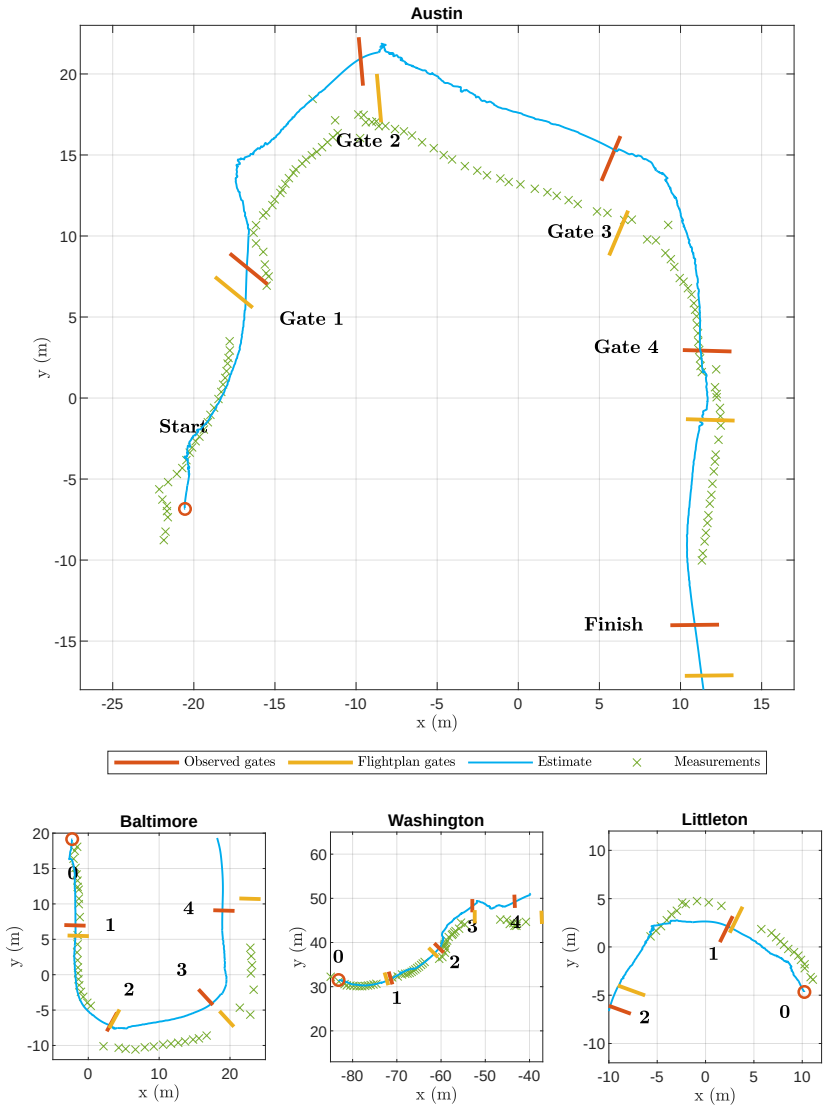


Figure 2.19: Overview of our performance in the 2019 AIRR competition. Top view of the estimated executed path with the rough map of each track received by the organizers.

2.5 Discussion

AI purists may raise the question of how much the competition, and our approach, was actually about AI. In a "pure" AI scenario, the drone's perception and control would have been learned from scratch, making use of the provided simulator. Such an approach would, however, have clashed with the competition setup and timeline. The simulator was ready only a few weeks before the first race and had a substantial reality gap in terms of the

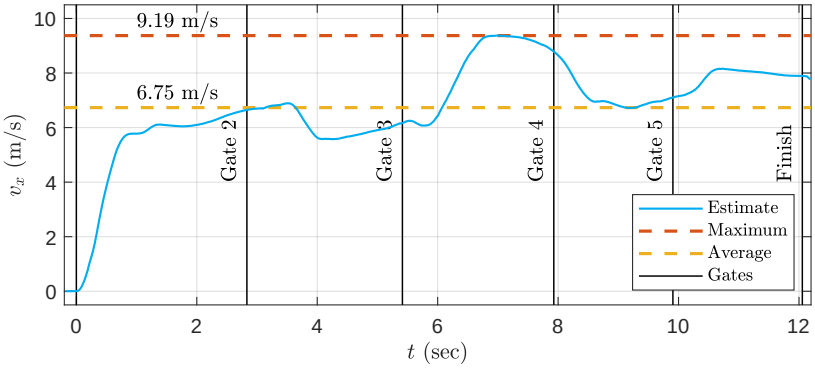


Figure 2.20: Estimated speed profile at the championship race. Our drone flew with an average speed of 6.75 m/s and reached a top speed of 9.19 m/s.

drone's dynamics and image capturing. For example, the images in the simulation had a variable delay, going up to 0.5 s (which was worse than on the real platform). Combined with the extremely scarce access to the drone and outsourced testing, this would have left very little time for end-to-end training and a successful crossing of the reality gap.

Robotics competitions like AIRR reveal highly relevant research areas for AI. In this case: *How can AI best be designed, so that robots need minimal time and data to reach robust and highly agile flight?* A monolithic neural network trained end-to-end purely in simulation likely requires too many training samples to form the best answer to this question. And, if we equate the experience accumulated in a simulator with the evolutionary experience before the birth of an animal, this is not the strategy that we observe in animals either. Animals "even from the same species" are all different physically, and their intelligence is set up in such a way as to deal effectively with these differences. Whereas humans need a long development time before becoming operational, many flying insects can almost immediately fly and perform successful behaviors. The reason for this is that evolution has put in place various mechanisms to deal with, e.g., the physical differences between members of the same species, ranging from adaptation to various learning mechanisms. This means that true AI will require not only reinforcement learning [138], but also, various types of self-supervised learning [139], unsupervised learning [140], and lower-level adaptations as used for instance in adaptive control [135, 141]. This last level of learning, arguably at the lowest level, is hugely important for crossing the reality gap in robotics [114].

2.6 Future directions

To make our approach work in time and robustly enough for the competition, the employed AI still relied quite a lot on us as human system designers. We learned the drone's model based on flight data, used supervised learning with human labeling of images, and designed an active vision algorithm for finding corners in the segmented images. In the future, the generation of large amounts of training data from simulation could reduce this manual

work, but in this event, the qualification round forced teams to label large amounts of images to at least assess the quality of their detector (see Fig. 2.6 first column). Using this data for training resulted in a DNN that segmented so well that it was only complemented with actual flight data.

Please note though that we and others are quickly developing deep learning approaches that can cross the reality gap for performing visual odometry [142], tracking of predetermined optimal trajectories [134, 143] or even for full optimal control [144, 145]. AIRR has already been a driving force to develop AI methods that will successfully bridge the reality gap, even for robots that are difficult to model in detail upfront.

But to beat human pilots in multi-robot races in random, complex, windy environments with multiple gate types, a lot of elements still need further development. Game theory on balancing risks of collisions with the ambition to overtake other drones, detection of randomly shaped gates after having been shown their appearance only minutes before the race, and adapting to competitor tactics are just a few of the many additional challenges that future robotics research will need to face in the competition against human pilots.

Ultimately, reducing the computational load while increasing the speed of algorithms, or in other words *improving the computational efficiency*, will play a deciding role in determining how fast and maneuverable flying robots can become, as power and especially weight spent in computing adversely influence performance. Facing these robotic challenges will bring the technology closer to applications for the benefit of the real world. We expect the applications of very fast, agile, and situation-aware flying robots to range from ambulance drones or package delivery drones swiftly planning around obstacles in cluttered environments, to search and rescue drones. But most of all, autonomous racing helps develop solutions that will, sooner or later, improve the characteristics of all flying robots. If we succeed to do this in heavily resource-constrained and time-pressed racing drones, then it will also generalize to other types of robots and tasks, such as autonomous vacuum cleaners or self-driving cars.

Supplementary material

Video summary of the approach: https://youtu.be/yN5QVl07F2Q

Videos of all the best competition entries: https://youtu.be/ihfUckB16wU

Logfiles of the 2019 AIRR winning entry: https://tinyurl.com/56pvbzmp

3

Optical-Flow-Aided, Self-Supervised Frame Reconstruction From Events

After exploring the design of a vision-based navigation solution for flying robots using conventional sensing and processing in the previous chapter, we now delve into the realm of neuromorphic sensing. Event-based cameras are novel vision sensors that sample, in an asynchronous fashion, brightness increments with low latency and high temporal resolution. The resulting streams of events are of high value by themselves, especially for high speed motion estimation. However, there has been a growing interest in reconstructing intensity frames from these events, as this allows bridging the gap with existing literature on appearance- and frame-based computer vision. Recent work has mostly approached this problem using neural networks trained with synthetic, ground-truth data. In this chapter we approach, for the first time, the intensity reconstruction problem from a self-supervised learning perspective. Our method, which leverages the knowledge of the inner workings of event-based cameras, combines estimated optical flow and the event generative model to train neural networks without the need for any ground-truth or synthetic data. Results across multiple datasets show that the performance of the proposed self-supervised approach is in line with the state-of-the-art at the time of publication. Additionally, we propose a novel, lightweight neural network for optical flow estimation that achieves high speed inference with only a minor drop in performance.

The contents of this chapter have been published in:

F. Paredes-Vallés, G. C. H. E. de Croon, *Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021.

3.1 Introduction

Unlike conventional cameras recording intensity frames at fixed time intervals, event cameras sample light based on scene dynamics by asynchronously measuring per-pixel brightness[1] changes at the time they occur [18]. This results in streams of sparse events encoding the polarity of the perceived changes. Because of this paradigm shift, event cameras offer several advantages over their frame-based counterparts, namely low power consumption, high dynamic range (HDR), low latency and high temporal resolution.

Despite the advantages, the novel output format of event cameras poses new challenges in terms of algorithm design. Unless working with spiking neural networks [101, 102], events are usually converted into intermediate representations that facilitate the extraction of information [18, 29]. Among others, intensity frames are an example of a powerful representation since they allow the evaluation of the appearance of a visual scene, thus bridging the gap between event cameras and the existing frame-based computer vision literature [146, 147]. For this reason, there has been a significant research drive to develop new methods to reconstruct images from events with similar statistics to those captured by standard cameras.

Recent work has mostly approached this problem from a machine learning perspective. With their E2VID artificial neural network, Rebecq *et al.* [146, 147] were the first to show that learning-based methods trained to maximize perceptual similarity via supervised learning outperform hand-crafted techniques by a large margin in terms of image quality. Later, Scheerlinck *et al.* [148] achieved high speed inference with FireNet, a simplified model of E2VID. Despite the high levels of accuracy reported, these architectures were trained with large sets of synthetic data from event camera simulators [149], which adds extra complexity to the reconstruction problem due to the simulator-to-reality gap. In fact, Stoffregen *et al.* [150] recently showed that if the statistics of the synthetic training datasets do not closely resemble those seen during inference, image quality degrades and the generalizability of these architectures remains limited.

In this work, we propose to come back to the theoretical basics of event cameras to relax the dependency of learning-based reconstruction methods on ground-truth and synthetic data. Specifically, we introduce the self-supervised learning (SSL) framework in Fig. 3.1, which consists of two artificial neural networks, *FlowNet* and *ReconNet*, for optical flow estimation and image reconstruction, respectively. FlowNet is trained through the contrast maximization proxy loss from Zhu *et al.* [33], while ReconNet makes use of the flow-intensity relation in the *event-based photometric constancy* [151] to reconstruct the frames that best satisfy the input events and the estimated optical flow. Using our method, we retrain several networks from the image reconstruction [146, 148] and optical flow [25] literature. In terms of accuracy, results show that the reconstructed images are in line with those generated by most learning-based approaches despite the lack of ground-truth data during training. Additionally, we propose *FireFlowNet*, a lightweight architecture for optical flow estimation that, inspired by [148], achieves high speed inference with only a minor drop in performance.

In summary, this chapter contains two main contributions. First, a novel SSL framework to train artificial neural networks to perform event-based image reconstruction that, with

[1] Defined as the logarithm of the pixel intensity, i.e., $L \doteq \log(I)$.

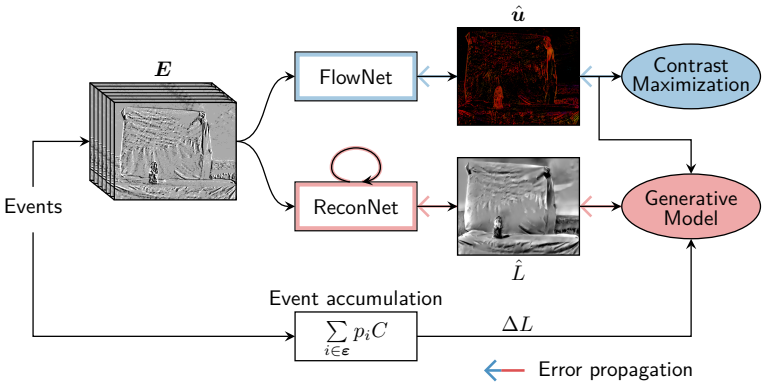


Figure 3.1: Overview of the proposed framework. Our model is trained in a self-supervised fashion to perform optical flow estimation and image reconstruction from event data using contrast maximization and the event-based photometric constancy, respectively. Colored reverse arrows indicate error propagation for each loss.

the aid of optical flow, does not require ground truth of any kind and can learn directly on real event data. Second, we introduce FireFlowNet: a novel, lightweight neural network architecture that performs fast optical flow estimation from events. We validate our self-supervised method and optical flow network through extensive quantitative and qualitative evaluations on multiple datasets.

3.2 Related work

Early methods to image reconstruction from event data approached the problem through the photometric constancy: each event provides one equation relating the intensity gradient and the optical flow [151]. Kim *et al.* [152] were the first in the field and developed an extended Kalman filter that, under rotational and static scene assumptions, reconstructs a gradient image that is later transformed into the intensity space via Poisson integration. They later extended this approach to 6 degrees-of-freedom camera motion [61]. Under the same assumptions, Cook *et al.* [153] simultaneously recovered intensity images, optical flow, and angular velocity through bio-inspired, interconnected network of interacting maps. Bardow *et al.* [154] developed a variational energy minimization framework to simultaneously estimate optical flow and intensity from sliding windows of events, relaxing for the first time the static scene assumption.

Instead of relying on the photometric constancy, several approaches based on direct event integration have been proposed, which do not assume scene structure or motion dynamics. Reinbacher *et al.* [155] formulated intensity reconstruction as an energy minimization problem via direct integration with periodic manifold regularization. Scheerlinck *et al.* [156] achieved computationally efficient reconstruction by filtering events with a high-pass filter prior to integration.

Several machine learning approaches have also been proposed. Training generative adversarial networks with real grayscale frames was proposed by Wang *et al.* [157] and Pini *et al.* [158]. However, Rebecq *et al.* [146, 147] showed that training in a supervised

fashion with a large synthetic dataset allowed for higher quality reconstructions with their E2VID architecture. Focused on computational efficiency, Scheerlinck *et al.* [148] managed to significantly reduce E2VID complexity with FireNet, with only a minor drop in accuracy. Inspired by these works, Choi *et al.* [159] and Wang *et al.* [160] recently proposed hybrid approaches that incorporate super resolution aspects in the training process and architecture design to improve image quality. Lastly, Stoffregen *et al.* [150] recently highlighted that, when training with ground truth, the statistics of the training dataset play a major role in the reconstruction quality. They showed that a slight change in the training statistics of E2VID leads to significant improvements across multiple datasets.

Our proposed SSL framework (see Fig. 3.1) is based on the event-based photometric constancy used by early reconstruction methods. Similarly to Bardow *et al.* [154], we simultaneously estimate intensity and optical flow from the input events. However, instead of relying on a joint optimization scheme, we achieve it via two independent neural networks that only share information during training. Further, we reconstruct intensity directly from the photometric constancy, instead of from an oversimplified model of the event camera. This approach allows, for the first time, to relax the strong dependency of learning-based approaches on ground-truth and synthetic data.

3.3 Method

An event camera consist of an array of independent pixels that respond to changes in the brightness signal $L(t)$, and transmit these changes through streams of sparse and asynchronous events [12]. For an ideal camera, an event $\boldsymbol{e}_i = (\boldsymbol{x}_i, t_i, p_i)$ is triggered at pixel $\boldsymbol{x}_i = (x_i, y_i)^T$ and time t_i whenever the brightness change since the last event at that pixel reaches a contrast sensitivity threshold C. Therefore, the brightness increment occurred in a time window Δt_k is encoded in the event data via pixel-wise accumulation:

$$\Delta L_k(\boldsymbol{x}) = \sum_{\boldsymbol{e}_i \in \Delta t_k} p_i C \tag{3.1}$$

where $C > 0$, and the polarity $p_i \in \{+, -\}$ encodes the sign of the brightness change.

As in [151], under the assumptions of Lambertian surfaces, constant illumination and small Δt, we can linearize Eq. 3.1 to obtain the event-based photometric constancy:

$$\Delta L_k(\boldsymbol{x}) \approx -\nabla L_{k-1}(\boldsymbol{x}) \cdot \boldsymbol{u}_k(\boldsymbol{x}) \Delta t_k \tag{3.2}$$

which encodes that events are caused by the spatial gradients of the brightness signal, $\nabla L = (\delta_x L, \delta_y L)^T$, moving with optical flow $\boldsymbol{u} = (u, v)^T$. The dot product conveys that no events are generated if the flow vector is parallel to an edge ($\boldsymbol{u} \perp \nabla L$), while they are generated at the highest rate if perpendicular ($\boldsymbol{u} \parallel \nabla L$). Thus, events are caused by the projection of the optical flow vector in the ∇L direction, the so-called normal optical flow.

3.3.1 Overview

Our goal is to learn, in an SSL fashion, to transform a continuous stream of events into a sequence of intensity images $\{\hat{I}_k\}$. To achieve this, we propose the pipeline in Fig. 3.1 in which two neural networks are jointly trained. On the one hand, FlowNet is a convolutional network that learns to estimate optical flow by compensating for the motion blur in the

input events. On the other hand, ReconNet is a recurrent convolutional network that learns to perform image reconstruction through the event-based photometric constancy.

3.3.2 Input event representation

As proposed in [33], the input to both our networks is a voxel grid E_k with B temporal bins that gets populated with consecutive, non-overlapping partitions of the event stream $\boldsymbol{\varepsilon}_k \doteq \{\boldsymbol{e}_i\}_{i=0}^{N-1}$, each containing a fixed number of events, N. For each partition, every event (with index i) distributes its polarity p_i to the two closest bins according to:

$$E(\boldsymbol{x}_i, t_b) = \sum_i p_i \kappa(t_b - t_i^*(B-1)) \tag{3.3}$$

$$\kappa(a) = \max(0, 1 - |a|) \tag{3.4}$$

$$t_i^* = \frac{\left(t_i - t_0^k\right)}{\left(t_{N-1}^k - t_0^k\right)} \tag{3.5}$$

where b is the bin index, and $t_i^* \in [0,1]$ denotes the normalized event timestamp. This representation adaptively normalizes the temporal dimension of the input depending on the timestamps of each partition of events.

3.3.3 Learning optical flow via contrast maximization

We aim to learn to reconstruct L through the photometric constancy in Eq. 3.2, which, besides the spatial and temporal derivatives of the brightness itself, also depends on the optical flow $\boldsymbol{u}$. One could use ground-truth optical flow to solve for this ill-posed problem. However, due to the limited availability of event-camera datasets with accurate, high-frequency ground-truth data, we opt for training our FlowNet to perform flow estimation in a self-supervised manner, using the contrast maximization proxy loss for motion compensation [96].

A partition of events is said to be blurry whenever there is a spatiotemporal misalignment among its events, i.e., events generated by the same portion of a moving edge are captured with different timestamps and pixel locations. The idea behind the motion compensation framework [96] is that accurate optical flow can be retrieved by finding the motion model of each event that best deblurs $\boldsymbol{\varepsilon}_k$. Knowing the per-pixel optical flow, the events can be propagated to a reference time t_{ref} through:

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + (t_{\text{ref}} - t_i)\boldsymbol{u}(\boldsymbol{x}_i) \tag{3.6}$$

In this work, we adopt the deblurring quality measure proposed by Mitrokhin *et al.* [161] and later refined by Zhu *et al.* [33]: the per-pixel and per-polarity average timestamp of the resulting image of warped events (IWE), H. The lower this metric, the better the deblurring. As in [33], we generate an image of the average (normalized) timestamp at each pixel for each polarity p' via bilinear interpolation:

$$T_{p'}(\boldsymbol{x};\boldsymbol{u}|t_{\text{ref}}^*) = \frac{\sum_j \kappa(x - x_j')\kappa(y - y_j')t_j^*}{\sum_j \kappa(x - x_j')\kappa(y - y_j') + \epsilon}$$
$$j = \{i \mid p_i = p'\}, \qquad p' \in \{+,-\}, \qquad \epsilon \approx 0 \tag{3.7}$$

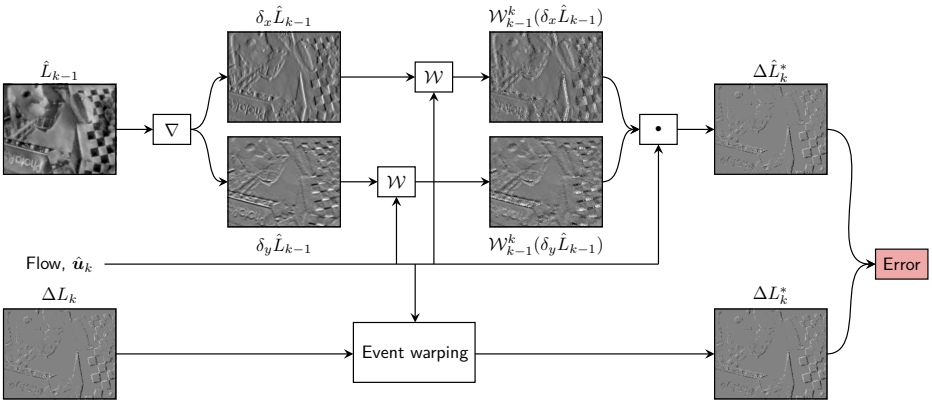


Figure 3.2: Brightness reconstruction via the event-based photometric constancy formulation proposed in this work. The most recent event-based optical flow estimate from FlowNet $\hat{\boldsymbol{u}}_k$ is used to (i) warp the input events, (ii) warp the spatial gradients of the last reconstructed image $\hat{L}_{k-1}$, and (iii) in the dot product with the warped gradients. The predicted brightness increment image $\Delta\hat{L}^*_k$ is compared to that obtained with the deblurred input events, ΔL^*_k, and the error is propagated backwards towards ReconNet to improve reconstruction accuracy.

and minimize the sum of the squared images resulting from warping the events forward and backward to prevent scaling issues during backpropagation:

$$\mathcal{L}_{\text{contrast}}(t^*_{\text{ref}}) = \sum_{\boldsymbol{x}} T_+(\boldsymbol{x};\boldsymbol{u}|t^*_{\text{ref}})^2 + T_-(\boldsymbol{x};\boldsymbol{u}|t^*_{\text{ref}})^2 \tag{3.8}$$

$$\mathcal{L}_{\text{contrast}} = \mathcal{L}_{\text{contrast}}(1) + \mathcal{L}_{\text{contrast}}(0) \tag{3.9}$$

The total loss used to train FlowNet is then given by:

$$\mathcal{L}_{\text{FlowNet}} = \mathcal{L}_{\text{contrast}} + \lambda_1 \mathcal{L}_{\text{smooth}} \tag{3.10}$$

where $\mathcal{L}_{\text{smooth}}$ is a Charbonnier smoothness prior [162], and λ_1 is a scalar balancing the effect of the two losses. Note that, since $\mathcal{L}_{\text{contrast}}$ does no propagate the error back to pixels without events, we mask FlowNet's output so that null optical flow vectors are returned at these pixel locations.

3.3.4 Learning reconstruction via photometric constancy

We formulate the SSL reconstruction problem from an image registration perspective [163] via brightness increment images. Specifically, we propose to use the difference between the reference increment image ΔL (event integration, Eq. 3.1) and the predicted $\Delta\hat{L}$ (photometric constancy, Eq. 3.2) to reconstruct the brightness signal that best explains the input events, assuming known, error-free optical flow. This reconstructed brightness is denoted by $\hat{L}$. FlowNet predictions are used in the computation of $\Delta\hat{L}$, and as registration parameters to warp both increment images to a common temporal frame (indicated by the superscript *). A schematic of the proposed formulation is shown in Fig. 3.2.

To minimize motion blur in the reconstructed frames, instead of directly integrating the input events, we define the reference brightness increment ΔL^* via the per-pixel and per-polarity average number of warped events:

$$\Delta L^*(\boldsymbol{x};\boldsymbol{u}) \doteq C\,(G_+(\boldsymbol{x};\boldsymbol{u}|1) - G_-(\boldsymbol{x};\boldsymbol{u}|1)) \tag{3.11}$$

$$G_{p'}(\boldsymbol{x};\boldsymbol{u}|t^*_{\text{ref}}) = \frac{H_{p'}(\boldsymbol{x};\boldsymbol{u}|t^*_{\text{ref}})}{P_{p'}(\boldsymbol{x};\boldsymbol{u}|t^*_{\text{ref}}) + \epsilon} \tag{3.12}$$

where P is a two-channel image containing the number of pixel locations from where the IWE H receives events in the event warping process. Therefore, ΔL^* is a deblurred representation of the contrast change encoded in the input events. An ablation study on the impact of event deblurring prior to event integration can be found in Section 3.4.3.

On the other hand, we adapt the event-based photometric constancy in Eq. 3.2 and compute $\Delta\hat{L}$ by warping the spatial gradients of the last reconstructed image to the current time instance via spatial transformers [164]:

$$\Delta\hat{L}^*(\boldsymbol{x};\boldsymbol{u}) \doteq -\mathcal{W}^k_{k-1}(\nabla\hat{L}_{k-1}(\boldsymbol{x}))\cdot\hat{\boldsymbol{u}}_k(\boldsymbol{x}) \tag{3.13}$$

where $\mathcal{W}^k_{k-1}$ is the warping function of the optical flow $\hat{\boldsymbol{u}}_k$.

Following a maximum likelihood approach [12, 60], we define the photometric reconstruction loss as the squared L_2 norm of the difference of the warped brightness increments:

$$\mathcal{L}_{\text{PE}} = \left\|\Delta L^*(\boldsymbol{x};\boldsymbol{u}) - \Delta\hat{L}^*(\boldsymbol{x};\boldsymbol{u})\right\|^2_2 \tag{3.14}$$

where, besides $\hat{L}$, the contrast threshold C is the only remaining unknown. To relax the dependency on this parameter, our ReconNet uses linear activation in its last layer instead of the frequently used sigmoid function [147, 148]. The resulting unbounded brightness estimate is first transformed into the intensity space through $\hat{I}_k = \exp(\hat{L}_k)$, and then normalized to get the final reconstruction $\hat{I}^f_k$:

$$\hat{I}^f_k = \frac{\hat{I}_k - m}{M - m} \tag{3.15}$$

where m and M are the 1% and 99% percentiles of $\hat{I}_k$, and $\hat{I}^f_k$ is clipped to $[0,1]$. This normalization allows the use of any value of C for training as long as the ratio of positive and negative thresholds resembles that of the evaluation sequences. We assume that most event-camera datasets were recorded with $C_+/C_- \approx 1$, and set both thresholds to 1.

On its own, Eq. 3.14 is not sufficient for the reconstruction of temporally consistent images. Because of the dot product in Eq. 3.13, the absence of input events can be ambiguously understood as lack of apparent motion, lack of spatial image gradients, or both. To solve for this issue, we introduce an explicit temporal consistency loss based on the frame-based formulation of the photometric constancy [165]. In essence, we define the temporal loss as the photometric error between two successive reconstructed frames:

$$\mathcal{L}_{\text{TC}} = \left\|\hat{L}_k - \mathcal{W}^k_{k-1}(\hat{L}_{k-1})\right\|_1 \tag{3.16}$$

The total loss used to train ReconNet is then given by:

$$\mathcal{L}_{\text{ReconNet}} = \sum_{k=0}^{S}\mathcal{L}_{\text{PE}} + \lambda_2\sum_{k=S_0}^{S}\mathcal{L}_{\text{TC}} + \lambda_3\sum_{k=0}^{S}\mathcal{L}_{\text{TV}} \tag{3.17}$$

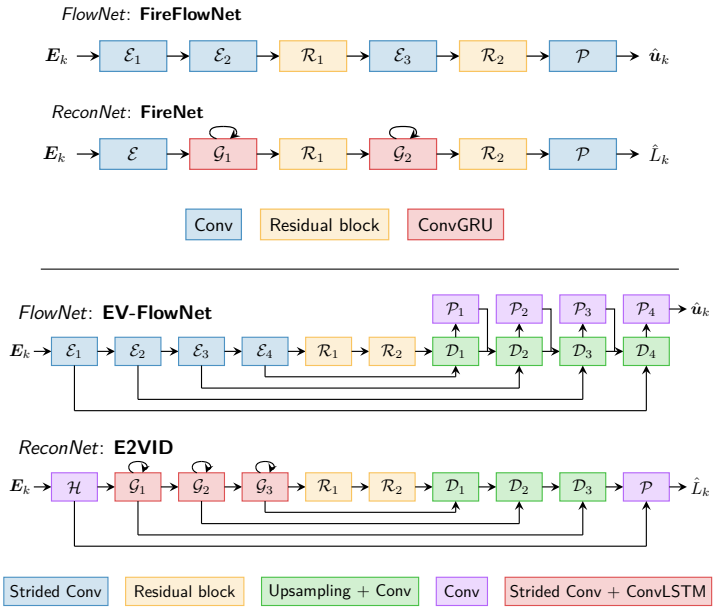


Figure 3.3: Neural networks evaluated in this work.

where S denotes the number of steps we unroll the recurrent network for during training, $\mathcal{L}_{\text{TV}}$ is a smoothness total-variation constraint [166], and λ_2 and λ_3 are scalars balancing the effect of the three losses.

3.3.5 Network architectures

We evaluate the two trends on network design for event cameras when trained with our SSL framework. The evaluated architectures are shown in Fig. 3.3.

	EV-FlowNet [25]	FireFlowNet (Ours)
No. params.	14130.28 k	57.03 k
Memory	53.90 MB	0.22 MB
Downsampling	Yes	No

Table 3.1: Main architectural differences between our FireFlowNet and EV-FlowNet [25]. FireFlowNet has 250× fewer parameters, consuming only 0.41% of the memory.

FlowNet: FireFlowNet. FireFlowNet is our proposed lightweight architecture for fast optical flow estimation. Inspired by FireNet [148], the network consists of three encoder layers that perform single-strided convolutions, two residual blocks [167], and a final prediction layer that performs depthwise (i.e., 1×1) convolutions with two output channels. All layers have 32 output channels and use 3×3 kernels and ReLU activations except for the final, which uses TanH activations. A comparison of the key architectural differences between our FireFlowNet and the current state-of-the-art is shown in Table 3.1.

FlowNet: EV-FlowNet [25]. The input voxel grid $\boldsymbol{E}_k$ is passed through four strided convolutional layers with output channels doubling after each layer starting from 64. The resulting activations are then passed through two residual blocks [167] and four decoder

	outdoor_day1		indoor_flying1		indoor_flying2		indoor_flying3	
	EPE↓	$\%_{\text{Outlier}}$↓	EPE↓	$\%_{\text{Outlier}}$↓	EPE↓	$\%_{\text{Outlier}}$↓	EPE↓	$\%_{\text{Outlier}}$↓
EV-FlowNet$_{\text{GT-SIM}}$ [150]	0.68	1.0	**0.56**	1.0	**0.66**	**1.0**	**0.59**	**1.0**
EV-FlowNet$_{\text{FW-MVSEC}}$ [25]	0.49	0.2	1.03	2.2	1.72	15.1	1.53	11.9
EV-FlowNet$_{\text{EW-MVSEC}}$ [33]	**0.32**	**0.0**	0.58	**0.0**	1.02	4.0	0.87	3.0
EV-FlowNet$_{\text{EW-DR}}$ (Ours)	0.92	5.4	0.79	1.2	1.40	10.9	1.18	7.4
FireFlowNet$_{\text{EW-DR}}$ (Ours)	1.06	6.6	0.97	2.6	1.67	15.3	1.43	11.0

Table 3.2: Quantitative evaluation of our FlowNet architectures on the MVSEC dataset [168]. Best in bold, runner up underlined.

layers that perform bilinear upsampling followed by convolution. After each decoder, there is a (concatenated) skip connection from the corresponding encoder, as well as another depthwise convolution to produce a lower scale flow estimate, which is then concatenated with the activations of the previous decoder. The $\mathcal{L}_{\text{FlowNet}}$ loss (see Eq. 3.10) is applied to each intermediate flow estimate via flow upsampling. All layers use 3×3 convolutional kernels and ReLU activations except for the flow prediction layers, which use TanH activations.

ReconNet: FireNet [148]. Same architecture as FireFlowNet except for the second and third encoder, which are recurrent ConvGRU layers [169]. As in [148], each layer has 16 output channels, but we use linear activation in the final layer.

ReconNet: E2VID [147]. The input voxel grid $\boldsymbol{E}_k$ is passed through a convolutional head layer, three recurrent encoders performing strided convolution followed by ConvLSTM [170], two residual blocks [167], three decoder layers that perform bilinear upsampling followed by convolution, and a final depthwise convolutional prediction layer. There are (element-wise sum) skip connections between symmetric encoder and decoder layers, and the number of output channels in the head layer is 32 and doubles after each encoder. Head, encoder, and decoder layers use 5×5 kernels, while the rest uses 3×3. All layers use ReLU activations except for the final prediction layer which uses linear.

3.4 Experiments

We train our networks on the indoor forward facing sequences from the UZH-FPV Drone Racing Dataset (DR) [171], which is characterized by a much wider distribution of optical flow vectors than other datasets, such as MVSEC [168], the Event-Camera Dataset (ECD)[2] [172], or the High Quality Frames (HQF) dataset [150]. Our training sequences consist of approximately 15 minutes of event data recorded with a racing quadrotor flying aggressive six-degree-of-freedom trajectories. We split these recordings and generate 440 128×128 (randomly cropped) sequences of 2 seconds each, and use them for training with $B = 5$. We further augment this data using random horizontal, vertical and polarity flips, besides with artificial pauses of the input event stream (i.e., forward-pass with null input voxel). For training, we fixed the number of input events per pixel to 0.3.

[2]The DAVIS frames accompanying the ECD dataset [172] usually suffer from motion blur and under/overexposure. For this reason, we only evaluate reconstruction accuracy on sections of this dataset in which the frames appear to be of high quality. The exact cut times are adopted from [150]. Additionally, we only evaluate optical flow accuracy on these sections to remain comparable to the results reported in [150].

	GPU (ms)		FLOPs (G)	
	EV-FlowNet	FireFlowNet	EV-FlowNet	FireFlowNet
240 × 180	4.33	**1.97**	8.91	**2.47**
346 × 260	7.05	**3.81**	18.60	**5.14**
640 × 480	17.04	**12.55**	61.47	**17.59**
1280 × 720	49.32	**34.24**	184.41	**52.67**

Table 3.3: Computational cost evaluation of our FireFlowNet against EV-FlowNet [25]. We report inference time on GPU and the floating point operations (FLOPs) per forward-pass at common sensor resolutions. We used a single NVIDIA GeForce GTX 1080 Ti GPU for all experiments.

Our framework is implemented in PyTorch. We use the Adam optimizer [173] and a learning rate of 0.0001 for both networks, and train with a batch size of 1 for 120 epochs. We empirically set the weights for each loss to $\{\lambda_1, \lambda_2, \lambda_3\} = \{1.0, 0.1, 0.05\}$, ReconNet's unrolling S to 20 steps, and S_0 to 10 steps.

3.4.1 Optical flow evaluation

To validate FireFlowNet as a lightweight alternative to the current state-of-the-art in event-based optical flow estimation, we evaluated both of our FlowNet architectures on the indoor_flying and outdoor_day sequences from the MVSEC dataset [168] with the ground-truth data provided by Zhu *et al.* [25]. Optical flow predictions were generated at each grayscale frame timestamp (i.e., at 45 Hz), and scaled to be the displacement between two successive frames. This evaluation setting is usually referred to as dt = 1 in the literature [25, 33, 34, 102, 150]. The predicted optical flow is converted from units of pixels/grid to units of pixel displacement by multiplying it with $\mathrm{dt}_{\mathrm{gt}}/\mathrm{dt}_{\mathrm{grid}}$.

Quantitative results are presented in Table 3.2 and qualitative results in Fig. 3.4. We use the average endpoint error (EPE↓, lower is better) and the percentage of points with EPE greater than 3 pixels to compare our FlowNet architectures against three EV-FlowNet from literature; two of them trained with frame- (FW) [25] and event-warping (EW) [33] SSL proxy losses on MVSEC [168], and one trained with synthetic ground-truth data (GT) [150]. For our networks, the number of input events per pixel was set to 0.3. Error metrics were only acquired over pixels with valid ground-truth data and at least one event; and, for comparison, we used the quantitative results reported in [33, 150].

From Table 3.2, the first noticeable aspect is the accuracy gap between EV-FlowNet$_{\text{GT-SIM}}$ and the rest of networks. Training with ground-truth dense optical flow entails certain ability to resolve the aperture problem [105] that most SSL approaches lack. Regarding the latter, our EV-FlowNet performs consistently better than EV-FlowNet$_{\text{FW-MVSEC}}$ in all sequences except for outdoor_day1, but underperforms EV-FlowNet$_{\text{EW-MVSEC}}$ despite using the same architecture and training procedure. We believe this is mostly due to the different training datasets and the fact that we did not fine-tune the number of input events for this evaluation. Further, note that these literature architectures were trained on a very similar driving sequence from MVSEC, while our training data is much more diverse in terms of optical flow vectors [171].

Using our EV-FlowNet as reference, Table 3.2 shows that the proposed FireFlowNet is characterized by a comparable accuracy despite the significant reduction in model complex-

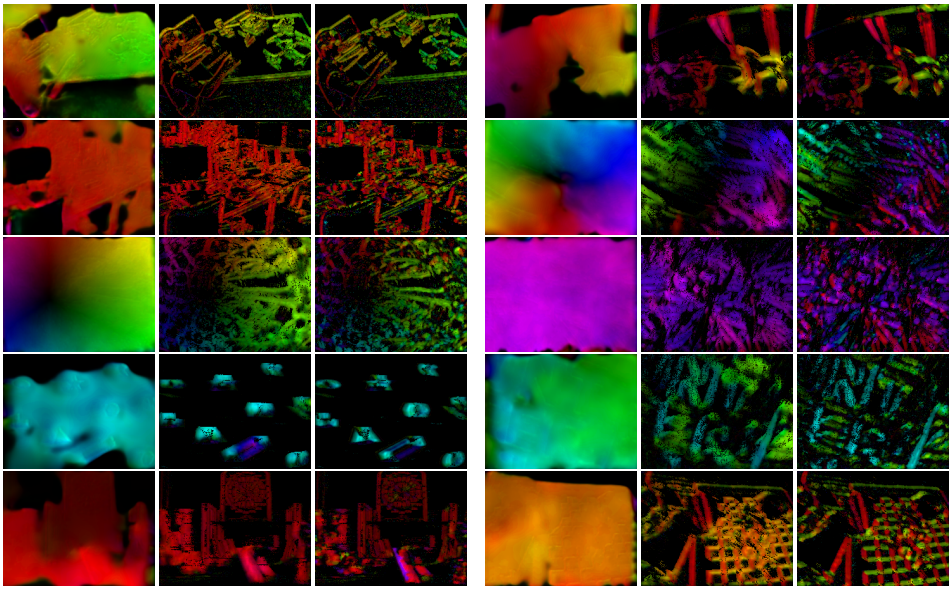


Figure 3.4: Qualitative comparison of our FlowNet architectures with the state-of-the-art EV-FlowNet [150] on sequences from the ECD [172] (*left*) and HQF [150] dataset (*right*). The optical flow color-coding scheme can be found in Fig. 3.9.

ity. This performance drop is likely due to the narrow receptive field of the architecture, which entails limitations due to the aperture problem. Regarding computational cost, Table 3.3 shows that FireFlowNet runs ~1.3-2.2 times faster than EV-FlowNet on GPU, requiring less than ~30% of FLOPS per forward-pass.

For completeness, we also evaluate our FlowNet architectures on the ECD [172] and HQF [150] datasets via the Flow Warp Loss (FWL↑, higher is better) [150]. This metric measures the sharpness of the IWE in relation to that of the original partition of events. Similarly to [150], we set the number of input events to 50k for all sequences in this evaluation[3]. Table 3.4 shows that both our FlowNet architectures, which are specifically trained to perform event deblurring, are in line with or outperform the state-of-the-art EV-FlowNet trained with either frames [25] or synthetic ground truth [150] according to this metric. More interestingly, FireFlowNet outperforms our EV-FlowNet in both datasets.

	ECD*	HQF
EV-FlowNet$_{\text{FW-MVSEC}}$ [25]	1.36	1.25
EV-FlowNet$_{\text{GT-SIM}}$ [150]	**1.51**	1.39
EV-FlowNet$_{\text{EW-DR}}$ (Ours)	1.31	<u>1.51</u>
FireFlowNet$_{\text{EW-DR}}$ (Ours)	<u>1.39</u>	**1.58**

Table 3.4: Quantitative evaluation of our FlowNet architectures on the ECD [172] and HQF [150] datasets. For each dataset, we report the mean FWL↑ [150]. Best in bold, runner up underlined.

[3]Note that the formulation of the FWL metric is sensitive to the number of input events [150].

	ECD*			HQF		
	MSE↓	SSIM↑	LPIPS↓	MSE↓	SSIM↑	LPIPS↓
E2VID [147]	0.08	0.54	0.37	0.14	0.46	0.45
FireNet [147]	0.06	0.57	0.29	0.07	0.48	0.42
E2VID+ [150]	**0.04**	**0.60**	**0.27**	**0.03**	**0.57**	**0.26**
FireNet+ [150]	0.06	0.51	0.32	0.05	0.47	0.36
$E2VID_F$ (Ours)	0.07	0.52	0.38	0.07	0.44	0.47
$E2VID_E$ (Ours)	0.06	0.55	0.37	0.06	0.48	0.47
$FireNet_F$ (Ours)	0.06	0.52	0.38	0.06	0.46	0.47
$FireNet_E$ (Ours)	0.06	0.51	0.41	0.06	0.46	0.51

Table 3.5: Quantitative evaluation of our ReconNet architectures on the ECD [172] and HQF [150] datasets. Best in bold; runner up underlined.

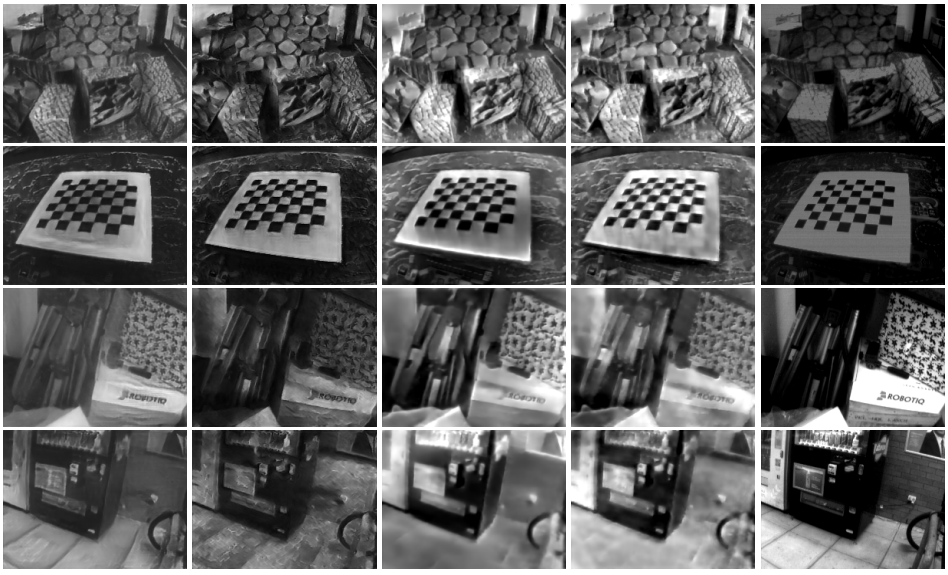


Figure 3.5: Qualitative comparison of our method with the E2VID+ and FireNet+ architectures [150] on sequences from the ECD [172] and HQF [150] datasets. Local histogram equalization not used for this comparison.

3.4.2 Reconstruction evaluation

We evaluated the accuracy of our ReconNet architectures against the DAVIS240C [174] frames from the ECD [172] and HQF [150] datasets, and compared their performance to the state-of-the-art of image reconstruction networks trained with ground-truth supervision: E2VID [147], FireNet [148], E2VID+ [150], and FireNet+ [150]. We used the results and code provided by Stoffregen *et al.* [150] for the quantitative and qualitative evaluations. The subscripts F and E indicate whether our networks were trained together with FireFlowNet or EV-FlowNet.

For all methods, reconstructions were generated at each DAVIS frame timestamp. We first applied local histogram equalization [175] to both frames, and then computed mean

Figure 3.6: Additional qualitative results on Prophesee's high-resolution automotive dataset [178]. The optical flow color-coding scheme can be found in Fig. 3.9.

squared error (MSE↓, lower is better), structural similarity (SSIM↑, higher is better) [176], and perceptual similarity (LPIPS↓, lower is better) [177]. For this evaluation, instead of using a fixed number of input events, we used all the events in between DAVIS frames. Results are presented in Table 3.5 and Figs. 3.5 and 3.6.

Despite not using any ground-truth data during training, results show that our method is in line with the state-of-the-art in terms of reconstruction accuracy. Quantitatively, the error metrics of all our ReconNet architectures closely resemble the results obtained with the original E2VID and FireNet, but the accuracy gap increases if compared against these same networks trained with the refined data augmentation mechanisms from Stoffregen *et al.* [150]. This gap is particularly notable in the LPIPS loss because these literature networks are specifically trained to maximize perceptual similarity to ground-truth frames. On the other hand, there is no major quantitative difference between the evaluated versions of ReconNet, regardless of their architecture or the accompanying flow network.

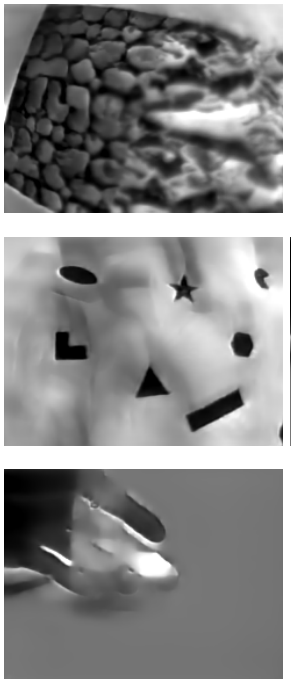


Figure 3.7: Common failure cases of our SSL framework.

Qualitative results confirm that our method reconstructs high quality HDR images. However, it is possible to identify several differences with respect to the state-of-the-art. Firstly, our images appear less sharp. Our architectures learn to correlate the spatial gradients of the estimated brightness $\hat{L}$ to the IWE (see Section 3.3.4). This entails that the reconstructed images are affected by the accuracy of the optical flow. Suboptimal optical flow estimations lead to imperfect event deblurring during training, which in turn is reflected in the reconstructed images as motion blur. Note that this blur diminishes when using an appropriate fixed number of input events for each sequence. Secondly, the dynamic range of the images differs. State-of-the-art methods learn to map the input events into bounded estimates of $\hat{L}$ via supervised learning. On the contrary, our brightness estimate is unbounded, and normalization is used to encode this signal as bounded images. Besides this, there is no significant difference between the evaluated ReconNet versions, despite the limited smoothing capabilities of FireNet. Lastly, although our method does not suffer from the stretch marks mostly

	ECD*			HQF		
	MSE↓	SSIM↑	LPIPS↓	MSE↓	SSIM↑	LPIPS↓
$E2VID_E$ (w/ deblurring)	**0.06**	**0.55**	**0.37**	**0.06**	**0.48**	**0.47**
$E2VID_E$ (w/o deblurring)	0.14	0.30	0.58	0.11	0.28	0.64

Table 3.6: Quantitative evaluation of the impact of event deblurring on the ECD [172] and HQF [150] datasets. For each dataset, we report the mean MSE (↓), SSIM [176] (↑) and LPIPS [177] (↓). Best in bold.

Figure 3.8: Qualitative evaluation of the impact of event deblurring on the quality of the reconstructed frames on sequences from the ECD [172] dataset.

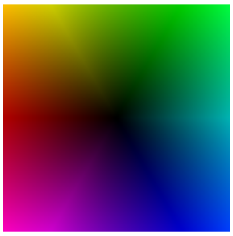


Figure 3.9: Optical flow field color-coding scheme. Direction is encoded in color hue, and speed in color brightness.

present in FireNet+ images, it is characterized by three common failure cases. As shown in Fig. 3.7, these are: (i) the aforementioned motion blur, (ii) "ghosting" artifacts in large texture-less regions due to limited extrapolation of edge information, and (iii) incoherent reconstructions due to the lack of information about the initial brightness L_0.

3.4.3 Impact of event deblurring

As discussed in this work, our self-supervised image reconstruction framework is designed around the event-based photometric constancy equation. While the right-hand side of this equation is obtained via the dot product between the warped spatial gradients of the last reconstructed image and the estimated optical flow; we propose that the left-hand side is obtained by integrating the deblurred input events. Since the main supervisory signal used to train our image reconstruction architectures comes from the comparison of the two sides of this equation, after training, the spatial gradients of the reconstructed images are correlated with the integrated events. These events, if not warped to the timestamp of the reconstructed frame, would introduce motion blur into the images. The amount of motion blur would depend on the density of events and on the length of the partition of events.

To validate this approach, we conducted an ablation study in which we trained the same ReconNet architecture (accompanied by the same pre-trained optical flow network) with and without event deblurring prior to event integration. Quantitative results are presented in Table 3.6, and are supported by qualitative results in Fig. 3.8. As shown, event deblurring is a crucial mechanism to reconstruct sharp images from the events. Without it, the reconstructed frames appear less sharp for the same number of input events, and the network is characterized by significantly worse error metrics on the evaluation datasets.

3.5 Conclusion

In this chapter, we went back to the basics of event cameras and presented the first self-supervised learning-based approach to event-based image reconstruction, which does not rely on any ground-truth or synthetic data during training. Instead, our SSL method makes use of the flow-intensity relation used by early methods to reconstruct the frames that best satisfy the input events and the estimated optical flow. Results confirm that our method performs almost as well as the state-of-the-art, but that the reconstructed images are characterized by several artifacts that need to be addressed by future work. Additionally, we presented FireFlowNet: a fast, lightweight neural network that performs event-based optical flow estimation. We believe this work shows the exciting potential of SSL to take over the research on image reconstruction from event data, and it opens up avenues for further improvement by leveraging the great amount of unlabeled event data available. Moreover, we have proposed a general self-supervised learning framework that can be extended in multiple ways via more sophisticated reconstruction losses and other event-based optical flow algorithms.

Supplementary material

Video summary of the approach: https://youtu.be/eiIIhY7HeQM

Project code: https://github.com/tudelft/ssl_e2vid

4

Unsupervised Learning of Event-Based Optical Flow with SNNs

Although the reconstruction of sharp images from event data presented in the previous chapter offers the potential for higher rates compared to frame-based cameras, it is important to consider that downstream applications would still have to process conventional frames with per-pixel data. Consequently, these algorithms are unable to fully leverage the inherent sparsity and asynchrony of event-based cameras. For this reason, here we take a leap into spiking neural networks, as their combination with event-based vision sensors holds the potential of highly efficient and high-bandwidth processing. This chapter presents the first hierarchical spiking architecture in which motion (direction and speed) selectivity emerges in an unsupervised fashion from the raw stimuli generated with an event-based camera. A novel adaptive neuron model and stable spike-timing-dependent plasticity formulation are at the core of this neural network governing its spike-based processing and learning, respectively. After convergence, the neural architecture exhibits the main properties of biological visual motion systems, namely feature extraction and local and global motion perception. Convolutional layers with input synapses characterized by single and multiple transmission delays are employed for feature and local motion perception, respectively; while global motion selectivity emerges in a final fully-connected layer. The proposed solution is validated using synthetic and real event sequences.

The contents of this chapter have been published in:

F. Paredes-Vallés, K. Y. W. Scheper, G. C. H. E. de Croon, *Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception*, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2019.

4.1 Introduction

Whenever an animal endowed with a visual system navigates through an environment, turns its gaze, or simply observes a moving object from a resting state, motion patterns are perceivable at the retina level as spatiotemporal variations of brightness [179]. These patterns of apparent motion, formally referred to as optical flow [21], are a crucial source of information for these animals to estimate their ego-motion and to have a better understanding of the visual scene. A great example of the efficacy of these cues in nature is in flying insects [179, 180], which are believed to heavily rely on these visual cues to perform high-speed maneuvers such as horizontal translation or landing [181].

Considering their size and weight limitations, insects are a clear indicator of the efficiency, robustness, and low latency of the optical flow estimation conducted by biological systems. The ability to reliably mimic this procedure would have a significant impact on the field of micro-robotics due to the limited computational capacity of their on-board processors. As an example, micro air vehicles (MAVs), such as the DelFly Explorer [182] or the DelFly Nimble[11], could benefit from a bio-inspired visual motion estimation for high-speed autonomous navigation in cluttered environments.

Biological visual systems receive their input from photoreceptors in the retina. These light-sensitive neurons absorb and convert incoming light into electrical signals which serve as input to the so-called ganglion cells. The activity of these neurons consists of temporal sequences of discrete spikes (i.e., voltage pulses) that are sent to large networks of interconnected cells for motion estimation, among other tasks. Since it is spike-driven, these biological architectures are characterized by a sparse, asynchronous, and massively parallelized computation. Further, they are seen to adapt their topology, i.e., connectivity pattern, in response to visual experience [183, 184]. This adaptation, or learning mechanism, allows them to operate robustly in different environments under a wide range of lighting conditions.

In contrast, the working principle of the majority of cameras used for artificial visual perception is categorized as frame-based: data is obtained by measuring the brightness levels of a pixel array at fixed time intervals. Although convenient for some computer vision applications, these sensors are inefficient for the task of motion estimation as the frame rate is independent of the dynamics of the visual scene. Additionally, due to the limited temporal resolution of these sensors, rapidly moving objects may introduce motion blur, limiting the accuracy of optical flow estimation.

However, not all artificial systems rely on conventional frame-based cameras for visual motion estimation. Inspired by biological retinas, several event-based cameras have recently been presented [12, 174, 185, 186]. Similar to ganglion cells, each of the elements of the pixel array reacts asynchronously to brightness changes in its corresponding receptive field by generating events. A microsecond temporal resolution, latencies in this order of magnitude, a wide dynamic range, and a low power consumption make these sensors an ideal choice for visual perception.

Regardless of the vision sensor, the estimation of optical flow by artificial systems is normally performed algorithmically, with solutions that are built on simplifying assumptions that make this problem tractable [187, 188]. In spite of this, the recent progress in parallel computing hardware has enabled artificial motion perception to be addressed from a more bio-inspired perspective: artificial neural networks (ANNs). Similar to biological

architectures, ANNs consist of large sets of artificial neurons whose interconnections can be optimized for the task at hand. However, despite the high accuracy reported with both frame- [189] and event-based sensors [25, 190], there is still a fundamental difference: the underlying communication protocol in ANNs relies on synchronous packages of floating-point numbers, rather than on trains of asynchronous discrete spikes. As a consequence, these architectures are often computationally expensive.

Taking further inspiration from nature, spiking neural networks (SNNs) have been proposed as a new generation of ANNs [191]. As the name suggests, the computation carried out by these bio-realistic neural models is asynchronous and spike-based, which makes them a suitable processing framework for the sparse data generated by event-based sensors [192]. Moreover, SNNs can benefit from an efficient real-time implementation in neuromorphic hardware, such as IBM's TrueNorth chip [48] or Intel's Loihi processor [14]. Despite these promising characteristics, the spiking nature of these networks limits the application of the successful gradient-based optimization algorithms normally employed in ANNs. Instead, at the time of publication, learning in SNNs is dominated by spike-timing-dependent plasticity (STDP) [41], a biologically plausible protocol that adapts the strength of a connection between two neurons based on their correlated activity. STDP has been successfully applied to relatively simple image classification tasks [43–47]. However, until now, no study has discussed the use of this learning rule for the estimation of event-based optical flow.

This chapter contains three main contributions. First, a novel adaptive mechanism for the leaky integrate-and-fire (LIF) spiking neuron model [193] is introduced. Second, a novel, inherently-stable STDP implementation is proposed. With this learning rule, the strength of neural connections naturally converges to an equilibrium distribution without the need for the ad-hoc mechanisms used by most of the existing formulations. Third, the proposed neuron model and STDP rule are combined in a hierarchical SNN architecture that, after learning, resembles the main functionalities of biological visual systems: feature extraction and local and global motion perception. To the best of the authors' knowledge, this chapter shows, for the first time, that neural selectivity to the local and global motion of input stimuli can emerge from visual experience in a biologically plausible unsupervised fashion.

4.2 Related work

Event-based cameras

Inspired by biological retinas, each of the pixels of an event-based camera reacts asynchronously to local changes in brightness by generating discrete temporal events. Specifically, the generation of an event is triggered whenever the logarithmic change of the image intensity $L(x, y, t)$ exceeds a predefined threshold C such that $\left|\Delta \log\big(L(x, y, t)\big)\right| > C$ [12]. This variation is computed with respect to a reference brightness level set by the last occurring event at that pixel.

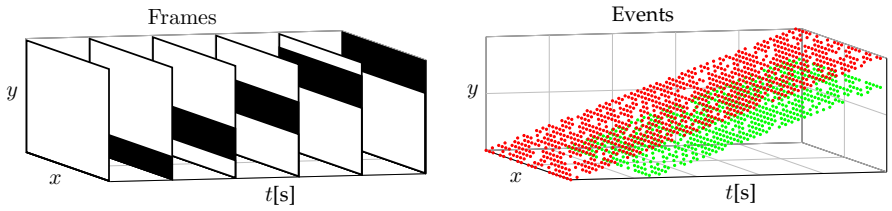


Figure 4.1: Comparison of the output of frame- and event-based cameras under the stimulus of a black horizontal bar moving upward over a homogeneous white background.

Each event encodes information about its timestamp t, its corresponding (x, y) location in the pixel array, and the polarity $P \in \{-1, 1\}$ of the intensity change. This communication protocol is referred to as address-event representation (AER). A visual comparison of the output of frame- and event-based sensors under the same stimulus is illustrated in Fig. 4.1.

Spiking neural networks

Models of spiking neurons In biological networks, neural communication consists in the exchange of voltage pulses [191]. For the reproduction of this asynchronous and spike-based mechanism in SNNs, multiple models of spiking neurons have been presented at various levels of abstraction. Biophysical formulations lead to accurate representations of neural dynamics [194], however, their complexity limits their use in large-scale networks. Alternatively, phenomenological formulations offer a compromise between computational load and biological realism. The most used models are the aforementioned LIF [193], the Izhikevich [195], and the spike response model [196].

From a conceptual perspective, the majority of these models share some fundamental principles and definitions. The junction of two neurons is called synapse; and relative to these cells, the transmitting neuron is labeled as presynaptic, while the receiving as postsynaptic. Each spiking neuron, as processing unit, is characterized by an internal state variable, known as membrane potential $U_i(t)$, which temporally integrates presynaptic spikes over time. If the arrival of a spike leads to an increase (decrease) in $U_i(t)$, then the spike is said to have an excitatory (inhibitory) effect on the neuron. $U_i(t)$ decays to a resting potential U_{rest} in case no input is received. Lastly, a postsynaptic spike is triggered whenever $U_i(t)$ crosses the firing threshold θ. Afterwards, the neuron resets its membrane potential to U_{reset}, and enters in a refractory period Δt_{refr} during which new

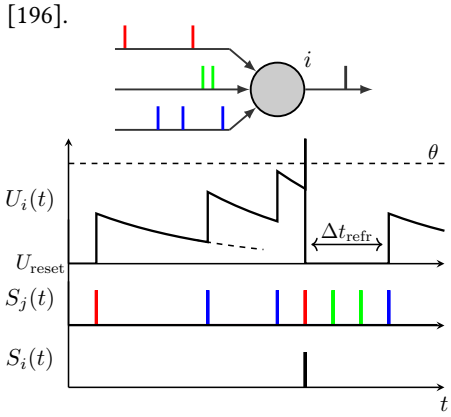


Figure 4.2: A model of a LIF neuron. The graphic (*bottom*) shows the temporal course of the membrane potential $U_i(t)$ of the ith neuron (*top*), driven by a sample presynaptic spike train $S_j(t)$ from three input neurons $j = 1, 2, 3$. Spikes are depicted as vertical bars at the time at which they are received (if presynaptic) or emitted (if postsynaptic). In this schematic, the reset U_{reset} and resting U_{rest} potentials are equal in magnitude.

incoming spikes have negligible effect on $U_i(t)$. Fig. 4.2 illustrates these concepts for the case of a LIF neuron [193].

Synaptic plasticity Defined as the ability to modify the efficacy (weight) of neural connections, synaptic plasticity is the basic mechanism underlying learning in biological networks [197]. These architectures are seen to rely on different learning paradigms depending on their duty [198]. For instance, information encoding in biological visual systems is established in an unsupervised fashion, while reinforcement and supervised learning are employed for tasks such as decision making and motor control. Accordingly, various forms of synaptic plasticity have been proposed for SNNs.

In the context of SNNs, unsupervised learning is generally referred to as Hebbian learning, since plasticity rules from this paradigm are based on Hebb's postulate: "*cells that fire together, wire together*" [42]. In essence, these methods adapt the weight of a connection based on the correlated activity of pre- and postsynaptic cells. Among others, the biologically plausible STDP protocol is, by far, the most popular Hebbian rule for SNNs [41]. With STDP, the repeated arrival of presynaptic spikes to a neuron shortly before it fires leads to synaptic strengthening, also known as long-term potentiation (LTP); whereas if the arrival occurs shortly after the postsynaptic spike, synapses are weakened through long-term depression (LTD). Therefore, the change of weight ΔW is normally expressed as a function of the relative timing between these two events. STDP formulations exclusively dependent on this parameter are referred to as additive rules [199]. These models, despite their success in pattern recognition problems [43–46, 200], are inherently unstable and require the use of constraints for the synaptic weights, thus resulting in bimodal distributions [41]. On the other hand, multiplicative STDP rules incorporate the current weight value in the computation of ΔW in an inversely proportional manner. As shown in [47, 201], this additional dependency leads to stable unimodal weight distributions. However, the stability of these approaches results from a complex temporal LTP-LTD balance, and it is not theoretically guaranteed.

Several lines of research can be distinguished regarding the use of supervised learning in SNNs, being the most promising based on the well-known error backpropagation algorithm [202]. Firstly, numerous adaptations to the discontinuous dynamics of SNNs have recently been proposed for learning temporally precise spike patterns [203–206]. Alternatively, due to the popularity of this method in ANNs, SNNs commonly rely on transferring optimization results from their non-spiking counterparts [207–209]. In both cases, high accuracy levels are reported in image classification tasks, but still far below from those obtained with conventional ANNs.

With respect to reinforcement learning in SNNs, various models have been presented, the majority of which consist in the modulation of STDP with a reward function [210, 211]. However, applications of this paradigm are mainly focused on neuroscience research [212, 213], besides several goal-directed navigation problems [214, 215] and the digit-recognition application recently presented in [216].

Event-based optical flow estimation

The recent introduction of event-based cameras and other retinomorphic vision sensors has precipitated the development of several novel approaches to event-based optical flow esti-

mation. Depending on their working principle, these solutions are divided into algorithmic and neural methods.

Gradient-, plane-fitting-, frequency-, and correlation-based approaches set the basis of the algorithmic state-of-the-art. These techniques compute sparse optical flow estimates for each newly detected event (or group of events) based on its spatiotemporal, polarity-specific neighborhood. Firstly, adaptations of the gradient-based Lucas-Kanade algorithm [163] were presented in [57, 217]. Secondly, the methods proposed in [70, 187, 218] extract optical flow by computing the gradients of a local plane fitted to a spatiotemporal surface of events. Thirdly, multiple adaptations of the bio-inspired frequency-based methods have been introduced [217, 219, 220], which allow the implementation in neuromorphic hardware [221]. Lastly, the recent correlation-based approaches presented in [96, 161, 222, 223] employ convex optimization algorithms to associate groups of events over time, and report the highest algorithmic accuracy to date. Part of this category is also the block-matching method recently proposed in [224], which employs conventional search techniques to find the best matching group of events in previous temporal slices of the input.

The estimation of event-based optical flow with neural models is dominated by SNNs. However, there are a couple of ANN-based approaches worth remarking. In [25], a self-supervised learning scheme was employed to train a convolutional ANN to estimate dense image flow. The input to the network consists of the per-pixel last timestamp and count of events over a specific time window. Using the average timestamp instead, in [190], the authors presented the first neural model to approach the full structure-from-motion problem using event-based input. Here, two ANNs are employed for depth and dense optical flow estimation. Regarding the latter task, accuracy levels considerably higher than those from [25] are reported.

Though the main goal of [225] is for predicting future input activations, this work presented the first neural architecture capable of learning spatiotemporal features from raw event data. For this purpose, a combination of multiple recurrent ANNs with a single layer of spiking neurons was employed. As explained below, these features are potentially useful as efficient motion detectors, but no study has discussed their learning with SNNs.

With respect to pure SNN-based approaches, in [226, 227], the authors propose an architecture in which motion selectivity results from the synaptic connections of a bursting neuron to two neighboring photoreceptors, one excitatory and the other inhibitory. If the edge is detected first by the excitatory cell, spikes are emitted at a fixed rate until the inhibitory pulse is received. Otherwise, the neuron remains inactive. Optical flow is consequently encoded in the burst length and in the relative orientation of the photoreceptors.

In contrast, the SNNs presented in [228, 229] extract motion information through synaptic delays and spiking neurons acting as coincidence detectors. A simple spike-based adaptation of the Reichardt model [230] is introduced in [228] to show the potential of this approach. This idea is explored in more detail in [229], in which the authors propose the convolution of event sequences with a bank of spatiotemporally-oriented filters, each of which is comprised of non-plastic synapses with equal weights, but with delays tuned to capture a particular direction and speed. Similarly to frequency-based methods [231], these filters compute a confidence measure, encoded in the neural activity, rather than the optical flow components. Additionally, this solution employs a second spike-based pooling layer for mitigating the effect of the aperture problem [232].

Whether, and how, direction and speed selectivity emerge in biological networks from visual experience still remains an open question. Some initial work by [233–235] shows that robust local direction selectivity arises in neural maps through STDP if, apart from presynaptic feedforward connections, neurons receive spikes from cells in their spatial neighborhood through plastic synapses with distance-dependent transmission delays. However, no study has assessed the speed selectivity of these cells, which is crucial for optical flow estimation.

4.3 Method

4.3.1 Optical flow visual observables

The optical flow formulation employed throughout this study is introduced in the present section. This model relates the ego-motion of a downward-looking camera over a static planar scene to the perceived optical flow and its corresponding visual observables. We use this setting as it corresponds to the widely studied problem of optical-flow-based landing [70, 72, 74, 75]. However, note that we employ this optical flow formulation to improve the interpretability of the results. The proposed approach can also handle unstructured motion and scenes.

The derivation of this optical flow model relies on the two reference frames illustrated in Fig. 4.3. The inertial world frame is denoted by $\mathcal{W}$, whilst $\mathcal{C}$ describes the camera frame centered at the focal point of the event-based camera. In each of these frames, position is defined through the coordinates (X, Y, Z), with (U, V, W) as the corresponding velocity components. The orientation of $\mathcal{C}$ with respect to $\mathcal{W}$ is described by the Euler angles ϕ, θ, and ψ, denoting roll, pitch, and yaw, respectively. Similarly, p, q, and r denote the corresponding rotational rates.

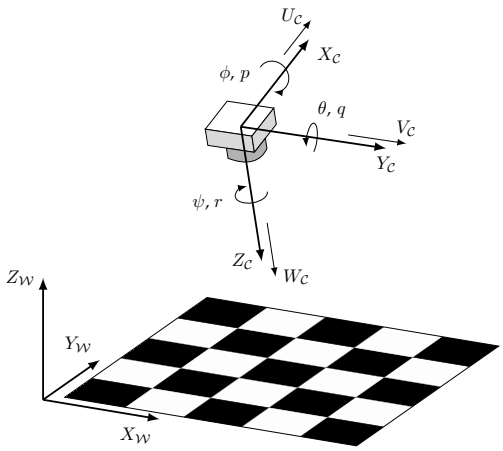


Figure 4.3: Definitions of the world ($\mathcal{W}$) and camera ($\mathcal{C}$) references frames. The Euler angles, rotational rates, and translational velocities that describe the motion of $\mathcal{C}$ are shown as well.

The relations between sensor ego-motion, optical flow, and visual observables are based on the pinhole camera model [236]. In this formulation, pixel coordinates in the sensor's pixel array are denoted by (x, y), while (u, v) represent optical flow components, measured in pixels per second. Note that this model assumes an undistorted vision sensor.

Consider the situation depicted in Fig. 4.3, in which the sensor $\mathcal{C}$ moves arbitrarily through a static environment subjected to translational $(U_\mathcal{C}, V_\mathcal{C}, W_\mathcal{C})$ and rotational (p, q, r) velocities. Due to this ego-motion, the projection of a world point onto the image plane leads to an optical flow of components:

$$
\begin{aligned}
u &= -\frac{U_C}{Z_C} + \frac{W_C}{Z_C}x - q + ry + pxy - qx^2 \\
v &= -\frac{V_C}{Z_C} + \frac{W_C}{Z_C}y + p - rx - qxy + py^2
\end{aligned}
\tag{4.1}
$$

From Eq. 4.1, the optical flow of a point can be resolved into translational and rotational components [236]. Since the latter is independent of the three-dimensional structure of the visual scene, these expressions can be derotated if information on the rotational rates of the sensor is available. This derotation leads to pure translational optical flow components, denoted by (u_T, v_T). Moreover, if the scene is a planar surface, the depth Z_C of all visible world points is interrelated through:

$$
Z_C = Z_0 + Z_X X_C + Z_Y Y_C \tag{4.2}
$$

where Z_0 is defined as the distance to the surface along the optical axis of the sensor, and Z_X and Z_Y represent the slopes of the planar scene with respect to the X- and Y-axis of C [72]. If information on the attitude of the sensor is available, these slopes can be computed from the pitch and roll angles as:

$$
Z_X = \tan\theta, \quad Z_Y = -\tan\phi \tag{4.3}
$$

In [236], the relation between the position of an arbitrary point in C and its projection onto the image plane is given by $(x, y) = (X_C/Z_C, Y_C/Z_C)$. Consequently, Eq. 4.2 may also be written in the form:

$$
\frac{Z_C - Z_0}{Z_C} = Z_X x + Z_Y y \tag{4.4}
$$

Further, let the *scaled velocities* of the sensor ϑ_x, ϑ_y, and ϑ_z be defined as follows:

$$
\vartheta_x = \frac{U_C}{Z_0}, \quad \vartheta_y = \frac{V_C}{Z_0}, \quad \vartheta_z = \frac{W_C}{Z_0} \tag{4.5}
$$

Then, according to the derivations in [72], substituting Eqs. 4.4 and 4.5 into Eq. 4.1 leads to the following expressions for translational optical flow:

$$
\begin{aligned}
u_T &= (-\vartheta_x + \vartheta_z x)(1 - Z_X x - Z_Y y) \\
v_T &= (-\vartheta_y + \vartheta_z y)(1 - Z_X x - Z_Y y)
\end{aligned}
\tag{4.6}
$$

From Eq. 4.6, and under the aforementioned assumptions, the scaled velocities, which provide non-metric information on sensor ego-motion, can be derived from the translational optical flow of multiple image points. ϑ_x and ϑ_y are the opposites of the so-called *ventral flows*, a quantification of the average flows in the X- and Y-axis of C respectively [70]. Hence, $\omega_x = -\vartheta_x$ and $\omega_y = -\vartheta_y$. On the other hand, ϑ_z is proportional to the *divergence* of the optical flow field, $D = 2\vartheta_z$ [70]. Throughout this work, these optical flow visual observables, more specifically the ventral flow components, are employed to refer to the stimulus speed in the image plane.

4.3.2 Adaptive spiking neuron model

Let $j = 1, 2, \dots, n^{l-1}$ denote a group of presynaptic neurons, from layer $l-1$, fully connected in a feedforward fashion to a set of postsynaptic cells $i = 1, 2, \dots, n^{l}$, from layer l. As depicted in Fig. 4.4, these neural connections can be considered as multisynaptic, i.e., the link between two cells is not restricted to a single synapse, but several can coexist. In this work, the number of multisynaptic connections m is layer-specific, and each synapse has its own transmission delay as given by $\boldsymbol{\tau} \in \mathbb{R}^m$. In addition to this delay vector, layer connectivity is also characterized by a weight matrix $\boldsymbol{W} \in \mathbb{R}^{n^l \times n^{l-1} \times m}$, which determines the synaptic efficacy of the connections.

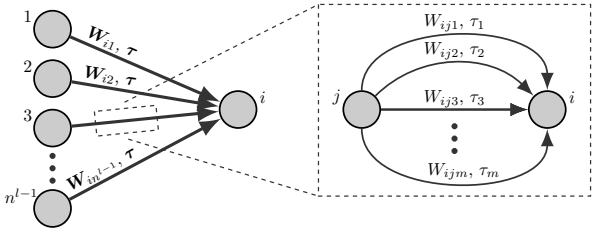


Figure 4.4: Schematic of the feedforward connectivity between neurons from two adjacent layers (*left*). These connections can be considered as being multisynaptic (*right*), each one having its own weight, transmission delay, and trace.

Apart from $\boldsymbol{W}$ and $\boldsymbol{\tau}$, each synapse keeps track of an additional parameter that captures the recent history of spikes transmitted. Referred to as the presynaptic trace [237], and defined as $\boldsymbol{X} \in \mathbb{R}^{n^l \times n^{l-1} \times m}$, its dynamics are given by:

$$\alpha_X \frac{dX_{ijd}(t)}{dt} = -X_{ijd}(t) + \lambda S_j^{l-1}(t - \tau_d) \tag{4.7}$$

where α_X is the time constant, λ is a scaling factor, and $\boldsymbol{S}^l(t) \in \mathbb{R}^{n^l}$ denotes the (binary) record of neural activity, or spike train, of cells from layer l. Note that $d = 1, 2, \dots, m$ serves to refer both to connections within a multisynaptic group and their corresponding delays.

From Eq. 4.7, whenever a spike arrives at a neuron i via a synapse with transmission delay τ_d, the corresponding presynaptic trace $X_{ijd}(t)$ increases by a factor of λ. In case no spike is received, the trace decays exponentially towards zero according to α_X.

The LIF model is the most widely used spiking neural model in literature. This is due to its main assumption that in SNNs, information is not encoded in the spike amplitude, but rather in the firing time. Consequently, neural activity is reduced to discrete and binary temporal events, thus ensuring computational tractability. The spiking neural model used in this chapter is a modified LIF model, defined as:

$$\alpha_U \frac{dU_i(t)}{dt} = -\big(U_i(t) - U_{\text{rest}}\big) + I_i(t) \tag{4.8}$$

$$I_i(t) = \sum_{j=1}^{n^{l-1}} \sum_{d=1}^{m} \big(W_{ijd} S_j^{l-1}(t - \tau_d) - X_{ijd}(t)\big) \tag{4.9}$$

where α_U denotes the time constant of the membrane potential, and $\boldsymbol{I}(t)$ is the so-called forcing function of the system.

From Eqs. 4.8 and 4.9, the membrane potential $U_i(t)$ of a neuron evolves over time by integrating scaled presynaptic spikes from its input synapses, similarly to the conventional

LIF model [193]. Whenever $U_i(t)$ reaches (or surpasses) the firing threshold θ, a postsynaptic spike is generated, i.e., $S_i^l(t) = 1$, and $U_i(t)$ is reset to U_{reset}. In addition, the neuron enters in a refractory period Δt_{refr} during which presynaptic spikes have no effect on $U_i(t)$ to ensure the temporal separation of postsynaptic pulses. In case no spike is fired at time t, this is reflected in the neuron's spike train as $S_i^l(t) = 0$.

Unlike traditional LIF [193], the forcing function $\boldsymbol{I}(t)$ of our neuron model includes an additional term, further referred to as the homeostasis parameter, which is inspired by the internal regulatory mechanisms of biological organisms [238]. This is used to adapt the neural response to the varying input statistics—in particular, to the per-pixel firing rate—using the presynaptic trace $\boldsymbol{X}$ as an excitability indicator. Inferring from Eq. 4.9, this parameter acts as an inhibitory penalty in the update rule of $\boldsymbol{U}(t)$. A postsynaptic neuron connected to a group of highly-active presynaptic cells is said to have low excitability due to its relatively high $\boldsymbol{X}$. For this neuron to fire, it needs to receive a large number of presynaptic spikes shortly separated in time. Conversely, the same cell connected to poorly-active neurons is highly excitable; and thus, the firing threshold θ can still be reached despite the considerably larger time difference between input spikes. Note that, to get the desired neural adaptation, the scaling factor α, from Eq. 4.7, needs to be selected in accordance with the neural parameters, mainly θ and the range of possible $\boldsymbol{W}$ values.

When dealing with an event-based camera as source of input spikes, the firing rate of the sensor is not only correlated to the appearance of features from the visual scene, but also to their optical flow and the sensitivity settings of the camera. Slow apparent motion leads to successive events being more distant in time than those captured from fast motion. Consequently, if these events are to be processed with a network of spiking neurons, a homeostasis mechanism is required to ensure that similar features are detected regardless of the encoding spike rate.

Other approaches to homeostasis have been presented in the literature, such as threshold balancing [239] or weight scaling [201]. However, these methods use postsynaptic spikes to adjust the homeostatic inhibition through an adaptive mechanism. With this neural feedback, there is a delay in adjusting the excitability of the neurons. These approaches are therefore less suitable for the rapidly varying statistics of the data generated by a moving event-based camera.

4.3.3 Stable STDP learning rule

In this work, we propose a novel multiplicative STDP implementation that, contrary to the state-of-the-art of this learning protocol, is inherently stable by combining the weight-dependent exponential rule from [47] with presynaptic trace information. Hereafter, we will simply refer to it as STDP.

Whenever a neuron i fires a spike, the weight of its presynaptic connections is updated as follows:

$$\Delta W_{ijd} = \eta(\text{LTP} + \text{LTD}) \tag{4.10}$$

$$\begin{aligned} \text{LTP} &= \text{LTP}_W \cdot \text{LTP}_{\hat{X}}, & \text{LTD} &= \text{LTD}_W \cdot \text{LTD}_{\hat{X}} \\ \text{LTP}_W &= e^{-(W_{ijd} - w_{\text{init}})}, & \text{LTD}_W &= -e^{(W_{ijd} - w_{\text{init}})} \\ \text{LTP}_{\hat{X}} &= e^{\hat{X}_{ijd}(t)} - a, & \text{LTD}_{\hat{X}} &= e^{(1-\hat{X}_{ijd}(t))} - a \end{aligned} \tag{4.11}$$

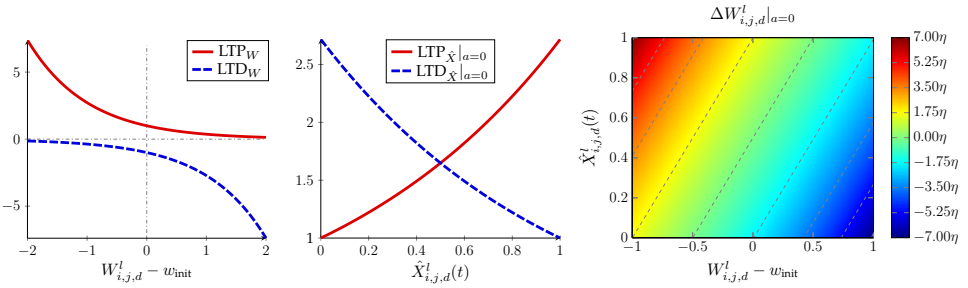


Figure 4.5: Illustration of the novel multiplicative STDP rule proposed in this work. The weight update (*right*) results from the linear combination of the non-exclusive LTP and LTD processes. These, in turn, are characterized by symmetrical dependencies on the synaptic weights (*left*) and normalized presynaptic traces (*center*). Note that, in the schematic of the weight update (*right*), the weight axis is limited to the $[-1, 1]$ range only for the purpose of a better visualization of the equilibrium weights (dashed thick line) for $a = 0$.

where η is the learning rate of the rule, w_{init} refers to the initialization weight of all synapses at the beginning of the learning process, and $\hat{\boldsymbol{X}}_i \in [0, 1]$ denotes the presynaptic traces of neuron i normalized to the current maximum at the moment of firing. Further, for stability, $\eta > 0$ and $a < 1$ regardless of the value of w_{init} (see Section 4.4.4).

From Eqs. 4.10 and 4.11, the weight update $\Delta \boldsymbol{W}_i$ results from the linear combination of the output of two non-mutually exclusive processes: LTP, for strengthening, and LTD, for weakening synaptic connections. Both of these processes are dependent on the weights (LTP_W, LTD_W) and normalized traces ($\text{LTP}_{\hat{X}}$, $\text{LTD}_{\hat{X}}$) of the synapses under analysis. On the one hand, the weight dependency of our learning rule takes inspiration from the STDP formulation presented in [47]. LTP_W and LTD_W are inversely proportional to $\boldsymbol{W}_i$ in an exponential fashion, and are centered around w_{init} (see Fig. 4.5, left). Consequently, the effect of LTP_W decreases (increases) the larger (smaller) a synaptic weight is in comparison to w_{init}. The opposite relation holds true for LTD_W. On the other hand, rather than relying on the precise spike timing [47], our rule employs normalized presynaptic trace information as a measure of the relevance of a particular connection to the postsynaptic spike triggering the update. The higher (lower) the value of $\hat{X}_{ijd}(t)$, the larger (smaller) the effect of $\text{LTP}_{\hat{X}}$, and vice versa for $\text{LTD}_{\hat{X}}$ (see Fig. 4.5, center).

With this formulation, a weight is established for each value of $\hat{X}_{ijd}(t)$ through a stable equilibrium of LTP-LTD contributions on $\Delta \boldsymbol{W}_i$ (see Fig. 4.5, right). The parameter a has control over this non-linear mapping through the steepness of $\text{LTP}_{\hat{X}}$ and $\text{LTD}_{\hat{X}}$ in $\hat{\boldsymbol{X}}_i \in [0, 1]$. The higher (lower) the value of a—below the stability limit—, the wider (narrower) the distribution of synaptic weights after convergence. As such, no additional mechanism is required for preventing weights from vanishing or exploding. Synapses characterized by weights that are higher (lower) than their corresponding equilibrium state are consistently depressed (potentiated) until synapse-specific stability is achieved.

To track the convergence of the learning process, we propose the use of the following mean square error criterion, where $\hat{\boldsymbol{W}}_i \in [0, 1]$ denotes the presynaptic weights of neuron i after an update, normalized to the current maximum:

$$\mathcal{L}_i = \frac{1}{n^{l-1} m} \sum_{j=1}^{n^{l-1}} \sum_{d=1}^{m} \left(\hat{X}_{ijd}(t) - \hat{W}_{ijd} \right)^2 \tag{4.12}$$

As the learning progresses, the moving average of $\mathcal{L}_i$ converges to a (close-to-zero) equilibrium state. In this work, we stop synaptic plasticity using a fixed threshold on this parameter, denoted by $\mathcal{L}_{\text{th}}$.

Local inter-lateral competition

For neurons to learn distinct features from the input data through STDP, this learning rule needs to be combined with what is known as a winner-take-all (WTA) mechanism [240]. This form of competition implies that, when a neuron fires a spike and updates its presynaptic weights according to Eqs. 4.10 and 4.11, the rest of postsynaptic cells (from the same layer) locally connected to the same input neurons get inhibited. As a result, these cells are prevented from triggering STDP while the neuron that fired first, i.e., the winner, remains in the refractory period.

Instead of relying on non-plastic synapses transmitting inhibitory spikes with a certain delay, our implementation assumes that the internal dynamics of these neurons are inter-correlated. Whenever the winner resets its membrane potential and enters in the refractory period, neurons affected by the WTA mechanism do the same immediately afterwards. In case multiple neurons fire simultaneously, the cell with the highest membrane potential has preference for triggering the weight update. Further, the postsynaptic spikes from the other firing neurons are not considered. To ensure coherence between the training and inference phases of our proposed SNN, layers trained with STDP maintain the WTA mechanism after the learning process.

4.3.4 Network architecture for motion perception

To extract a robust measure of motion from the raw camera input, we propose the multi-layer SNN illustrated in Fig. 4.6. This section highlights the unique goal of each of the layers comprising this architecture, together with the variations of the proposed neuron model and learning rule that are required depending on their connectivity scheme.

Input layer

Being the first stage of the network, the input layer encodes the event-based sensor data in a compatible format for the rest of the architecture. This layer can be understood as to be comprised of spiking neurons with no internal dynamics, whose neural activity is determined by event arrival. Neurons are arranged in two-dimensional neural maps, one per polarity, resembling the grid-like topology of the vision sensor. Depending on the spatial resolution of these maps, each neuron is assigned with the polarity-specific events of one or multiple pixels with no overlap.

SS-Conv layer: Feature extraction

The goal of the single-synaptic convolutional layer, or SS-Conv, is to extract visual features from the input, and by doing so, to filter out the input events that may otherwise corrupt the learning process, and hence the performance, of subsequent layers in the architecture.

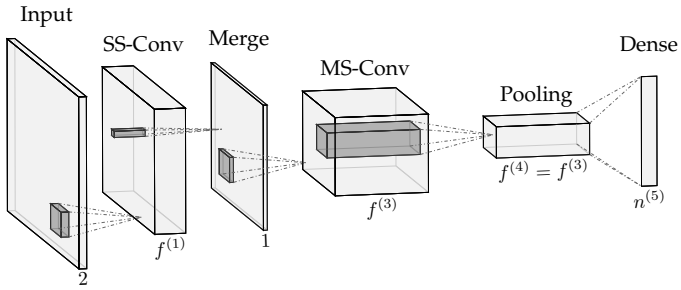


Figure 4.6: Overview of the feedforward SNN architecture.

Neurons in this layer are retinotopically arranged in $k = 1, 2, \ldots, f^{(1)}$ two-dimensional maps. Each of these neurons receives spikes from presynaptic cells within a specific spatial receptive field, of size r, in all maps of the previous layer. This sparse connectivity is characterized by a set of excitatory synaptic weights, formally referred to as a convolutional kernel $\boldsymbol{W}_k \in \mathbb{R}^{r \times f^{(0)}}$, that is equal for all neurons belonging to the same map. This form of weight sharing ensures that, within a map, neurons are selective to the same feature but at different spatial locations.

Let the input connectivity of neuron i from the map k be characterized by the aforementioned convolutional kernel $\boldsymbol{W}_k$, the presynaptic trace $\boldsymbol{X}_i \in \mathbb{R}^{r \times f^{(0)}}$, and the spike train $S_{ik}^{(0)}(t)$. Further, let $\boldsymbol{N}_{ik}$ refer to the map-specific direct neural neighborhood of the cell, including itself. Then, considering neural connections as single-synaptic with transmission delay τ, the forcing function driving the internal dynamics of neurons in this layer is defined as follows:

$$I_{ik}(t) = \sum_{j=1}^{r} \sum_{c=1}^{f^{l-1}} W_{jck} S_{jc}^{l-1}(t - \tau) - \max_{\forall b \in \boldsymbol{N}_{ik}} \sum_{j=1}^{r} \sum_{c=1}^{f^{l-1}} X_{bjc}(t) \tag{4.13}$$

Apart from the sparse connectivity, the only difference between this expression and the fully-connected formulation, i.e., Eq. 4.9, is in the homeostasis parameter. When arranged retinotopically, the neurons' dynamics do not only depend on their own presynaptic trace $\boldsymbol{X}_i$, but also on the synaptic traces characterizing their direct spatial neural neighborhood $\boldsymbol{N}_{ik}$. By using the maximum trace, neurons are prevented from specializing to the leading edge of moving visual features, rather than to the features themselves (see Section 4.4.4).

An augmentation of the proposed STDP rule is also required to handle the fact that multiple updates can be generated simultaneously in different spatial locations of the same map. Since these neurons share convolutional kernel, $\Delta \boldsymbol{W}_k$ is computed through synapse-specific averages of the local contributions. Additionally, due to the high overlap of presynaptic receptive fields, the WTA inhibitory mechanism described in Section 4.3.3 is expanded to cells within a small neighborhood of the firing neurons, regardless of the neural map. Note that, after learning, only the neuron-specific competition is maintained.

Merge layer: Feature aggregation

Due to the aperture problem [232], the different types of local motion that can be perceived at this stage of the architecture are exclusively dependent on the spatial configuration

of input features, i.e., their appearance, and not on their polarity. Consequently, the $f^{(1)}$ neural maps of the SS-Conv layer can be merged into a single combined map without losing useful information for motion perception. The merge layer is used for this purpose. Compared to when local motion is to be perceived directly from the SS-Conv output, this operation results in a decrease of both the number of convolutional kernels required in the subsequent layer, and the amount of per-kernel trainable parameters.

Similarly to SS-Conv, the merge layer is convolutional and single-synaptic. The internal dynamics of its neurons are driven by Eq. 4.13 (with $l = 2$ in this case), but without the need for $\boldsymbol{N}_{ik}$ since presynaptic connections are not plastic. Because of the latter, the application of the WTA mechanism is also neglected. Instead, this layer is characterized by a single 1×1 convolutional kernel with unitary connections to each of the neural maps of the previous layer.

MS-Conv layer: Local motion perception

MS-Conv is presented as a variation of the SS-Conv layer whose role is to provide local motion estimates of the features extracted in the previous layers, by means of velocity-selective neurons. Similarly to feature identification, this selectivity emerges from visual experience through STDP.

For the purpose of local motion perception, we propose an augmentation of Eq. 4.13 based on the foundations of frequency-based optical flow methods [231] and bio-inspired motion detectors [230, 241]. Firstly, motion is to be extracted as orientation in the spatiotemporal domain. Therefore, neural connections in the MS-Conv layer are considered multisynaptic with different constant transmission delays as given by $\boldsymbol{\tau} \in \mathbb{R}^m$. Secondly, since these delays (and the rest of neural parameters) are equal for all (spatiotemporal) convolutional kernels, inhibitory synapses are required to prevent the firing of erroneous postsynaptic spikes when the input trace only fits part of the excitatory component of the kernels. To account for this, each MS-Conv kernel is defined by a pair of excitatory and inhibitory plastic weight matrices, denoted by $\boldsymbol{W}_k^{\text{exc}} \in \mathbb{R}^{r \times m}$ and $\boldsymbol{W}_k^{\text{inh}} \in \mathbb{R}^{r \times m}$, respectively. According to these additions, the forcing function of cells in this layer is expressed as:

$$I_{ik}(t) = \sum_{j=1}^{r} \sum_{d=1}^{m} (W_{jdk}^{\text{exc}} + \beta W_{jdk}^{\text{inh}}) S_j^{(2)}(t - \tau_d) - \max_{\forall b \in \boldsymbol{N}_{ik}} \sum_{j=1}^{r} \sum_{d=1}^{m} X_{bjd}(t) \tag{4.14}$$

where $\beta \in [0, 1]$ scales the impact of inhibitory synapses, and the presynaptic trace is defined as $\boldsymbol{X}_i \in \mathbb{R}^{r \times m}$.

Due to the neural spatial disposition, the implementation of STDP in this layer is, in essence, identical to the one employed for SS-Conv. The only difference comes from the fact that, for inhibitory synapses, the weights are initialized at 0, and $w_{\text{init}}^{\text{inh}}$ is set to $-w_{\text{init}}^{\text{exc}}$ (see Eq. 4.11). This discrepancy between $w_{\text{init}}^{\text{inh}}$ and the initialization weight enables neurons in this layer to be reactive to different input features until specialization.

Pooling layer: From local to global

As an intermediate stage between the MS-Conv and dense layers, the pooling layer is employed in the SNN architecture as a means to reduce the spatial dimensionality of the former, and hence to facilitate the learning process of the latter. The intuition of this layer is that, by pooling local motion estimates over large portions of the visual scene, a

more accurate measure of the global motion in each of these regions can be obtained, thus mitigating the effect of the aperture problem [232].

Similarly to the merge layer, the pooling layer is convolutional and single-synaptic, and its presynaptic connections are not plastic. This layer is characterized by the same number of neural maps as the MS-Conv, each one assigned with an excitatory kernel $\boldsymbol{W}_k$ that has unitary weights with its presynaptic counterpart and null with the rest. In addition, there is no overlap between receptive fields.

Dense layer: Global motion perception

The dense layer, as the final stage of the architecture, is comprised of individual neurons fully connected to cells in the pooling layer via single-synaptic plastic connections. Similarly to final regions of biological visual motion systems [179, 180], neurons in this layer develop selectivity to the global motion of the scene from visual experience through STDP.

With respect to implementation details, synaptic plasticity is conducted as described in Section 4.3.3, and the forcing function of dense neurons resembles Eq. 4.9, but referring to the convolutional presynaptic layer to which these cells are connected. This expression is then defined as:

$$I_i(t) = \sum_{j=1}^{n^{(4)}} \sum_{c=1}^{f^{(4)}} \left(W_{ijc} S_{jc}^{(4)}(t-\tau) - X_{ijc}(t) \right) \tag{4.15}$$

where the weights and trace of input connections are defined as $\boldsymbol{W}_i \in \mathbb{R}^{n^{(4)} \times f^{(4)}}$ and $\boldsymbol{X}_i \in \mathbb{R}^{n^{(4)} \times f^{(4)}}$, respectively.

4.4 Experiments

In this section, we evaluate the performance of our convolutional SNN on synthetic [172] and real event sequences [56, 172]. All the experiments are conducted using our open-source CUDA-based *cuSNN* library, with a simulation timestep of Δt_{sim}=1 ms. Concerning learning, the networks are trained in a layer-by-layer fashion using the unsupervised STDP rule presented in Section 4.3.3. Regardless of the layer type, the parameter a from Eq. 4.11 is set to 0, the initialization weight to $w_{\text{init}} = 0.5$, the learning rate η to 1×10^{-4}, and the convergence threshold to $\mathcal{L}_{\text{th}}$=5×10^{-2}. As shown in Figs. 4.5 (right) and 4.16c, these parameters lead to weight distributions that, after convergence, are naturally constrained in the range $\boldsymbol{W} \in [0, 1]$ for excitatory synapses, and $\boldsymbol{W} \in [-1, 0]$ for inhibitory. Throughout the learning phase, short event sequences are presented sequentially at random following a uniform distribution. Throughout the learning phase, and regardless of the data type and sensor employed, we perform random (with 50% chance) spatial (i.e., horizontal and vertical) and polarity flips to the event sequences as data augmentation mechanisms. Moreover, in both the learning and inference phases, the spatial resolution of each sequence is downsampled to half its original size for computational efficiency purposes. For more information on the hyperparameters of these networks and their initialization, please refer to [101].

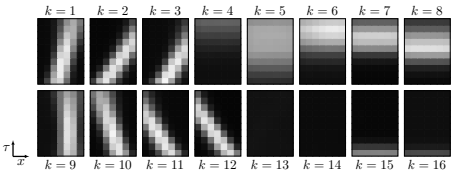
4.4.1 Synthetic data experiment

Firstly, we assess our motion-selective architecture on several noise-free sequences restricted to the pure vertical and horizontal image motion of a checkerboard pattern. These very structured texture and motion facilitate the understanding of the behavior and main properties of the network. Visual stimuli and ground truth were generated with the event camera simulator from [172], and this analysis is based on the planar optical flow formulation from Section 4.3.1.

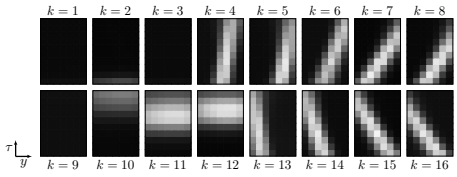
Starting with the SS-Conv layer, Fig. 4.7 shows the four convolutional kernels learned from these sequences. With this kernel scale, our learning rule leads to the successful identification of edges at the different spatial orientations present in the input data, and with the two combinations of event polarity. Using these kernels for feature extraction, and aggregating their spiking activity in the merge layer, an MS-Conv layer consisting of sixteen spatiotemporal kernels was trained thereafter. Fig. 4.8 shows the appearance of these kernels after convergence, and the response of their corresponding neural maps as a function of the ventral flow components (ω_x, ω_y).

Figure 4.7: SS-Conv kernels learned from the checkerboard texture. Weights are encoded in color brightness: green for input neurons with positive (event) polarity, and red for negative.

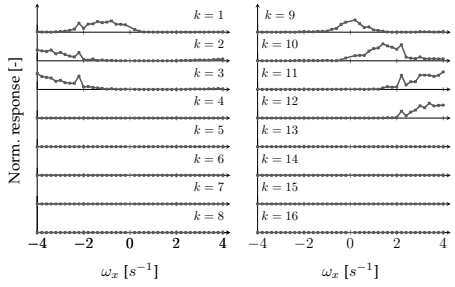
This figure confirms that, with the connectivity pattern of the MS-Conv layer, STDP leads to the successful identification of the spatiotemporally-oriented traces of input features, and hence their local motion. Out of the sixteen kernels trained, seven specialized



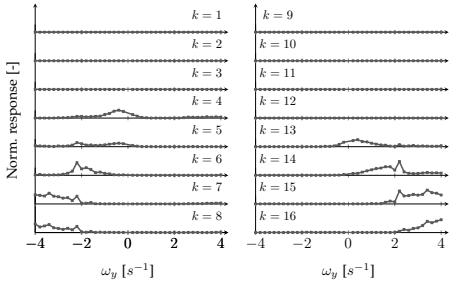
(a) x-τ representation



(b) y-τ representation

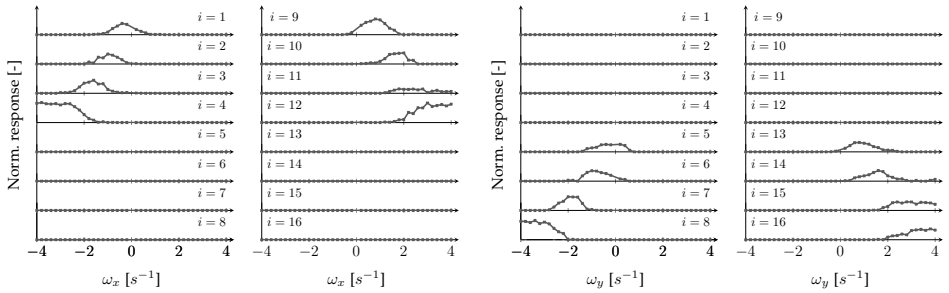


(c) Pure horizontal motion



(d) Pure vertical motion

Figure 4.8: Appearance (*top*) and neural response (*bottom*) of the sixteen spatiotemporal kernels learned from the checkerboard texture in the MS-Conv layer. Response plots are normalized by the maximum kernel response on the stimuli evaluated: 8.2763 spikes/ms by $k = 11$ for $\omega_x = 4.0\ s^{-1}$. Synaptic strength is encoded with brightness using the kernel formulation from Eq. 4.14, i.e., $W^{\text{exc}} + \beta W^{\text{inh}}$.



(a) Pure horizontal global motion

(b) Pure vertical global motion

Figure 4.9: Neural response of the sixteen individual neurons from the dense layer trained in the checkerboard texture. Response plots are normalized by the maximum neural response on the stimuli evaluated: 0.3 spikes/ms by $i = 4$ for $\omega_x = -3.8\ s^{-1}$.

to pure horizontal motion, and the remaining nine to pure vertical. Each direction of motion (up, down, left, right) was captured by at least four kernels, which, in turn, were selective to a particular stimulus speed. For instance, upward motion was identified by kernels $k = \{13, 14, 15, 16\}$, from slow to fast tuning speed. Therefore, kernels in this layer can be understood as local, velocity-tuned filters that resemble those employed in frequency-based optical flow methods [217, 219, 229, 231]. However, instead of being manually designed, these filters emerge from visual experience in an unsupervised fashion. A three-dimensional illustration of two MS-Conv kernels can be found in Fig. 4.10.

In addition, remarkable is the fact that two of the (generally) four kernels that specialized to each of the aforementioned motion directions have overlapping neural responses despite the WTA mechanism described in Section 4.3.3. This is indicative of the relatively weak speed selectivity of MS-Conv neurons in comparison to their strong direction selectivity. Section 4.4.4 confirms these results through an evaluation of both selectivities as a function of β.

Lastly, selectivity to global motion emerges in neurons from a dense layer trained as the final stage of the SNN, using the low-dimensional activity of the pooling layer. Fig. 4.9 shows the neural response (after convergence) of the sixteen cells in this layer as a function of (ω_x, ω_y). From this figure, it can be seen that neurons are successful at capturing the dominant global motion pattern from the spatial distribution of local motion estimates from previous layers. Out of the neurons trained, groups of four specialized to each motion direction, with different tuning speeds. Note that the velocity-selective properties of these neurons are exclusively dependent on those of the MS-Conv kernels. Section 4.4.4 includes an evaluation of the temporal activity of these neurons in response to speed profiles that differ from the constant-speed sequences employed for learning.

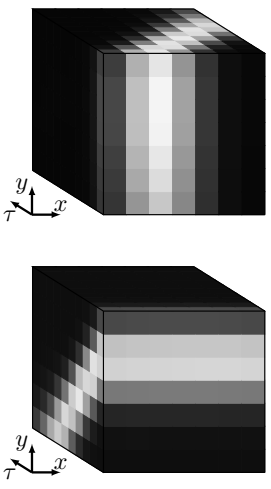
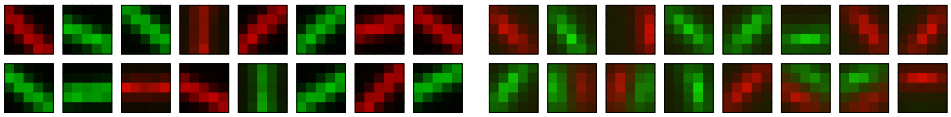


Figure 4.10: 3D illustration of two MS-Conv kernels learned from the checkerboard texture. Synaptic strength is encoded with brightness.



(a) Rotating disk (b) Roadmap

Figure 4.11: SS-Conv kernels learned from real sequences. Synaptic strength is encoded in color brightness.

4.4.2 Real data experiments

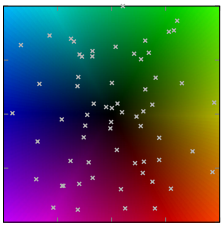
For the experiments with real data, we use samples from different sources. In a first evaluation, we employ the rotating-disk sequence from [56], which provides input events corresponding to a disk slowly turning at a constant speed. Furthermore, several unconstrained recordings of a roadmap pattern (recorded with a SEES1 [186]) are used in a second experiment characterized by more unstructured and noisy visual stimuli. For this, we also use natural scene sequences from the Event Camera Dataset [172] for validation.

Rotating-disk sequence

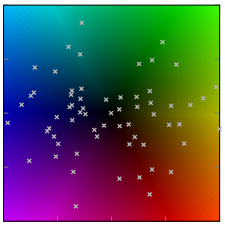
Fig. 4.11a shows the appearance of the SS-Conv kernels trained on the rotating-disk sequence. Similarly to the checkerboard case, neurons in this layer become selective to the most frequent input features, which are edges at different spatial orientations, and of different event polarity.

With respect to the MS-Conv layer of this architecture, Fig. 4.12a shows its 64 kernels in the (normalized) optical flow space. From this figure, we observe that, through our STDP rule, these MS-Conv kernels learn to identify a wide variety of optical flow vectors, including diagonal motion at different speeds. The performance of this layer in local motion perception can be assessed from the qualitative results in Fig. 4.13 (first two rows). Here, we compare the response of the network at this stage to the output of EV-FlowNet [25], which represents the state-of-the-art of conventional ANNs in event-based optical flow estimation. From these results, in both the clockwise and counterclockwise sequences, the response of the MS-Conv layer resembles that of EV-FlowNet, thus confirming the validity of our SNN in local motion perception.

Lastly, a dense layer comprised of sixteen neurons was trained, and the response of its cells is shown in Fig. 4.14. As expected, the two global motion patterns present in the data are successfully captured: half of the neurons react to clockwise rotation, and the rest to counterclockwise. Besides competition, the different response levels are due to distinct distributions of local motion estimates in the pooling layer leading to the same global motion pattern.



(a) Rotating disk



(b) Roadmap

Figure 4.12: MS-Conv kernels learned from real sequences in the (normalized) optical flow space. Motion direction is encoded in color hue, and speed in color brightness. Each kernel is depicted as a cross.

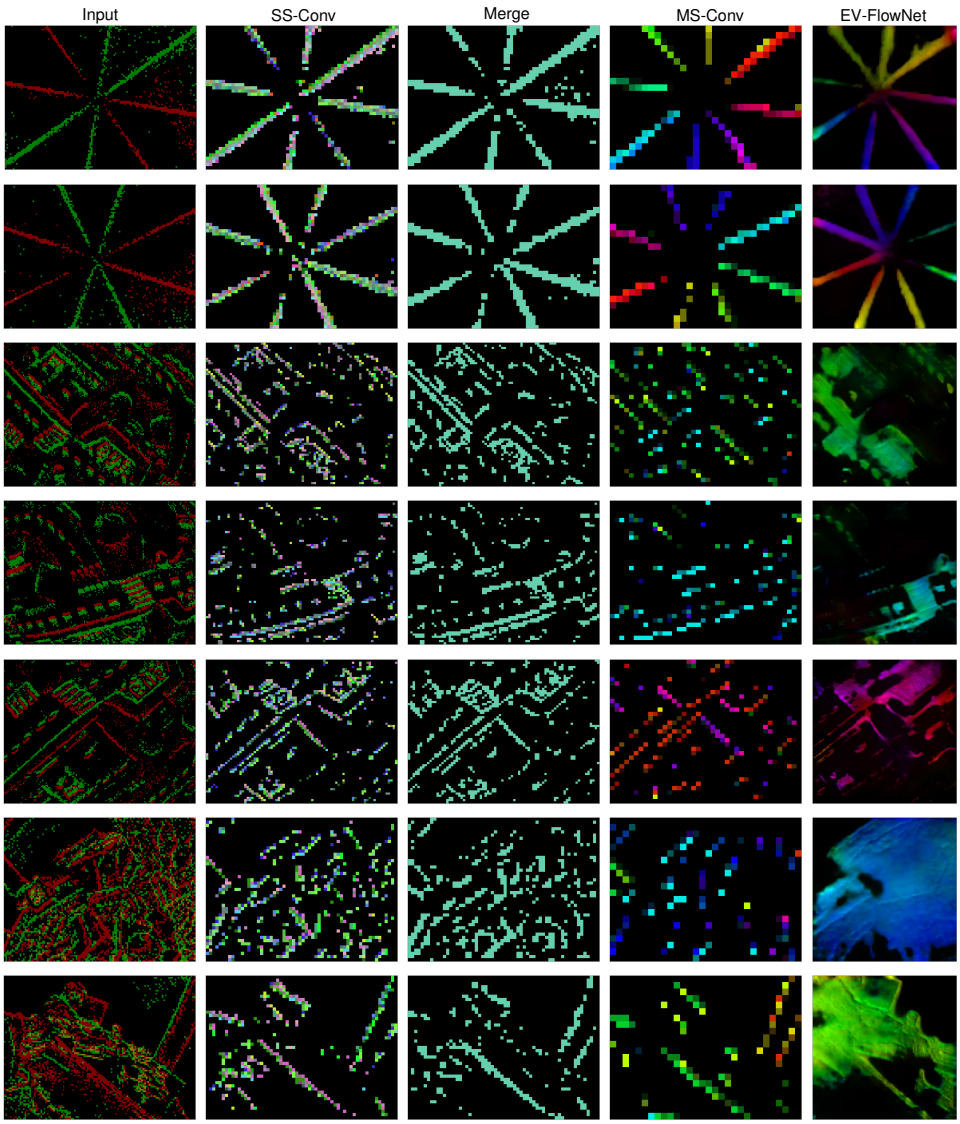


Figure 4.13: Qualitative results from the evaluation on real event sequences. From left to right, the first column corresponds to the input events, the following three to the spiking response of the SS-Conv, merge, and MS-Conv layers, respectively; and the last column to the optical flow estimation of EV-FlowNet [25]. A color is assigned to each of the kernels comprising the SS-Conv, merge, and MS-Conv layers. MS-Conv color reference shown in Fig. 4.12, and computed through a 2D histogram-matching method as in [242].

Roadmap texture and natural scenes

Fig. 4.11b shows the appearance of the SS-Conv kernels from the SNN trained on roadmap recordings. Similarly to those obtained with the rotating disk, these kernels learned edges (and combinations thereof) at several orientations, and of different polarities. However,

note that kernel appearance is significantly less smooth due to the unstructured and low-contrast features of this texture, besides the sensor noise.

Regarding the MS-Conv layer, Fig. 4.12b shows its 64 spatiotemporal kernels in the (normalized) optical flow space. In this figure, we observe that despite the wide variety of vectors learned, these are not as uniformly distributed as for the rotating-disk case. One can see that, first, horizontal motion is the most frequent local image motion type in the roadmap recordings; and second, the unsupervised nature of STDP prioritizes frequent features over others, less frequent, that may be more distant in this two-dimensional space.

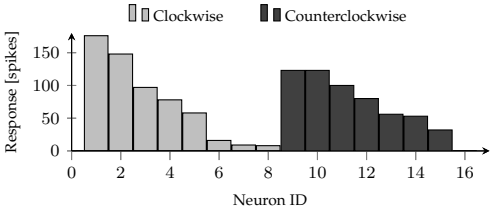
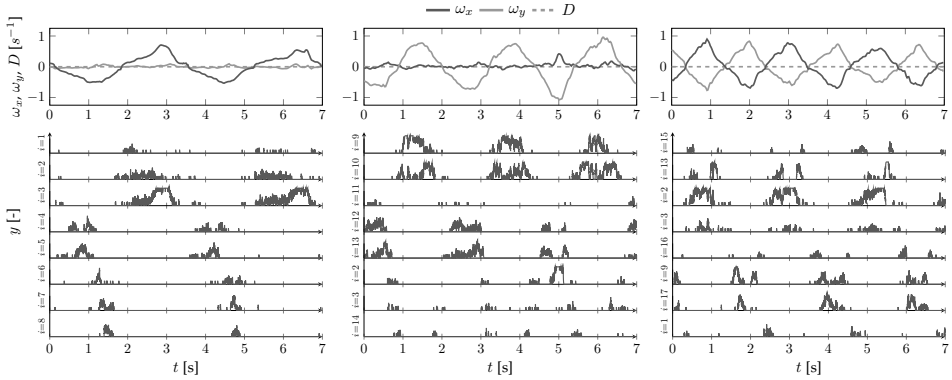


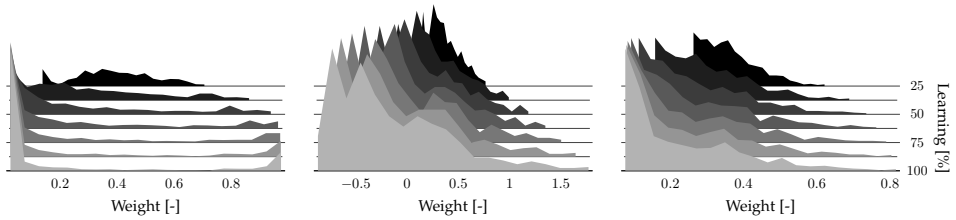
Figure 4.14: Neural activity of the dense layer trained in the rotating-disk sequence, in response to the two global motion patterns in this recording.

Qualitative results of the network performance up to this layer are shown in Fig. 4.13 for roadmap and natural scene recordings (last five rows). We draw several conclusions from these results. Firstly, the SS-Conv layer is a key component of the architecture, since it successfully filters out inconsistent local events sequences, which benefits the learning and performance of subsequent layers. Secondly, the optical flow estimation of EV-FlowNet [25] validates our MS-Conv layer, since it estimates highly similar optical flow vectors. However, there is a significant difference between the estimates of these two approaches, besides resolution (i.e., detail level). EV-FlowNet [25] performs best in high texture regions, providing a semi-dense estimate of the local motion. On the other hand, our network only provides local motion estimates whenever and wherever it discerns features whose



(a) Horizontal global motion (b) Vertical global motion (c) Diagonal global motion

Figure 4.15: Temporal course of the postsynaptic trace (as in Section 4.4.4) of the eight most-active neurons (for each case) from the dense layer learned from the roadmap texture (*bottom*), in response to different global planar motion patterns (*top*). Plots are normalized by the maximum trace on the stimuli evaluated: 1.0 by $i = 3$ at $t = 3.0$ s for the horizontal motion case. Optical flow visual observables (ω_x, ω_y, D) computed from the event sequences with the planar optical flow formulation from Section 4.3.1.



(a) Kheradpisheh *et al.* [45] (b) Shrestha *et al.* [47] (c) Ours

Figure 4.16: Evolution of the weight distribution of sixteen SS-Conv kernels throughout the learning process, using Kheradpisheh's [45], Shrestha's [47], and our STDP formulation. Results obtained with the roadmap texture, the same learning rate, and the same budget of training sequences. Each distribution is normalized by its maximum value in the learning process: 534 synapses with $W \approx 0.025$ for (a), 130 with $W \approx -0.871$ for (b), and 219 with $W \approx 0.077$ for (c); all from the 100% learning segment.

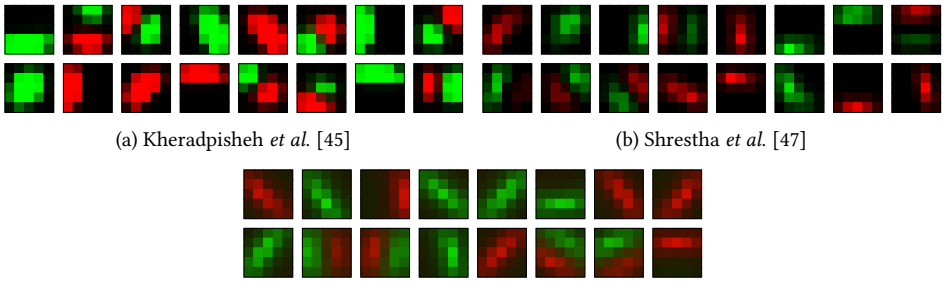
spatiotemporal trace fits one of the MS-Conv kernels. Due to trace overlap, no estimation is provided for image regions with high feature density. This limitation comes from the working principle of this layer, which takes inspiration from frequency-based methods [231] and bio-inspired motion detectors [230, 241], and for which these regions are also problematic.

Lastly, Fig. 4.15 shows the temporal activity of some of the 32 neurons comprising the dense layer of this architecture, in response to several global planar motion patterns. These results confirm the validity of this layer, and hence of the entire SNN, in becoming selective to this motion information through STDP. Moreover, similarly to the rotating-disk case, these results reinforce that, since notably different distributions of local motion estimates may correspond to the same global motion type, multiple dense neurons can specialize to the same motion pattern without overlapping responses.

4.4.3 STDP evaluation

We now evaluate several STDP formulations in the task of learning the kernels of an SS-Conv layer from the recordings of the roadmap texture. Specifically, we compare our rule, as in Section 4.3.3, to those proposed by Kheradpisheh *et al.* [45], and Shrestha *et al.* [47]; two of the most recent multiplicative formulations that have successfully been used for image classification with SNNs. Fig. 4.16 shows the weight distribution evolution of the SS-Conv kernels throughout the learning process, using each of the aforementioned formulations. Kernel appearance after learning is shown in Fig. 4.17.

The working principle of all STDP formulations is essentially the same. Whenever a neuron fires, the presynaptic connections that transferred the input spikes causing the firing are potentiated, while those that did not are depressed. The differences are in how the relevance of a connection is determined, and in how it is taken into account to compute the weight update ΔW. Both Kheradpisheh's [45] and Shrestha's [47] formulations use temporal windows of fixed length to determine whether an input spike, and so its corresponding synapse, had an influence on the postsynaptic firing. However, this information is only employed to determine whether a synapse is potentiated or depressed, and not



(a) Kheradpisheh *et al.* [45] (b) Shrestha *et al.* [47]

(c) Ours

Figure 4.17: Appearance of sixteen SS-Conv kernels after the learning process, using Kheradpisheh's [45], Shrestha's [47], and our STDP formulation. Results obtained with the roadmap texture, the same learning rate, and the same budget of training sequences. Synaptic strength is encoded in color brightness.

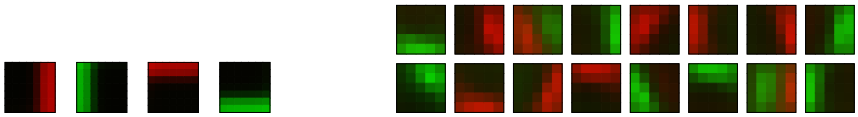
in the computation of ΔW. On the one hand, Kheradpisheh's weight update is proportional to the current weight: $\Delta W \propto W_{ijd}(1 - W_{ijd})$. Results show that this rule leads to the learning of ambiguous features that fail to capture the spatiotemporal properties of the input, since all the weights become either null or unitary (see Fig. 4.16a). On the other hand, Shrestha's rule incorporates the weight dependency in an inversely proportional manner: $\Delta W \propto e^{-W_{ijd}}$ for potentiation, and $\Delta W \propto -e^{W_{ijd}}$ for depression. As shown, even though the ΔW for potentiation (depression) diminishes as the weights increase (decrease), weights keep increasing (decreasing) throughout the learning process (see Fig. 4.16b), and hence constraints to prevent them from exploding (vanishing) are required. The use of these constraints would, in turn, result in a bimodal weight distribution similar to that of Kheradpisheh's rule, with the aforementioned drawbacks.

As explained in Section 4.3.3, and to the best of the authors' knowledge, our STDP implementation is the first multiplicative formulation in incorporating synaptic relevance in the computation of ΔW, resulting in an update rule whose LTP and LTD processes are not mutually exclusive. We combine (normalized) presynaptic trace information as a measure of synaptic relevance, with the inversely proportional weight dependency from [47]. Results, and the stability proof included in Section 4.4.4, confirm that with our novel STDP formulation, an equilibrium weight is established for each synapse, towards which the weights converge throughout the learning process (see Fig. 4.16c). Since the equilibrium state depends on synaptic relevance, the features learned are successful at capturing the spatiotemporal properties of the input.

4.4.4 Additional experiments

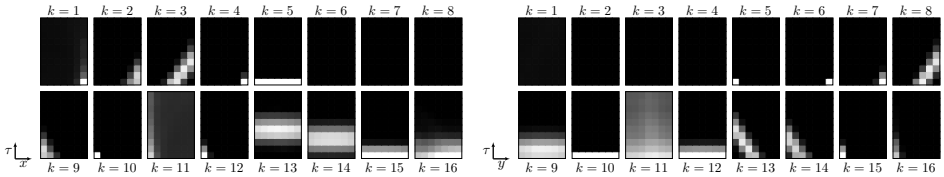
Effect of the max-based homeostasis formulation

Figs. 4.18 and 4.19 illustrate the need for the homeostasis parameter as detailed in Eqs. 4.13 and 4.14, which considers the maximum presynaptic trace of the direct spatial neighborhood $\boldsymbol{N}_{ik}$ of the neuron under analysis, when dealing with layers characterized by retinotopically-arranged cells. For a better understanding, Fig. 4.18 should be compared to Figs. 4.7 and 4.11a, and Fig. 4.19 to Figs. 4.8a and 4.8b.



(a) Checkerboard (b) Roadmap

Figure 4.18: SS-Conv kernels learned from synthetic (*left*) and real event sequences (*right*) with the neuron-specific homeostasis formulation. Synaptic strength is encoded in color brightness.



(a) x-τ representation (b) y-τ representation

Figure 4.19: MS-Conv kernels learned from the checkerboard texture with the neuron-specific homeostasis formulation. Synaptic strength is encoded with brightness.

As shown, when the neuron-specific presynaptic trace is employed instead of the full homeostasis formulation, convolutional kernels specialize to the leading edge of moving features, and hence most of these kernels are characterized by more ambiguous synaptic configurations in which the strong synapses are mainly located on the receptive field borders. The effect of using this incomplete model on the performance of the SS-Conv layer is that a greater number of kernels is required to extract the same number of spatial features. However, the impact of this formulation is more visible in the MS-Conv layer. As shown in Fig. 4.19, the vast majority of MS-Conv kernels lose their velocity-selective properties, simply because the spatiotemporally-oriented traces of input features are no longer captured. The leading-edge specialization also makes the learning process more complex, since kernel overlap increases. This, in turn, leads to some of these kernels being always prevented from firing, i.e., prevented from triggering STDP (e.g. k = 11).

Velocity selectivity of MS-Conv neurons

Fig. 4.20 shows the velocity selectivity of MS-Conv neurons as a function of β, after training with the synthetic checkerboard texture. These results confirm that, while selectivity to motion direction emerges regardless of the value of β, the inhibitory component of MS-Conv kernels is crucial for the development of speed selectivity. A more extensive sensitivity analysis of these properties can be found in [243].

Temporal response of dense neurons

Fig. 4.21 is shown to assess the temporal activity of neurons from the dense layer learned using the checkerboard texture in response to speed profiles that differ from the constant-speed sequences employed for learning. Due to the pure leftward motion of the stimuli used for this evaluation, only the activity of neurons specialized to this motion direction is shown. Neural activity is measured through the *postsynaptic trace* $y_i(t)$ of these cells,

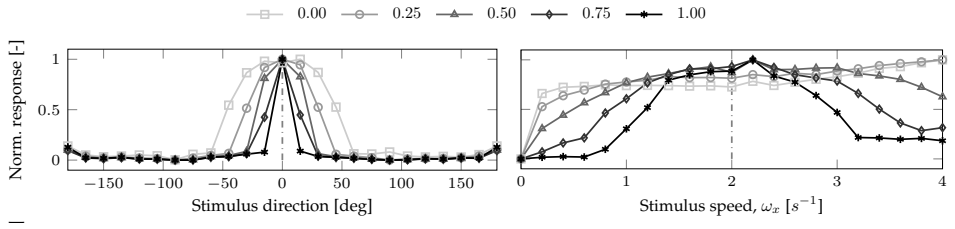


Figure 4.20: Direction and speed selectivity of neurons in the MS-Conv layer as a function of β. The dashed lines indicate the training configuration, and each response plot is normalized by its maximum value. Results obtained with the checkerboard texture. $\beta = 0$ means no inhibition, while $\beta = 1$ that inhibitory and excitatory weights contribute equally to the response of this layer.

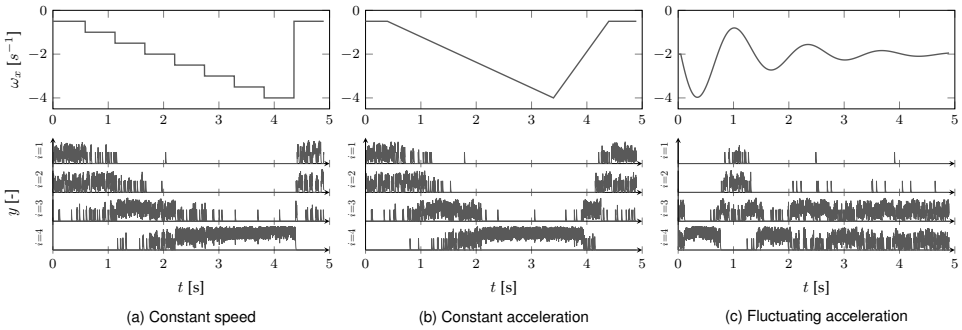


Figure 4.21: Temporal course of the postsynaptic trace of neurons $i = 1-4$ from the dense layer learned from the checkerboard texture (*bottom*, see Fig. 4.9), in response to leftward input stimuli with different speed profiles (*top*). Plots are normalized by the maximum trace on the stimuli evaluated: 0.4098 by $i = 4$ at $t = 0.36$ s for the fluctuating acceleration case.

which, similarly to Eq. 4.7, keeps track of the recent history of postsynaptic spikes emitted by a particular neuron, and is given by:

$$\alpha_y \frac{dy_i(t)}{dt} = -y_i(t) + S_i^l(t) \tag{4.16}$$

Response aspects, such as the overlap of neural activity for some ventral flow ranges, or the dominance of $i = 4$ for fast motion, are in line with the results shown in Fig. 4.9a.

Proof of stability of the proposed STDP

We use Lyapunov theorem to investigate the global stability of the STDP implementation that we propose in this work. To simplify the proof, we assume that neural connectivity is single-synaptic with no transmission delay. The STDP rule defined in Eqs. (4) and (5) can then be rewritten as:

$$\Delta W_{ij} = \eta\Big(e^{-(W_{ij}-w_{\text{init}})}\big(e^{\hat{X}_{ij}} - a\big) - e^{(W_{ij}-w_{\text{init}})}\big(e^{(1-\hat{X}_{ij})} - a\big)\Big) \tag{4.17}$$

Equilibrium weights $\bar{W}_{ij}$ are given by $\Delta W_{ij} = 0$:

$$\bar{W}_{ij} = \frac{1}{2} \ln \left(\frac{e^{\hat{X}_{ij}} - a}{e^{(1-\hat{X}_{ij})} - a} \right) + w_{\text{init}} \tag{4.18}$$

If we let $z = W_{ij} - \bar{W}_{ij}$, Eq. 4.17 becomes:

$$\begin{aligned} \Delta W_{ij} =& \eta \left(e^{\hat{X}_{ij}} - a \right)^{\frac{1}{2}} \left(e^{(1-\hat{X}_{ij})} - a \right)^{\frac{1}{2}} \left(e^{-z} - e^{z} \right) \\ \Delta W_{ij} =& A \left(e^{-z} - e^{z} \right) \end{aligned} \tag{4.19}$$

where $A(\hat{X}_{ij}, a)$ is a convenience function containing all components that are not a function of z. Then we define the positive definite energy function $V(z) = \frac{1}{2} z^2$. As such, $\dot{V}(z)$ can be solved as follows:

$$\dot{V}(z) = z\dot{z} = z \left(\Delta W_{ij} - \Delta \bar{W}_{ij} \right) \tag{4.20}$$

where $\Delta \bar{W}_{ij}$ can be computed from the time derivative of Eq. 4.18 as:

$$\Delta \bar{W}_{ij} = \frac{1}{2} \Delta \hat{X}_{ij} \left(\frac{e^{\hat{X}_{ij}}}{e^{\hat{X}_{ij}} - a} + \frac{e^{(1-\hat{X}_{ij})}}{e^{(1-\hat{X}_{ij})} - a} \right) \tag{4.21}$$

and $\Delta \hat{X}_{ij}$ can be determined using Section 4.7:

$$\Delta \hat{X}_{ij} = \frac{\lambda}{\alpha X \hat{X}_{i,m}} \left(S_j^{l-1} - \hat{X}_{ij} S_m^{l-1} \right) \tag{4.22}$$

where, here, the subscript m denotes the index of the neuron with the maximum presynaptic trace. Combining Eqs. 4.21 and 4.22 we are left with:

$$\begin{aligned} \Delta \bar{W}_{ij} =& \frac{1}{2} \frac{\lambda}{\alpha_X \hat{X}_{i,m}} \left(\frac{e^{\hat{X}_{ij}}}{e^{\hat{X}_{ij}} - a} + \frac{e^{(1-\hat{X}_{ij})}}{e^{(1-\hat{X}_{ij})} - a} \right) \left(S_j^{l-1} - \hat{X}_{ij} S_m^{l-1} \right) \\ =& B \left(S_j^{l-1} - \hat{X}_{ij} S_m^{l-1} \right) \end{aligned} \tag{4.23}$$

where $B(\hat{X}_{ij}, a)$ is a convenience expression containing all elements not a function of S_j^{l-1} and S_m^{l-1}. Now the energy derivative can be expressed simply as:

$$\dot{V}(z) = Az \left(e^{-z} - e^{z} \right) - Bz \left(S_j^{l-1} - \hat{X}_{ij} S_m^{l-1} \right) \tag{4.24}$$

Using the Taylor expansion of e^z and e^{-z}, we are left with:

$$\dot{V}(z) = -2A \left(z^2 + \frac{z^4}{3!} + \cdots \right) - Bz \left(S_j^{l-1} - \hat{X}_{ij} S_m^{l-1} \right) \tag{4.25}$$

Now if we look at the case where there is no external input to the neurons (i.e., the normalized presynaptic trace is constant, $S_j^{l-1} = S_m^{l-1} = 0$), we have that global asymptotic stability is guaranteed for $A > 0$, which can be ensured by setting $\eta > 0$ and $a < 1$.

When considering the input, we can define bounded error z with input-state stability inequality for a bounded input $u = \hat{X}_{ij} S_m^{l-1} - S_j^{l-1}$:

$$\|z(t)\| \leq \beta(\|z(t)\|, t - t_0) + \gamma \left(\sup_{\tau \geq t_0} \|u(\tau)\| \right), \forall t \geq t_0 \tag{4.26}$$

where γ is the so-called Lyapunov gain, which will lead to input-state stability if positive definite. Now, using the first order approximation for the Taylor expansion from Eq. 4.25, we can show:

$$\dot{V}(z) \leq -Az^2, \forall |z| \geq \frac{B\left(\hat{X}_{ij} S_m^{l-1} - S_j^{l-1}\right)}{2A} \tag{4.27}$$

with Lyapunov gain $\gamma = \frac{B}{2A}$.

As A must be positive for global asymptotic stability, for γ to be positive definite, B must also be positive. As such, λ_X and α must have the same sign. Additionally, the values of the constants in A and B can be used to control the bounds of the error z.

To give some physical meaning to these parameters we can see that adjusting a will change the sensitivity of the STDP update to the presynaptic trace. The larger the difference $|1 - a|$, the less sensitive the update will be to the input. The time constant λ_X will adjust the rate at which the presynaptic trace is updated from the inputs S_j^{l-1} and S_m^{l-1}. The larger the time constant the slower the presynaptic trace will change and the more bounded the error will become. The scaling factor α changes the magnitude of the presynaptic trace and therefore the magnitude of the rate of change of the presynaptic trace.

One thing to note here is the discontinuity as $X_{i,m} \longrightarrow 0$. This shows that the bound of the error can become large if the maximum presynaptic trace is small and the current neuron being updated is not the neuron with the maximum presynaptic trace. Physically, this would mean that the network cannot accurately learn when the input is infinitely sparse. For the case where the input is measurably sparse, the learning can be improved by compensating with a larger time constant λ_X.

4.5 Conclusion

In this chapter, we presented the first SNN in which selectivity to the local and global motion of the visual scene emerges through STDP from event-based stimuli. The success of this emergence depends on three contributions. First, an adaptive spiking neuron model is necessary to handle the rapidly varying input statistics of event-based sensors, and we present a novel suitable formulation for this purpose. Second, we introduce a novel STDP implementation that, contrary to the current state-of-the-art of this learning protocol, is inherently stable. Third, we propose an architecture that learns to perform a hierarchical feature extraction, effectively capturing geometric features, identifying the local motion of these features, and integrating this information into a global ego-motion estimate.

Supplementary material

Video summary of the approach: https://youtu.be/FJrba02kZII

Project code: https://github.com/tudelft/cuSNN

5

Self-Supervised Learning of Event-Based Optical Flow with SNNs

The unsupervised nature of the training framework from the previous chapter posed challenges in controlling the learned features and comparing their performance with other approaches. Therefore, in this chapter, we focus on the task of learning to estimate low-latency optical flow from event-based camera inputs in a self-supervised manner, by modifying the state-of-the-art training pipeline (for conventional artificial neural networks) to encode minimal temporal information in its inputs. Moreover, we reformulate the self-supervised loss function for event-based optical flow to improve its convexity. We perform experiments with various types of recurrent, spiking and non-spiking, architectures using the proposed pipeline. Concerning spiking networks, we investigate the effects of elements such as parameter initialization and optimization, surrogate gradient shape, and adaptive neuronal mechanisms. We find that initialization and surrogate gradient width play a crucial part in enabling learning with sparse inputs, while the inclusion of adaptivity and learnable neuronal parameters can improve performance. We show that the accuracy levels of the proposed networks are on par with the state-of-the-art at the time of publication.

The contents of this chapter have been published in:

J. J. Hagenaars†, **F. Paredes-Vallés**†, G. C. H. E. de Croon, *Self-supervised learning of event-based optical flow with spiking neural networks*, Advances in Neural Information Processing Systems (NeurIPS), 2021.

† Equal contribution, with alphabetical ordering.

Contribution: The research leading to this chapter's work was the result of a collaborative effort with ir. Jesse J. Hagenaars, from the Micro Air Vehicle Laboratory (Delft University of Technology). We both equally contributed to the conception of the study, to performing the experiments, and to the analysis and interpretation of the results. Specifically, I designed the self-supervised framework for learning low-latency, event-based optical flow, while Jesse did most of the implementation work for simulating the spiking neural networks.

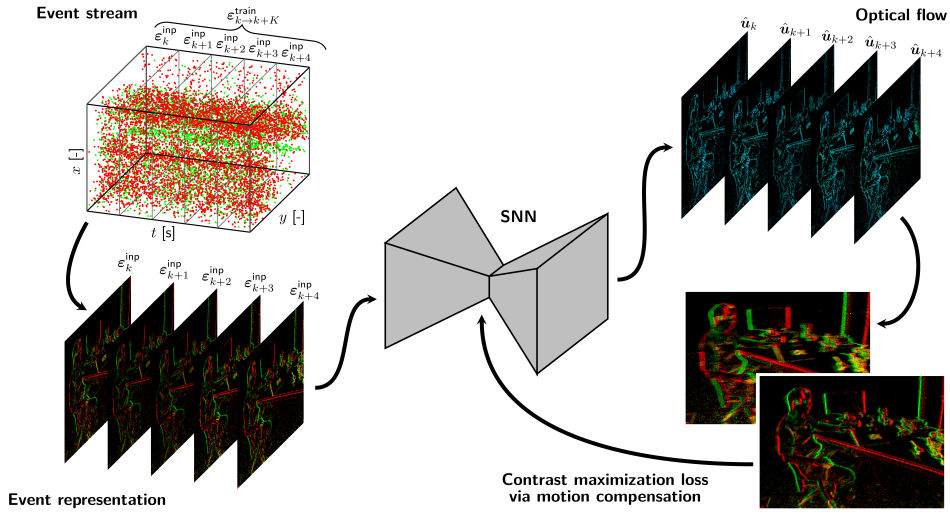


Figure 5.1: Self-supervised event-based optical flow pipeline for deep spiking neural networks. In order of processing, the event stream is split into small partitions with the same number of events, which are formatted and then fed to the network in a sequential fashion. An optical flow map is predicted for each partition, associating every input event with a motion vector. Once a sufficient number of events has been processed, we perform a backward pass using our contrast maximization loss [97].

5.1 Introduction

Neuromorphic hardware promises highly energy-efficient and low-latency sensing and processing thanks to its sparse and asynchronous nature. Event cameras capture brightness changes at microsecond resolution [18], while neuromorphic processors have demonstrated orders of magnitude lower energy consumption and latency compared to von-Neumann architectures [48, 244]. To realize the full potential of such neuromorphic pipelines, we have to move towards an event-based communication and processing paradigm, where single events are passed as-is between the event-based sensor/camera and the neuromorphic processor running a spiking neural network (SNN), without processing or accumulation of any kind in between. Because of this, all temporal integration of information needs to happen inside the network itself. Most work on employing SNNs to event-based computer vision follows this approach [39, 40], but is limited to problems of limited temporal complexity (like classification). On the other hand, most state-of-the-art artificial neural network (ANN) pipelines for event-based computer vision combine a stateless feedforward architecture with encoding temporal information in the input [25, 33].

Apart from incompatible pipelines, one of the larger impediments to widespread neuromorphic adoption is the fact that learning algorithms designed for traditional ANNs do not transfer one-to-one to SNNs, which exhibit sparse, binary activity and more complex neural dynamics. On the one hand, this has driven research into the conversion of ANNs to SNNs without loss of accuracy, but with the promised efficiency gains [35]. On the other hand, it has limited the application of directly-trained SNNs in the computer vision domain to less complicated and often discrete problems like image classification [39, 40] on constrained datasets such as N-MNIST [23] or DVS128 Gesture [24].

Still, many ongoing developments in the area of direct SNN training are promising and may form building blocks for tackling more complex tasks. Surrogate gradients [37, 38, 204, 206], which act as stand-in for the non-differentiable spiking function in the backward pass, enable traditional backpropagation with few adjustments. Similarly, the inclusion of parameters governing the neurons' internal dynamics in the optimization was demonstrated to be beneficial [40, 245]. Many works also include some form of activity regularization to keep neurons from excessive spiking [38, 246] or to balance excitability through adaptation [101, 247]. Kickstarting initial activity (hence gradient flow) is often not the goal of these regularization terms, even though [38] shows that there is a narrow activity band in which learning is optimal. This ties in with the initialization of parameters, which has not been rigorously covered for SNNs with sparse inputs yet, leaving room for improvement.

Our goal is to demonstrate the potential of neuromorphic sensing and processing on a complex task. To this end, we tackle a real-world, large-scale problem by learning, in a self-supervised fashion and using SNNs, to estimate the optical flow encoded in a continuous stream of events; a task that is usually tackled with deep, fully convolutional ANNs [25, 33]. By focusing on such a problem, we aim to identify and tackle emerging knowledge gaps regarding SNN training, while approximating a truly asynchronous pipeline.

In summary, the main contribution of this chapter is two-fold. First, we propose a novel self-supervised learning (SSL) framework for event-based optical flow estimation that puts emphasis on the networks' capacity to integrate temporal information from small, subsequent slices of events. This training pipeline, illustrated in Fig. 5.1, is built around a reformulation of the self-supervised loss function from [33] that improves its convexity. Second, through this framework, we train the first set of deep SNNs that successfully solve the problem at hand. We validate our proposals through extensive quantitative and qualitative evaluations on multiple datasets. Additionally, for the SNNs, we investigate the effects of elements such as parameter initialization and optimization, surrogate gradient shape, and adaptive neural mechanisms.

5.2 Related work

Due to the potential of event cameras to enable low-latency optical flow estimation, extensive research has been conducted on this topic since these sensors were introduced [96, 187, 217, 248]. Regarding learning-based approaches, in [25], Zhu *et al.* proposed the first convolutional ANN for this task, which was trained in an SSL fashion with the supervisory signal coming from the photometric error between subsequent grayscale frames captured with the active pixel sensor (APS) of the DAVIS240C [174]. Alongside this network, the authors released the MVSEC dataset [168], the first event camera dataset with ground-truth optical flow estimated from depth and ego-motion sensors. A similar SSL approach was introduced in [190], but here optical flow was obtained through an ANN estimating depth and camera pose. Later, Zhu *et al.* refined their pipeline and, in [33], proposed an SSL framework around the contrast maximization for motion compensation idea from [96, 97]; with which, as explained in Section 5.3.2, the supervisory signal comes directly from the events and there is no need for additional sensors. More recently, Stoffregen and Scheerlinck *et al.* showed in [150] that, if trained with synthetic event sequences (from an event camera simulator [149]) and ground-truth data in a pure supervised fashion,

the ANN from [25, 33] reaches higher accuracy levels when evaluated on MVSEC. Lastly, to hold up to the promise of high-speed optical flow, there has been a significant effort toward the miniaturization of optical flow ANNs [100, 249].

With respect to learning-based SNNs for optical flow estimation, only the works of Paredes-Vallés *et al.* [101] and Lee *et al.* [250, 251] are to be highlighted. In [101], the authors presented the first convolutional SNN in which motion selectivity emerges in an unsupervised fashion through Hebbian learning [42] and thanks to synaptic connections with multiple delays. However, this learning method limits the deployability of this architecture to event sequences with similar statistics to those used during training. On the other hand, in [250], the authors proposed a hybrid network, integrating spiking neurons in the encoder with ANN layers in the decoder, trained through the SSL pipeline from [25]. This architecture was later expanded in [251] with a secondary ANN-based encoder used to retrieve information from the APS frames. Lastly, SNNs have also been implemented in neuromorphic hardware for optical flow estimation [227], although this did not involve learning. Hence, until now, no one has yet attempted the SSL of optical flow with a pure SNN approach.

Most of the SNN work in other computer vision domains has so far been focused on discrete problems like classification [39, 40, 252, 253] and binary motion-based segmentation [254]. A notable exception is the work from Gehrig *et al.* [255], who propose a convolutional spiking encoder to continuously predict angular velocities from event data. However, until now, no one has yet attempted a dense (i.e., with per-pixel estimates) regression problem with deep SNNs that requires recurrency.

5.3 Method

5.3.1 Input event representation

An event camera consists of a pixel array that responds, in a sparse and asynchronous fashion, to changes in brightness through streams of events [18]. For an ideal camera, an event $\boldsymbol{e}_i = (\boldsymbol{x}_i, t_i, p_i)$ of polarity $p_i \in \{+, -\}$ is triggered at pixel $\boldsymbol{x}_i = (x_i, y_i)^T$ and time t_i whenever the brightness change since the last event at that pixel reaches the contrast sensitivity threshold for that polarity.

The great majority of learning-based models proposed to date for the problem of event-based optical flow estimation encode, in one form or another, spatiotemporal information into the input event representation before passing it to the neural architectures. This allows stateless (i.e., non-recurrent) ANNs to accurately estimate optical flow at the cost of having to accumulate events over relatively long time windows for their apparent motion to be perceivable. The most commonly used representations make use of multiple discretized frames of event counts [33, 100, 150, 250, 251] and/or the per-pixel average or the most recent event timestamps [25, 190, 249].

Ideally, SNNs would immediately receive spikes at event locations, which implies that temporal information should not be encoded in the input representation, but should be extracted by the network. To enforce this, we use a representation consisting only of per-pixel and per-polarity event counts, as in Fig. 5.1. This representation gets populated with consecutive, non-overlapping partitions of the event stream $\boldsymbol{\varepsilon}_k^{\text{inp}} \doteq \{\boldsymbol{e}_i\}_{i=0}^{N-1}$ (referred to as input partition) each containing a fixed number of events, N.

5.3.2 Learning optical flow via contrast maximization

We use the contrast maximization proxy loss for motion compensation [97] to learn to estimate optical flow from the continuous event stream in a self-supervised fashion. The idea behind this optimization framework is that accurate optical flow information is encoded in the spatiotemporal misalignments among the events triggered by the same portion of a moving edge (i.e., blur) and that, to retrieve it, one has to compensate for this motion (i.e., deblur the event partition). Knowing the per-pixel optical flow $\boldsymbol{u}(\boldsymbol{x}) = (u(\boldsymbol{x}), v(\boldsymbol{x}))^T$, the events can be propagated to a reference time t_{ref} through:

$$\boldsymbol{x}'_i = \boldsymbol{x}_i + (t_{\text{ref}} - t_i)\boldsymbol{u}(\boldsymbol{x}_i) \tag{5.1}$$

In this work, we reformulate the deblurring quality measure proposed by Mitrokhin *et al.* [161] and Zhu *et al.* [33]: the per-pixel and per-polarity average timestamp of the image of warped events (IWE). The lower this metric, the better the event deblurring and the more accurate the optical flow estimation. We generate an image of the average timestamp at each pixel for each polarity p' via bilinear interpolation:

$$\begin{aligned} T_{p'}(\boldsymbol{x};\boldsymbol{u}|t_{\text{ref}}) &= \frac{\sum_j \kappa(x - x'_j)\kappa(y - y'_j)t_j}{\sum_j \kappa(x - x'_j)\kappa(y - y'_j) + \epsilon} \\ \kappa(a) &= \max(0, 1 - |a|) \\ j = \{i \mid p_i = p'\}, &\qquad p' \in \{+, -\}, \qquad \epsilon \approx 0 \end{aligned} \tag{5.2}$$

Previous works minimize the sum of the squared temporal images resulting from the warping process [33, 100]. However, we scale this sum prior to the minimization with the number of pixels with at least one warped event in order for the loss function to be convex:

$$\mathcal{L}_{\text{contrast}}(t_{\text{ref}}) = \frac{\sum_{\boldsymbol{x}} T_+(\boldsymbol{x};\boldsymbol{u}|t_{\text{ref}})^2 + T_-(\boldsymbol{x};\boldsymbol{u}|t_{\text{ref}})^2}{\sum_{\boldsymbol{x}} [n(\boldsymbol{x}') > 0] + \epsilon} \tag{5.3}$$

where $n(\boldsymbol{x}')$ denotes a per-pixel event count of the IWE. As shown in Section 5.4.4, without the scaling, the loss function is not well-defined as the optimal solution is to always warp events with large timestamps out of the image space so they do not contribute to Eq. 5.2. Previous works circumvented this issue by limiting the maximum magnitude of the optical flow vectors that could be estimated through scaled TanH activations in the prediction layers [33, 100].

As in [33], we perform the warping process both in a forward ($t_{\text{ref}}^{\text{fw}}$) and in a backward fashion ($t_{\text{ref}}^{\text{bw}}$) to prevent temporal scaling issues during backpropagation. The total loss used to train our event-based optical flow networks is then given by:

$$\mathcal{L}_{\text{contrast}} = \mathcal{L}_{\text{contrast}}(t_{\text{ref}}^{\text{fw}}) + \mathcal{L}_{\text{contrast}}(t_{\text{ref}}^{\text{bw}}) \tag{5.4}$$

$$\mathcal{L}_{\text{flow}} = \mathcal{L}_{\text{contrast}} + \lambda \mathcal{L}_{\text{smooth}} \tag{5.5}$$

where λ is a scalar balancing the effect of the two losses and $\mathcal{L}_{\text{smooth}}$ is a Charbonnier smoothness prior [162], as in [25, 33]. Since $\mathcal{L}_{\text{contrast}}$ does not propagate the error back to pixels without input events, we mask the output of our networks so that null optical flow vectors are returned at these pixel locations. Furthermore, we mask the computation of

$\mathcal{L}_{\text{smooth}}$ so that this regularization mechanism only considers optical flow estimates from neighboring pixels with at least one event.

As hinted by the motion model in Eq. 5.1 and discussed in [97, 256], there has to be enough linear blur in the input event partition for $\mathcal{L}_{\text{contrast}}$ to be a robust supervisory signal. This is usually not the case in our training pipeline due to the small number of input events N that we pass to the networks at each forward pass. For this reason, we define a secondary event partition, the so-called training partition $\boldsymbol{\varepsilon}^{\text{train}}_{k\longrightarrow k+R} \doteq \{(\boldsymbol{\varepsilon}^{\text{inp}}_i, \hat{\boldsymbol{u}}_i)\}^{k+R}_{i=k}$, which is a buffer that gets populated every forward pass with an input event partition and its corresponding optical flow estimates. At training time, we perform a backward pass with the content of the buffer using backpropagation through time once it contains R successive event-flow tuples, after which we detach the state of the networks from the computational graph and clear the buffer. Note that $\mathcal{L}_{\text{smooth}}$ is also applied in the temporal dimension by smoothing optical flow estimates at the same pixel location from adjacent tuples.

5.3.3 Spiking neuron models

We compare various spiking neuron models from literature on the task of event-based optical flow estimation. All models are based on the leaky-integrate-and-fire (LIF) neuron, whose membrane potential U and synaptic input current I at timestep k can be written as:

$$U^k_i = (1 - S^{k-1}_i)\alpha U^{k-1}_i + (1 - \alpha)I^k_i \tag{5.6}$$

$$I^k_i = \sum_j W^{\text{ff}}_{ij} S^k_j + \sum_r W^{\text{rec}}_{ir} S^{k-1}_r \tag{5.7}$$

where j and r denote presynaptic neurons while i is for postsynaptic, α is the membrane decay or leak, $S \in \{0, 1\}$ a neuron spike, and W^{ff} and W^{rec} feedforward and recurrent connections, respectively. Membrane decays can either be fixed or learned. A neuron fires an output spike S if the membrane potential exceeds a threshold θ, which can either be fixed, learned, or adaptive (see below). Firing also triggers a reset of U, which is either *hard*, as in Eq. 5.6, or *soft*, as in [247]. The former is said to be more suitable for deeper networks, as it gets rid of errors accumulated by the surrogate gradient [257].

Following [247], we introduce an adaptive threshold to make up the adaptive LIF (ALIF) model. A second state variable T acts as a low-pass filter over the output spikes, adapting the firing threshold based on the neuron's activity:

$$\theta^k_i = \beta_0 + \beta_1 T^k_i \tag{5.8}$$

$$T^k_i = \eta T^{k-1}_i + (1 - \eta)S^{k-1}_i \tag{5.9}$$

where $\beta_{\{0,1\}}$ are (learnable) constants, and η is the (learnable) threshold decay/leak. ALIF's equations for U and I are identical to the LIF formulation. By decaying the threshold very slowly, T can act as a longer-term memory of the neuron [246].

Instead of postsynaptic adaptivity, we can keep a trace of presynaptic activity and use that to regularize neuron firing, giving the presynaptic LIF (PLIF) model. In [101], this adaptation is implemented by subtracting a presynaptic trace P from the input current:

$$I_i^k = \sum_j W_{ij}^{\text{ff}} S_j^k + \sum_r W_{ir}^{\text{rec}} S_r^{k-1} - \rho_0 P_i^k \tag{5.10}$$

$$P_i^k = \rho_1 P_i^{k-1} + \frac{1-\rho_1}{|R_i|} \sum_{j \in R_i} S_j^{k-1} \tag{5.11}$$

where $\rho_{\{0,1\}}$ are (learnable) addition and decay constants, and R_i is the set of receptive fields of neuron i over all channels (i.e., the second term in Eq. 5.11 is an average pooling averaged over all channels). Adaptation based on presynaptic instead of postsynaptic activity minimizes adaptation delay, making it especially suited to the fast-changing nature of event data [101]. In this spirit, we also propose the XLIF model, a crossover between ALIF and PLIF, which adapts its threshold based on presynaptic activity:

$$\theta_i^k = \beta_0 + \beta_1 P_i^k \tag{5.12}$$

As surrogate gradient for the spiking function σ, we opt for the derivative of the inverse tangent $\sigma'(x) = \text{aTan}' = 1/(1 + \gamma x^2)$ [40] because it is computationally cheap, with γ being the surrogate width and $x = U - \theta$. In order to ensure gradient flow (hence learning) in the absence of neuron firing, the width should be sufficient to cover at least a range of subthreshold membrane potentials, while the height should be properly scaled (i.e., ≤ 1) for stable learning [38]. Exact shape is of less importance for final accuracy. Further details on this can be found in Section 5.4.4.

5.3.4 Network architectures

We evaluate the two trends on neural network design for event cameras through (spiking) recurrent variants of EV-FlowNet [25] (encoder-decoder) and FireNet [148] (lightweight, no downsampling). An overview of the evaluated architectures can be found in Fig. 5.2. The use of explicit recurrent connections in all our ANNs and SNNs is justified through the ablation study in Section 5.4.4.

The base architecture referred to as EV-FlowNet is a recurrent version of the network proposed in [25]. Once represented as in Section 5.3.1, the input event partition is passed through four recurrent encoders performing strided convolution followed by ConvGRU [169] with output channels doubling after each encoder (starting from 64), two residual blocks [167, 258], and four decoder layers that perform bilinear upsampling followed by convolution. After each decoder, there is a (concatenated) skip connection from the corresponding encoder, as well as a depthwise (i.e., 1×1) convolution to produce a lower scale flow estimate, which is then concatenated with the activations of the previous decoder. The $\mathcal{L}_{\text{flow}}$ loss (see Eq. 5.5) is applied to each intermediate optical flow estimate via upsampling. All layers use 3×3 kernels and ReLU activations except for the prediction layers, which use TanH activations.

The FireNet architecture in Fig. 5.2 is an adaptation of the lightweight network proposed in [148], which was originally designed for event-based image reconstruction. However, as shown in [100], this architecture is also suitable for fast optical flow estimation. The base architecture consists of five encoder layers that perform single-strided convolution, two ConvGRUs, and a final prediction layer that performs depthwise convolution. All layers

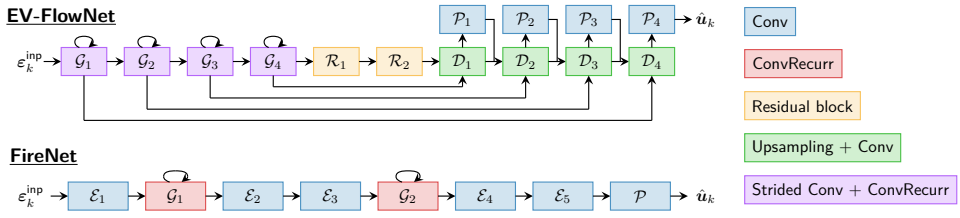


Figure 5.2: Schematic of the base neural networks used in this work. All evaluated variants inherit from these architectures and only vary the neuron model and/or the convolutional recurrent layer.

have 32 output channels and use 3 × 3 kernels and ReLU activations except for the final layer, which uses a TanH activation.

Based on these architectures, we have designed several variants: (i) RNN-EV-FlowNet and RNN-FireNet, which use vanilla ConvRNNs instead of ConvGRUs; (ii) Leaky-EV-FlowNet and Leaky-FireNet, which use ConvRNNs and whose neurons are stateful cells with leaks (for a more direct comparison with the SNNs); and (iii) SNN-EV-FlowNet and SNN-FireNet (with SNN being LIF, ALIF, PLIF or XLIF), the SNN variants that use ConvRNNs and whose neurons are spiking and stateful according to the neuron models in Section 5.3.3. The prediction layers of all SNN variants are kept real-valued with TanH activation, acting as a learned decoder from binary spikes to a dense optical flow estimate. The first layer of the SNNs can likewise be viewed as a learned spike encoder, receiving integer event counts and emitting spikes.

Clarifications on implicit and explicit recurrency

The definition of the implicit temporal dynamics of the leaky, non-spiking variants of our EV-FlowNet and FireNet base architectures closely resembles that of the membrane potential for spiking neurons (see Eq. 5.6) but without the reset mechanism. With ReLU as non-linearity, the activation Y of a leaky neuron is given by:

$$Y_i^k = \mathrm{ReLU}\left(\alpha Y_i^{k-1} + (1-\alpha)\sum_j W_{ij}^{\mathrm{ff}} I_i^k\right) \tag{5.13}$$

where j and i denote presynaptic and postsynaptic neurons respectively, α is the decay or leak of the neuron, k the timestep, and W^{ff} the feedforward weights multiplying the input signal I.

Regarding explicit recurrency, there is a slight difference between the vanilla ConvRNN layers used in our SNN and ANN architectures. On the one hand, the ConvRNNs that we use in our SNNs are defined through Eqs. 5.6 and 5.7 with two convolutional gates, one for the input and one for the recurrent signal, followed by the spiking function. On the other hand, the ConvRNNs in our ANNs are characterized by the same two convolutional gates but in this case followed by a TanH activation, and thereafter by a third output gate with ReLU activation. This augmentation was introduced to improve the convergence of the RNN and leaky variants of the base ANN architectures.

5.4 Experiments

To highlight the robustness of our SSL pipeline, we train our networks on the indoor forward-facing sequences from the UZH-FPV Drone Racing Dataset [171], which is characterized by a much wider distribution of optical flow vectors than the datasets that we use for evaluation, i.e., MVSEC [25], High Quality Frames (HQF) [150], and the Event-Camera Dataset (ECD) [172]. The selected training sequences consist of approximately 15 minutes of event data that we split into 140 128 × 128 (randomly cropped) sequences with 500k events each. We further augment this data using random horizontal, vertical, and polarity flips.

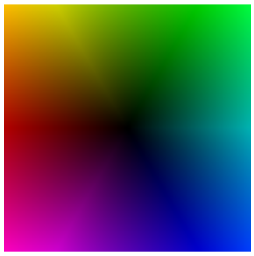


Figure 5.3: Optical flow color-coding scheme. Direction is encoded in color hue, and speed in color brightness.

Our framework is implemented in PyTorch. We use the Adam optimizer [173] and a learning rate of 0.0002, and train with a batch size of 8 for 100 epochs. We clip gradients based on a global norm of 100. We fix the number of events for each input partition to $N = 1\text{k}$, while we use 10k events for each training event partition. This is equivalent to $K = 10$ forward passes per backward pass (i.e., the network's unrolling), as described in Section 5.3.2 and illustrated in Fig. 5.1. Lastly, we empirically set the scaling weight for $\mathcal{L}_{\text{smooth}}$ to $\lambda = 0.001$.

We evaluated our architectures on the MVSEC dataset [168] with the ground-truth optical flow data provided by Zhu *et al.* in [25], which was generated at each APS frame timestamp, and scaled to be the displacement for the duration of one (dt = 1) and four (dt = 4) APS frames. Optical flow predictions were also generated at each frame timestamp by using all the events in the time window as input for dt = 1, or 25% of the window events at a time for dt = 4 (due to the larger displacements). For comparison against the ground truth, the predicted optical flow is converted from units of pixels/partition to units of pixel displacement by multiplying it with $\text{dt}_{\text{gt}}/\text{dt}_{\text{input}}$. We compare our recurrent ANNs and SNNs against the state-of-the-art on self-supervised event-based optical flow estimation: the original (non-recurrent) EV-FlowNet [25] trained with either photometric error as in [25] or contrast maximization [33], and the hybrid SNN-ANN network from [250]. Quantitative results of this evaluation are presented in Table 5.1. We report the average endpoint error (EPE↓, lower is better) and the percentage of points with EPE greater than 3 pixels and 5% of the magnitude of the optical flow vector, denoted by $\%_{\text{Outlier}}$, over pixels with valid ground-truth data and at least one input event. Qualitative results of our best performing networks on this dataset are shown in Fig. 5.4.

For the sake of completeness, as in [100, 150] we also evaluate our architectures on the ECD [172] and HQF [150] datasets using metrics derived from the contrast maximization framework [96, 97]. The details and results of this evaluation can be found in Section 5.4.4.

Regarding the SNN variants, the results in Table 5.1, Fig. 5.4 and Section 5.4.4 correspond to networks whose neural parameters (i.e., leaks, thresholds, adaptive mechanisms) were also optimized when applicable, unless specified. See Section 5.4.4 for an ablation study on the learnable parameters.

5.4.1 Evaluation of the ANN and SNN architectures

Firstly, the quantitative results in Table 5.1 confirm the validity of the proposed SSL framework for event-based optical flow estimation with recurrent networks. As shown, our

dt = 1	outdoor_day1		indoor_flying1		indoor_flying2		indoor_flying3	
	EPE↓	%Outlier↓	EPE↓	%Outlier↓	EPE↓	%Outlier↓	EPE↓	%Outlier↓
EV-FlowNet* [25]	0.49	0.20	1.03	2.20	1.72	15.10	1.53	11.90
EV-FlowNet* [33]	**0.32**	**0.00**	**0.58**	**0.00**	**1.02**	**4.00**	**0.87**	**3.00**
Hybrid-EV-FlowNet* [250]	0.49	-	0.84	-	1.28	-	1.11	-
EV-FlowNet	0.47	0.25	0.60	0.51	1.17	8.06	0.93	5.64
RNN-EV-FlowNet	0.56	1.09	0.62	0.97	1.20	8.82	0.93	5.51
Leaky-EV-FlowNet	0.53	0.28	0.71	0.60	1.43	11.37	1.14	8.12
LIF-EV-FlowNet	0.53	0.33	0.71	1.41	1.44	12.75	1.16	9.11
ALIF-EV-FlowNet	0.57	0.42	1.00	2.46	1.78	17.69	1.55	15.24
PLIF-EV-FlowNet	0.60	0.52	0.75	0.85	1.52	13.38	1.23	9.48
XLIF-EV-FlowNet	0.45	0.16	0.73	0.92	1.45	12.18	1.17	8.35
FireNet	0.55	0.35	0.89	1.93	1.62	14.65	1.35	10.64
RNN-FireNet	0.62	0.52	0.96	2.60	1.77	17.55	1.48	13.60
Leaky-FireNet	0.52	0.41	0.90	2.66	1.67	16.09	1.43	13.16
LIF-FireNet	0.57	0.40	0.98	2.48	1.77	16.40	1.50	12.81
ALIF-FireNet	0.62	0.45	1.04	3.02	1.85	18.88	1.58	15.00
PLIF-FireNet	0.56	0.38	0.90	1.93	1.67	14.47	1.41	11.17
XLIF-FireNet	0.54	0.34	0.98	2.75	1.82	18.19	1.54	14.57
dt = 4								
EV-FlowNet* [25]	1.23	**7.30**	2.25	24.70	4.05	45.30	3.45	39.70
EV-FlowNet* [33]	1.30	9.70	2.18	24.20	3.85	46.80	3.18	47.80
Hybrid-EV-FlowNet* [250]	**1.09**	-	2.24	-	**3.83**	-	3.18	-
EV-FlowNet	1.69	12.50	**2.16**	**21.51**	3.90	**40.72**	**3.00**	**29.60**
RNN-EV-FlowNet	1.91	16.39	2.23	22.10	4.01	41.74	3.07	30.87
Leaky-EV-FlowNet	1.99	17.86	2.59	30.71	4.94	54.74	3.84	42.33
LIF-EV-FlowNet	2.02	18.91	2.63	29.55	4.93	51.10	3.88	41.49
ALIF-EV-FlowNet	2.13	20.96	3.81	50.36	6.40	66.03	5.53	61.07
PLIF-EV-FlowNet	2.24	23.76	2.80	34.34	5.21	52.98	4.12	45.31
XLIF-EV-FlowNet	1.67	12.69	2.72	31.69	4.93	51.36	3.91	42.52
FireNet	2.04	20.93	3.35	42.50	5.71	61.03	4.68	53.42
RNN-FireNet	2.35	24.31	3.64	46.54	6.33	63.89	5.20	56.60
Leaky-FireNet	1.96	18.26	3.42	42.03	5.92	58.80	4.98	52.57
LIF-FireNet	2.12	21.00	3.72	48.27	6.27	64.16	5.23	58.43
ALIF-FireNet	2.36	25.82	3.94	52.35	6.65	67.61	5.60	61.93
PLIF-FireNet	2.11	20.64	3.44	44.02	5.94	64.02	4.98	57.53
XLIF-FireNet	2.07	18.83	3.73	47.89	6.51	67.25	5.43	60.59

*Non-recurrent ANNs with input event representations encoding spatiotemporal information, as described in [25, 33, 250].

Table 5.1: Quantitative evaluation on MVSEC [168]. Best in bold, runner up underlined.

base architectures EV-FlowNet and FireNet perform on par with the current state-of-the-art, even though these non-recurrent networks from literature encode explicit temporal information in their input event representations, and were trained on other very similar sequences from MVSEC [168] to prevent the input statistics from deviating from the training distribution during inference [25, 33, 250]. Since we train on a very different dataset [171], this on-par performance also confirms the generalizability of our ANNs and SNNs to distinctly different scenes and distributions of optical flow vectors. This claim is further supported by qualitative results in Fig. 5.4 and additional results in Section 5.4.4.

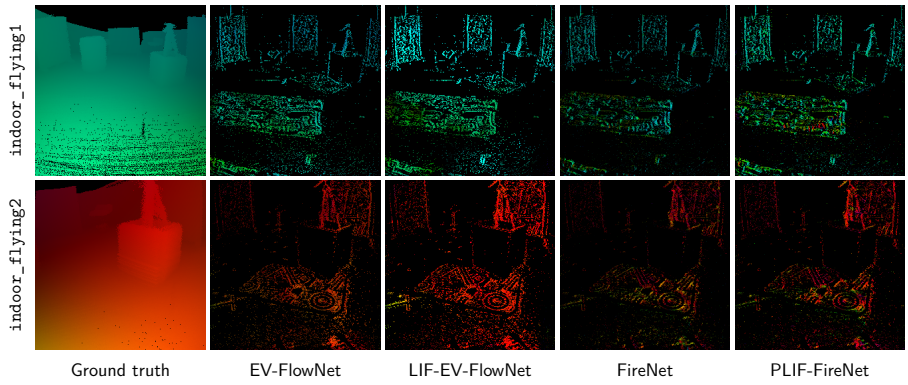


Figure 5.4: Qualitative evaluation of our best performing ANNs and SNNs on sequences from the MVSEC dataset [168]. The optical flow color-coding scheme can be found in Fig. 5.3.

Secondly, from the comparison between our base ANN architectures and their spiking counterparts without adaptation mechanisms (i.e., LIF-EV-FlowNet and LIF-FireNet), we can conclude that, although there is a general increase in the EPE and the percentage of outliers when going spiking, the proposed SNNs are still able to produce high quality event-based optical flow estimates. In fact, according to Table 5.1, the main drop in accuracy does not come from the incorporation of the spiking function (and the selection of aTan′ as surrogate gradient), but mainly from the use of vanilla convolutional recurrent layers instead of gated recurrent units. As shown, our spiking LIF architectures perform very close to their RNN and leaky counterparts, despite the latter being ANNs. This highlights the important need for more powerful convolutional recurrent units for SNNs, similar to ConvLSTMs [170] and ConvGRUs [169] for ANNs, as this would narrow the performance gap between these two processing modalities according to our observations. Interestingly, a previous comparison of the performance of recurrent ANNs and SNNs for event-based classification [259] suggested similar improvements to SNN units.

5.4.2 Impact of adaptive mechanisms for spiking neurons

Table 5.1 also allows us to draw conclusions about the effectiveness of the adaptive mechanisms for spiking neurons introduced in Section 5.3.3. For both EV-FlowNet and FireNet, we observe that threshold adaptation based on postsynaptic activity (i.e., the ALIF model) performs worse compared to other models. While the ALIF model was shown to be effective for learning long temporal dependencies from relatively low-dimensional data as in [246, 247, 260], the adaptation delay introduced by relying on a postsynaptic signal seems detrimental when working with fast-changing, high-dimensional event data. This is in line with suggestions by Paredes-Vallés *et al.* in [101], who use presynaptic adaptation for this reason. Our own results with presynaptic adaptation (i.e., PLIF and XLIF models) are somewhat inconclusive. While PLIF performs better in the case of FireNet, this is not the case for EV-FlowNet. On the other hand, XLIF's performance is very similar to the LIF model for both FireNet and EV-FlowNet architectures. Based on these observations, we think that adaptivity based on presynaptic activity should be considered for further

development. In this regard, the XLIF model has the advantage that it is able to generate activity (leading to gradient flow, and thus learning) even for very small inputs, whereas PLIF is incapable of this for a given threshold (because P is always positive). A more detailed comparison of activity levels for the different variants is given in Section 5.4.4, along with an approximation of the energy efficiency gains of SNNs compared to ANNs.

5.4.3 Further lessons on training deep SNNs

Multiple problems arise when training deep SNNs for a regression task that involves sparse data, as is done here. Regarding learning, we find that gradient vanishing poses the main issue. Even considering dense inputs/loss and a shallow (in timesteps or in layers) SNN, sufficient gradient flow is a result of wide enough (in our case, covering at least $|U - \theta| \leq \theta$) and properly scaled (i.e., ≤ 1) surrogate gradients [38, 257, 260], and parameter initializations that lead to non-negligible amounts of spiking activity [38]. Sparse data and deep networks make finding the proper settings more difficult, and for this reason, we have tried to increase the robustness of various of these hyperparameter settings. First, we looked at the learning performance and gradient flow of networks with various surrogate gradient shapes and widths. Compared to the aTan′ surrogate specified in Section 5.3.3, SuperSpike [261] with $\gamma = 10$ and $\gamma = 100$ (both narrower) show little learning due to negligible gradient flow (see Section 5.4.4 for more details). One way of reducing the effect of a too narrow surrogate gradient would be to trigger spiking activity through regularization terms in the loss function, as done in, e.g., [38, 246]. These form a direct connection between loss and the neuron in question, bypassing most of the gradient vanishing that would happen in later layers. We tried the variant proposed in [38], which is aimed at achieving at least a certain fraction of neurons to be active at any given time. With this fraction set to 5%, we saw that for SuperSpike with $\gamma = 10$ there was some learning happening, while for $\gamma = 100$ there was no effect. Plots of the loss curves and gradient magnitudes are available in Section 5.4.4. Of course, more research into these and other regularization methods is necessary. Alternatively, as done in [257], batch normalization (or other presynaptic normalization mechanisms) could be used to ensure proper activity and gradient flow.

Regarding the network output, there seems to be an intuitive gap between classification and regression tasks, with the latter requiring a higher resolution to be solved successfully. In our view, there are two aspects to this that might pose an issue to SNNs. First, given the single prediction layer that the here-presented SNNs have to go from binary to real-valued activations, one could expect a loss in output resolution compared to equivalent ANN architectures. Second, even for moderate activity levels, the outputs of a spiking layer can be much larger in magnitude than an equivalent ANN layer, even for comparable parameter initializations. Intuitive solutions to these shortcomings are (i) to increase the number of channels to increase the resolution, and (ii) to initialize the weights of the non-spiking prediction layer as to have a smaller magnitude. While increasing the number of output channels in $\mathcal{E}_5$ (LIF-FireNet, see Fig. 5.2) did not lead to significantly improved performance or learning speed, decreasing the initialization magnitude of the weights in layer $\mathcal{P}$ did.

5.4.4 Additional experiments

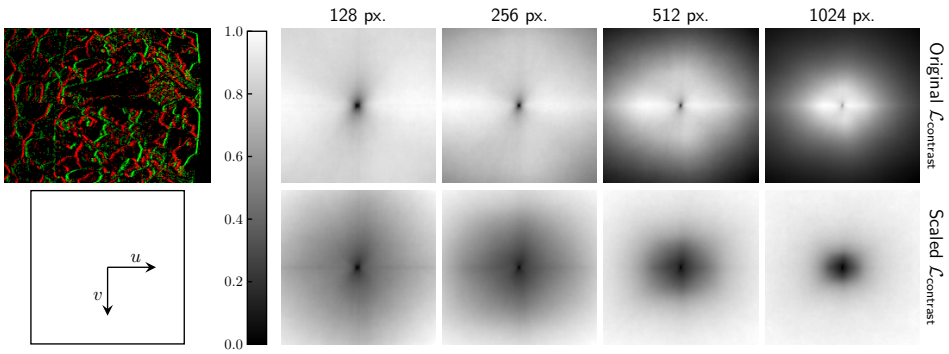
Convexity of the self-supervised loss function

To evaluate the convexity of the self-supervised loss function for event-based optical flow estimation from [33] and the adaptation that we propose in this work, we conducted an experiment with two partitions of 40k events from the ECD dataset [172]. In this experiment, for the selected partitions, we computed the value of Eq. 5.4 (with and without the scaling) for four sets of optical flow vectors given by:

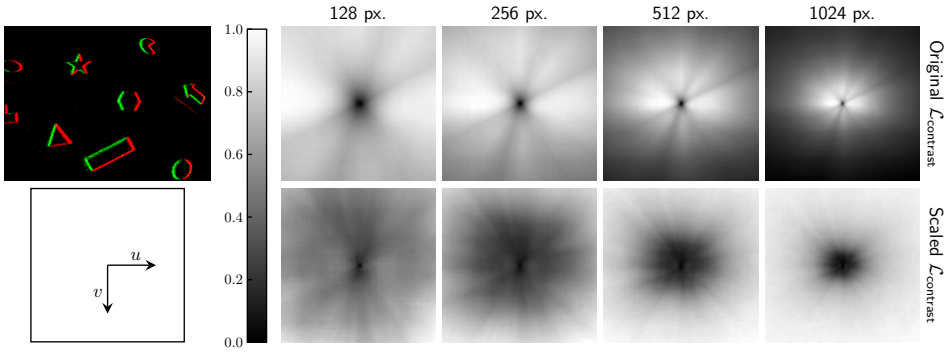
$$\boldsymbol{u}_s(d) = \{(u : u = g(i,d), v : v = g(j,d)), i \in \{0,1,...,128\}, j \in \{0,1,...,128\}\} \tag{5.14}$$

$$g(x,d) = \frac{2xd}{128} - d \tag{5.15}$$

where d denotes the per-axis maximum displacement, which is drawn from the set $D = \{128, 256, 512, 1024\}$. This is equivalent to performing a grid search for the lowest $\mathcal{L}_{\text{contrast}}$ over an optical flow space ranging from $(-d, -d)$ to (d, d) with 128 samples for each axis. Fig. 5.5 highlights the main difference between the original and our adapted



(a) Sequence: poster_6dof.



(b) Sequence: shapes_6dof.

Figure 5.5: Effect of scaling $\mathcal{L}_{\text{contrast}}$ for different optical flow vectors for two event partitions from the Event-Camera Dataset [172]. Numbers on top indicate the maximum per-axis pixel displacement for each column.

formulation. Although for the smaller values of d the two normalized losses look qualitatively similar, for larger values it is possible to discern that the original $\mathcal{L}_{\text{contrast}}$ is not convex, and that its optimal solution is to throw events out of the image space during the warping process so they do not contribute to the computation of the loss. On the contrary, the scaling that we propose in Section 5.3.2 fixes this issue, and results in a convex loss function for any value of d.

Self-supervised evaluation and additional qualitative results

Apart from the quantitative and qualitative evaluation on the MVSEC dataset [168] included in Section 5.4, we also evaluate our architectures on the ECD [172] and HQD [150] datasets, as in [100, 150]. Since these datasets lack ground-truth data, we use the Flow Warp Loss (FWL↑, higher is better) [150], which measures the sharpness of the IWE relative to that of the original event partition using the variance as a measure of the contrast of the event images [97]. In addition to FWL, we propose the Ratio of the Squared Average Timestamps (RSAT↓, lower is better) as a novel, alternative metric to measure the quality of the optical flow without ground-truth data. Contrary to FWL, RSAT makes use of Eq. 5.3 to measure the contrast of the event images and is defined as:

$$\text{RSAT} \doteq \frac{\mathcal{L}_{\text{contrast}}(t_{\text{ref}}^{\text{fw}}|\boldsymbol{u})}{\mathcal{L}_{\text{contrast}}(t_{\text{ref}}^{\text{fw}}|\mathbf{0})} \tag{5.16}$$

where RSAT<1 implies that the predicted optical flow is better than a baseline consisting of null vectors. Since both metrics are sensitive to the number of input events [101], we set N=15k events for all sequences in this evaluation. Quantitative results of this evaluation can be found in Table 5.2, while qualitative results on these datasets are shown in Fig. 5.6.

	ECD		HQF	
	FWL↑	RSAT↓	FWL↑	RSAT↓
EV-FlowNet	1.31	**0.94**	1.37	**0.92**
RNN-EV-FlowNet	1.36	<u>0.95</u>	1.45	<u>0.93</u>
Leaky-EV-FlowNet	1.34	<u>0.95</u>	1.39	<u>0.93</u>
LIF-EV-FlowNet	1.21	<u>0.95</u>	1.24	0.94
ALIF-EV-FlowNet	1.17	0.98	1.21	0.98
PLIF-EV-FlowNet	1.24	<u>0.95</u>	1.28	<u>0.93</u>
XLIF-EV-FlowNet	1.23	<u>0.95</u>	1.25	<u>0.93</u>
FireNet	**1.43**	0.99	**1.57**	0.99
RNN-FireNet	1.34	0.99	1.42	0.99
Leaky-FireNet	<u>1.40</u>	0.99	1.52	0.99
LIF-FireNet	1.28	0.99	1.34	1.00
ALIF-FireNet	1.35	1.00	<u>1.49</u>	1.00
PLIF-FireNet	1.30	0.97	1.35	0.98
XLIF-FireNet	1.29	0.99	1.39	0.99

Table 5.2: Quantitative evaluation on the ECD [172] and HQF [150] datasets. Best in bold, runner up underlined.

Apart from further confirming the generalizability of our architectures to other datasets and the on-par performance of our SNNs with respect to the recurrent ANNs (and thus to the state-of-the-art), results from this evaluation reveal the lack of robustness of the self-supervised FWL metric from Stoffregen and Scheerlinck *et al.* [150] in capturing the quality of the learned event-based optical flow. As shown in Table 5.2, FWL results do not correlate with the EPEs reported in Table 5.1. For instance, FireNet variants are characterized by higher values (thus better, according to [150]) than their computationally more powerful EV-FlowNet counterparts overall, while, according to Table 5.1, it should be the opposite. On the other hand, according to its correlation with the reported EPEs in Table 5.1, RSAT, which is

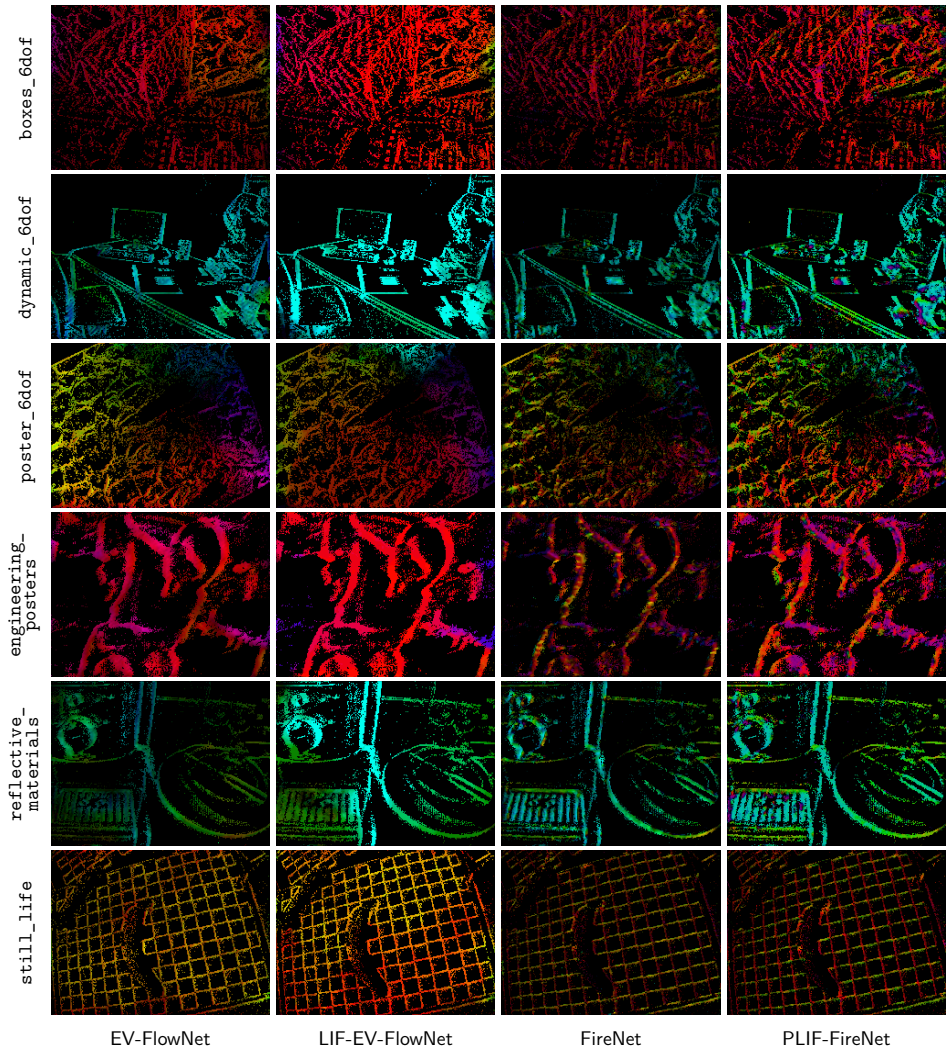


Figure 5.6: Additional qualitative results of our best performing ANNs and SNNs on sequences from the ECD [172] (top three) and HQF [150] (bottom three) datasets. The optical flow color-coding scheme can be found in Fig. 5.3.

based on our reformulation of the self-supervised loss function from [33], is a more reliable metric to assess the quality of event-based optical flow without ground-truth data.

Ablation study on recurrent connections

In this ablation study, we evaluate the importance of explicit recurrent connections for event-based optical flow estimation with ANNs and SNNs when using our input event representation (see Section 5.3.1) and training settings (see Section 5.4). To do this, we use

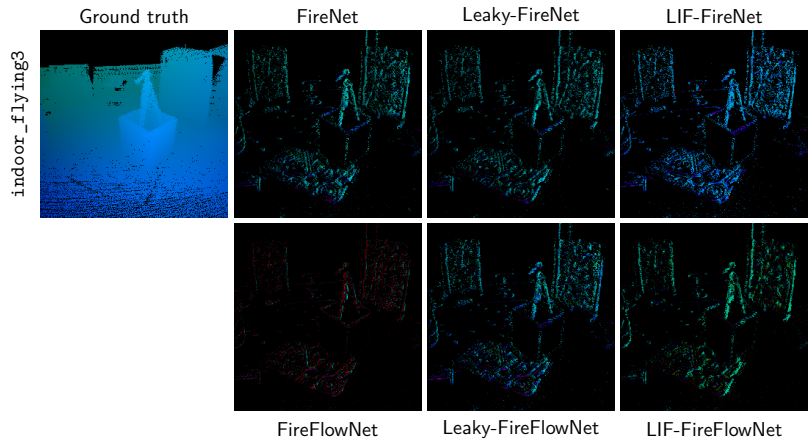


Figure 5.7: Qualitative results of FireNet and FireFlowNet variants on a sequence from MVSEC [168]. The optical flow color-coding scheme can be found in Fig. 5.3.

the base FireNet architecture and its leaky and LIF variants (as introduced Section 5.3.4), and compare their performance on MVSEC [168] to their non-recurrent counterparts. As in [100], the non-recurrent version of FireNet that we use, which substitutes the ConvGRUs with convolutional encoders, is further referred to as FireFlowNet. The qualitative and quantitative results for this ablation study are shown in Fig. 5.7 and Table 5.3, respectively.

Firstly, from these results, we can conclude that stateless ANNs (such as FireFlowNet) are not capable of learning to estimate optical flow using our input event representation and training pipeline. This observation confirms the claim made in Section 5.3.1 about the fact that our event representation minimizes the amount of temporal information encoded in the input to the networks. Secondly, these results also confirm that, in order to successfully learn optical flow, the networks need to be able to build an internal (hidden) state through explicit recurrent connections and/or neural dynamics. As shown, the only architecture that is not able to learn optical flow is FireFlowNet. If this network is augmented with recurrent connections (i.e., FireNet), neural dynamics (i.e., Leaky-FireFlowNet, LIF-FireFlowNet), or both (i.e., Leaky-FireNet, LIF-FireNet), optical flow can be learned with our proposed pipeline and event representation. However, from the quantitative results in Table 5.3, we can observe that learning optical flow through neural dynamics without explicit recurrent connections (i.e., Leaky-FireFlowNet, LIF-FireFlowNet), although possible, is quite complex and results in networks with lower accuracy. For this reason, we conclude that recurrent connections are an important driver for learning accurate event-based optical flow with our training pipeline, and hence, we use them in the ANNs and SNNs that we propose in this work.

Ablation study on learnable parameters for SNNs

Several works emphasize the importance of including neural parameters in the optimization [40, 245, 260, 262], agreeing that including the various decays or leaks is beneficial for performance. Some also argue and show that learning thresholds adds little value [40, 245],

dt = 1	outdoor_day1		indoor_flying1		indoor_flying2		indoor_flying3	
	EPE↓	%Outlier↓	EPE↓	%Outlier↓	EPE↓	%Outlier↓	EPE↓	%Outlier↓
FireNet	0.55	**0.35**	**0.89**	**1.93**	**1.62**	**14.65**	**1.35**	**10.64**
Leaky-FireNet	**0.52**	0.41	0.90	2.66	1.67	16.09	1.43	13.16
LIF-FireNet	0.57	0.40	0.98	2.48	1.77	16.40	1.50	12.81
FireFlowNet	1.02	1.62	1.37	6.86	2.24	25.74	2.00	21.09
Leaky-FireFlowNet	0.61	0.56	0.97	2.71	1.76	17.68	1.52	14.16
LIF-FireFlowNet	0.84	1.15	1.22	5.55	2.06	22.25	1.80	18.13
dt = 4								
FireNet	2.04	20.93	**3.35**	42.50	**5.71**	61.03	**4.68**	53.42
Leaky-FireNet	**1.96**	**18.26**	3.42	**42.03**	5.92	**58.80**	4.98	**52.57**
LIF-FireNet	2.12	21.00	3.72	48.27	6.27	64.16	5.23	58.43
FireFlowNet	3.88	55.47	5.29	68.37	8.26	79.42	7.33	78.69
Leaky-FireFlowNet	2.29	24.22	3.68	47.12	6.29	62.30	5.37	58.29
LIF-FireFlowNet	3.24	43.08	4.67	60.34	7.54	74.68	6.54	71.45

Table 5.3: Quantitative evaluation of FireNet and FireFlowNet variants on MVSEC [168]. Best in bold, runner up underlined.

which makes intuitive sense given that the same effect can be achieved through scaling the synaptic weights. To confirm these observations, we perform an ablation study on the learning of per-channel leaks and thresholds for LIF-FireNet. All instances of a parameter are initialized to the same value, but can be adapted over time in the case of learning. The results in Table 5.4 suggest that, for our task, learning at least the leaks is beneficial for performance. However, despite these differences in EPE, the training loss curves for all variants, as shown in Fig. 5.8, do not vary a lot. For this we can follow the same explanation as in the ablation study on recurrent connections: without the optimization of leaks, the network could focus on decreasing $\mathcal{L}_{\text{smooth}}$, which does not lead (as much) to the actual learning of optical flow.

Looking at the learned leaks also gives us insight into how information is integrated throughout the network. Fig. 5.9 shows the distribution of the parameter a, from which the membrane potential leaks are computed as $\alpha = \frac{1}{1+\exp(-a)}$, for the LIF-FireNet variants with learnable leaks. Initially, all $a = -4$; after learning, earlier layers mostly end up with faster leaks (lower a), while later layers end up with slower leaks (higher a). This intuitively makes sense: we want earlier layers to respond quickly to changing inputs, while we need later layers to (more slowly) integrate information over time and produce an optical flow estimate.

Details on further lessons

We looked at the effect of surrogate gradient width on learning performance by trying out three variants on LIF-FireNet: aTan′ with $\gamma = 10$ (default), and SuperSpike [261] with $\gamma \in \{10, 100\}$. The resulting loss curves are plotted in Fig. 5.11. The width of aTan′-10 is such that there is sufficient gradient flow for learning; this is less so for SuperSpike-10, and not at all for SuperSpike-100. The plots of per-layer mean gradient magnitude in Fig. 5.10 confirm this: SuperSpike-10 only shows non-negligible gradient flow for the last two layers, while the mean gradients for SuperSpike-100 are practically zero.

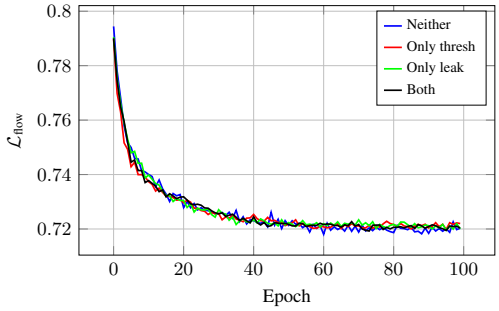
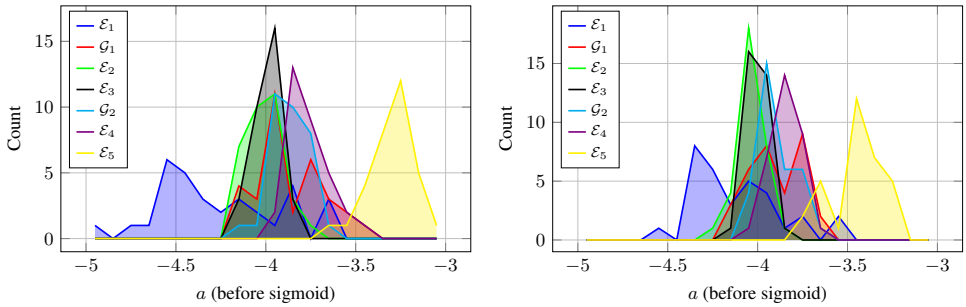


Figure 5.8: Training loss curves for LIF-FireNet variants with different sets of learnable parameters.

Learnable thresholds		X		X
Learnable leaks			X	X
outdoor_day1	0.65	0.68	**0.57**	0.58
indoor_flying1	1.14	1.04	0.97	**0.96**
indoor_flying2	1.88	1.89	**1.70**	1.82
indoor_flying3	1.62	1.61	**1.45**	1.52

Table 5.4: Ablation study on learnable parameters for SNNs on MVSEC [168] using variants of LIF-FireNet. We report the EPE↓ for dt = 1. Best in bold, runner up underlined.



(a) Variant with learnable leaks.

(b) Variant with learnable leaks and thresholds.

Figure 5.9: LIF-FireNet distribution of learned leaks $\alpha = \frac{1}{1+\exp(-a)}$, initialized at $a = -4$.

As mentioned in Section 5.4.3, one possible way of mitigating gradient vanishing would be to connect each layer to the loss directly, through, e.g., a regularization term on minimum activity as in [38]:

$$\mathcal{L}_{\text{act}} = \sum_{l}^{L} \max(0, f_{\text{desired}} - f_{\text{actual}})^2 \tag{5.17}$$

with L being all the spiking layers, f_{desired} the desired per-timestep fraction of active neurons, and f_{actual} the actual per-timestep fraction of active neurons. By taking the maximum, we ensure that $\mathcal{L}_{\text{act}}$ goes to zero as soon as the activity is above the desired level. The effect of adding activity regularization with $f_{\text{desired}} = 0.05$ can be observed in Fig. 5.11. While the direct connection between each layer and the loss is able to start learning for SuperSpike-10, it has little effect for SuperSpike-100. The bottom row of Fig. 5.10 shows that the gradient flow for SuperSpike-10 becomes non-negligible for earlier layers after step 40,000 or so; for SuperSpike-100, the gradients have increased significantly, but are still not enough to allow learning. These results are in line with the recent SNN literature, which shows that SuperSpike-100 can enable learning for shallow networks [38, 263], but that it degrades performance as the number of layers increases beyond four [257], and that tuning of the surrogate width is necessary. Note that [257] also demonstrates learning

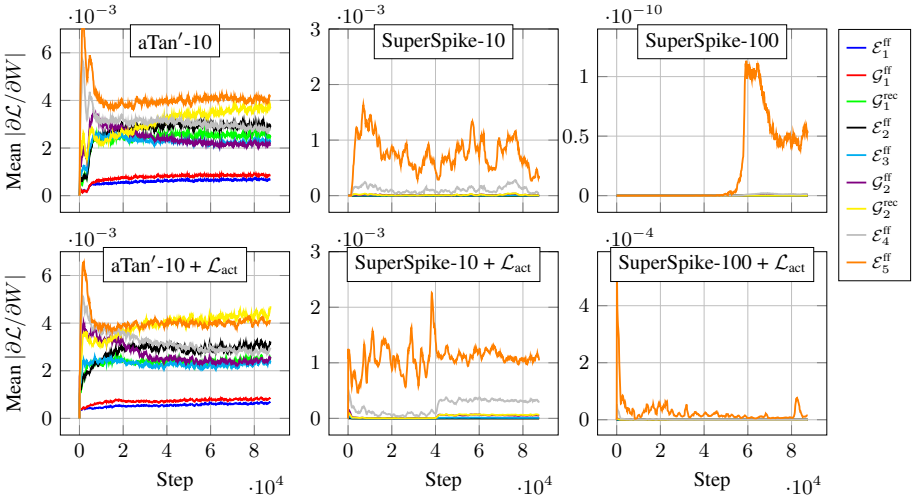


Figure 5.10: Per-layer mean of absolute gradient values during training of LIF-FireNet with various surrogate gradients, and activity regularization $\mathcal{L}_{act}$. aTan′-10 denotes $\sigma'(x) = 1/(1 + 10x^2)$, where $x = U - \theta$; SuperSpike-γ [261] denotes $\sigma'(x) = 1/(1 + \gamma|x|)^2$, with $\gamma \in \{10, 100\}$. Data is smoothed with a 1000-step moving average and a stride of 100.

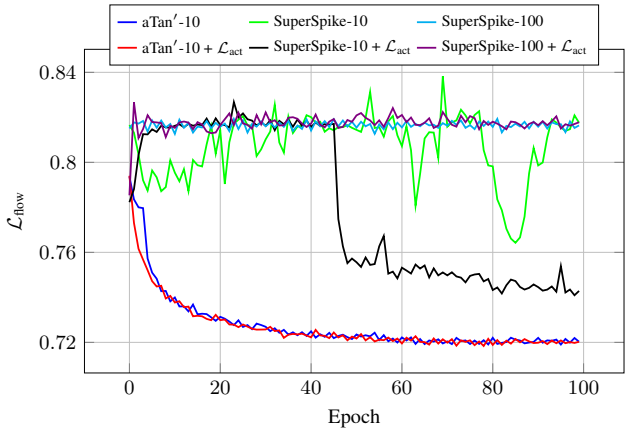
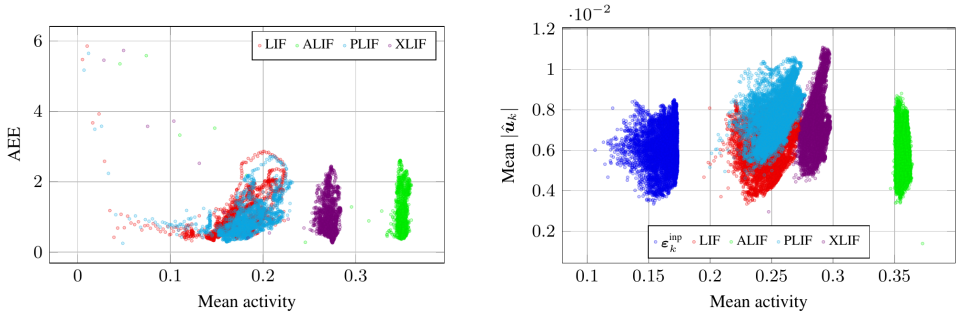


Figure 5.11: Impact of the choice of surrogate gradient σ' and activity regularization $\mathcal{L}_{act}$. aTan′-10 denotes $\sigma'(x) = 1/(1 + 10x^2)$, where $x = U - \theta$; SuperSpike-γ [261] denotes $\sigma'(x) = 1/(1 + \gamma|x|)^2$, with $\gamma \in \{10, 100\}$. Training loss curves belonging to LIF-FireNet.

with SuperSpike-10 for deeper networks, but this probably works because they use batch normalization.

Comparison of activity levels for adaptive SNNs

SNNs implemented in neuromorphic hardware consume less energy as their activity decreases [244], which makes it important to investigate how activity levels vary across spiking neuron models, and how they correlate with the outputs of the network: because



(a) EPE against mean activity (over all spiking layers).

(b) Mean (normalized) optical flow magnitude (over all pixels that have at least one event) against mean activity (over all spiking layers).

Figure 5.12: Recorded activity (fraction of nonzero values) of spiking FireNet variants during (a) indoor_flying1 of MVSEC [168] with dt = 1 and (b) boxes_6dof of the ECD dataset [172] with N = 15k events. Each marker represents one timestep.

5

spiking layers emit only binary spikes, in some cases more spikes would be needed for output values larger in magnitude. We recorded the activity (fraction of nonzero values) and EPE of the {LIF, ALIF, PLIF, XLIF}-FireNet variants during the indoor_flying1 sequence of MVSEC [168] with dt = 1, as well as the mean normalized output optical flow magnitude during the boxes_6dof sequence of the ECD dataset [172] with N = 15k input events. Fig. 5.12 shows the results. One observation we can make from Fig. 5.12a is that the neuron models with an adaptive threshold (ALIF, XLIF) are more active than those without, while achieving similar EPEs. While this excessive spiking could be the result of initializing the base threshold β_0 too low, the similarity in EPE certainly suggests that these models spike too much for the performance they achieve, and that there is a certain redundancy in their activity.

Looking at Fig. 5.12b, we again observe that the models with adaptive threshold are more active than those without. On the other hand, it seems that ALIF and XLIF are more consistent in their activity across the range of outputs (narrower, more vertically oriented clusters). Looking at the clusters of LIF and PLIF, we can see that they both have roughly the same shape, but the latter's average output is larger in magnitude. This indicates that presynaptic and postsynaptic adaptive mechanisms can both serve a purpose: the former helps in increasing the absolute output range, while the latter helps in keeping activity (and therefore energy consumption) constant across this range. This makes the XLIF model especially interesting to investigate further in future work.

To approximate the efficiency gains of SNNs running on neuromorphic hardware and compare it with equivalent ANN implementations, we can look at the number of accumulate (AC) and multiply-and-accumulate (MAC) operations of each, as is also done in [260]. Using energy numbers for ACs and MACs from [264], this gives us a very rough 25x increase in energy efficiency of SNNs compared to ANNs, assuming that (i) floating-point MAC operations cost five times as much energy as floating point AC operations; (ii) SNNs only make use of AC operations, while ANNs only make use of MAC operations; (iii) the average activity level of the SNN is 20%, as in Fig. 5.12a. However, as rightly pointed out in

[244] (which contains a more elaborate quantification of efficiency gains of SNNs running on the Loihi neuromorphic processor), the comparison using AC and MAC operations for respectively SNNs and ANNs may not be a fair one for all tasks, considering, e.g., overhead in neuromorphic chips and the optimization of MACs in ANN accelerators.

5.5 Conclusion

In this chapter, we presented the first set of deep SNNs to successfully solve the real-world, large-scale problem of event-based optical flow estimation. To achieve this, we first reformulated the state-of-the-art training pipeline for ANNs to considerably shorten the time windows presented to the networks, approximating the way in which SNNs would receive spikes directly from the event camera. Additionally, we reformulated the state-of-the-art self-supervised loss function to improve its convexity. Prior to their training with this framework, we augmented several ANN architectures from literature with explicit and/or implicit recurrency, besides the addition of the spiking behavior. Extensive quantitative and qualitative evaluations were conducted on multiple datasets. Results confirm not only the validity of our training pipeline, but also the on-par performance of the proposed set of recurrent ANNs and SNNs with the self-supervised state-of-the-art. To the best of our knowledge, and especially due to the addition of explicit recurrent connections, the proposed SNNs correspond to the most complex spiking networks in the computer vision literature, architecturally speaking. For the SNNs, we also conducted several additional studies and (i) concluded that parameter initialization and the width of the surrogate gradient have a significant impact on learning: smaller weights in the prediction layer speed up convergence, while a too narrow surrogate gradient prevents learning altogether; and (ii) observed that adaptive mechanisms based on presynaptic activity outperform those based on postsynaptic activity, and perform similarly or better than the baseline without adaptation. Overall, we believe this chapter sets the groundwork for future research on neuromorphic processing for not only the event-based structure-from-motion problem, but also for other, similarly complex computer vision applications. For example, our results suggest the need for more powerful recurrent units for SNNs.

Supplementary material

Video summary of the approach: https://youtu.be/T7-9GGYnuZ4

Project code: https://github.com/tudelft/event_flow

6

Robustifying the SSL of Low-Latency, Event-Based Optical Flow

In the previous chapter, we introduced a novel self-supervised pipeline specifically designed for estimating low-latency, event-based optical flow. Despite its novelty and the research direction that it unlocked, we identified two limitations that originated from the underlying principle of contrast maximization, which serves as the theoretical basis for the self-supervision. Firstly, the pipeline assumed that events move linearly within the timespan of their loss function. This assumption restricts the ability of the pipeline to accurately capture the true (and potentially nonlinear) trajectory of scene points over time. Secondly, the success of the self-supervised training heavily depended on the hyperparameters controlling the amount of motion information perceived by the networks before computing the loss. In this chapter, we present a novel self-supervised learning framework that addresses these limitations, and that enables the sequential estimation of event-based optical flow while ensuring the scalability of the models to high inference frequencies without sacrificing accuracy. At the heart of our approach is a continuously-running stateful neural model that is trained using our novel formulation of contrast maximization that makes it robust to nonlinearities and varying statistics in the input events. Extensive experiments conducted on multiple datasets demonstrate the effectiveness of our method. The proposed pipeline establishes a new state of the art (given the literature at the time of publication) in terms of accuracy for approaches trained or optimized without ground truth data.

The contents of this chapter have been published in:

F. Paredes-Vallés, K. Y. W. Scheper, C. De Wagter, G. C. H. E. de Croon, *Taming contrast maximization for learning sequential, low-latency, event-based optical flow*, IEEE International Conference on Computer Vision (ICCV), 2023.

6.1 Introduction

Event cameras capture per-pixel log-brightness changes at microsecond resolution [18]. This operating principle results in a sparse and asynchronous visual signal that, under constant illumination, directly encodes information about the apparent motion (i.e., optical flow) of contrast in the image space. These cameras offer several advantages, such as low latency and robustness to motion blur [18], and hence hold the potential of a high-bandwidth estimation of this optical flow information. However, the event-based nature of the generated visual signal poses a paradigm shift in the processing pipeline, and traditional, frame-based algorithms become suboptimal and often incompatible. Despite this, the majority of learning-based methods that have been proposed so far for event-based optical flow estimation are still highly influenced by frame-based approaches. This influence is normally reflected in two key aspects of their pipelines: (i) the design of the network architecture, and (ii) the formulation of the loss function.

Regarding architecture design, most literature methods format subsets of the input events as dense volumetric representations [33] that are processed at once by stateless (i.e., non-recurrent) models [33, 34, 100, 150]. Similarly to their frame-based counterparts [265–267], these models estimate the per-pixel displacement over the time-length of the event volume using only the information contained within it. Consequently, these volumes need to encode enough spatiotemporal information for motion to be discernible. However, if done over relatively long time periods, the subsequent models suffer from limitations such as high latency or having to deal with large pixel displacements [34, 268, 269].

With respect to the loss function, multiple options have been explored due to the lack of real-world datasets with per-event ground truth. Pure supervised learning can be used with datasets such as MVSEC [168] or DSEC-Flow [34], but their ground truth only contains per-pixel displacement at low frequencies, which makes models difficult to train. On the other hand, a self-supervised learning (SSL) framework can be formulated using either the accompanying frames with the photometric error as a loss [25, 270, 271], or through an events-only contrast maximization [96, 97] proxy loss for motion compensation (i.e., event deblurring) [33, 100, 102, 161, 272]. However, despite not relying on ground truth, all the literature on SSL for optical flow assumes that the events move linearly within the time window of the loss, which ignores much of the potential of event cameras and their high temporal resolution (see Fig. 6.1, bottom left).

In this work, we focus on the estimation of high frequency event-based optical flow and how this can be learned in an SSL fashion using contrast maximization with a relaxed linear motion assumption. To achieve this, we build upon the continuous-operation pipeline from Hagenaars *et al.* [102] (i.e., Chapter 5 of this dissertation), which retrieves optical flow by *sequentially* processing small partitions of the event stream with a stateful (i.e., recurrent) model over time, instead of dealing with large volumes of input events. We augment (and train) this pipeline with a novel contrast maximization formulation that performs event motion compensation in an iterative manner at multiple temporal scales, as shown in Fig. 6.1. Using this framework, we achieve the best accuracy of all contrast-maximization-based approaches on multiple datasets, only being outperformed by pure supervised learning methods trained with ground-truth data.

In summary, the extensions that we propose to the SSL framework in [102], i.e., our main contributions, are: (i) the first iterative event warping module in the context of contrast

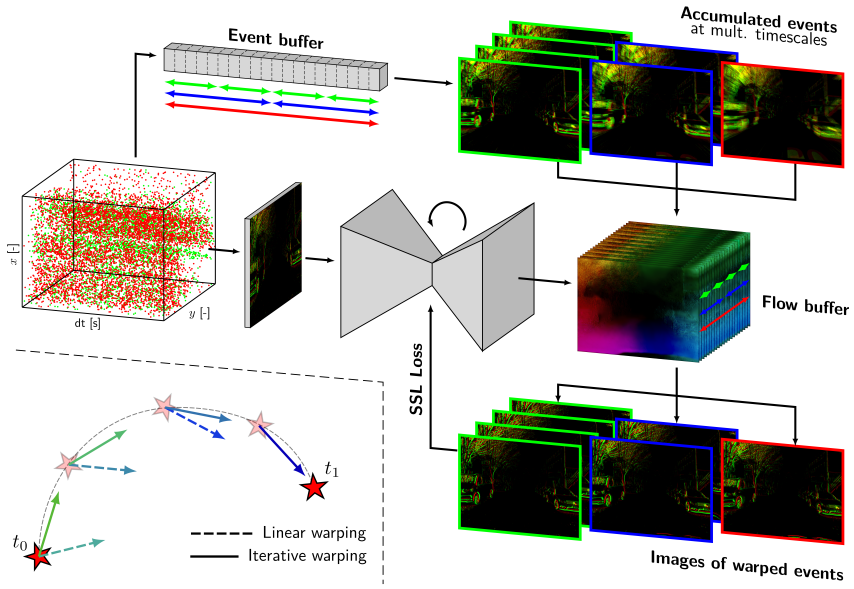


Figure 6.1: Our pipeline estimates event-based optical flow by sequentially processing small partitions of the event stream with a recurrent model. We propose a novel self-supervised learning framework based on a multi-timescale contrast maximization formulation that is able to exploit the high temporal resolution of event cameras via iterative warping to produce accurate optical flow predictions.

maximization, which unlocks a novel multi-reference loss function that better captures the trajectory of scene points over time, thus improving the accuracy of the predictions (see Section 6.3.2); and (ii) the first multi-timescale approach to contrast maximization, which adds robustness, improves convergence, and reduces tuning requirements of loss-related hyperparameters (see Section 6.3.3). As a result, we present the first self-supervised optical flow method for event cameras that relaxes the linear motion assumption, and hence that has the potential of exploiting the high temporal resolution of the sensor by producing estimates in a close-to continuous manner. We validate the proposed framework through extensive evaluations on multiple datasets. Additionally, we conduct ablation studies to show the effectiveness of each individual component.

6.2 Related work

Due to the aforementioned advantages of event cameras for optical flow estimation, extensive research has been carried out since these sensors were first introduced [154, 187, 217, 248, 272, 273]. Regarding learning-based approaches, the first method was proposed by Zhu *et al.* in [25] with EV-FlowNet: a UNet-like [137] architecture trained with SSL, with the supervisory signal coming from the photometric error between subsequent frames captured with an accompanying camera. To avoid the need for a secondary vision sensor, in [33], Zhu *et al.* proposed an SSL framework around the contrast maximization for motion compensation idea from [96, 97], hence relying solely on event data. This pipeline was used and further improved in [100, 102, 272]. Other approaches trained EV-FlowNet in

a supervised fashion with synthetic/real ground-truth data and showed higher accuracy levels through evaluations on public benchmarks [34, 150]. However, they also highlighted EV-FlowNet's inability to deal with large pixel displacements [34]. Because of this, inspired by the frame-based literature [267], Gehrig *et al.* departed from the use of UNet-like networks and proposed E-RAFT in [34]: the first architecture to introduce the use of correlation volumes in the event camera literature. This model, which was recently augmented with attention mechanisms [274], achieved state-of-the-art performance in multiple datasets.

A novel perspective to learning event-based optical flow is to move away from event accumulation over long timespans, and instead rely on continuously-running stateful models that integrate information over time. The first methods of this kind were proposed by Paredes-Vallés *et al.* [101] and Hagenaars *et al.* [102], but this idea has recently gained interest in the event camera literature [269, 270, 275]. The reason is that leveraging memory through sequential processing can potentially lead to lightweight and low-latency solutions that are also robust to large pixel displacements without the need for correlation volumes [269]. However, despite their potential of being scaled to high inference frequencies, all these solutions assume that events move linearly in the timespan of their loss function, and hence cannot capture the true trajectory of scene points over time.

When it comes to learning nonlinear pixel trajectories from event data, only the work of Gehrig *et al.* in [268] is to be highlighted. However, their approach uses multiple event and correlation volumes to fit Bézier curves to the trajectory of scene points, resulting in a high-latency solution. In contrast, in this work, we extend the continuous-operation pipeline from Hagenaars *et al.* [102] with an SSL framework in which discrete trajectories are regressed at high frequency by leveraging memory within the models. This approach allows, for the first time, to exploit the high temporal resolution of event cameras while capturing more accurately the trajectory of scene points over time thanks to the proposed iterative event warping mechanism.

Among non-learning-based approaches, the work of Shiba *et al.* in [272] bears particular relevance due to certain similarities with our framework. Specifically, Shiba *et al.* propose a tile-based method for event-based optical flow estimation that extends contrast maximization [96, 97] by incorporating (i) multiple spatial scales, (ii) a multi-reference focus loss, and (iii) a "time-aware" optical flow formulation. Similarly, our learning-based approach also employs a multi-scale strategy and utilizes a multi-reference focus loss. However, instead of making assumptions about the events' motion over extended time periods, we propose to learn their potentially nonlinear trajectories at high frequency by leveraging the iterative event warping module.

6.3 Method

The goal of this work is to learn to sequentially estimate optical flow at high frequencies from a continuous stream of events. In such a pipeline, if the inference frequency is sufficiently high, events need to be processed nearly as soon as they are triggered by the sensor, with no pre-processing in between. Because of this, the integration of temporal information needs to happen in the network itself. To accomplish this, we propose the framework in Fig. 6.1, in which a stateful model is trained using our novel formulation of contrast maximization for sequential processing. The components of this framework are described in the following sections.

6.3.1 Input format and contrast maximization

For an ideal camera, an event $\boldsymbol{e}_i = (\boldsymbol{x}_i, t_i, p_i)$ of polarity $p_i \in \{+,-\}$ is triggered at pixel $\boldsymbol{x}_i = (x_i, y_i)^T$ and time t_i whenever the change in log-brightness since the last event at that pixel location reaches the contrast sensitivity threshold for that polarity [18].

As in [102], we use a two-channel event count image as input representation, which gets populated with consecutive, non-overlapping, fine discrete partitions of the event stream $\boldsymbol{\varepsilon}_k^{\text{inp}} \doteq \{\boldsymbol{e}_i\}$ (further referred to as *input partitions*), each containing all the events in a time window of a certain duration, i.e., $t_i \in [t_k^{\text{begin}}, t_k^{\text{end}}]$. This representation does not contain temporal information by itself, and recurrent models are hence required.

Regarding learning, we use the contrast maximization framework [97] to train, in an SSL fashion, to continuously estimate dense (i.e., per-pixel) optical flow from the event stream. Assuming brightness constancy, accurate flow information is encoded in the spatiotemporal misalignments (i.e., blur) among the events triggered by the same portion of a moving edge. To retrieve it, one has to compensate for this motion by geometrically transforming the events using a motion model. As in [33, 100, 102, 272], we transport each event to a reference time t_{ref} through:

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + (t_{\text{ref}} - t_i)\boldsymbol{u}(\boldsymbol{x}_i) \tag{6.1}$$

where $\boldsymbol{u}(\boldsymbol{x}) = (u(\boldsymbol{x}), v(\boldsymbol{x}))^T$ denotes the optical flow map used to transport each event from t_i to t_{ref}. The result of aggregating the transformed events is further referred to as the image of warped events (IWE) at t_{ref}.

As the loss function of our SSL framework, we adopt the time-based focus objective function from [102]. Using the warped events at a given t_{ref}, we generate an image of the per-pixel average timestamps for each polarity p' via bilinear interpolation:

$$\begin{aligned} T_{p'}(\boldsymbol{x};\boldsymbol{u}|t_{\text{ref}}) &= \frac{\sum_j \kappa(x - x_j')\kappa(y - y_j')\bar{t}_j(t_{\text{ref}}, t_j)}{\sum_j \kappa(x - x_j')\kappa(y - y_j') + \epsilon} \\ \kappa(a) &= \max(0, 1 - |a|) \\ j = \{i \mid p_i = p'\}, &\qquad p' \in \{+,-\}, \qquad \epsilon \approx 0 \end{aligned} \tag{6.2}$$

where $\bar{t}_i$ denotes the normalized timestamp contribution of the ith event, according to Fig. 6.3 and Eq. 6.5.

Then, the contrast maximization loss function at t_{ref} is defined as the scaled sum of the squared temporal images:

$$\mathcal{L}_{\text{CM}}(t_{\text{ref}}) = \frac{\sum_{\boldsymbol{x}} T_+(\boldsymbol{x};\boldsymbol{u}|t_{\text{ref}})^2 + T_-(\boldsymbol{x};\boldsymbol{u}|t_{\text{ref}})^2}{\sum_{\boldsymbol{x}} [n(\boldsymbol{x}') > 0] + \epsilon} \tag{6.3}$$

where $n(\boldsymbol{x}')$ denotes a per-pixel event count of the IWE. The lower the $\mathcal{L}_{\text{CM}}$, the better the event deblurring at t_{ref}.

As discussed in [97, 102, 256], for any focus objective function to be a robust supervisory signal for contrast maximization, the event partition used in the optimization needs to contain enough motion information (i.e., blur) so it can be compensated for. However, as in [102], that is not the case in our input partitions due to the fine discretization of the event stream that we are targeting. Therefore, only at training time, we define the so-called

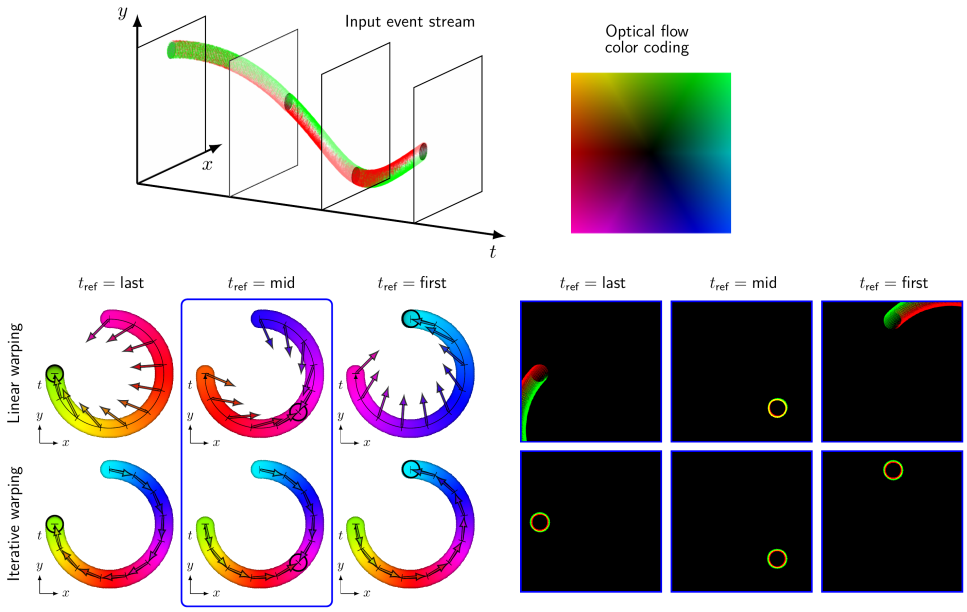


Figure 6.2: Incompatibility of multi-reference deblurring and linear event warping in the presence of nonlinearities in the pixel trajectories. *Top*: Events generated by a moving dot following circular motion, and optical flow color-coding scheme. *Bottom left*: Optical flow solutions required to produce sharp IWEs at different t_{ref} using linear and iterative warping. While the former requires a different solution for each t_{ref}, the proposed iterative warping can achieve this using a single solution. Arrows illustrate the direction of the required displacement $\Delta \boldsymbol{x}_i = (t_{\text{ref}} - t_i)\boldsymbol{u}(\boldsymbol{x}_i)$ at each (discretized) spatial location. *Bottom right*: Resulting IWEs at different t_{ref} using the optimal optical flow map for $t_{\text{ref}} = \text{mid}$.

training partition (or event buffer in Fig. 6.1) $\boldsymbol{\varepsilon}^{\text{train}}_{k \longrightarrow k+R} \doteq \{\boldsymbol{\varepsilon}^{\text{inp}}_i\}^{k+R}_{i=k}$, which stacks together the events in the input partitions of R successive forward passes through the network. Once these are performed, we use this partition and the estimated optical flow maps to compute the loss, and use truncated backpropagation through time to update the model parameters. After this update, we detach the states of the network from the computational graph and clear the training partition and optical flow buffer. The importance of sequential processing, as an alternative to training stateless models in short input partitions, is corroborated in Section 6.4.3.

6.3.2 Iterative event warping

In order to better approximate the trajectories of scene points over time, in this work we relax the linear motion assumption in the SSL framework for event-based optical flow by augmenting the sequential-estimation pipeline from [102] with *iterative event warping*. Instead of transporting events to a given t_{ref} assuming linear motion regardless of the length of the warping interval (as in [33, 100, 102, 272]), we perform a finer discretization of the event trajectories and assume that motion is only linear between optical flow estimates. Therefore, to express a group of events at a given t_{ref}, we geometrically transform them using all the intermediate optical flow estimates through multiple iterations of Eq. 6.1.

Fig. 6.2 shows the differences between the linear event warping in [102] and the proposed iterative augmentation, as well as the limitations of the former. Note that our iterative warping is fundamentally different from that of the work of Wu *et al.* in [269], where input events are deblurred before being passed to the models using residual optical flow estimates until convergence.

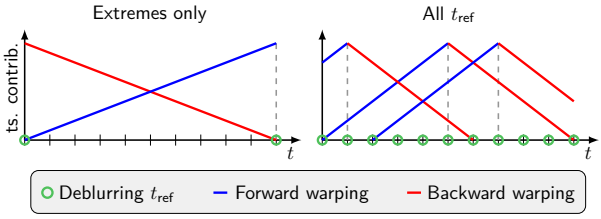


Figure 6.3: Timestamp normalization profiles for the per-event contributions to the images of average timestamps in Eq. 6.2, for R=10. *Left*: Deblurring only done at the extremes of the training partition, as in [33, 100, 102]. *Right*: Deblurring done at all reference times, thanks to the proposed iterative event warping. Normalization profiles only shown for three t_{ref} for a better visualization.

Literature methods on event-based optical flow with contrast maximization compute the focus objective function at multiple reference times in the training partition (usually at the extremes) to prevent overfitting and/or scaling issues during backpropagation [33, 100, 102, 272]. However, since these approaches assume optical flow constancy in the span of their loss function, they suffer with nonlinearities in the pixel trajectories (see the blurry IWEs in Fig. 6.2, bottom right). On the contrary, because of the finer discretization of the event trajectories coming with our sequential processing pipeline and the proposed iterative warping, we can use any (combination of) reference time(s) for the computation of the focus objective function. In fact, as shown in Fig. 6.3 (right), we propose the use of *all* the discretization points as reference times for event deblurring. Apart from the aforementioned regularizing benefits, having to produce sharp IWEs at any t_{ref} forces the models to estimate a sequence of optical flow maps that is consistent with the velocity profile of the event stream. For a given training partition of length R, the loss is computed as follows:

$$\mathcal{L}_{\text{CM}}^{R} = \frac{1}{R+1} \sum_{t_{\text{ref}}=0}^{R} \mathcal{L}_{\text{CM}}(t_{\text{ref}}) \tag{6.4}$$

which ensures that the IWEs (and the corresponding images of average timestamps in Eq. 6.2) at all reference times $t_{\text{ref}} \in [0, R]$ contribute equally to the loss.

As in [33, 100, 102], for $\mathcal{L}_{\text{CM}}(t_{\text{ref}})$ in Eq. 6.3 to be a valid supervisory signal, events that are temporally close to the reference time t_{ref} need to contribute to the temporal image in Eq. 6.2 with larger timestamp values than events that are far in time. Therefore, once the events have been transported to t_{ref}, their timestamp is normalized prior to the computation of Eq. 6.2 as follows:

$$\bar{t}_i(t_{\text{ref}}, t_i) = 1 - \frac{|t_{\text{ref}} - t_i|}{R}, \quad \text{with } t_i \in [0, R] \tag{6.5}$$

which results in the normalization profiles in Fig. 6.3, for a given training partition of length R.

Inspired by [276], we mask individual events from the computation of the loss whenever we detect that they are transported outside the image space through the event warping

process in the span of a deblurring window defined at the reference time t_{ref}. This prevents our models from learning incorrect deformations at the image borders, and is motivated by the fact that the complete trajectory of the pixel is only partially observable in the training partition. An ablation study on the impact of this masking strategy can be found in Section 6.4.3.

Lastly, we do not augment the loss function in Eq. 6.4 with smoothing priors acting as regularization mechanisms. With iterative event warping, the error propagates through all the pixels covered in the warping process, regardless of whether they have input events or not. Therefore, the spatial coherence of the resulting optical flow maps is enhanced.

6.3.3 Deblurring at multiple timescales

Despite the addition of iterative event warping, the success of our SSL framework still heavily depends on the hyperparameters that control the amount of motion information perceived by the networks in the span of a deblurring window. In our pipeline, these are: dt_{input}, the timestep used to discretize the event stream; and R, the number of forward passes, and hence optical flow maps, per loss. Thus, the effective length (in units of time) of the event window used for motion compensation is given by $\text{dt}_{\text{input}} \times R$. We hypothesize that, for each training dataset, there is an optimal length for this window that depends on the statistics of the data (e.g., event density, distribution of optical flow magnitudes) and model architecture, and that deviations from this optimal length lead to the learning of suboptimal solutions. E.g., shorter windows may converge to solutions that are more selective to fast rather than slow moving objects, and vice versa. Note that not only our method is sensitive to the tuning of these parameters, but also previous approaches based on contrast maximization [33, 100, 102, 272].

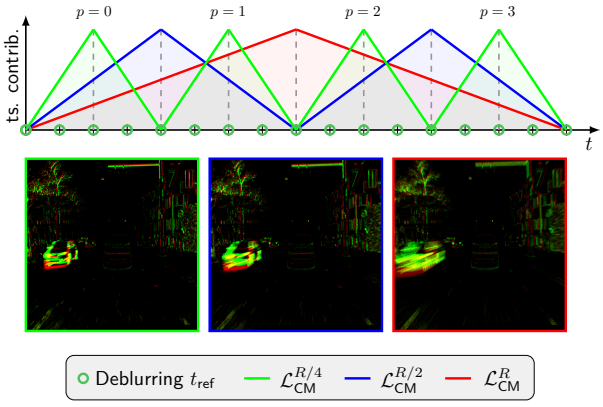


Figure 6.4: Multi-timescale approach to contrast maximization. For a given training partition of length R, we fit multiple sub-partitions of different lengths (in the figure: one of length R in red, two $R/2$ in blue, and four $R/4$ in green) and compute the loss in each of them according to Eq. 6.4. The global loss is computed as in Eq. 6.6. This figure only shows the timestamp normalization profiles of the central t_{ref} of each sub-partition, but the losses are still computed at all reference times. An image representation of the accumulated input events in a sub-partition of each timescale is also shown.

To add a layer of robustness to the framework and relax its strong dependency on hyperparameter optimization, we propose the *multi-timescale approach* illustrated in Figs. 6.1 and 6.4. For a given training partition of length R, instead of computing a single focus loss through Eq. 6.4, we compute this loss at S temporal scales of length $R/2^s$, with $0 \leq s \leq S-1$, and combine them as follows:

$$\mathcal{L}_{\text{CM}}^{\text{multi}} = \frac{1}{S}\sum_{s=0}^{S-1}\frac{1}{2^s}\sum_{p=0}^{2^s-1}\mathcal{L}_{\text{CM},p}^{R/2^s} \tag{6.6}$$

As shown, we fit multiple non-overlapping sub-partitions in the training buffer if $S > 0$. The subscript p indicates their location in this buffer, starting from the earliest (see Fig. 6.4).

Note that, through this multi-timescale approach to contrast maximization, our models need to converge to a solution that is suitable for all the timescales in the optimization, regardless of their length. An alternative formulation would be to incorporate per-pixel learnable masks (i.e., an attention module in the loss space) so that, depending on the input statistics, learning only happens at the most adequate scale. However, for this to happen, the loss function would have to be augmented to stimulate this behavior, and it is unclear how that would be done in practice.

6.3.4 Network architecture

We use the recurrent version of EV-FlowNet [25] proposed in [102][1] (see Fig. 6.5). The events are represented as event count images (see Section 6.3.1), then passed through four encoders with strided convolutions followed by ConvGRUs [169] (channels doubling, starting from 64), two residual blocks [167], and then four decoders performing bilinear upsampling followed by convolution. After each decoder, there is a skip connection (using element-wise summation) from the corresponding encoder, as well as a depthwise convolution to produce estimates at lower scales, which are then concatenated with the activations of the previous decoder. Note that the proposed focus loss function (see Eq. 6.6) is applied to each intermediate optical flow estimate via upsampling. Lastly, all layers use 3×3 kernels and ReLU activations except for the prediction layers, which use TanH.

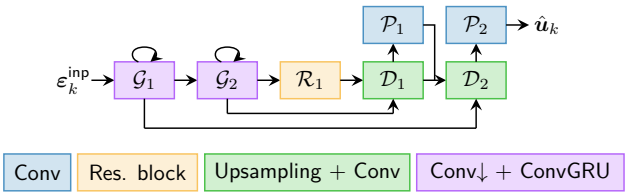


Figure 6.5: Schematic of the model architecture used in this work. It is characterized by $N_{\mathcal{G}}$ recurrent encoders, $N_{\mathcal{R}}$ residual blocks, and $N_{\mathcal{G}}$ decoder layers. Optical flow estimates are produced at all decoder levels. In this diagram, $N_{\mathcal{G}} = 2$ and $N_{\mathcal{R}} = 1$.

6.4 Experiments

We evaluate our method on the DSEC-Flow [34, 277] and MVSEC [168] datasets. We evaluate the accuracy of the predictions based on the following metrics: (i) EPE (lower is better, ↓), the endpoint error; (ii) $\%_{\text{3PE}}$ (↓), the percentage of points with EPE greater than 3 pixels; (iii) FWL (↑) [150], a deblurring quality metric based on the variance of the

[1] Our architecture is equivalent to ConvGRU-EV-FlowNet [102]. However, for the purpose of clarity within the rest of the chapter, we will use this term to specifically refer to the original model from [102].

IWEs; and (iv) RSAT (↓) [102], a deblurring quality metric based on the per-pixel average timestamps of the IWEs. We compare our solution to the published baselines, which range from supervised learning (SL) methods trained with ground truth, to SSL methods trained with grayscale images (SSL_F) or events (SSL_E), and model-based approaches (MB).

We train all our models on a subset of sequences from the training dataset of DSEC-Flow (only daylight recordings[2]). This corresponds to 19 minutes of training data, which we split into 572 128 × 128 (randomly-cropped) sequences of 2 seconds each. We use a batch size of 8 and train until convergence with the Adam optimizer [173] and a learning rate of 1e-5. To keep memory usage within limits, we only propagate error gradients through up to 1e3 randomly-chosen events per millisecond of data. Despite this, note that we warp and use all the input events for the computation of the loss function.

6.4.1 Evaluation procedure

When evaluating our sequential models, if $dt_{gt} > dt_{input}$, we need to reconstruct the estimated per-pixel displacement in the ground-truth time window from the multiple optical flow maps estimated in this period. We do this by first averaging the (bilinearly interpolated) optical flow vectors that describe the trajectory of each scene point, and then by scaling the resulting optical flow vectors by dt_{gt}/dt_{input}. An illustration of this reconstruction is shown in Fig. 6.6 for a scene point following a nonlinear trajectory. Note that our solution is subject to cumulative errors when evaluated through this reconstruction on benchmarks with ground truth provided at low rates (e.g., 10 Hz in DSEC-Flow [34]). Therefore, it will compare unfavorably to other, non-sequential, methods that only produce a single optical flow estimate in the timespan of a ground-truth sample.

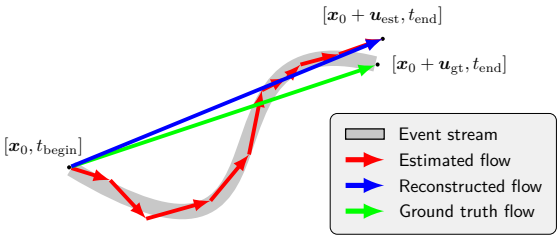


Figure 6.6: Reconstruction of the pixel displacement of a scene point in the time window of a ground-truth sample from the multiple optical flow maps estimated in this period, i.e., $\forall \boldsymbol{u}_k \in [t_{\text{begin}}, t_{\text{end}}]$. The error of the last optical flow estimate is magnified for clarity.

6.4.2 Optical flow evaluation

Evaluation on DSEC-Flow

Quantitative results of our evaluation on DSEC-Flow are presented in Table 6.1, and are supported by the qualitative comparison in Fig. 6.7. For this experiment, we trained multiple models with the same $dt_{input} = 0.01$s (i.e., ×10 faster than DSEC's ground truth) but different lengths of the training partition, and with and without the multi-timescale approach.

[2]The contrast maximization framework for motion compensation assumes constant illumination [96, 97]. Under this assumption, all the events captured with an event camera are generated by the apparent motion of objects in the image space. However, the constant illumination assumption is often violated in sequences recorded at night because the main source of light in these environments comes from flashing, artificial lights (e.g., street lamps). In addition, the signal-to-noise ratio of these sensors decreases under low light conditions, which means that a large percentage of the captured events are not triggered by motion but by sensor noise [18].

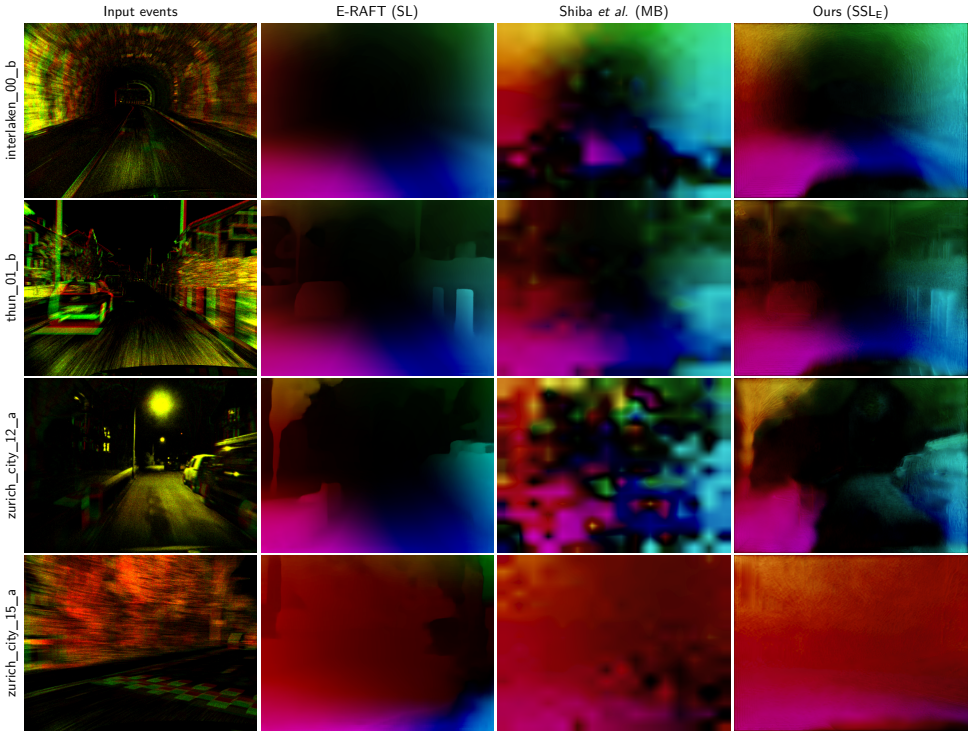


Figure 6.7: Qualitative comparison of our method with the state-of-the-art E-RAFT architecture [34] and the model-based approach from Shiba *et al.* [272] on sequences from the test partition of the DSEC-Flow dataset [34]. Ground truth not included due to unavailability. The optical flow color coding can be found in Fig. 6.2 (*top*), and the corresponding IWEs in Fig. 6.8.

Multiple conclusions can be derived from the reported results. Firstly, our best performing model (i.e., $R = 10$, $S = 1$) achieves the best accuracy of all contrast-maximization-based approaches on this dataset according to the EPE and the percentage of outliers. Specifically, it outperforms the baselines with an improvement in the EPE in the 33% – 45% range, only being outperformed by SL methods trained with ground truth on the same dataset [34, 269]. This confirms that (i) the timestamp-based loss function in Section 6.3.1 allows us to learn accurate event-based optical flow (contrary to the findings of [278]); and that (ii) our augmentations to the sequential pipeline in [102] lead to a significant improvement in the accuracy of the model (i.e., 45% improvement in the EPE).

Secondly, these results also confirm our hypothesis that, for each training dataset, there is an optimal length for the training partition R in terms of the EPE. According to Table 6.1, the optimal R for this dataset, our model architecture, and our dt_{input} is 10 (i.e., 0.1s of event data), with the EPE increasing if the training partition is made shorter or longer. However, as also shown in this table, we can relax the strong dependency of the contrast maximization framework on this parameter through the proposed multi-timescale approach. Our $S > 1$ models converged to solutions that slightly underperform our best

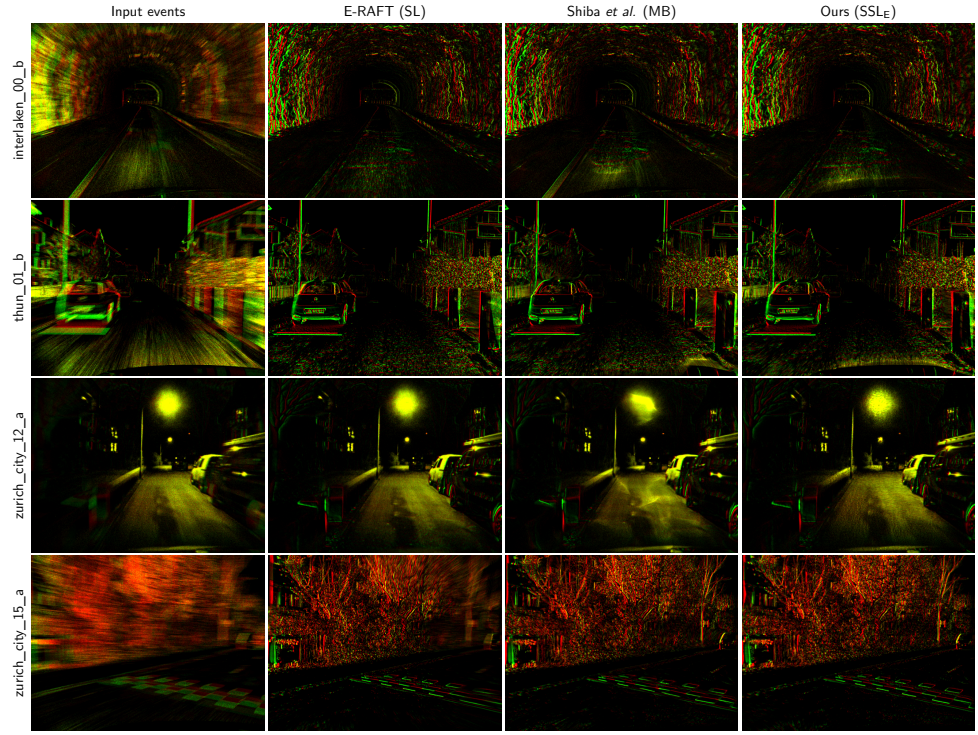


Figure 6.8: IWEs corresponding to the qualitative comparison of our method with the state-of-the-art E-RAFT architecture [34] and the model-based approach from Shiba *et al.* [272] on sequences from the test partition of the DSEC-Flow dataset [34] (see Fig. 6.7).

performing single-scale model (EPE went up by 17% – 21%), but were trained without the need to fine-tune the length of the training partition. Note that, despite this slight drop in accuracy, these multi-timescale solutions still outperform the other non-SL baselines. To further support these results, a visualization of the distribution of the EPE of our models as a function of the ground truth magnitude is provided in Section 6.4.3.

Lastly, Table 6.1 also allow us to conclude that deblurring quality metrics FWL [150] and

		EPE↓	%$_{3PE}$↓	FWL↑	RSAT↓
SL	E-RAFT [34]	0.79	2.68	1.33	0.87
	EV-FlowNet, Gehrig *et al.* [34]	2.32	18.60	-	-
	Gehrig *et al.* [268]	0.75	2.44	-	-
	IDNet [269]	**0.72**	**2.04**	-	-
	TIDNet [269]	0.84	3.41	-	-
	TMA [279]	0.74	2.30	-	-
	Cuadrado *et al.* [280]	1.71	10.31	-	-
	E-Flowformer [281]	0.76	2.68	-	-
SSL$_E$	EV-FlowNet* [33]	3.86	31.45	1.30	**0.85**
	ConvGRU-EV-FlowNet* [102]	4.27	33.27	1.55	0.90
	dt = 0.01s, R = 2, S = 1 (Ours)	9.66	86.44	**1.91**	1.07
	dt = 0.01s, R = 5, S = 1 (Ours)	4.05	52.22	1.58	0.97
	dt = 0.01s, R = 10, S = 1 (Ours)	2.33	17.77	1.26	0.88
	dt = 0.01s, R = 20, S = 1 (Ours)	16.63	33.67	1.06	1.10
	dt = 0.01s, R = 10, S = 3 (Ours)	2.82	27.09	1.37	0.92
	dt = 0.01s, R = 20, S = 4 (Ours)	2.73	23.73	1.24	0.90
MB	Shiba *et al.* [272]	3.47	30.86	1.37	0.89

*Retrained by us on DSEC-Flow, linear warping.

Table 6.1: Quantitative evaluation on the DSEC-Flow dataset [34]. Best in bold, runner up underlined.

RSAT [102] are not reliable indicators of the quality of the estimated optical flow. The reason for this is their inability to capture "event collapse" issues (as described in [282]), and would give favorable scores to undesirable solutions that warp all the events into a few pixels. According to our results, the FWL metric, being the spatial variance of the IWE relative to that of the identity warp, suffers more from this issue: the best FWL value is obtained with a model with 9.66 EPE.

Regarding qualitative results, Fig. 6.7 shows a comparison of our best performing model with the state-of-the-art E-RAFT architecture [34] and the contrast-maximization-based approach from Shiba *et al.* [272] on multiple sequences from the test partition of DSEC-Flow (i.e., ground truth is unavailable). These results confirm that our models are able to estimate high quality event-based optical flow despite not having access to ground-truth data during training, and also show the superiority of our method over the current best contrast-maximization-based approach [272]. Two limit cases in which our models provide suboptimal solutions are also shown in this figure: (i) sequences recorded at night (e.g., zurich_city_12_a) due to the presence of large amounts of events triggered by flashing lights and not by motion; and (ii) the car hood, which is also problematic for E-RAFT (i.e., does not capture it) and for [272]. Note that, in our case, (ii) is an artifact of the pixel displacements reconstructed from multiple optical flow estimates, and could be mitigated by having an occlusion handling mechanism in this reconstruction process.

	EPE↓	$\%_{3PE}$↓	FWL↑	RSAT↓
dt = 0.1s*	3.48	34.72	0.98	**0.87**
dt = 0.05s	3.09	27.36	1.11	0.91
dt = 0.01s	**2.33**	**17.77**	1.26	<u>0.88</u>
dt = 0.005s	<u>2.34</u>	<u>17.92</u>	<u>1.38</u>	0.89
dt = 0.002s	2.66	21.83	**2.04**	0.90

*Non-recurrent, volum. event repr. with 10 bins.

Table 6.2: Impact of the input window length on the DSEC-Flow dataset [34]. Best in bold, runner up underlined.

In addition to the evaluation in Table 6.1 and Fig. 6.7, we also conducted an experiment in which we trained multiple models with different dt_{input} (ranging from 0.1s to 0.002s) but with the same amount of information in the training partition: 0.1s of event data. Results in Table 6.2 show that our sequential pipeline leads to an improvement in the accuracy of the predicted optical flow maps with respect to the stateless EV-FlowNet, which processes the 0.1s of event data at once. This improvement is due to the fact that the complexity of dealing with large pixel displacements gets reduced when processing the input data sequentially using shorter input windows. In addition to this, Table 6.2 also shows that the accuracy of our models is not compromised when estimating optical flow at higher frequencies, despite the high sparsity levels in the input data at those rates.

Evaluation on MVSEC

Quantitative results of our evaluation on MVSEC are presented in Table 6.3, and are supported by the qualitative comparison in Fig. 6.9. For this experiment, since (i) there is no consensus in the literature with respect to the training dataset [25, 33, 34, 100, 102, 150, 272], and (ii) the outdoor_day2 sequence (i.e., the other daylight, automotive sequence) is only 9 minutes of duration during which the event camera is subject to high frequency vibrations [168], we decided to transfer one of our models trained on DSEC-Flow to MVSEC. More specifically, we chose the model trained with dt_{input} = 0.005s and R = 20 from Table 6.2, as a model trained with a short input window on DSEC-Flow is expected to be robust to the slow motion statistics of MVSEC [34]. We deployed the model at the same frequency as

		outdoor_day1		indoor_flying1		indoor_flying2		indoor_flying3	
		EPE↓	$\%_{3PE}$↓	EPE↓	$\%_{3PE}$↓	EPE↓	$\%_{3PE}$↓	EPE↓	$\%_{3PE}$↓
SL	EV-FlowNet+ [150]	0.68	0.99	0.56	1.00	0.66	1.00	0.59	1.00
	E-RAFT [34]	**0.24**	1.70	-	-	-	-	-	-
	EV-FlowNet [34]	0.31	**0.00**	-	-	-	-	-	-
	TMA [279]	0.25	0.07	1.06	3.63	1.81	27.29	1.58	23.26
	Cuadrado *et al.* [280]	0.85	-	0.58	-	0.72	-	0.67	-
SSL_F	EV-FlowNet [25]	0.49	0.20	1.03	2.20	1.72	15.1	1.53	11.9
	Ziluo *et al.* [270]	0.42	**0.00**	0.57	0.10	0.79	1.60	0.72	1.30
SSL_E	EV-FlowNet [33]	0.32	**0.00**	0.58	**0.00**	1.02	4.00	0.87	3.00
	EV-FlowNet [100]	0.92	5.40	0.79	1.20	1.40	10.9	1.18	7.40
	EV-FlowNet [272]	0.36	0.09	-	-	-	-	-	-
	ConvGRU-EV-FlowNet [102]	0.47	0.25	0.60	0.51	1.17	8.06	0.93	5.64
	Ours (transferred from DSEC)	0.27	0.05	0.44	**0.00**	0.88	4.51	0.70	2.41
MB	Akolkar *et al.* [283]	2.75	-	1.52	-	1.59	-	1.89	-
	Brebion *et al.* [273]	0.53	0.20	0.52	0.10	0.98	5.50	0.71	2.10
	Shiba *et al.* [272]	0.30	0.11	**0.42**	0.09	**0.60**	**0.59**	**0.50**	**0.29**

Table 6.3: Quantitative evaluation on all MVSEC sequences [168]. Best in bold, runner up underlined.

the temporally-upsampled ground truth (i.e., 45 Hz). Results in Table 6.3 show that, in the outdoor_day1 sequence, our model outperforms the great majority of methods in terms of the EPE (even some SL methods trained on this dataset), and is only surpassed by the current state-of-the-art E-RAFT [34] and TMA [279] architectures. Note that this is also an automotive sequence, so it presents some similarities with dataset used to train this model. For the case of the indoor sequences, recorded with a drone flying in an indoor environment, our transferred model demonstrates (on average) an improvement of 25% in EPE compared to the architecturally-equivalent ConvGRU-EV-FlowNet model from Hagenaars *et al.* [102], while showing an error increase of 30% compared to Shiba et al. [272]. However, note that the latter method is not learning-based, so it is not subject to generalization issues besides those inherent to contrast maximization.

6.4.3 Additional experiments

Impact of sequential processing

Here, we study the impact of the proposed contrast maximization framework for sequential event-based optical flow estimation (i.e., short input partitions, longer training partitions; see Section 6.3) and compare it to the non-sequential pipeline from Zhu *et al.* [33] (i.e., input and training partitions are of the same length). To do this, we trained multiple models on DSEC-Flow with different dt_{input}, but with dt_{train}= 0.1s for the sequential models and dt_{train}= dt_{input} for the non-sequential. Quantitative results in Table 6.4 confirm the claims made in Section 6.3.1 about the fact that, for contrast maximization to be a robust supervisory signal, the training event partition used for the computation of the supervisory signal needs to contain enough motion information (i.e., blur) so it can be compensated for. As shown, non-sequential models converge to worse solutions the shorter the input window. On the other hand, our sequential

	EPE↓	$\%_{3PE}$↓
dt = 0.1s*	3.48	34.72
dt = 0.05s*	3.24	32.45
dt = 0.01s*	15.85	90.52
dt = 0.005s*	28.45	97.27
dt = 0.002s*	15.28	94.79
dt = 0.05s	3.09	27.36
dt = 0.01s	**2.33**	**17.77**
dt = 0.005s	2.34	17.92
dt = 0.002s	2.66	21.83

Table 6.4: Quantitative evaluation of the impact of sequential processing on DSEC-Flow [34]. Best in bold, runner up underlined. *: Non-recurrent, volumetric event representation with 10 bins.

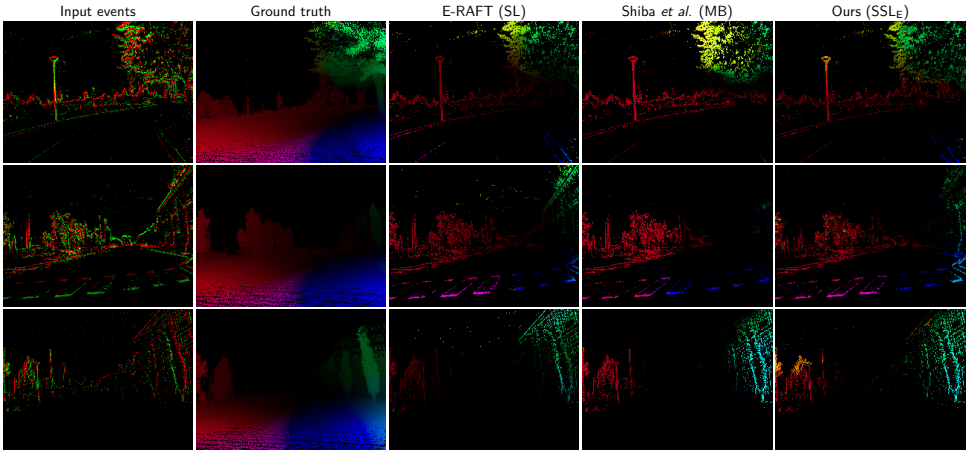


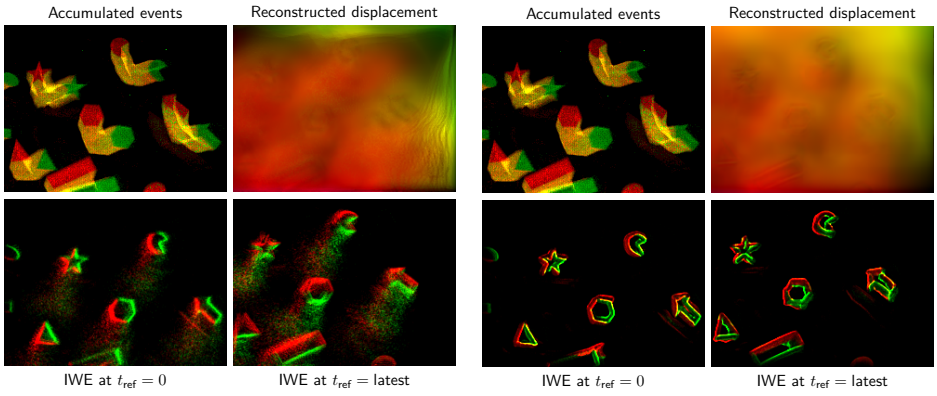
Figure 6.9: Qualitative comparison of our method with the state-of-the-art E-RAFT architecture [34] and the model-based approach from Shiba *et al.* [272] on the outdoor_day1 sequence from the MVSEC dataset [168]. Optical flow predictions are masked with the input events to be consistent with the evaluation proposed in [25]. The optical flow color coding can be found in Fig. 6.2 (*top*).

pipeline allows us to shorten the input window without compromising the performance, as discussed in 6.4.2.

Linear vs. iterative event warping

Here, we examine the effect of the type of event warping (linear [102] vs. iterative) on the performance of the sequential, stateful architecture introduced in Section 6.3.4 when it is trained on the DSEC-Flow dataset. To do this, we trained four models in total: two variants (with and without image border compensation) of ConvGRU-EV-FlowNet [102], which is trained with linear warping; and another two variants of the *same architecture*, but trained with the proposed iterative warping module. Quantitative and qualitative results are presented in Table 6.5 and Fig. 6.11, respectively. In both cases (with and without image-border compensation), the models trained with iterative warping (i.e., ours) outperform those trained with linear warping (EPE dropped by 28% without compensation, and 62% with it). This is expected, as the iterative warping module is able to better capture the trajectory of scene points over time, as explained in Section 6.3.2.

To support the arguments presented in Section 6.3.2 and Fig. 6.2 regarding the limitations of linear warping, we also conducted an experiment in which we deployed models trained on DSEC-Flow with linear and iterative warping on a sequence from the Event Camera Dataset [172] with strong nonlinearities in the trajectories of scene points. Note that this sequence, known as shapes_6dof, was recorded with a different event camera and that its statistics are significantly different from those of DSEC-Flow (i.e., hand-held camera looking at a planar scene [172] vs. automotive scenario [277]). Qualitative results are presented in Fig. 6.10. In addition to showing that the models generalize (to some extent) to this new sequence, these results demonstrate that only the models trained with iterative event warping are able to produce sharp IWEs at multiple reference times.



(a) ConvGRU-EV-FlowNet*† [102]. (b) Ours.

Figure 6.10: Qualitative results of the ablation study with respect to the type of event warping. Inference settings: dt = 0.01s, accumulation window of 0.5s. Both models were trained on the DSEC-Flow dataset [34], with dt = 0.01s, $R = 10$, $S = 1$. *: Retrained by us on DSEC-Flow [34]. †: Without border compensation. The optical flow color coding can be found in Fig. 6.2 (*top*).

Optical flow at the image borders

As discussed in Section 6.3.2, for a given temporal scale, we mask the events that are transported outside the image space at any time during the warping process from the computation of the loss to prevent learning incorrect optical flow at the image borders. Here we study the impact of this masking mechanism on the performance of not only the proposed SSL framework but also of two other literature methods: EV-FlowNet [33] and ConvGRU-EV-FlowNet [102]. For this experiment, we trained two versions of each model, one with and one without the proposed image-border compensation technique, on the DSEC-Flow dataset. Note that EV-FlowNet is a stateless model trained with a volumetric event representation [33] with 10 bins, and hence processes all the input events in between ground-truth samples at once.

Quantitative and qualitative results are presented in Table 6.5 and Fig. 6.11, respectively. These results highlight that, for both EV-FlowNet and our model, adding the proposed image-border compensation improves performance (EPE dropped by 10% and 24%, respectively). However, the performance degraded when adding it to the training pipeline of ConvGRU-EV-FlowNet (EPE went up by 43%). We believe that the reason for this drop in performance is the event warping method used during training. While the proposed iterative warping allows for the error to propagate through all the pixels covered in the

	EPE↓	$\%_{3PE}$↓
EV-FlowNet*† [33]	3.86	31.45
EV-FlowNet* [33]	3.48	34.72
ConvGRU-EV-FlowNet*† [102]	4.27	33.27
ConvGRU-EV-FlowNet* [102]	6.09	36.36
dt = 0.01s, $R = 10$, $S = 1$† (Ours)	3.08	21.38
dt = 0.01s, $R = 10$, $S = 1$ (Ours)	2.33	17.77

*Retrained by us on DSEC-Flow, linear warping.
†Without border compensation.

Table 6.5: Quantitative results of the ablation study on the DSEC-Flow dataset [34] with respect to the effectiveness of the proposed warping module and image-border compensation mechanism.

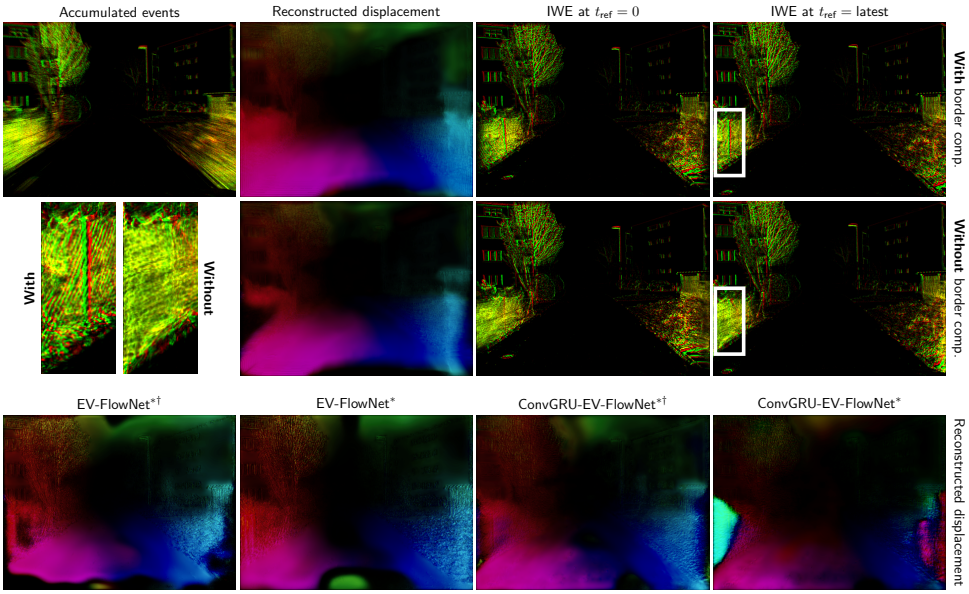


Figure 6.11: Qualitative results of the ablation study on the DSEC-Flow dataset [34] with respect to the effectiveness of the proposed event warping module and image-border compensation mechanism. *Top*: Models trained with the proposed SSL framework (dt = 0.01s, $R = 10$, $S = 1$). *Bottom*: Literature methods EV-FlowNet [33] and ConvGRU-EV-FlowNet [102]. $*$: Retrained by us on DSEC-Flow. †: Without border compensation. The optical flow color coding can be found in Fig. 6.2 (*top*).

warping process, the linear warping used to train ConvGRU-EV-FlowNet only propagates the error through pixels with input events [102]. Therefore, if events are removed from the computation of the loss, the error is not propagated through the corresponding pixels, and then the spatial coherence of the resulting optical flow maps degrades. Despite sharing the same warping methodology, this is less of an issue for EV-FlowNet since it processes the events from longer temporal windows in a single forward pass, producing a single optical flow map per loss. The longer this window, the more likely it is that a pixel contains events triggered by multiple moving objects (i.e., reflected as events with different timestamps), and hence the higher the probability that the error is propagated through that pixel.

Visualizing the endpoint error

To support the hypothesis in Section 6.3.3 that the length of the training partition R has a significant impact on the quality of the training, here we study the distribution of the EPE of our models in Table 6.1 as a function of the ground truth optical flow magnitude in the thun_00_a[3] sequence from DSEC-Flow [34]. The error distributions are shown in Fig. 6.12 and confirm the conclusions derived from Table 6.1 in Section 6.4.2. Models trained with short training partitions (i.e., $R \in [2, 5]$) converge to solutions that are less accurate (i.e., high EPE) for low ground truth magnitudes, while long partitions (i.e., $R \geq 20$) do the same

[3]Note that, since ground truth is required for this experiment, this sequence belongs to the training partition of DSEC-Flow [34]. Consequently, this means that our models have had access to a randomly cropped version of it during training.

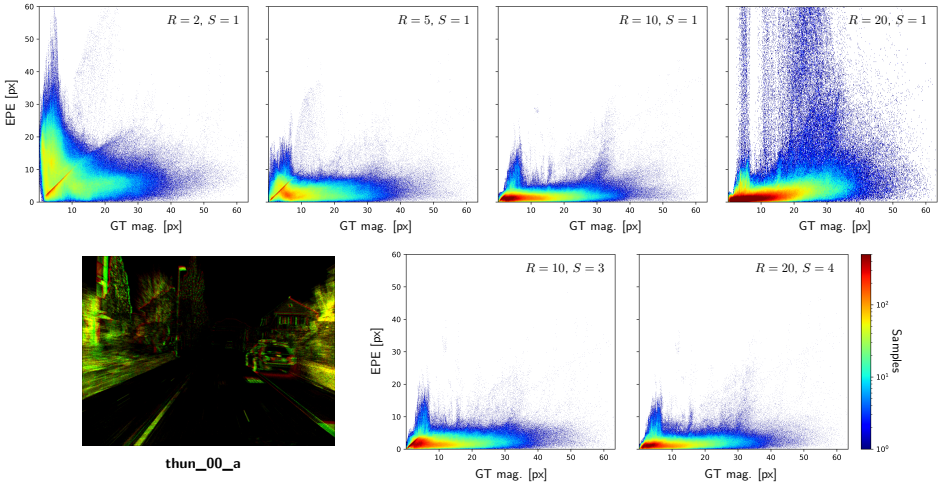


Figure 6.12: Distribution of the EPE of our models in Table 6.1 as a function of the ground truth magnitude in the thun_00_a sequence from DSEC-Flow [34]. For this experiment, and as in Table 6.1, all models were trained and deployed with $\mathrm{dt}_{\mathrm{input}} = 0.01$s.

but for high ground truth magnitudes. The proposed multi-timescale approach (i.e., $S > 1$) to contrast maximization alleviates this issue and allows for the training of models that are accurate for all ground truth magnitudes without having to fine-tune the length of the training partition. As shown in this figure, the error distribution of the $S > 1$ models closely resembles that of our best performing solution: $R = 10$, $S = 1$.

6.5 Limitations

The self-supervised method for event-based optical flow presented in this work, while demonstrating highly accurate and promising results, is not without limitations. Two critical challenges that need to be acknowledged are the brightness constancy assumption and the aperture problem. Firstly, the contrast maximization framework [96, 97] assumes constant illumination, leading our models to face difficulty in learning from events that are not due to motion in the image space but that arise from changes in illumination. Since this limitation is inherent to contrast maximization, it extends to other approaches based on the same principle [100, 102, 272]. Due to this assumption, we excluded sequences recorded at night from our training dataset. Secondly, akin to many other optical flow methods, our approach is susceptible to the aperture problem. This indicates that only motion components normal to the orientation of an edge in the image space, also known as normal optical flow, can be reliably resolved [105]. Consequently, the proposed method might face challenges in accurately determining the true motion direction in certain ambiguous scenarios. The regularizing effect of the iterative event warping and the multiple spatial scales at which dense optical flow is estimated in our architecture are mechanisms in our proposed solution that collectively strive to counteract the aperture problem's influence.

6.6 Conclusion

In this chapter, we presented the first learning-based approach to event-based optical flow estimation that is scalable to high inference frequencies while being able to accurately capture the true trajectory of scene points over time. The proposed pipeline is designed around a continuously-running stateful model that sequentially processes fine discrete partitions of the input event stream while integrating spatiotemporal information. We train this model through a novel, self-supervised, contrast maximization framework (i.e., event deblurring for supervision) that is characterized by an iterative event warping module and a multi-timescale loss function that add robustness and improve the accuracy of the predicted optical flow maps. We demonstrated the effectiveness of our approach on multiple datasets, where our models outperform the self-supervised and model-based baselines by large margins. Future research should look into how to learn to better combine the information from multiple timescales, as well as into the design of lightweight architectures that can keep up with real-time constraints.

We believe that the proposed approach opens up avenues for future research, especially in the field of neuromorphic computing. Spiking networks running on neuromorphic hardware have the potential of exploiting the main benefits of event cameras, but for that they need to process the input events shortly after they arrive. Our proposed framework is a step toward this objective, as it enables the estimation of optical flow in a close to continuous manner, with all the integration of information happening within the model itself.

Supplementary material

Video summary of the approach: https://youtu.be/vkYimENc494

Project code: https://github.com/tudelft/taming_event_flow

7

Fully Neuromorphic Vision and Control for Autonomous Drone Flight

The scientific discoveries from previous chapters converge here, where we present the first fully neuromorphic vision-to-control pipeline for controlling a flying robot. Specifically, we train a spiking neural network that accepts event-based camera data and outputs low-level control actions. The vision part of the network maps incoming events to ego-motion estimates and is trained with self-supervised learning on real event data. The control part is learned with an evolutionary algorithm in simulation. Robotic experiments show a successful sim-to-real transfer. The robot can accurately follow different ego-motion setpoints, allowing for hovering, landing, and maneuvering sideways—even while yawing at the same time. The neuromorphic pipeline runs on board on Intel's Loihi neuromorphic processor with an execution frequency of 200 Hz, spending only 27 μJ per inference. These results illustrate the potential of neuromorphic sensing and processing for enabling smaller, more intelligent robots.

The contents of this chapter have been submitted for publication to:

F. Paredes-Vallés†, J. J. Hagenaars†, J. D. Dupeyroux†, S. Stroobants, Y. Xu, G. C. H. E. de Croon, *Fully neuromorphic vision and control for autonomous drone flight*, Science Robotics, 2023.

Although not covered in this dissertation, this chapter also builds on the following publications:

J. D. Dupeyroux, J. J. Hagenaars, **F. Paredes-Vallés**, G. C. H. E. de Croon, *Neuromorphic control for optic-flow-based landing of MAVs using the Loihi processor*, IEEE International Conference on Robotics and Automation (ICRA), 2021.
J. J. Hagenaars, **F. Paredes-Vallés**, G. C. H. E. de Croon, *Evolved neuromorphic control for high speed divergence-based landings of MAVs*, IEEE Robotics and Automation Letters (RA-L), 2020.

† Equal contribution.

Contribution: The research leading to this chapter's work was the result of a collaborative effort with multiple researchers, all from the Micro Air Vehicle Laboratory (Delft University of Technology). Apart from contributing to the conception of the study, to performing the experiments, and to the analysis and interpretation of the results, I specifically designed the vision-based state estimation pipeline. This required simulating, training, validating and deploying spiking neural networks for event-based optical flow estimation.

7.1 Introduction

Over the past decade, deep artificial neural networks (ANNs) have revolutionized the field of artificial intelligence. Among the successes has been the significant improvement of visual processing, to an extent that computer vision can now outperform humans on specific tasks [284]. Also the field of robotics has benefited from this development, with deep ANNs achieving state-of-the-art performance in tasks such as stereo vision [285, 286], optical flow estimation [189, 266, 267], segmentation [287, 288], object detection [289–291], and monocular depth estimation [292–294]. However, this high performance typically relies on substantial neural network sizes that require quite heavy and power-hungry processing hardware. This limits the number of tasks that can be performed by larger (ground) robots, and even prevents deployment on smaller robots with highly stringent resource constraints, like small flying drones.

Neuromorphic hardware may provide a solution to this problem, since it mimics the energy-efficient, sparse and asynchronous nature of sensing and processing in biological brains [295, 296]. For example, the pixels in neuromorphic, event-based cameras only transmit information on brightness changes [18]. Since typically only a fraction of the pixels change in brightness significantly, this leads to sparse vision inputs with subsequent events that are in the order of a microsecond apart. The asynchronous and sparse nature of visual inputs from event-based cameras represents a paradigm shift compared to traditional, frame-based computer vision. Ideally, processing would exploit these properties for quicker, more energy-efficient processing. However, currently, learning-based approaches to event-based vision involve accumulating events over a substantial amount of time, creating an "event window" that represents extended temporal information. This window is then processed similarly to a traditional image frame with an ANN [25, 33, 34, 100]. While there is work that employs much shorter event windows [101–103], the full potential of neuromorphic vision will only be achieved when events are processed asynchronously as they come in by means of neuromorphic processors designed for implementing spiking neural networks (SNNs) [191, 297]. These networks have temporal dynamics more similar to biological neurons. In particular, the neurons have a membrane voltage that integrates incoming inputs and causes a spike when it exceeds a threshold. The binary nature of spikes allows for much more energy-efficient processing than the floating point arithmetic in traditional ANNs [298, 299]. The energy gain is further improved by reducing the spiking activity as much as possible, as is also a main property of biological brains [300]. Coupling neuromorphic vision to neuromorphic processing promises low-energy and low-latency visual sensing and acting, as exhibited by agile animals such as flying insects [301].

In this chapter, we present the first fully neuromorphic vision-to-control pipeline for controlling a flying drone, demonstrating the potential of neuromorphic hardware. To achieve this, we overcome several challenges related to present-day neuromorphic sensing and processing. For example, training is currently still much more difficult for SNNs than for ANNs [302, 303], mostly due to their sparse, binary, and asynchronous nature. The most well-known difficulty of SNN learning is the non-differentiability of the spiking activation function, which prevents naive application of backpropagation. Currently, this is tackled rather successfully with the help of surrogate gradients [37, 38], although longer sequences (as would be the case for event-by-event processing) can still lead to gradient vanishing. Moreover, while the richer neural dynamics can potentially represent more

complex temporal functions, they are also harder to shape; and neural activity may saturate or dwindle during training, preventing further learning. The causes for this are hard to analyze, as there are many parameters that can play a role. Depending on the model, the relevant parameters may range from neural leaks and thresholds to recurrent weights and time constants for synaptic traces. A solution may lie in learning these parameters [40, 304], but this further increases the dimensionality of the learning problem. Finally, when targeting a robotics application, SNN training and deployment is further complicated by the restrictions of existing embedded neuromorphic processing platforms, which are typically still rather limited in terms of numbers of neurons and synapses. As an illustration, the ROLLS chip [84] accommodates 256 spiking neurons, the Intel Kapoho Bay (featuring two Loihi chips [14] in a USB stick form factor) 262.1k neurons [85], and the SpiNNaker version in [82] 768k neurons. Although these chips differ in many more aspects than only the number of neurons, this small sample already shows that current state-of-the-art SNNs cannot be easily embedded on robots. SNNs that have recently been trained on visually complex tasks such as optical flow determination [101, 102], still feature far too large network sizes for implementation on current neuromorphic processing hardware for embedded systems. The smallest size SNN in these studies is LIF-FireFlowNet for optical flow estimation [102], which still has 3.7M neurons (at an input resolution of 128×128).

As a consequence, pioneering work in this area has been limited in complexity. Very early work involved the evolution of spiking neural network connectivity to map the 16 visual brightness inputs of a wheeled Kephera robot to its two motor outputs [305]. The evolved SNN, simulated in software, allowed the robot to avoid the walls in a black-and-white-striped environment. Most work exploring SNNs for robotic vision focuses on simulation. For example, in [215], the events from a simulated event-based camera with 128×128 pixels are accumulated into frames, compressing them over time into 8×4 Poisson input neurons. These inputs, which capture the clear white lines of the road border, are then directly mapped to two output neurons for staying in the center of the road with the help of reward-modulated spike-time-dependent plasticity learning. Robotic examples of in-hardware neuromorphic processing for vision are more rare. An early example is the one in [82], in which an event-based camera with 128×128 pixels is connected to a SpiNNaker neuromorphic processor to allow a driving robot to differentiate between two lights flashing at different frequencies with a 128-neuron winner-takes-all network. In [83] a spiking neural network is designed for following a light target in the top half of the field of view, while avoiding regions with many events in the bottom half of the field of view. This network is successfully implemented in the ROLLS neuromorphic chip [84] and tested in an office environment. Recent years have seen an increasing focus on flying robots, i.e., drones, because they need to react quickly while being extremely restricted in terms of size, weight, and power (SWaP). In [85], an SNN is implemented on a bench-fixed dual-rotor to align the roll angle with a black-and-white disk located in front of the camera. The SNN involved both a visual Hough transform [306] for finding the line, and a proportional-derivative (PD) controller for generating the propeller commands. Finally, in [307], an SNN was first evolved in simulation and then implemented in Loihi for vision-based landing of a flying drone. This control network only consisted of 35 neurons since the visual processing was still performed with conventional, frame-based computer vision methods. Additionally, it is worth noting that only the vertical motion of the drone was controlled with the SNN; its

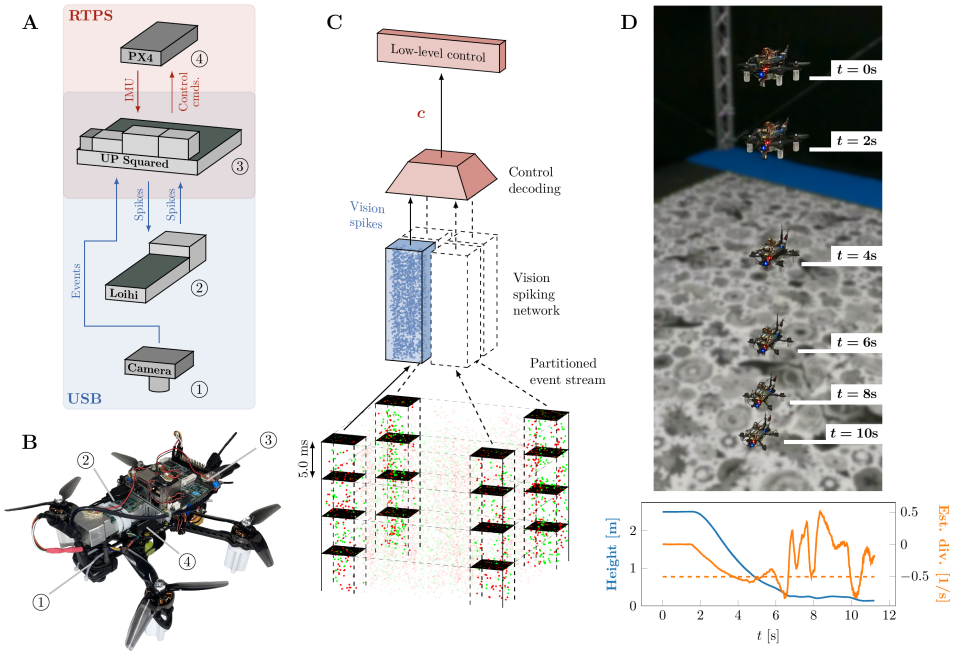


Figure 7.1: Overview of the proposed system. (*A*) Hardware overview showing the communication between event-camera, neuromorphic processor, single-board computer and flight controller. RTPS (real-time publish-subscribe) and USB refer to the used communication protocols. (*B*) Quadrotor used in this work (total weight 994 g, tip-to-tip diameter 35 cm). (*C*) Pipeline overview showing events as input, processing by the vision network and decoding into a control command. (*D*) Demonstration of the system for an optical flow constant divergence landing.

lateral position was controlled using traditional control algorithms and an external motion capture system. The current work represents a step up in complexity by performing 3D visual ego-motion estimation and control of a flying drone with a fully neuromorphic vision-to-control pipeline.

A fully neuromorphic solution to vision-based navigation

The presented vision-to-control pipeline consists of an SNN that is trained to accept raw event-camera data and output low-level control actions for performing autonomous vision-based ego-motion estimation and control at approximately 200 Hz. A core property of our learning setup is that it splits vision and control, which provides two major advantages. First, it helps to prevent the reality gap on the camera event input side, as the vision part is trained based on raw events from the actual event camera on the drone. We use self-supervised learning, since this foregoes the need for ground truth measurements that are difficult to obtain for event-based vision. Second, as the output of the vision part is an ego-motion estimate, we can learn the control in a simple and extremely fast simulator. This allows us to evade the high-frequency generation of visually realistic images for event generation [149], something that would lead to excessive training times in an end-to-end

learning setup. The resulting pipeline, illustrated in Fig. 7.1C, was implemented on the Loihi neuromorphic processor [14] and used on board a small flying robot (see Fig. 7.1B) for vision-based navigation. A schematic of the hardware setup employed is shown in Fig. 7.1A. The system successfully follows ego-motion setpoints in a fully autonomous fashion, i.e., without any external aids such as a positioning system. Fig. 7.1D shows an example of a landing experiment with our neuromorphic pipeline in the control loop of the drone. The figure shows the smoothly decreasing height of the drone above the ground (blue line), and the estimated optical flow divergence (orange line), which is the vertical component of the velocity vector divided by this height. The divergence curve is typical of an optical flow divergence landing, first approaching the setpoint -0.5 1/s and then becoming more oscillatory when getting very close to the ground [73].

As mentioned, the main challenge of deploying such a pipeline on embedded neuromorphic hardware is that, due to the preliminary state of this technology, one has to work within very tight limits regarding the available computational resources. In this project, several design decisions were made to adapt to these limitations. Firstly, the vision processing pipeline assumes that the event-based camera on the drone, the DAVIS240C [174], looks down at a static, texture-rich, flat surface. Knowing the structure of the visual scene in advance simplifies the estimation of the ego-motion of the camera (and hence of the drone) with the help of optical flow information, as in [70, 72, 73, 307, 308]. Optical flow, i.e., the apparent motion of scene points in the image space, can be estimated from the output of an event-based camera with a wide variety of methods, ranging from sparse feature-tracking algorithms [57] to dense (i.e., per-pixel) machine learning models [25, 34, 102]. In the search for an efficient and high-bandwidth vision pipeline that achieves the desired 200 Hz operating frequency, the second design decision was to reduce the spatial resolution of the event-based vision data by only processing information from the image corner regions of interest (ROIs) rather than the entire image space, and to limit the number of events to 90 per ROI. More specifically, as depicted in Figs. 7.1 and 7.2, we propose the use of a small SNN that is applied independently at each ROI, with each ROI being 16 × 16 pixels in size after a nearest-neighbor downsampling operation. Each network consists of 7.2k neurons and 506.4k synapses distributed over five spiking layers, i.e., one input layer, three self-recurrent encoders, and a pooling layer. Its parameters (i.e., weights, thresholds, and leaks) are identical for the four ROIs, and it estimates the optical flow, in pixels per millisecond, of the corresponding ROI. Because of the static and planar scene assumption, the apparent motion of the scene points at the four corner ROIs encodes non-metric information about the velocity of the camera (i.e., divided by the distance to the surface along the optical axis) and its rotational rates in a linear manner [309].

Based on this relation, we perform control. To keep the pipeline maximally neuromorphic (minimum required processing happening outside of Loihi) and performant (sending more spikes to Loihi decreases execution frequency), we train a linear controller in simulation, and merge it with the decoding of the spikes coming from the vision network (representing optical flow). In other words, the linear controller takes vision spikes, a user-given control setpoint, and attitude of the drone and maps these linearly to thrust and attitude control commands. While opting for a linear controller allows for a fully neuromorphic vision-to-control pipeline, it also means we have to make some assumptions. For instance, angles in pitch and roll should be small, and the optical flow variables taken as

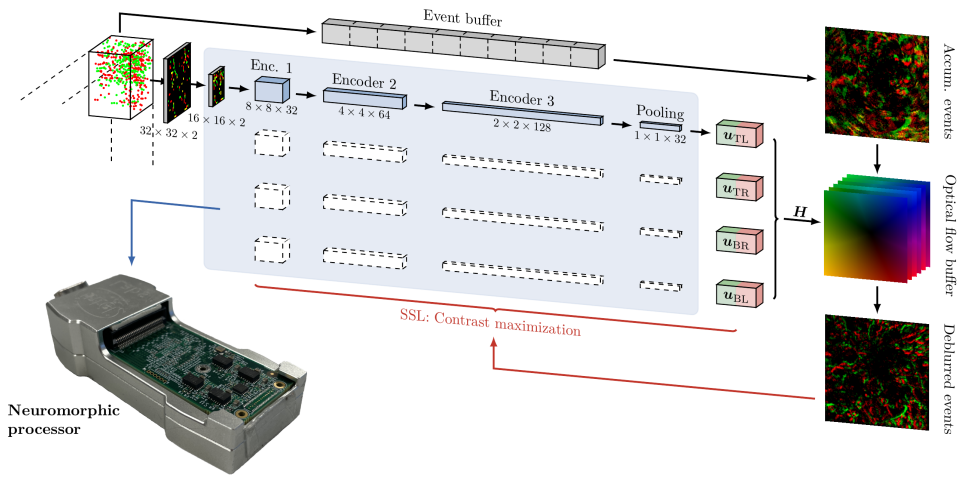


Figure 7.2: Overview of the spiking vision network. Running at approx. 200 Hz, events are accumulated (max. 90 events per corner ROI) and then fed through the vision network consisting of three encoders (kernel size 3×3, stride 2) and a spiking pooling layer. Spikes are decoded into two floats representing flow for that corner ROI. This network is replicated to the three other ROIs, in order to end up with four ROI optical flows vectors. During training, these are used in a homography transformation to derive dense flow, which is then used for the self-supervised loss. The full network is running on the neuromorphic processor during real-world flight tests.

input should be derotated in pitch and roll [72, 236]. Furthermore, we should keep in mind that a linear controller will be unable to compensate for any drift or steady-state offset through integration. We show that, despite all this, we are able to successfully perform control of a flying drone.

We split the training of our vision-to-control pipeline into two separate frameworks. On the one hand, the vision part of the pipeline, in charge of mapping input events to optical flow, is trained in a self-supervised fashion using the contrast maximization framework [96, 97]. The idea behind this approach is that, by compensating for the spatiotemporal misalignments among the events triggered by a moving edge (i.e., event deblurring), one can retrieve accurate optical flow information. In this work, we use the formulation proposed in [102] (Chapter 5 of this dissertation) and shown in Fig. 7.2. Corner ROI events within non-overlapping temporal windows of 5 milliseconds are processed sequentially by our spiking networks, which provide optical flow estimates at every timestep. Only during training, we use the motion information of the four corners ROIs to parameterize a homography transformation that, under the assumption of static planar surface, allows us to retrieve dense optical flow, as in [78, 309–311]. Following [102], we accumulate event and optical flow tuples over multiple timesteps for contrast maximization to be a robust self-supervisory signal, and only compute the deblurring loss function and perform a backward pass through the networks (using backpropagation through time) once 25 milliseconds of event data have been processed. To cope with the non-differentiable spiking function of our neurons, we use surrogate gradients [37].

On the other hand, the control part of the network, consisting of a linear mapping from the motion of the four corner ROIs to thrust and attitude control commands, is trained

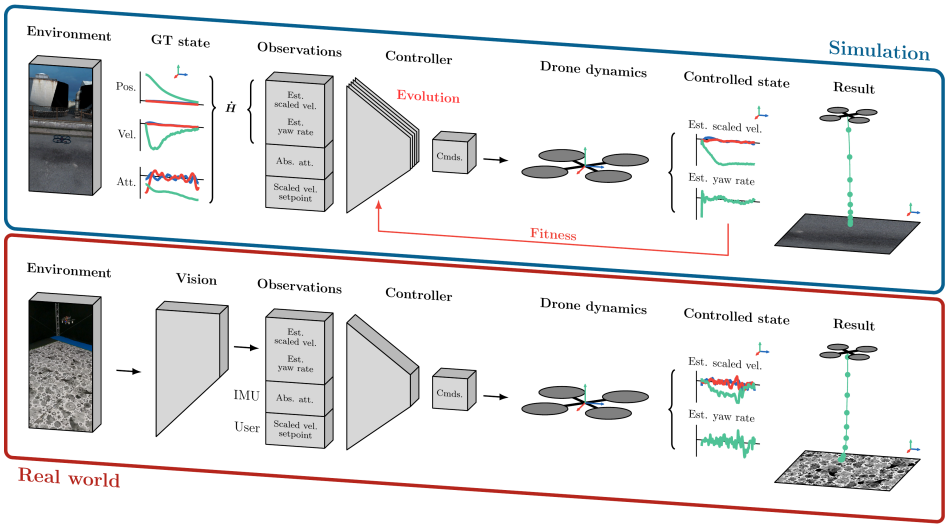


Figure 7.3: Overview of the control pipeline for simulation and real-world tests. During training in simulation, we construct visual observables (scaled velocities and yaw rate) from ground-truth using the continuous homography transform. The control decoding takes these observables together with roll and pitch and a setpoint to output commands, which control the drone dynamics. We train the controller using evolution based on a fitness signal that quantifies how well the controller can follow setpoints for horizontal and vertical flight. In the real world, we receive flows of the corner ROIs from the vision network, transform these to control commands in a single matrix multiplication, and send these commands to the autopilot.

7

in a drone simulator using an evolutionary algorithm. Evolutionary algorithms work by evaluating all the individuals in a population, where the best-performing (or fittest) individuals are varied upon to form the population of the next generation. Over generations, the individuals will get an increasingly high fitness, which in our case means that they become better at ego-motion control. Fig. 7.3 gives an overview of the simulator setup used in evolution. To get around the need to incorporate an event-based vision pipeline in simulation, we use the ground-truth state of the simulated drone to generate the expected flows per corner ROI using the continuous homography transform [312], and use these to construct the scaled velocity (i.e., velocity divided by height above ground) and yaw rate estimates that make up the visual observables of the camera's ego-motion. The velocity is divided by height, as optical flow vectors capture the *ratio* of velocity and distance [72, 236]. The inputs to the linear control mapping are then these visual observables, absolute roll and pitch (from the drone's inertial measurement unit or IMU) and a desired setpoint for the visual observables. The outputs of the controller (i.e., desired collective thrust, pitch and roll angles and yaw rate) are subsequently applied to the simulated drone model in order to control it. During evolution, the fitness of a controller is determined based on the accumulated visual observable error in an evaluation. We evaluate each of the individuals in the population on a set of (repeated) setpoints representing horizontal and vertical flight, create offspring through random mutations, and select the best individuals for the next generation. The trained controller is transferred directly to the real robot, without any retraining.

7.2 Method

Here, we explain the main components of the proposed fully-neuromorphic vision-to-control pipeline, starting with the neuron model of our SNN and how this is trained in a self-supervised fashion using real event data. Next, we describe how the vision-based state estimate can be used for navigation, and how we train a controller on top of it.

7.2.1 Simulating the on-chip spiking neuron model

In this study, we utilize a spiking neuron model based on the current-based leaky-integrate-and-fire (CUBA-LIF) neuron, whose membrane potential U and synaptic input current I at timestep t can be written as:

$$U_i^t = \alpha_U(1 - S_i^{t-1})U_i^{t-1} + I_i^t \tag{7.1}$$

$$I_i^t = \alpha_I I_i^{t-1} + \sum_j W_{ij}^{\text{ff}} S_j^t + W_{ii}^{\text{rec}} S_i^{t-1} \tag{7.2}$$

where j and i denote presynaptic (input) and postsynaptic (output) neurons within a layer, $S \in \{0, 1\}$ a neuron spike, and W^{ff} and W^{rec} feedforward and self-recurrent connections (if any), respectively. The decays (or leaks) of the two internal state variables of this neuron model are learned, and are denoted by α_U and α_I. A neuron fires an output spike if the membrane potential exceeds a threshold θ, which is also learned. The firing of a spike triggers a hard reset of the membrane potential. Note that, in this work, all neurons within a layer share the same decays and firing threshold.

Neurons on the Loihi neuromorphic processor also follow the CUBA-LIF model [14], however, several considerations must be taken into account to accurately simulate these on-chip neurons. Firstly, the two states variables are quantized in the integer domain. Hence, the parameters associated with these variables are also quantized in the same way: $w \in [-256 \,..\, 256 - \Delta w]$ with Δw being the quantization step for the synaptic weights, $\alpha_{\{U,I\}} \in [0 \,..\, 4096]$ for the decays, and $\theta \in [0 \,..\, 131071]$ for the threshold. We follow this quantization scheme with $\Delta w = 8$ (6-bit weights) in the simulation and training of our neural networks. Secondly, to emulate the arithmetic left (bit) shift operations carried out by the processor when updating the neuron states, we perform a rounding towards zero operation after the application of the decays. Taking these aspects into consideration, we obtain a matching score of 100% between the simulated and the on-chip spiking neurons. We use quantization-aware training (quantized forward pass, floating-point backward pass) to minimize the performance loss of our SNN when deployed on Loihi.

As surrogate gradient for the spiking function σ, we opt for the derivative of the inverse tangent $\sigma'(x) = \text{aTan}' = 1/(1 + \gamma x^2)$ [40], with $\gamma = 10$ being the surrogate width and $x = U - \theta$.

7.2.2 Four-point parametrization to estimate homography

Assuming that $\boldsymbol{x} = [x, y, 1]^T$ and $\boldsymbol{x'} = \left[x', y', 1\right]^T$ are two undistorted corresponding points from a planar scene expressed in homogeneous coordinates and captured by a pinhole camera at different time instances, a planar homography transformation is a linear projective transformation that maps $\boldsymbol{x} \leftrightarrow \boldsymbol{x'}$ such that:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}; \quad \text{with } \boldsymbol{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \tag{7.3}$$

where $\boldsymbol{H}$ is a 3x3 non-singular matrix, further referred to as the homography matrix, which is characterized by eight degrees of freedom and is defined up to a scale factor λ, and from which we obtain the normalized form by setting $h_{33} = 1$.

From Eq. 7.3, we can formulate $\boldsymbol{A}_k \boldsymbol{h} = \boldsymbol{b}_k$, a system of linear equations for the k-th point correspondence, where:

$$\boldsymbol{A}_k = \begin{bmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y \end{bmatrix} \tag{7.4}$$

$$\boldsymbol{h} = \begin{bmatrix} h_{11} & h_{12} & h_{13} & h_{21} & h_{22} & h_{23} & h_{31} & h_{32} \end{bmatrix}^T \tag{7.5}$$

$$\boldsymbol{b}_k = \begin{bmatrix} x' & y' \end{bmatrix}^T \tag{7.6}$$

As shown in Fig. 7.2A, our vision network predicts the displacement of the corner ROI pixels in a certain time window. Using this information, we can solve for the components of the homography matrix through $\boldsymbol{h} = \boldsymbol{A}^{-1}\boldsymbol{b}$, with $\boldsymbol{A}$ and $\boldsymbol{b}$ being the result of the concatenation of the individual $\boldsymbol{A}_k$ and $\boldsymbol{b}_k$ of each point correspondence $\forall k \in \{\text{TL}, \text{TR}, \text{BR}, \text{BL}\}$, resulting in a determined system of equations. This approach is referred to as the four-point parametrization of the homography transformation [309], and it has proved to be successful in the event-camera literature for robotics applications [78, 313].

Once the homography matrix is estimated, we can estimate a dense (i.e., per-pixel) optical flow map as follows:

$$\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{H}) = \begin{bmatrix} u(\boldsymbol{x}, \boldsymbol{H}) \\ v(\boldsymbol{x}, \boldsymbol{H}) \\ 1 \end{bmatrix} = \boldsymbol{H} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} - \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{7.7}$$

which encodes the displacement of pixel $\boldsymbol{x}$ in the time window of $\boldsymbol{H}$. With a slight abuse of notation, $\boldsymbol{u}$ further denotes optical flow in Euclidean coordinates.

7.2.3 Self-supervised learning of event-based optical flow

To train our spiking architecture to estimate the displacement of the pixels of the four corner ROIs in a self-supervised fashion, we use the contrast maximization framework for motion compensation [96, 97]. Assuming constant illumination, accurate optical flow information is encoded in the spatiotemporal misalignments among the events triggered by a moving edge (i.e., blur). To retrieve it, one has to learn to compensate for this motion (i.e., deblur the event partition) by transporting the events through space and time. Once we get a per-pixel optical flow estimate $\boldsymbol{u}(\boldsymbol{x}, \boldsymbol{H})$ from Eq. 7.7, we can propagate the events to a reference time t_{ref} through the following linear motion model:

$$\boldsymbol{x}'_i = \boldsymbol{x}_i + (t_{\text{ref}} - t_i)\boldsymbol{u}(\boldsymbol{x}_i, \boldsymbol{H}) \tag{7.8}$$

where t and t_{ref} are normalized relative to the time window between $\boldsymbol{x}$ and $\boldsymbol{x}'$. The result of aggregating the propagated events is referred to as the image of warped events (IWE) at t_{ref}, and it having a high contrast indicates good motion compensation/deblurring.

As loss function, we use the reformulation from [102] of the focus objective function based on the per-pixel and per-polarity average timestamp of the IWE [33, 161]. The lower this metric, the better the event deblurring and hence the more accurate the estimated optical flow. We generate an image of the per-pixel average timestamp for each polarity p' via bilinear interpolation:

$$\begin{aligned} T_{p'}(\boldsymbol{x};\boldsymbol{u}|t_{\text{ref}}) &= \frac{\sum_j \kappa(x - x'_j)\kappa(y - y'_j)t_j}{\sum_j \kappa(x - x'_j)\kappa(y - y'_j) + \epsilon} \\ \kappa(a) &= \max(0, 1 - |a|) \\ j &= \{i \mid p_i = p'\}, \qquad p' \in \{+, -\}, \qquad \epsilon \approx 0 \end{aligned} \tag{7.9}$$

Following [102], we first scale the sum of the squared temporal images resulting from the warping process with the number of pixels with at least one warped event:

$$\mathcal{L}_{\text{contrast}}(t_{\text{ref}}) = \frac{\sum_{\boldsymbol{x}} T_+(\boldsymbol{x};\boldsymbol{u}|t_{\text{ref}})^2 + T_-(\boldsymbol{x};\boldsymbol{u}|t_{\text{ref}})^2}{\sum_{\boldsymbol{x}} [n(\boldsymbol{x}') > 0] + \epsilon} \tag{7.10}$$

where $n(\boldsymbol{x}')$ denotes a per-pixel event count of the IWE.

As in [33, 100, 102], we perform the warping process both in a forward ($t_{\text{ref}}^{\text{fw}}$) and in a backward fashion ($t_{\text{ref}}^{\text{bw}}$) to prevent temporal scaling issues during backpropagation. The total loss used to train our event-based optical flow networks is then given by:

$$\mathcal{L}_{\text{contrast}} = \mathcal{L}_{\text{contrast}}(t_{\text{ref}}^{\text{fw}}) + \mathcal{L}_{\text{contrast}}(t_{\text{ref}}^{\text{bw}}) \tag{7.11}$$

$$\mathcal{L}_{\text{flow}} = \mathcal{L}_{\text{contrast}} + \lambda_{\mathcal{L}} \mathcal{L}_{\text{smooth}} \tag{7.12}$$

where $\mathcal{L}_{\text{smooth}}$ is a Charbonnier smoothness prior [162] applied in the temporal domain to subsequent per-corner-ROI optical flow estimates, while $\lambda_{\mathcal{L}}$ is a scalar balancing the effect of the two losses. We empirically set this weight to $\lambda_{\mathcal{L}} = 0.1$.

As discussed in [102, 103], there has to be enough linear blur in the accumulated input event partition for this loss function to be a robust supervisory signal [97, 256]. Since we process the event stream sequentially, with only a few events being considered at each forward pass, we define the so-called training partition $\boldsymbol{\varepsilon}_{k \longrightarrow k+R}^{\text{train}} \doteq \{(\boldsymbol{\varepsilon}_i^{\text{inp}}, \hat{\boldsymbol{u}}_i)\}_{i=k}^{k+R}$, which is a buffer that gets populated every forward pass with the input events and their corresponding optical flow estimates. This is illustrated in Fig. 7.2A. At training time, we perform a backward pass with the content of the buffer using backpropagation through time once it contains 5 successive event-flow tuples (i.e., 25 milliseconds of event data), after which we update the model parameters, detach its states from the computational graph, and clear the buffer. Note that the selection of input and training partition lengths represents deliberate design choices [314], made in alignment with our target execution frequency of 200 Hz, the fact that we do not have direct connectivity between the event camera and the neuromorphic processor, and the statistical attributes of our dataset. We use a batch size of 16 and train until convergence with the Adam optimizer [173] and a learning rate of 1e-4.

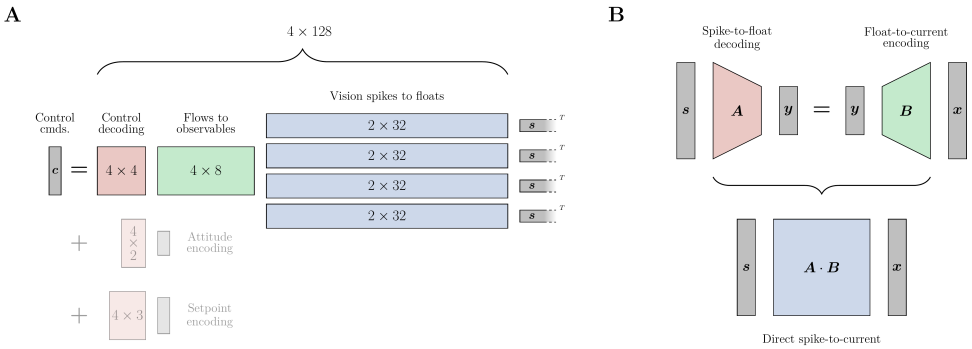


Figure 7.4: Merging linear transformations. (*A*) We go directly from output spikes $\boldsymbol{s}$ of the vision network to control commands $\boldsymbol{c}$ in a single linear decoding by multiplying the involved linear transformation matrices. (*B*) The same principle can be applied to connect two separately trained spiking networks in a spiking manner, from spikes $\boldsymbol{s}$ to currents $\boldsymbol{c}$, suitable for neuromorphic hardware.

7.2.4 From a vision-based state estimate to control

The corner ROI flows $\left[\boldsymbol{u}_{\text{TL}}^T, \boldsymbol{u}_{\text{TR}}^T, \boldsymbol{u}_{\text{BR}}^T, \boldsymbol{u}_{\text{BL}}^T\right]^T \in \mathbb{R}^{8\times 1}$ resulting from the vision-based state estimation can be used to control the drone. More specifically, we can transform the flows to visual observable estimates [72], consisting of scaled velocities $\hat{\boldsymbol{\nu}}^{\mathcal{C}} \in \mathbb{R}^{3\times 1}$ and yaw rate $\hat{\omega}_z^{\mathcal{C}}$ in the camera frame $\mathcal{C}$, as follows [236]:

$$\boldsymbol{u} = \begin{bmatrix} -1 & 0 & x & x \\ 0 & -1 & y & -y \end{bmatrix} \begin{bmatrix} \boldsymbol{\nu}^{\mathcal{C}} \\ \omega_z^{\mathcal{C}} \end{bmatrix} \tag{7.13}$$

where $\boldsymbol{u}$ is the optical flow of a world point with pixel array coordinate $\boldsymbol{x} = [x, y]^T$, and where it is assumed that 1) the scene is static and planar, 2) angles in pitch and roll are small and 3) optical flow is derotated in pitch and roll (meaning the observed flow is only due to translation and yawing). Concatenating Eq. 7.13 for all four corners of the field of view ($\boldsymbol{u}_k, \boldsymbol{x}_k \; \forall \, k \in \{\text{TL}, \text{TR}, \text{BR}, \text{BL}\}$) allows us to do a least-squares estimation of the scaled velocities $\hat{\boldsymbol{\nu}}^{\mathcal{C}}$ and the yaw rate $\hat{\omega}_z^{\mathcal{C}}$, which can then be transformed to the body frame $\mathcal{B}$. To perform control, we can let a user select setpoints $\boldsymbol{\nu}_{\text{sp}}^{\mathcal{B}}$ and $\omega_{\text{z,sp}}^{\mathcal{B}}$, and use a trained or manually tuned controller to minimize the difference between the estimated visual observables and their setpoints.

Because Eq. 7.13 is a linear transformation, it can be "merged" with other transformations if these are also linear. This holds for the decoding from spikes to corner ROI flows in the vision SNN, meaning that we can use a single linear transformation from spikes to control commands if we use a linear controller. In a similar fashion, we can use this idea to connect separately trained SNNs, merging their linear decodings and encodings. If both are implemented on neuromorphic hardware, this would mean that no off-chip transfer is necessary. Fig. 7.4 illustrates these concepts.

7.2.5 Training control in simulation

We perform control by linearly transforming the visual observable estimates $\hat{\boldsymbol{\nu}}^{\mathcal{B}} \in \mathbb{R}^{3\times 1}$ and $\hat{\omega}_z^{\mathcal{B}}$, the drone's absolute roll $|\phi|$ and pitch $|\theta|$ and the scaled velocity setpoint $\boldsymbol{\nu}_{\text{sp}}^{\mathcal{B}} \in \mathbb{R}^{3\times 1}$

to a control command $\boldsymbol{c} \in \mathbb{R}^{4\times1}$, which consists of an upward, mass-normalized collective thrust offset from hover $\bar{f}_{0,c}$ in the body frame $\mathcal{B}$, a roll angle ϕ_c and pitch angle θ_c, and a yaw rate $\omega^{\mathcal{B}}_{z,c}$, in order to reach a certain setpoint of scaled velocities $\boldsymbol{v}^{\mathcal{B}}_{\text{sp}}$ and yaw rate $\omega^{\mathcal{B}}_{z,\text{sp}}$ (always 0).

The control part is trained separately from the vision part because of the cost of accurately simulating event-based camera inputs (this needs subpixel displacements between frames, hence high frame rate for fast motion). Simulation is done with a modified version of the drone simulator Flightmare [315]. To mimic the output of the vision-based state estimation network, we first compute the ground-truth continuous homography [312, 316] from the state of the drone:

$$\dot{\boldsymbol{H}} = \boldsymbol{K}\left([\boldsymbol{\omega}^{\mathcal{C}}]_\times + \frac{1}{p_z^{\mathcal{WC}}}\boldsymbol{v}^{\mathcal{C}}(\boldsymbol{e}^{\mathcal{W}}_{-z})^T\right)\boldsymbol{K}^{-1} \tag{7.14}$$

where $\dot{\boldsymbol{H}}$ is the continuous homography, $\boldsymbol{K}$ is the camera intrinsic matrix, $[\boldsymbol{\omega}^{\mathcal{C}}]_\times \in \mathbb{R}^{3\times3}$ is a skew-symmetric matrix representing infinitesimal rotations, $p_z^{\mathcal{WC}}$ is the Z-component of the position vector from the world frame $\mathcal{W}$ to the camera frame $\mathcal{C}$ (representing perpendicular distance from the ground plane to the camera), $\boldsymbol{v}^{\mathcal{C}}$ is the velocity of the camera, and $\boldsymbol{e}^{\mathcal{W}}_{-z}$ is the unit vector in the negative Z-direction of the world frame. To obtain angular rates and velocities in the camera frame, we use the camera extrinsics, consisting of a rotation $\boldsymbol{R}^{\mathcal{CB}}$ and a translation $\boldsymbol{T}^{\mathcal{CB}}$:

$$\boldsymbol{\omega}^{\mathcal{C}} = \boldsymbol{R}^{\mathcal{CB}}\boldsymbol{\omega}^{\mathcal{B}} \tag{7.15}$$

$$\boldsymbol{v}^{\mathcal{C}} = \boldsymbol{R}^{\mathcal{CB}}\left(\boldsymbol{v}^{\mathcal{B}} + [\boldsymbol{\omega}^{\mathcal{B}}]_\times \boldsymbol{T}^{\mathcal{CB}}\right) \tag{7.16}$$

where the right-hand sides of Eqs. 7.15 and 7.16 are known from the simulator. Next, we use the continuous homography to get the flow of the four corners [312, 316]:

$$\begin{bmatrix}\boldsymbol{u}_k \\ 1\end{bmatrix} = -\left(\boldsymbol{1} - \boldsymbol{x}_k(\boldsymbol{e}^{\mathcal{W}}_{-z})^T\right)\dot{\boldsymbol{H}}\boldsymbol{x}_k \tag{7.17}$$

where $\boldsymbol{u}_k$ is the flow in Euclidean coordinates, $\boldsymbol{1}$ is the identity matrix, and $\boldsymbol{x_k} = [x_k, y_k, 1]^T$ is the projection of the world points in the corners of the field of view onto the pixel array in homogeneous coordinates. We add $\mathcal{N}(0, 0.025)$ noise to the flows $\boldsymbol{u}_k$ (based on a characterization of the vision SNN). Eq. 7.13 is subsequently used to go from flows of the corner ROIs to visual observables in the camera frame, which is then transformed back to the body frame for control.

We use a mutation-only evolutionary algorithm with a population size of 100 to evolve the weights of the linear controller matrix $\in \mathbb{R}^{4\times9}$, whose initial values are drawn from $\mathcal{U}(-0.1, 0.1)$. More specifically, we generate offspring by adding mutations drawn from $\mathcal{N}(0, 0.001)$ to all parameters of each parent and then evaluate the fitness of both parents and offspring. The next generation is comprised of the best 100 individuals, and we repeat this process until convergence (approx. 25k generations). We use Flightmare to assess fitness at flying various visual observable setpoints. Every individual is evaluated across a set of 16 setpoints, with each scaled velocity setpoint $\boldsymbol{v}^{\mathcal{B}}_{\text{sp}}$ having at most one nonzero element $\in \{\pm0.2, \pm0.5, \pm1.0\}$ 1/s, skipping the positive setpoints for the Z-direction, and

including hover. The yaw rate setpoint is set to $\omega^{\mathcal{B}}_{z,\text{sp}} = 0$ for all. Each setpoint is repeated ten times, meaning a total of 160 evaluations per individual. Fitness F is computed as:

$$F = \frac{1}{N_{\text{eval}}} \sum_{i \in N_{\text{eval}}} \sum_{j \in N_{\text{steps}}} \boldsymbol{w} \cdot \left(\boldsymbol{v}^{\mathcal{B}}_{\text{sp},i} - \begin{bmatrix} \hat{v}^{\mathcal{B}}_x \\ \hat{v}^{\mathcal{B}}_y \\ v^{\mathcal{W}}_z \end{bmatrix}_j \right)^2 + \left(\hat{\omega}^{\mathcal{B}}_z \right)^2 \tag{7.18}$$

Here, N_{eval} is the number of evaluations, $N_{\text{steps}} = 1000$ is the number of steps per evaluation, and $\boldsymbol{w} = [1, 1, w_z]^T$ is a vector weighing the fitness for different axes, where we set $w_z = 10$ for setpoints where $v^{\mathcal{B}}_{z,\text{sp}} = 0$. Note that, for the Z-direction, we use the ground-truth scaled velocity in the world frame $v^{\mathcal{W}}_z$ instead of the one in the body frame, as the latter is zero in the case of the drone ascending or descending at a slope equal to its attitude, and would hence go unpunished, leading to extra vertical drift. Furthermore, if the simulated drone goes out of bounds or crashes before the end of an evaluation, it is reset without any additional fitness penalty.

We use domain randomization [317] to obtain a more robust controller and reduce the reality gap: for each of the ten repeats, a random constant bias $\mathcal{U}(-0.001, 0.001)$ rad is added to the absolute pitch and roll received by the control layer. This bias is shared among the population to keep things fair. Furthermore, for each of the 160 evaluations per individual, we randomly vary the initial position $\boldsymbol{p}^{\mathcal{WB}} = [0, 0, 2]^T + \mathcal{U}^{3\times1}(-1, 1)$ m (except for horizontal setpoints, where we fix $p^{\mathcal{WB}}_z$ to 1.5 m, due to the linear nature of the controller we have here), initial velocity $\boldsymbol{v}^{\mathcal{WB}} \sim \mathcal{U}^{3\times1}(-0.02, 0.02)$ m/s, initial attitude quaternion $\boldsymbol{q}^{\mathcal{WB}} \sim \mathcal{U}^{4\times1}(-0.02, 0.02)$ (normalized), and initial angular rates $\boldsymbol{\omega}^{\mathcal{B}} \sim \mathcal{U}^{3\times1}(-0.02, 0.02)$ rad/s.

We modify Flightmare to include drag $\boldsymbol{f}_{\text{drag}}$ occurring as a result of translational motion, and we take it to be acting in the so-called "flat-body" frame $\mathcal{B}'$, which is the body frame rotated by the roll and pitch of the drone, such that the Z-axis is aligned with the world Z-axis. Following [99], we use a drag model that is linear with respect to velocity in X and Y, but using a drag coefficient $k_{v,x} = k_{v,y} = 0.5$. This results in the following:

$$\boldsymbol{f}^{\mathcal{B}}_{\text{drag}} = -\boldsymbol{R}^{\mathcal{BB}'} \left(\begin{bmatrix} k_{v,x} \\ k_{v,y} \\ 0 \end{bmatrix} \circ \boldsymbol{R}^{\mathcal{B}'\mathcal{B}} \boldsymbol{v}^{\mathcal{B}} \right) \tag{7.19}$$

The outputs of the linear controller $\boldsymbol{c} \in \mathbb{R}^{4\times1}$ are clamped to $[-1, 1]$ and fed to different parts of the cascaded low-level (thrust, attitude and rate) controllers. To accommodate some of the shortcomings of the linear controller, we compensate thrust for the attitude of the drone. Lastly, note that all dynamics equations are integrated with 4$^{\text{th}}$-order Runge-Kutta with a timestep of 2.5 ms. The frequency of the simulation is 50 Hz.

7.2.6 Hardware setup

Real-world experiments were performed with a custom-built quadrotor carrying the event-based camera (DAVIS240C), a single-board computer (UP Squared) and a neuromorphic processor (Intel Kapoho Bay with two Loihi neuromorphic research chips). A high-level overview can be found in Fig. 7.1, while all components are listed in Table 7.1. We use PX4 as autopilot firmware, and ROS for communication. More specifically, events coming

Component	Product	Mass [g]	~Power [W]
Frame	GEPRC Mark 4 225 mm		
Motor	Emax 2306 Eco II Series		
Propellor	Ethix S5 5 inch	508	259
Flight Controller	Pixhawk 4 Mini		
ESC	SpeedyBee 45A BL32 4in1		
Battery	Tattu FunFly 1800mAh 4S	195	-
Single-board computer	UP Squared ATOM Quad Core 08/64	202	18
Event-based camera	DAVIS240C	27	1
Neuromorphic processor	Intel Loihi, Kapoho Bay form factor	62	1

Table 7.1: List of hardware components used for the real-world test flights.

from the event-based camera are passed to the UP Squared over USB using ROS1. These events are processed (downsampling, cropping, limiting to 90 per image corner ROI) on the UP Squared, and sent as spikes to the vision network running on the Kapoho Bay over USB. After processing, the output spikes are sent back over USB to the UP Squared, where they are decoded into flows for each corner ROI. The ROI flows are then published by a ROS1 node, and sent to ROS2 over a ROS1-ROS2 bridge. The linear controller (or PI controller, for that matter) and the processing around it, running as a ROS2 node, takes the ROI flows together with the attitude estimate coming from PX4 (IMU) and the setpoint provided by the user, and outputs the control command. This command is then sent over ROS2 to PX4, and processed by the low-level controllers there. ROS2 makes use of RTPS (real-time publish-subscribe) for communication, which allows for high-frequency and high-bandwidth messaging between the UP and PX4, meaning our entire pipeline can run at 200 Hz. For position control between test runs, and as failsafe, we use an OptiTrack motion capture system.

7.3 Experiments

Because of the split between the vision and control parts of the pipeline, we can evaluate their performance separately. The estimated corner ROI flows of the vision part are compared against ground truth data obtained from a motion capture system, while the control part is evaluated in simulation. Connecting vision and control together, we then demonstrate the performance of our fully neuromorphic vision-to-control pipeline through real-world flight tests. To further illustrate the robustness of our vision-based state estimation, we perform real-world tests with changing setpoints, and tests in various lighting conditions. Lastly, we compare energy consumption against possible on-board GPU solutions.

7.3.1 Robust vision-based state estimation

To prevent reality-gap issues when simulating an event-based camera, we train and evaluate the vision part of our pipeline using real-world event sequences recorded with the same platform (i.e., drone and downward-facing event-based camera) and in the same indoor environment (i.e., static and planar, constant illumination). This dataset consists of

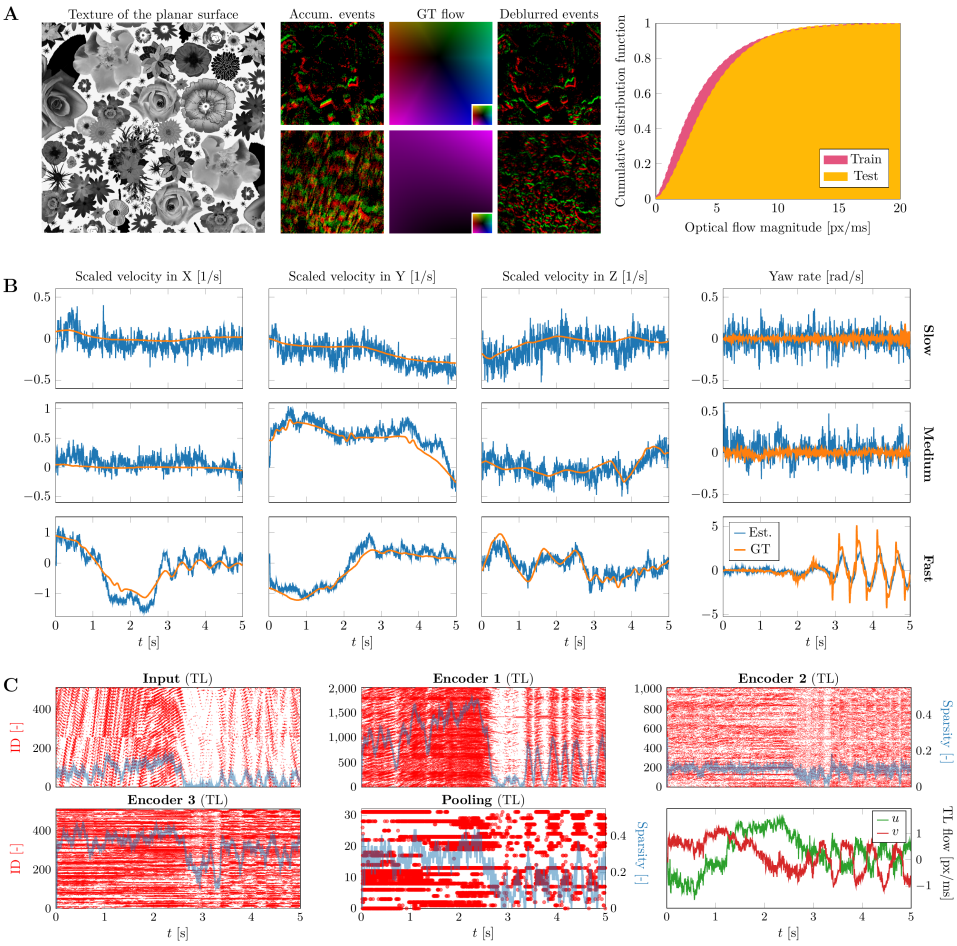


Figure 7.5: Overview of results for the vision-based state estimation. (*A*) Characteristics of the dataset for estimating planar optical flow. *Left:* Grayscale flower texture used to cover the floor. *Center:* Accumulated event windows showing the blur arising from motion, ground truth flow fields as determined with a motion tracking system and based on a flat floor assumption (only for evaluation), and the result when using the flow fields to deblur the event windows (only for illustration). *Right:* Ground-truth optical flow distributions for the training and test datasets. (*B*) Comparison of estimated and ground-truth visual observables for sequences with different motion speeds (slow, medium, fast). (*C*) Network activity resulting from the events in the top-left-corner ROI in the fast motion sequence.

approximately 40 minutes of event data, which we split into 25 minutes for training and 15 for evaluation, and its motion statistics are shown in Fig. 7.5A. In addition to the visual data, the ground truth pose (i.e., position and attitude) of the drone over time is provided at a rate of 180 Hz, and is used solely for evaluation. Examples of this ground truth, which can be converted to dense optical flow using the camera calibration, are shown in Fig. 7.5A alongside the floor texture of the indoor environment.

					$\cdot 10^{-2}$
	Full image	+2x Down.	+Corner ROI crop	+Limit events	+Loihi quant.
Conv-GRU ANN	5.88	5.56	5.61	5.72	-
Conv-RNN SNN	7.92	7.89	7.91	6.97	6.96
Self-RNN SNN (ours)	10.79	9.80	8.22	7.71	**8.34**

Table 7.2: Quantitative comparison between different vision architectures. Bottom right corner, in bold, indicates the final architecture. Architecture choices (row-wise) and design decisions (column-wise) impact test performance in terms of the average endpoint error (EPE↓, lower is better). Baseline architectures feature the same feedforward connectivity pattern as ours, but vary the neuron model and the type of recurrent connections.

We train our vision SNN with the self-supervised contrast maximization framework from [102] and a quantization-aware training routine that simulates the neuron and synapse models in the target neuromorphic hardware. Once this is done, we evaluate the performance of our spiking network on the task of planar event-based optical flow estimation using sequences with varying amounts of motion. Qualitative results are presented in Fig. 7.5B, where the estimated visual observables (scaled velocities and yaw rate, constructed from the estimated optical flow vectors at the image corner ROIs) are compared to their ground-truth counterparts. These results confirm the validity of our approach. Despite the architectural limitations of the proposed solution (e.g., spike-based processing, limited field of view, only self-recurrency, weight and state quantization) and the fact that it does not have access to ground-truth information during training, it is able to produce optical flow estimates that accurately capture the motion encoded in the input event stream, i.e., the ego-motion of the camera. This is especially remarkable for the shown fast sequence, where towards the end the camera is spinning with approximately 4 rad/s. Note that, similarly to any other optical-flow-based state-estimation solution, our SNN is subject to the aperture problem not only due to the limited receptive field of the corner ROIs but also because of the use of event cameras as vision sensors [18].

In Fig. 7.5C, we show the internal spiking activity of our vision SNN as it processes the top-left corner ROI from the fast sequence shown in Fig. 7.5B, along with the decoded optical flow vectors. These qualitative results provide insight into the type of processing carried out by the proposed architecture, which is spike-based and therefore sparse and asynchronous. Notably, despite the rapid motion in the input sequence, all layers of the SNN maintain activation levels below 50% of the available neurons. Note that the network was not explicitly trained to promote sparse activations. Furthermore, we can distinguish layers with activity levels that are highly correlated with the input activity (i.e., encoder 1 and pooling), while others rely on their explicit recurrent connections to maintain activity levels that are relatively independent of the input statistics (i.e., encoder 2 and encoder 3).

In Table 7.2, we provide a quantitative comparison of our solution with other similar recurrent architectures (that are not compatible or do not fit in the Intel Kapoho Bay), based on the average endpoint error (EPE↓, lower is better). This evaluation not only demonstrates the performance of our spiking network, but also assesses the impact of each mechanism that was incorporated into the pipeline to achieve a solution that could be deployed on Loihi at the target frequency of 200 Hz. Several conclusions can be drawn from these results. Firstly, the ANN outperforms its spiking variants by a large margin, and

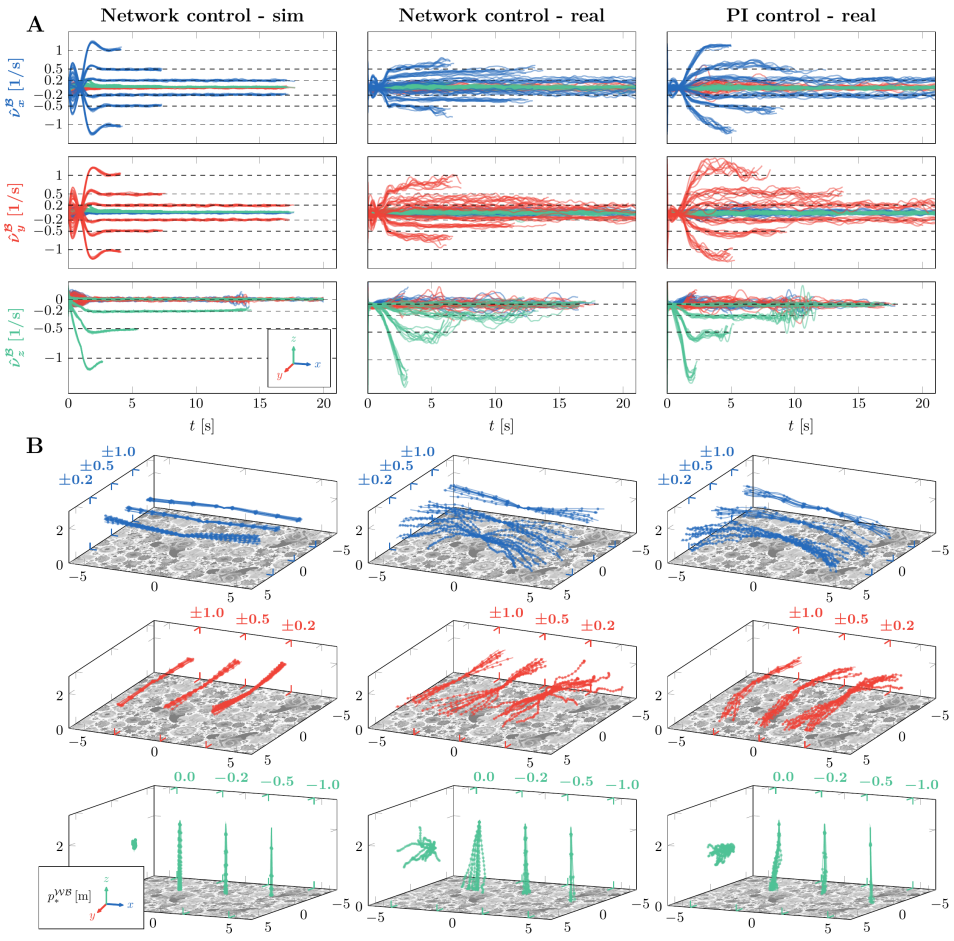


Figure 7.6: Comparison of results obtained in simulation and during real-world flight tests. (*A*) Estimated scaled velocities for 16 different setpoints in three axes, across three scenarios: linear network controller in simulation and the real world, and a hand-tuned proportional-integral (PI) controller in the real world. Real-world tests use the vision network to obtain visual observable estimates. Setpoints are nonzero in one direction, indicated with dashed lines. Rows represent the different motion axes. (*B*) 3D world position trajectories for the same flight tests. Each cube in (*B*) matches the plot in the corresponding location in (*A*).

self-recurrency is the weakest form of explicit recurrency among those tested. Secondly, deploying one architecture to each image corner ROI instead of processing the entire image space at once is beneficial for our architecture, while only having a slight detrimental effect on the baselines. Limiting the number of events that can be processed at once to 90 per corner ROI is also helpful for the evaluated SNNs, as it helps reduce the internal activity levels. Lastly, the incorporation of the Loihi-specific weight and state quantization leads to an error increase for our architecture.

7.3.2 Control through visual observables: From sim to real

Separately from the vision part, we train and evaluate the control part of our pipeline. This is a linear mapping from a visual observable estimate, absolute roll and pitch and a visual observable setpoint to thrust and attitude commands. The visual observable estimate is made up of scaled velocity $\hat{\boldsymbol{\nu}}^{\mathcal{B}}$ and yaw rate $\hat{\omega}_z^{\mathcal{B}}$; the visual observable setpoint consists of the corresponding setpoints $\boldsymbol{\nu}_{\text{sp}}^{\mathcal{B}}$ and $\omega_{z,\text{sp}}^{\mathcal{B}}$. A population of these linear mappings is evolved in simulation for a set of 16 scaled velocity and yaw rate setpoints. Each setpoint $\boldsymbol{\nu}_{\text{sp}}^{\mathcal{B}}$ has at most one nonzero element $\in \{\pm 0.2, \pm 0.5, \pm 1.0\}$ 1/s. In other words: they represent hover, vertical flight in the form of landing at three speeds (no ascending flight), and horizontal flight in four directions at three speeds. Unless mentioned otherwise, the setpoint for yaw rate $\omega_{z,\text{sp}}^{\mathcal{B}} = 0$. The first column of Fig. 7.6 shows the performance of the evolved linear network controller in simulation in terms of the estimated scaled velocities $\hat{\boldsymbol{\nu}}^{\mathcal{B}}$ (Fig. 7.6A) and the world position $\boldsymbol{p}^{\mathcal{WB}}$ over time (Fig. 7.6B) for all setpoints. The controller reaches the setpoint in all cases, and is capable of keeping the scaled velocities for the non-flight direction close to zero. Especially for $\nu_{*,\text{sp}}^{\mathcal{B}} = \pm 1.0$ 1/s, there is overshoot, but this can be expected given that this is a linear mapping without any kind of derivative control.

We get the second column of Fig. 7.6 by deploying this controller in the real world, and replacing the ground-truth visual observables with those estimated by the vision network. Looking at the scaled velocity plots for the different setpoints, we see that these become less noisy for higher setpoints and faster flight, as can also be seen from the 3D position plots. This is due to the fact that the signal-to-noise ratio of the vision-based state estimation increases with motion magnitude (little motion means most events are due to noise, as can be seen in Fig. 7.5). Also, the inertia of the drone provides some stability at higher speeds. Overall, the results demonstrate successful deployment of the fully neuromorphic vision-to-control pipeline. Nevertheless, apart from several setpoints (e.g., landings, $\nu_{\{x,y\},\text{sp}}^{\mathcal{B}} = \pm 0.2$ 1/s), the controller is not able to reach the desired setpoint: the steady-state error looks to be proportional to the setpoint magnitude. This can be attributed to the fact that while the controller is a *linear* mapping, the relationship between attitude angle and resulting forward/sideways velocity is *nonlinear* as a result of drag. Providing absolute attitude input to the network, and simulating the drag (as in [99]) during training turned out not to be enough to compensate. Furthermore, there can be mismatches between the dynamics of the simulated drone (body characteristics, motor dynamics) with which the controller was trained and the real drone on which the flight tests were performed, even though we abstracted the control outputs to attitude commands. Lastly, inaccuracies of the drag model can also be a source of error (in this case, it seems that drag was higher in reality than in simulation).

The third column of Fig. 7.6 shows the results obtained by connecting a hand-tuned proportional-integral (PI) controller to the vision-based state estimation. We compare this to the linear network controller. Looking at all directions and setpoints, we see that the PI controller reaches the setpoint faster than the network controller. For horizontal flight, the network controller is not at all able to reach the setpoint $\nu_{\{x,y\},\text{sp}}^{\mathcal{B}} = \pm 1.0$ 1/s and only just in the case of ±0.5 1/s, supposedly due to the limitations of linear control. The PI controller does not have this problem, as it can increment its control command to eliminate the steady-state error. For vertical flight, both the network and the PI controller have quite some overshoot for $\nu_{z,\text{sp}}^{\mathcal{B}} = -1.0$ 1/s. That this is so obvious, however, has to do with the

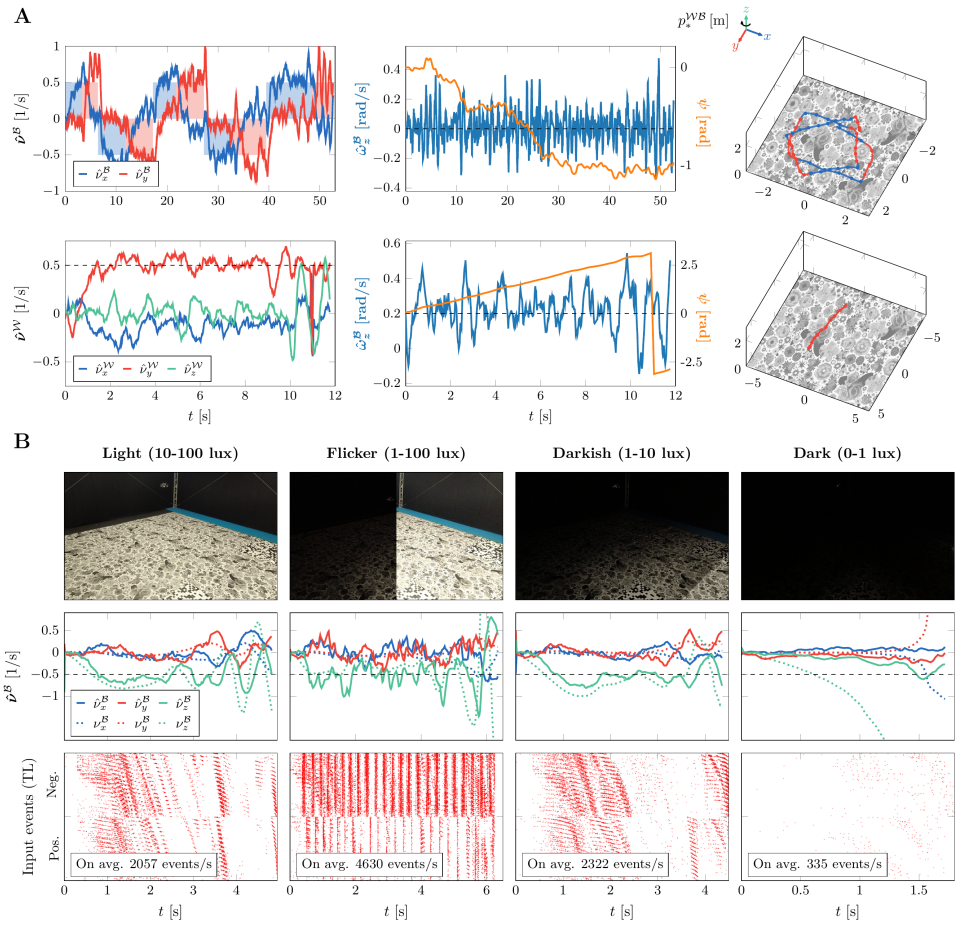


Figure 7.7: Additional results with vision network and proportional-integral (PI) controller. (*A*) *Top row*: alternating setpoints in X and Y (shaded areas) in order to fly a square. *Bottom row*: rotating the scaled velocity setpoint by the yaw angle, while maintaining a yaw rate setpoint of 0.2 rad/s (dashed line), leads to the drone spinning around its Z-axis while flying in a straight line. (*B*) Landing experiments with different lighting conditions. While flickering lights lead to many more events, visual observable estimates (and hence control) only diverge when it is so dark that there are almost no events.

fact that at such speeds from such heights (i.e., 2.5 m), the drone barely reaches the setpoint before reaching the ground, and therefore has little time to compensate for any overshoot (look at the PI controller for $v^{\mathcal{B}}_{z,\text{sp}} = -0.5$ 1/s; there overshoot is similar but is corrected shortly after). A slightly lower gain or a derivative term could help here.

7.3.3 Other examples of versatility and robustness

We can combine the vision-based state estimation with the PI controller to show the versatility and robustness of the former through various other tests, with the benefit of not having to include these in training for the controller network. Fig. 7.7A shows these

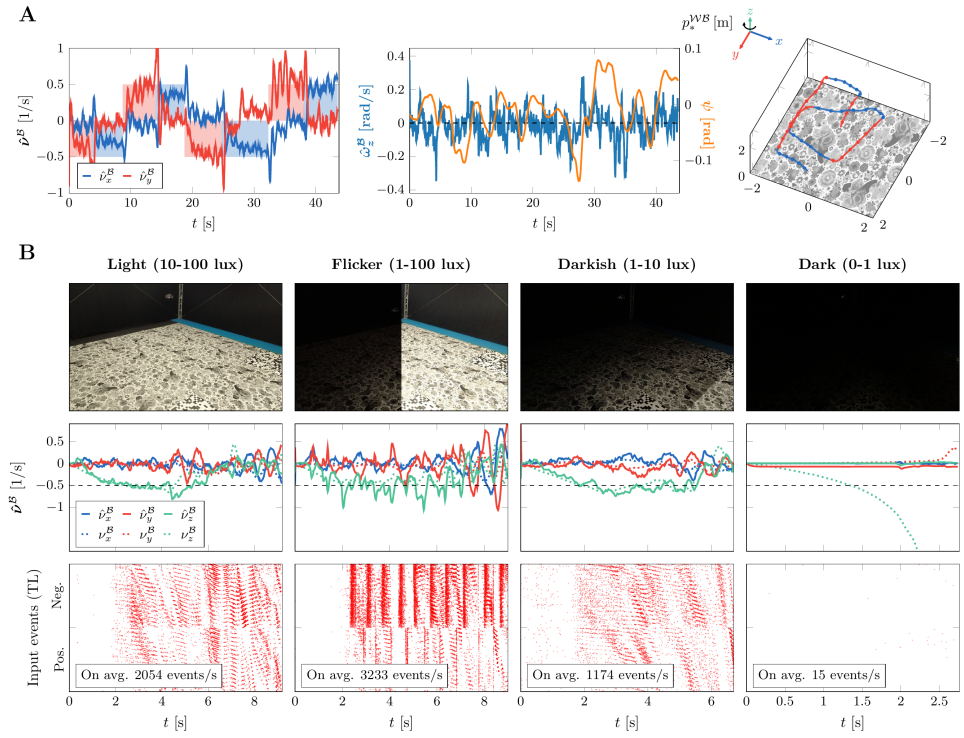


Figure 7.8: Additional results with vision network and linear network controller. (*A*) Alternating setpoints in X and Y (shaded areas) in order to fly a square. (*B*) Landing experiments with different lighting conditions. While flickering lights lead to many more events, visual observable estimates (and hence control) only diverge when it is so dark that there are almost no events.

tests. The top row displays the user alternating through different scaled velocity setpoints in X and Y (while keeping yaw constant) in order to let the drone fly a square. While the controller is able to reach the desired setpoint quite quickly, allowing for rather sharp corners, there is significant drift in yaw (0.97 rad), leading to a rotated second square.

The bottom row of Fig. 7.7A shows an experiment in which the drone has to fly in a straight line while spinning around its Z axis like a frisbee. With this experiment we aim to investigate how well the neuromorphic vision can separate rotational and translational optical flow, without relying on optical flow derotation with the help of gyro measurements. The drone receives a nonzero yaw rate setpoint $\omega^{\mathcal{B}}_{z,\text{sp}} = 0.2$ rad/s. In combination with a setpoint of $v^{\mathcal{B}}_{y,\text{sp}} = 0.5$ 1/s this would lead to the drone flying in a circle. To prevent this and achieve the frisbee-like spinning effect, we rotate $v^{\mathcal{B}}_{y,\text{sp}}$ by the yaw angle. The first and last plot show that this works: despite some drift in X and Z, the setpoints are followed well and the 3D position trajectory is quite straight. The second plot shows that the desired yaw rate is tracked well and that the yaw angle is constantly increasing.

Fig. 7.7B shows landing with divergence $v^{\mathcal{B}}_{z,\text{sp}} = -0.5$ 1/s for various lighting conditions (quantified with lux measurements). With these experiments we aim to investigate the

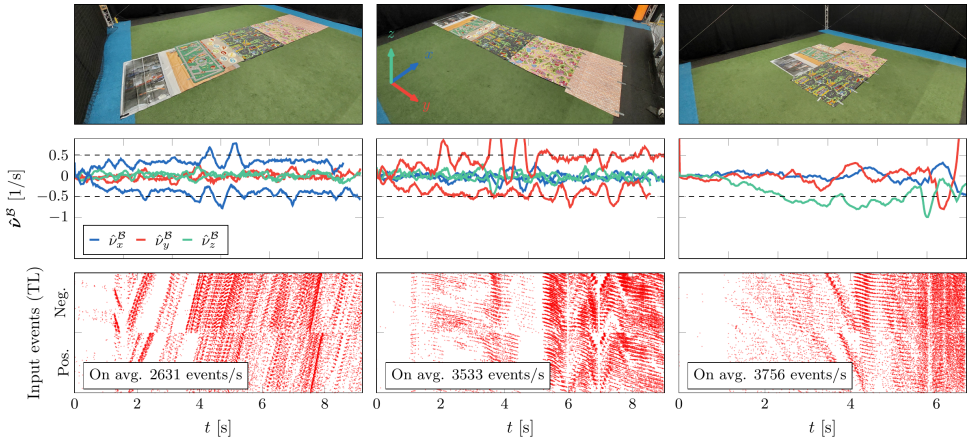


Figure 7.9: Results with vision network and linear network controller for various textures. Flight tests in X, Y and Z over variously textured mats show that the vision network is able to estimate optical flow regardless of texture. Input events are for the $\nu^{\mathcal{B}}_{\{x,y\},\text{sp}} = 0.5$ 1/s and $\nu^{\mathcal{B}}_{z,\text{sp}} = -0.5$ 1/s tests. Dashed lines show the nonzero setpoints in the motion axis.

robustness of the approach to wildly varying event statistics. For instance, in a darker environment, contrasts are less visible, which means that motion will generate many fewer events and that there will be more spurious, noisy events—not unlike our own human vision when we walk in the dark. When lights are switched on and off, this generates massive numbers of events that are unrelated to motion, hence violating the brightness constancy assumption underlying optical flow determination. The events for the top left corner ROI are shown in the bottom row of Fig. 7.7B. The results in the light and darkish settings look alike, but flickering lights lead to a large increase in events, while the darkest setting gives almost no events. As the middle row of plots shows, despite the challenging light conditions, the controller is able to track the setpoint (black dashed line) quite well, and the estimated scaled velocities approximate their ground thruths. Only the darkest setting poses a real problem for the state estimation: in that case, the estimated scaled velocities $\hat{\boldsymbol{\nu}}^{\mathcal{B}}$ diverge too much from the ground truth scaled velocities $\boldsymbol{\nu}^{\mathcal{B}}$ to perform a successful landing. Note that robustness to lighting conditions is independent of the performed maneuver; we chose landing as it involves a wider range of visual motion than horizontal flight, allowing us to better see the impact of lower-light conditions, and because divergence-based landings inherently lead to oscillations [73], which makes for a more challenging scenario in the case of flickering lights. For completeness, the disco, darkness, and squares experiments have also been performed with the neuromorphic controller. Results are shown in Fig. 7.8. In most cases, setpoint tracking performance is similar to that of the vision network and PI controller combination in Fig. 7.7. For the square-flying, the network controller even seems to outperform the PI controller in terms of yaw drift.

Lastly, to demonstrate that successful real-world flight transfers textures other than those the vision network was trained on, we successfully performed additional tests on other textured surfaces, as shown in Fig. 7.9. Oscillations in scaled velocities are similar to those observed above the training texture.

	Seq.	Static [W]	Dynamic [W]	Idle [W]	Running [W]	Delta [W]	inf/s	mJ/inf
Nahuku 32 (empty)	any	0.86	0.04	0.90	0.90	4 e-3	60496	71 e-6
Nahuku 32: SNN	slow	0.90	0.05	0.94	0.95	12 e-3	1637	7 e-3
	medium	0.90	0.04	0.94	0.95	8 e-3	411	21 e-3
	fast	0.90	0.04	0.94	0.95	7 e-3	274	27 e-3
Jetson Nano: SNN (5W)	slow	-	-	1.05	2.23	1.18	14	86.11
	medium	-	-	1.03	2.25	1.22	14	85.58
	fast	-	-	1.03	2.24	1.21	14	86.19
Jetson Nano: SNN (10W)	slow	-	-	1.04	2.98	1.93	26	75.25
	medium	-	-	1.06	2.98	1.92	26	75.35
	fast	-	-	1.04	2.99	1.95	25	76.52
Jetson Nano: ANN (5W)	slow	-	-	1.05	2.66	1.61	59	27.46
	medium	-	-	1.07	2.64	1.57	56	27.91
	fast	-	-	1.06	2.64	1.58	57	27.52
Jetson Nano: ANN (10W)	slow	-	-	1.04	3.30	2.27	83	27.36
	medium	-	-	1.07	3.33	2.26	80	28.09
	fast	-	-	1.07	3.30	2.23	80	27.80

Table 7.3: Approximate energy and power characteristics for various devices on three sequences: slow, medium and fast. On average, slow has 28.6 events/inf, medium has 106.9 events/inf, and fast has 186.6 events/inf. Delta power is the difference between idle and running (total) power, and is used to compute energy per inference. Dynamic power is the power needed for switching and short-circuiting, while static power is due to leakage. Together, these components contribute to the total running power. Nahuku is a board with 32 Loihi chips (Kapoho Bay has 2). A Nahuku configuration where no spikes are sent and only chips and cores are allocated (no synapses) is included as 'empty'. Jetson Nano has a low-power (5W) and high-power (10W) mode. One inference (inf) is the processing of one set of inputs by the network, resulting in an output or prediction. On Jetson Nano, we test both our vision SNN as well as the Conv-GRU ANN mentioned in Table 7.2.

7.3.4 Benchmarking inference speed and energy consumption

Table 7.3 shows a comparison in terms of power/energy and runtime between the Loihi neuromorphic processor and an NVIDIA Jetson Nano for running the vision network (both SNN as well as equivalent ANN) on sequences with varying amounts of motion and hence varying input event density. The SNN runs in hardware on Loihi and in software (PyTorch) on Jetson Nano. The tests for Loihi were performed on a Nahuku board, which contains 32 Loihi chips. We confirmed, insofar possible, that using two chips on Nahuku is representative of a Kapoho Bay (at least in terms of execution time), which is the two-chip form factor used on the drone. Still, neither of these benchmarks is completely representative of the tests performed in the real world: the benchmarks use data already loaded in memory, and therefore only quantify the processing by the network without any bottlenecks or impacts due to I/O and preprocessing, whereas the flight tests involve streaming event data that is coming in and is being processed in an online fashion. This shows in Loihi's execution frequencies in Table 7.3, which are well above the 200 inferences[1] per second (inf/s) achieved during flight tests.

Because Jetson Nano does not provide static and dynamic power components, we compare the difference between idle and running power, and use that to compute energy per inference. Loihi, depending on the sequence, outperforms Jetson Nano by three to

[1] An inference is the processing of one set of inputs by a machine learning model, such as a deep neural network, to produce an output or prediction. In our case, the set of inputs are the events occurring in a time window of 5 ms, and the predictions are the motion estimates of the four image corner ROIs.

four orders of magnitude, providing a one to two orders of magnitude improvement in execution frequency. Furthermore, the benefits of neuromorphic processing show in Loihi's increasing execution frequency as event sparsity increases (from fast to slow motion sequences). Because a GPU like Jetson Nano is not optimized to simulate SNNs, we also run an equivalent ANN (Conv-GRU with downsampling, corner crop ROI and event limiting from Table 7.2). The increased inference speed for the ANN on Jetson Nano confirms that this is indeed a more suitable architecture for GPUs. Nonetheless, while energy consumption per inference has decreased compared to the SNN on Jetson Nano, energy efficiency and inference speed still do not come close to those of the SNN on Loihi.

7.4 Conclusion

We presented the first fully neuromorphic vision-to-control pipeline for controlling a flying drone. Specifically, we trained a spiking neural network that takes in raw event-based camera data and produces low-level control commands. Real-world experiments demonstrated a successful sim-to-real transfer: the drone can accurately follow various ego-motion setpoints, performing hovering, landing, and lateral maneuvers—even under constant yaw rate.

Our study confirms the potential of a fully neuromorphic vision-to-control pipeline by running on board with an execution frequency of 200 Hz, spending only 27 µJ per network inference. A major question is whether such an impressive energy gain will make a difference on a system level even if future neuromorphic chips weigh in the order of grams and will have a negligible idle power. Compared to the power required for hovering, 277 W, a difference between 3 W and 7 mW (see Table 7.3) may seem like a small difference. However, on flying robots, such small differences can have substantial effects [318]. A heavier, more power-consuming computing unit does not only require more power from the battery, but it also needs to be lifted in the air. This requires a bigger battery and possibly bigger motors that themselves also have to be lifted. Moreover, as argued in [318], a lower latency, as accomplished with neuromorphic processing, allows for faster flight. This in turn leads to drones accomplishing their missions with less flight time. Still, the main point is not necessarily what you gain on a ~1-kilogram drone if you switch from a conventional embedded GPU to a lightweight neuromorphic processor (which is not negligible), but that neuromorphic processing will enable many more networks to run on such larger drones and even enable deep neural networks on much lighter platforms that cannot even carry an embedded GPU. An example of the latter are 30-gram flapping wing drones, which use ~6 W to fly[11].

To reach this potential to the fullest, the entire drone sensing, processing, and actuation hardware should be neuromorphic, from its accelerometer sensors to the processor and motors, allowing for sparse and event-driven/asynchronous computation all the way through. Because such hardware is currently not available, we have limited ourselves to the vision-to-control pipeline, ending at thrust and attitude commands.

Until then, advancements could come from improved I/O bandwidth and interfacing options for the neuromorphic processor and event camera. The current processor is limited to x86 host boards (preventing potentially lighter but equally performant ARM boards to be used), and can only be connected directly via AER to a specific model of event-based camera (this is of course also a limitation on the part of the available event cameras). While

the former is limiting for all works implementing neuromorphic hardware on constrained systems like drones, the latter is especially relevant to our advanced use case, where we reached the limits of the number of spikes that can be sent to and received from the neuromorphic processor at the desired high execution frequency of 200 Hz. To achieve this frequency, we had to 1) limit the number of events per input window to 90 per image corner ROI, and 2) limit ourselves to a linear network controller, which avoids having to send additional input spikes that encode the setpoint and attitude. Ultimately, further gains in terms of efficiency might be obtained by moving from digital neuromorphic processors to mixed-signal hardware, but this will pose even larger development and deployment challenges given the noise sensitivity of such hardware [296, 319].

Despite the above-mentioned limitations, the current work presents a substantial step towards neuromorphic sensing and processing for drones. The results are encouraging, because they show that neuromorphic sensing and processing may bring deep neural networks within reach of small autonomous robots. In time this may allow them to approach the agility, versatility and robustness of small animals such as flying insects.

Supplementary material

Video playlist of the approach: https://tinyurl.com/4edsypye

8

Conclusion

In this concluding chapter, we revisit and address the research questions and problem statement presented in Chapter 1, which have structured the work conducted throughout this dissertation. We then discuss the broader implications of our findings and the potential impact that they may have on the broader scientific research field. Finally, we explore several potential avenues for future research.

8.1 Answers to research questions

The first research question derived from our problem statement was formulated as follows:

> **RQ1**: How can fast, autonomous flight through gates be achieved with frame-based perception and conventional processing in a GPS-denied environment?

This question was addressed in Chapter 2 with the development of a robust and efficient vision-based navigation solution for autonomous drone racing. To enable fast and agile flight using conventional sensing and processing, a lightweight monocular pipeline was devised, focusing exclusively on gate information. This was achieved by leveraging the fact that the gates were the only objects in the environment whose approximate appearance, location, and orientation were known a priori. The first step in the proposed pipeline is to detect the corners of the next gate to be traversed according to the flight plan, in the image space. This is done by first segmenting gate pixels using an artificial neural network (ANN) and then searching for the corners in the resulting mask using a light active-vision algorithm. The corners that can be validated using the robot's prior expectations of the gate's appearance and geometry are then used to estimate the pose of the camera in the world frame with a perspective-n-point algorithm. These vision measurements are enhanced with model-based predictions through random sample consensus, resulting in the state estimates used to control the robot. To finalize, a risk-aware control strategy is employed to balance the trade-off between speed and safety. The proposed solution was validated in hardware-in-the-loop simulation and real-world experiments. In fact, it was benchmarked against other state-of-the-art visual-inertial navigation solutions in the first

Artificial Intelligence Robotic Racing season in 2019, where it was the fastest and most robust approach in those conditions, with the runner-up team being (only) three seconds slower in the final race. Significantly, this chapter highlights the importance of minimizing latency in the perception pipeline to achieve high-speed autonomous flight. Our solution prioritized real-time performance at the system's maximum operating frequency (i.e., 60 Hz) over the use of more complex and computationally expensive algorithms.

> **RQ2**: How can we leverage the knowledge of the inner working of event cameras to learn event-based frame reconstruction in a self-supervised fashion?

This second research question was motivated by the limited adoption of event cameras in robotics, despite their numerous advantages. This limited adoption is primarily due to the lack of a mature algorithmic ecosystem capable of effectively utilizing the sparse and asynchronous nature of event camera output. In Chapter 3, we addressed this question by proposing a novel self-supervised learning (SSL) framework for event-based frame reconstruction. Our proposed solution is based on the event generative model, which, under constant illumination, establishes a relationship between events, brightness, and optical flow. Specifically, we demonstrate that this model can be leveraged to learn to reconstruct brightness frames from the events without relying on ground-truth data, as long as the optical flow encoded in the events is known or can be estimated. To achieve this, we train ANNs to minimize the discrepancy between the input events and the model-based predictions. The effectiveness of our training pipeline was validated using multiple datasets, where our networks demonstrated performance comparable to the state-of-the-art methods, despite not having access to reference frames during the training process. Additionally, we addressed the task of event-based optical flow estimation within the SSL framework. Our approach utilized the concept of contrast maximization for motion compensation, allowing us to learn event-based optical flow directly from the input events. Furthermore, we proposed a lightweight neural network architecture for event-based optical flow, which achieved high-speed inference while maintaining a minimal decrease in performance.

> **RQ3**: How can a spiking neural network learn to develop event-based motion selectivity in an unsupervised fashion?

This third research question, focusing on event-based optical flow estimation with spiking neural networks (SNNs), was addressed in Chapter 4. Here, we introduced the first SNN that develops selectivity to motion, including direction and speed, in an unsupervised manner from the input event stream. The success of our approach was facilitated by the development of several key components. Firstly, we proposed a spiking neuron model capable of effectively handling the rapidly varying input statistics of event cameras through pre-synaptic adaptation. Secondly, we formulated a novel version of the correlation-based spike-timing-dependent plasticity (STDP) rule, which differs from the existing state-of-the-art approaches by being inherently stable. And finally, we designed a convolutional SNN architecture that learns to perform hierarchical feature extraction. Specifically, it starts by extracting geometric features, followed by capturing their local motion using

multi-synaptic connections with different temporal delays, and eventually inferring global motion estimates via spatial integration. The effectiveness of this approach was validated through experimentation using both synthetic and real event sequences. However, due to the absence of supervision, quantitative comparisons with the state-of-the-art methods posed challenges. As a result, we relied on extensive qualitative analysis to assess and compare the performance of our approach. Most significantly, this chapter highlights the potential of SNNs to perform low-latency, event-based optical flow estimation.

RQ4: How can low-latency, event-based optical flow be learned in a self-supervised fashion with spiking neural networks?

This fourth research question, driven by the limitations of unsupervised learning, was addressed in Chapters 5 and 6. In Chapter 5, we presented the first set of deep SNNs to successfully solve the problem of event-based optical flow estimation. To accomplish this, we reformulated the state-of-the-art training pipeline for ANNs (i.e., the aforementioned contrast maximization) to significantly reduce the time windows presented to the networks. Additionally, we refined the SSL loss function to enhance its convexity. Prior to training with our framework, we augmented various ANN architectures from literature with explicit and/or implicit recurrency, alongside the incorporation of the spiking behavior. Extensive quantitative and qualitative evaluations were conducted using multiple datasets. Our results not only confirm the efficacy of our training pipeline, but also demonstrate that the proposed set of recurrent ANNs and SNNs perform comparably to the state-of-the-art self-supervised methods.

Despite the significant accomplishments of Chapter 5, the proposed training pipeline assumes linear motion of events within the timeframe of their loss function, limiting its ability to accurately capture the true trajectory of scene points over time. To overcome this, Chapter 6 introduces a reformulated pipeline that addresses the scalability to high inference frequencies while accurately capturing the true trajectory of scene points. An iterative event warping module and a multi-timescale loss function are the main additions to the pipeline. The former unlocks a novel multi-reference loss function that improves the accuracy of the predicted optical flow, while the latter enhances the robustness of the training process. The effectiveness of this new approach was validated using multiple datasets, where our models demonstrated superior performance compared to both the self-supervised and model-based baselines, surpassing them by significant margins. Please note that, although this reformulation of the training pipeline was validated with ANNs, it is extrapolable to SNNs as well.

RQ5: How can a spiking neural network be trained in a self-supervised fashion to perform event-based optical flow estimation while running on a neuromorphic processor in the control loop of an autonomous flying robot?

This fifth and last research question was tackled in Chapter 7, where we introduced the groundbreaking concept of a fully neuromorphic vision-to-control pipeline for controlling a freely flying robot. The experimental setup involved equipping the robot with a

downward-facing event camera, which captured data from a static planar surface, and a specialized neuromorphic processor. The latter was used to run a compact SNN that was trained to process high-dimensional raw event-camera data and output low-level control actions. This allowed for autonomous vision-based ego-motion estimation and control at approximately 200 Hz, spending only 27 μJ per network inference. The proposed learning setup effectively addresses the challenge of slow and inaccurate simulation of event-based data, as it allows for the independent training of vision and control. While the vision part of the network is trained using an adapted version of the self-supervised pipelines from Chapters 5 and 6, the control policy is learned through evolution in simulation without the need to simulate events. Real-world experiments were conducted, wherein the event camera and neuromorphic processor were integrated into the control loop of the flying robot. The results showcased the effectiveness of our approach, as the robot accurately followed various ego-motion setpoints and successfully performed hovering, landing, lateral maneuvers, and even constant yaw rate control.

The answer to this final research question, which builds upon the contributions of the previous chapters, also serves as the answer to our initial problem statement, which was formulated as follows:

> **Problem Statement**: How can optical-flow-based autonomous navigation be realized with an event-based camera and a neuromorphic processor in the control loop of a flying robot?

Our studies demonstrate the benefits of incorporating neuromorphic technology into the vision-based state estimation pipeline of autonomous flying robots, particularly in terms of latency and power consumption. The results obtained from our experiments are highly encouraging and suggest that the integration of neuromorphic sensing and processing could make deep neural networks more accessible for small autonomous robots (and other edge devices with limited resources). This advancement has the potential to enhance the agility, versatility, and robustness of these robots, bringing them closer to the capabilities exhibited by flying insects.

8.2 Discussion

The pursuit of incorporating neuromorphic technology into the control loop of autonomous flying robots has been the driving force behind the research presented in this dissertation. In this section, we delve into the main challenges encountered during this journey and the lessons we have gleaned along the way. Additionally, we shed light on the wider implications of our findings and the potential impact they can have on the broader scientific research field.

Efficient intelligence for flying robots

Flying robots face inherent limitations due to their restricted payload capacity and power budget. Throughout our research, we have recognized the critical significance of developing efficient perception and processing pipelines to enable the autonomy of these robots. This theme has been at the core of our work, driving our exploration and innovations. In

Chapter 2, we developed a vision-based pipeline specifically tailored for autonomous drone racing with frame-based cameras. To ensure fast, agile, and robust flight, we leveraged classical (lightweight) algorithms wherever possible, while ANNs were used selectively where needed. Our research then shifted to exploring event-based cameras and SNNs as alternatives to conventional sensing and processing in Chapters 3, 4, 5, and 6. These chapters highlight the potential of neuromorphic computing to achieve low-latency, low-power vision-based perception for robotics. Finally, in Chapter 7, we demonstrated the feasibility of a fully neuromorphic vision-based pipeline for controlling a freely flying robot. This solution, which is characterized by an energy consumption in the order of microjoules and a latency of milliseconds, represents a significant step towards developing efficient intelligence for flying robots, as it shows that heavy processing can be realized on-board with only a fraction of the power needed to fly.

Low-latency processing of event data with SNNs

The event-camera literature has primarily focused on the use of stateless ANNs to process the data, with only a limited number of studies exploring the use of SNNs in often less complicated tasks. In this dissertation, we argue that to realize the full potential of event cameras and achieve low-latency and low-power solutions, stateful SNNs should process the incoming events nearly as they are generated by the sensor, with no accumulation in between. To support this claim, we have demonstrated the training of architecturally-complex SNNs for real-world, large-scale problems, specifically event-based optical flow estimation. In Chapter 4, we approached the problem from an unsupervised learning perspective using STDP to train the SNNs, while in Chapters 5 and 6, we explored the self-supervised learning paradigm by employing new formulations of contrast maximization for motion compensation. In both approaches, we shortened the time windows presented to the networks and removed the temporal information from the input representations, approximating the way in which SNNs would receive events directly from the camera. This approach, which promotes the integration of spatiotemporal information within the models themselves through their internal dynamics, was then employed in Chapter 7 to realize the fully neuromorphic autonomous flight of a robot. Note that this idea has already sparked significant interest in the event-based optical flow literature, and several subsequent works have followed our findings [269, 270, 275], as they can potentially lead to lightweight solutions that are also robust to large pixel displacements.

Training SNNs with and without supervision

In this dissertation, we tackled the problem of training SNNs for event-based optical flow estimation from two different learning perspectives: unsupervised and self-supervised. In Chapter 4, our focus was on unsupervised learning using the STDP rule, which adjusts the synaptic weights based on the temporal correlation between pre- and postsynaptic spikes. Despite the simplicity of this local learning rule, we successfully solved the task at hand by leveraging our knowledge of optical flow and the characteristics of event cameras to design an SNN capable of developing motion selectivity using this learning rule. However, the lack of supervision in STDP presents challenges. Firstly, it is difficult to directly benchmark our approach against the state-of-the-art since the learned features have limited controllability (i.e., we cannot specify what the network should learn). Secondly, careful fine-tuning

of hyperparameters such as firing thresholds, decays, and synaptic delays is required to maintain a balanced network activity. On the other hand, the SSL paradigm explored in Chapters 5, 6, and 7 involves defining a loss function to be minimized during training, using the well-known backpropagation (through time) algorithm. This approach, once compensated for the non-differentiability of the spiking activation function [37, 38], enables us to leverage the vast array of tools and techniques developed for training ANNs. In these chapters, we demonstrated the effectiveness of this approach by training SNNs capable of estimating event-based optical flow with accuracy levels comparable to those of their ANN counterparts, and with the added benefit of being able to run on a neuromorphic processor. Note that the findings from our investigations on estimating event-based optical flow with SNNs have gotten the attention of the broader research community, as evidenced by the numerous follow-up works [250, 254, 320–333].

8.3 Outlook

In this section, we explore several potential avenues for future research, which we believe can build upon the findings presented in this dissertation.

Toward per-event processing with SNNs

One of the key insights from our research is that to fully harness the potential of event-based solutions, SNNs need to be trained to process incoming events nearly instantaneously as they are generated by the sensor. However, achieving this per-event processing approach is more complex than it might appear. There are two main challenges associated with this goal. Firstly, SNNs are recurrent networks with intricate internal dynamics. When the simulation timestep is decreased to enable per-event processing, the training process becomes significantly slower, and the memory requirements increase drastically if training with backpropagation through time (BPTT). BPTT, which has been shown in this dissertation to be a robust and effective gradient-based learning rule for SNNs, relies on accessing the network's internal states from previous timesteps to perform credit assignment over time. Consequently, these states need to be stored in memory throughout the BPTT process, which adds to the memory requirements. Secondly, as the simulation timestep decreases, the sparsity of the input increases. This means that there are fewer events occurring within each timestep, making it more challenging to extract meaningful patterns from the input data. We believe that these challenges could be addressed in multiple ways. One approach is to explore alternative learning rules that do not require the unrolling of networks in the backward pass, such as forward propagation through time [334] or the forward-forward algorithm [335]. These rules can reduce memory requirements and alleviate the computational burden of training SNNs. Additionally, leveraging the sparsity of SNNs and employing sparse computations during simulation can help reduce memory usage and increase inference speed. Techniques and frameworks that support sparse computations, such as the sparse module in PyTorch[1], can be utilized for optimization. Lastly, another avenue of exploration is the use of more complex recurrent units, including potentially gated units, to enhance the ability of SNNs to extract meaningful patterns from sparse input data.

[1] See torch.sparse at https://pytorch.org/docs/stable/sparse.html (in beta at the time of writing this document).

Robustifying unsupervised learning

Unsupervised learning opens the door to backpropagation-free online learning in SNNs. However, an observation from our research is that while unsupervised learning rules can be effective in tasks where prior knowledge can be utilized in the network design, their applicability becomes limited when such knowledge is unavailable or cannot be incorporated into the architecture. For instance, in Chapter 4, we successfully trained an SNN to develop motion selectivity using STDP with a hierarchical architecture that enabled motion information to be discernible as clusters of spatiotemporal patterns. However, unsupervised learning may face challenges solving other tasks. To address this limitation and enhance the controllability of the learning process, we believe that future research on unsupervised learning for SNNs should focus on meta-learning, i.e., learning to learn. Specifically, we propose exploring the learning of the parameters of local learning rules, such as STDP, through either self-supervised or pure supervised approaches. By enabling the network to optimize its online learning rule in an offline training phase according to the needs of the task at hand, there is potential to enhance the robustness and deployability of SNNs. It is worth noting that some neuromorphic processors, such as Intel's Loihi [14, 51], already offer support for online learning through customizable local learning rules.

The future of neuromorphic flying robotics

One of the primary contributions of this dissertation is the successful demonstration of a neuromorphic vision-to-control pipeline for controlling a freely flying robot with minimal latency and power consumption. This achievement represents a significant step forward in the field of event-based cameras and SNNs, and it marks the beginning of a long journey towards the development of fully neuromorphic, small (flying) robots. The development of such robots, surpassing the capabilities of their counterparts equipped with conventional sensors and processors, holds great promise for the future. However, to realize this vision, several challenges need to be addressed, both in terms of hardware and software. On the hardware side, a significant advancement could come from improving input/output (I/O) bandwidth in the processors. Enhancing the I/O capabilities can facilitate efficient data transfer between event-based sensors and neuromorphic processors, enabling faster and more seamless processing. Additionally, considerations for small form factors and low power consumption are essential for both the sensors and processors. Furthermore, there is potential for further efficiency gains by transitioning to analog neuromorphic processors, but this will pose even larger challenges in terms of their development and deployment. Regarding software, we reiterate the importance of per-event processing with SNNs, the need for scalable training pipelines and more complex recurrent units, and the potential of meta-learning. Once these challenges in hardware and software are overcome, the journey towards fully neuromorphic robots will gain significant momentum, enabling advancements in various areas of robotics and paving the way for a new era of intelligent and efficient machines.

A

How Do Neural Networks Estimate Optical Flow?

This chapter presents the first research topic that was explored in parallel to the main research questions of this dissertation. We investigate how deep artificial neural networks estimate frame-based optical flow. A better understanding of how these networks function is important for not only assessing their generalization capabilities to unseen inputs, but also for suggesting changes to improve their performance. For our investigation, we focus on FlowNetS, as it is the prototype of an encoder-decoder neural network for optical flow estimation. Furthermore, we use a filter identification method that has played a major role in uncovering the motion filters present in animal brains in neuropsychological research. The method shows that the filters in the deepest layer of FlowNetS are sensitive to a variety of motion patterns. Not only do we find translation filters, as demonstrated in animal brains, but thanks to the easier measurements in artificial neural networks, we even unveil dilation, rotation, and occlusion filters. Furthermore, we find similarities in the refinement part of the network and the perceptual filling-in process which occurs in the mammal primary visual cortex.

The contents of this chapter have been published in:

D. B. de Jong, **F. Paredes-Vallés**, G. C. H. E. de Croon, *How do neural networks estimate optical flow? A neuropsychology-inspired study*, IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2021.

Contribution: The research leading to this chapter's work was the result of an M.Sc. graduation project conducted by ir. David B. de Jong, whom I supervised together with prof. dr. Guido C. H. E. de Croon. Apart from the conceptual collaboration, I specifically helped designing and conducting the experiments, as well as analyzing the obtained results. Moreover, I contributed to extending the original work for the revised version of the paper, which was eventually published.

A.1 Introduction

Optical flow is a visual cue defined as the projection of the apparent motion of objects in a scene onto the image plane of a biological vision system or a visual sensor [21]. This cue is important for the behavior of animals of varying size [336], ranging from small flying insects [180] to humans [179], as it allows these animals to estimate their ego-motion and to have a better understanding of the visual scene. Optical flow is also important in computer vision and robotics applications for tasks such as object tracking [337] and autonomous navigation [73].

Many algorithms have been introduced to determine optical flow [338], including correlation-based matching methods [339, 340], frequency-based methods [341, 342], and differential methods [163, 343]. Correlation-based matching methods try to maximize the similarity between different intensity regions across multiple frames. Finding the best match then corresponds to finding the shift which maximizes the similarity score. Frequency-based methods exploit either the amplitude or phase component of the complex valued response of a Gabor quadrature filter pair [344] convolved with an image sequence. Lastly, differential methods compute optical flow based on a Taylor expansion of the image signal, subject to the brightness constancy assumption.

All these methods assume that the brightness of a moving pixel remains constant over time and, when applied locally, are subject to the *aperture problem* [232]. Only motion components normal to the orientation of an edge in the image can be resolved.

A global smoothness constraint has been added for differential methods, which assumes that neighboring pixels undergo a similar motion [343]. This has led to *variational* methods that minimize a global energy function consisting of a data and a smoothness term. These methods have played a dominant role for many years due to their high performance. However, a main drawback is that the iterative minimization of the energy function leads to long computation times. Moreover, the brightness constancy assumption is a coarse approximation to reality and thus limits performance [345]. Research has focused on extra energy terms to deal with deviations from this assumption and improve the robustness of global smoothness constraints, leading to slow but steady progress.

As in many other computer vision areas, currently, the best-performing algorithms are trained deep neural networks. Initially, training such networks was challenging due to the lack of ground-truth optical flow data and the excessive human effort required for manual optical flow labeling. Dosovitskiy *et al.* [265] were the first to successfully train deep neural networks to estimate optical flow by using a synthetically generated dataset with optical flow ground truth. Their networks, FlowNetS and FlowNetC, initially performed slightly worse than the state-of-the-art variational methods [346]. However, trained deep neural networks became the new state-of-the-art method for optical flow estimation by subsequent researchers who focused on improving the architecture and training data [189, 266, 347].

Until now, the functioning of these networks is poorly understood. In this chapter we investigate *how* deep neural networks perform optical flow estimation. There are two main reasons why this is important. First, understanding the method's functioning brings insights into its limits and robustness, for example concerning generalization to test distributions. Second, it may lead to valuable recommendations for improving the performance, for instance, by changing properties of the architecture or training data.

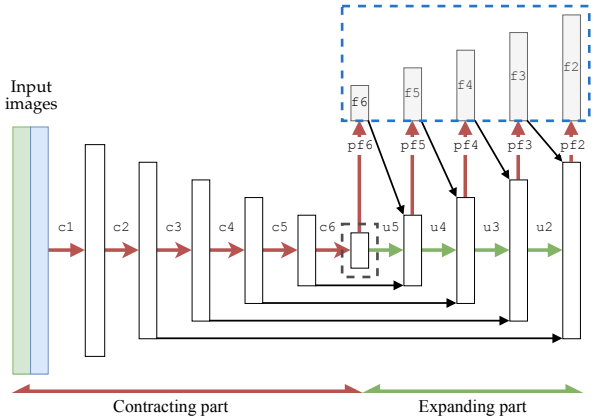


Figure A.1: Schematic of the FlowNetS architecture [265]. The contracting part compresses spatial information through the use of strided convolutions (c), while the expanding part uses upconvolutions (u) for refinement. The predict-flow (pf) layers transform feature map activations into dense flow estimates (f). The feature map corresponding to the output of the c6 layer (gray dashed box) is studied in Sections A.4 and A.5, while the flow refinement process (blue dashed box) is discussed in Section A.6.

In our analysis of deep optical flow networks, we make use of a method that has helped unveiling the workings of motion-sensitive brain areas in neuropsychology [348]. Specifically, we measure the response of neurons in FlowNetS [265] to stimuli with varying spatiotemporal frequencies and construct a spectral response profile. The input stimuli used are translating plane waves, as this input type proved to be more selective in the frequency domain than moving bars [349]. Based on the earlier findings of Gabor filters [344] in biological vision systems [350, 351] and other learning-based methods [352, 353], we expect to find these filters in FlowNetS as well. Therefore, we fit a Gabor function to the spectral response profile of neurons in the network and study the residual error patterns. We find that the Gabor translational motion filter model is suitable for the majority of the neurons. Additionally, we find neurons sensitive to motion patterns such as dilation, rotation, and occlusion. Interestingly, neurons sensitive to these motion patterns have not been mentioned in neuropsychology. Furthermore, our analysis strongly suggests that the resolution in the temporal frequency domain can be significantly improved if more than two frames would be used as input to the neural network. Lastly, we find that the optical flow refinement process in the decoder part of the network behaves in a manner akin to flow refinement within biological vision systems.

A.2 Related work

A.2.1 Dense optical flow estimation with CNNs

Ever since the pioneering work of Horn *et al.* [343], variational optical flow methods [354] have played a dominant role in optical flow estimation due to their high performance. Most modern variational optical flow estimation pipelines consist of four stages: matching, filtering, interpolation, and variational refinement. Various improvements have been proposed over time to deal with issues such as long-range matching [355] and occlusion [356].

Furthermore, improvements such as dense correspondence matching based on convolution response maps of the reference image with the target image [357], and supervised data-driven interpolation of a sparse optical flow map [358] were also proposed. These last two improvements introduced elements of deep learning into the variational pipeline.

Dosovitskiy *et al.* [265], however, were the first to introduce a supervised end-to-end trained convolutional neural network (CNN). CNNs have three major advantages when it comes to estimating optical flow. First, CNNs outperform variational optical flow estimation methods in terms of accuracy [189, 266, 347]. Second, the runtime of CNN-based optical flow algorithms, when executed on the appropriate hardware, is significantly lower than variational methods [189]. Third, CNN-based methods can learn from data and can exploit statistical patterns not realized by a human designer. This is an advantage over variational methods which require explicit, and sometimes inaccurate, assumptions on the input. However, CNNs also have three disadvantages. First, the results depend on the quality and size of the training data. Second, CNN-based methods face the risk of overfitting, which is relevant for optical flow estimation because it is difficult to obtain ground truth [346]. Third, there is no guarantee that the trained models will generalize to scenarios not contained in the training dataset. Due to the *"black-box"* nature of the solution, it is difficult to get insight into its workings and limitations.

In [265], Dosovitskiy *et al.* introduced two networks based on the U-net architecture [137]: FlowNetS and FlowNetC. While FlowNetS is an encoder-decoder network consisting of simple convolutions, FlowNetC creates two separate processing streams and combines them in a *correlation-layer*. This layer performs a multiplicative patch comparison between feature maps. Due to the explicit use of a correlation-layer, it is more straightforward to understand the workings of FlowNetC. However, not much is known about the workings of FlowNetS. Inspired by this architecture, Ranjan *et al.* [359] introduced SpyNet, a spatial image pyramid with simple convolutional layers at each pyramid level and a warping operation between pyramid levels. SpyNet's coarse-to-fine approach brings a higher computational and memory efficiency at the cost of a more limited set of perceivable motion types. Ranjan *et al.* also visualized the weights of the first layer of their network and observe that these filters resemble Gabor filters [344], which provided a glimpse into the working principle of this architecture. Finally, Teney *et al.* [360] built a shallow CNN-architecture by integrating domain knowledge, such as invariance to brightness and in-plane rotations. On small motion their architecture performs well, but performance declines on large motion near occlusions. They conclude good occlusion performance requires reasoning over a larger spatiotemporal extent.

The generalization performance of CNN-based methods can be evaluated for specific instances by determining the epistemic uncertainty [361]. Indeed, Ilg *et al.* [362] used a modified FlowNetC that produces multiple hypotheses per forward pass, which are then merged to a single distributional flow output. They showed that their network produces highly uncertain flow estimates when optical flow estimation is difficult (shadows, translucency, etc.). Lastly, Ranjan *et al.* [363] highlighted another downside of deep neural networks, which is the ability of adversarial examples to fool neural networks and produce erroneous results. They showed that especially networks using an encoder-decoder architecture are affected, while networks using a spatial pyramid framework are less vulnerable. None of the works above, however, explain how their architecture estimates optical flow.

A.2.2 Receptive field mapping

There are two main threads of research to understand what neural networks have learned: attribution and feature visualization. Attribution methods [364, 365] are used to *attribute* filter outputs, like optical flow, to parts of the input by visualizing the gradient. However, it is hard to see where an optical flow estimate comes from. Feature visualization is concerned with understanding what neurons, filters, or layers in a neural network are sensitive to by optimizing the input [366]. The result is usually an image with noisy and visually difficult to interpret high-frequency patterns [367]. Three methods of regularization can be applied to cope with this phenomenon. First, frequency penalization discourages the forming of these patterns. The downside is that this approach also discourages the forming of legitimate high-frequency patterns which are of interest for optical flow estimation. Second, small transformations like scaling, rotation, or translation can be applied in between optimization steps [368]. This approach is also not viable because transformation affects the ground truth of optical flow. Third, priors can be used which can keep the optimized input interpretable. Such approaches typically involve learning a generative model [369] or enforcing priors based on statistics from the training data [370]. This approach is often very complex and it may be unclear what can be attributed to the prior and what can be attributed to what the network has learned.

Due to these reasons, we look at the field of neuropsychology and specifically study what methods researchers have used to determine what stimuli activate neurons in mammalian vision systems and what functions best describe the neural responses. It was shown that Gabor functions [344] best modeled the spatial response of simple cells in the mammal visual cortex [350]. It can be shown that Gabor filters are optimal for simultaneously localizing a signal in the spatial and frequency domain [371], making them ideal for motion estimation. Later, DeAngelis *et al.* [372] examined the spatiotemporal response of cells and their space-time separability. In functional form, space-time separable Gabor filters are frequency-tuned with a stationary Gaussian envelope, and space-time inseparable Gabor filters are velocity-tuned with a moving Gaussian envelope [373]. In this work we only consider fitting frequency-tuned Gabor filters, due to their simplicity and the low number of input frames used by the FlowNet architectures.

Two approaches to receptive field mapping in neuropsychology can be discerned: the reverse-correlation approach and the spectral response profile approach. The former presents a rapid random sequence of flashing bars at various imaging locations to the mammal. The spike train emitted by the neuron in the subject is correlated to the sequence in which the stimuli were presented. This approach allows for a rapid measurement of the receptive field profile in the spatiotemporal domain [351]. Instead, the spectral response profile approach presents translating plane waves to the mammal at varying orientations and spatiotemporal frequencies [374, 375]. Jones *et al.* used both the reverse-correlation approach to construct a spatial receptive field profile [376] and measured the response to plane waves to construct a spectral response profile [348]. Subsequently, the spatial and spectral responses were compared to the Gabor filter model in the spatial and frequency domain, and the filter parameters obtained from both methods proved to be highly correlated [350]. A similar correspondence in outcome between the methods was found by DeAngelis *et al.* [372, 375] in the visual cortex of cats.

In this work we extend the approach of Jones *et al.* [350] to the spatiotemporal domain and measure spectral responses of the network to translating plane waves, to which frequency-tuned spatiotemporal Gabor filters are fitted. A benefit of measuring the spatiotemporal spectral responses for optical flow is that translation is more easily described in the frequency domain [373].

A.2.3 Aperture problem

Optical flow methods are only able to resolve motion components normal to the orientation of an edge in the intensity pattern. This is known as the aperture problem [232]. In CNNs the size of the aperture of a neuron is referred to as the receptive field, which is defined as the region in the input which affects the activation of the neuron. In this work we show that the receptive field size is related to the aperture problem by training different versions of FlowNetS with varying receptive field sizes.

In neuropsychology, Komatsu [377] has shown the existence of a perceptual filling-in mechanism in the mammalian visual cortex for cues such as color, brightness, texture, or motion. While the precise neural workings are still under discussion, edge structure [378] and the interaction between neighboring neurons play an important role in this process [379]. In neural networks, attempts have been made to implement such a mechanism as well. To allow for the interaction between neurons, a recurrent model can be used [380]. Zweig *et al.* [358], however, used an unfolded feed-forward version of a recurrent network and a multi-layer loss to allow for interaction between neurons. Their CNN-based motion interpolation architecture takes a sparse flow map and edge structure as input. They showed their motion interpolation method refines motion estimates similarly to the human visual cortex by demonstrating the filling-in effect of the network on a Kanizsa illusion. FlowNetS also features a multi-layer loss, and, in Section A.6, the ability of the expanding part of FlowNetS to interpolate and refine flow maps is highlighted.

A.3 Model details

Fig. A.1 shows a schematic representation of the FlowNetS architecture, which takes two consecutive images as input. Multiple versions of FlowNetS exist. Dosovitskiy *et al.* [265] mention the use of the ReLU activation function in their work. The release of their pre-trained models, however, uses a leakyReLU activation function. In order to facilitate interpretability of the motion filter analysis, we choose to use the ReLU version. With the same aim, we introduce two small adjustments. First, the bias terms are removed in the predict-flow `pf` layers because the flow is assumed to be zero-centered. Second, the kernel size in the `pf` layers is reduced from 3×3 to 1×1 to allow clearer location identification.

Regarding training, as in [265], we use the same data augmentation *on both* frames, but we do not use incremental flow and color augmentation *between* frames, since the authors do not specify the parameters of these mechanisms. Furthermore, the network is trained for fewer iterations (300k iterations versus 600k iterations) due to limited availability of computational resources. Evaluation on the MPI-Sintel [381] and FlyingChairs [265] datasets shows comparable performance between the slightly modified FlowNetS and the original version, as can be seen in Section A.7.1.

The synthetic dataset FlyingChairs [265], which was used to train the original and our slightly modified FlowNetS, consists of approximately 22k image pairs. The image pairs are composed of a varying numbers of chairs and background images from natural scenes. Between image pairs, a composition of translation, rotation, and scaling motion is applied. As stated in the supplementary material of [265], the size of the chairs[1] is sampled from a Gaussian with a mean and standard deviation of 200 pixels, clamped between 50 and 640 pixels. Note that the synthetic scenes also contain occlusion. Further details about the composition of affine motion can be found in [265].

A.4 Gabor fitting for translation

We investigate to what motion patterns the neurons in FlowNetS are sensitive. In neuropsychology, the responses of simple cells turned out to be captured very well by Gabor filters [348, 350, 375, 382]. That simple cells act like Gabor filters makes sense, since Gabor filters are known to be optimal in the sense that they achieve maximal resolution in both the spatiotemporal and the associated frequency domains. As a consequence, they require a minimal number of filters to represent spatiotemporal information [350, 375].

Although artificial neural networks are very different in many aspects from biological ones, they were inspired by them and inherit similar traits. In particular, they seem suitable to represent spatiotemporal filters and may be subject to a similar pressure as biological networks to succinctly represent spatiotemporal patterns when having to estimate optical flow. This was our motivation to first investigate whether FlowNetS' neural responses resemble those of Gabor filters. In our investigation, we mainly focus on the deepest encoding layer in the network, the `c6` layer. As shown in Fig. A.1, the activations of the feature maps of these layers are directly, linearly transformed (via `pf6`) into an initial coarse-scale horizontal and vertical flow estimate (i.e. `f6`), which is later used as the basis for refinement. Hence, the coarsest, most direct representation of optical flow is encoded in this layer. Although we focus our analysis on `c6`, the earlier layers play an important role as well. They do this not only by the determination of the activations in layer `c6` but also (in the case of `c2` - `c5`) by contributing to the refinement of optical flow via skip connections.

In this section, first the theory behind Gabor filters and the spectral response fitting method is discussed, followed by the results obtained. Thereafter, we discuss the resolution in the temporal frequency domain of the fitted Gabor filters.

A.4.1 Methodology

As in [341, 344, 373], the spatiotemporal frequency-tuned Gabor filter g in Cartesian coordinates centered at the origin can be written as the product of a Gaussian w and a translating plane wave s:

$$g(x, y, t) = s(x, y, t)w(x, y, t) \tag{A.1}$$

[1] Note that, in [265], the authors do not specify how the size of a chair is determined, so there is a certain ambiguity around this parameter.

The (non-normalized) Gaussian w is defined by:

$$w(x, y, t) = \exp\left(-\frac{1}{2}\left(\frac{x_R^2}{\sigma_x^2} + \frac{y_R^2}{\sigma_y^2} + \frac{t^2}{\sigma_t^2}\right)\right) \tag{A.2}$$

where σ_x, σ_y, and σ_t control the spread of the spatiotemporal Gaussian window. To decrease the number of parameters in the fitting process, it is assumed that the center of the Gaussian coincides with the center pixel of the receptive field. Furthermore, the subscript R denotes a rotation operation which allows the Gaussian to be aligned along orientation θ_0, and is defined as:

$$\begin{aligned} x_R &= x\cos(\theta_0) + y\sin(\theta_0) \\ y_R &= -x\sin(\theta_0) + y\cos(\theta_0) \end{aligned} \tag{A.3}$$

where a positive value of θ_0 corresponds to a clockwise rotation with respect to the positive x-axis. The subscript 0 indicates the parameter value corresponding to the peak response of the Gabor filter. This orientation, which corresponds to the preferred direction of motion of the filter, is related to the spatial frequencies via $\theta_0 = \tan^{-1}\left(f_{y_0}/f_{x_0}\right)$.

A translating plane wave s in the Cartesian coordinate system can be written as:

$$s(x, y, t) = \cos\left(2\pi\left(F_0 x_R - f_{t_0} t\right) + \varphi_0\right) \tag{A.4}$$

where the spatial frequency magnitude F_0 is related to the spatial frequencies via $F_0 = (f_{x_0}^2 + f_{y_0}^2)^{1/2}$, f_{t0} indicates the temporal frequency, and φ_0 denotes the phase of the filter. The dependence of s on y is due to x_R, which is a function of x and y (see Eq. A.3). A Gabor filter is said to be even when $\varphi_0 = 0$ and odd when $\varphi_0 = \pm\pi$. Further, note that the preferred velocity of the filter v_0 is related to F_0 and the temporal frequency f_{t_0} via $v_0 = f_{t_0}/F_0$, as in [341]. A higher spatial frequency F_0 allows tracking of motion of thinner image structures. When a signal is sampled in time or space, frequency components which are larger than or equal to 0.5 cycles per frame (i.e., the Nyquist frequency) become undersampled and aliasing occurs. Thus, if we limit ourselves to signals which do not suffer from aliasing, the maximum velocity a signal can have is limited by its F_0. Fig. A.2 shows the 3D frequency space with the half-magnitude profile of a Gabor filter.

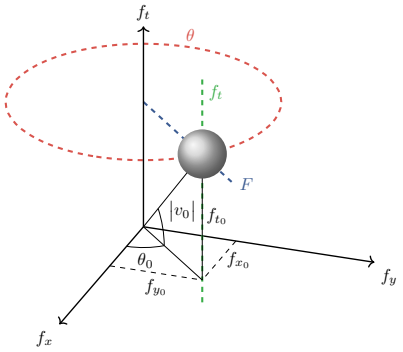


Figure A.2: Illustration of the half-magnitude profile in the 3D frequency domain of a spatiotemporal Gabor filter. The three ranges along which the responses of the Gabor half-magnitude profile are evaluated for the spectral response profile fitting process are shown in color.

Because we will fit the response of phase-sensitive Gabor filters, we highlight three phase-dependent convolution phenomena. Note that a valid convolution[2] of two tensors

[2]We use "convolution" to refer to the correlation of a filter over an image to remain consistent with the CNN terminology.

A

with equal size corresponds to their dot product. First, because a sine is an odd signal, the dot product of two sines at opposite frequencies is negative. Second, the dot product of a cosine at opposite frequencies will be positive due to the even nature of the function. Third, sine and cosine are decorrelated and thus the dot product will be zero between these two signals.

Gabor spectral response profile fitting

In the Gabor spectral response fitting process, translating grayscale plane waves s are used as input to the network, and we try to minimize the difference in response between filters in the c6 layer of our FlowNetS and spatiotemporal Gabor filters g. To better approximate the response of c6 filters, we enhance the Gabor filter output with a gain term K, a bias term b, and pass the response through a ReLU non-linearity. Then, the response r to a convolution with a translating plane wave s and a Gabor filter g is given by:

$$r = \text{ReLU}\left(K(s(x, y, t) * g(x, y, t)) + b\right) \tag{A.5}$$

where r is a function of nine parameters (i.e., F_0, θ_0, f_{t_0}, φ_0, $\sigma_x, \sigma_y, \sigma_t$, K, b), which are estimated in a two-step process.

First, a gridsearch is performed to determine the location in the spatiotemporal frequency domain with the highest response per filter in the c6 layer. We denote the response of the filters in the network by $\hat{r}$, and their peak response value by $\hat{r}_0$. Because the fitted Gabor filters are phase sensitive, this amounts to estimating four parameters (i.e., F_0, θ_0, f_{t_0}, φ_0). Therefore, a four-dimensional grid of translating plane waves (i.e., the input to the network) is constructed using all combinations of these parameters within a given range and step size (see [105]). The range for the value of half spatial wavelength $\lambda/2 = 1/2F$ is chosen so that it captures the sizes of the chairs present in the training dataset (as explained in Section A.3).

Second, once the peak response of the c6 filters is found, we estimate the spatiotemporal spread of the Gaussian (determined by $\sigma_x, \sigma_y, \sigma_t$), the gain K, and the bias b. This is done by minimizing the difference in response between the fitted Gabor filters r (see Eq. A.5) and the corresponding c6 filters $\hat{r}$ along three separate ranges in the spatiotemporal frequency space (F, θ, and f_t). These ranges are illustrated in Fig. A.2, and further described in [105]. We define the cost function $\mathcal{L}$ in response to a convolution with a translating plane wave s as:

$$\begin{aligned} \mathcal{L} &= \sum_i (r_i - \hat{r}_i)_F^2 + \sum_j (r_j - \hat{r}_j)_\theta^2 + \sum_k (r_k - \hat{r}_k)_{f_t}^2 \\ &= \mathcal{L}_F + \mathcal{L}_\theta + \mathcal{L}_{f_t} \end{aligned} \tag{A.6}$$

where $\mathcal{L}_F$, $\mathcal{L}_\theta$, and $\mathcal{L}_{f_t}$ denote the sum of squared errors over the respective ranges. We constrain the bounds of the Gabor filter parameters to obtain reasonable values, which leads to a non-linear bounded convex optimization problem which is solved using the robust trust-region-reflective algorithm [383]. In order to compare the obtained cost values between c6 filters, we construct a normalized cost value $\mathcal{L}_{norm}$ by dividing the cost by the squared peak response of the filter: $\mathcal{L}_{norm} = \mathcal{L}/\hat{r}_0^2$.

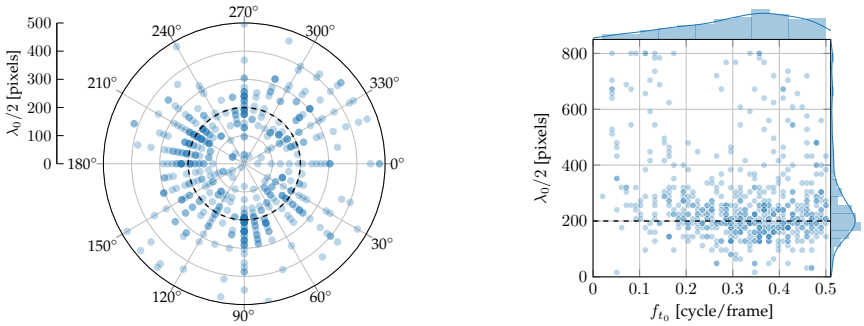


Figure A.3: Location of peak response $\hat{r}_0$ per c6 filter in the spatiotemporal frequency domain in response to translating plane waves. *Left:* Half spatial wavelength $\lambda_0/2$ and orientation θ_0 corresponding to peak response $\hat{r}_0$ per filter. *Right:* Half spatial wavelength $\lambda_0/2$ and temporal frequency f_{t_0} corresponding to peak response $\hat{r}_0$ per filter. In both plots, the black dashed lines indicate the peak of the distribution in the half spatial wavelength dimension, which is around 200 pixels.

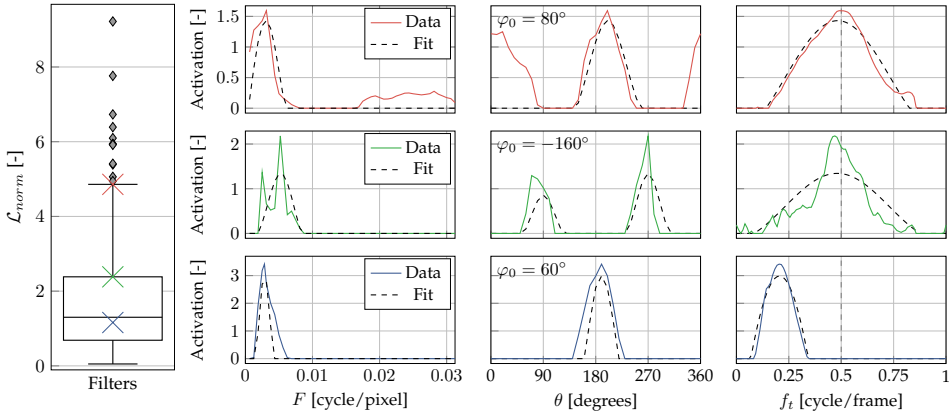


Figure A.4: Quantitative results of the spectral Gabor filter fitting process. *Left:* Boxplot containing the total normalized cost $\mathcal{L}_{\text{norm}}$ per filter (592 filters). *Right 3x3 plots:* Row-wise, the measured responses of three different c6 filters and their corresponding Gabor fits. The blue, green, and red c6 filters correspond to the crosses at the median, near the 75th percentile and near the upper whisker limit of the boxplot, respectively.

A.4.2 Results

We found 592 of the 1024 filters in the c6 layer of FlowNetS to have an activation larger than zero when using the aforementioned input waves. The location of the peak response of the active c6 filters in terms of half spatial wavelength $\lambda_0/2$, orientation θ_0, and temporal frequency f_{t_0} can be seen in Fig. A.3 (left). As shown, the locations of the peak responses of the filters are well distributed over all angles. Radially, there is a

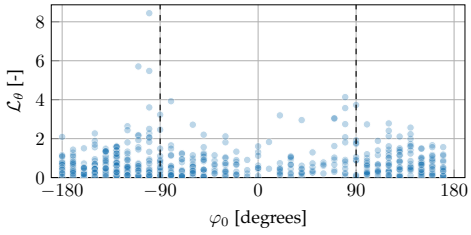


Figure A.5: Orientation cost $\mathcal{L}_\theta$ per filter as a function of φ_0.

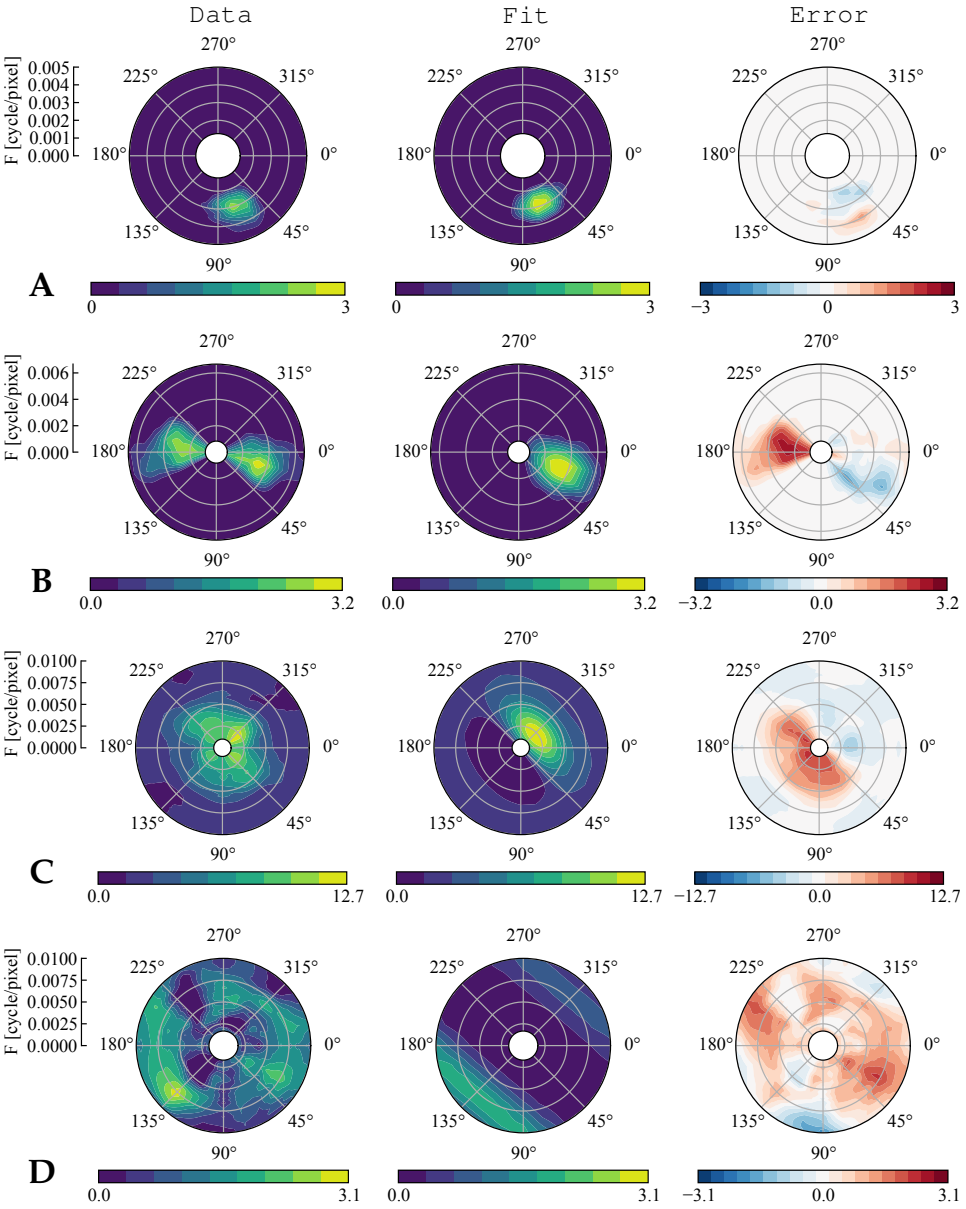


Figure A.6: Qualitative results of the error patterns of the spectral Gabor fitting process. The spectral response profiles are shown as a function of spatial frequency F and orientation θ. `Data` shows the measured response of a `c6` filter, `Fit` is the response of the corresponding fitted Gabor filter, and `Error` shows their difference. Evaluations are with respect to f_{t_0} and φ_0. (*A*) `c6` filter whose response profile is accurately captured by the Gabor model. (*B*) Red `c6` filter from Fig. A.4, which activates on opposite spatial frequencies. (*C*) `c6` filter with a very weak directional bias. (*D*) Noisy `c6` filter pattern (further discussed in Section A.5).

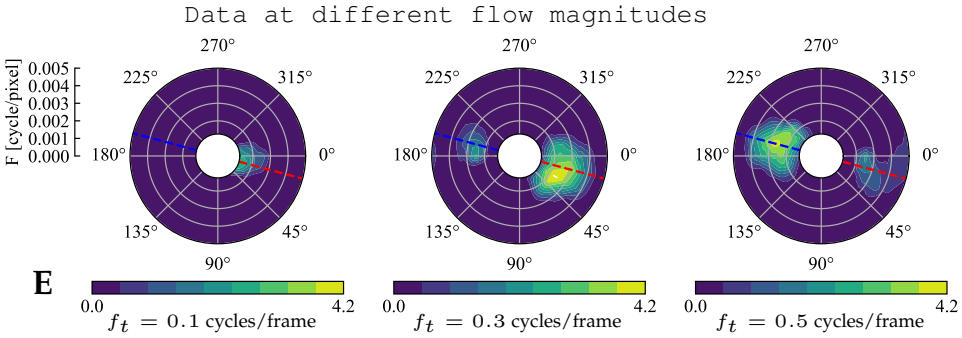


Figure A.6: (continued) (*E*) For this c6 filter, the spectral response profile for three different temporal frequency f_t values is visualized. Two different Gaussian peak responses at opposite orientation can be observed at f_t = 0.3 and f_t = 0.5 cycles per frame. The blue and red lines correspond to the axes of the 2D representation of this filter shown in Figure A.7.

concentration around a half spatial wavelength of 200 pixels. Two possible explanations for this are the fact that (i) the average size of the chairs in the training dataset is 200 pixels, or that (ii) the half of the receptive field size of c6 filters is 192 pixels. The concentration of the peak responses becomes even more apparent in Fig. A.3 (right), which shows the distribution along the temporal and half spatial wavelength axes. Furthermore, we note that the distribution of the temporal frequencies is skewed toward the Nyquist limit of 0.5 cycles per frame. A possible reason for this is the low resolution in the temporal frequency due to the low number frames used as input to the network. This is further discussed in Section A.4.3.

The main observation of our spectral analysis is that the fitted modified Gabor functions (i.e., Eq. A.5) capture the spatiotemporal frequency selectivity of the active c6 filters of FlowNetS accurately. In order to give insight into the goodness of fits for all neural responses in the c6 layer, we show three example responses corresponding to different normalized cost values $\mathcal{L}_{norm}$ in Fig. A.4. Note that the fitted Gabor filters correspond well to the response of the blue and green c6 filters (with $\mathcal{L}_{norm}$ at 50%, 75% of the distribution); but, in the red case (an outlier), the fitted Gabor shows a substantial deviation from the measured c6 response near $\theta = 0$.

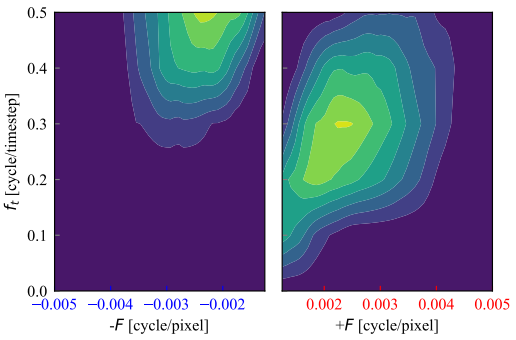


Figure A.7: Spatiotemporal frequency representation of the measured filter response in Fig. A.6E. The positive and negative *F*-axes correspond to the blue and red lines in Fig. A.6E.

This experiment was also performed for the other convolutional layers of the network's encoder segment. As shown in Table A.1, the lower the layer, the smaller the receptive field

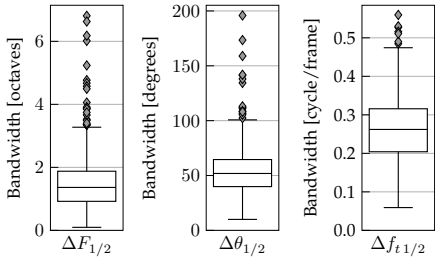


Figure A.8: Bandwidth of spatial frequency F, orientation θ, and temporal frequency f_t of the fitted Gabor filters of the 75% active c6 filters with the lowest $\mathcal{L}_{norm}$.

Layer	$\mathcal{L}_{norm}$	Max. $\lambda/2$	Num. active filters/filters
conv6_1	1.65	800	592/1024
conv5_1	1.42	270	372/512
conv4_1	1.44	270	408/512
conv3_1	1.67	95	234/256
conv2	3.37	47	62/128
conv1	4.71	10	64/64

Table A.1: Result of the Gabor spectral response fitting procedure for different convolutional layers of the encoder part of FlowNetS.

size and hence the upper limit for the half spatial wavelength is decreased. According to the average (normalized) fitting error per layer $\mathcal{L}_{norm}$, the response of neurons in the c3–c6 layers fits well the translational Gabor filter model, while our methodology suggests that neurons in c1 and c2 are not yet as motion-selective as Gabor filters. Table A.1 also shows that c6 is characterized by a higher $\mathcal{L}_{norm}$ than its preceding layer. A possible explanation for this is that, in the earlier layers, the network is only able to perceive less complex motions which better fit the Gabor filter model.

Coming back to c6, the good fit for the majority of neurons supports the choice for the Gabor filter as opposed to other types of models. Of course, one can argue that the Gabor filter does not perfectly capture the response and a more complex model may lead to a better fit. Below, we will extensively delve into the cases in which the Gabor model seems to fall short of explaining c6's neural responses. Here, it is important to note that in principle, we already have such a complex model: the neural network itself. The advantage of the Gabor model is that it has a low number of parameters that can be readily interpreted. Indeed, in neuropsychology, the step to more complex filters was only made when it became necessary for characterizing "complex" cells that did not respond to simple stimuli [382]. The fits and error patterns above the 75% percent threshold (corresponding to the green c6 filter) are very interesting, and we visually inspected them for systematic deviations. Visual inspection is performed instead of an auto-correlation procedure since the latter is not possible due to a non-uniformly spaced polar 3D frequency grid [350]. Fig. A.6 contains the qualitative results used for this analysis, while Section A.7.2 evaluates the generalizability of the fitted Gabor filters to more complex natural stimuli.

Similarly to the blue filter in Fig. A.4, Fig. A.6A shows a c6 filter whose response fits nicely in the Gabor filter framework. On the other hand, we find three types of systematic deviations (i.e., Fig. A.6B, A.6C, A.6E) from the Gabor model, and also conclude that some patterns are too complex for interpretation, such as the c6 filter shown in Fig. A.6D.

The filter in Fig. A.6B shows a deviation from the fitted Gabor 180 degrees away from θ_0. This filter is responsive to edge structure (i.e., $|\varphi_0| \approx 90^\circ$) and is thus approximately odd, since the dot product of two odd signals at opposite frequencies results in a negative value. However, this filter still produces a positive activation at the opposite spatial frequency, corresponding to 180 degrees away from θ_0. In Fig. A.5 the distribution of the phase values

φ_0 versus orientation cost $\mathcal{L}_\theta$ for all filters is depicted. As shown, there are multiple filters responsive to edge structure that have a high $\mathcal{L}_\theta$ (e.g., the red filter in Fig. A.4). One possible reason for this systematic deviation from the Gabor response is that the network is able to learn flow filters that are invariant to polarity (meaning white-black or black-white transitions).

We find two c6 filters that exhibit weak directional bias, an example of which can be found in Fig. A.6C. Moreover, we also find filters that exhibit two or more Gaussian peaks with similar peak response magnitudes but tuned to different spatial frequencies F_0, orientations θ_0, and temporal frequencies f_{t_0}. An example of such a filter can be found in Fig. A.6E, and its 2D spatiotemporal representation is shown in Fig. A.7. A possible explanation is that these filters are sensitive to occlusion, as discussed in Section A.5. Lastly, we find filters that appear noisy and are hard to interpret given the limitations of our methodology (further discussed in Section A.5). Such an example can be seen in Fig. A.6D.

A.4.3 Temporal bandwidth

For orientation θ and temporal frequency f_t, the bandwidth is defined as the width of the filter which provides an output above half the maximum response. This leads to a bandwidth in degrees $\Delta\theta_{1/2}$ and cycles per frame $\Delta f_{t_{1/2}}$ for orientation and temporal frequency respectively:

$$\Delta f_{t_{1/2}} = f_{t_{\max}} - f_{t_{\min}} \tag{A.7}$$

$$\Delta\theta_{1/2} = \theta_{\max} - \theta_{\min} \tag{A.8}$$

For spatial frequency F, the bandwidth is defined in terms of octaves as follows:

$$\Delta F_{1/2} = \log_2 \left(F_{\max}/F_{\min}\right) \tag{A.9}$$

Although we estimate the Gabor parameters of the active c6 filters in the fitting process, the apparent bandwidth of these filter differs due to the non-linear transform in Eq. A.5. The bandwidth is therefore measured based on the fitted Gabor filter response. In Fig. A.8, the bandwidth of F, θ, and f_t can be seen. As shown, the interquartile range for spatial frequency bandwidth is between 1 and 2 octaves and the median orientation bandwidth

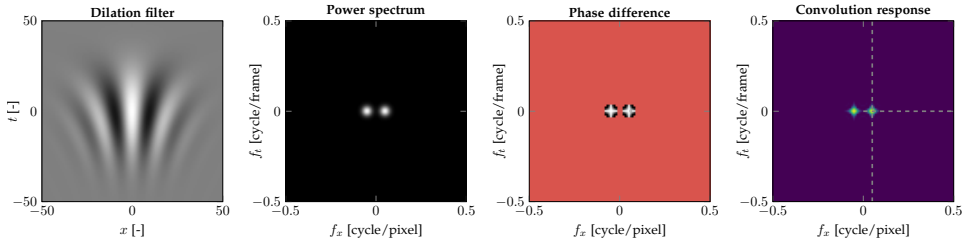


Figure A.9: Convolution response of a dilation filter dw with a translating plane wave s evaluated with spatiotemporal frequencies at k integer multiples of the fundamental frequency. In the ψ plot, a larger phase difference corresponds to a darker color with black being equal to or greater than $\pi/2$. A red mask is applied to frequency components with low power. The dashed lines indicate the Gaussian pattern perceived by the spectral fitting procedure.

is approximately 50°. Lastly, the temporal frequency bandwidth is of large extent with a median of approximately 0.27 cycles per frame.

We note that the network is able to narrow the extent of the filter response in the temporal domain using the non-linear transform in Eq. A.5. An illustration of this mechanism can be seen in Fig. A.10. As shown, the extent of the half-magnitude profile is wider if the non-linear transformation is not employed. This figure also shows what happens when more frames are added to the input and the other parameters are kept the same (see Fig. A.10, bottom). This suggests that an even narrower extent could be reached by feeding the network with more images over time than just the two subsequent images used in FlowNetS. A higher resolution in the frequency domain is beneficial as it allows for a more precise measurement of the flow.

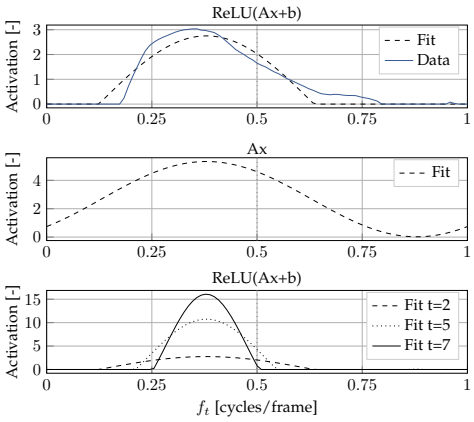


Figure A.10: Illustration of how the network is able to decrease the extent of the filter response in the temporal domain. *Top:* Fit and measured data for the median c6 filter (see Fig. A.4). *Middle:* The response of the fitted Gabor filter without the bias term and ReLU non-linearity. *Bottom:* Response of the fitted Gabor filter when the number of frames is increased.

A.5 Network response to dilation & rotation

In this section, the sensitivity of c6 filters to dilation and rotation is analyzed. First, we explain the limitations of the spectral Gabor response profile fitting process and why we are not able to discern filters activating on translation, dilation, rotation, and occlusion with this methodology. Second, the theory used to identify filters sensitive to dilation and rotation is presented. Lastly, our results are discussed.

Note that Gabor translation filters [344] and occlusion filters [384] already have an analytical description in both the space-time and frequency domain. Such a description of dilation and rotation is, to the best of the authors' knowledge, missing. Therefore, fitting c6 filters to a dilation and rotation motion filter model requires a novel mathematical foundation which is outside of the scope of this work.

A.5.1 Limitations of the spectral response profile fitting

In the first part of the spectral response fitting process, a gridsearch is performed to find the peak response. In the subsequent fitting process, three response lines are generated by varying either F, f_t, or θ, whilst keeping φ constant. This method only allows the measurement of the relative attenuation in amplitude with respect to the peak response $\hat{r}_0$. This is sufficient for translation, which can be defined as a single constant phase Gaussian in the 3D frequency spectrum and thus produces a Gaussian in response. However, it is insufficient for other more complex motion types.

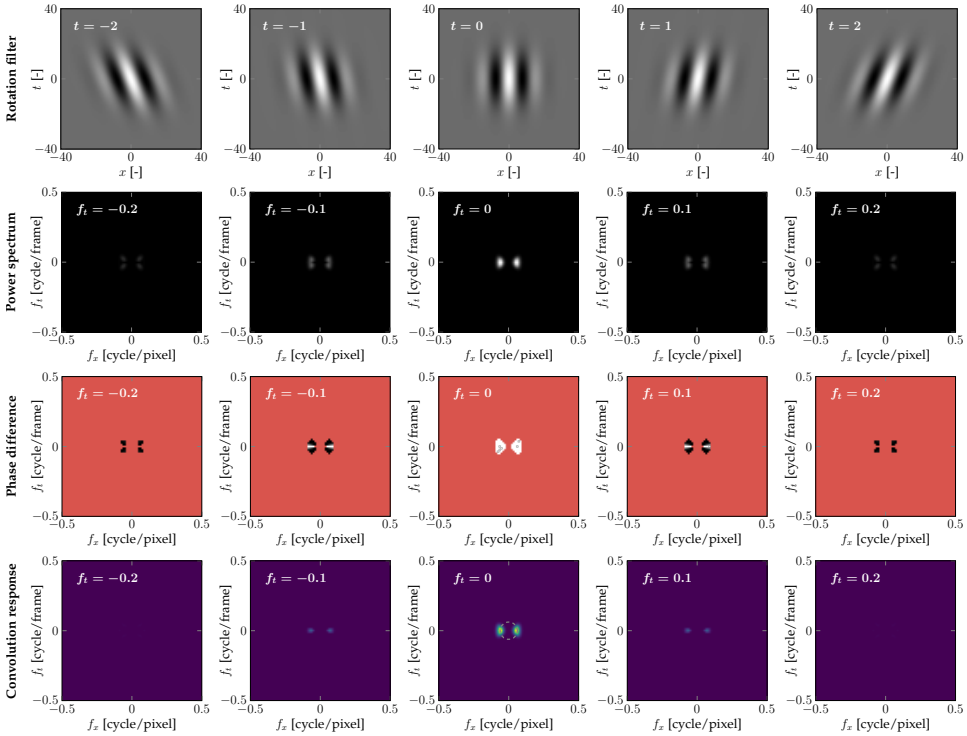


Figure A.11: Convolution response of a rotation filter cw with a translating plane wave s evaluated with spatiotemporal frequencies at k integer multiples of the fundamental frequency. In the ψ plot, a larger phase difference corresponds to a darker color with black being equal to or greater than $\pi/2$. A red mask is applied to frequency components with low power. The dashed circle indicates the double lobe Gaussian pattern perceived by the spectral fitting procedure.

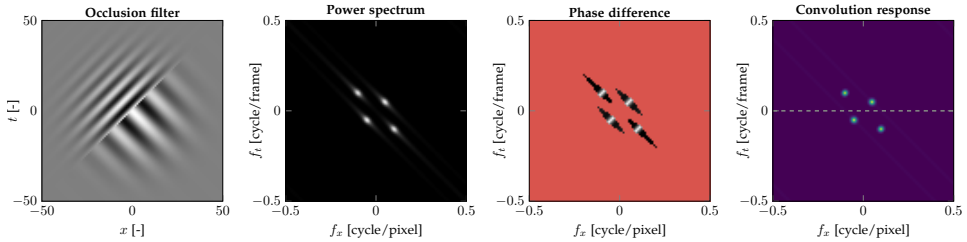


Figure A.12: Convolution response of an occlusion filter with a translating plane wave s evaluated with spatiotemporal frequencies at k integer multiples of the fundamental frequency. *Left:* Example occlusion signal following the description of Beauchemin *et al.* [384]. *Middle left:* The power spectrum of the Fourier-transformed occlusion filter. *Middle right:* The angle ψ indicating the phase difference between the Fourier components of the occlusion filter and s. A larger phase difference corresponds to a darker color with black being equal to or greater than $\pi/2$. A red mask is applied to frequency components with low power. *Right:* Convolution response between the occlusion filter and s. The pattern above the dashed gray line resembles that of Figs. A.6E and A.7.

A

Due to the ReLU activation function, the dot product of two translating plane waves at the same frequency, which are more than or equal to 90 degrees out-of-phase, is zero. Note that a convolution in the space-time domain equals to multiplication in the frequency domain according to the convolution theorem [371]. Because we evaluate the convolution response only at discrete frequencies of k integer multiples along the f_x, f_y, and f_t axis, only a single frequency component of the Fourier-transformed translating plane wave S will contain power[3]. Then, if we define the k-th frequency component of S as the complex vector $\mathbf{p}$, and the k-th frequency component of the Fourier transformation of the filter to be analyzed as $\mathbf{q}$, the phase difference between these two complex vectors is defined as the angle ψ and given by:

$$\psi = \cos^{-1}\left(\frac{\mathbf{p} \cdot \mathbf{q}}{|\mathbf{p}||\mathbf{q}|}\right) \tag{A.10}$$

where the maximum value of ψ is π, and values of $\psi \geq \pi/2$ result in a zero response due to the ReLU in Eq. A.5.

Convolution response: Dilation & rotation filters

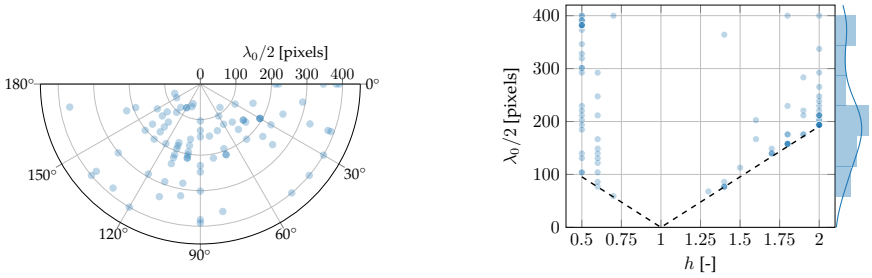
To determine which frequency components of dilation, rotation, and occlusion are more than 90 degrees out of phase, the discrete Fourier transform [371] is used to transform a simulated space-time signal to a representation in the frequency domain. Fig. A.9 shows the convolution response of a dilation filter dw with a translating plane wave s. From this figure, it can be observed that a diamond-like pattern emerges in the response, due to the immeasurable out-of-phase components of dw and s. Because we evaluate the responses along lines orthogonal to the peak response, the pattern perceived is indicated by the dashed lines in the right-most plot of this figure, which correspond to the colored linear patterns in Fig. A.2. Thus, a Gaussian will be perceived along the spatial and the temporal frequency ranges. Hence, we are not able to discern between dilation and translation filters.

Similarly, Fig. A.11 shows the convolution response of a rotation filter cw with s. Note that the 3D power spectrum of cw is different from a Gaussian. At high temporal frequencies (i.e., ±0.2 cycles per frame), the frequency components of cw and s are out-of-phase. Thus, these frequency components will not be detected. The pattern perceived along the varying θ (also shown in Fig. A.2) is two Gaussian lobes at opposite frequency. This pattern is similar to the convolution response of a cosine Gabor filter tuned to stationary patterns (i.e., zero temporal frequency). Therefore, our methodology is also not able to detect rotation filters.

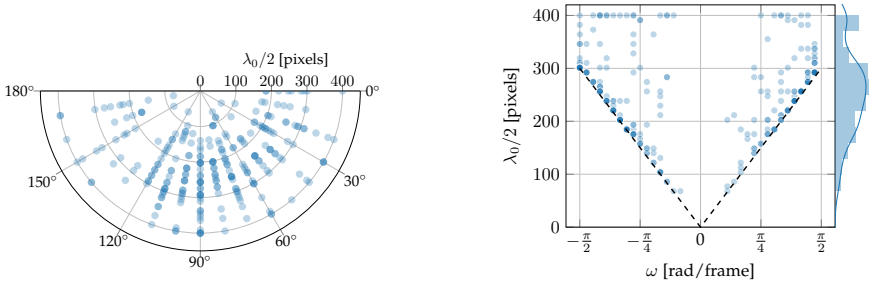
Convolution response: Occlusion filters

Furthermore, we convolve an occlusion filter, using the description of Beauchemin *et al.* [384], with translating plane waves s. Occlusion in the spatiotemporal domain can be described as the combination of a Gaussian, a Heaviside step function, and two translating plane waves translating with different frequencies, as shown in Fig. A.12. The power spectrum of the Fourier-transformed filter can be described as two Gaussian filter pairs with *tails* due to the Heaviside step function. The angle ψ demonstrates that these tails have a large phase difference. Consequently, only the pattern above the dashed line is detected using our methodology, which corresponds to two different Gaussian lobes tuned

[3]Not taking into account the complex conjugate component.



(a) *Left:* Half spatial wavelength $\lambda_0/2$ and initial orientation θ_0. *Right:* Half spatial wavelength $\lambda_0/2$ and scale factor h. The black dashed line indicates the temporal aliasing constraint given by Eq. A.14.



(b) *Left:* Half spatial wavelength $\lambda_0/2$ and initial orientation θ_0. *Right:* Half spatial wavelength $\lambda_0/2$ and angular temporal frequency ω. The black dashed line indicates the temporal aliasing constraint given by Eq. A.17.

Figure A.13: Location of peak response $\hat{r}_0$ per c6 filter in the spatiotemporal frequency domain in response to dilating (*top*) and rotating waves (*bottom*). Only filters whose peak response $\hat{r}_0$ was higher than the maximum found in the translation gridsearch are shown.

to different frequencies. This pattern resembles that of Figs. A.6E and A.7, thus making it likely that the filter represented in these figures is responsive to occlusion. However, it should be noted that we are not able to discern such a pattern from the superposition of two regular Gabor filter pairs tuned to different frequencies.

A.5.2 Methodology

In order to still assess the sensitivity of the c6 filters to dilation and rotation, we come up with a different methodology in which two gridsearches are performed. We assess the locations of the peak responses for filters which have a higher response to dilation or rotation than to translation. We do not classify a filter as either a rotation or dilation filter, since a filter can be sensitive to a composition of these respective motions.

Dilation parametrization

As in [342], a dilating wave d is given by:

$$d(x, y, t) = \cos\left(2\pi F_0(x_r - \alpha x_r t) + \varphi_0\right) \tag{A.11}$$

where α denotes the dilation factor. The training dataset used to train FlowNetS, i.e. FlyingChairs [265], defines scaling motion in terms of the affine scaling factor h. Because

A

the network only takes two frames as input, we define the relation between h and α as follows:

$$h = \frac{1}{1-\alpha} \tag{A.12}$$

The gridsearch is performed for the $[0.5, 2.0]$ range of h, as it encapsulates the values encountered during training. More details about this search space can be found in [105]. In order to mitigate the effect of temporal aliasing, the search space is constrained so that the velocity of a point is not more than half its spatial wavelength $\lambda_0/2$. For a dilating wave, this velocity is given by:

$$v = \Big(\frac{1}{1-\alpha} - 1\Big)x = (h-1)x \tag{A.13}$$

Then, the temporal aliasing constraint for dilating waves is given by:

$$(h-1)x \leq \frac{1}{2}\lambda_0 \tag{A.14}$$

Rotation parametrization

A rotation wave c is given by:

$$c(x, y, t) = \cos\left(2\pi F_0 x_r(t) + \varphi_0\right) \tag{A.15}$$

where $x_r(t)$ varies with time, and is defined as:

$$x_r(t) = x\cos(\theta_0 + \omega t) + y\sin(\theta_0 + \omega t) \tag{A.16}$$

where ω denotes the angular velocity in radians per frame.

The search space for the rotation gridsearch can be found in [105]. A constraint was also added to limit the effect of temporal aliasing. ω can be related to a point at distance m from the center of rotation by $v = \omega m$. The maximum distance from the center of rotation to the edge is equal to half the receptive field size, which is 383 pixels in the c6 layer of our FlowNetS. As the wave rotates around the center pixel, the velocity at this point should thus be lower than half the spatial wavelength. The constraint is given by the following relation:

$$\omega m_{\max} \leq \frac{1}{2}\lambda_0 \tag{A.17}$$

A.5.3 Results

The peak responses of c6 filters which have a higher activation to dilation than to translation (i.e., approximately 15% of the active filters) are shown in Fig. A.13a. These filters show a radially dispersed pattern along the θ-axis, and a peak in the distribution of half spatial wavelengths near 200 pixels. Lastly, peak responses are often close to the temporal aliasing limit and the maximum scaling value of the gridsearch. This is similar to the temporal peak response location for the translation gridsearch (see Fig. A.3).

In Fig. A.13b, the peak responses of the c6 filters for the rotation gridsearch are shown. It can be observed that most filters are active near the temporal translation and temporal rotational aliasing limit. Also, a peak in the distribution of half spatial wavelengths can be identified around 250 pixels, which is slightly higher than expected. A possible explanation

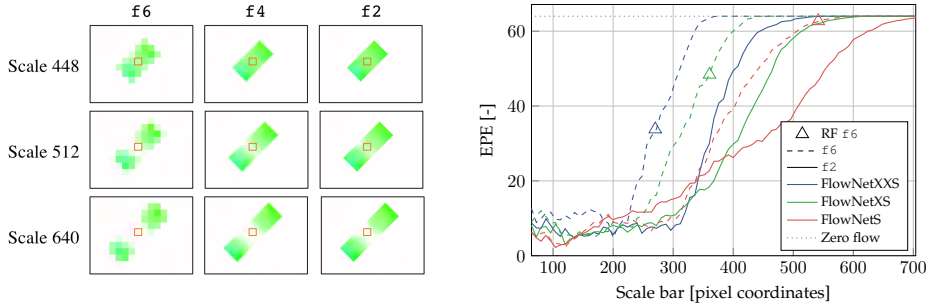


Figure A.14: Response of our FlowNetS and its two variations, FlowNetXS and FlowNetXXS, to diagonally translating bars with motion magnitude $|\mathbf{u}|$ = 64 pixels. *Left:* f6, f4 and f2 FlowNetS flow maps in response to downward-left diagonally translating bars of different scales, using the color-coding scheme from [385]. The red squares highlight the output region used for evaluating the error. *Right:* Endpoint error (EPE↓, lower is better) versus scale of the bar in pixel coordinates. RF f6 indicates the diagonal receptive field size in pixel coordinates corresponding to the f6 flow map.

for this discrepancy is that rotation is actually a 3D motion and thus the scale should also be limited along its radial axis. Approximately 45% of the active c6 filters activate more on rotation than on translation, which could be due to the fact that we do not limit the wavelength along the axis of rotation. The points in the motion field at the far end of the receptive field then move with a very high velocity, and therefore, the response of the filters is higher.

A.6 Solving the aperture problem

In order to determine until what scale of input stimuli FlowNetS can resolve the aperture problem, three different versions of this network are trained under the same circumstances with varying receptive field sizes. The receptive field size is defined as the region in the input images which affects the value of the feature map at a particular layer and feature map location. Therefore, we modify the filter size of the convolutional kernels in c6, which is actually composed of two layers: c6_0 and c6_1. The original (and our) FlowNetS uses 3x3 kernels in these layers, which leads to a receptive field size of 383 pixels in the f6 flow map. We train two additional models with kernels sizes (1x1, 3x3) and (1x1, 1x1) for c6_0 and c6_1, which we name FlowNetXS and FlowNetXXS, and whose f6 receptive field size is 255 pixels and 191 pixels, respectively. For the three of these networks, the receptive field size increases in the expanding part of the architecture due to the upconvolutional layers.

As input, we use a diagonally translating bar of different scales with motion magnitude $|\mathbf{u}|$ = 64 pixels. We determine the error at the center of the bar, and at three flow maps of different resolutions: f6, f4, and f2 (see Fig. A.1).

In Fig. A.14 (left), the FlowNetS response to a downward left translating bar of varying scale is shown. Firstly, the flow becomes more and more refined in the expanding part of the architecture. Secondly, the network is able to extrapolate motion cues from the edges of the bar towards the center, but only to an extent determined by the scale of the bar.

Model name	FlyingChairs test [EPE↓]	MPI Sintel clean train [EPE↓]	MPI Sintel Final train [EPE↓]
FlowNetS [265]	2.71	4.50	5.45
FlowNetS-ours	3.10	5.06	5.81

Table A.2: Performance comparison between the original version of FlowNetS and ours on the MPI-Sintel [381] and FlyingChairs [265] datasets.

Fig. A.14 (right) shows the average endpoint error (EPE↓, lower is better) of FlowNetS, FlowNetXS, and FlowNetXXS in response to two translating bars of different scales moving upward right and downward left, respectively. As shown, the network's robustness to the aperture problem is related to the receptive field size, and networks with larger receptive fields are able to resolve the aperture problem at larger scales.

A.7 Additional experiments

A.7.1 Original vs. our FlowNetS

In Table A.2 a performance comparison between the slightly modified version of FlowNetS studied in this chapter and the original version of Dosovitskiy *et al.* [265] can be found on the FlyingChairs [265] and MPI sintel [381] datasets. As described in Section A.3, our version uses ReLu activations, `pf` layers with 1x1 kernels, no biases, and it was trained for 300k iterations with no data augmentation between frames. As shown, our version has a slightly worse, but comparable performance to the original version. The change in the `pf` size from 3×3 to 1×1 helps interpretability of the analysis, but does bring the total receptive field size in the `c6` layer to 383 pixels as opposed to the original size of 511 pixels.

A.7.2 Generalizability to natural images

To evaluate the generalizability of the fitted bank of translational Gabor filters to natural stimuli, we used the first 500 image pairs of the FlyingChairs dataset [265] and compared the response of the Gabor bank to that of the corresponding filters from the convolutional `c6` layer of FlowNetS [265].

In order to obtain the response of the Gabor filters to a pair of input images, we first rendered the filter bank so that they have the same receptive field as the convolutional filters in the `c6` layer (i.e., 383×383). We convolved the rendered filters over the padded image pair so that the output has the same spatial dimensions as the input images; and then applied the corresponding gain, bias, and the ReLU non-linearity as in Eq. A.5. Lastly, we passed the resulting activations through a series of average pooling operations (same strides and padding as in FlowNetS) to obtain feature maps of the same spatial reso-

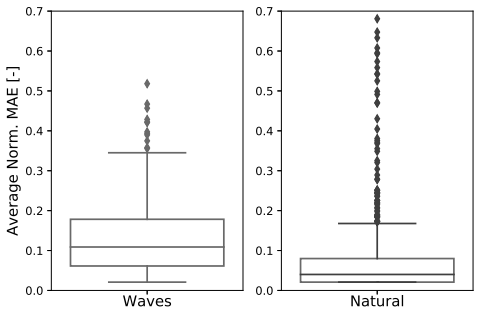


Figure A.15: Average normalized MAE between the activations of convolutional filters in the `c6` layers and the corresponding fitted Gabor filters on datasets comprised of translational plane waves (*left*) and natural stimuli (*right*).

lution as those from c6 filters. These maps are directly comparable and we used the mean absolute error (MAE↓, lower is better) for this purpose. We normalized the MAE for each filter and image pair using the maximum activation, and compute the average MAE over all image pairs in the dataset under analysis with which either the Gabor or the c6 filter present a non-zero response.

Fig. A.15 shows the distributions of the errors on the natural stimuli from FlyingChairs and on a dataset comprised of the translating plane waves that maximally activate each of the fitted Gabor filters (i.e., 592 image pairs). These results show that, in both cases, the behavior of the fitted Gabor filters closely resembles that of the c6 layer. The average error on the translating plane waves is 0.13 (13% of the maximal response), with the error distribution being characterized by only a few outliers (12) starting at 0.34. On the natural stimuli, the average error is 0.08 (8% of the maximal response) with outliers (61) starting at 0.17. We believe the reason for the error being generally lower on the natural stimuli is due to the different image statistics between datasets, where the natural images come from the training data set.

Lastly, with this experiment, we also confirmed that the c6 filters that we found silent during the fitting process (432 filters out of 1024) remain silent when presented with the natural stimuli from FlyingChairs. This, together with the error distributions in Fig. A.15, validates the Gabor fitting process and the use of the translational plane waves as input images.

A.8 Discussion

Impact on Computer Vision

Our results show that the neural responses in the deepest encoding layer of FlowNetS, c6, are well captured by Gabor-like filters. This finding provides insight into the limits and robustness of the approach. Given this core mechanism for estimating optical flow, it is to be expected that the network generalizes quite well to out-of-training-set samples. However, it also raises some concerns, since traditional Gabor filters for optical flow estimation had certain disadvantages. They deal badly with deviations from translation, varying contrast due to changing lighting conditions, and are subject to the uncertainty relation, which corresponds to the balance between localization of the stimuli in the spatial domain and resolution in the frequency domain.

FlowNetS successfully copes with all of these issues. We have shown that deviations from translations are dealt with by additional filters that are sensitive to more complex motion types. Moreover, Mayer *et al.* [386] showed that FlowNet is able to cope with varying contrast over time due to changing lighting conditions. Lastly, we have demonstrated that FlowNetS is able to achieve a better spatial localization of motion cues in the expanding part of the network, thus coping with the uncertainty relation.

In terms of accuracy, FlowNetS did not reach the levels of state-of-the-art methods. For example, it has poor performance on sub-pixel flow [189]. One reason for this might be the large number of strides utilized before the initial flow prediction is made. Also, our analysis shows that a Gabor filter based on two frames results in a large temporal frequency bandwidth, and hence limited performance concerning flow velocity estimation. This is narrowed somewhat by the non-linear transformations due to the ReLU activation

function and bias term. However, our analysis indicates that this could be further improved by using more frames and thus providing more temporal information to the network. Please note that there is an increasing number of multi-frame methods for deep optical flow estimation, e.g., [387–390]. As remarked in [387], most of these methods use multiple images in order to track flow to future frames and track flow back to the past, in order to enhance consistency of the flow. Methods such as StarFlow [390] additionally pass the flow and extracted features from the previous image pairs as input to the deep net, while other methods make use of LSTMs [389]. However, the basic matching still happens between two frames with FlowNetC-like neural correlation blocks. What we propose here is to enter multiple images directly into a FlowNetS-like network in order to reduce the temporal bandwidth, something which to our knowledge has not been investigated yet.

The Gabor-like nature of the neural filters in `c6` may also be a reason for less accuracy; These responses are mapped to coarse flow in a linear way by `pf6`. This means that optical flow velocities that are higher than the filter's tuned velocity, actually lead to an underestimation of the optical flow (due to the bell-shape of the response, see, e.g., Fig. A.4). The network likely copes with this in the following ways. First, it can narrow the response bandwidth with the nonlinear activation function. To see why this helps, think of the extreme in which a neuron would respond in a Dirac-like way to a very specific optical flow velocity. Of course, such a narrow response would then require a very large number of neurons to cover all optical flow velocities. This brings us to the second coping mechanism: the final flow is mostly determined by the neurons in the neural filter bank that are tuned closer to the true optical flow velocity, as they will react more intensely. Finally, the biases in the network can be set in a way to deal with this problem, which is biased since it mostly involves underestimation. Still, it may be worth investigating if different mechanisms would lead to a better accuracy, for instance by introducing a winner-take-all mechanism.

We observed that only 592 of the 1024 `c6` filters have an activation larger than zero. However, the high similarity of the active filters to the Gabor model already suggests that it would also be worth studying a hybrid FlowNetS network, in which there is a fixed Gabor filter bank (extended with rotation and dilation features) followed by a convolutional multi-layer loss flow refinement. This would greatly reduce training time, and, most probably, improve the generalizability of the network.

Finally, our findings for FlowNetS may also be relevant to "PoseNets" (e.g., [391, 392]) that take as input subsequent images and output the relative pose, i.e., an estimate of the translation and rotation between them. Typically, for such relative pose estimation networks a simple encoder structure is used, which is very similar to FlowNetS's structure up to and including `c6`. We expect that optical flow plays a large role in the estimation of translation and rotation between subsequent images, and, given the similar network structure, it is possible that PoseNets also implicitly determine flow with Gabor-like filters before synthesizing the information into a translation and rotation estimate.

Impact on biology

We have used and extended methods from neuropsychology for determining the types of motion filters represented by neurons in the deep `c6` layer of FlowNetS. The analysis gave very similar results to those on neurons in the mammalian visual cortex. First, many filter responses fit very accurately with Gabor filters that capture translational motion. Second,

the spatial and orientation bandwidth statistics show similarity to bandwidths of neurons found in the mammalian visual cortex. We report a median spatial frequency bandwidth of 1.36 octaves, while De Valois *et al.* [393] report 1.4 octaves for the macaque visual cortex. Similarly, we find a median orientation bandwidth of 52 degrees, while De Valois *et al.* [394] find 65 degrees. These similarities may be due to similar optical flow statistics being perceived both by the network and the animals. Third, as in neuropsychological experiments [372], we observed that some filters respond poorly to translating plane waves. Our analysis shows that such poor response may be due to the filters being sensitive to more complex motions such as dilation and rotation. Indeed, in the human brain, channels sensitive to dilation have been found [395]. However, this did not provide conclusive evidence of neurons sensitive to dilation. Our analysis and results suggest that it is worth looking for dilation- and rotation-sensitive neurons in animal brains. In fact, one could even extend the analysis to also check for shear, as this forms an additional basis for the flow field derivatives [236].

A.9 Conclusion

We have employed a spectral response fitting approach from neuropsychology to demonstrate that the deepest layer of FlowNetS essentially encodes a bank of spatiotemporal Gabor filters. Although accurate fits were obtained, the spectral response fitting approach is limited, since it is not able to identify the exact motion pattern causing the maximum activation of a filter. In this work, we have already shown that the network also contains a large number of filters that are more sensitive to dilation and rotation than to translation, but more complex motion filters may be present. Finally, we have studied how FlowNetS tackles the aperture problem. Our results suggest that, on the one hand, the receptive field size is highly correlated to the scale at which the network can resolve the aperture problem. On the other hand, the expanding part of the network allows to solve the aperture problem at slightly larger scales by performing a filling-in function similar to that in mammal vision systems.

Future work could: (i) perform a similar analysis on SpyNet [359], (ii) study the neural response to more complex motion patterns like compositions of affine and 3D motion, as present in more realistic synthetic training datasets (e.g., FlyingThings [396]), (iii) attempt to improve FlowNetS' performance by using smaller strides or more input images, and (iv) employ our extended spectral response fitting method to investigate if animal brains have dilation- and rotation-sensitive neurons as well.

B

Real-Time, Frame-Based, Dense Optical Flow on a Nano Quadcopter

This chapter presents the second research topic that was explored in parallel to the main research questions of this dissertation. We investigate how optical-flow-based autonomous navigation can be realized with a neural network solution on board a nano quadcopter with limited computational resources. To this end, we present NanoFlowNet, a lightweight convolutional neural network for real-time dense optical flow estimation on edge computing hardware. We draw inspiration from recent advances in semantic segmentation for the design of this network. Additionally, we guide the learning of optical flow using motion boundary ground truth data, which improves performance with no impact on latency. Validation results on the MPI-Sintel dataset show the high performance of the proposed network given its constrained architecture. Additionally, we successfully demonstrate the capabilities of NanoFlowNet by deploying it on the ultra-low power GAP8 microprocessor and by applying it to vision-based obstacle avoidance on board a Bitcraze Crazyflie, a 34 g nano quadcopter.

The contents of this chapter have been published in:

R. J. Bouwmeester, **F. Paredes-Vallés**, G. C. H. E. de Croon, *NanoFlowNet: Real-time dense optical flow on a nano quadcopter*, IEEE International Conference on Robotics and Automation (ICRA), 2023.

Contribution: The research leading to this chapter's work was the result of an M.Sc. graduation project conducted by ir. Rik J. Bouwmeester, whom I supervised together with prof. dr. Guido C. H. E. de Croon. Apart from the conceptual collaboration, I specifically helped designing and conducting the flight experiments, as well as analyzing the obtained results.

B.1 Introduction

Safe and reliable navigation of autonomous aerial systems in narrow, cluttered, GPS-denied, and unknown environments is one of the main open challenges in the field of robotics. Because of their small size and agility, micro air vehicles (MAVs) are optimal for this task [5, 397]. Nano quadcopters are a variety of MAVs that are characterized by minimal weight (approx. 30 g) and size (approx. 10 cm rotor-to-rotor) and hence are well suited for deployment under the aforementioned conditions. With the right algorithm design, these nano quadcopters have been demonstrated to be able to perform complex tasks such as exploration [398] or gas source seeking [399]. However, conventional approaches to these problems rely on computationally expensive "map-based" methods that require an array of sensors (e.g., stereo camera, LiDAR) and processors that, in the majority of cases, exceed the payload capacity of these vehicles.

The main approach to autonomous flight of MAVs is based on monocular vision, since a single camera can be light-weight and energy-efficient, while providing rich information on the environment. One of the most important monocular visual cues for navigation is optical flow. Until now, it has been extensively exploited on aerial vehicles with relatively high payload capacity for tasks such as obstacle avoidance [130, 400], and several bio-inspired methods for autonomous navigation [73, 91, 401–403].

Traditionally, the task of monocular optical flow estimation has been performed by hand-crafted methods [163, 343]. However, the field recently shifted toward deep learning approaches [189, 265–267, 347, 359, 404–409], which deliver not only a better performance than the conventional methods but also a faster runtime. Although the focus has largely been on improving performance, efforts have been made to find models of reduced size and faster inference [189, 266, 347, 359, 405, 408, 410]. However, these methods remain computationally expensive, with runtime ranging from several to tens of frames per second (FPS) on desktop GPUs and requiring millions of parameters (and hence large amounts of memory), rendering these models incompatible with edge hardware.

In this work, instead of improving the accuracy of state-of-the-art approaches, we focus on their inference speed and, more particularly, on the deployment of a dense optical flow network on edge devices. To this end, we present *NanoFlowNet*, a lightweight convolutional

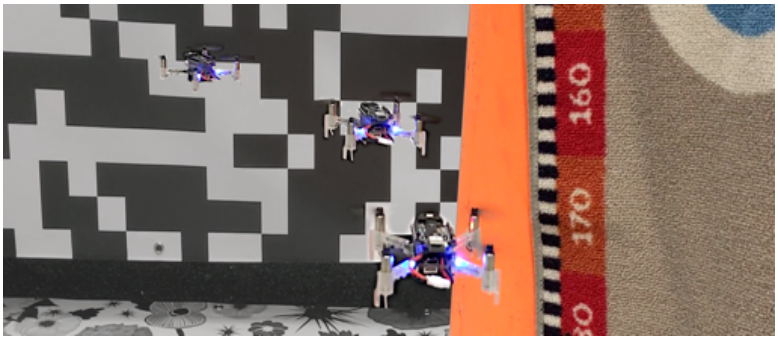


Figure B.1: We demonstrate NanoFlowNet in an obstacle avoidance application on board a nano quadcopter (time-lapse image).

neural network (CNN) architecture for optical flow estimation that, inspired by the semantic segmentation network STDC-Seg [411], achieves real-time inference on the ultra-low power GAP8 multi-core microprocessor on the Bitcraze AI-deck. An overview of the proposed network architecture and its training pipeline can be found in Fig. B.2.

The key contributions of this chapter are listed as follows. First, we introduce NanoFlowNet, a novel lightweight neural network architecture that performs, for the first time, real-time dense optical flow estimation on edge hardware. We validate this network, which runs at 5.5-9.3 FPS on the tiny GAP8 microprocessor, through extensive quantitative and qualitative evaluations on multiple datasets. Second, we show, for the first time, that using motion boundary ground truth to guide the learning of optical flow improves performance while having zero impact on inference latency. Last, we demonstrate the proposed NanoFlowNet in a real-world obstacle avoidance application on board a Bitcraze Crazyflie nano quadcopter.

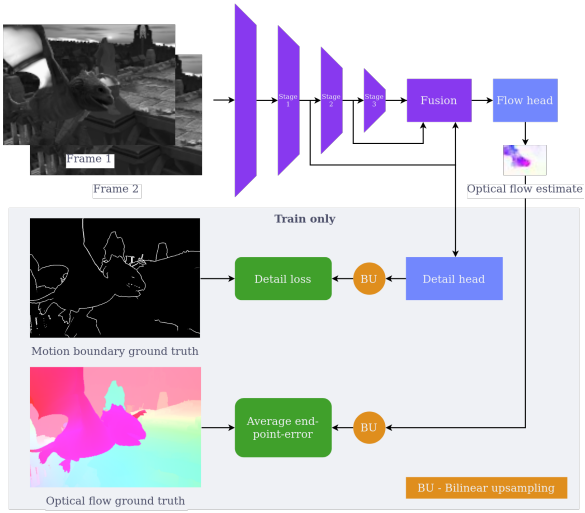


Figure B.2: NanoFlowNet consists of (i) an encoder that extracts features from the input images, (ii) a fusion module that combines features from different levels, and (iii) a motion-boundary-guided detail head, which is only enabled during training, to guide the learning with zero cost to inference latency.

B.2 Related work

B.2.1 Autonomous navigation of nano quadcopters

The limited computational capacity of nano quadcopters (and MAVs in general) puts a constraint on the types of methods that can be used for autonomous navigation. Methods demonstrated on board nano quadcopters can be broadly grouped in model-based reinforcement learning for hovering [412], obstacle avoidance based on dedicated laser ranging sensors [398, 399, 413], and self-motion estimation using optical flow from dedicated optical flow sensors [414] or estimated with external, multi-camera setups [415, 416]. Other methods circumvent the computational constraints of these vehicles by running methods off-board [417–419].

Regarding edge computing hardware, recent works have focused on augmenting the computational power of nano quadcopters without exceeding their payload limitations. Methods based on application-specific integrated circuits [420–423] can efficiently provide information for specific tasks such as simultaneous localization and mapping and visual-inertial odometry but have not yet been presented on a flying drone. More recently, parallel ultra-low power processors introduce energy-efficient multi-core processing to parallelize

visual workloads on edge devices [424]. In this work, we exploit the commercially available off-the-shelf AI-deck from Bitcraze, equipped with the GreenWaves GAP8 system-on-chip (SoC) and an ultra-low power grayscale camera. This nine-core SoC has been used for several end-to-end methods that integrate perception and navigation by directly regressing inputs through a CNN into control commands [424–426]. Instead, in our approach, we calculate optical flow as an intermediate step. This gives us direct control over vehicle behavior and can support multiple optical-flow-based tasks to be performed simultaneously or interchangeably. Our work, motivated by these benefits, is the first to present a fully convolutional neural network for a dense prediction task on board the AI-deck.

B.2.2 Real-time dense inference with CNNs

For the design of NanoFlowNet, we draw inspiration from recent semantic segmentation literature in order to significantly speed up optical flow estimations while retaining performance. More specifically, we draw inspiration from the BiSeNet [427] and STDC-Seg [411] architectures. First, BiSeNet identified a sacrifice of low-level spatial information in previous real-time methods and improved performance by proposing a multi-path architecture in which low-level spatial information is encoded in a separate path. A feature fusion module was proposed to fuse information from the high- and low-level paths, while an attention refinement module refined features through channel attention. Then, the STDC-Seg architecture introduced the STDC module, which increases the receptive field size per layer at a low computational cost. Furthermore, it identified that BiSeNet's spatial path pronounces edges, and replaced the convolutions from the path with a train-time-only "detail head" and "detail loss" to mimic the information passed from the removed convolutions, thus shrinking the model and decreasing latency. The "detail guidance ground truth" was generated by convolving the ground truth segmentation map with a Laplacian kernel.

A few elements of STDC-Seg and BiSeNet have been separately investigated in the context of optical flow. AD-Net [428] showed that channel attention can be beneficial for optical flow estimation, while EDOF [429] fused features from an edge-detector network and an optical flow encoder network for detail-guided optical flow estimation. Similar to STDC-Seg, we use edges to guide the learning.

B.3 Method

For the design of NanoFlowNet, we adopt the STDC-Seg network [411] and modify it to our needs. We replace all regular convolutions with depthwise separable convolutions, and we globally reduce the number of filters by a factor of four to further reduce latency and the number of parameters. We introduce an even smaller model with half of NanoFlowNet's filters (globally) and call it *NanoFlowNet-s*. Further modifications to the architecture are discussed in detail in the following sections.

B.3.1 Motion boundary detail guidance

The closest analogy to detail guidance as used in STDC-Seg is to generate edges from the optical flow ground truth. Instead, we replace this "edge-detect" detail guidance ground truth with motion boundary ground truth from the optical flow datasets. We adopt the focal loss [430] to counter the class imbalance problem.

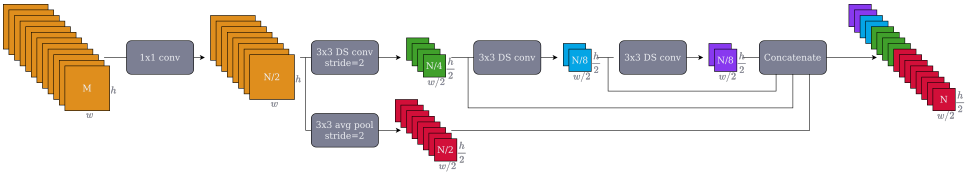


Figure B.3: Original strided STDC module from [411], with the exception that we use depthwise separable convolutions in place of all non-pointwise convolutions. We use ReLU activations after all layers in the block. M denotes the number of input features, while N is the number of output features.

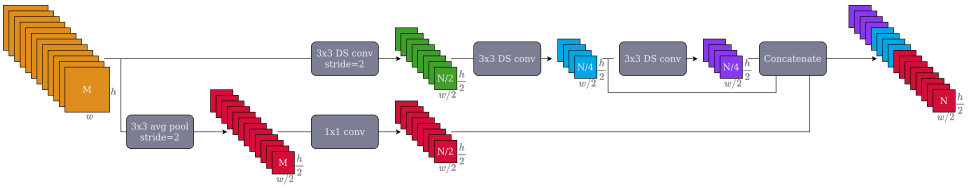


Figure B.4: Our modified strided STDC module. We reorganize the operations to minimize the spatial resolution pointwise convolutions have to perform on.

B.3.2 Strided STDC module redesign

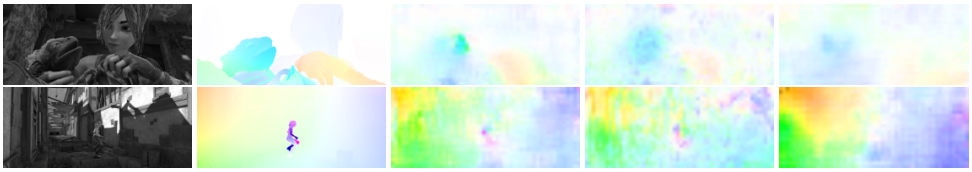
We modify the strided STDC modules from STDC-Seg [411] to further decrease latency. The original and modified strided STDC modules can be found in Figs. B.3 and B.4, respectively. First, following the insights of several low-latency literature methods [431–435], we replace all convolutions in the STDC module with depthwise separable convolutions due to their low computational expense. Second, we identify that the pointwise convolution operation in the strided STDC module is the most expensive in terms of the number of multiply-accumulate (MAC) operations. By relocating this operation to the bottom path after the average pooling operation, we make the strided STDC block computationally more tractable overall while increasing the number of features in the top path and the number of features with a large receptive field size in the concatenated output. Our modified blocks lead to a reduction of over 50% of the MAC operations in stage 1, and of over 10% in stages 2 and 3.

B.3.3 Reduced input/output dimensionality

We design the network for low-resolution input and downscale all dataset's input frames, optical flow, and motion boundary ground truth accordingly. The scaling factor is picked such that the resulting data resolution closely matches the target application resolution (approx. qqVGA, 160 × 120 pixels). Horizontal and vertical scalings are identical, to fix the aspect ratio in an attempt to retain naturalism. This allows us to make the network shallower by dropping the first (expensive) convolution altogether and thus decrease latency while maintaining feature sizes in the deepest layers. The downscaled training data matches the low-resolution cameras found on nano quadcopters more closely, making our synthetic dataset more naturalistic for our intended application. As an added benefit, working with downscaled data significantly speeds up training. The primary downside of reduced input resolution is the loss of information, in particular we will miss out on

Method	EPE↓		FPS↑		No. Params. (k)
	Clean	Final	GPU[1]	GAP8[2]	
FlowNet2-xs	9.05	9.46	150	-	1,978.25
NanoFlowNet (ours)	**7.12**	**7.98**	141	5.57	170,88
NanoFlowNet-s (ours)	9.56	10.05	**151**	**9.34**	46.75

Table B.1: Quantitative results on MPI Sintel (train). [1]At a resolution of 96 × 224. [2]At a resolution of 112 × 160, including vision thread. Best in bold, runner up underlined.



(a) Input frame I (b) Ground truth (c) NanoFlowNet (d) NanoFlowNet-s (e) FlowNet2-xs

Figure B.5: Qualitative comparison of optical flow estimates by NanoFlowNet(-s) and FlowNet2-xs on the MPI Sintel (train) clean pass.

small objects and small displacements that are not captured by the resolution. To be able to compare with existing optical flow works, we benchmark performance at native dataset resolution, since downscaling of flow magnitudes results in lower endpoint error (EPE↓, lower is better) without a qualitative improvement.

Lastly, we design our network for grayscale input images, saving two thirds of the on-board memory dedicated to the input frames and decreasing the computational cost of the first layer (at a loss of color information).

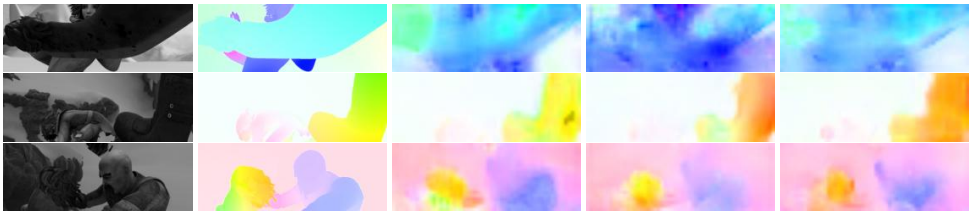
B.4 Experiments

B.4.1 Implementation details

All models are trained for 300 epochs on FlyingChairs2 [265, 436], a regenerated FlyingChairs dataset with motion boundary ground truth. We use the Adam optimizer [173], with learning rate 1e-3 and a batch size of 8. After this, we fine-tune our architectures on FlyingThings3D [396] for 200 epochs with a learning rate of 1e-4.

Given the scaling and conversion to grayscale of the input data, our network is not directly comparable with results reported by other works. For comparison, we retrain one of the fastest networks in literature, Flownet2-s [189], on the same data. Given the reduction in resolution, we drop the deepest two layers to maintain a reasonable feature size in the deepest layers, and name the model *Flownet2-xs*.

We run all experiments in a docker environment with TensorFlow 2.8.0, CUDA 11.2, CUDNN 8.1.0, TensorRT 7.2.2 on an NVIDIA GeForce GTX 1070 Max-Q with batch size 1 for benchmarking latency.



(a) Input frame I (b) Ground truth (c) Motion-boundary guidance (d) Edge-detect guidance (e) No detail guidance

Figure B.6: Qualitative comparison of different detail guidance methods on the MPI Sintel (train) clean pass.

B.4.2 Performance and latency on public benchmarks

We evaluate the trained networks on the unseen MPI Sintel train subset, on both the clean and final pass. Quantitative results can be found in Table B.1. Regarding accuracy, according to these results, our NanoFlowNet performs better than the squeezed FlowNet2-xs architecture, despite using less than 10% of the parameters. With respect to runtime, FlowNet2-xs does not fit on the GAP8 microprocessor due to the network size (i.e., lack of memory). To put the achieved latency of NanoFlowNet in perspective, we execute FlowNet2-xs' first two convolutions and the final prediction layer on the GAP8. The three-layer architecture achieves 4.96 FPS, which is slower than running the entire NanoFlowNet (5.57 FPS). On laptop GPU hardware, NanoFlowNet achieves comparable FPS to FlowNet2-xs. NanoFlowNet-s has lower performance than both other models, but has a low parameter count with only 27% of NanoFlowNet's and 2.4% of FlowNet2-xs's parameters, and is the fastest out of all the networks tested.

Qualitative results, presented in Fig. B.5, confirm that NanoFlowNet makes the most accurate optical flow estimates out of all the networks tested. Interestingly, both NanoFlowNet and NanoFlowNet-s appear to detect displacements of smaller objects, which FlowNet2-xs misses. However, NanoFlowNet-s' flow estimates are highly noisy.

B.4.3 Additional experiments

Motion boundaries detail guidance

We verify the effectiveness of motion boundary detail guidance by retraining two additional networks, one with detail guidance based on the optical flow ground truth convolved with a Laplacian kernel (further referred to as "edge-detect guidance"), and another one with no detail guidance. Quantitative and qualitative results can be found in Table B.2 and Fig. B.6. As shown, motion boundary detail guidance improves results and outperforms edge detect detail guidance. Since all these guidance methods only affect (i.e., guide) the training behavior, all methods have identical latency. Qualitative results show that motion-boundary-guided optical flow best defines moving objects, and shows the least "leakage" of foreground objects into the background.

Detail guidance	EPE↓	
	Clean	Final
None	7.636	<u>8.119</u>
Edge detect	<u>7.404</u>	8.141
Motion boundaries	**7.122**	**7.979**

Table B.2: Quantitative comparison of different methods of detail guidance on MPI Sintel (train). Best in bold, runner up underlined.

Strided STDC module redesign

Table B.3 shows the effects of the strided STDC module redesign. The network with the redesigned module is both faster (both on laptop GPU and the GAP8 microprocessor) and more accurate.

STDC block	EPE↓		FPS↑	
	Clean	Final	GPU[1]	GAP8[2]
Unmodified	7.483	8.114	136	4.84
Modified	**7.122**	**7.979**	**141**	**5.57**

Table B.3: Quantitative comparison of the original and the modified STDC block on MPI Sintel (train). [1]At a resolution of 96 × 224. [2]At a resolution of 112 × 160, including vision thread. Best in bold.

Reduced input dimensionality

A comparison between training and inferring on grayscale images compared to color images can be found in Table B.4. Our grayscale model outperforms the color variant. We hypothesize that this is due to the limited capacity of the network. The latency of the grayscale model on the GAP8 is lower due to reduced data transfer and a cheaper first convolution.

B.4.4 Obstacle avoidance application

We deploy the proposed NanoFlowNet architecture on a Crazyflie 2.x equipped with the AI-deck and the flow-deck for the task of vision-based obstacle avoidance. We use the AI-deck to capture images with the front-facing camera and to run optical flow inference and processing. The downward-facing optical flow deck is used for positioning only. The total flight platform weighs in at 34 g. See Fig. B.8 for a picture of the platform.

Mode	EPE↓		FPS↑	
	Clean	Final	GPU[1]	GAP8[2]
Color	7.726	8.344	**141**	5.18
Grayscale	**7.122**	**7.979**	**141**	**5.57**

Table B.4: Quantitative comparison of grayscale vs. color input frame-based architectures. [1]At a resolution of 96 × 224. [2]At a resolution of 112 × 160, including vision thread. Best in bold.

Control strategy

We implement the horizontal balance strategy from [437, 438], with which the yaw rate $\dot{\psi}$ is set based on the error e_{rl} between the sum of flow magnitudes in the left and right half

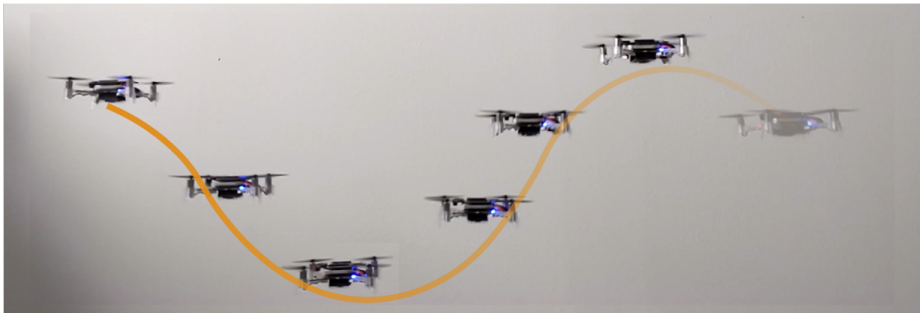


Figure B.7: Inspired by GapFlyt [130], we deliberately let the quadcopter oscillate vertically to generate additional optical flow.

of the flow estimate (see Eq. B.1). We set gains $k_p = 0.0126$ and $k_d = 0.0018$ experimentally. The forward velocity of the quadcopter is set at 0.2 m/s.

$$\dot{\psi} = k_p e_{rl} + k_d \dot{e_{rl}} \tag{B.1}$$

We augment the balance strategy by implementing active oscillations (a cyclic up-down movement, see Fig. B.7) which results in additional optical flow being generated across the field of view (FOV). This is particularly helpful for avoiding objects in the direction of horizontal travel. Up-down rather than left-right surveying favors detecting obstacles wider than taller in nature, but is much simpler to combine with the left-right balance strategy. Additionally, left-right surveying requires rolling, which introduces rotational flow that does not contain depth information.

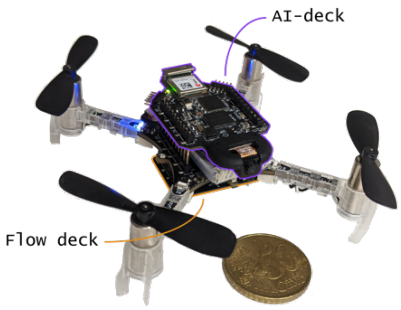


Figure B.8: Crazyflie 2.x equipped with (i) the AI-deck used for image acquisition using a front-facing camera and to run optical flow inference, and (ii) the downward-facing flow-deck used only for positioning.

We implement both the CNN and calculation of e_{rl} on the GAP8 microprocessor of the AI-deck. Calculating the flow error on the AI-deck significantly reduces the amount of data that needs to be transmitted over UART to the autopilot. The calculation of the yaw rate is done on the Crazyflie 2.x, and fed into the controller.

AI-deck implementation

The CNN processing power on the AI-deck comes from the GreenWaves Technologies GAP8. The chip is organized around the central single-core fabric controller (FC) and the eight-core cluster (CL) for parallelized workloads. For our application, we run FC@250MHz, CL@230MHz, and VDD@1.2V.

Our AI-deck is equipped with the HM01B0 monochrome camera, which supports a resolution of up to 324×324, a QVGA (244×324) window mode, a 2×2 monochrome binning mode, and cropping. For our application we enable both the window mode and binning mode (122 × 162) and take a central crop of 112 × 160, to ensure a matched spatial resolution of upsampled and skipped features in the network architecture. At our input resolution, using grayscale versus color reduces the L2 memory usage on the AI-deck by 14%. This additional L2 memory is made available to the AutoTiler, which improves inference time by reducing the number of data transfers.

In this work, we utilize the GreenWaves Technologies GAPflow toolset for porting our CNN to the GAP8. NNTool takes a TensorFlow Lite or ONNX CNN description and maps all operations and parameters to a representation compatible with AutoTiler, the GAPflow tiling solver. We use NNTool to implement 8-bit post-training quantization to our CNN. We quantize on images from the MPI Sintel dataset [381] and achieve an average signal to quantization noise ratio of 10.

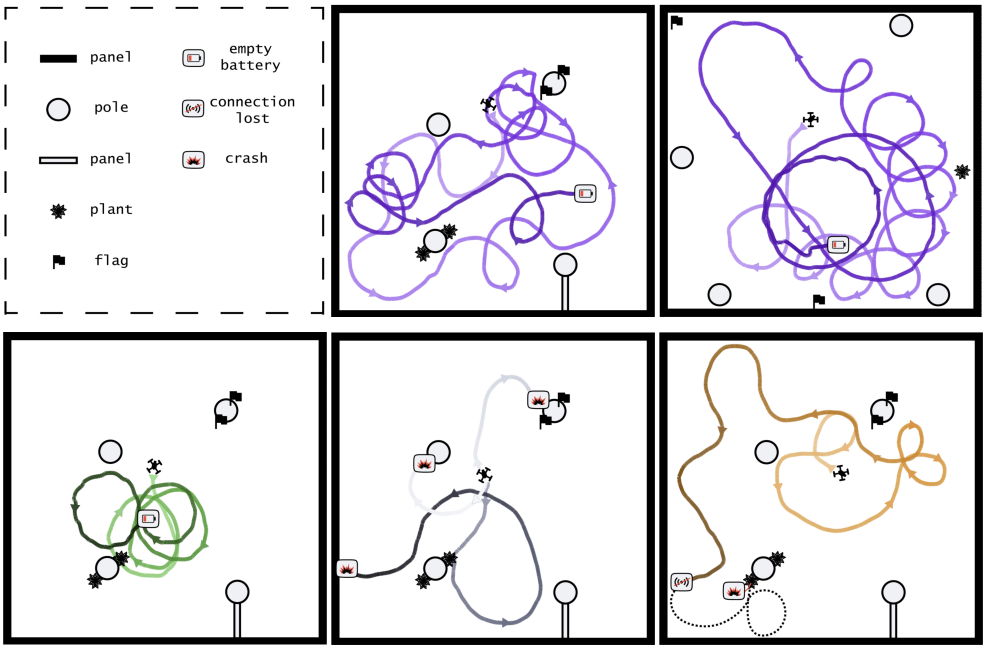


Figure B.9: Results of multiple obstacle avoidance runs in cluttered and open environments. Position recorded with a motion capture system.

Experimental setup

We compose two indoor environments for obstacle avoidance. First, an open environment, with obstacles exclusively placed at the outline of the environment. Second, a cluttered environment, with obstacles placed throughout (see Fig. B.10). Obstacles include textured and untextured poles, synthetic plants, flags, or panels. Both environments are enclosed with textured panels to trap the quadcopter inside. Panel textures consist of forest texture, data matrix texture, and a drone racing gate texture. In both environments, we augment the enclosure's texture with highly textured mats and curtains.

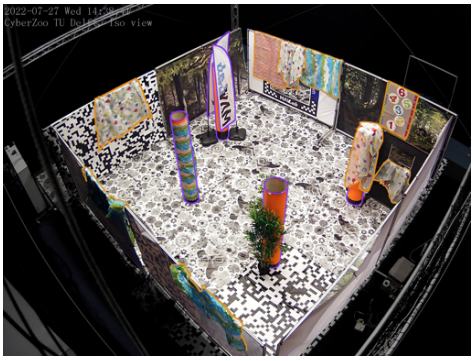


Figure B.10: Overview of the cluttered, obstacle avoidance environment. Obstacles are outlined in purple, while texture-enhancing mats and curtains in orange.

The simple proof-of-concept control algorithm has no dedicated method of dealing with head-on collisions. By placing obstacles around the perimeter of the open environment we minimize the risk of a head-on collision with the panels as they introduce an imbalance of optical flow, even on a fully perpendicular collision path with a panel.

For each experiment, we start the quadcopter at approximately the same location, with varying heading. We let the quadcopter run until a collision or empty battery. We record flight positioning data with a motion capture system for post-flight analysis only and record experiments with an ISO view and top view camera.

Results

Flight paths extracted from the motion capture system are plotted on maps of the environment and can be found in Fig. B.9. The control algorithm is most robust in the open environment, with the quadcopter managing to drain a full battery without crashing. In the cluttered environment, performance is much more variable. Especially in occasions where obstacles are in close proximity to one another, the quadcopter tends to successfully avoid an obstacle, only to collide with another during the maneuver. Adding a head-on collision detection based on the detection of the focus-of-expansion (FOE) and divergence estimation (e.g., [91]) could help avoid obstacles in these cases.

In several of the successful avoidances, the quadcopter initially responds weakly to the obstacle, only to turn away more harshly when the course has already been corrected sufficiently. This behavior is expected because of two reasons. First, the optical flow due to forward movement is zero at the FOE and maximum at the edge of the peripheral vision. Second, due to the fact that the obstacles take up more of the FOV when they are in closer proximity to the quadcopter, they generate more optical flow. This behavior could be corrected by weighing the optical flow more heavily towards the center of the image.

Another notable feature of the flight paths is that the nano quadcopter frequently appears to enter a spiraling path. The control algorithm is overreacting to stimuli from across the environment. Despite this, the behavior is consistent, the resulting paths are still exploring the environments, and the nano quadcopter is able to break out of the spiraling motion by approaching a panel (see Fig. B.9, top right) or approaching an obstacle (see Fig. B.9, top center).

B.5 Conclusion

In this work, we introduced a lightweight CNN architecture for dense optical flow estimation on edge hardware, called NanoFlowNet. We achieved real-time latency on the AI-deck. Furthermore, we showed that training our network guided on motion boundaries improves performance at zero cost to latency. Finally, we implemented NanoFlowNet in a real-world obstacle avoidance application on board a Bitcraze Crazyflie nano quadcopter. For future work, we expect examples that take more advantage of the dense information in the generated optical flow field.

Supplementary material

Video summary of the approach: https://youtu.be/lKkO1VvE4VU

Project code: https://github.com/tudelft/nanoflownet

References

References

[1] Jamie Enoch, Leanne McDonald, Lee Jones, Pete R. Jones, and David P. Crabb. Evaluating whether sight is the most valued sense. *JAMA Ophthalmology*, 137(11):1317–1320, 2019.

[2] Berthold Horn, Berthold Klaus, and Paul Horn. *Robot vision*. MIT press, 1986.

[3] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Adv. Neural Inform. Process. Syst.*, pages 1097–1105, 2012.

[4] Jurgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Netw.*, 61:85–117, 2015.

[5] Dario Floreano and Robert J. Wood. Science, technology and the future of small autonomous drones. *Nature*, 521(7553):460–466, 2015.

[6] J. T. Vance, I. Faruque, and J. S. Humbert. Kinematic strategies for mitigating gust perturbations in insects. *Bioinspiration Biomimetics*, 8(1):016004, 2013.

[7] S. A. Combes, D. E. Rundle, J. M. Iwasaki, and James Dewitt Crall. Linking biomechanics and ecology through predator–prey interactions: Flight performance of dragonflies and their prey. *J. Experiment. Biology*, 215(6):903–913, 2012.

[8] Florian T. Muijres, Michael J. Elzinga, Johan M. Melis, and Michael H. Dickinson. Flies evade looming targets by executing rapid visually directed banked turns. *Science*, 344(6180):172–177, 2014.

[9] Randolf Menzel and Uwe Greggers. The memory structure of navigation in honeybees. *J. Comparative Physiology A*, 201:547–561, 2015.

[10] Guido C. H. E. de Croon, K. M. E. De Clercq, Remes Ruijsink, Bart D. W. Remes, and Christophe De Wagter. Design, aerodynamics, and vision-based control of the DelFly. *Int. J. Micro Air Vehicles*, 1(2):71–97, 2009.

[11] Matěj Karásek, F. T. Muijres, Christophe De Wagter, Bart D. W. Remes, and Guido C. H. E. de Croon. A tailless aerial robotic flapper reveals that flies use torque coupling in rapid banked turns. *Science*, 361(6407):1089–1094, 2018.

[12] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. A 128×128 120 db 15μs latency asynchronous temporal contrast vision sensor. *IEEE J. Solid-State Circuits*, 43(2):566–576, 2008.

[13] Christoph Posch, Teresa Serrano-Gotarredona, Bernabe Linares-Barranco, and Tobi Delbruck. Retinomorphic event-based vision sensors: bioinspired cameras with spiking output. *IEEE Proc.*, 102(10):1470–1484, 2014.

[14] Mike Davies, Narayan Srinivasa, Tsung-Han Lin, Gautham Chinya, Yongqiang Cao, Sri Harsha Choday, Georgios Dimou, Prasad Joshi, Nabil Imam, Shweta Jain, Yuyun Liao, Chit-Kwan Lin, Andrew Lines, Ruokun Liu, Deepak Mathaikutty, Steven McCoy, Arnab Paul, Jonathan Tse, Guruguhanathan Venkataramanan, Yi-Hsin Weng, Andreas Wild, Yoonseok Yang, and Hong Wang. Loihi: A neuromorphic manycore processor with on-chip learning. *IEEE Micro*, 38(1):82–99, 2018.

[15] Misha Mahowald and Carver A. Mead. The silicon retina. *An Analog VLSI System for Stereoscopic Vision*, pages 4–65, 1994.

[16] Carver A. Mead and Misha A. Mahowald. A silicon model of early visual processing. *Neural Netw.*, 1(1):91–97, 1988.

[17] Donhee Ham, Hongkun Park, Sungwoo Hwang, and Kinam Kim. Neuromorphic electronics based on copying and pasting the brain. *Nature Electronics*, 4(9):635–644, 2021.

[18] Guillermo Gallego, Tobi Delbruck, Garrick Orchard, Chiara Bartolozzi, Brian Taba, Andrea Censi, Stefan Leutenegger, Andrew Davison, Jörg Conradt, Kostas Daniilidis, et al. Event-based vision: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[19] Maxence Bouvier, Alexandre Valentian, Thomas Mesquida, Francois Rummens, Marina Reyboz, Elisa Vianello, and Edith Beigne. Spiking neural networks hardware implementations and challenges: A survey. *ACM J. Emerging Tech. Comput. Syst.*, 15(2):1–35, 2019.

[20] Jason K. Eshraghian, Max Ward, Emre Neftci, Xinxin Wang, Gregor Lenz, Girish Dwivedi, Mohammed Bennamoun, Doo Seok Jeong, and Wei D. Lu. Training spiking neural networks using lessons from deep learning. *arXiv:2109.12894*, 2021.

[21] James J. Gibson. The perception of the visual world. *Houghton Mifflin*, 1950.

[22] Github repository: Event-based vision resources. https://github.com/uzh-rpg/event-based_vision_resources, 2018–Present.

[23] Garrick Orchard, Ajinkya Jayawant, Gregory K. Cohen, and Nitish Thakor. Converting static image datasets to spiking neuromorphic datasets using saccades. *Frontiers Neuroscience*, 9:437, 2015.

[24] Arnon Amir, Brian Taba, David Berg, Timothy Melano, Jeffrey McKinstry, Carmelo Di Nolfo, Tapan Nayak, Alexander Andreopoulos, Guillaume Garreau, Marcela Mendoza, et al. A low power, fully event-based gesture recognition system. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7243–7252, 2017.

[25] Alex Z. Zhu and Liangzhe Yuan. EV-FlowNet: Self-supervised optical flow estimation for event-based cameras. In *Robot.: Science Syst.*, 2018.

[26] Zhaoning Sun, Nico Messikommer, Daniel Gehrig, and Davide Scaramuzza. ESS: Learning event-based semantic segmentation from still images. In *European Conf. Comput. Vis.*, pages 341–357, 2022.

[27] Nico Messikommer, Carter Fang, Mathias Gehrig, and Davide Scaramuzza. Data-driven feature tracking for event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[28] Manasi Muglikar, Leonard Bauersfeld, Diederik Paul Moeys, and Davide Scaramuzza. Event-based shape from polarization. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023.

[29] 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.*, pages 5633–5643, 2019.

[30] Simon Schaefer, Daniel Gehrig, and Davide Scaramuzza. AEGNN: Asynchronous event-based graph neural networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12371–12381, 2022.

[31] Mathias Gehrig and Davide Scaramuzza. Recurrent vision transformers for object detection with event cameras. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 13884–13893, 2023.

[32] Nikola Zubić, Daniel Gehrig, Mathias Gehrig, and Davide Scaramuzza. From chaos comes order: Ordering event representations for object detection. *arXiv:2304.13455*, 2023.

[33] Alex Z. 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.*, pages 989–997, 2019.

[34] Mathias Gehrig, Mario Millhäusler, Daniel Gehrig, and Davide Scaramuzza. E-RAFT: Dense optical flow from event cameras. In *Int. Conf. 3D Vis.*, pages 197–206. IEEE, 2021.

[35] Christoph Stöckl and Wolfgang Maass. Optimized spiking neurons can classify images with high accuracy through temporal coding with two spikes. *Nature Mach. Intell.*, pages 1–9, 2021.

[36] Tong Bu, Wei Fang, Jianhao Ding, PengLin Dai, Zhaofei Yu, and Tiejun Huang. Optimal ANN-SNN conversion for high-accuracy and ultra-low-latency spiking neural networks. *arXiv:2303.04347*, 2023.

[37] Emre O. Neftci, Hesham Mostafa, and Friedemann Zenke. Surrogate gradient learning in spiking neural networks: Bringing the power of gradient-based optimization to spiking neural networks. *IEEE Signal Process. Magazine*, 36(6):51–63, 2019.

[38] Friedemann Zenke and Tim P. Vogels. The remarkable robustness of surrogate gradient learning for instilling complex function in spiking neural networks. *Neural Comput.*, pages 1–27, 2021.

[39] Loïc Cordone, Benoît Miramond, and Sonia Ferrante. Learning from event cameras with sparse spiking convolutional neural networks. In *Int. Joint Conf. Neural Netw.*, pages 1–8, 2021.

[40] Wei Fang, Zhaofei Yu, Yanqi Chen, Timothee Masquelier, Tiejun Huang, and Yonghong Tian. Incorporating learnable membrane time constant to enhance learning of spiking neural networks. In *Int. Conf. Comput. Vis.*, pages 2661–2671, 2021.

[41] Natalia Caporale and Yang Dan. Spike timing–dependent plasticity: A Hebbian learning rule. *Annual Review Neuroscience*, 31:25–46, 2008.

[42] Donald O. Hebb. *The organization of behavior: A neuropsychological theory.* Wiley, 1952.

[43] Timothée Masquelier and Simon J. Thorpe. Unsupervised learning of visual features through spike timing dependent plasticity. *Public Library Science: Comput. Biology*, 3(2):247–257, 2007.

[44] Peter U. Diehl and Matthew Cook. Unsupervised learning of digit recognition using spike-timing-dependent plasticity. *Frontiers Comput. Neuroscience*, 9:1–9, 2015.

[45] Saeed R. Kheradpisheh, Mohammad Ganjtabesh, Simon J. Thorpe, and Timothée Masquelier. STDP-based spiking deep convolutional neural networks for object recognition. *Neural Netw.*, 99:56–67, 2018.

[46] Amirhossein Tavanaei and Anthony S. Maida. Multi-layer unsupervised learning in a spiking convolutional neural network. In *Int. Joint Conf. Neural Netw.*, pages 2023–2030, 2017.

[47] Amar Shrestha, Khadeer Ahmed, Yanzhi Wang, and Qinru Qiu. Stable spike-timing dependent plasticity rule for multilayer unsupervised and supervised learning. In *Int. Joint Conf. Neural Netw.*, pages 1999–2006, 2017.

[48] Paul A. Merolla, John V. Arthur, Rodrigo Alvarez-Icaza, Andrew S. Cassidy, Jun Sawada, Filipp Akopyan, Bryan L. Jackson, Nabil Imam, Chen Guo, Yutaka Nakamura, et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science*, 345(6197):668–673, 2014.

[49] Johannes Schemmel, Daniel Brüderle, Andreas Grübl, Matthias Hock, Karlheinz Meier, and Sebastian Millner. A wafer-scale neuromorphic hardware system for large-scale neural modeling. In *IEEE Int. Symp. Circuits Syst.*, pages 1947–1950. IEEE, 2010.

[50] Qian Liu, Ole Richter, Carsten Nielsen, Sadique Sheik, Giacomo Indiveri, and Ning Qiao. Live demonstration: Face recognition on an ultra-low power event-driven convolutional neural network ASIC. In *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2019.

[51] Garrick Orchard, E Paxon Frady, Daniel Ben Dayan Rubin, Sophia Sanborn, Sumit Bam Shrestha, Friedrich T Sommer, and Mike Davies. Efficient neuromorphic signal processing with Loihi 2. In *IEEE Signal Process. Syst. Worksh.*, pages 254–259, 2021.

[52] Steve B. Furber, Francesco Galluppi, Steve Temple, and Luis A. Plana. The SpiNNaker project. *IEEE Proc.*, 102(5):652–665, 2014.

[53] Massimo Antonio Sivilotti. *Wiring considerations in analog VLSI systems, with application to field-programmable networks.* California Institute of Technology, 1991.

[54] Kwabena A. Boahen. Point-to-point connectivity between neuromorphic chips using address events. *IEEE Trans. Circuits Syst. II: Analog Digital Signal Process.*, 47(5):416–434, 2000.

[55] Shih-Chii Liu, Tobi Delbruck, Giacomo Indiveri, Adrian Whatley, and Rodney Douglas. *Event-based neuromorphic systems.* John Wiley & Sons, 2014.

[56] Bodo Rueckauer and Tobi Delbruck. Evaluation of event-based algorithms for optical flow with ground-truth from inertial measurement sensor. *Frontiers Neuroscience*, 10:1–17, 2016.

[57] Ryad Benosman, Sio-Hoi Ieng, Charles Clercq, Chiara Bartolozzi, and Mandyam Srinivasan. Asynchronous frameless event-based optical flow. *Neural Netw.*, 27:32–37, 2012.

[58] Elias Mueggler, Chiara Bartolozzi, and Davide Scaramuzza. Fast event-based corner detection. In *British Mach. Vis. Conf.*, pages 33–1, 2017.

[59] Ignacio Alzugaray and Margarita Chli. Asynchronous corner detection and tracking for event cameras in real time. *IEEE Robot. Autom. Lett.*, 3(4):3177–3184, 2018.

[60] Daniel Gehrig, Henri Rebecq, Guillermo Gallego, and Davide Scaramuzza. EKLT: Asynchronous photometric feature tracking using events and frames. *Int. J. Comput. Vis.*, 128(3):601–618, 2020.

[61] Hanme Kim, Stefan Leutenegger, and Andrew J. Davison. Real-time 3D reconstruction and 6-DoF tracking with an event camera. In *European Conf. Comput. Vis.*, pages 349–364, 2016.

[62] Yi Zhou, Guillermo Gallego, Henri Rebecq, Laurent Kneip, Hongdong Li, and Davide Scaramuzza. Semi-dense 3D reconstruction with a stereo event camera. In *European Conf. Comput. Vis.*, pages 235–251, 2018.

[63] Henri Rebecq, Guillermo Gallego, Elias Mueggler, and Davide Scaramuzza. EMVS: Event-based multi-view stereo—3D reconstruction with an event camera in real-time. *Int. J. Comput. Vis.*, 126(12):1394–1414, 2018.

[64] Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Continuous-time visual-inertial odometry for event cameras. *IEEE Trans. Robot.*, 34(6):1425–1440, 2018.

[65] Yi Zhou, Guillermo Gallego, and Shaojie Shen. Event-based stereo visual odometry. *IEEE Trans. Robot.*, 37(5):1433–1450, 2021.

[66] Javier Hidalgo-Carrió, Guillermo Gallego, and Davide Scaramuzza. Event-aided direct sparse odometry. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5781–5790, 2022.

[67] Antoni Rosinol Vidal, Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Ultimate SLAM? Combining events, images, and IMU for robust visual SLAM in HDR and high-speed scenarios. *IEEE Robot. Autom. Lett.*, 3(2):994–1001, 2018.

[68] Jorg Conradt, Raphael Berner, Matthew Cook, and Tobi Delbruck. An embedded aer dynamic vision sensor for low-latency pole balancing. In *Int. Conf. Comput. Vis. Worksh.*, pages 780–785. IEEE, 2009.

[69] Tobi Delbruck and Manuel Lang. Robotic goalie with 3 ms reaction time at 4% cpu load using event-based dynamic vision sensor. *Frontiers Neuroscience*, 7, 2013.

[70] Bas J. Pijnacker Hordijk, Kirk Y. W. Scheper, and Guido C. H. E. de Croon. Vertical landing for micro air vehicles using event-based optical flow. *J. Field Robot.*, 35(1):69–90, 2018.

[71] Kirk Y. W. Scheper and Guido C. H. E. de Croon. Evolution of robust high speed optical-flow-based landing for autonomous MAVs. *Robot. Autom. Syst.*, 124, 2020.

[72] Guido C. H. E. de Croon, H. W. Ho, Christophe De Wagter, Erik van Kampen, Bart Remes, and Q. P. Chu. Optic-flow based slope estimation for autonomous landing. *Int. J. Micro Air Vehicles*, 5(4):287–297, 2013.

[73] Guido C. H. E. de Croon. Monocular distance estimation with optical flow maneuvers and efference copies: A stability-based strategy. *Bioinspiration Biomimetics*, 11, 2016.

[74] Hann W. Ho and Guido C. H. E. de Croon. Characterization of flow field divergence for MAVs vertical control landing. In *AIAA Guidance, Navigation, and Control Conf.*, pages 1–13, 2016.

[75] Hann W. Ho, Christophe De Wagter, Bart D. W. Remes, and Guido C. H. E. de Croon. Optical-flow based self-supervised learning of obstacle appearance applied to MAV landing. *Robot. Autonomous Systems*, 100:78–94, 2018.

[76] Sihao Sun, Giovanni Cioffi, Coen De Visser, and Davide Scaramuzza. Autonomous quadrotor flight despite rotor failure with onboard vision sensors: Frames vs. events. *IEEE Robot. Autom. Lett.*, 6(2):580–587, 2021.

[77] Davide Falanga, Kevin Kleber, and Davide Scaramuzza. Dynamic obstacle avoidance for quadrotors with event cameras. *Science Robot.*, 5(40), 2020.

[78] Nitin J. Sanket, Chethan M. Parameshwara, Chahat Deep Singh, Ashwin V. Kuruttukulam, Cornelia Fermüller, Davide Scaramuzza, and Yiannis Aloimonos. EVDodgeNet: Deep dynamic obstacle dodging with event cameras. In *IEEE Int. Conf. Robot. Autom.*, pages 10651–10657, 2020.

[79] Raoul Dinaux, Nikhil Wessendorp, Julien Dupeyroux, and Guido C. H. E. de Croon. FAITH: Fast iterative half-plane focus of expansion estimation using optic flow. *IEEE Robot. Autom. Lett.*, 6(4):7627–7634, 2021.

[80] A Gómez Eguíluz, Juan Pablo Rodríguez-Gómez, R. Tapia, Francisco Javier Maldonado, José Ángel Acosta, J. R. Martínez-de Dios, and Aníbal Ollero. Why fly blind? Event-based visual guidance for ornithopter robot flight. In *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 1958–1965. IEEE, 2021.

[81] Juan Pablo Rodríguez-Gómez, Raul Tapia, Maria del Mar Guzmán Garcia, Jose Ramiro Martínez-de Dios, and Anibal Ollero. Free as a bird: Event-based dynamic sense-and-avoid for ornithopter robot flight. *IEEE Robot. Autom. Lett.*, 7(2):5413–5420, 2022.

[82] Francesco Galluppi, Christian Denk, Matthias C. Meiner, Terrence C. Stewart, Luis A. Plana, Chris Eliasmith, Steve Furber, and Jörg Conradt. Event-based neural computing on an autonomous mobile platform. In *IEEE Int. Conf. Robot. Autom.*, pages 2862–2867, 2014.

[83] Moritz B. Milde, Hermann Blum, Alexander Dietmüller, Dora Sumislawska, Jörg Conradt, Giacomo Indiveri, and Yulia Sandamirskaya. Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog / digital neuromorphic processing system. *Frontiers Neurorobot.*, 11, 2017.

[84] Ning Qiao, Hesham Mostafa, Federico Corradi, Marc Osswald, Fabio Stefanini, Dora Sumislawska, and Giacomo Indiveri. A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses. *Frontiers Neuroscience*, 9, 2015.

[85] Antonio Vitale, Alpha Renner, Celine Nauer, Davide Scaramuzza, and Yulia Sandamirskaya. Event-driven vision and control for UAVs on a neuromorphic chip. In *IEEE Int. Conf. Robot. Autom.*, pages 103–109, 2021.

[86] Rika Sugimoto Dimitrova, Mathias Gehrig, Dario Brescianini, and Davide Scaramuzza. Towards low-latency high-bandwidth control of quadrotors using event cameras. In *IEEE Int. Conf. Robot. Autom.*, pages 4294–4300. IEEE, 2020.

[87] Stein Stroobants, Julien Dupeyroux, and Guido C. H. E. de Croon. Design and implementation of a parsimonious neuromorphic PID for onboard altitude control for MAVs using neuromorphic processors. In *Int. Conf. Neuromorphic Syst.*, pages 1–7, 2022.

[88] Stein Stroobants, Julien Dupeyroux, and Guido C. H. E. de Croon. Neuromorphic computing for attitude estimation onboard quadrotors. *Neuromorphic Comput. Engineering*, 2(3), 2022.

[89] Stein Stroobants, Christophe De Wagter, and Guido C. H. E. de Croon. Neuromorphic control using input-weighted threshold adaptation. *arXiv:2304.08778*, 2023.

[90] Guido C. H. E. de Croon, Julien Dupeyroux, Christophe De Wagter, Abhishek Chatterjee, Diana A. Olejnik, and Franck Ruffier. Accommodating unobservability to control flight attitude with optic flow. *Nature*, 610(7932):485–490, 2022.

[91] Guido C. H. E.. de Croon, C. De Wagter, and T. Seidl. Enhancing optical-flow-based control by learning visual appearance cues for flying robots. *Nature Mach. Intell.*, 3(1):33–41, 2021.

[92] Hyungpil Moon, Jose Martinez-Carranza, Titus Cieslewski, Matthias Faessler, Davide Falanga, Alessandro Simovic, Davide Scaramuzza, Shuo Li, Michael Ozo, Christophe De Wagter, et al. Challenges and implemented technologies used in autonomous drone racing. *Intell. Service Robot.*, 12(2):137–148, 2019.

[93] Christophe De Wagter. *Hover and fast flight of minimum-mass mission-capable flying robots.* Ph.D. Thesis, Delft University of Technology, 2022.

[94] Philipp Foehn, Dario Brescianini, Elia Kaufmann, Titus Cieslewski, Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. AlphaPilot: Autonomous drone racing. In *Robot.: Science Syst.*, 2020.

[95] Zhe Jiang, Yu Zhang, Dongqing Zou, Jimmy Ren, Jiancheng Lv, and Yebin Liu. Learning event-based motion deblurring. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3320–3329, 2020.

[96] 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.*, pages 3867–3876, 2018.

[97] 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.*, pages 12280–12289, 2019.

[98] Christophe De Wagter, Federico Paredes-Vallés, Nilay Sheth, and Guido C. H. E. De Croon. Learning fast in autonomous drone racing. *Nature Mach. Intell.*, 3(10):923–923, 2021.

[99] Christophe De Wagter, Federico Paredes-Vallé, Nilay Sheth, and Guido C. H. E. de Croon. The sensing, state-estimation, and control behind the winning entry to the 2019 Artificial Intelligence Robotic Racing competition. *Field Robot.*, 2(1):1263–1290, March 2022.

[100] Federico Paredes-Vallés and Guido C. H. E. de Croon. Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021.

[101] Federico Paredes-Vallés, 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. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(8):2051–2064, 2020.

[102] Jesse J. Hagenaars, Federico Paredes-Vallés, and Guido C. H. E. de Croon. Self-supervised learning of event-based optical flow with spiking neural networks. In *Adv. Neural Inform. Process. Syst.*, volume 34, pages 7167–7179, 2021.

[103] Federico Paredes-Vallés, Kirk Y. W. Scheper, Christophe De Wagter, and Guido C. H. E. de Croon. Taming contrast maximization for learning sequential, low-latency, event-based optical flow. *Int. Conf. Comput. Vis.*, 2023.

[104] Federico Paredes-Vallés, Jesse J. Hagenaars, Julien Dupeyroux, Stein Stroobants, Yingfu Xu, and Guido C. H. E. de Croon. Fully neuromorphic vision and control for autonomous drone flight. *arXiv:2303.08778*, 2023.

[105] David B. de Jong, Federico Paredes-Vallés, and Guido C. H. E. de Croon. How do neural networks estimate optical flow? A neuropsychology-inspired study. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(11):8290–8305, 2021.

[106] Rik J. Bouwmeester, Federico Paredes-Vallés, and Guido C. H. E. de Croon. NanoFlowNet: Real-time dense optical flow on a nano quadcopter. *IEEE Int. Conf. Robot. Autom.*, 2023.

[107] Murray Campbell, A. Joseph Hoane Jr, and Feng-hsiung Hsu. Deep blue. *Artificial Intell.*, 134(1-2):57–83, 2002.

[108] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[109] Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

[110] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemys Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, Rafal Józefowicz, Scott Gray, Catherine Olsson, Jakub Pachocki, Michael Petrov, Henrique P. d. O. Pinto, Jonathan Raiman, Tim Salimans, Jeremy Schlatter, Jonas Schneider, Szymon Sidor, Ilya Sutskever, Jie Tang, Filip Wolski, and Susan Zhang. Dota 2 with large scale deep reinforcement learning. *arXiv:1912.06680*, 2019.

[111] Guang-Zhong Yang, Jim Bellingham, Pierre E. Dupont, Peer Fischer, Luciano Floridi, Robert Full, Neil Jacobstein, Vijay Kumar, Marcia McNutt, Robert Merrifield, et al. The grand challenges of Science Robotics. *Science Robot.*, 3(14), 2018.

[112] Nick Jakobi, Phil Husbands, and Inman Harvey. Noise and the reality gap: The use of simulation in evolutionary robotics. In *European Conf. Artificial Life*, pages 704–720, 1995.

[113] Sylvain Koos, Jean-Baptiste Mouret, and Stéphane Doncieux. The transferability approach: Crossing the reality gap in evolutionary robotics. *IEEE Trans. Evolutionary Comput.*, 17(1):122–145, 2012.

[114] Kirk Y. W. Scheper and Guido C. H. E. de Croon. Abstraction as a mechanism to cross the reality gap in evolutionary robotics. In *Int. Conf. Simulation Adaptive Behavior*, pages 280–292, 2016.

[115] Sergei Lupashin, Angela Schöllig, Michael Sherback, and Raffaello D'Andrea. A simple learning strategy for high-speed quadrocopter multi-flips. In *IEEE Int. Conf. Robot. Autom.*, pages 1642–1648, 2010.

[116] Daniel Mellinger, Nathan Michael, and Vijay Kumar. Trajectory generation and control for precise aggressive maneuvers with quadrotors. *Int. J. Robot. Research*, 31(5):664–674, 2012.

[117] Daniel Mellinger and Vijay Kumar. Minimum snap trajectory generation and control for quadrotors. In *IEEE Int. Conf. Robot. Autom.*, pages 2520–2525, 2011.

[118] Davide Falanga, Elias Mueggler, Matthias Faessler, and Davide Scaramuzza. Aggressive quadrotor flight through narrow gaps with onboard sensing and computing using active vision. In *IEEE Int. Conf. Robot. Autom.*, pages 5774–5781, 2017.

[119] Giuseppe Loianno, Chris Brunner, Gary McGrath, and Vijay Kumar. Estimation, control, and planning for aggressive flight with a small quadrotor with a single camera and imu. *IEEE Robot. Autom. Lett.*, 2(2):404–411, 2016.

[120] Hyungpil Moon, Yu Sun, Jacky Baltes, and Si Jung Kim. The IROS 2016 competitions. *IEEE Robot. Autom. Magazine*, 24(1):20–29, 2017.

[121] Benjamin Morrell, Marc Rigter, Gene Merewether, Robert Reid, Rohan Thakker, Theodore Tzanetos, Vinay Rajur, and Gregory Chamitoff. Differential flatness transformations for aggressive quadrotor flight. In *IEEE Int. Conf. Robot. Autom.*, pages 1–7, 2018.

[122] Elia Kaufmann, Mathias Gehrig, Philipp Foehn, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Beauty and the beast: Optimal methods meet learning for drone racing. In *IEEE Int. Conf. Robot. Autom.*, pages 690–696, 2019.

[123] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *IEEE Int. Conf. Robot. Autom.*, pages 2502–2509, 2018.

[124] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D. Tardos. ORB-SLAM: A versatile and accurate monocular SLAM system. *IEEE Trans. Robot.*, 31(5):1147–1163, 2015.

[125] Shuo Li, Michaël MOI Ozo, Christophe De Wagter, and Guido C. H. E. de Croon. Autonomous drone race: A computationally efficient vision-based navigation and control strategy. *Robot. Autonomous Syst.*, 133, 2020.

[126] Shuo Li, Erik van der Horst, Philipp Duernay, Christophe De Wagter, and Guido C. H. E. de Croon. Visual model-predictive localization for computationally efficient autonomous racing of a 72-g drone. *J. Field Robot.*, 2020.

[127] Sunggoo Jung, Sunyou Hwang, Heemin Shin, and David Hyunchul Shim. Perception, guidance, and navigation for indoor autonomous drone racing using deep learning. *IEEE Robot. Autom. Lett.*, 3(3):2539–2544, 2018.

[128] Jos Arturo Cocoma-Ortega and J. Martinez-Carranza. A CNN based drone localisation approach for autonomous drone racing. In *Int. Micro Air Vehicle Compet. Conf.*, 2019.

[129] John Aloimonos, Isaac Weiss, and Amit Bandyopadhyay. Active vision. *Int. J. Comput. Vis.*, 1(4):333–356, January 1988.

[130] Nitin J. Sanket, Chahat Deep Singh, Kanishka Ganguly, Cornelia Fermuller, and Yiannis Aloimonos. GapFlyt: Active vision based minimalist structure-less gap detection for quadrotor flight. *IEEE Robot. Autom. Lett.*, 3(4):2799–2806, 2018.

[131] Ruzena Bajcsy, Yiannis Aloimonos, and John K. Tsotsos. Revisiting active perception. *Autonomous Robots*, 42(2):177–196, 2018.

[132] Antonio Loquercio, Elia Kaufmann, René Ranftl, Alexey Dosovitskiy, Vladlen Koltun, and Davide Scaramuzza. Deep drone racing: From simulation to reality with domain randomization. *IEEE Trans. Robot.*, 36:1–14, 2019.

[133] Leticia Oyuki Rojas-Perez and Jose Martinez-Carranza. DeepPilot: A CNN for autonomous drone racing. *Sensors*, 20(16), 2020.

[134] Elia Kaufmann, Antonio Loquercio, Rene Ranftl, Matthias Müller, Vladlen Koltun, and Davide Scaramuzza. Deep drone acrobatics. In *Robot.: Science Syst.*, 2020.

[135] Eric N. Johnson and Suresh K. Kannan. Adaptive trajectory control for autonomous helicopters. *J. Guidance, Control, Dynamics*, 28(3):524–538, 2005.

[136] Varun Murali, Igor Spasojevic, Winter Guerra, and Sertac Karaman. Perception-aware trajectory generation for aggressive quadrotor flight using differential flatness. In *American Control Conf.*, pages 3936–3943, 2019.

[137] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Int. Conf. Medical Image Comput. Comput.-Assisted Intervention*, pages 234–241. Springer, 2015.

[138] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[139] Sebastian Thrun, Mike Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, et al. Stanley: The robot that won the DARPA Grand Challenge. *J. Field Robot.*, 23(9):661–692, 2006.

[140] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1):59–69, 1982.

[141] Karl J. Åström and Björn Wittenmark. *Adaptive control*. Courier Corporation, 2013.

[142] Nitin J. Sanket, Chahat Deep Singh, Cornelia Fermüller, and Yiannis Aloimonos. PRGFlow: Unified SWAP-aware deep global optical flow for aerial robot navigation. *Electronics Lett.*, 57(16):614–617, 2021.

[143] Guillem Torrente, Elia Kaufmann, Philipp Fohn, and Davide Scaramuzza. Data-driven MPC for quadrotors. *IEEE Robot. Autom. Lett.*, 6(2):3769–3776, 2021.

[144] Shuo Li, Ekin Öztürk, Christophe De Wagter, Guido CHE de Croon, and Dario Izzo. Aggressive online control of a quadrotor via deep network representations of optimality principles. In *IEEE Int. Conf. Robot. Autom.*, pages 6282–6287, 2020.

[145] Philipp Foehn, Angel Romero, and Davide Scaramuzza. Time-optimal planning for quadrotor waypoint flight. *Science Robot.*, 6(56), 2021.

[146] 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.*, pages 3857–3866, 2019.

[147] Henri Rebecq, René Ranftl, Vladlen Koltun, and Davide Scaramuzza. High speed and high dynamic range video with an event camera. *IEEE Trans. Pattern Anal. Mach. Intell.*, 2019.

[148] 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.*, pages 156–163, 2020.

[149] Henri Rebecq, Daniel Gehrig, and Davide Scaramuzza. ESIM: An open event camera simulator. In *Conf. Robot Learn.*, pages 969–982, 2018.

[150] 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 *European Conf. Comput. Vis.*, 2020.

[151] Guillermo Gallego, Christian Forster, Elias Mueggler, and Davide Scaramuzza. Event-based camera pose tracking using a generative event model. *arXiv:1510.01972*, 2015.

[152] Hanme Kim, Ankur Handa, Ryad Benosman, Sio-Hoi Ieng, and Andrew J. Davison. Simultaneous mosaicing and tracking with an event camera. *J. Solid-State Circ.*, 43:566–576, 2008.

[153] Matthew Cook, Luca Gugelmann, Florian Jug, Christoph Krautz, and Angelika Steger. Interacting maps for fast visual interpretation. In *Int. Joint Conf. Neural Netw.*, pages 770–776. IEEE, 2011.

[154] 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.*, pages 884–892, 2016.

[155] Christian Reinbacher, Gottfried Graber, and Thomas Pock. Real-time intensity-image reconstruction for event cameras using manifold regularisation. *Int. J. Comput. Vis.*, 126(12):1381–1393, 2018.

[156] Cedric Scheerlinck, Nick Barnes, and Robert Mahony. Continuous-time intensity estimation using event cameras. In *Asian Conf. Comput. Vis.*, pages 308–324, 2018.

[157] Lin Wang, Yo-Sung Ho, Kuk-Jin Yoon, et al. Event-based high dynamic range image and very high frame rate video generation using conditional generative adversarial networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 10081–10090, 2019.

[158] Stefano Pini, Guido Borghi, and Roberto Vezzani. Learn to see by events: Color frame synthesis from event and rgb cameras. In *Int. Joint Conf. Comput. Vis., Imaging Comput. Graphics Theory Appl.*, 2019.

[159] Jonghyun Choi, Kuk-Jin Yoon, et al. Learning to super resolve intensity images from events. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2768–2776, 2020.

[160] Lin Wang, Tae-Kyun Kim, and Kuk-Jin Yoon. Eventsr: From asynchronous events to image reconstruction, restoration, and super-resolution via end-to-end adversarial learning. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8315–8325, 2020.

[161] Anton Mitrokhin, Cornelia Fermüller, Chethan Parameshwara, and Yiannis Aloimonos. Event-based moving object detection and tracking. In *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2018.

[162] Pierre Charbonnier, Laure Blanc-Feraud, Gilles Aubert, and Michel Barlaud. Two deterministic half-quadratic regularization algorithms for computed imaging. In *IEEE Int. Conf. Image Process.*, volume 2, pages 168–172, 1994.

[163] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Int. Joint Conf. Artificial Intell.*, 1981.

[164] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Adv. Neural Inform. Process. Syst.*, pages 2017–2025, 2015.

[165] J. Yu Jason, Adam W. Harley, and Konstantinos G. Derpanis. Back to basics: Unsupervised learning of optical flow via brightness constancy and motion smoothness. In *European Conf. Comput. Vis.*, pages 3–10, 2016.

[166] Leonid I. Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: Nonlinear Phenomena*, 60(1-4):259–268, 1992.

[167] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016.

[168] Alex Z. Zhu, Dinesh Thakur, Tolga Özaslan, Bernd Pfrommer, Vijay Kumar, and Kostas Daniilidis. The multivehicle stereo event camera dataset: An event camera dataset for 3D perception. *IEEE Robot. Autom. Lett.*, 3(3):2032–2039, 2018.

[169] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. Delving deeper into convolutional networks for learning video representations. *Int. Conf. Learn. Representations*, 2015.

[170] Shi Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-Chun Woo. Convolutional LSTM network: A machine learning approach for precipitation nowcasting. In *Adv. Neural Inform. Process. Syst.*, pages 802–810, 2015.

[171] Jeffrey Delmerico, Titus Cieslewski, Henri Rebecq, Matthias Faessler, and Davide Scaramuzza. Are we ready for autonomous drone racing? The UZH-FPV drone racing dataset. In *IEEE Int. Conf. Robot. Autom.*, pages 6713–6719, 2019.

[172] 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. *Int. J. Robot. Research*, 36(2):142–149, 2017.

[173] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Int. Conf. Learn. Representations*, 2014.

[174] Christian Brandli, Raphael Berner, Minhao Yang, Shih-Chii Liu, and Tobi Delbruck. A 240×180 130 dB 3μs latency global shutter spatiotemporal vision sensor. *IEEE J. Solid-State Circuits*, 49(10):2333–2341, 2014.

[175] Garima Yadav, Saurabh Maheshwari, and Anjali Agarwal. Contrast limited adaptive histogram equalization based enhancement for real time video system. In *Int. Conf. Advances Comput., Commun. Informatics*, pages 2392–2397. IEEE, 2014.

[176] Zhou Wang, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli. Image quality assessment: From error visibility to structural similarity. *IEEE Trans. Image Process.*, 13(4):600–612, 2004.

[177] 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.*, pages 586–595, 2018.

[178] Etienne Perot, Pierre de Tournemire, Davide Nitti, Jonathan Masci, and Amos Sironi. Learning to detect objects with a 1 megapixel event camera. In *Adv. Neural Inform. Process. Syst.*, 2020.

[179] Alexander Borst and Moritz Helmstaedter. Common circuit design in fly and mammalian motion vision. *Nature Neuroscience*, 18(8):1067–1076, 2015.

[180] Alexander Borst, Juergen Haag, and Dierk F. Reiff. Fly motion vision. *Annual Review Neuroscience*, 33:49–70, 2010.

[181] M. V. Srinivasan, Shaowu Zhang, M. Lehrer, and T. Collett. Honeybee navigation en route to the goal: Visual flight control and odometry. *J. Experimental Biology*, 199(1):237–244, 1996.

[182] Christophe De Wagter, Sjoerd Tijmons, Bart D. W. Remes, and Guido C. H. E. de Croon. Autonomous flight of a 20-gram flapping wing MAV with a 4-gram onboard stereo vision system. In *IEEE Int. Conf. Robot. Autom.*, pages 4982–4987, 2014.

[183] Alfred Kirkwood and Mark F. Bear. Hebbian synapses in visual cortex. *J. Neuroscience*, 14(3):1634–1645, 1994.

[184] Larry C. Katz and Carla J. Shatz. Synaptic activity and the construction of cortical circuits. *Science*, 274(5290):1133–1138, 1996.

[185] Christoph Posch, Daniel Matolin, and Rainer Wohlgenannt. A QVGA 143 dB dynamic range frame-free PWM image sensor with lossless pixel-level video compression and time-domain CDS. *IEEE J. Solid-State Circuits*, 46(1):259–275, 2011.

[186] Christian Brandli, Raphael Berner, Marc Osswald, and N. Baumli. Silicon eye event sensor SEES1. *Insightness AG*, 2018.

[187] Ryad Benosman, Charles Clercq, Xavier Lagorce, Sio-Hoi Ieng, and Chiara Bartolozzi. Event-based visual flow. *IEEE Trans. Neural Netw. Learn. Syst.*, 25(2):407–417, 2013.

[188] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. Optical flow modeling and computation: A survey. *Comput. Vis. Image Understand.s*, 134:1–21, 2015.

[189] Eddy Ilg, Nikolaus Mayer, Tonmoy Saikia, Margret Keuper, Alexey Dosovitskiy, and Thomas Brox. FlowNet 2.0: Evolution of optical flow estimation with deep networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2462–2470, 2017.

[190] C. Ye, A. Mitrokhin, C. Fermüller, J. A. Yorke, and Y. Aloimonos. Unsupervised learning of dense optical flow, depth and egomotion with event-based sensors. In *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 5831–5838, 2020.

[191] Wolfgang Maass. Networks of spiking neurons: The third generation of neural network models. *Neural Netw.*, 10(9):1659–1671, 1997.

[192] Garrick Orchard and Ralph Etienne-Cummings. Bioinspired visual motion estimation. *IEEE Proc.*, 102(10):1520–1536, 2014.

[193] Richard B. Stein. A theoretical analysis of neuronal variability. *Biophysical J.*, 5(2):173–194, 1965.

[194] Alan L. Hodgkin and Andrew F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *J. Physiology*, 117(4):25–71, 1952.

[195] Eugene M. Izhikevich. Simple model of spiking neurons. *IEEE Trans. Neural Netw.*, 14(6):1569–1572, 2003.

[196] Werner M. Kistler, Wulfram Gerstner, and J. Leo van Hemmen. Reduction of the Hodgkin-Huxley equations to a single-variable threshold model. *Neural Comput.*, 9(5):1015–1045, 1997.

[197] Michel Baudry. Synaptic plasticity and learning and memory: 15 years of progress. *Neurobiology Learning Memory*, 70(1):113–118, 1998.

[198] Kenji Doya. What are the computations of the cerebellum, the basal ganglia and the cerebral cortex? *Neural Netw.*, 12(7-8):961–974, 1999.

[199] Wulfram Gerstner and Werner M. Kistler. *Spiking neuron models: Single neurons, populations, plasticity*. Cambridge University Press, 2002.

[200] T. Iakymchuk, A. Rosado-Muñoz, J. F. Guerrero-Martínez, M. Bataller-Mompeán, and J. V. Francés-Víllora. Simplified spiking neural network architecture and STDP learning algorithm applied to image classification. *J. Image Video Process.*, pages 1–11, 2015.

[201] Mark C. W. Van Rossum, Guo Q. Bi, and Gina G. Turrigiano. Stable Hebbian learning from spike timing-dependent plasticity. *J. Neuroscience*, 20(23):8812–8821, 2000.

[202] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Cognitive Modeling*, 5(3):533–536, 1988.

[203] Jun H. Lee, Tobi Delbruck, and Michael Pfeiffer. Training deep spiking neural networks using backpropagation. *Frontiers Neuroscience*, 10:1–13, 2016.

[204] Yujie Wu, Lei Deng, Guoqi Li, Jun Zhu, and Luping Shi. Spatio-temporal backpropagation for training high-performance spiking neural networks. *Frontiers Neuroscience*, 12, 2018.

[205] Aboozar Taherkhani, Ammar Belatreche, Yuhua Li, and Liam P. Maguire. A supervised learning algorithm for learning precise timing of multiple spikes in multilayer spiking neural networks. *IEEE Trans. Neural Netw. Learn. Syst.*, pages 1–14, 2018.

[206] Sumit B. Shrestha and Garrick Orchard. Slayer: Spike layer error reassignment in time. In *Adv. Neural Inform. Process. Syst.*, 2018.

[207] J. A. Pérez-Carrasco, B. Zhao, C. Serrano, B. Acha, T. Serrano-Gotarredona, S. Chen, and B. 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. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(11):2706–2719, 2013.

[208] Davide Zambrano, Roeland Nusselder, H Steven Scholte, and Sander M. Bohté. Efficient computation in adaptive artificial spiking neural networks. *arXiv:1710.04838*, 2017.

[209] 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. *Frontiers Neuroscience*, 11:1–12, 2017.

[210] Razvan V. Florian. Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity. *Neural Computation*, 19(6):1468–1502, 2007.

[211] Eugene M. Izhikevich. Solving the distal reward problem through linkage of STDP and dopamine signaling. *Cerebral Cortex*, 17(10):2443–2452, 2007.

[212] Jaldert O Rombouts, Pieter R. Roelfsema, and Sander M. Bohté. Neurally plausible reinforcement learning of working memory tasks. In *Adv. Neural Inform. Process. Syst.*, pages 1871–1879, 2012.

[213] Jaldert O Rombouts, Arjen van Ooyen, Pieter R. Roelfsema, and Sander M. Bohté. Biologically plausible multi-dimensional reinforcement learning in neural networks. In *Int. Conf. Artificial Neural Netw.*, pages 443–450, 2012.

[214] Johannes Friedrich and Máte Lengyel. Goal-directed decision making with spiking neurons. *J. Neuroscience*, 36(5):1529–1546, 2016.

[215] Zhenshan Bing, Claus Meschede, Kai Huang, Guang Chen, Florian Rohrbein, Mahmoud Akl, and Alois Knoll. End to end learning of spiking neural network based on R-STDP for a lane keeping vehicle. In *IEEE Int. Conf. Robot. Autom.*, pages 4725–4732, 2018.

[216] Milad Mozafari, Mohammad Ganjtabesh, Abbas Nowzari-Dalini, Simon J Thorpe, and Timothée Masquelier. Bio-inspired digit recognition using reward-modulated spike-timing-dependent plasticity in deep convolutional networks. *Pattern Recog.*, 94:87–95, 2019.

[217] Tobias Brosch, Stephan Tschechne, and Heiko Neumann. On event-based optical flow detection. *Frontiers Neuroscience*, 9:137, 2015.

[218] Myo Tun Aung, Rodney Teo, and Garrick Orchard. Event-based plane-fitting optical flow for dynamic vision sensors in FPGA. In *IEEE Int. Symp. Circuits Systems*, pages 1–5, 2018.

[219] Stephan Tschechne, Roman Sailer, and Heiko Neumann. Bio-inspired optic flow from event-based neuromorphic sensor input. In *Artificial Neural Netw. Pattern Recog. Worksh.*, pages 171–182, 2014.

[220] Francisco Barranco, Cornelia Fermuller, and Yiannis Aloimonos. Bio-inspired motion estimation with event-driven sensors. In *Int. Conf. Artificial Neural Netw.*, pages 309–321, 2015.

[221] Tobias Brosch and Heiko Neumann. Event-based optical flow on neuromorphic hardware. In *Int. Conf. Bio-inspired Inform. Commun. Tech.*, pages 551–558, 2016.

[222] Alex Zihao Zhu, Nikolay Atanasov, and Kostas Daniilidis. Event-based feature tracking with probabilistic data association. In *IEEE Int. Conf. Robot. Autom.*, pages 4465–4470, 2017.

[223] Guillermo Gallego and Davide Scaramuzza. Accurate angular velocity estimation with an event camera. *IEEE Robot. Autom. Lett.*, 2(2):632–639, 2017.

[224] Min Liu and Tobi Delbruck. ABMOF: A novel optical flow algorithm for dynamic vision sensors. *arXiv:1805.03988*, 2018.

[225] Xavier Lagorce, Sio-Hoi Ieng, Xavier Clady, Michael Pfeiffer, and Ryad B. Benosman. Spatiotemporal features for asynchronous event-based data. *Frontiers Neuroscience*, 9:1–13, 2015.

[226] Massimiliano Giulioni, Xavier Lagorce, Francesco Galluppi, and Ryad B. Benosman. Event-based computation of motion flow on a neuromorphic analog neural platform. *Frontiers Neuroscience*, 10:1–13, 2016.

[227] Germain Haessig, Andrew Cassidy, Rodrigo Alvarez, Ryad Benosman, and Garrick Orchard. Spiking optical flow for event-based sensors using IBM's TrueNorth neurosynaptic system. *IEEE Trans. Biomed. Circuits Syst.*, 12(4):860–870, 2018.

[228] Christoph Richter, Florian Röhrbein, and Jörg Conradt. Bio-inspired optic flow detection using neuromorphic hardware. *Bernstein Conf. Comput. Neuroscience*, 2014.

[229] Garrick Orchard, Ryad Benosman, Ralph Etienne-Cummings, and Nitish V. Thakor. A spiking neural network architecture for visual motion estimation. In *IEEE Biomedical Circuits Syst. Conf.*, pages 298–301, 2013.

[230] W. Reichardt. Autocorrelation, a principle for the evaluation of sensory information by the central nervous system. *Sensory Commun.*, pages 303–317, 1961.

[231] Edward H. Adelson and James R. Bergen. Spatiotemporal energy models for the perception of motion. *J. Optical Society America*, 2(2):284–299, 1985.

[232] Shimon Ullman. *The interpretation of visual motion.* MIT Press, 1979.

[233] Aaron P. Shon, Rajesh P. N. Rao, and Terrence J. Sejnowski. Motion detection and prediction through spike-timing dependent plasticity. *Netw.: Comput. Neural Syst.*, 15(3):179–198, 2004.

[234] Oliver G. Wenisch, Joachim Noll, and J. Leo Van Hemmen. Spontaneously emerging direction selectivity maps in visual cortex through STDP. *Biological Cybernetics*, 93(4):239–247, 2005.

[235] Samantha V. Adams and Christopher M. Harris. A computational model of innate directional selectivity refined by visual experience. *Scientific Reports*, 5:1–13, 2015.

[236] Hugh Christopher Longuet-Higgins and K. Prazdny. The interpretation of a moving retinal image. *Royal Society London. Series B. Biological Sciences*, 1980.

[237] Abigail Morrison, Ad Aertsen, and Markus Diesmann. Spike-timing-dependent plasticity in balanced random networks. *Neural Comput.*, 19(6):1437–1467, 2007.

[238] Michael Abercrombie, Clarence J. Hickman, and Minnie L. Johnson. *A dictionary of biology*. Routledge, 2017.

[239] Sander M. Bohté. Efficient spike-coding with multiplicative adaptation in a spike response model. In *Adv. Neural Inform. Process. Syst.*, pages 1835–1843, 2012.

[240] Simon J. Thorpe. Spike arrival times: A highly efficient coding scheme for neural networks. *Parallel Process. Neural Syst.*, pages 91–94, 1990.

[241] H. B. Barlow and W. R. Levick. The mechanism of directionally selective units in rabbit's retina. *J. Physiology*, 178(3):477–504, 1965.

[242] Kimberly McGuire, Guido C. H. E. de Croon, Christophe De Wagter, Bart D. W. Remes, Karl Tuyls, and Hilbert Kappen. Local histogram matching for efficient optical flow computation applied to velocity estimation on pocket drones. In *IEEE Int. Conf. Robot. Autom.*, pages 3255–3260, 2016.

[243] Federico Paredes-Vallés. *Neuromorphic Computing of Event-Based Data for High-Speed Vision-Based Navigation.* M.Sc. Thesis, Faculty of Aerospace Engineering, Delft University of Technology, 2018. [Online]. Available: TU Delft Education Repository.

[244] Mike Davies, Andreas Wild, Garrick Orchard, Yulia Sandamirskaya, Gabriel A. Fonseca Guerra, Prasad Joshi, Philipp Plank, and Sumedh R. Risbud. Advancing neuromorphic computing with Loihi: A survey of results and outlook. *IEEE Proc.*, pages 1–24, 2021.

[245] Nicolas Perez-Nieves, Vincent C. H. Leung, Pier Luigi Dragotti, and Dan F. M. Goodman. Neural heterogeneity promotes robust learning. *Nature Commun.*, 12(1):5791, 2021.

[246] Guillaume Bellec, Franz Scherr, Anand Subramoney, Elias Hajek, Darjan Salaj, Robert Legenstein, and Wolfgang Maass. A solution to the learning dilemma for recurrent networks of spiking neurons. *Nature Commun.*, 11(1):3625, 2020.

[247] Guillaume Bellec, Darjan Salaj, Anand Subramoney, Robert Legenstein, and Wolfgang Maass. Long short-term memory and learning-to-learn in networks of spiking neurons. In *Adv. Neural Inform. Process. Syst.*, volume 31, pages 787–797, 2018.

[248] Mohammed Almatrafi, Raymond Baldwin, Kiyoharu Aizawa, and Keigo Hirakawa. Distance surface for event-based optical flow. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(7):1547–1556, 2020.

[249] Zhuoyan Li, Jiawei Shen, and Ruitao Liu. A lightweight network to learn optical flow from event data. In *Int. Conf. Pattern Recog.*, pages 1–7, 2021.

[250] Chankyu Lee, Adarsh Kumar Kosta, Alex Z. Zhu, Kenneth Chaney, Kostas Daniilidis, and Kaushik Roy. Spike-FlowNet: Event-based optical flow estimation with energy-efficient hybrid neural networks. In *European Conf. Comput. Vis.*, pages 366–382, 2020.

[251] Chankyu Lee, Adarsh Kumar Kosta, and Kaushik Roy. Fusion-FlowNet: Energy-efficient optical flow estimation using sensor fusion and deep fused spiking-analog network architectures. *IEEE Int. Conf. Robot. Autom.*, 2022.

[252] Yannan Xing, Gaetano Di Caterina, and John Soraghan. A new spiking convolutional recurrent neural network (SCRNN) with applications to event-based hand gesture recognition. *Frontiers Neuroscience*, 14, 2020.

[253] Hanle Zheng, Yujie Wu, Lei Deng, Yifan Hu, and Guoqi Li. Going deeper with directly-trained larger spiking neural networks. *AAAI Conf. Artificial Intell.*, 35(12):11062–11070, 2021.

[254] Chethan M. Parameshwara, Simin Li, Cornelia Fermüller, Nitin J. Sanket, Matthew S. Evanusa, and Yiannis Aloimonos. SpikeMS: Deep spiking neural network for motion segmentation. *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2021.

[255] M. Gehrig, S. B. Shrestha, D. Mouritzen, and D. Scaramuzza. Event-based angular velocity regression with spiking networks. In *IEEE Int. Conf. Robot. Autom.*, pages 4195–4202, 2020.

[256] Timo Stoffregen and Lindsay Kleeman. Event cameras, contrast maximization and reward functions: An analysis. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12300–12308, 2019.

[257] Eimantas Ledinauskas, Julius Ruseckas, Alfonsas Juršėnas, and Giedrius Buračas. Training deep spiking neural networks. *arXiv:2006.04436*, 2020.

[258] Wei Fang, Zhaofei Yu, Yanqi Chen, Tiejun Huang, Timothée Masquelier, and Yonghong Tian. Deep residual learning in spiking neural networks. *Adv. Neural Inform. Process. Syst.*, 2021.

[259] Weihua He, YuJie Wu, Lei Deng, Guoqi Li, Haoyu Wang, Yang Tian, Wei Ding, Wenhui Wang, and Yuan Xie. Comparing SNNs and RNNs on neuromorphic vision datasets: Similarities and differences. *Neural Netw.*, 132:108–120, 2020.

[260] Bojian Yin, Federico Corradi, and Sander M. Bohté. Accurate and efficient time-domain classification with adaptive spiking recurrent neural networks. *Nature Mach. Intell.*, 3(10):905–913, 2021.

[261] Friedemann Zenke and Surya Ganguli. SuperSpike: Supervised learning in multilayer spiking neural networks. *Neural Comput.*, 30(6):1514–1541, 2018.

[262] Friedemann Zenke, Sander M. Bohté, Claudia Clopath, Iulia M. Comşa, Julian Göltz, Wolfgang Maass, Timothée Masquelier, Richard Naud, Emre O. Neftci, Mihai A. Petrovici, Franz Scherr, and Dan F. M. Goodman. Visualizing a joint future of neuroscience and neuromorphic engineering. *Neuron*, 109(4):571–575, 2021.

[263] Nicolas Perez-Nieves and Dan F. M. Goodman. Sparse spiking gradient descent. *Adv. Neural Inform. Process. Syst.*, 2021.

[264] Mark Horowitz. 1.1 Computing's energy problem (and what we can do about it). In *IEEE Int. Solid-State Circ. Conf. Digest Tech. Papers*, pages 10–14, 2014.

[265] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning optical flow with convolutional networks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2758–2766, 2015.

[266] 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.*, pages 8934–8943, 2018.

[267] Zachary Teed and Jia Deng. RAFT: Recurrent all-pairs field transforms for optical flow. In *Eur. Conf. Comput. Vis.*, pages 402–419. Springer, 2020.

[268] Mathias Gehrig, Manasi Muglikar, and Davide Scaramuzza. Dense continuous-time optical flow from events and frames. *arXiv:2203.13674*, 2022.

[269] Yilun Wu, Federico Paredes-Vallés, and Guido C. H. E. de Croon. Rethinking event-based optical flow: Iterative deblurring as an alternative to correlation volumes. *arXiv:2211.13726*, 2022.

[270] Ziluo Ding, Rui Zhao, Jiyuan Zhang, Tianxiao Gao, Ruiqin Xiong, Zhaofei Yu, and Tiejun Huang. Spatio-temporal recurrent networks for event-based optical flow estimation. In *AAAI Conf. Artificial Intell.*, volume 36, pages 525–533, 2022.

[271] Zhexiong Wan, Yuchao Dai, and Yuxin Mao. Learning dense and continuous optical flow from an event camera. *IEEE Trans. Image Process.*, 2022.

[272] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. Secrets of event-based optical flow. In *Eur. Conf. Comput. Vis.*, 2022.

[273] Vincent Brebion, Julien Moreau, and Franck Davoine. Real-time optical flow for vehicular perception with low-and high-resolution event cameras. *IEEE Trans. Intell. Transp. Syst.*, 2021.

[274] Qimin Wang, Yongjun Zhang, Shaowu Yang, Zhe Liu, Lin Wang, and Luoxi Jing. E-HANet: Event-based hybrid attention network for optical flow estimation. In *IEEE Int. Conf. Syst., Man, Cybernetics*, pages 2189–2194. IEEE, 2022.

[275] Wachirawit Ponghiran, Chamika Mihiranga Liyanagedera, and Kaushik Roy. Event-based temporally dense optical flow estimation with sequential neural networks. *arXiv:2210.01244*, 2022.

[276] Simon Meister, Junhwa Hur, and Stefan Roth. Unflow: Unsupervised learning of optical flow with a bidirectional census loss. In *AAAI Conf. Artificial Intell.*, volume 32, 2018.

[277] Mathias Gehrig, Willem Aarents, Daniel Gehrig, and Davide Scaramuzza. DSEC: A stereo event camera dataset for driving scenarios. *IEEE Robot. Autom. Lett.*, 6(3):4947–4954, 2021.

[278] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. A fast geometric regularizer to mitigate event collapse in the contrast maximization framework. *Adv. Intell. Syst.*, 2022.

[279] Haotian Liu, Guang Chen, Sanqing Qu, Yanping Zhang, Zhijun Li, Alois Knoll, and Changjun Jiang. TMA: Temporal motion aggregation for event-based optical flow. *Int. Conf. Comput. Vis.*, 2023.

[280] Javier Cuadrado, Ulysse Rançon, Benoît Cottereau, Francisco Barranco, and Timothée Masquelier. Optical flow estimation with event-based cameras and spiking neural networks. *arXiv:2302.06492*, 2023.

[281] Yijin Li, Zhaoyang Huang, Shuo Chen, Xiaoyu Shi, Hongsheng Li, Hujun Bao, Zhaopeng Cui, and Guofeng Zhang. Blinkflow: A dataset to push the limits of event-based optical flow estimation. *arXiv:2303.07716*, 2023.

[282] Shintaro Shiba, Yoshimitsu Aoki, and Guillermo Gallego. Event collapse in contrast maximization frameworks. *Sensors*, 22(14):5190, 2022.

[283] Himanshu Akolkar, Sio-Hoi Ieng, and Ryad Benosman. Real-time high speed motion prediction using fast aperture-robust event-driven visual flow. *IEEE Trans. Pattern Anal. Mach. Intell.*, 44(1):361–372, 2020.

[284] Dan Cireşan, Ueli Meier, Jonathan Masci, and Jürgen Schmidhuber. A committee of neural networks for traffic sign classification. In *Int. Joint Conf. Neural Netw.*, pages 1918–1921, 2011.

[285] Xuelian Cheng, Yiran Zhong, Mehrtash Harandi, Yuchao Dai, Xiaojun Chang, Hongdong Li, Tom Drummond, and Zongyuan Ge. Hierarchical neural architecture search for deep stereo matching. In *Adv. Neural Inform. Process. Syst.*, volume 33, pages 22158–22169, 2020.

[286] Xiaodong Gu, Zhiwen Fan, Siyu Zhu, Zuozhuo Dai, Feitong Tan, and Ping Tan. Cascade cost volume for high-resolution multi-view stereo and stereo matching. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2495–2504, 2020.

[287] Yuhui Yuan, Xiaokang Chen, Xilin Chen, and Jingdong Wang. Segmentation transformer: Object-contextual representations for semantic segmentation. *arXiv:1909.11065*, 2021.

[288] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer V2: Scaling up capacity and resolution. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12009–12019, 2022.

[289] Ross Girshick. Fast R-CNN. In *Int. Conf. Comput. Vis.*, pages 1440–1448, 2015.

[290] Joseph Redmon and Ali Farhadi. YOLOv3: An incremental improvement. *arXiv:1804.02767*, 2018.

[291] Mengde Xu, Zheng Zhang, Han Hu, Jianfeng Wang, Lijuan Wang, Fangyun Wei, Xiang Bai, and Zicheng Liu. End-to-end semi-supervised object detection with soft teacher. In *Int. Conf. Comput. Vis.*, pages 3060–3069, 2021.

[292] Ravi Garg, Vijay Kumar B. G., Gustavo Carneiro, and Ian Reid. Unsupervised CNN for single view depth estimation: Geometry to the rescue. In *European Conf. Comput. Vis.*, pages 740–756, 2016.

[293] Clement Godard, Oisin Mac Aodha, and Gabriel J. Brostow. Unsupervised monocular depth estimation with left-right consistency. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 270–279, 2017.

[294] Weihao Yuan, Xiaodong Gu, Zuozhuo Dai, Siyu Zhu, and Ping Tan. NeW CRFs: Neural window fully-connected CRFs for monocular depth estimation. *arXiv:2203.01502*, 2022.

[295] Giacomo Indiveri and Rodney Douglas. Neuromorphic vision sensors. *Science*, 288(5469):1189–1190, 2000.

[296] Yulia Sandamirskaya, Mohsen Kaboli, Jorg Conradt, and Tansu Celikel. Neuromorphic computing hardware and neural architectures for robotics. *Science Robot.*, 7, 2022.

[297] Andre Grüning and Sander M. Bohté. Spiking neural networks: Principles and challenges. In *European Symp. Artificial Neural Netw.*, 2014.

[298] M. Davies, A. Wild, G. Orchard, Y. Sandamirskaya, G. A. F. Guerra, P. Joshi, P. Plank, and S. R. Risbud. Advancing neuromorphic computing with Loihi: A survey of results and outlook. *IEEE Proc.*, 2021.

[299] Fabrizio Ottati, Chang Gao, Qinyu Chen, Giovanni Brignone, Mario R. Casu, Jason K. Eshraghian, and Luciano Lavagno. To spike or not to spike: A digital hardware perspective on deep learning acceleration. *arXiv:2306.15749*, 2023.

[300] Peter Sterling and Simon Laughlin. *Principles of neural design.* MIT Press, 2015.

[301] Florian T. Muijres, Michael J. Elzinga, Johan M. Melis, and Michael H. Dickinson. Flies evade looming targets by executing rapid visually directed banked turns. *Science*, 344(6180):172–177, 2014.

[302] Michael Pfeiffer and Thomas Pfeil. Deep learning with spiking neurons: Opportunities and challenges. *Frontiers Neuroscience*, 12, 2018.

[303] Amirhossein Tavanaei, Masoud Ghodrati, Saeed Reza Kheradpisheh, Timothée Masquelier, and Anthony Maida. Deep learning in spiking neural networks. *Neural Netw.*, 111:47–63, 2019.

[304] Sayeed Shafayet Chowdhury, Chankyu Lee, and Kaushik Roy. Towards understanding the effect of leak in spiking neural networks. *Neurocomput.*, 464:83–94, 2021.

[305] Dario Floreano and Claudio Mattiussi. Evolution of spiking neural controllers for autonomous vision-based robots. In *Evolutionary Robot.: From Intelligent Robot. to Artificial Life*, pages 38–61, 2001.

[306] D. H. Ballard. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recog.*, 13(2):111–122, 1981.

[307] Julien Dupeyroux, Jesse J. Hagenaars, Federico Paredes-Vallés, and Guido C. H. E. de Croon. Neuromorphic control for optic-flow-based landings of MAVs using the loihi processor. In *IEEE Int. Conf. Robot. Autom.*, 2021.

[308] Jesse J. Hagenaars, Federico Paredes-Vallés, Sander M. Bohté, and Guido C. H. E. de Croon. Evolved neuromorphic control for high speed divergence-based landings of MAVs. In *IEEE Int. Conf. Robot. Autom.*, pages 6239–6246, 2020.

[309] Simon Baker, Ankur Datta, and Takeo Kanade. Parameterizing homographies. Tech. Rep., Robot. Institute, 2006.

[310] Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Deep image homography estimation. *arXiv:1606.03798*, 2016.

[311] Ty Nguyen, Steven W. Chen, Shreyas S. Shivakumar, Camillo Jose Taylor, and Vijay Kumar. Unsupervised deep homography: A fast and robust homography estimation model. *IEEE Robot. Autom. Lett.*, 3(3):2346–2353, 2018.

[312] Yi Ma, Stefano Soatto, Jana Košecká, and S. Shankar Sastry. *An invitation to 3-D vision: From images to geometric models*. Springer, 2004.

[313] Takehiro Ozawa, Yusuke Sekikawa, and Hideo Saito. Accuracy and speed improvement of event camera motion estimation using a bird's-eye view transformation. *Sensors*, 22(3):773, 2022.

[314] Juan L. Valerdi, Chiara Bartolozzi, and Arren Glover. Insights into batch selection for event-camera motion estimation. *Sensors*, 23, 2023.

[315] Yunlong Song, Selim Naji, Elia Kaufmann, Antonio Loquercio, and Davide Scaramuzza. Flightmare: A flexible quadrotor simulator. In *Conf. Robot Learn.*, 2020.

[316] Shangkun Zhong and Pakpong Chirarattananon. Direct visual-inertial ego-motion estimation via iterated extended kalman filter. *IEEE Robot. Autom. Lett.*, 5(2):1476–1483, 2020.

[317] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 23–30, 2017.

[318] Behzad Boroujerdian, Hasan Genc, Srivatsan Krishnan, Wenzhi Cui, Aleksandra Faust, and Vijay Reddi. MAVBench: Micro aerial vehicle benchmarking. In *IEEE/ACM Int. Symp. Microarch.*, pages 894–907, 2018.

[319] Charlotte Frenkel, David Bol, and Giacomo Indiveri. Bottom-up and top-down approaches for the design of neuromorphic processing systems: Tradeoffs and synergies between natural and artificial intelligence. *IEEE Proc.*, 111:623–652, 2023.

[320] Alex Vigneron and Jean Martinet. A critical survey of STDP in spiking neural networks for pattern recognition. In *Int. Joint Conf. Neural Netw.*, pages 1–9. IEEE, 2020.

[321] Kenneth Chaney, Artemis Panagopoulou, Chankyu Lee, Kaushik Roy, and Kostas Daniilidis. Self-supervised optical flow with spiking neural networks and event based cameras. In *IEEE/RSJ Int. Conf. Intell. Robots Syst.*, pages 5892–5899. IEEE, 2021.

[322] Thomas Barbier, Céline Teulière, and Jochen Triesch. Spike timing-based unsupervised learning of orientation, disparity, and motion representations in a spiking neural network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1377–1386, 2021.

[323] Qianhui Liu, Dong Xing, Huajin Tang, De Ma, and Gang Pan. Event-based action recognition using motion information and spiking neural networks. In *Int. Joint Conf. Artificial Intell.*, pages 1743–1749, 2021.

[324] Xueyuan She and Saibal Mukhopadhyay. Speed: Spiking neural network with event-driven unsupervised learning and near-real-time inference for event-based vision. *IEEE Sensors J.*, 21(18):20578–20588, 2021.

[325] Francesca Peveri, Simone Testa, and Silvio P Sabatini. A cortically-inspired architecture for event-based visual motion processing: From design principles to real-world applications. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1395–1402, 2021.

[326] Kaiwei Che, Luziwei Leng, Kaixuan Zhang, Jianguo Zhang, Qinghu Meng, Jie Cheng, Qinghai Guo, and Jianxing Liao. Differentiable hierarchical and surrogate gradient search for spiking neural networks. *Adv. Neural Inform. Process. Syst.*, 35:24975–24990, 2022.

[327] Alfio Di Mauro, Moritz Scherer, Davide Rossi, and Luca Benini. Kraken: A direct event/frame-based multi-sensor fusion SoC for ultra-efficient visual processing in nano-UAVs. *arXiv:2209.01065*, 2022.

[328] Qiang Yu, Jialu Gao, Jianguo Wei, Jing Li, Kay Chen Tan, and Tiejun Huang. Improving multispike learning with plastic synaptic delays. *IEEE Trans. Neural Netw. Learn. Syst.*, 2022.

[329] Adarsh Kumar Kosta and Kaushik Roy. Adaptive-spikenet: Event-based optical flow estimation using spiking neural networks with learnable neuronal dynamics. *arXiv:2209.11741*, 2022.

[330] Benjamin Chamand and Philippe Joly. Self-supervised spiking neural networks applied to digit classification. In *Int. Conf. Content-based Multimedia Indexing*, pages 196–200, 2022.

[331] Shihao Zou, Yuxuan Mu, Xinxin Zuo, Sen Wang, and Li Cheng. Event-based human pose tracking by spiking spatiotemporal transformer. *arXiv:2303.09681*, 2023.

[332] Shubham Negi, Deepika Sharma, Adarsh Kumar Kosta, and Kaushik Roy. Best of both worlds: Hybrid SNN-ANN architecture for event-based optical flow estimation. *arXiv:2306.02960*, 2023.

[333] Xabier Iturbe, Nassim Abderrahmane, Jaume Abella, Sergi Alcaide, Eric Beyne, Henri-Pierre Charles, Christelle Charpin-Nicolle, Lars Chittka, Angélica Dávila, Arne Erdmann, et al. NimbleAI: Towards neuromorphic sensing-processing 3D-integrated chips. In *Europe Conf. Exhibition*, pages 1–6. IEEE, 2023.

[334] Bojian Yin, Federico Corradi, and Sander M. Bohté. Accurate online training of dynamical spiking neural networks through forward propagation through time. *Nature Mach. Intell.*, pages 1–10, 2023.

[335] Geoffrey Hinton. The forward-forward algorithm: Some preliminary investigations. *arXiv:2212.13345*, 2022.

[336] Jianfeng Feng. *Computational neuroscience: A comprehensive approach*. Chapman and Hall/CRC, 2003.

[337] Ibrahim Kajo, Aamir Saeed Malik, and Nidal Kamel. An evaluation of optical flow algorithms for crowd analytics in surveillance system. In *Int. Conf. Intell. Advanced Syst.*, 2017.

[338] Steven S. Beauchemin and John L. Barron. The computation of optical flow. *ACM Comput. Surveys*, 27(3):433–466, 1995.

[339] P. Anandan. A computational framework and an algorithm for the measurement of visual motion. *Int. J. Comput. Vis.*, 2(3):283–310, 1989.

[340] A. Singh. *Optic flow computation: A unified perspective*. Los Alamitos: IEEE Computer Society Press, 1991.

[341] David J. Heeger. Optical flow using spatiotemporal filters. *Int. J. Comput. Vis.*, pages 279–302, 1988.

[342] David J. Fleet and Allan D. Jepson. Computation of normal velocity from local phase information. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 379–386, 1989.

[343] Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intell.*, 17(1-3):185–203, 1981.

[344] Dennis Gabor. Theory of communication. *J. Institution Electrical Engineers - Part I: General*, 94(73):58–58, 1945.

[345] Henning Zimmer, Andrés Bruhn, and Joachim Weickert. Optic flow in harmony. *Int. J. Comput. Vis.*, 93(3):368–388, 2011.

[346] Zhigang Tu, Wei Xie, Dejun Zhang, Ronald Poppe, Remco C. Veltkamp, Baoxin Li, and Junsong Yuan. A survey of variational and CNN-based optical flow techniques. *Signal Process.: Image Commun.*, 72:9–24, 2019.

[347] Tak-Wai Hui, Xiaoou Tang, and Chen Change Loy. A lightweight optical flow CNN — Revisiting data fidelity and regularization. *IEEE Trans. Pattern Anal. Mach. Intell.*, 43(8):2555–2569, 2020.

[348] Judson P. Jones, Aaron Stepnoski, and Larry A. Palmer. The two-dimensional spectral structure of simple receptive fields in cat striate cortex. *J. Neurophysiology*, 58(6):1212–1232, 1987.

[349] Duane G. Albrecht, Russell L. De Valois, and Lisa G. Thorell. Visual cortical neurons: Are bars or gratings the optimal stimuli? *Science*, 207:88–90, 1980.

[350] Judson P. Jones and Larry A. Palmer. An evaluation of the two-dimensional Gabor filter model of simple receptive fields in cat striate cortex. *J. Neurophysiology*, 58(6):1233–1258, 1987.

[351] Gregory C. DeAngelis, Izumi Ohzawa, and Ralph D. Freeman. Receptive-field dynamics in the central visual pathways. *Trends Neurosciences*, 18(10):451–458, 1995.

[352] J. Hans Van Hateren and Dan L. Ruderman. Independent component analysis of natural image sequences yields spatio-temporal filters similar to simple cells in primary visual cortex. In *Royal Society London. Series B: Biological Sciences*, volume 265, pages 2315–2320, 1998.

[353] Bruno A. Olshausen. Learning sparse, overcomplete representations of time-varying natural images. In *IEEE Int. Conf. Image Process.*, volume 1, pages 41–44, 2003.

[354] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European Conf. Comput. Vis.*, pages 25–36, 2004.

[355] Thomas Brox and Jitendra Malik. Large displacement optical flow: Descriptor matching in variational motion estimation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(3):500–513, 2011.

[356] Jerome Revaud, Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. EpicFlow: Edge-preserving interpolation of correspondences for optical flow. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1164–1172, 2015.

[357] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. DeepFlow: Large displacement optical flow with deep matching. In *Int. Conf. Comput. Vis.*, pages 1385–1392, 2013.

[358] Shay Zweig and Lior Wolf. InterpoNet, a brain inspired neural network for optical flow dense interpolation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6363–6372, 2017.

[359] Anurag Ranjan and Michael J. Black. Optical flow estimation using a spatial pyramid network. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 2720–2729, 2017.

[360] Damien Teney and Martial Hebert. Learning to extract motion from videos in convolutional neural networks. In *Asian Conf. Comput. Vis.*, pages 412–428, 2016.

[361] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Adv. Neural Inform. Process. Syst.*, pages 5574–5584, 2017.

[362] Eddy Ilg, Cicek Ozgun, Silvio Galesso, Aaron Klein, Osama Makansi, Frank Hutter, and Thomas Brox. Uncertainty estimates and multi-hypotheses networks for optical flow. In *European Conf. Comput. Vis.*, pages 652–667, 2018.

[363] Anurag Ranjan, Joel Janai, Andreas Geiger, and Michael J. Black. Attacking optical flow. In *Int. Conf. Comput. Vis.*, pages 2404–2413, 2019.

[364] Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *arXiv:1412.6806*, 2014.

[365] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *European Conf. Comput. Vis.*, pages 818–833, 2014.

[366] Dumitru Erhan, Yoshua Bengio, Aaron Courville, and Pascal Vincent. Visualizing higher-layer features of a deep network. *University Montreal*, 1341(3):1, 2009.

[367] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017.

[368] Alexander Mordvintsev. Inceptionism: Going deeper into neural networks. *[Online] Available: https://ai.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html*, 2015.

[369] Anh Nguyen, Alexey Dosovitskiy, Jason Yosinski, Thomas Brox, and Jeff Clune. Synthesizing the preferred inputs for neurons in neural networks via deep generator networks. In *Adv. Neural Inform. Process. Syst.*, pages 3395–3403, 2016.

[370] Donglai Wei, Bolei Zhou, Antonio Torrabla, and William Freeman. Understanding intra-class knowledge inside CNN. *arXiv:1507.02379*, 2015.

[371] Ronald Newbold Bracewell and Ronald N. Bracewell. *The Fourier transform and its applications*. McGraw-Hill New York, 1986.

[372] Gregory C. Deangelis, Izumi Ohzawa, and R. D. Freeman. Spatiotemporal organization of simple-cell receptive fields in the cat's striate cortex. I. General characteristics and postnatal development. *J. Neurophysiology*, 69(4), 1993.

[373] Nicolai Petkov and Easwar Subramanian. Motion detection, noise reduction, texture suppression, and contour enhancement by spatiotemporal Gabor filters with surround inhibition. *Biological Cybernetics*, 97(5-6):423–439, 2007.

[374] Larry A. Palmer and Thomas L. David. Receptive-field structure in cat striate cortex. *J. Neurophysiology*, 46(2):260–276, 1981.

[375] Gregory C. DeAngelis, Izumi Ohzawa, and R. D. Freeman. Spatiotemporal organization of simple-cell receptive fields in the cat's striate cortex. II. Linearity of temporal and spatial summation. *J. Neurophysiology*, 69(4):1118–1135, 1993.

[376] Judson P. Jones and Larry A. Palmer. The two-dimensional spatial structure of simple receptive fields in cat striate cortex. *J. Neurophysiology*, 58(6):1187–1211, 1987.

[377] Hidehiko Komatsu. The neural mechanisms of perceptual filling-in. *Nature Reviews Neuroscience*, 7(3):220–231, 2006.

[378] Rüdiger Von Der Heydt, Howard S. Friedman, and Hong Zhou. Filling-in: From perceptual completion to cortical reorganization. *Oxford University Press*, pages 106–127, 2003.

[379] Jasper Poort, Florian Raudies, Aurel Wannig, Victor A.F. Lamme, Heiko Neumann, and Pieter R. Roelfsema. The role of attention in figure-ground segregation in areas V1 and V4 of the visual cortex. *Neuron*, 75(1):143–156, 2012.

[380] Ming Liang and Xiaolin Hu. Recurrent convolutional neural network for object recognition. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3367–3375, 2015.

[381] Daniel J. Butler, Jonas Wulff, Garrett B. Stanley, and Michael J. Black. A naturalistic open source movie for optical flow evaluation. In *European Conf. Comput. Vis.*, pages 611–625, 2012.

[382] Jonathan Olin Vahram Touryan. *Nonlinear analysis of complex cells in primary visual cortex*. University of California, Berkeley, 2004.

[383] Ya Xiang Yuan. A review of trust region algorithms for optimization. In *Int. Congress Industrial Applied Mathematics*, volume 99, pages 271–282, 2000.

[384] Steven S. Beauchemin and John L. Barron. The frequency structure of one-dimensional occluding image signals. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(2):200–206, 2000.

[385] Simon Baker, Daniel Scharstein, JP Lewis, Stefan Roth, Michael J. Black, and Richard Szeliski. A database and evaluation methodology for optical flow. *Int. J. Comput. Vis.*, 92(1):1–31, 2011.

[386] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, Thomas Brox, Eddy Ilg, Caner Hazirbas, Alexey Dosovitskiy, and Thomas Brox. What makes good synthetic training data for learning disparity and optical flow estimation? *Int. J. of Comput. Vis.*, 126:942–960, 2018.

[387] Michal Neoral, Jan Šochman, and Jiří Matas. Continual occlusion and optical flow estimation. In *Asian Conf. Comput. Vis.*, pages 159–174, 2018.

[388] Pengpeng Liu, Michael Lyu, Irwin King, and Jia Xu. Selflow: Self-supervised learning of optical flow. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4571–4580, 2019.

[389] Shuosen Guan, Haoxin Li, and Wei-Shi Zheng. Unsupervised learning for optical flow estimation using pyramid convolution LSTM. In *IEEE Int. Conf. Multimedia Expo*, pages 181–186, 2019.

[390] Pierre Godet, Alexandre Boulch, Aurélien Plyer, and Guy Le Besnerais. STaRFlow: A spatiotemporal recurrent cell for lightweight multi-frame optical flow estimation. *arXiv:2007.05481*, 2020.

[391] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G. Lowe. Unsupervised learning of depth and ego-motion from video. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1851–1858, 2017.

[392] Yuhua Chen, Cordelia Schmid, and Cristian Sminchisescu. Self-supervised learning with geometric constraints in monocular video: Connecting flow, depth, and camera. In *Int. Conf. Comput. Vis.*, pages 7063–7072, 2019.

[393] Russell L. De Valois, Duane G. Albrecht, and Lisa G. Thorell. Spatial frequency selectivity of cells in macaque visual cortex. *Vision Research*, 22(5):545–559, 1982.

[394] Russell L. De Valois, E. William Yund, and Norva Hepler. The orientation and direction selectivity of cells in macaque visual cortex. *Vision Research*, 22(5):531–544, 1982.

[395] D. Regan and K. I. Beverley. Looming detectors in the human visual pathway. *Vision Research*, 18(4):415–421, 1978.

[396] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4040–4048, 2016.

[397] Bruno Bodin, Harry Wagstaff, Sajad Saecdi, Luigi Nardi, Emanuele Vespa, John Mawer, Andy Nisbet, Mikel Lujan, Steve Furber, Andrew J. Davison, Paul H.J. Kelly, and Michael F.P. O'Boyle. SLAMBench2: Multi-objective head-to-head benchmarking for visual SLAM. In *IEEE Int. Conf. Robot. Autom.*, pages 3637–3644, 2018.

[398] Kimberly N. McGuire, Christophe de Wagter, Karl Tuyls, Hilbert J. Kappen, and Guido C. H. E. de Croon. Minimal navigation solution for a swarm of tiny flying robots to explore an unknown environment. *Science Robot.*, 4(35), 2019.

[399] Bardienus P. Duisterhof, Shushuai Li, Javier Burgues, Vijay Janapa Reddi, and Guido C. H. E. de Croon. Sniffy Bug: A fully autonomous swarm of gas-seeking nano quadcopters in cluttered environments. In *IEEE Int. Conf. Intell. Robots Syst.*, pages 9099–9106, 2021.

[400] Pinghai Gao, Daibing Zhang, Qiang Fang, and Shaogang Jin. Obstacle avoidance for micro quadrotor based on optical flow. In *Chinese Control Decision Conf.*, pages 4033–4037, 2017.

[401] Joseph Conroy, Gregory Gremillion, Badri Ranganathan, and J. Sean Humbert. Implementation of wide-field integration of optic flow for autonomous quadrotor navigation. In *Autonomous Robots*, volume 27, pages 189–198, 2009.

[402] Simon Zingg, Davide Scaramuzza, Stephan Weiss, and Roland Siegwart. MAV navigation through indoor corridors using optical flow. In *IEEE Int. Conf. Robot. Autom.*, pages 3361–3368, 2010.

[403] Julien R. Serres and Franck Ruffier. Optic flow-based collision-free strategies: From insects to robots. *Arthropod Structure Develop.*, 46(5):703–717, 2017.

[404] Shanshan Zhao, Xi Li, and Omar El Farouk Bourahla. Deep optical flow estimation via multi-scale correspondence structure learning. In *Int. Joint Conf. Artificial Intell.*, pages 3490–3496, 2017.

[405] Tak Wai Hui, Xiaoou Tang, and Chen Change Loy. Liteflownet: A lightweight convolutional neural network for optical flow estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8981–8989, 2018.

[406] Zhichao Yin, Trevor Darrell, and Fisher Yu. Hierarchical discrete distribution decomposition for match density estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6037–6046, 2019.

[407] Gengshan Yang and Deva Ramanan. Volumetric correspondence networks for optical flow. In *Adv. Neural Inform. Process. Syst.*, volume 32, 2019.

[408] Tak Wai Hui and Chen Change Loy. LiteFlowNet3: Resolving correspondence ambiguity for more accurate optical flow estimation. In *European Conf. Comput. Vis.*, pages 169–184, 2020.

[409] Shengyu Zhao, Yilun Sheng, Yue Dong, Eric I.Chao Chang, and Yan Xu. Maskflownet: Asymmetric feature matching with learnable occlusion mask. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6277–6286, 2020.

[410] Junhwa Hur and Stefan Roth. Iterative residual refinement for joint optical flow and occlusion estimation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5747–5756, 2019.

[411] Mingyuan Fan, Shenqi Lai, Junshi Huang, Xiaoming Wei, Zhenhua Chai, Junfeng Luo, and Xiaolin Wei. Rethinking BiSeNet for real-time semantic segmentation. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9711–9720, 2021.

[412] Nathan O. Lambert, Daniel S. Drew, Joseph Yaconelli, Sergey Levine, Roberto Calandra, and Kristofer S. J. Pister. Low-level control of a quadrotor with deep model-based reinforcement learning. *IEEE Robot. Autom. Lett.*, 4(4):4224–4230, 2019.

[413] Bardienus P. Duisterhof, Srivatsan Krishnan, Jonathan J. Cruz, Colby R. Banbury, William Fu, Aleksandra Faust, Guido C. H. E. de Croon, and Vijay Janapa Reddi. Tiny robot learning for source seeking on a nano quadcopter. In *IEEE Int. Conf. Robot. Autom.*, pages 7242–7248, 2021.

[414] Adrien Briod, Jean-Christophe Zufferey, and Dario Floreano. Optic-flow based control of a 46g quadrotor. In *IEEE/RSJ Int. Conf. Intell. Robots Syst. Worksh.*, 2013.

[415] Richard J. D. Moore, Karthik Dantu, Geoffrey L. Barrows, and Radhika Nagpal. Autonomous MAV guidance with a lightweight omnidirectional vision sensor. In *IEEE Int. Conf. Robot. Autom.*, pages 3856–3861, 2014.

[416] Kimberly McGuire, Guido C. H. E. de Croon, Christophe De Wagter, Karl Tuyls, and Hilbert Kappen. Efficient optical flow and stereo vision for velocity estimation and obstacle avoidance on an autonomous pocket drone. In *IEEE Robot. Autom. Lett.*, pages 1070–1076, 2017.

[417] Oliver Dunkley, Jakob J. Engel, Jürgen Sturm, and D. Cremers. Visual-inertial navigation for a camera-equipped 25g nano-quadrotor. In *IEEE/RSJ Int. Conf. Intell. Robots Syst. Worksh.*, page 2, 2014.

[418] Fethi Candan, Aykut Beke, and Tufan Kumbasar. Design and deployment of fuzzy PID controllers to the nano quadcopter Crazyflie 2.0. In *IEEE Int. Conf. Innovations Intell. Systems Applications*, 2018.

[419] Aqeel Anwar and Arijit Raychowdhury. Autonomous navigation via deep reinforcement learning for resource constraint edge nodes using transfer learning. *IEEE Access*, 8:26549–26560, 2020.

[420] Amr Suleiman, Zhengdong Zhang, Luca Carlone, Sertac Karaman, and Vivienne Sze. Navion: A 2-mW fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones. *IEEE J. Solid-State Circuits*, 54(4):1106–1119, 2019.

[421] Ziyun Li, Yu Chen, Luyao Gong, Lu Liu, Dennis Sylvester, David Blaauw, and Hun Seok Kim. An 879GOPS 243mW 80FPS VGA fully visual CNN-SLAM processor for wide-range autonomous exploration. *IEEE J. Solid-State Circuits*, pages 134–136, 2019.

[422] Morteza Hosseini and Tinoosh Mohsenin. Binary precision neural network manycore accelerator. *ACM J. Emerging Tech. Comput. Systems*, 17(2):1–27, 2021.

[423] Nitheesh Kumar Manjunath, Aidin Shiri, Morteza Hosseini, Bharat Prakash, Nicholas R. Waytowich, and Tinoosh Mohsenin. An energy efficient EdgeAI autoencoder accelerator for reinforcement learning. *IEEE Open J. Circuits Systems*, 2:182–195, 2021.

[424] Daniele Palossi, Francesco Conti, and Luca Benini. An open source and open hardware deep learning-powered visual navigation engine for autonomous nano-UAVs. In *Int. Conf. Distributed Comput. Sensor Syst.*, pages 604–611, 2019.

[425] Daniele Palossi, Antonio Loquercio, Francesco Conti, Eric Flamand, Davide Scaramuzza, and Luca Benini. A 64-mW DNN-based visual navigation engine for autonomous nano-drones. *IEEE Internet Things J.*, 6(5):8357–8371, 2019.

[426] Daniele Palossi, Nicky Zimmerman, Alessio Burrello, Francesco Conti, Hanna Muller, Luca Maria Gambardella, Luca Benini, Alessandro Giusti, and Jerome Guzzi. Fully onboard AI-powered human-drone pose estimation on ultralow-power autonomous flying nano-UAVs. *IEEE Internet of Things J.*, 9(3):1913–1929, 2022.

[427] Changqian Yu, Jingbo Wang, Chao Peng, Changxin Gao, Gang Yu, and Nong Sang. BiSeNet: Bilateral segmentation network for real-time semantic segmentation. In *European Conf. Comput. Vis.*, pages 325–341, 2018.

[428] Mingliang Zhai, Xuezhi Xiang, Rongfang Zhang, Ning Lv, and Abdulmotaleb El Saddik. AD-Net: Attention guided network for optical flow estimation using dilated convolution. In *IEEE Int. Conf. Acoustics, Speech Signal Process.*, pages 2207–2211, 2019.

[429] Guoyu Zuo, Chengwei Zhang, Jiayuan Tong, Daoxiong Gong, and Mengqian You. Edge detection-based optical flow estimation method. In *IEEE Int. Conf. Cyber Tech. Autom., Control, Intell. Syst.*, pages 873–878, 2021.

[430] Tsung Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(2):318–327, 2020.

[431] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1800–1807, 2017.

[432] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. MobileNets: Efficient convolutional neural networks for mobile vision applications. *arXiv:1704.04861*, 2017.

[433] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. ShuffleNet: An extremely efficient convolutional neural network for mobile devices. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 6848–6856, 2018.

[434] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang Chieh Chen. MobileNetV2: Inverted residuals and linear bottlenecks. In *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 4510–4520, 2018.

[435] Valentin Bazarevsky, Yury Kartynnik, Andrey Vakunov, Karthik Raveendran, and Matthias Grundmann. BlazeFace: Sub-millisecond neural face detection on mobile GPUs. *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, 2019.

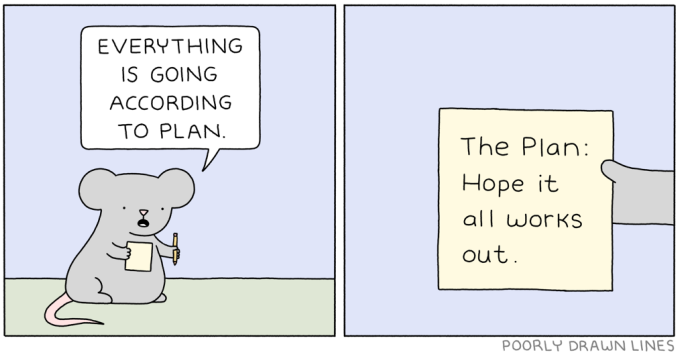
[436] Eddy Ilg, Tonmoy Saikia, Margret Keuper, and Thomas Brox. Occlusions, motion and depth boundaries with a generic network for disparity, optical flow or scene flow estimation. In *European Conf. Comput. Vis.*, pages 614–630, 2018.

[437] Kahlouche Souhila and Achour Karim. Optical flow based robot obstacle avoidance. *Int. J. Adv. Robot. Syst.*, 4(1):2, 2007.

[438] Gangik Cho, Jongyun Kim, and Hyondong Oh. Vision-based obstacle avoidance strategies for MAVs using optical flows in 3-D textured environments. *Sensors*, 19(11), 2019.

Acknowledgments

Over the course of my academic journey, there was a comic vignette by Reza Farazmand that played out like a little whisper of reassurance in the form of a mouse, saying, "*Everything is going according to plan.*" And what was this plan, you might ask? "*Hope it all works out.*" Little did I know, this seemingly light-hearted mantra would resonate with profound truth throughout my Ph.D. pursuit.



"The Plan". Picture credit: Reza Farazmand, author of Poorly Drawn Lines.

As I stand here at the conclusion of this challenging yet rewarding chapter of my life, I am profoundly aware that the successful outcome was not solely due to my personal dedication and optimism that everything would work out in the end. It is clear to me that a lot of people played vital roles in this achievement, and I owe them my most sincere gratitude. This journey, with all its accomplishments, simply would not have been possible without their unwavering support and guidance.

I want to start by extending a heartfelt thank you to Guido and Christophe, my supervisory team, for the invaluable opportunity they have given me. Back in 2015, you warmly welcomed me into the MAVLab, making my decision to relocate to The Netherlands and all that it entailed truly worthwhile. Through the trust you placed in me, I was fortunate to live countless unique experiences, from an internship at Georgia Tech to participating in (and winning, *yay!*) the AlphaPilot/AIRR competition. I am aware that I might not have been the easiest student to supervise, owing to my stubbornness and sometimes overly critical attitude, but I felt your confidence in me was always firm, and this allowed me to push even further. Thank you for your support both professionally and personally throughout this journey. It was an honor learning from and working with you.

To co-authors who later became friends, thank you for making my Ph.D. journey more fulfilling and enjoyable. Kirk, you were there from the very beginning of my time at

the MAVLab, first as a supervisor and then as a colleague. You have been particularly influential during this phase of my life, and I cannot thank you enough for your invaluable contributions to my work and for facilitating my transition into Sony. Yingfu and Nilay, thank you for being my family at work. Your constant support and dedication, positiveness, and willingness to share perspectives from diverse backgrounds have enriched my understanding and broadened my horizons in ways I could have never imagined. Simply put, thank you. Jesse, Stein, and Julien, thank you for the transformative influence you had on the second half of my Ph.D. Your commitment, creativity, engaging discussions, and openness to entertain my, at times, unstructured ideas have been instrumental, and for that, I am truly grateful. I am confident that there are bright futures awaiting each of you, and I eagerly anticipate being a part of them.

I would also like to extend my appreciation to all the members of the MAVLab and C&S who actively promote and maintain an inclusive and friendly work environment that fosters scientific advancements. Your dedication to creating such an environment has had a positive impact on my academic journey and the experiences of many others. In particular, thanks to Diana, Mario, Tom, Sven, Kimberly, Matěj, and many others who have been a part of this, whether they were here for a while or have moved on. Thank you for sharing the challenges and joys of our personal lives and academic endeavors.

During my time as a Ph.D. candidate, I was also fortunate to join Sony Europe for an internship. I would like to extend my gratitude to Oliver Erdler and Guido for making it possible, and to my colleagues Kirk, Diederik, Valentina, Vincent, and Manuel for making this experience both valuable and enjoyable. Having this break halfway through my Ph.D. allowed me to acquire valuable insights and skills that enriched my research journey and enabled me to better shape my professional career. For that, I am truly grateful.

It goes without saying that I hold deep appreciation for my friends outside university. I understand that at times, I can be somewhat distant, both physically and emotionally, but your support and understanding have meant the world to me. Thank you for always being there, whether through in-person interactions or via videogames. It is with you that I can truly find moments of disconnection.

Last but most definitely not least, quiero dar las gracias a mi familia por su apoyo incondicional a lo largo de mi vida, incluso en los momentos en los que mi ambición nos ha distanciado más de lo deseado. Papá y Mamá, gracias por vuestra generosidad. Vuestro esfuerzo y sacrificio constante durante todos estos años ha hecho que, tanto Jesús como yo, hayamos crecido con amor, respeto y educación, además de con acceso a innumerables oportunidades. Sois nuestro ejemplo a seguir y siempre os estaremos agradecidos. Jesús, creciste siendo el pequeño de dos hermanos, siguiendo muchos de mis pasos. Irónicamente, ahora eres tú el que actúa como referente para mí en muchos aspectos. Gracias. Me siento muy afortunado de ser tu hermano, y siempre voy a estar aquí para lo que necesites. Clara, es difícil poner en palabras lo importante que eres para mí. Simplemente, gracias por darle equilibrio a mi vida y por cuidarme. Te quiero y admiro profundamente. Espero poder compensar en los próximos años todos los sacrificios que has hecho por mí y por este doctorado, el cual estoy orgulloso de poder compartir hoy contigo. Por último, gracias de todo corazón a mis abuelos/as, tíos/as (incluyendo, por descontado, a los presidentes de Komory's Club) y primos/as. Todos habéis contribuido de una forma u otra, pero siempre positivamente, a que hoy pueda estar donde estoy. A todos, os quiero mucho.

As I close this chapter and step into new beginnings, I carry not only the knowledge and skills I've gained, but also the understanding that having a supportive community can truly transform one's journey. The mouse's simple but strangely profound mantra now reflects the gratitude I hold for each individual who played a part in making sure that, in the end, everything did work out.

Fede
Delft, November 2023

Curriculum Vitæ

Federico Paredes-Vallés

01/04/1993	Born in Murcia, Spain

Education

2018–2023	Ph.D. in Aerospace Engineering Delft University of Technology, Delft, The Netherlands *Thesis*: Self-supervised neuromorphic perception for autonomous flying robots *Promotors*: prof. dr. G. C. H. E. de Croon dr. ir. C. De Wagter
2015–2018	M.Sc. in Aerospace Engineering Delft University of Technology, Delft, The Netherlands *Thesis*: Neuromorphic computing of event-based data for high-speed vision-based navigation
2011–2015	B.Sc. in Aerospace Engineering Universidad Politécnica de Valencia, Valencia, Spain
1999–2011	Primary and secondary school Colegio San Buenaventura, Murcia, Spain

Experience

2023–Present	Research Engineer Advanced Sensing and Modelling Group, Sony Schlieren, Switzerland
2020–2021	Research Engineer Vision and Control Group, Sony Schlieren, Switzerland

2016–2017	Visiting Research Intern Unmanned Aerial Vehicle Research Facility Georgia Institute of Technology Atlanta, USA

Awards

2019	World Champion, AlphaPilot/AIRR Autonomous Drone Racing
2016	Justus and Louise van Effen Research Grant

List of Publications

† Equal contribution.
Included in this thesis.

Journal Papers

7. ***F. Paredes-Vallés***†, *J. J. Hagenaars*†, *J. D. Dupeyroux*†, *S. Stroobants, Y. Xu, G. C. H. E. de Croon*: Fully neuromorphic vision and control for autonomous drone flight. Under review at Science Robotics. 2023.

6. *C. De Wagter*†, ***F. Paredes-Vallés***†, *N. Sheth*†, *G. C. H. E. de Croon*: The sensing state-estimation and control behind the winning entry to the 2019 Artificial Intelligence Robotic Racing competition. Field Robotics. 2022.

5. *C. De Wagter*†, ***F. Paredes-Vallés***†, *N. Sheth*†, *G. C. H. E. de Croon*: Learning fast in autonomous drone racing. Nature Machine Intelligence (NMI). 2021.

4. *D. B. de Jong*, ***F. Paredes-Vallés***, *G. C. H. E. de Croon*: How do neural networks estimate optical flow? A neuropsychology-inspired study. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). 2021.

3. *J. J. Hagenaars*, ***F. Paredes-Vallés***, *G. C. H. E. de Croon*: Evolved neuromorphic control for high speed divergence-based landings of MAVs. IEEE Robotics and Automation Letters (RA-L). 2020.

2. ***F. Paredes-Vallés***, *K. Y. W. Scheper, G. C. H. E. de Croon*: Unsupervised learning of a hierarchical spiking neural network for optical flow estimation: From events to global motion perception. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI). 2019.

1. *S. García-Nieto, J. Velasco-Carrau,* ***F. Paredes-Vallés***, *J. V. Salcedo, R. Simarro*: Motion equations and attitude control in the vertical flight of a VTOL bi-rotor UAV. Electronics. 2019.

Conference Papers

8. *Y. Wu,* ***F. Paredes-Vallés***, *G. C. H. E. de Croon*: Lightweight event-based optical flow estimation via iterative deblurring. Under review at IEEE International Conference on Robotics and Automation (ICRA). 2024.

7. ***F. Paredes-Vallés***, *K. Y. W. Scheper, C. De Wagter, G. C. H. E. de Croon*: Taming contrast maximization for learning sequential, low-latency, event-based optical flow. IEEE International Conference on Computer Vision (ICCV). 2023.

6. *R. J. Bouwmeester,* ***F. Paredes-Vallés***, *G. C. H. E. de Croon*: NanoFlowNet: Real-time dense optical flow on a nano quadcopter. IEEE International Conference on Robotics and Automation (ICRA). 2023.

5. *J. J. Hagenaars*†, ***F. Paredes-Vallés***†, *G. C. H. E. de Croon*: Self-supervised learning of event-based optical flow with spiking neural networks. Advances in Neural Information Processing Systems (NeurIPS). 2021.

4. *J. D. Dupeyroux, J. J. Hagenaars, **F. Paredes-Vallés**, G. C. H. E. de Croon*: Neuromorphic control for optic-flow-based landing of MAVs using the Loihi processor. IEEE International Conference on Robotics and Automation (ICRA). 2021.

3. ***F. Paredes-Vallés**, G. C. H. E. de Croon*: Back to event basics: Self-supervised learning of image reconstruction for event cameras via photometric constancy. IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 2021.

2. *J. J. Hagenaars, **F. Paredes-Vallés**, G. C. H. E. de Croon*: Evolved neuromorphic control for high speed divergence-based landings of MAVs. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). 2020.

1. ***F. Paredes-Vallés**, D. P. Magree, E. N. Johnson*: Direct feature correspondence in vision-aided inertial navigation for unmanned aerial vehicles. International Conference on Unmanned Aircraft Systems (ICUAS). 2017.

Propositions

accompanying the dissertation

Self-Supervised Neuromorphic Perception for Autonomous Flying Robots

by

Federico Paredes-Vallés

1. Embracing simplicity and interpretability as core tenets in engineering nurtures effective solutions, mitigating complexity risks, and cultivating a profound system understanding. [This thesis]
2. Adopting self-supervised learning in robotics harnesses the power of real, unlabeled data, bypassing the need for sensor simulation and hence addressing the reality gap in perception systems. [This thesis]
3. Neuromorphic technology has the potential to catalyze intelligence democratization in robotics, revolutionizing small robot integration, empowering advanced capabilities, and expanding their applications. [This thesis]
4. Nature, as source of inspiration for robotics, guides a path towards practical solutions when embraced as a muse rather than a blueprint. [This thesis]
5. Event cameras, driven by the essence of change, unveil a profound truth in robotics: perception thrives on dynamism.
6. In an AI-dominated media landscape, education, logic, and critical thinking will remain as our last refuge, protecting truth and upholding integrity.
7. The journey to become an independent researcher transcends solo pursuits, as fruitful collaborations cultivate breakthrough innovation and amplify the impact of our work.
8. In a multi-cultural work environment, excelling in one's job and embracing cross-cultural understanding are two sides of the same coin.
9. In the realm of creativity, where pixels and ideas collide, videogames and computer vision research coexist as distinct yet interconnected domains, both inviting exploration, discovery, and the unleashing of untapped potentials.
10. Preserving personal time poses a formidable challenge for Ph.D. candidates, transcending research domains while being integral to academic endeavors.

These propositions are regarded as opposable and defendable, and have been approved as such by the promotors prof. dr. G. C. H. E. de Croon, and dr. ir. C. De Wagter.